GEAD
Grupo de Estudo da Educação a Distância

Centro Paula Souza

GOVERNO DO ESTADO
SÃO PAULO

CURSO TÉCNICO EM
DESENVOLVIMENTO DE SISTEMAS
MODALIDADE EAD

LOGICA DE PROGRAMAÇÃO EM JAVA

Eliana Cristina Nogueira Barion
Marcelo Fernando Iguchi
Paulo Henrique Mendes Carvalho
Rute Akie Utida

Sumário

AGENDA 8
RACIOCINANDO A LÓGICA

MOMENTO DE REFLEXÃO

Você já parou para pensar que tudo o que fazemos no nosso dia a dia é resultado de uma sequência ordenada de passos? Pensou também que por mais simples que seja a tarefa que nos propusermos a fazer, precisamos ordenar nossos pensamentos para que possamos chegar ao resultado desejado?

A lógica sempre nos acompanha! Quando falamos, escrevemos ou fazemos alguma ação estamos pensando de forma ordenada e sequenciada para que as coisas aconteçam de forma correta. Logo a lógica consiste em colocar "ordem no nosso pensamento".

Imagem 01

Figura 1 - Arquivo GEEaD

Observe a figura acima e pense em trocar a ordem de algum dos passos ilustrados. Por exemplo:

1 – Tomar conteúdo
2 – Abrir a tampa
3 – Pegar o recipiente
Será que é possível beber o refrigerante da lata?

Claro que não! É preciso seguir a ordem dos passos ilustrados acima para que conseguimos atingir o objetivo final, que neste caso, é tomar o conteúdo.

A mesma coisa acontece com o computador. Quando pensamos em fazer um programa no computador, temos que ter em mente que a máquina desconhece totalmente alguns conceitos que para nós são muito óbvios. Por isto, devemos descrever cada passo, por mais simples que seja, para que haja uma sequência lógica em nossa programação e assim, o computador executar todas as instruções necessárias para uma determinada tarefa. Este é o objetivo desta agenda!
Entender a necessidade de descrever com exatidão cada uma das tarefas a serem executadas pela máquina. Então, vamos começar?

A lógica para programação é o processo de saber pensar na mesma sequência em que o computador executa as tarefas.
Aprende-se a imaginar como as ações serão executadas partindo-se do estudo de um problema até chegar à solução dele por meio da construção de um algoritmo. Por isso, a lógica de programação é tão importante! Sendo assim, esta agenda traz conceitos para você compreender a lógica de programação, conhecer e desenvolver algoritmos e fluxogramas, interpretar algoritmos, pseudocódigo e outras especificações para codificar programas.

PARA COMEÇAR O ASSUNTO...

Imagem 02

Quando pensamos em desenvolver um programa, devemos ter domínio sobre ele, analisando cada etapa do problema, para que possamos fornecer ao computador uma sequência lógica de passos que a máquina deverá executar para que resolva o nosso problema. Esta sequência lógica é representada pelo algoritmo. Só depois da análise de cada etapa do problema é que se deve começar a programar o computador!

Para executar qualquer tarefa, devemos seguir as etapas numa sequência lógica. Com a programação de um software não é diferente, pois ela é muito parecida com qualquer outra atividade corriqueira do nosso dia a dia.

Um exemplo disso é o trajeto que fazemos de casa à escola. É uma sequência de procedimentos que devemos cumprir para chegar ao nosso destino sem complicações. Pense na sequência que você deve seguir para ir de casa à escola de ônibus.

Agora confira com a solução abaixo:

Trajeto_Casa_Escola

1. Andar até o ponto de ônibus;
2. Aguardar o ônibus;
3. Ao avistar o ônibus correto, fazer sinal;
4. Entrar no ônibus pela porta traseira;
5. Pagar passagem;
6. Escolher um assento e sentar;
7. Quando chegar próximo do local a saltar, dar o sinal para descida;
8. No ponto, descer do ônibus, pela porta dianteira;
9. Andar até à escola.

Pronto, você acabou de escrever um algoritmo e já está pronto para mergulhar no tema desta agenda!

MERGULHANDO NO TEMA...

Voltando ao exemplo citado em “Momento de Reflexão”, podemos dizer que já temos o algoritmo (sequência de ações):

Imagem 03

Figura 2- Arquivo GEEaD

Porém, neste algoritmo podemos detalhar um pouco mais colocando outras etapas intermediárias.

Lembre-se! Quanto mais detalhado for a instrução para o computador, mais rápido e fácil ele compreenderá e executará atingindo o objetivo final.

Imagem 04

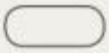
Figura 3- Arquivo GEEaD

Observe que o algoritmo acima está bem mais detalhado e mesmo uma pessoa que não saiba como tomar um líquido de uma lata, seguindo estas instruções, irá conseguir sem dificuldades tomar o líquido.

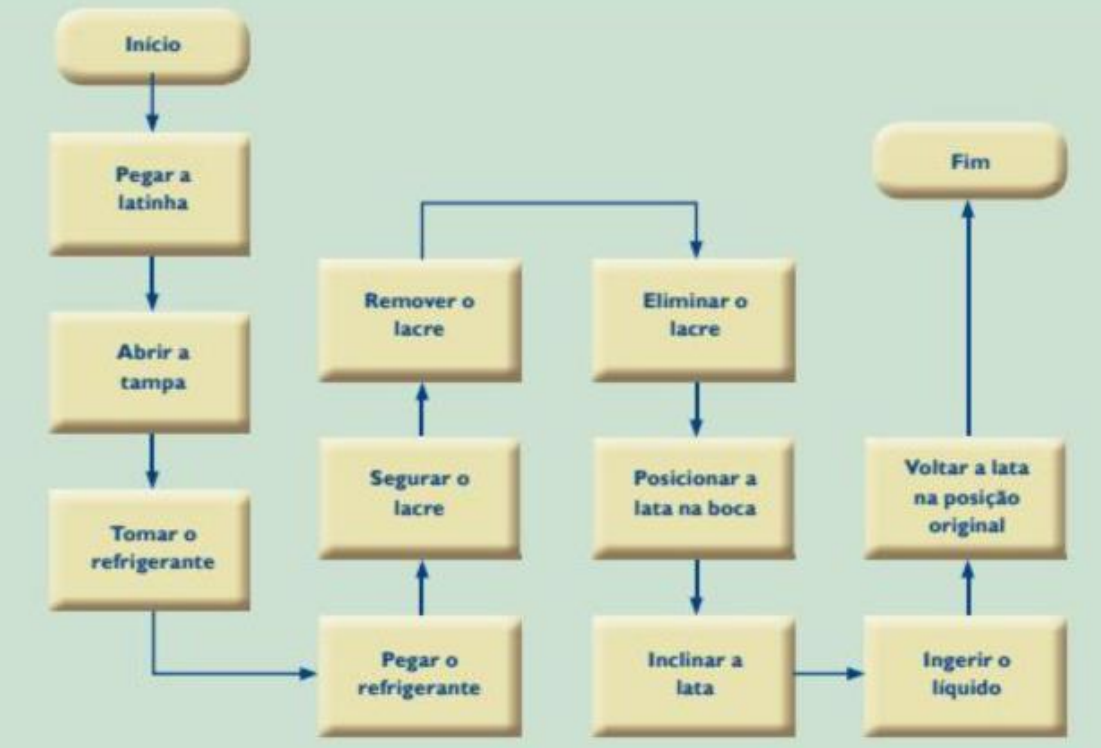
Este é o primeiro passo para resolver um problema. Para que todos possam compreender o seu algoritmo, é necessário utilizar o **Fluxograma**.

Mas o que é o Fluxograma? É a representação gráfica da sua sequência lógica (seu algoritmo).

Podemos usar qualquer diagrama ou qualquer desenho? Não. Existem as formas corretas com seus respectivos significados a serem utilizados como veremos a seguir na tabela:

SIMBOLOGIA DO FLUXOGRAMA	
SÍMBOLO	NOME E FUNÇÃO
	NOME = TERMINAL FUNÇÃO = Indica INÍCIO ou FIM de um processamento
	NOME = PROCESSAMENTO FUNÇÃO = Definição de variáveis ou processamentos em geral (Cálculos)
	NOME = ENTRADA MANUAL FUNÇÃO = Entrada de dados via teclado, idêntico ao comando LEIA
	NOME = DISPLAY FUNÇÃO = Saída de Dados, mostra um texto e/ou variável na tela, idêntico ao comando ESCREVA
	NOME = DOCUMENTO FUNÇÃO = Saída de Dados, envia um texto e/ou variável para a impressora, usado em relatórios. Idêntico ao comando IMPRIMA
	NOME = DECISÃO FUNÇÃO = Decisão a ser tomada, retornando verdadeiro ou falso, idêntico ao comando SE
	NOME = CONECTOR FUNÇÃO = Desvia o fluxo para uma outra página, sendo interligados pelo conector
	NOME = ENTRADA/SAÍDA FUNÇÃO = Leitura de gravação de arquivos
	NOME = SETA FUNÇÃO = Indica a direção do fluxo
	NOME = LOOP FUNÇÃO = Realiza o controle de loop

Seguindo o mesmo exemplo, vamos montar o Fluxograma:



Para montar um Fluxograma, você poderá utilizar as seguintes Ferramentas como apoio: Microsoft Visio ou Microsoft Word.

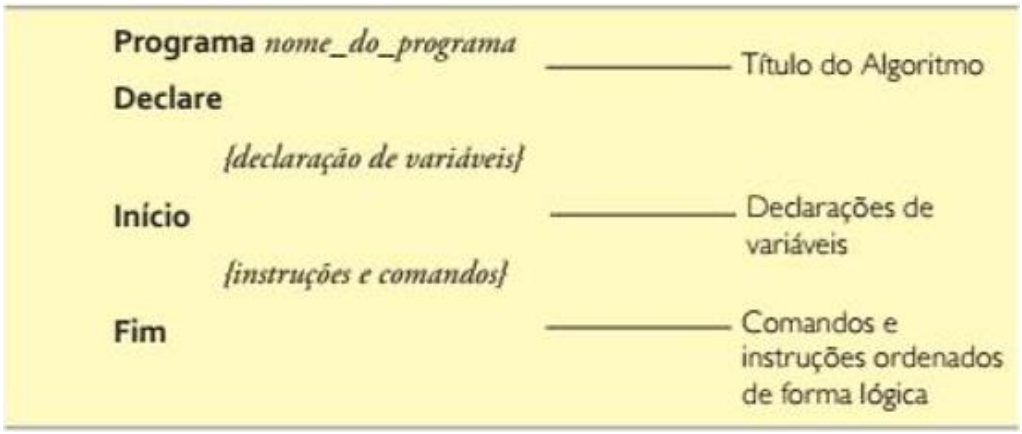
Pronto! Você já sabe o que é um Algoritmo e como usá-lo como também o que é um Fluxograma e como deve ser utilizado. Porém não acabou. Além destas duas etapas, temos mais duas para que ele seja executado em um computador como programa.

Para que o seu problema torne um programa, existem, pelo menos, quatro etapas a serem seguidas. São elas:

Seguindo o ciclo acima, falta informar sobre Pseudocódigo, Linguagem de Programação e Programa. Nesta agenda iremos abordar até Pseudocódigo e ao longo das outras serão abordados sobre Linguagem de Programação e Programa.

Mas o que é **Pseudocódigo**? É uma escrita mais próxima da Linguagem de Programação, ou seja, não usaremos nenhuma informação técnica da Linguagem e apenas utilizaremos o nosso idioma (português) escrevendo mais próximo das instruções computacionais. Muitos autores chamam o Pseudocódigo de "Portugol" ou "Português Estruturado" devido a estas características.

A seguir um pequeno modelo, como orientação, para escrever um Pseudocódigo:



Utilizando os conceitos apresentados...

1. Crie um algoritmo para fritarmos um ovo. Faça este algoritmo com uma sequência de no mínimo 15 passos.
2. Elabore o fluxograma do algoritmo do exercício anterior.
3. Fluxograma para solucionar o problema da Torre de Hanói

Confira abaixo se você conseguiu resolver os desafios propostos!

Respostas:

1. Algoritmo para fritar um ovo
   1 - Pegue um ovo;
   2 - Pegue uma frigideira;
   3 - Pegue o óleo;
   4 - Pegue o saleiro;
   5 - Coloque o óleo na frigideira;
   6 - Acenda o fogo;
   7 - Aqueça a frigideira com óleo;
   8 - Quebre o ovo;
   9 - Coloque o ovo na frigideira;
   10 - Adicione sal a gosto;
   11 - Espere o ovo fritar;
   12 - Retire o ovo da frigideira;
   13 - Coloque o ovo no prato;
   14 - Sirva o ovo;
   15 - Lave a frigideira.

2. Fluxograma para fritar um ovo

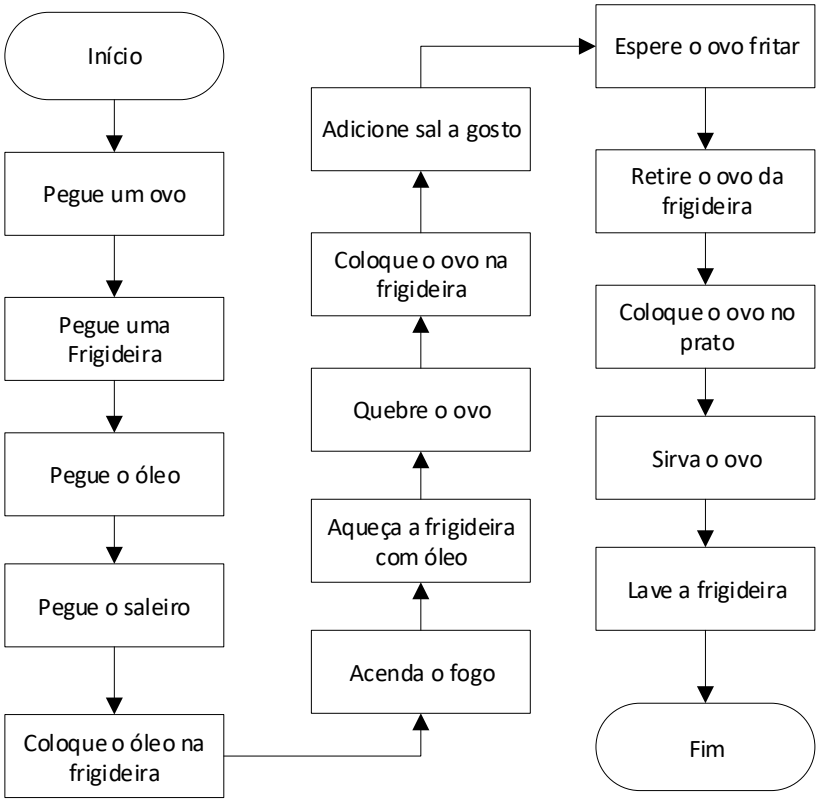
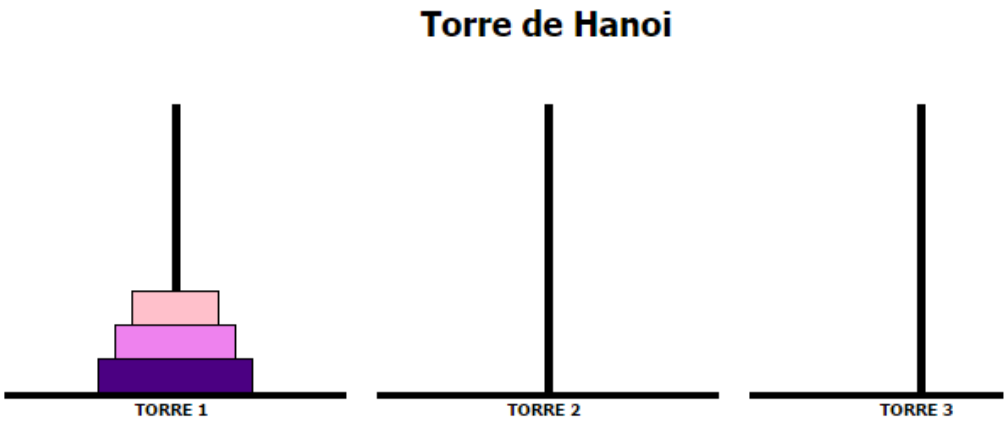


Figura 4- Arquivo GEEaD

Você conhece o Jogo da Torre de Hanói?

Imagem 05



O objetivo desse jogo é mover todos os discos para a estaca da direita, seguindo as seguintes **regras**:

Mover um disco de cada vez, sendo que um disco maior nunca pode ficar em cima de um disco menor.

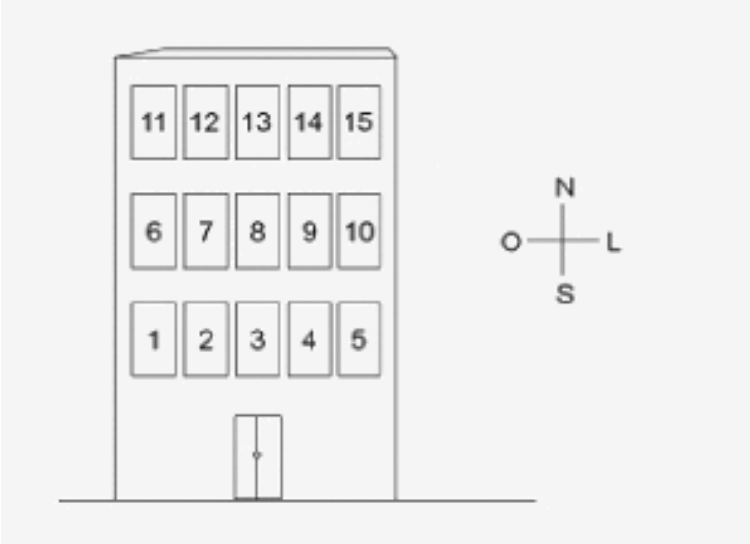
- De forma virtual através do link: http://www.somatematica.com.br/jogos/hanoi/ ou de forma física, tente solucionar a situação problema da torre de Hanói e descrever as ações passo a passo.

Estes exercícios devem ser entregues de forma on-line como atividades da agenda.

Questionários online

1. (FCC - 2004 - Analista Judiciário - TRT) A figura mostra a localização dos apartamentos de um edifício de três pavimentos que tem apenas alguns deles ocupados: Sabe-se que:
- Maria não tem vizinhos no seu andar, e seu apartamento localiza-se o mais a leste possível;
- Taís mora no mesmo andar de Renato, e dois apartamentos a separam do dele;
- Renato mora em um apartamento no segundo andar exatamente abaixo do de Maria;
- Paulo e Guilherme moram no andar mais baixo, não são vizinhos e não moram abaixo de um apartamento ocupado.

- No segundo andar estão ocupados apenas dois apartamentos.



Se Guilherme mora a sudoeste de Tais, o apartamento de Paulo pode ser:

a) 1 ou 3
b) 1 ou 4
c) 3 ou 4
d) 3 ou 5
e) 4 ou 5

Vídeo explicativo: https://www.youtube.com/embed/K0uaSwYki8s?rel=0

2. Escreva o número que falta na sequencia a seguir.

18 20 24 32 _____

3. Há três suspeitos de um crime: o cozinheiro, a governanta e o mordomo. Sabe-se que o crime foi efetivamente cometido por um ou por mais de um deles, já que podem ter agido individualmente ou não. Sabe-se, ainda que:

- se o cozinheiro é inocente, então a governanta é culpada;

- ou o mordomo é culpado ou a governanta é culpada, mas não os dois;

- o mordomo não é inocente.

Logo:

(A) a governanta e o mordomo são os culpados
(B) o cozinheiro e o mordomo são os culpados
(C) somente a governanta é culpada
(D) somente o cozinheiro é inocente
(E) somente o mordomo é culpado.

4. Qual o número que completa a sequência: 1, 3, 6, 10, ...
(A) 11
(B) 12
(C) 13
(D) 15
(E) 18

5. Um frasco contém um casal de melgas. As melgas reproduzem-se e o seu número dobra todos os dias. Em 50 dias o frasco está cheio. Em que dia o frasco esteve meio cheio?
(A) 02
(B) 24

(C) 25
(D) 26
(E) 49

6. Qual o número que completa a sequência: 1, 1, 2, 3, 5, ...
(A) 5
(B) 6
(C) 7
(D) 8
(E) 9

7. Num concurso de saltos, Maria foi, simultaneamente, a 13ª melhor e 13ª pior. Quantas pessoas estavam em competição?
(A) 13
(B) 25
(C) 26
(D) 27
(E) 28

8. Bruno é mais alto que Joaquim. Renato é mais baixo que o Bruno. Então, Joaquim é o mais alto dos três.

( ) Verdadeiro

( ) Falso

9. O preço de um produto foi reduzido em 20% numa liquidação. Qual deverá ser a percentagem de aumento do preço do mesmo produto para que ele volte a ter o preço original?

(A) 15%
(B) 20%
(C) 25%
(D) 30%
(E) 40%

10.

Imagem 06

Adaptado de "Alice no País das Maravilhas", de Lewis Carroll.

Chapeuzinho Vermelho ao entrar na floresta, perdeu a noção dos dias da semana. A Raposa e o Lobo Mau eram duas estranhas criaturas que frequentavam a floresta. A Raposa mentia às segundas, terças e quartas-feiras, e falava a verdade nos outros dias da semana. O Lobo Mau mentia às quintas, sextas e sábados, mas falava a verdade nos outros dias da semana.

Um dia Chapeuzinho Vermelho encontrou a Raposa e o Lobo Mau descansando à sombra de uma árvore. Eles disseram:

Raposa: "Ontem foi um dos meus dias de mentir"

Lobo Mau: "Ontem foi um dos meus dias de mentir"

A partir dessas afirmações, Chapeuzinho Vermelho descobriu qual era o dia da semana. Qual era?

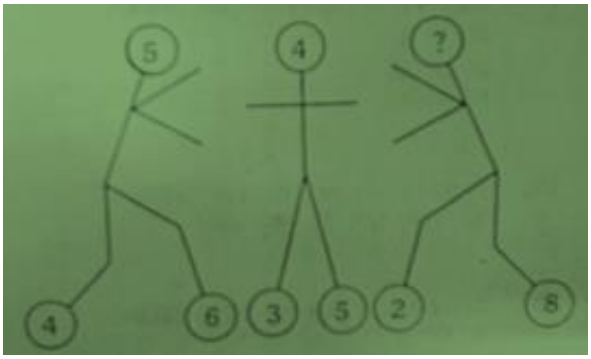
(A) Segunda-feira
(B) Terça-feira
(C) Quarta-feira
(D) Quinta-feira
(E) Sexta-feira

11. (ESAF) José quer ir ao cinema assistir ao filme "Fogo Contra Fogo", mas não tem certeza se o mesmo está sendo exibido. Seus amigos, Maria, Luís e Júlio têm opiniões discordantes sobre se o filme está ou não em cartaz. Se Maria estiver certa, então Júlio está enganado. Se Júlio estiver enganado, então Luís está enganado. Se Luís estiver enganado, então o filme não está sendo exibido. Ora, ou o filme "Fogo conta Fogo" está sendo exibido, ou José não irá ao cinema. Verificou-se que Maria está certa. Logo,....

O filme "Fogo Contra Fogo" está sendo exibido

(A) Luís e Júlio não estão enganados
(B) Júlio está enganado, mas Luís não.
(C) Luís está enganado, mas Júlio não.
(D) José não irá ao cinema.

12. Analise a imagem a seguir:



Qual o número que corresponde à sequência lógica dessa imagem?

(A) 1
(B) 3
(C) 4
(D) 5
(E) 6

AMPLIANDO HORIZONTES

Não deixe também de assistir aos vídeos:

Informática - Módulo I - Agenda 10 - Lógica de Programação: Conceitos Básicos

Link: https://www.youtube.com/watch?v=vAIBmbCcmD4. Acessado em 21/12/2017.

Informática - Módulo I - Agenda 9

Link: https://www.youtube.com/watch?v=PhEs1vRDNLE. Acessado em 21/12/2017.

Para aprofundamento dos temas discutidos nesta aula, seguem abaixo algumas dicas de filmes, livros e artigos que se relacionam com o conteúdo estudado. Estas dicas são muito importantes para você!

Livros:

FORBELLONE, André L. V.; ELBERSPACHER, Henri Frederico. Lógica de Programação: A Construção de Algoritmos e Estrutura de Dados. Editora Pearson, 2000.

MANZANO, José Augusto N. G; OLIVEIRA, Jayr Figueiredo. Algoritmos: Lógica para Desenvolvimento de Programação. Editora Érica, 2007.

PUGA, SANDRA; RISSETTI, GERSON. Lógica de Programação e Estruturas de Dados com Aplicações em Java. Editora Pearson. 2009

SCHILDT, HEBERT. Java para Iniciantes. Editora Bookman.2015

Artigos:

MORAES, Paulo Sérgio de. ;"Lógica de Programação", 2000, disponível em http://www.siban.com.br/destaque/21_carta.pdf. Acessado em 29/10/2017.

AGENDA 9
DESENVOLVENDO A LÓGICA

MOMENTO DE REFLEXÃO

Imagem 07

Você já passou por algum problema gerado por uma deficiência de comunicação em situações do cotidiano? Provavelmente, sim! Quando não há uma comunicação clara, objetiva e padronizada nas ações do cotidiano, a chance de que as informações sejam repassadas de forma errada é muito grande. Se na vida real as pessoas precisam se comunicar de forma clara, a fim de se entenderem, imagine quando essa comunicação deve ser estendida a um computador. Da mesma forma, se pensarmos que na comunicação entre humanos é necessário estabelecer padrões, o que diremos da comunicação humano para computador? Assim, o principal objetivo deste capítulo é apresentar algumas técnicas que possam ajudá-lo no desenvolvimento da lógica de programação e promover a "ordenação do pensamento", de forma que você possa enxergar essa lógica. Boa sorte!

POR QUE APRENDER?

O desenvolvimento de programas e algoritmos depende totalmente do completo entendimento de sua estrutura. Saber os comandos que introduzem informações nos sistemas, bem como os comandos que exibem informações processadas é de vital importância para alguém que queira ingressar ou evoluir no conceito da programação de computadores. Nesta aula você verá como tudo isto se relaciona e qual é o procedimento que os programadores devem adotar para alimentar um programa e extrair dele as informações requeridas.

PARA COMEÇAR O ASSUNTO...

Imagem 08

Algoritmo, pseudocódigo, fluxograma, dados, variáveis, ufa! Quantos termos! Mas o que na verdade querem dizer todas estas palavras um tanto quanto "desconhecidas"? Antes de mais nada, precisamos entender que cada uma delas ocupa um espaço bem definido e importante dentro do conceito da lógica computacional. Então, é preciso entender direitinho o que cada uma significa. Já vimos no capítulo anterior que algoritmo é como se fosse uma receitinha de bolo. Para termos o bolo pronto, precisamos seguir alguns passos. A maneira como estes passos são descritos é que indica se estamos usando um

"pseudocódigo" (linguagem parecida com a linguagem humana) ou um "fluxograma" (diagrama com formas geométricas).

E as variáveis? Bem, podemos entender uma variável como uma gavetinha no armário onde guardamos algumas peças de roupas (dados). Para não misturar e bagunçar o armário, podemos etiquetar as gavetas indicando o nome das peças de roupas a que elas foram destinadas. Da mesma forma, a variável precisa ser identificada (nome) e possuir um único tipo de dado (peças de roupa). Fácil né? Para entender melhor estes e outros conceitos, que tal mergulhar fundo no tema?

Variável

Durante os seus estudos, você notou que até o momento trabalhamos apenas com números e palavras fixas? Ou seja, não pudemos alterar o seu conteúdo. Essa é a definição de uma **Constante**, isto é um local na memória do computador que armazena um dado que não se altera ao longo da execução do programa.

Se um computador trabalha em constante interação com o usuário, por que até este momento não trabalhamos com esta interação?

A resposta é simples! Antes de iniciarmos a interação homem-máquina efetivamente, precisamos compreender os aspectos básicos de qualquer Linguagem de Programação, sendo assim, agora que você já sabe estes aspectos estamos prontos para trabalharmos um novo conceito: **Variável**.

O computador é um objeto que não possui a capacidade de pensar por conta própria, sendo necessário sempre uma programação para ensina-lo a trabalhar.

Quando iniciamos uma interação homem-máquina, não há como o computador saber o que o usuário fará na sequência de seus atos, por isto, devemos "ensiná-lo" a se preparar para uma interação com o usuário do computador.

Esta interação é feita através de uma estrutura chamada **variável.** Ela consiste na alocação de um pedacinho da memória RAM do computador para que ele receba uma informação vinda de um dispositivo de entrada de dados, no nosso caso, o teclado.

Para entendermos este conceito, imagine a seguinte situação:

Em sua primeira aula de Lógica de programação, você conheceu um colega de turma chamado Juvenal. Para que em um futuro próximo vocês possam trocar informações sobre as atividades propostas, você decidiu pedir ao Juvenal seu **Número de telefone.**

No momento em que você pede ao Juvenal o seu número de telefone, você inconscientemente prepara um lugar para armazena-lo, podendo ser em seu cérebro, celular ou em um papel não é mesmo?

Você sabia qual seria o tipo de dado (Você saberia quais seriam os números) que o Juvenal lhe passaria? Provavelmente não, mas saberia com certeza que ele lhe passaria um número de telefone.

Resumindo: Se você fosse um computador, estaria utilizando uma variável (espaço na memória) do tipo numérica para receber o dado que seria passado pelo Juvenal (agente externo ao programa). Fácil, não?

Logo, se você estivesse criando um programa para que o Juvenal e outros amigos inserissem seus números de telefone para que você os consulte depois, precisaria ensinar o computador que ele deve reservar um pequeno espaço em sua memória RAM para receber estes valores.

Será que para o computador é tão simples somar 1+1 como para nós, seres humanos?

Imagem 09

Vamos analisar ...

Primeiramente é necessário armazenar o primeiro número fornecido pelo usuário na memória do computador.

- Ler e armazenar o primeiro número seria a primeira tarefa.

Por que armazenar?

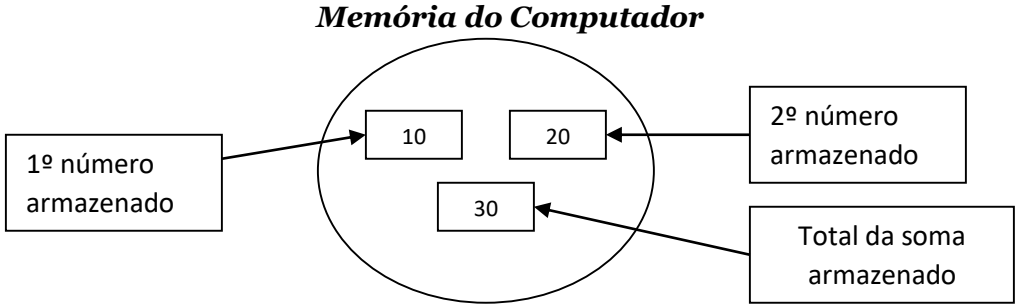
Porque ele somará o primeiro número com o próximo número fornecido! Enquanto isto, ele deve memorizar o primeiro número, senão, quando fornecermos o próximo, ele já esqueceu o primeiro!

- Ler e armazenar o segundo número pelo mesmo motivo que o primeiro!

- Executar a operação de soma

- Memorizar o resultado da soma para mostrar em sua tela!

Quando falamos em memorizar, estamos dizendo que o dado deve ser colocado na memória do computador. Inserimos valores na memória de um computador através de **VARIÁVEIS**.



Portanto, temos que dar nome aos lugares onde estes valores estão armazenados e indicar qual tipo de dado eles podem receber, ou seja de qual tipo de dado poderá ser seu conteúdo. Fazemos isto através de Declaração de Variável. Veja:

Para a caixinha onde está armazenado o 1º número (10), poderei escolher um nome qualquer desde que:

- ✓ não seja número ou que comece com um número;
- ✓ não tenha acento;
- ✓ não tenha espaços em branco!

Estas são algumas regras para declaração de nome de variável!

Operadores

Quando utilizamos a lógica, sendo ela em pseudocódigo ou em uma Linguagem de Programação, devemos utilizar alguns símbolos, ou palavras, para que o computador entenda o que queremos que ele faça, estes símbolos são chamados de Operadores.

Existem muitos tipos de operadores, mas neste momento estudaremos os operadores Aritméticos, Relacionais e Lógicos.

Operadores Aritméticos.

Se, por acaso, for necessário desenvolver uma operação matemática utilizando pseudocódigo ou uma linguagem de programação, você precisará dos operadores aritméticos para criar o seu programa. Apesar do computador utilizar a mesma regra da matemática para a resolução de cálculos, a simbologia utilizada nem sempre é a mesma da matemática. A seguir você conhecerá os operadores aritméticos, sua simbologia em pseudocódigo, em Java a sua utilização e exemplo:

Nome da operação	Sintaxe em pseudocódigo	Sintaxe em Java	Função	Exemplo
Soma	+	+	Efetua a soma entre 2 valores numéricos.	3 + 2 = 5
Subtração	-	-	Efetua a subtração entre 2 valores numéricos.	3 – 2 = 1
Multiplicação	*	*	Efetua a multiplicação entre 2 valores numéricos.	2 * 4 = 8
Divisão	/	/	Efetua a divisão entre 2 valores numéricos.	4 / 2 = 2
Potenciação	exp(b,e) b = base e e = expoente)	^	Efetua a potenciação entre 2 valores	Exp(3,2) = 9 (Pseudocódigo) 3 ^ 2 = 9 (Java)
Resto da Divisão	mod	%	Efetua a divisão entre 2 valores, mas retorna o valor do resto da divisão.	3 mod 2 = 1 (Pseudocódigo) 3 % 2 = 1 (Java)
Concatenação	+	+	Junta 2 valores do tipo caractere.	“Lógica” + “Programação” = “LógicaProgramação”

Operadores Relacionais

Estes operadores são os responsáveis por efetuar comparações entre dados, com o objetivo de mostrar ao programa como proceder dependendo da situação apresentada. O programa de computador verá o resultado de uma comparação em duas situações lógicas, sendo **verdadeiro** ou **falso**. Assim como os operadores Aritméticos, segue uma pequena tabela indicando suas finalidades:

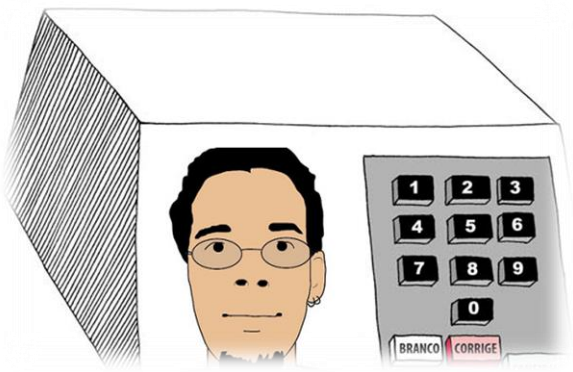
Nome do Operador Relacional	Sintaxe em pseudocódigo	Sintaxe em Java	Função	Exemplo	Resultado
Maior	>	>	Compara se o primeiro valor é	7 > 3	Verdadeiro

			maior que o segundo valor.		
Menor	<	<	Compara se o primeiro valor é menor que o segundo valor.	(3+1) < (5*0)	Falso
Maior ou Igual	>=	>=	Compara se o primeiro valor é maior ou igual ao segundo valor.	(4 + 4) >= 8	Verdadeiro
Menor ou Igual	<=	<=	Compara se o primeiro valor é menor ou igual ao segundo valor.	4 <= 4	Verdadeiro
Igual	=	==	Compara se o primeiro valor é igual ao segundo valor.	3 = 2 (Pseudocódigo) 3 == 2 (Java)	Falso
Diferente	<>	!=	Compara se o primeiro valor é diferente do segundo valor.	7 <> 3 (Pseudocódigo) 7 != 3 (Java)	Verdadeiro

Operadores Lógicos

Os operadores lógicos são responsáveis pela elaboração de comparações especiais, possibilitando que uma única expressão de comparação receba mais de um operador relacional. Ele geralmente é utilizado em situações complexas, por exemplo:

Imagem 10



Jovanir, um jovem de 20 anos, gostaria de saber se na próxima eleição ele será obrigado a votar ou se poderá votar de modo facultativo.
Para ser obrigado a votar, o eleitor deve ter a idade maior ou igual a dezoito anos e também o eleitor deve ser menor que 70 anos.

Neste caso, para que a expressão seja corretamente resolvida, necessitamos de um operador Lógico, que serão apresentados a seguir:

Nome do Operador Lógico	Sintaxe em pseudocódigo	Sintaxe em Java	Função	Exemplo	Resultado
E	E	&&	Para que o resultado da comparação seja verdadeiro, os dois lados da expressão devem ser verdadeiros.	(16 >= 16) E (16 < 18) (Pseudocódigo) (16 >= 16) && (16 < 18) (Java)	Verdadeiro
OU	OU	||	Para que o resultado da comparação seja verdadeiro, apenas um dos lados da expressão deve ser verdadeiro.	((3 + 2) < 5) OU ((3*2)=6) (Pseudocódigo) ((3+4) < 5) || ((3*2)==6) (Java)	Verdadeiro
NÃO	NAO	!	Inverte o resultado da expressão, ou seja, caso a expressão seja verdadeira, se tornará falso e vice-versa.	NAO(4 < 8) (pseudocódigo) ! (4 < 8) (Java)	Falso

Observando a tabela dos Operadores Lógicos, podemos concluir que a expressão que resolveria o problema do Jovanir seria: ((20 >=18) && (20 < 70)).
Como a idade dele é maior ou igual a 18 e também é menor que 70, Jovanir é obrigado a votar.

A tabela Verdade

Uma forma muito simples de lembrar qual é o operador correto para satisfazer uma expressão é utilizando a **Tabela Verdade.** Com ela podemos prever e entender melhor o funcionamento dos Operadores Lógicos.
A tabela consiste em separamos a comparação em dois blocos, sendo o primeiro antes do Operador Lógico, e o segundo logo após o operador. Para simularmos os resultados, definimos os resultados como verdadeiro (V) ou falso (F), para facilitar a simulação e não perdemos tempo com a resolução das expressões.
Na tabela verificamos todas as possibilidades, sendo ambas as comparações verdadeiras, ambas as comparações falsas ou apenas uma das comparações verdadeira, assim testamos as possibilidades que o operador pode proporcionar, como as tabelas a seguir nos mostram:

Operador E		
Entrada 1	**Entrada 2**	**Saída**
V	V	V
V	F	F
F	V	F
F	F	F

Operador OU		
Entrada 1	**Entrada 2**	**Saída**
V	V	V
V	F	V
F	V	V
F	F	F

Para o operador NÃO, o resultado da expressão é apenas invertido.

Aplicando a tabela Verdade em Pseudocódigo e Java temos:

Operador NAO (!)		
Expressão em Pseudocódigo	**Expressão em Java**	**Resultado**
NAO V	! V	FALSO
NAO F	! F	VERDADEIRO

Note que assim como na definição do operador E, o resultado da expressão foi verdadeiro, apenas quando ambos os lados da expressão são verdadeiros.

Operador NÃO	
Entrada	**Saída**
V	F
F	V

Operador E (&&)		
Expressão em Pseudocódigo	**Expressão em Java**	**Resultado**
V E V	V && V	V
V E F	V && F	F
F E V	F && V	F
F E F	F && F	FALSO

Operador OU (||)		
Expressão em Pseudocódigo	Expressão em Java	Resultado
V OU V	V || V	V
V OU F	V || F	V
F E V	F || V	V
F OU F	F || F	F

Desta vez, apenas quando ambas as expressões são falsas o resultado foi falso.

VOCÊ NO COMANDO

Aplicação da lógica na resolução de problemas.

Imagem 11

Manoel é um comerciante de ferragens para construção onde ele vende pregos, parafusos, porcas e braçadeiras por unidade. Para que Manoel saiba que obteve lucro em sua venda ele deve ter vendido pelo menos 100 pregos e então 70 parafusos, mas se ele vender 30 braçadeiras e 10 porcas também ficará no lucro.
Certo dia, Manoel vendeu 300 pregos, 180 parafusos, 10 porcas e 32 braçadeiras, sendo assim ele obteve _____

Confira abaixo se você conseguiu resolver os desafios propostos!

Sabemos que a meta é 100 pregos e 70 parafusos, mas a meta também pode ser considerada caso ele venda pelo menos 10 porcas e 30 braçadeiras, neste caso podemos montar a seguinte expressão:

((?? >= 100)	E	(?? >= 70))	OU	((?? >= 10)	E	(?? >= 30))
PREGOS		PARAFUSOS		PORCAS		BRAÇADEIRAS

Agora que indicamos a expressão, basta substituirmos as interrogações pelos valores que foram vendidos pelo Manuel, sendo assim:

((300 >=100) E (180 >= 100) OU ((10 >= 10) E (32 >= 30))

Resolvendo cada lado da expressão principal:

(VERDADEIROE VERDADEIRO) OU (FALSO E VERDADEIRO)

VERDADEIRO OU FALSO

VERDADEIRO

Logo, no dia em questão, Manuel ficou com LUCRO (Verdadeiro)

Observe as seguintes expressões lógicas e indique se o resultado é Verdadeiro ou Falso.

a)

Linguagem	Expressão	Resultado
Pseudocódigo	((33*1) > 27) OU (NAO (18<20))	VERDADEIRO*
Java	((33*1) > 27) || (!(18 <20))	

*Passo a passo para a resolução do exercício:

Passo 1: Separamos as expressões pelo Operador OU, sendo assim, temos as expressões

((33 * 1) > 27) **OU** **(NÃO (18 < 20))**

Passo 2: Resolvemos as expressões individualmente:

Expressão 1:	Expressão 2:
((33*1) >27)	NAO (18<20)
33*1 = 33	NAO (18 é MENOR que 20?)
33 é maior que 27?	NAO (VERDADEIRO)
VERDADEIRO	FALSO

Logo,

VERDADEIRO OU **FALSO** ,

segundo a tabela verdade é:

VERDADEIRO

Os operadores lógicos são lidos e resolvidos pela ordem em que forem lidos, salvo no caso do operador estar dentro de parênteses, neste caso ele terá prioridade como o caso do operador NAO na resolução acima.

b)

Linguagem	Expressão	Resultado
Pseudocódigo	NAO(3>4)	
Java	!(3>4)	

c)

Linguagem	Expressão	Resultado
Pseudocódigo	(5>(3+1)) OU (8 > 7)	
Java	(5>(3+1)) || (8 > 7)	

d)

Linguagem	Expressão	Resultado
Pseudocódigo	(((exp(3,2) +1) = 10) E (NAO (10 = 10))	
Java	(((3^2)+1) == 10) && (!(10==10))	

e)

Linguagem	Expressão	Resultado
Pseudocódigo	(NAO((8*2) = 16)) E ((15 mod 4) >=2)	
Java	(! (8*2) == 16)) && ((15 % 2) >= 2)	

f)

Linguagem	Expressão	Resultado
Pseudocódigo	((3 mod 2) = 1) E ((13mod3)= 1) OU ((3*4)<16)	
Java	((3 % 2) = 1) && ((13 % 3) == 1) || ((3*4) < 16)	

g)

Linguagem	Expressão	Resultado
Pseudocódigo	((18 > 13) E (exp(2, 5)+1 = 33)) E (((24 mod 8) > 0) E (72 < 80))	
Java	((18 > 13) && (2 ^ 5)+1 = 33)) && (((24 % 8) > 0) && (72 < 80))	

h)

Linguagem	Expressão	Resultado
Pseudocódigo	NAO ((15*2)>(exp(3,2)*3) OU ((2*5)+(2.5*2)>=12))	
Java	! ((15*2)>(3^2*3)) || ((2*5)+(2.5 * 2) >=12))	

Confira a resolução dos desafios:

b)

Linguagem	Expressão	Resultado
Pseudocódigo	NAO(3>4)	Verdadeiro
Java	!(3>4)	

NAO(3>4)
NAO(FALSO)
VERDADEIRO

c)

Linguagem	Expressão	Resultado
Pseudocódigo	(5>(3+1)) OU (8 > 7)	Verdadeiro
Java	(5>(3+1)) || (8 > 7)	

(5 > (3 + 1)) OU (8 > 7)
VERDADEIRO OU VERDADEIRO
VERDADEIRO

d)

Linguagem	Expressão	Resultado
Pseudocódigo	(((exp(3,2) +1) = 10) E (NAO (10 = 10))	Falso
Java	(((3^2)+1) == 10) && (!(10==10))	

(((exp(3,2) + 1) = 10) E (NAO (10 = 10))
VERDADEIRO E FALSO
FALSO

e)

Linguagem	Expressão	Resultado
Pseudocódigo	(NAO((8*2) = 16)) E ((15 mod 4) >=2)	Falso
Java	(! (8*2) == 16)) && ((15 % 2) >= 2)	

(NAO ((8 * 2) = 16)) E ((15 mod 4) >= 2)
FALSO E VERDADEIRO
FALSO

f)

Linguagem	Expressão	Resultado
Pseudocódigo	((3 mod 2) = 1) E ((13mod3)= 1) OU ((3*4)<16)	Verdadeiro
Java	**((3 % 2) = 1) && ((13 % 3) == 1) || ((3*4) < 16)**	

((3 mod 2) = 1) E ((13 mod 3) = 1) OU ((3 * 4) < 16)
(VERDADEIRO E VERDADEIRO) OU (FALSO)
VERDADEIRO OU VERDADEIRO
VERDADEIRO

g)

Linguagem	Expressão	Resultado
Pseudocódigo	((18 > 13) E (exp(2, 5)+1 = 33)) E (((24 mod 8) > 0) E (72 < 80))	Falso
Java	((18 > 13) && (2 ^ 5)+1 = 33)) && (((24 % 8) > 0) && (72 < 80))	

((18 > 13) E (exp (2,5)+1 = 33)) E (((24 mod 8) > 0) E(72 < 80))
VERDADEIRO E VERDADEIRO E FALSO E VERDADEIRO
VERDADEIRO E FALSO
FALSO

h)

Linguagem	Expressão	Resultado
Pseudocódigo	NAO ((15*2)>(exp(3,2)*3) OU ((2*5)+(2.5*2)>=12))	Falso
Java	! ((15*2)>(3^2*3)) || ((2*5)+(2.5 * 2) >=12))	

NAO ((15 * 2) > (exp(3,2)*3) OU ((2*5) + (2.5 *2) >=12))
NAO (VERDADEIRO OU VERDADEIRO)
NAO (VERDADEIRO)
FALSO

ATIVIDADE ONLINE

Alguns problemas são resolvidos utilizando os fundamentos básicos de expressões numéricas. Para resolver esses problemas, deve-se estar atento à organização das informações fornecidas, bem como à utilização correta dos sinais operatórios, dos parênteses, colchetes e chaves.

Com base no que foi estudado nessa agenda, resolva as seguintes expressões lógicas elaboradas em Java, apontando se a expressão é verdadeira ou falsa:

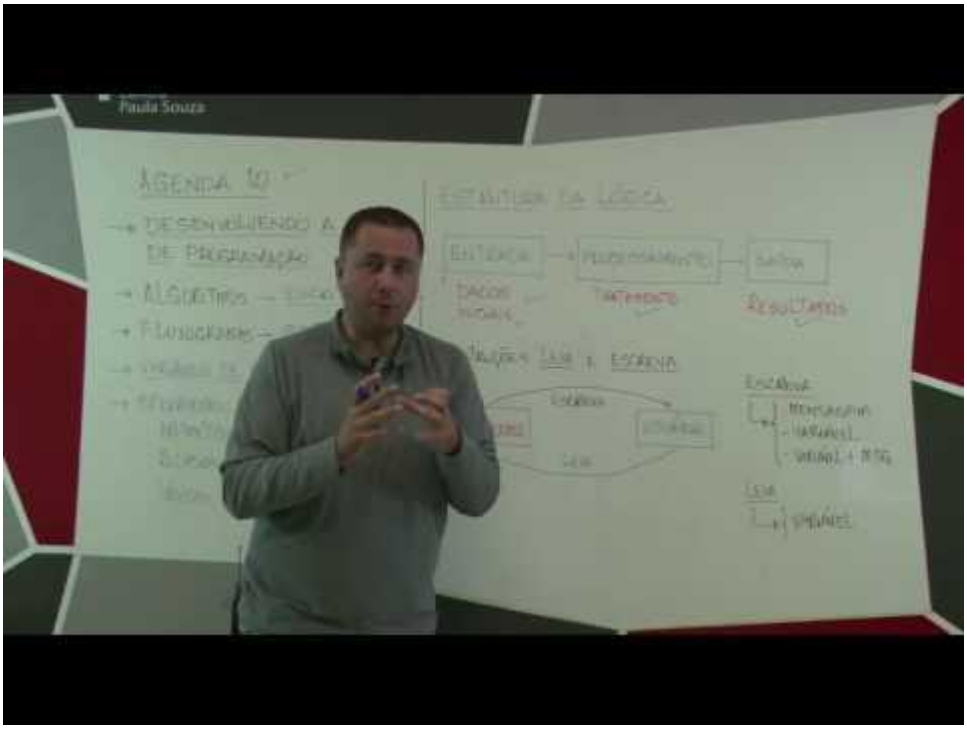
A) ! ((72 % 8) == 0)
B) ((3 ^ 2 * 1) > 9) || ((17 * 0) > 10)
C) ((3 + 2 / 2) >3) && ((12 – 5 % 2) == 11)
D) ! (2 + 3 == 5) || ! (3 > 2)

Este exercício deve ser entregue de forma on-line como atividade da agenda.

AMPLIANDO HORIZONTES

Não deixe também de assistir aos vídeos:

Desenvolvendo a Lógica – Aula Gravada



Link: https://youtu.be/aYYpwb7zAsU - Acessado em 14/12/2017

Informática - Módulo I - Agenda 11 - Operadores aritméticos, op. relacionais, operadores Lógicos

Link: https://www.youtube.com/watch?v=ntCQmyfhA30 - Acessado em 14/12/2017

Para aprofundamento dos temas discutidos nesta aula, seguem abaixo algumas dicas de filmes, livros e artigos que se relacionam com o conteúdo estudado. Estas dicas são muito importantes para você!

Vídeo:

Procure no YouTube uma Palylist denominada ;"Curso de Java para Iniciantes – Grátis, Completo e com Certificado " e assista as vídeo aulas de 1 a 8 disponível em : https://www.youtube.com/playlist?list=PLHz_AreHm4dkI2ZdjTwZA4mPMxWTfNSpR. Acessado em 14/12/2017.

Livros:

FORBELLONE, André L. V.; ELBERSPACHER, Henri Frederico. Lógica de Programação: A Construção de Algoritmos e Estrutura de Dados. Editora Pearson, 2000.

MANZANO, José Augusto N. G; OLIVEIRA, Jayr Figueiredo. Algoritmos: Lógica para Desenvolvimento de Programação. Editora Érica, 2007.

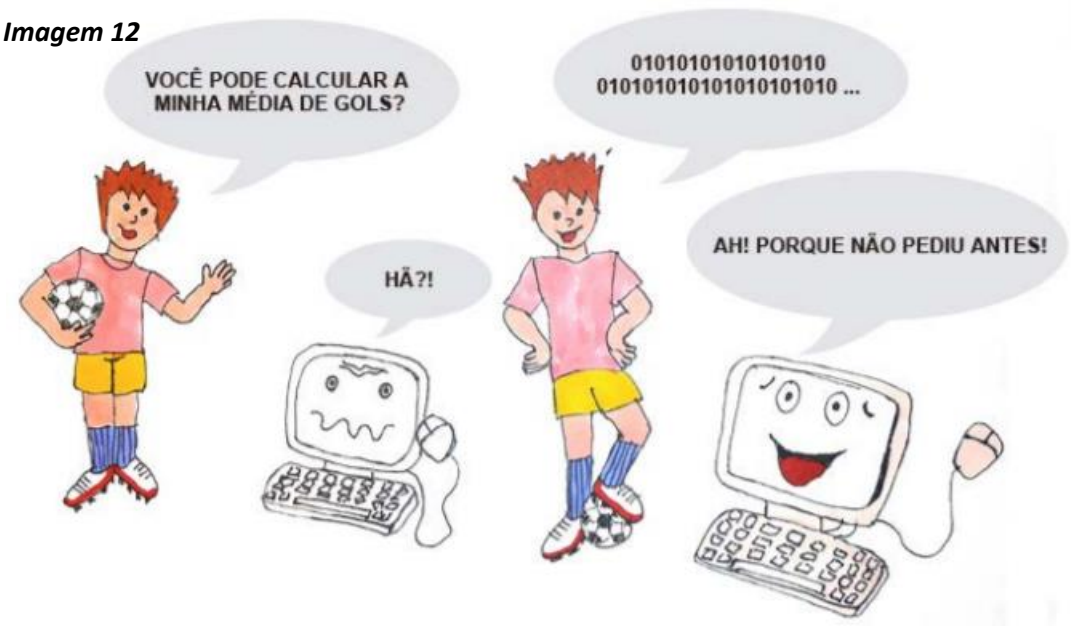
PUGA, SANDRA; RISSETTI, GERSON. Lógica de Programação e Estruturas de Dados com Aplicações em Java. Editora Pearson. 2009.

SCHILDT, HEBERT. Java para Iniciantes. Editora Bookman.2015.

AGENDA 10
A LÓGICA APLICADA EM JAVA

MOMENTO DE REFLEXÃO

Imagem 12



Os computadores não pensam por nós, nem entendem nossa língua! Para que eles possam executar uma ação, para que façam qualquer coisa é preciso explicar tudo nos mínimos detalhes e na língua deles. O problema é que a linguagem dos computadores é uma grande sequência de números binários, ou seja, zeros e uns como, por exemplo: 10111011101101100111011001100010101 0... Isto traz muita dificuldade para nós, seres humanos.

Você consegue se imaginar lendo ou escrevendo instruções de mais de mil páginas, decorando centenas de códigos binários? Será que você estaria fazendo um curso técnico de informática, se soubesse que teria que programar binários?

A fim de facilitar a comunicação entre homem e máquina, foram desenvolvidas as linguagens de programação que dispõem de um compilador que interpreta os comandos da linguagem e transforma em binários as instruções que serão entendidas pelo processador do computador. Uma das linguagens de programação é o Java que vamos estudar nesta agenda!

POR QUE APRENDER?

Assim como nós, o computador também tem um idioma para comunicação. Este Idioma é chamado de Linguagem de Programação. É como na realidade, onde em cada país se fala um idioma diferente para se comunicar.

Na prática, a Lógica de Programação, será os verbos, substantivos e características presentes em todos os idiomas. Aquelas frases que todos devem saber, principalmente caso viaje para outro pais, frases como “Como faço para chegar ali”, “Onde fica o banheiro” e “Preciso de ajuda”.

Em uma Linguagem de Programação, também precisamos aprender seus comandos básicos, que são derivados da Lógica de Programação, sendo assim, se aprendermos o momento certo para exibir uma mensagem na tela do usuário, podemos executar o processo com o auxílio de qualquer linguagem, basta “traduzir” o comando.

Para os nossos estudos, utilizaremos a linguagem de Programação Java, por ser gratuita e uma das linguagens mais utilizadas no mercado, como veremos a seguir.

PARA COMEÇAR O ASSUNTO...

Imagem 13

Você já pensou em quantas vezes por dia as pessoas acessam sites diversos a fim de consultar extratos bancários, fazer compras, estudar, pesquisar, ver as notícias que estão acontecendo no momento, etc...etc...etc. Você acredita que todas estas aplicações, inclusive aquelas envolvidas no processo de atualização das notícias que aparecem a cada minuto na tela do seu computador, podem ser desenvolvidas a partir do Java? A partir de agora vamos aplicar a lógica de programação na linguagem Java, fazendo um paralelo entre fluxogramas e pseudocódigos com a Linguagem Java. Preparado? Então vamos mergulhar no tema dessa agenda!

Java é uma linguagem orientada a objetos[1] que foi desenvolvida nos anos de 1990 pela Sun Microsystems, projetada para ser pequena, simples e portável a todas as plataformas e sistemas operacionais. É utilizada para desenvolver aplicativos corporativos, páginas web com conteúdo dinâmico e interativo, aprimorar a funcionalidade de servidores www e está cada vez mais sendo utilizada para desenvolvimento de aplicativos móveis para telefones celulares, pagers e PDAs.
Muito provavelmente você já deve ter ouvido falar sobre a linguagem Java. Basta ler uma revista de informática ou matérias sobre desenvolvimento de softwares que logo encontra alguma informação sobre Java, dado o sucesso que esta linguagem tem tido no mercado, principalmente pelo fato dos programas escritos em Java poderem ser executados virtualmente em qualquer plataforma e aceitos em qualquer tipo de computador ou outros aparelhos, uma característica marcante da Internet. Por isso, o Java tem se destacado muito no mercado e aprender esta linguagem é importante para você e para a sua profissão!

Java é uma Linguagem de Programação Orientada a Objetos, porém para que você compreenda melhor a Lógica utilizando esta Linguagem, iremos utilizar o Console (uma forma estruturada) para exercitar os comandos.

Para escrevermos um programa desenvolvido em Pseudocódigo de forma que o computador possa compila-lo[2] e executa-lo, precisamos utilizar uma Linguagem de Programação que nada mais é do que o "idioma" necessário para conversar com o computador. No desenvolvimento de nossos estudos em Lógica de Programação utilizaremos a linguagem de programação Java.

O Java atualmente é uma das linguagens de programação mais utilizadas no Mercado de Trabalho, ele também é capaz de fornecer uma portabilidade muito grande, sendo compatível com a maioria dos Sistemas Operacionais disponíveis para usuários gerais.
O mesmo programa escrito com a Linguagem de Programação Java, poderá ser utilizado em um computador com os Sistemas Operacionais Windows, Linux ou Mac OS sem que nenhuma linha do código fonte[3] necessite ser alterada, ganhando tempo no desenvolvimento de novas aplicações.

[1] Linguagem de Programação Orientada a Objetos (P.O.O) significa que esta Linguagem de Programação utilizará os conceitos de Orientação a Objetos (O.O.) o qual veremos no módulo seguinte com mais profundidade.
[2] Compilação é a ação de transformar um código amigável escrito com uma Linguagem de Programação em um programa executável baseado em Código de Máquina.
[3] Código Fonte é o conjunto de instruções criadas pelo programador utilizando uma Linguagem de Programação.

Além disto, o Java é uma linguagem de programação que pode ser utilizada para desenvolver páginas da Internet, através de um Servidor Web configurado para executar páginas do tipo Java Server Pages (jsp). Com ele também podemos desenvolver aplicativos para celulares que utilizam o Sistema Operacional Android.

Pela sua vasta lista de plataformas suportada, utilizaremos o Java como Linguagem de Programação de apoio no desenvolvimento da Lógica de Programação.

VOCÊ NO COMANDO

Você sabia que o Java não é a única linguagem de Programação que funciona em todos os principais Sistemas Operacionais do Mercado? Pesquise e Reflita sobre como outras Linguagens de Programação também existem no mercado e qual é a sua eficiência em relação ao Java.

Para iniciar o estudo, primeiramente é necessário conhecer as diferentes ferramentas para desenvolvimento que o Java oferece (JRE, JVM, JSE, JEE, JME, JDK...) parece uma sopa de letrinha, não é mesmo? Em seguida, com base na aplicação que pretende desenvolver, deve selecionar as tecnologias e ferramentas necessárias para este fim.

Imagem 14

É mais ou menos assim: Imagine que você precisa fazer um brigadeiro! Com a receita em mãos você vai até o supermercado, compra os ingredientes: leite condensado, chocolate em pó, manteiga e chocolate granulado e ainda providencia as ferramentas necessárias para a confecção do doce: panela e colher de pau.

Imagem 15

A mesma coisa acontece com o desenvolvimento de um aplicativo em Java! Conhecendo o tipo de aplicação que deseja desenvolver, você seleciona as ferramentas necessárias para tal finalidade.

Vamos lá:

Java Virtual Machine – JVM

O que faz com que a portabilidade do Java seja eficiente é uma aplicação responsável por executar programas desenvolvidos na linguagem. Sua função é simular um computador permitindo a execução do código fonte, por isto recebe o nome de Máquina Virtual.
Na prática, basta instalar em seu computador a JVM desenvolvida para o Sistema Operacional que você estará pronto para executar programas desenvolvidos em Java.

O Java Devlopment Kit (JDK) e a Aplication Programming Interface (API) Eclipse

Como em breve seremos desenvolvedores e não usuários, também precisaremos instalar em nosso computador o kit para desenvolvimento de programas feitos em Java, o que inclui o compilador[4] da linguagem de programação e a Máquina Virtual (JVM). Sem ele não é possível finalizar um programa desenvolvido em Java, mesmo que você escreva o código fonte completo. Apesar de já ser possível criar programas apenas com o JDK, utilizaremos uma interface de desenvolvimento (API) para nos auxiliar na escrita, compilação e testes dos nossos programas. O Java tem como principais APIs de desenvolvimento o NetBeans e o Eclipse. Independente da API escolhida os comandos sempre serão os mesmos, logo, se você aprende a programar em Java, conseguirá utilizar qualquer uma das APIs sem maiores problemas.
Durante o desenvolvimento das atividades utilizaremos a API Eclipse, porém você pode utilizar qualquer outra API desde que siga fielmente as estruturas listadas neste material.

Durante os nossos estudos, utilizaremos a distribuição do Java SE(Standard Edition), que é voltada para o Desenvolvimento de Sistemas Desktop. Existem também as Distribuições Java ME (Micro Edition) que é voltada para dispositivos de pequeno porte e Java EE(Enterprise Edition) que é direcionado a aplicações corporativas e Web.

Criando um projeto utilizando a API Eclipse

Na estrutura de organização do Eclipse, cada programa desenvolvido é caracterizado como um Projeto. E cada código fonte, será tratado como uma Classe.

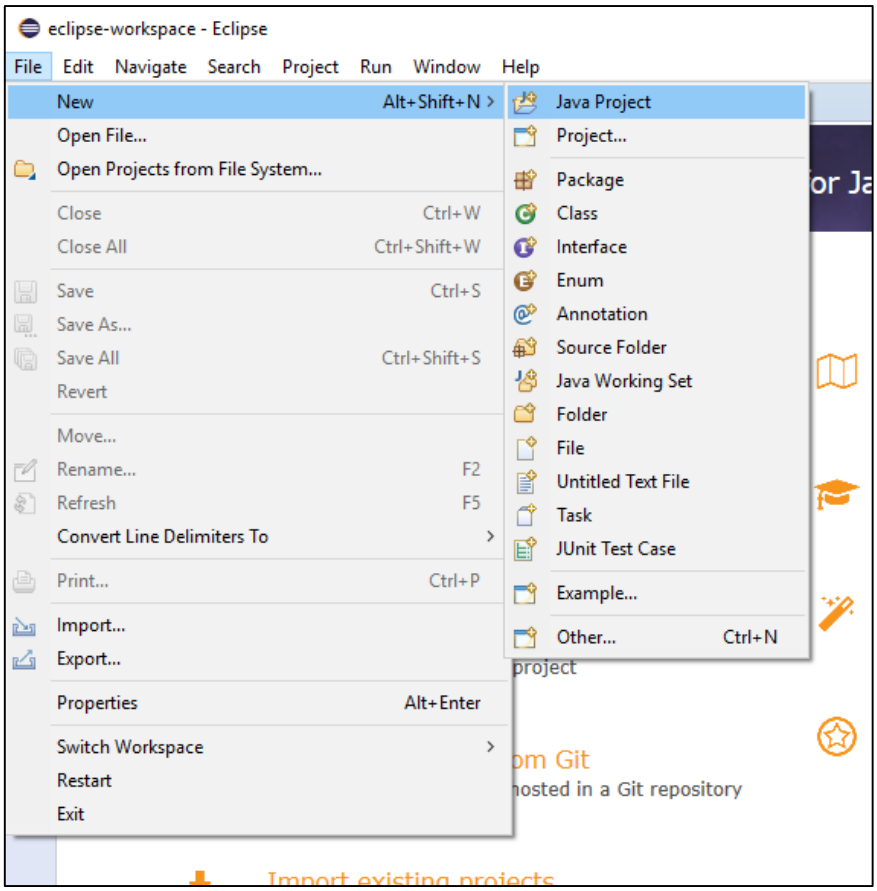
[4] Compilador é a ferramenta utilizada para compilar um programa desenvolvido com o auxílio de uma Linguagem de Programação.

Como utilizaremos o Java como uma Linguagem de Programação estruturada[5], não entraremos a fundo no significado de Projeto, Pacote e Classe, pois estas definições são utilizadas em programas Orientados a Objetos[6].

VOCÊ NO COMANDO

Quais outras APIs que suportam a Linguagem de Programação Java existem no Mercado? Todas elas são gratuitas? Quais são suas Vantagens e Desvantagens em relação a API Eclipse? Pesquise e Reflita antes de prosseguir com a leitura.

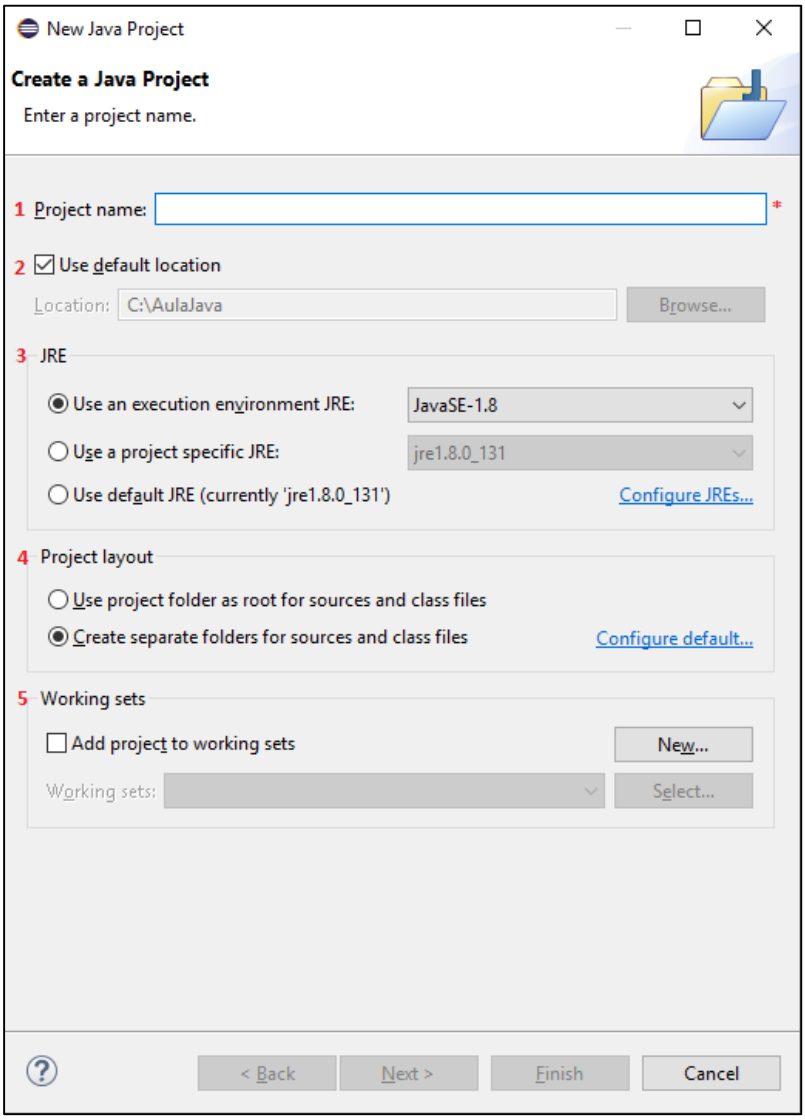
Para criarmos um novo programa, após abrir a janela principal do Eclipse, selecionamos o menu File, New, Java Project, conforme a figura a seguir:



[5] A Programação Estruturada é uma forma de desenvolver programas baseando-se no conceito de inicio – meio – fim de forma linear e em um único código fonte.

[6] A Orientação a Objetos é uma forma de Desenvolver Programas baseada em Classes e Objetos, afim de aproximar a forma na qual o código fonte é desenvolvido das nossas ações do dia a dia.

A seguinte janela será exibida:



As opções fornecidas são:

1. Nome do Projeto: nesta opção você deve indicar um nome para o seu programa. O nome do programa deve ser simples e objetivo. Não é recomendado que seja utilizado espaços para separar palavras. Caso seja necessário, utilize o underline. Este campo é obrigatório.

2. Selecionar onde ficará salvo o projeto: é altamente recomendável que todos os projetos desenvolvidos com o Eclipse sejam salvos em um único lugar, preferencialmente no local indicado ao iniciar o Eclipse. Desmarque a opção "Use default location" apenas se a sua intenção for salvar o seu projeto em um local diferente do padrão.

3. JRE: você só deve alterar esta opção caso você tenha o JDK instalado em seu computador em mais de uma versão e deseje testar seus programas em uma versão mais antiga do JDK.

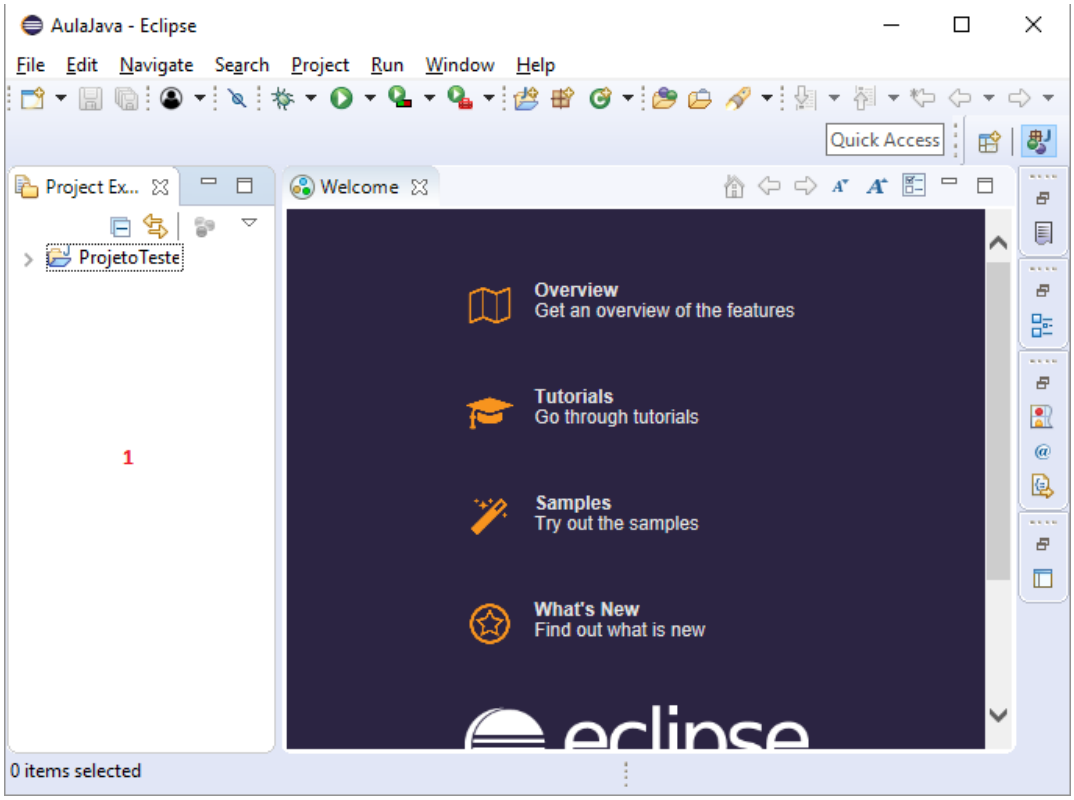
4. Project Layout: nesta opção você indicará como o Eclipse deve organizar os arquivos de seu programa. O ideal é manter a opção padrão selecionada, pois ela manterá seus códigos fontes sempre em uma pasta separada do restante do projeto.

5. Working Sets: apenas para usuários avançados. Com esta opção você poderá mesclar dados do seu projeto com o de outros projetos localizados em diferentes workspaces.

Após inserir um nome para o seu projeto e alterar as demais opções, caso seja necessário, clique em Finish. Você voltará para a tela inicial, porém desta vez haverá um pequeno botão ao lado esquerdo ou direito da do Eclipse, dependendo da versão utilizada. O botão aparenta conforme a imagem a seguir:

Figura 5- Arquivo GEEaD

Ao clicar no botão, aparecerá a nossa interface do projeto conforme a imagem a seguir:



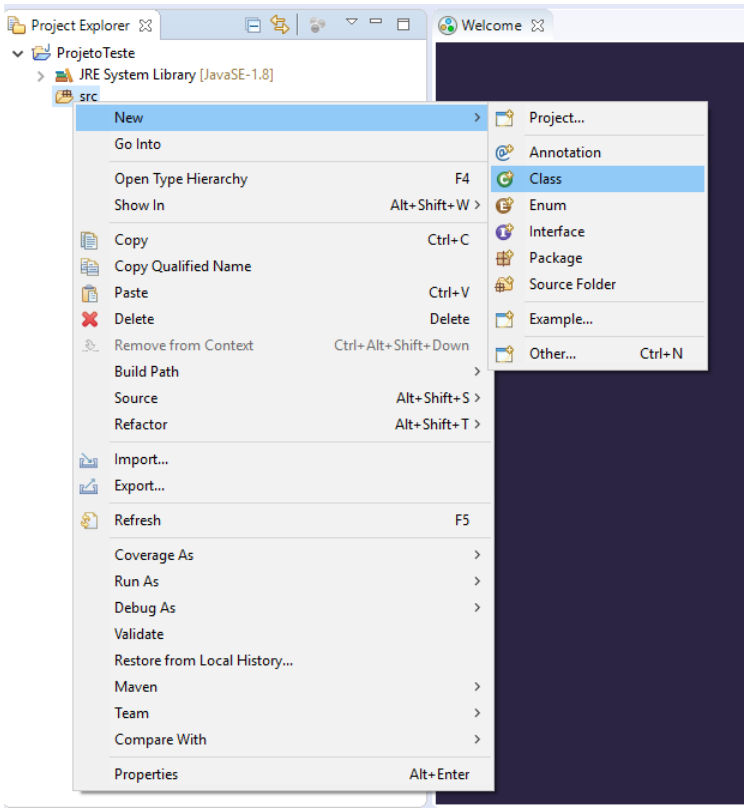
Note que ao lado direito, marcado com o número **1**, temos a janela Project Explorer. Será por esta Janela que localizaremos os arquivos contidos no nosso projeto.

Importante: Caso você feche acidentalmente a janela Project Explorer, poderá acessa-la novamente através do menu Window > Show View > Project Explorer.

Agora que temos o projeto, vamos criar um arquivo para o nosso código fonte:

Em Project Explorer, entre na pasta referente ao seu projeto, localize a pasta chamada src[7], clique com o botão direito nela e selecione New > Class:

[7] A pasta src (abreviação para source Code) é a pasta padrão do eclipse para receber códigos fontes.



Por não utilizarmos a Orientação a Objetos na programação Java, precisaremos apenas definir um nome para o arquivo **(1)** e selecionar a opção "public static void main(String[] args)" **(2)** :

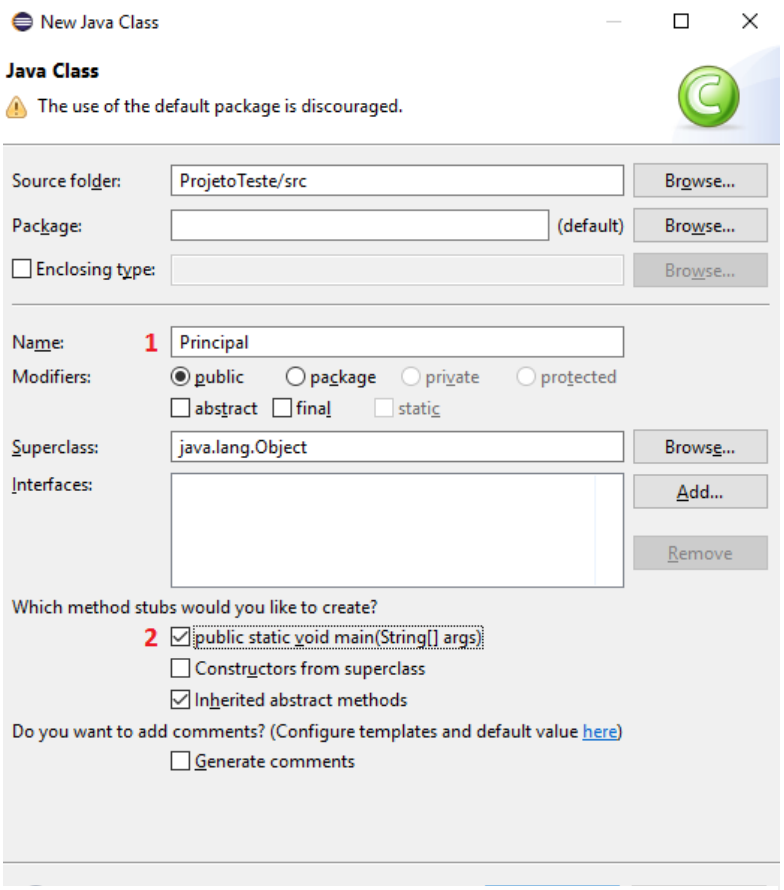
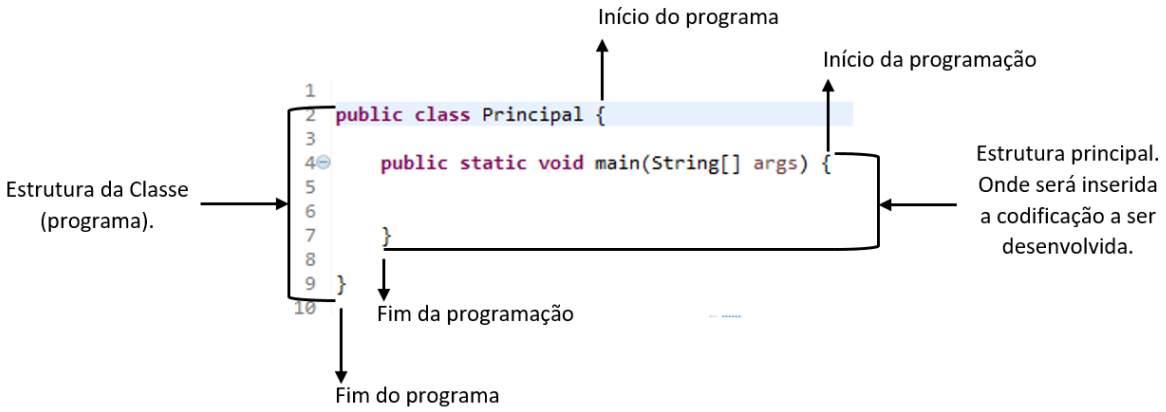


Figura 6- Arquivo GEEaD

Assim como na criação do projeto, o nome do arquivo de código fonte não pode conter espaços, acentuação e caracteres especiais. Fique atento!

A estrutura de um programa feito em Java

Agora que você já criou um código Fonte em Java, você poderá aprender a estrutura básica de um programa. A seguir temos uma imagem representando um código fonte em Java:



Você pode ver na imagem que o programa em si está dentro de uma **Classe**. Esta estrutura é necessária para permitir que no futuro o seu programa em Java se torne um programa Orientado a Objetos.

Os comandos Início e Fim contidos no Pseudocódigo, são substituídos por chaves { } em Java, sendo a abertura de Chave { o início e, o fechamento de chave } o final do seu programa. Em algumas estruturas que veremos posteriormente, também utilizaremos marcações de início e fim através de chaves.

A partir de agora, os novos comandos apresentados a você sempre serão apresentados em Java, além da sua forma utilizada no Fluxograma e no Pseudocódigo.

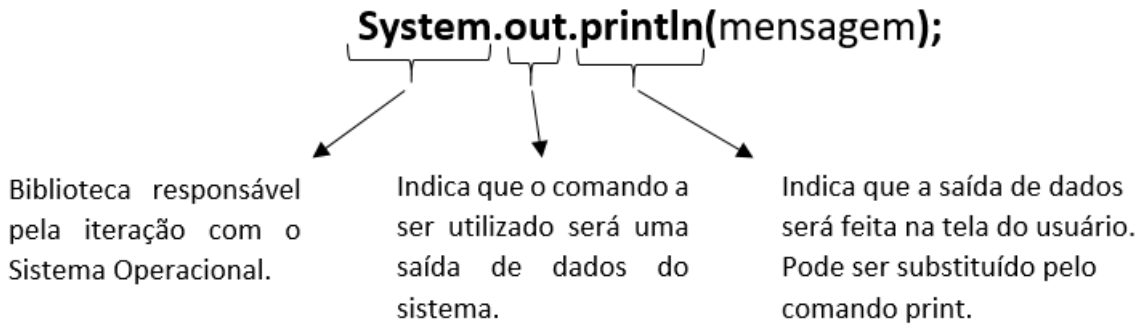
O comando Escreva

Quando desejamos exibir alguma mensagem na tela do usuário utilizando o pseudocódigo, utilizamos o comando:

escreva *mensagem*

Com o auxílio dele é possível desde enviar uma simples mensagem ao usuário do programa, como também mostrar o resultado de uma conta. Na prática: tudo o que você deseja que o usuário veja, deverá utilizar este comando para mostrar.

Em Java, por ser uma linguagem de programação, devemos indicar o caminho completo da operação de exibição de mensagem para o usuário, que é feita da seguinte forma:



VOCÊ NO COMANDO

Agora que vimos o comando print e println reflita: Qual dos dois comandos você acha que é o mais eficiente? Existe alguma situação específica para a utilização de cada um dos comandos?

A resposta é muito simples: Depende do caso!

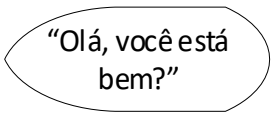
O comando print escreverá a mensagem na tela sem efetuar nenhuma modificação na mensagem.

O comando println pulará uma linha na tela, antes de iniciar a escrita da mensagem.

Portanto, se a sua intenção é exibir a mensagem **grudada na mensagem exibida anteriormente**, utilize o **print**, caso você deseje que a mensagem **seja exibida em uma nova linha**, utilize o **println**.

Por exemplo: Desejamos que o usuário veja a seguinte mensagem:

Olá, você está bem?

Fluxograma	Pseudocódigo	Java
	escreva "Olá, você está bem?"	System.out.println("Olá, você está bem?");

Declarando uma variável

Tipos de variáveis Pseudocódigo	Tipos de variáveis Java	Valores compreendidos dentro do tipo	Exemplo de valores em Java.
Inteiro	byte	Números entre -128 e 127.	10
	short	Números entre -32768 e 32767.	13320
	Int	Números entre -2147483648 e 2147483647.	-170000000
	long	Números entre -9223372036854775808 e 9223372036854775807.	14000222999333
Real	float	Números reais entre -10^{38} até 10^{38}.	0.134
	double	Números reais entre -10^{308} até 10^{308}.	143.4938293019283
Caractere	char	Um único caractere (letra), entre apóstrofos. Exemplo: 'a'.	'd'
	String	Mais de um caractere, entre aspas. Exemplo: "Técnico em Informática".	"Informática"
Lógico	boolean	Verdadeiro ou Falso.	falso

Note que em Java, o único tipo de dado que se inicia com letra maiúscula é o String. Vale lembrar também que os números decimais são separados por ponto (.) ao invés de vírgula.

Voltando ao caso da Agenda Telefônica, qual seria o tipo de dados que o número de telefone dado pelo Juvenal seria?

O número de telefone seria uma variável do tipo String, pois apesar de ser um número de telefone, não efetuamos nenhum tipo de cálculo com este número, sendo assim, não é necessário definir este tipo de dado como inteiro.

Após identificar qual será o tipo da variável que você utilizará, basta declara-la[8] em seu programa, seguindo o padrão para cada linguagem de programação, sendo:

PSEUDOCÓDIGO	JAVA
declare	
nomedavariavel *como* ***tipododado***	***tipododado*** **nomedavariavel;**

Abaixo, temos um exemplo de sua aplicação utilizando o fluxograma, pseudocódigo e o Java:

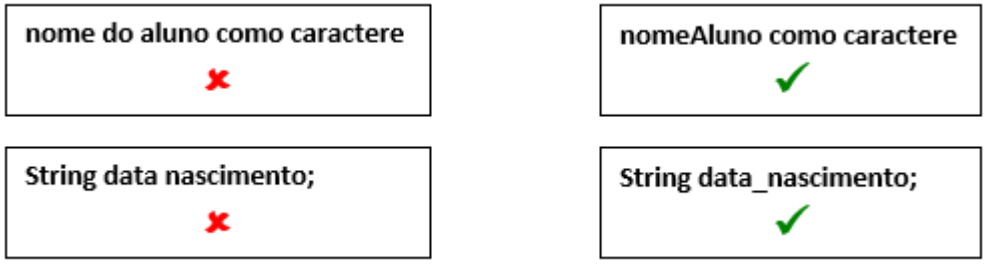
	Fluxograma	Pseudocódigo	Java
Exemplo	nota1 como real nota2 como real nota3 como real media como real	declare nome como caractere idade como inteiro preco como real	String nome; Int idade; Double preco;

Como nomear uma variável

As variáveis devem ser nomeadas de forma objetiva, que esclareça facilmente o programador qual é a sua função, para garantir um rápido entendimento e continuidade do desenvolvimento do software caso necessário.

Quando nomeamos as variáveis, é imprescindível também seguir algumas regras, que são:

1. As variáveis nunca podem conter um espaço em seu nome.



Caso você necessite de mais de uma palavra para definir o nome de uma variável, junte as palavras ou então separe-as apenas com um **underline** _ .

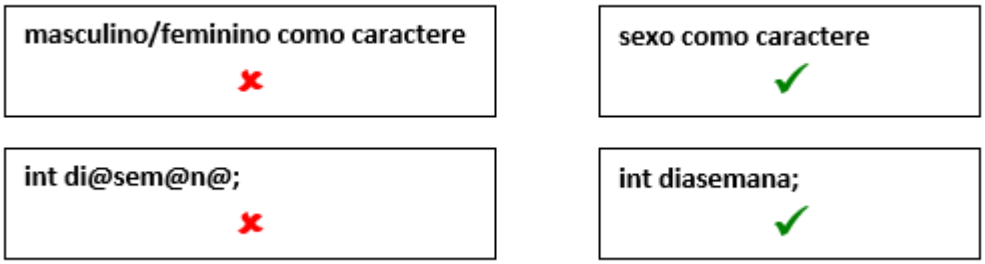
Remova as preposições do nome da variável, assim o nome dela ficará mais objetivo e fácil de entender.

Não inclua mais do que duas palavras em um nome de variável, seja sempre objetivo.

2. As variáveis nunca podem conter caracteres especiais em seu nome.

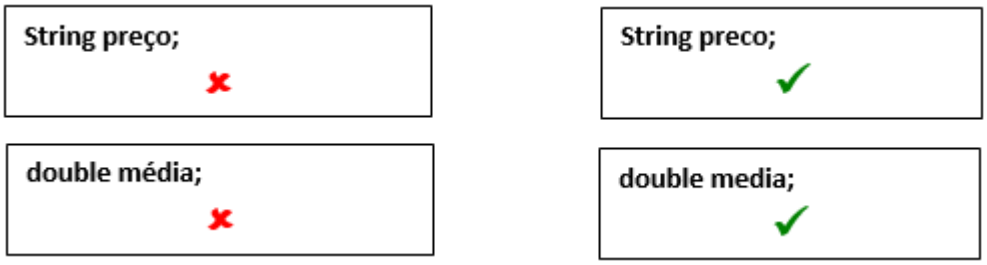
[8] Com a declaração de variável avisamos o computador que ele deve criar um espaço na memória para receber um valor posteriormente. Todo o espaço na memória declarado deve ter um nome e ser vinculado a um tipo de variável.

Em uma linguagem de programação, os caracteres especiais são palavras reservadas que são utilizadas pela linguagem para trabalhar comandos especiais, cálculos etc. Se você utilizar caracteres especiais em nome de variáveis, o computador não entenderá se ele deverá demarcar o espaço da variável na memória ou se iniciará algum procedimento especial do computador, por isto tentar colocar um nome destes resultará em erro de compilação[9].



Entende-se por caracteres especiais os seguintes sinais: !, @, #, $, %, \ , /,], [, (,), {, }, e todos os caracteres não alfanuméricos.

3. Nomes de variáveis não podem receber acentuação.
   Como as linguagens de programação são todas escritas no idioma Inglês, onde não há acentuação nas palavras, não podemos utiliza-los em declarações de variáveis. O cê-cedilha (Ç) apesar de ser um caractere, também entra nesta regra.



4. Nomes de variáveis não podem ser iniciados por números.
   Quando a linguagem de programação encontra um número em uma codificação, logo ela entende que deverá ser feito um cálculo. Caso este número venha junto com um caractere em seguida, o computador não identificará o caractere como sendo um número válido para efetuar um cálculo, assim gerando um erro de compilação.
   Você poderá utilizar um número normalmente no nome da sua variável desde que este número não seja o primeiro caractere de seu nome.

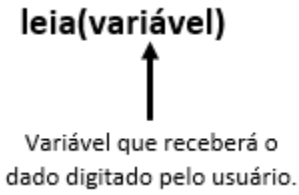
[9] Erros de compilação são configurados quando há erros na estrutura do código fonte. Caso este tipo de erro ocorra, não será possível a execução do programa.

1nota como real ✘	nota1 como real ✓
String 10convidado; ✘	String convidado10; ✓

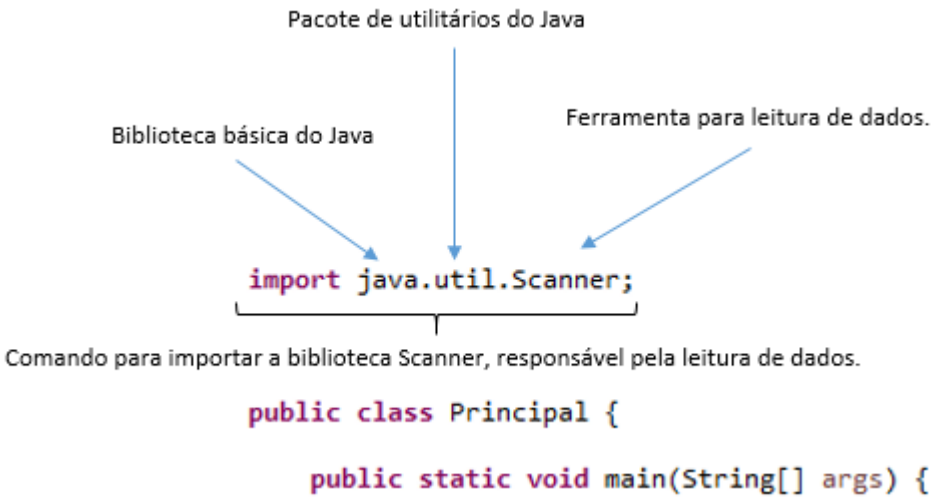
O comando Leia

O comando Leia é o comando responsável por receber dados inseridos pelo usuário. Em geral estes dados são inseridos através do teclado, podendo ser numérico ou caractere dependendo do tipo de dados que a variável que receberá o valor estiver configurada.
Como o computador não saberá qual será o valor que o usuário digitará, sempre teremos que utilizar uma variável para armazenar o valor obtido através do comando Leia. A sintaxe do comando Leia é:

Pseudocódigo:



Java:
Em Java um programa inicial contém apenas o suporte a exibição de mensagens no monitor, processamentos básicos de dados e utilização de variáveis com o objetivo de garantir a otimização do espaço de seu programa em disco e consequentemente, o peso da sua aplicação.
Para que seja possível a utilização de recursos diferentes, é necessário realizar uma importação de uma biblioteca de classes para o seu Projeto.
Estas bibliotecas contém as instruções necessárias para que o Java consiga trabalhar com novas funções conforme a necessidade do programador. Vale lembrar que não é recomendado que você faça uma importação de biblioteca em seu programa caso não seja a intenção utiliza-la, podendo acarretar perca de desempenho desnecessária na sua aplicação.
Para importar uma biblioteca, basta seguir o seguinte comando:



Note que o comando import deve ser inserido na primeira linha do seu código, antes mesmo de todos os códigos gerados automaticamente pela IDE Eclipse.

Para entender melhor o comando:

Imagem 16

Imagine que você está trabalhando na construção de um objeto sobre uma mesa. Na mesa ficarão apenas as ferramentas mais utilizadas, como chave de fenda e alicate. Caso você necessite de uma chave inglesa, você deverá ir até a sua mala de ferramentas, no compartimento de chaves e trazer a chave inglesa para a sua mesa, não é mesmo?

Na prática, caso a ferramenta que você necessite não esteja disponível, provavelmente você encontrará na sua mala de ferramentas (biblioteca Java), e dentro do compartimento de chaves (util).

Mas existe apenas a biblioteca Java para ser importada? Não! O Java possui inúmeras bibliotecas que poderão ser importadas sempre que necessário. Além disto você também poderá importar bibliotecas feitas por outras pessoas, com o objetivo de poupar muito trabalho no desenvolvimento de uma nova ferramenta da estaca zero.
Voltando a leitura de dados. Após a importação da ferramenta Scanner, precisamos “cria-la” dentro do nosso programa, utilizando o seguinte comando:

O comando para criar o leitor dentro do nosso programa é chamado de instância. Na instância é onde a sua ferramenta importada cria vida, tornando-se funcional e utilizável na sua aplicação. A partir deste momento, o leitor será carregado na memória do computador junto com a sua aplicação.

Leitura de dados utilizando a ferramenta Scanner

Agora que já temos o nosso leitor, estamos prontos para ler uma entrada de dados feita pelo usuário através do teclado e armazena-la em uma variável. Para que esta leitura seja feita de forma adequada pelo Java, devemos adotar uma leitura específica para cada tipo de variável, conforme a tabela a seguir:

Tipo da variável que receberá o dado (Java)	Comando utilizado pelo leitor	Exemplo
byte	leitor.nextByte();	byte numero; numero = leitor.nextByte();
short	leitor.nextShort();	short numero; numero = leitor.nextShort();
int	leitor.nextInt();	Int numero; numero = leitor.nextint();
long	leitor.nextLong();	long numero; numero = leitor.nextLong();
float	leitor.nextFloat();	float numero; numero = leitor.nextFloat();
double	leitor.nextDouble();	double numero; numero = leitor.nextDouble();
char	leitor.next().charAt(0);	char letra; letra = leitor.next().charAt(0);
String	leitor.next();	String palavra; palavra = leitor.next();
boolean	leitor.nextBoolean();	Boolean teste;

		teste = leitor.nextBoolean();

Exemplo prático de um programa

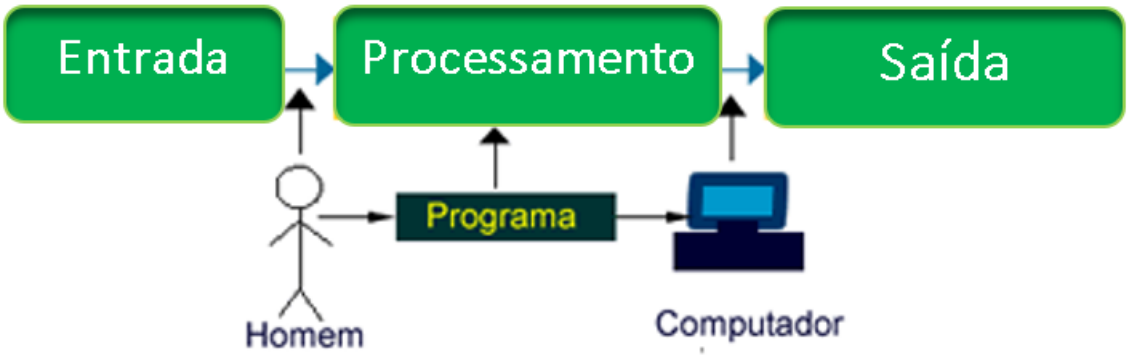
Agora que já vimos individualmente as principais ferramentas de uma linguagem de programação, chegou a hora de praticarmos, produzindo um programa completo. Para tanto criar um programa que calcule a soma de dois números digitados pelo usuário:

Passo 1: Definir a sequência lógica do programa.

Por ser o nosso primeiro programa, antes de construir o fluxograma, precisamos identificar a entrada, o processamento e a saída dos dados dentro do nosso programa:

Entrada: O programa deverá solicitar para que o usuário digite dois números, do tipo numérico.
Processamento: O programa calculará a soma destes números.
Saída: O programa exibirá a soma destes números.

Imagem 17



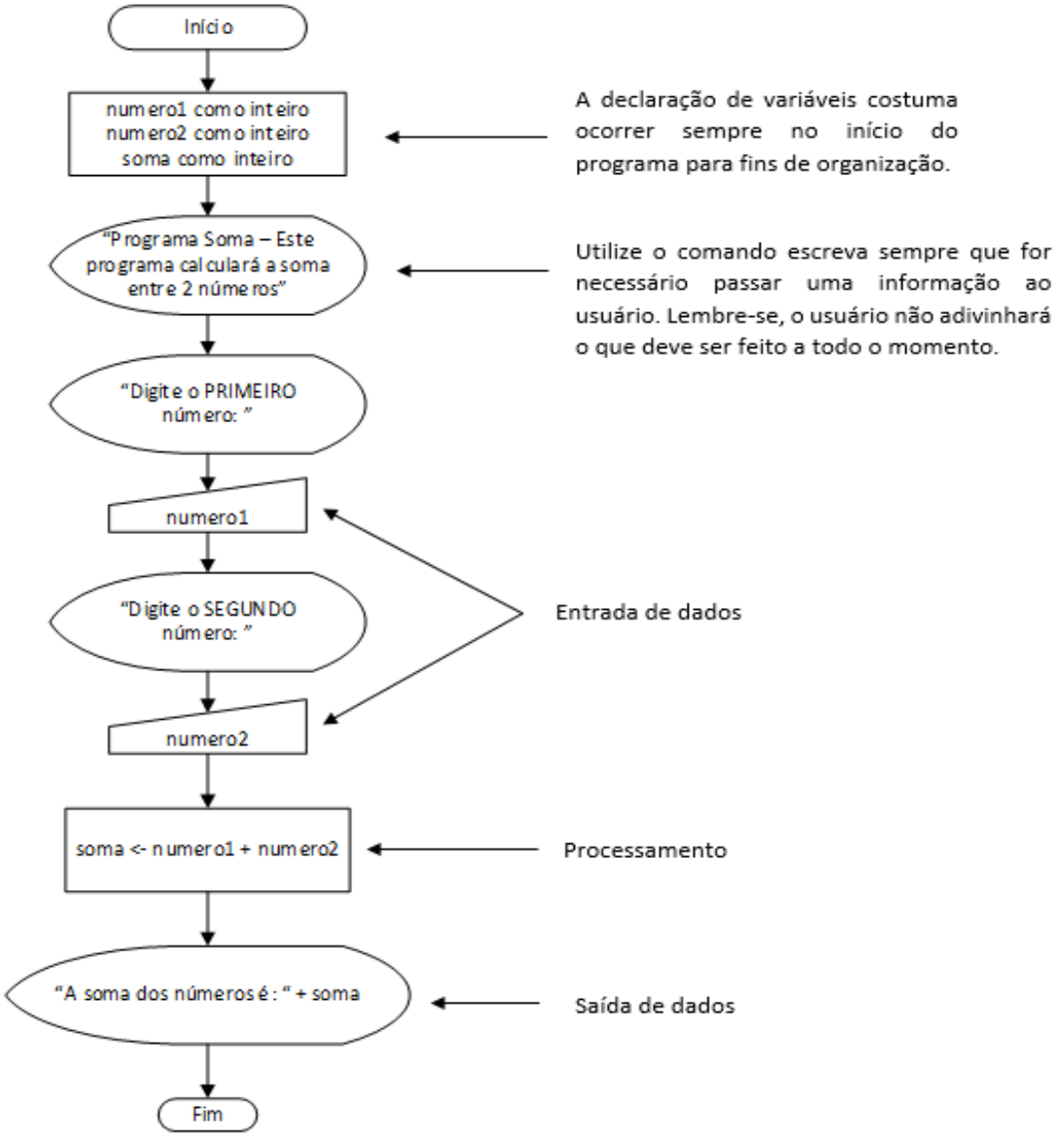
Passo 2: Definir quais serão as variáveis necessárias para o programa.

Seguindo a definição de variáveis, precisamos identificar quais dados não serão fixos no nosso programa. Verificando as informações construídas no passo 1, nota-se que os dois números que o usuário digitar e a soma deles não são fixos, podendo ser modificado a cada vez em que o usuário utilizar o programa.
Logo, as variáveis serão:

numero1 como inteiro
numero2 como inteiro
Soma como inteiro

Passo 3: Construir o Fluxograma.

Agora que já sabemos as variáveis e a sequência lógica do nosso programa, podemos construir o fluxograma conforme a simbologia apresentada anteriormente:



Sempre procure identificar a entrada, o processamento e a saída dos seus dados em qualquer fluxograma, assim a chance de conter erros no fluxograma cairá consideravelmente.

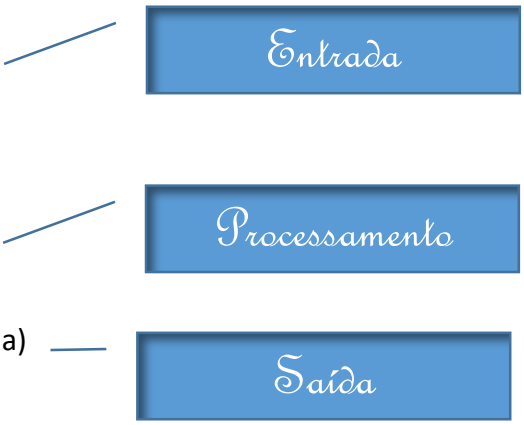
Passo 4: **Construir o Pseudocódigo**

Após construir o fluxograma, ficará muito fácil criar o Pseudocódigo, basta "traduzir" a simbologia do Fluxograma para o Pseudocódigo:

Declare
Numero 1 como inteiro
Numero 2 como inteiro
Soma como inteiro
Inicio
Escreva(“Programa Soma – Este programa calculará a soma entre 2 números”)
Escreva(“Digite o PRIMEIRO número”)
Leia(numero1)
Escreva(“Digite o SEGUNDO número”)
Leia(numero2)

Soma <- numero1 + numero2

Escreva(“A soma dos números é: “ + soma)
Fim



VOCÊ NO COMANDO

Com o Pseudocódigo já é possível testar o nosso código, com o auxílio da ferramenta VisualG, caso você tenha interesse em testar o pseudocódigo execute-o utilizando a ferramenta.

Passo 5: Construir o programa em Java.

Assim como fizemos com o Pseudocódigo, basta aplicar as regras básicas da linguagem Java.

Em primeiro lugar, criaremos um novo projeto para este exemplo da mesma forma na qual vimos anteriormente.

A seguir, aplicamos o Pseudocódigo, adaptando-o para a linguagem Java. Não podemos esquecer de importar a biblioteca responsável pela leitura de dados:

*SomaValores.java

```java
import java.util.Scanner;
public class SomaValores {

    public static void main(String[] args) {
        /*  O código desenvolvido por você deverá estar aqui.
            As chaves marcam o inicio do código
            Diferente do pseudocódigo, as variáveis em java podem ser declaradas diretamente
            no corpo do programa, mas a recomendação continua: devemos declarar as variáveis
            logo no incio do código fonte.
        */
        //declaração das variáveis
        int numero1;
        int numero2;
        int soma;
        //para facilitar o entendimento, também habilitamos o leitor no inicio do código.
        Scanner leitor = new Scanner(System.in);
        //inicio do programa
        System.out.println("Programa Soma - Este programa calculará a soma entre 2 números");
        System.out.println("Digite o PRIMEIRO valor");
        //leitura do primeiro valor
        numero1 = leitor.nextInt();
        System.out.println("Digite o SEGUNDO valor");
        //leitura do segundo valor
        numero2 = leitor.nextInt();
        //processamento
        soma = numero1 + numero2;
        //saída de dados
        System.out.println("O resultado da soma é "+ soma);
    }
}
```

Algumas considerações:

1. Você percebeu que há explicações precedidas por duas barras // ou /* ? Estes sinais indicam comentários do Java, com eles você poderá escrever qualquer mensagem para o programador ou então anotar informações importantes sobre o seu código. Estes códigos não serão processados pelo computador.
    a. Os comandos para iniciar um comentário em Java são:
    b. // comentário de uma única linha

    c. /*
        i. Comentário de múltiplas linhas
    d. */
    e. A codificação indicada como comentário ficará por padrão na cor verde.

2. O Java é uma linguagem Case Sensitive, ou seja, o Java diferencia as letras maiúsculas de minúsculas. Para o Java a letra “a” minúscula é diferente da letra “A” maiúscula. Por este motivo, os comandos da linguagem devem ser reproduzidos fielmente a sua apresentação, caso contrário o comando não será reconhecido. É o caso do System.out.println(), onde o S deve ser sempre digitado em letra maiúscula.

3. Todo o comando feito em Java deve ser finalizado por um ponto e virgula (;). É através do ponto e virgula que o Java entende que o comando foi finalizado, executando-o e preparando-se para receber o próximo comando. Caso você esqueça do ponto e vírgula (que por sinal é o erro mais comum), o seu programa não funcionará.

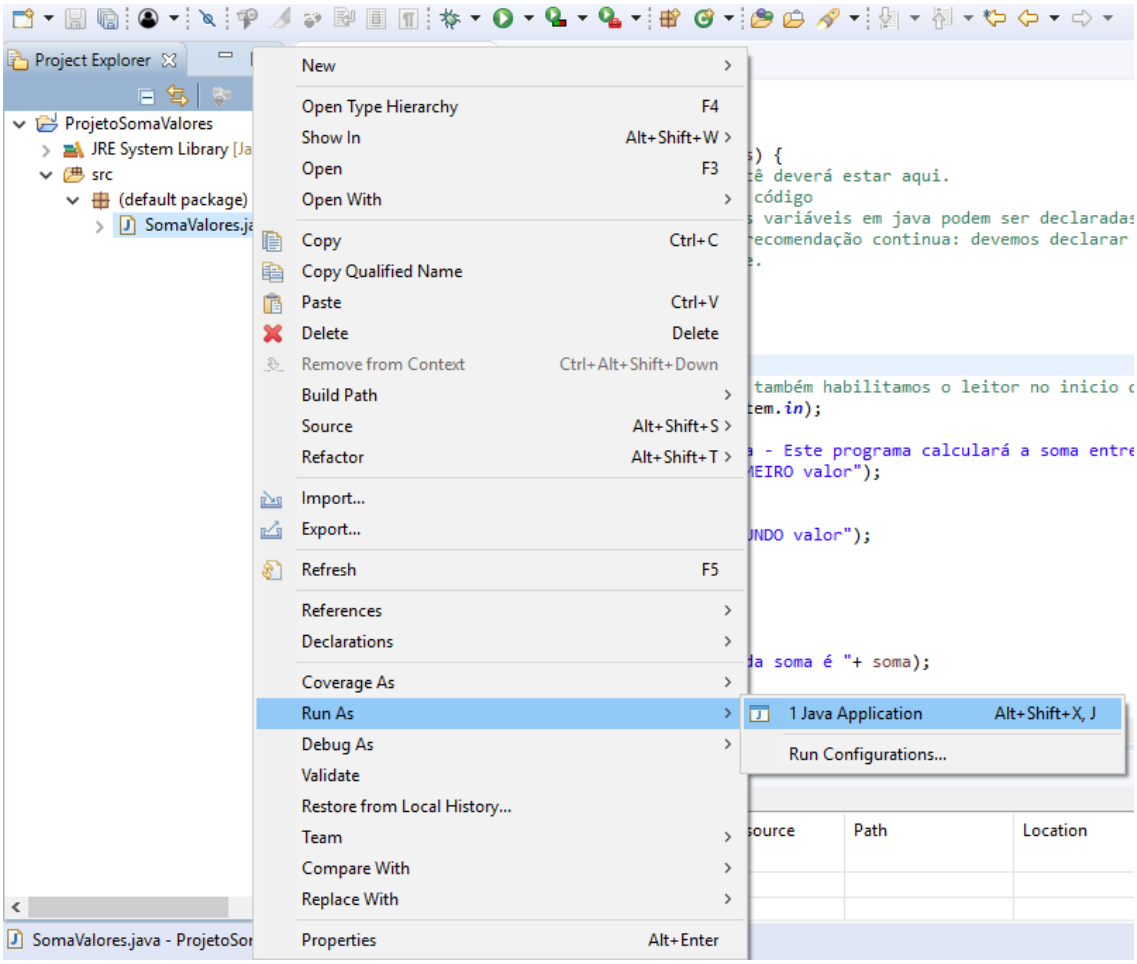
4. Caso o seu texto digitado com auxílio do Eclipse fique sublinhado em vermelho, significa que ele está com erro. Caso isto ocorra, primeiro você deve sempre corrigi-lo, caso contrário seu programa não funcionará. Geralmente estes erros são causados por "erro de sintaxe", em outras palavras, o comando foi digitado incorretamente.

5. Comandos sublinhados em amarelo não são erros. Geralmente são recomendações ou aviso de variáveis declaradas, mas não utilizadas.

6. Se você passar o mouse em cima de palavras sublinhadas, a IDE apontará as recomendações necessárias para corrigir o problema. Mas atenção: nem sempre a IDE acertará o palpite, lembre-se que o computador não é perfeito, portanto o ideal é sempre rever o código e entender o que está errado.

Passo 6: Executando o programa.

A última etapa do nosso programa é executa-lo e testar para ver se tudo ocorreu bem. Para testa-lo, basta clicar no nome da sua classe com o botão direito (em Project Explorer) e selecionar a opção Run > Java Application:



Abaixo do seu Código Fonte, você verá uma janela chamada Console, onde você poderá inserir os dados para a utilização de seu programa:

SomaValores.java

```
1  import java.util.Scanner;
2  public class SomaValores {
3
4      public static void main(String[] args) {
5          /*  O código desenvolvido por você deverá estar aqui.
6              As chaves marcam o inicio do código
7              Diferente do pseudocódigo, as variáveis em java podem ser declaradas diretamente
8              no corpo do programa, mas a recomendação continua: devemos declarar as variáveis
9              logo no incio do código fonte.
10         */
11         //declaração das variáveis
12         int numero1;
13         int numero2;
14         int soma;
15         //para facilitar o entendimento, também habilitamos o leitor no inicio do código.
16         Scanner leitor = new Scanner(System.in);
17         //inicio do programa
18         System.out.println("Programa Soma - Este programa calculará a soma entre 2 números");
19         System.out.println("Digite o PRIMEIRO valor");
```

Problems @ Javadoc Declaration Console

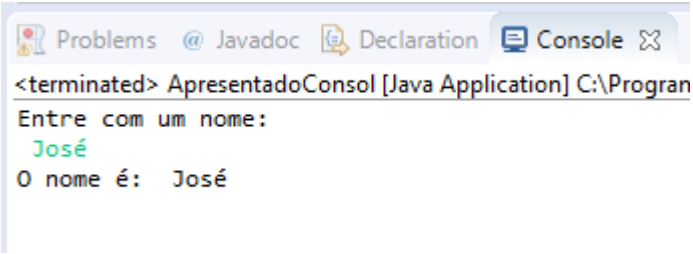
```
<terminated> SomaValores [Java Application] C:\Program Files\Java\jre1.8.0_131\bin\javaw.exe (19 de set de 2017 21:14:10)
Programa Soma - Este programa calculará a soma entre 2 números
Digite o PRIMEIRO valor
3
Digite o SEGUNDO valor
2
O resultado da soma é 5
```

Caso a janela Console não apareça, você pode abri-la acessando o menu Window > Show View > Console.

Vale lembrar que caso haja erros em seu Código Fonte, o Eclipse retornará uma mensagem de erro, não sendo possível executar o seu código enquanto os erros (sublinhados em vermelho) não sejam corrigidos.

Aprimorando a comunicação com o usuário

Antes de mais nada, os programas devem ser fáceis de se utilizar e os usuários devem fazer isso intuitivamente. Para tanto todo programador deve atentar-se a interface com o usuário. Vamos aprender uma maneira alternativa de entrada de saída de dados mais elegante para utilizarmos em nossos futuros programas em Java. Até aqui utilizamos o que chamamos de modo console, ou seja, uma tela de terminal que não aceitava elementos gráficos para interagirmos com o programa.



VOCÊ NO COMANDO

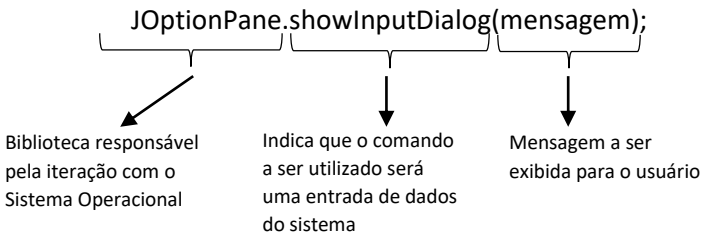
Um programador deve sempre tentar desenvolver uma interface com o usuário que seja simples de ser utilizada. Contudo essa interface deve agradar ao cliente que está contratando os seus serviços. Pensando como o programador, reflita como isso pode interferir na prática de programação antes de prosseguir a leitura.

Atualmente vivemos em um mundo onde os elementos gráficos predominam em todos os aspectos de comunicação. Portanto será apresentado um modo gráfico utilizando as janelas do sistema operacional para realizarmos a entrada e a saída de dados.

Entrada de dados

Para realizarmos a entrada de dados para um programa utilizaremos o comando **JOptionPane** do Java.

A sintaxe do comando é a seguinte:

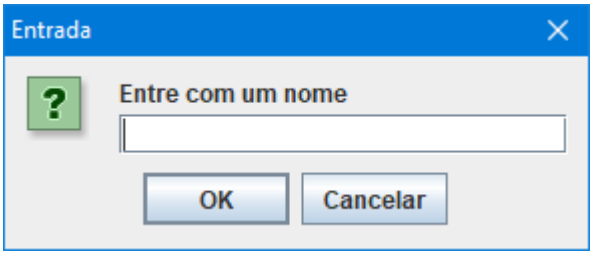


Exemplo de aplicação em um programa:

```
*JoptionPane.java
1  import javax.swing.JOptionPane;
2
3  public class JoptionPane {
4
5      public static void main(String[] args) {
6
7          String entrada; // variável de entrada
8          entrada = JOptionPane.showInputDialog("Entre com um nome");
9
10     }
11
12 }
13
```

Na linha 7 realizamos a declaração da variável **entrada** do tipo String que armazenará o conteúdo inserido pelo usuário.

Na linha 8 a variável **entrada** recebe o conteúdo do comando JOptionPane.showInputDialog(“Entre com um nome”);. Vamos analisar como esta linha se comporta. Este comando solicita ao usuário que digite algum dado para o sistema. Mas qual dado é este? No nosso exemplo o dado solicitado é o nome – Entre com um nome. O resultado para o usuário será:



Um resultado muito mais bonito esteticamente que o modo console.

Saída de dados

Para realizarmos a saída dos dados de um programa do Java também utilizaremos o comando **JOptionPane**, mas com a seguinte sintaxe:

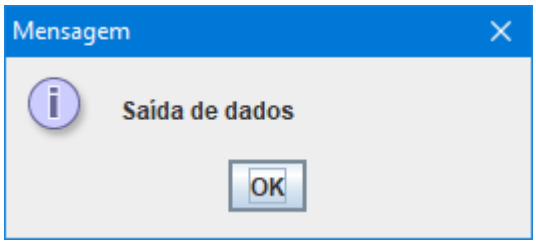
Figura 7 - Arquivo GEEaD

Exemplo de aplicação em um programa:

JoptionPaneShow.java

```
1  import javax.swing.JOptionPane;
2
3  public class JoptionPaneShow {
4
5      public static void main(String[] args) {
6          JOptionPane.showMessageDialog(null, "Saída de dados");
7
8      }
9
10 }
11
```

Na linha 6 o comando JOptionPane.showMessageDialog(null, “saída de dados”); faz a saída da mensagem para o usuário. O primeiro argumento sempre será null, seguido pela mensagem que queremos exibir para o usuário – Saída de dados. O resultado será:



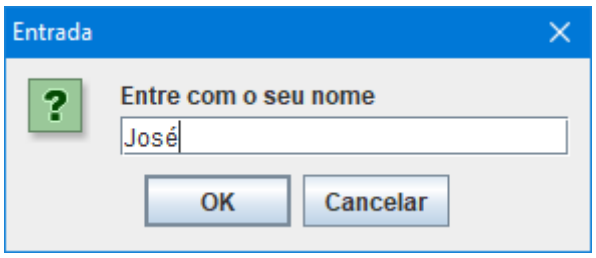
Agora sabendo como realizar a entrada e a saída de dados de forma gráfica, conseguimos fazer um programa que leia e exiba dados para o usuário em modo gráfico, como no exemplo a seguir:

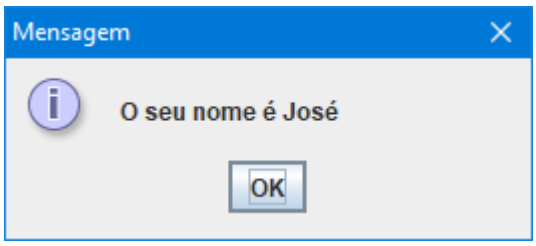
JOptionPane_E_S.java

```
1  import javax.swing.JOptionPane;
2
3  public class JOptionPane_E_S {
4
5      public static void main(String[] args) {
6
7          String nome; //Variável nome do tipo String
8
9          //entrada de dados
10         nome = JOptionPane.showInputDialog("Entre com o seu nome");
11
12         //saída de dados
13         JOptionPane.showMessageDialog(null, "O seu nome é " + nome);
14     }
15
16 }
17
```

Figura 8 - Arquivo GEEaD

Neste exemplo apresentado o programa exibe uma janela pedindo o nome do usuário na linha 10 e posteriormente exibe o nome digitado na linha 13. Resultado:





VOCÊ NO COMANDO

Pense em um programa que realize a leitura do nome do usuário, a idade e o telefone. Como o programador desenvolveria um programa simples usando os conceitos aprendidos até agora utilizando interface gráfica?

Conversão de tipos

O comando **JOptionPane.showInputDialog** sempre gera uma saída de dados do tipo String (sequência de caracteres alfanuméricos). Portanto se utilizarmos tipos de variáveis diferentes como, por exemplo, inteiro, devemos fazer a conversão de tipos antes de armazenarmos na sua variável correspondente conforme o exemplo a seguir:

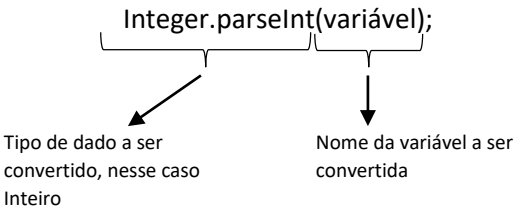
JoptionPaneCast.java

```
1  import javax.swing.JOptionPane;
2
3  public class JoptionPaneCast {
4
5      public static void main(String[] args) {
6          String auxiliar; // variável auxiliar
7          int numeroInteiro;
8          double numerodouble;
9          float numerofloat;
10
11         //entrada de dados salvando na variável auxiliar (String)
12         auxiliar = JOptionPane.showInputDialog("Entre com um número inteiro");
13
14         //conversão do tipo String para Inteiro - Integer.parseInt
15         numeroInteiro = Integer.parseInt(auxiliar);
16         numerodouble = Double.parseDouble(auxiliar);
17         numerofloat = Float.parseFloat(auxiliar);
18
19         //Mensagem de saída
20         JOptionPane.showMessageDialog(null, "O número inteiro é "+ numeroInteiro);
21         JOptionPane.showMessageDialog(null, "O número double é "+ numerodouble);
22         JOptionPane.showMessageDialog(null, "O número float é "+ numerofloat);
23     }
24
25 }
26
```

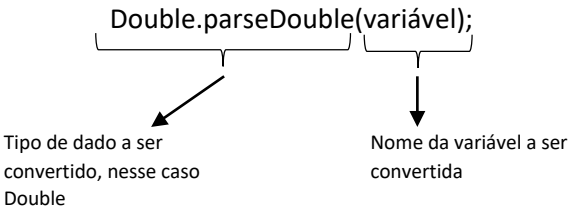
Um tipo nada mais é do que o conteúdo que uma variável consegue armazenar. Um tipo String pode armazenar caracteres e números. Um tipo int números inteiros e os tipos double e float números reais.

No exemplo da figura acima a entrada de dados feita na linha 12 é salva na variável **auxiliar** que é do tipo String. Porém estamos trabalhando com números, nesse exemplo, e um dado do tipo String armazena texto. Para isto devemos fazer a conversão de tipo (cast em inglês), ou seja, temos que realizar a conversão de um tipo de dado para outro. Ex: de String para int, de String para Double, etc. O Java possui comandos específicos para executar o cast.

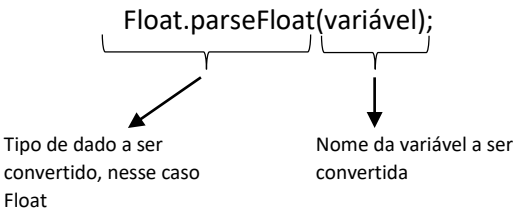
A sintaxe para a conversão de um tipo **String para Inteiro** é:



A sintaxe para a conversão de um tipo **String para Double** é:



A sintaxe para a conversão de um tipo **String para Float** é:



Sabendo a sintaxe do comando, na linha 15 a variável **numeroInteiro** recebe o resultado da conversão do valor da variável **auxiliar** de String para Inteiro. Assim como as linhas 16 e 17 realizam o cast para double e float, respectivamente.

Um aspecto interessante na linguagem Java é que no comando JOptionPane.showMessage Dialog, não precisamos realizar a conversão de tipos para o String. O Java automaticamente converte tipos numéricos para String antes de exibir a mensagem para o usuário.

Uma observação: existem outras formas de realizarmos a conversão de tipos e para mais tipos de dados, porém não serão apresentadas aqui nesse momento.

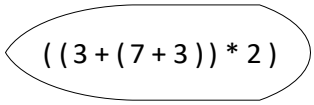
VOCÊ NO COMANDO

Pense em o que poderia acontecer se fizermos uma operação matemática qualquer, por exemplo uma adição, entre duas variáveis do tipo String sem que a tenhamos feito o cast para um tipo numérico.

1. Considerando o seu aprendizado sobre o comando escreva, e os operadores aritméticos, desenvolva o fluxograma, e as codificações em Java e pseudocódigo para exibir as mensagens que satisfaçam as seguintes situações:

As regras de precedência matemáticas funcionam da mesma forma na programação, com exceção dos parênteses ( ), colchetes [ ] e chaves { }, que são todos representador por parênteses.

a) Multiplicar por 2 o resultado de (7 + 3)+3.
Seguindo as regras de precedência matemática, de operadores aritméticos e o comando escreva, temos:

Fluxograma	Pseudocódigo	Java
	Escreva ((3+ (7+3))*2)	System.out.println((3+ (7+3)) * 2);

b) Dividir 168 por 46.

Fluxograma	Pseudocódigo	Java

c) Multiplicar o resto da divisão entre 7 e 4 por 2.

Fluxograma	Pseudocódigo	Java

d) Juntar as palavras “Técnico” com “Módulo” com “1”.

Fluxograma	Pseudocódigo	Java

e) Maria foi ao mercado e comprou 15 ovos para fazer bolos de chocolate para os seus netos. Sabendo que cada bolo utiliza-se 4 ovos e que maria já possuía 3 ovos em casa, quantos ovos sobrará para que ela possa fazer posteriormente uma omelete para almoçar?

Fluxograma	Pseudocódigo	Java

f) Exibir o resultado da seguinte conta: $[(7^3 + 2) + 5] \div 2$.

Fluxograma	Pseudocódigo	Java

2. Observando os seguintes problemas, identifique e declare as variáveis necessárias para a resolução dos mesmos. Faça a declaração em Fluxograma, Pseudocódigo e Java:

a. Martin é um professor de educação básica que ao final do bimestre passa horas calculando a média aritmética simples de seus alunos. Sabendo-se que em sua escola, a média é calculada com base em três notas sem pesos, quais serão as variáveis necessárias para automatizar o cálculo de médias feito por Martin?

Resolução:

Primeiramente precisamos descobrir como é feito o cálculo de uma Média Aritmética. Para este cálculo utilizamos a seguinte fórmula matemática:

Média = (soma das notas) / (quantidade de notas)

Sendo assim, observamos no problema que na escola onde Martin é professor, a média é composta por três notas. Neste caso, seguindo a definição de variável, precisaremos de três variáveis, uma para cada nota.

E a média? A média também deverá ser armazenada em uma variável, pois também não sabemos qual será seu valor após o cálculo das três notas, sendo necessário a alocação de um espaço na memória para armazena-la também.

Resposta final:

Fluxograma	Pseudocódigo	Java
nota1 como real nota2 como real nota3 como real media como real	declare nota1 como real nota2 como real nota3 como real media como real	double nota1; double nota2; double nota3; double media;

b. Hugo, um acionista financeiro individual, costuma investir constantemente em dólar em ações internacionais. Para que seja possível criar um conversor de moedas para Hugo, quais variáveis seriam necessárias?

Fluxograma	Pseudocódigo	Java

c. Para que seja possível verificar se um número é par o impar em um programa de computador, basta que o número em questão seja dividido por 2, armazenando o resto da divisão. Com base no resto da divisão verificamos se ele é par ou ímpar. Para que seja possível calcular o resto da divisão, quais são as variáveis necessárias?

Fluxograma	Pseudocódigo	Java

d. Quais são as variáveis necessárias para construir um programa que leia dois nomes e os exibam na ordem inversa da qual foi digitado?

Fluxograma	Pseudocódigo	Java

e. Anabilis, é uma cozinheira que vende marmitas para trabalhadores da construção civil em diversos empreendimentos. Para calcular a quantidade de marmitas que ela deve fazer por dia, ela multiplica por 5 o número de empreendimentos em que ela visitará, pois sabe que em média ela venderá 5 marmitas por empreendimento. Além disto ela costuma levar quatro marmitas extras para caso as vendas dela em algum empreendimento seja além do esperado. Sendo assim, quais são as variáveis necessárias para que Anabilis faça seu cálculo em um programa de computador?

Fluxograma	Pseudocódigo	Java

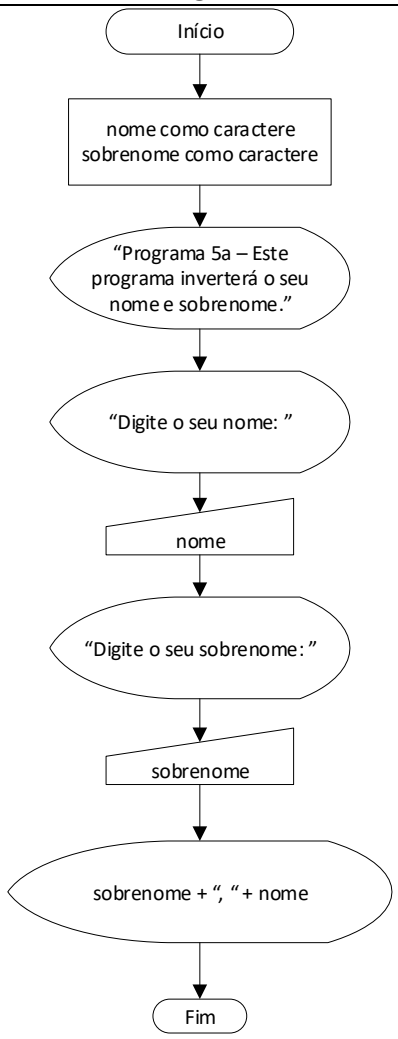
f. Um agricultor em sua pequena propriedade, cultiva feijões anualmente no período de chuvas. Ao final da colheita é feito o cálculo para descobrir qual foi o lucro na temporada em questão. O cálculo baseia-se na quantidade de sacos de feijão que foram colhidos, multiplicado com o valor de mercado para um saco no dia do cálculo. Do resultado da multiplicação, é subtraído o custo de produção total da temporada.
Quais variáveis devem ser utilizadas para que este cálculo seja feito em um programa?

Fluxograma	Pseudocódigo	Java

3. Crie um Fluxograma, Pseudocódigo e codificação Java que satisfaça a seguintes situações:

a. Um programa que leia o seu nome e sobrenome, e exiba-os na sua forma inversa (sobrenome, nome).

Resolução:

Fluxograma	Pseudocódigo
	Declare nome como caractere sobrenome como caractere Inicio Escreva(“Programa 5a – Este programa inverterá o seu nome e sobrenome”) Escreva(“Digite o seu nome: “) Leia(nome) Escreva(“Digite o seu sobrenome: “) Leia(sobrenome) Escreva(sobrenome + “, “ + nome); Fim

*Programa5a.java

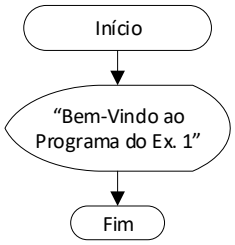
```java
import java.util.Scanner;

public class Programa5a {

    public static void main(String[] args) {
        //declaração de variáveis e inicialização do leitos
        String nome;
        String sobrenome;
        Scanner leitor = new Scanner(System.in);
        //inicio do programa
        System.out.println("Programa 5a - Este programa inverterá o seu nome e sobrenome");
        System.out.println("Digite o seu nome: ");
        nome = leitor.next();
        System.out.println("Digite o seu sobrenome");
        sobrenome = leitor.next();
        //saída de dados
        System.out.println(sobrenome + ", " + nome);
        //fim do programa.
    }
}

```

b. Um programa que leia 5 valores, some os 4 primeiros valores e divida o resultado pelo 5 valor.
c. Um programa que calcule a média aritmética simples, entre 4 notas.
d. Um programa que leia o seu ano de nascimento e mostre a sua idade. Lembre-se, para calcular a idade tendo-se apenas o ano de nascimento, subtraia o seu ano de nascimento do ano atual.
e. Um programa que calcule o resto da divisão entre 2 números.
f. Um programa que leia um nome e dois valores. Após a leitura dos valores, exiba o nome da pessoa e a divisão entre os dois valores.
g. Um programa que calcule a soma entre 3 valores digitados, após calcular a soma, o usuário digitará um quarto valor. O programa multiplicará a soma pelo quarto valor digitado e o exibirá.

4. Faça um programa que exiba uma mensagem de boas vidas ao usuário.
5. Elabore um programa que peça para o usuário digitar o seu nome e posteriormente exiba o nome digitado em uma outra janela.
6. Faça um programa que leia dois números inteiros, realize a soma de deste e exiba o resultado final.
7. Crie um programa que leia o código de um produto (número inteiro), o nome do produto e por final o seu preço. Ao final o programa deve exibir em uma única janela o código do produto, o nome e o preço.
8. Repita o exercício anterior utilizando dois números do tipo double.
9. Crie um programa que faça a leitura de um número inteiro, um número do tipo float e exiba a soma destes dois números para o usuário.

Respostas:

4.

Pseudocódigo	Fluxograma
Programa ex1 Início Escreva (“Bem-Vindo ao Programa do Ex. 1”) Fim.	

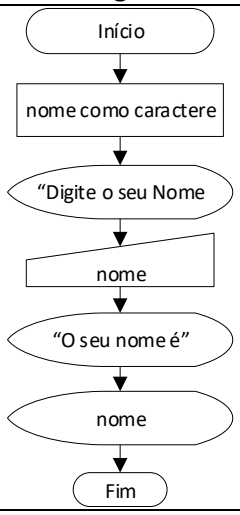
Java

```
*JOptionPane_ex1.java
1  import javax.swing.JOptionPane;
2
3  public class JOptionPane_ex1 {
4
5      public static void main(String[] args) {
6          //exercício 1
7
8          JOptionPane.showMessageDialog(null, "Bem-Vindo ao Programa do Exercício 1");
9
10     }
11
12 }
13
```

Figura 9- Arquivo GEEaD

5.

Pseudocódigo	Fluxograma

Programa ex2
 	
Declare
  nome como caractere
 	
Início
  Escreva (“Digite o seu nome”)
  Leia (nome)
  Escreva (“O seu nome é ”)
  Escreva (nome)
Fim.
	

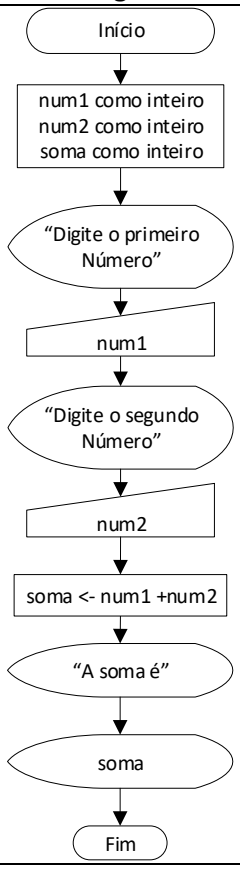
Java

JOptionPane_ex1.java | *JOptionPane_ex2.java

```
1  import javax.swing.JOptionPane;
2
3  public class JOptionPane_ex2 {
4
5      public static void main(String[] args) {
6          //exercício 2
7
8          //declaração de variáveis
9          String nome;
10
11         //entrada de dados
12         nome = JOptionPane.showInputDialog("Digite o seu nome");
13
14         //saída de dados
15         JOptionPane.showMessageDialog(null, "O seu nome é " + nome);
16
17     }
18
19 }
20
```

Figura 10- Arquivo GEEaD

6.

Pseudocódigo	Fluxograma
Programa ex4 Declare num1 como inteiro num2 como inteiro soma como inteiro Início Escreva (“Digite primeiro número”) Leia (num1) Escreva (“Digite segundo número”) Leia (num2) Soma <- num1 + num2 Escreva (“A soma é”) Escreva (soma) Fim.	

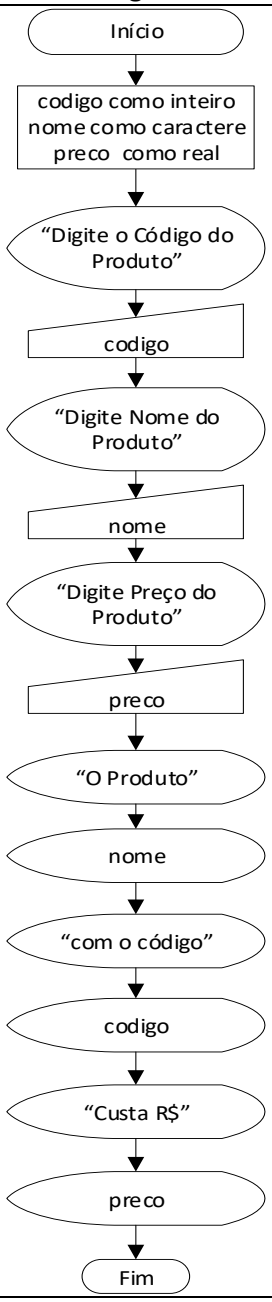
Java

JOptionPane_ex4.java

```java
import javax.swing.JOptionPane;

public class JOptionPane_ex4 {

    public static void main(String[] args) {
        //exercício 4

        //declaração de variáveis
        int num1, num2, soma;
        String aux; // variável aux é auxiliar

        //entrada de dados
        aux = JOptionPane.showInputDialog("Digite o primeiro número");
        num1 = Integer.parseInt(aux); // faz a conversão de tipo para inteiro

        num2 = Integer.parseInt(JOptionPane.showInputDialog("Digite o segundo número"));

        //processamento
        soma = num1 + num2;

        //saída de dados
        JOptionPane.showMessageDialog(null, " A soma é " + soma);

    }

}
```

Figura 11- Arquivo GEEaD

7.

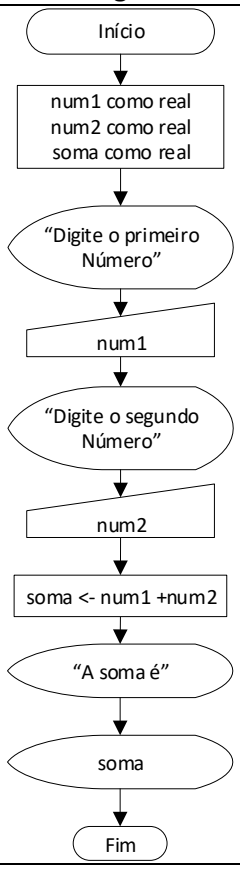
Pseudocódigo	Fluxograma
Programa ex3 Declare codigo como inteiro nome como caractere preco como real Início Escreva (“Digite Código do produto”) Leia (codigo) Escreva (“Digite Nome do produto”) Leia (nome) Escreva (“Digite Preço do produto”) Leia (preco) Escreva (“O Produto ”) Escreva (nome) Escreva (“com o código ”) Escreva (codigo) Escreva (“Custa R$ ”) Escreva (nome) Fim.	

Java

JOptionPane_ex3.java

```java
import javax.swing.JOptionPane;

public class JOptionPane_ex3 {

    public static void main(String[] args) {
        //exercício 3

        //declaração de variáveis
        int codigo;
        String nome, aux; // variável aux é auxiliar
        double preco;

        //entrada de dados
        aux = JOptionPane.showInputDialog("Digite o código do produto");
        codigo = Integer.parseInt(aux); // faz a conversão de tipo para inteiro

        nome = JOptionPane.showInputDialog("Digite o seu nome");

        aux = JOptionPane.showInputDialog("Digite o Preço do Produto");
        preco = Double.parseDouble(aux); // faz a conversão de tipo para double

        //saída de dados
        JOptionPane.showMessageDialog(null, " O Produto " + nome +
                "\n com o código " + codigo +
                "\n Custa R$" + preco);
        //faz a saída de dados em uma única janela. o comando \n pula uma linha

    }
}
```

8.

Pseudocódigo	Fluxograma
Programa ex5 Declare num1 como real num2 como real soma como real Início Escreva (“Digite primeiro número”) Leia (num1) Escreva (“Digite segundo número”) Leia (num2) Soma <- num1 + num2 Escreva (“A soma é”) Escreva (soma) Fim.	

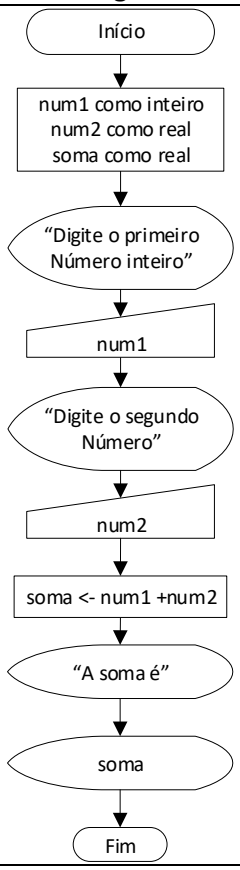
Java

JOptionPane_ex5.java

```
1  import javax.swing.JOptionPane;
2
3  public class JOptionPane_ex5 {
4
5      public static void main(String[] args) {
6          //exercício 5
7
8          //declaração de variáveis
9          double num1, num2, soma;
10         String aux; // variável aux é auxiliar
11
12         //entrada de dados
13         aux = JOptionPane.showInputDialog("Digite o primeiro número");
14         num1 = Double.parseDouble(aux); // faz a conversão de tipo para inteiro
15
16         num2 = Double.parseDouble(JOptionPane.showInputDialog("Digite o segundo número"));
17
18         //processamento
19         soma = num1 + num2;
20
21         //saída de dados
22         JOptionPane.showMessageDialog(null, " A soma é " + soma);
23     }
24
25 }
26
```

Figura 12- Arquivo GEEaD

9.

Pseudocódigo	Fluxograma
Programa ex6 Declare num1 como inteiro num2 como real soma como real Início Escreva (“Digite primeiro número inteiro”) Leia (num1) Escreva (“Digite segundo número”) Leia (num2) Soma <- num1 + num2 Escreva (“A soma é”) Escreva (soma) Fim.	

Java

JOptionPane_ex5.java | JOptionPane_ex6.java

```
1  import javax.swing.JOptionPane;
2
3  public class JOptionPane_ex6 {
4
5      public static void main(String[] args) {
6          //exercício 6
7
8          //declaração de variáveis
9          int num1;
10         float num2, soma;
11
12
13         //entrada de dados
14         num1 = Integer.parseInt(JOptionPane.showInputDialog("Digite o primeiro número Inteiro"));
15
16         num2 = Float.parseFloat(JOptionPane.showInputDialog("Digite o segundo número"));
17
18         //processamento
19         soma = num1 + num2;
20
21         //saída de dados
22         JOptionPane.showMessageDialog(null, " A soma é " + soma);
23
24     }
25
26 }
27
```

ATIVIDADE ONLINE

Exercício 1

Elabore o fluxograma, pseudocódigo e codificação Java que satisfaça as seguintes situações:

a) Ler dois números e efetuar as quatro operações matemáticas básicas (soma, subtração, multiplicação e divisão) exibindo o resultado de cada operação individualmente.

b) Ler um nome, serie, telefone e a média de um aluno. Após a leitura exibir os dados no seguinte formato:
---------- Técnico em Informática EAD ----------
Nome: <nome do aluno> Telefone: <telefone do aluno>
Série: <série do aluno>

Média: <média do aluno>

Cada linha em Java é um comando println.

Não deixe também de assistir ao vídeo:

Informática – Módulo I – Agenda 11 – Comando ESCREVA, comando LEIA

Link: https://www.youtube.com/watch?v=cD-GfbyVCM4. Acessado em 14/12/2017.

Para aprofundamento dos temas discutidos nesta aula, seguem abaixo algumas dicas de filmes, livros e artigos que se relacionam com o conteúdo estudado. Estas dicas são muito importantes para você!

Vídeo:

Procure no YouTube uma Palylist denominada ;"Curso de Java para Iniciantes – Grátis, Completo e com Certificado " e assista as vídeo aulas de 1 a 8 disponível em : https://www.youtube.com/playlist?list=PLHz_AreHm4dkI2ZdjTwZA4mPMxWTfNSpR. Acessado em 14/12/2017

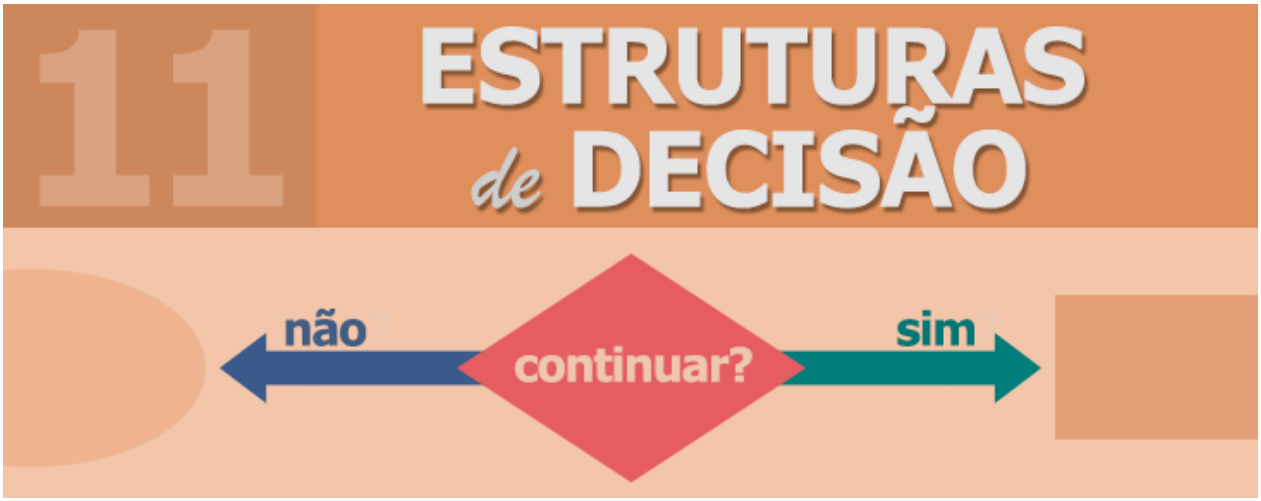
Livros:

Centro Paula Souza. (2010). Informática, programação de computadores (Vol. 4). São Paulo: Fundação Padre Anchieta.

Deitel, P., & Deitel, H. (2010). Java - Como Programar. São Paulo, SP: Pearson Prentice Hall.

Puga, S., & Rissetti, G. (2009). Lógica de programação e estrutura de dados com aplicações em Java. São Paulo: Pearson Prentice Hall.

Schildt, H. (2015). Java para iniciantes. Porto Alegre: Bookman.



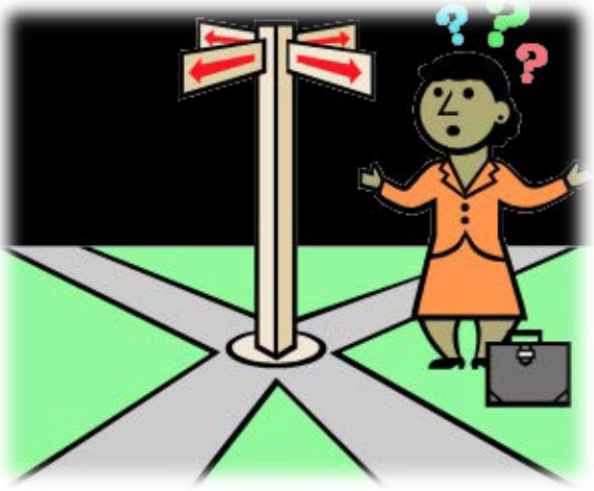
AGENDA 11

ESTRUTURAS DE DECISÃO "SE...SENÃO...FIM-SE"

MOMENTO DE REFLEXÃO

Há momentos na vida em que precisamos tomar algumas decisões, não é? E sabemos que dependendo da decisão que tomamos seguimos para um caminho ou outro na vida. Por exemplo: se você prestar um concurso para um emprego e obtiver nota suficiente para ser aprovado, provavelmente, se decidir por assumir este emprego, logo a sua vida tomará outro rumo. A partir daquele momento, você mudará de emprego e seguirá outra rotina de trabalho, com novos desafios, novos colegas e novo ambiente de trabalho.

Imagem 18



A partir de uma tomada de decisão optamos por fazer ou não fazer determinados procedimentos ou funções em nossas vidas. Bem, assim também acontece quando fazemos um programa de computador ou mais recentemente aplicativos para telefones celulares inteligentes, os chamados apps. Pode existir determinados momentos, dentro de um programa, que sejam necessários executar diferentes procedimentos de acordo com as condições estabelecidas. O assunto deste capítulo é exatamente este: saber utilizar as Estruturas de Decisão em um programa de computador. Essas estruturas permitem que o programa execute diferentes tipos de procedimentos baseados em uma determinada decisão. Vamos estudá-las?

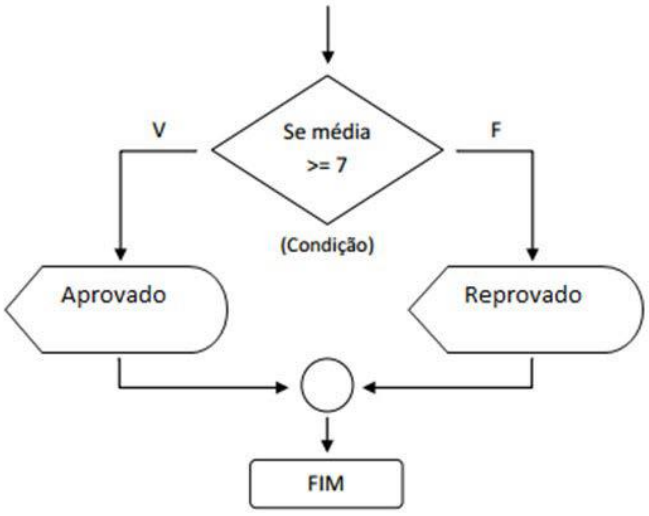
POR QUE APRENDER?

Um programa é uma lista de sequência de comandos que são executados sequencialmente. A menos que seja solicitado o contrário, um programa executa as instruções do início de um programa até o fim, mas, conforme aumenta a complexidade, eles precisam tomar decisões e alterar este processamento sequencial. É por este motivo que você estudará nesta agenda as noções de estruturas de dados Se..Senão..Fim-se.

PARA COMEÇAR O ASSUNTO...

Imagine a seguinte situação: Um programa que calcule a média escolar de um aluno e apresente o resultado na tela. Até aqui, sem problemas! Mas imagine que este programa também deva escrever na tela se o aluno está aprovado ou reprovado de acordo com a média calculada. Neste caso, em determinado momento será necessário que o programa verifique a média calculada e tome a decisão de escrever na tela: Aprovado ou Reprovado.

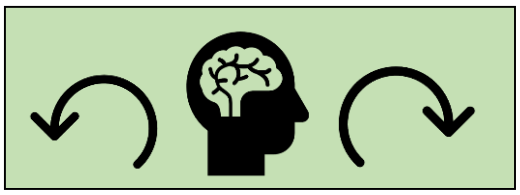
A estrutura de decisão tem a finalidade de tomar uma decisão. Veja o fluxograma desta estrutura:



Se a média for maior ou igual a 7 (se a condição for verdadeira), o aluno foi aprovado, senão, (se a condição for falsa), o aluno foi reprovado. Entendeu? Pronto, para mergulhar no tema deste capítulo?

MERGULHANDO NO TEMA...

Imagem 19



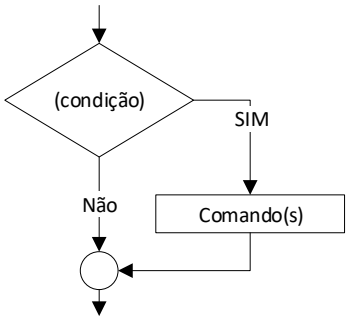
Estruturas de Decisão

Até agora trabalhamos somente com programas que realizavam tarefas simples como a realização de entrada e saída de dados e pequenos cálculos matemáticos. Contudo os algoritmos criados até aqui não possuem poder de decisão. Ou seja, eles sempre executam as mesmas tarefas independentemente dos resultados obtidos. Infelizmente, na vida real, temos que tomar decisões que muitas vezes são difíceis e que podem alterar o rumo de nossas vidas. Com os programas ocorre a mesma coisa.

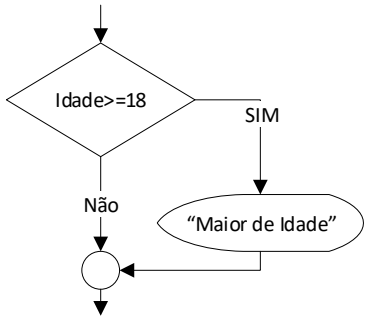
Em programação chamamos essas decisões de Estrutura de Decisão. No Java, chamamos de comando IF.

Estruturas de decisão IF (Se...Fim)

As estruturas de decisão ou testes condicionais nos permite que executemos um conjunto de comandos diferente dependendo do resultado de um teste utilizando operadores relacionais. Este resultado pode ser verdadeiro ou falso conforme podemos visualizar na tabela a seguir:

Pseudocódigo	Fluxograma	Java
SE (condição) **Então** {comando(s)} **Fim-Se**		**if** (condição)**{** {comando(s)}**;** **}**

Para um entendimento melhor é mais fácil e prático colocar um exemplo de aplicação: vamos imaginar que em um determinado trecho de um programa precisamos tomar uma decisão para saber se uma pessoa é maior de idade. O programa codificado é apresentado a seguir:

Pseudocódigo	Fluxograma	Java
SE (idade >=18) **Então** Escreva ("Maior de idade") **Fim-Se**		**if** (idade >=18)**{** JOptionPane.showMessageDialog(null, "Maior de idade"); **}**

O comando condicional **SE** testa a condição **idade >=18**, ou seja, se a idade for maior ou igual a 18 anos **Então**, condição sim ou verdadeira será executado o comando **Escreva ("Maior de idade")**, caso contrário nada será feito (**Fim-Se**). Se observarmos o fluxograma fica mais fácil de compreender o que ocorre.

Na programação Java conforme a sintaxe apresentada e tomando como base o exemplo da tabela acima a palavra **SE** é substituída pelo **IF**, a comparação permanece a mesma, a palavra **Então** é substituída pela **{**, o comando E**screva** é substituído pelo **JOptionPane.showMessageDialog** e finalmente o **Fim-Se** é substituído pelo **}**.

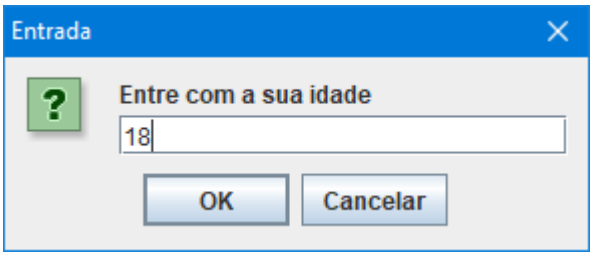
Realizando a codificação em Java de um programa completo teremos:

IfSimples.java

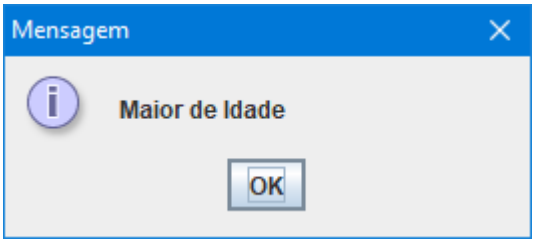
```
1  import javax.swing.JOptionPane;
2
3  public class IfSimples {
4
5      public static void main(String[] args) {
6          //declaracao de variáveis
7          int idade; // armazena a idade
8          String aux; //variável auxiliar
9
10         //entrada de dados
11         aux = JOptionPane.showInputDialog("Entre com a sua idade");
12         //conversão de tipos
13         idade = Integer.parseInt(aux);
14
15         //Decisão
16         if (idade >=18) {
17             JOptionPane.showMessageDialog(null, "Maior de Idade");
18         }//fim do if
19     }//fim do main
20
21 }//fim da classe
22
```

Explicando a Estrutura de decisão presente na linha 16. O comando de decisão if (idade >=18) irá executar o que estiver dentro das chaves até a linha 18, se e somente se o valor da idade for

maior ou igual a 18. Caso contrário não executará nenhum comando adicional e o programa será encerrado.
Se ao executarmos o programa, digitarmos a idade de 18 anos, por exemplo:



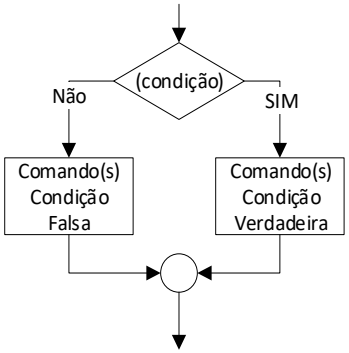
Teremos o seguinte retorno:



Note que se a condição for falsa (Não) nenhum comando é executado.

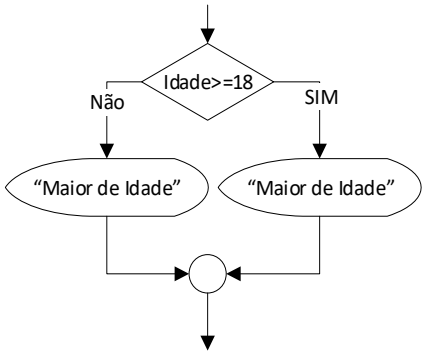
Estrutura de decisão IF-else (se...Senão...Fimse)

Acabamos de ver uma estrutura de decisão que somente realiza uma ação distinta caso a o teste condicional seja verdadeiro. Contudo, em geral, necessitamos que alguma ação seja tomada também caso o teste condicional seja falso. Para isso temos o comando If-Else:

Pseudocódigo	Fluxograma	Java
SE (condição) **Então** {comando(s) condição verdadeira} **Senão** {Comando(s) condição falsa} **Fim-Se**		**if** (condição)**{** {comando(s) condição verdadeira**};** **}** **else {** {comando(s) condição falsa}; **}**

Observando a tabela acima notamos que caso o teste lógico condicional falhe temos um comando ou grupo de comandos a serem executados (comandos após o **Senão**). Vamos observar o exemplo

a seguir que consiste no mesmo exemplo anterior, porém com comandos sendo executados caso o teste condicional seja falso.

Pseudocódigo	Fluxograma	Java
SE (idade >=18) **Então** Escreva (“Maior de idade”) **Senão** Escreva (“Menor de idade”) **Fim-Se**		**if** (idade >=18)**{** JOptionPane.showMessageDialog(null, “Maior de idade”)**;** **}** **else {** JOptionPane.showMessageDialog(null, “Menor de idade”)**;** **}**

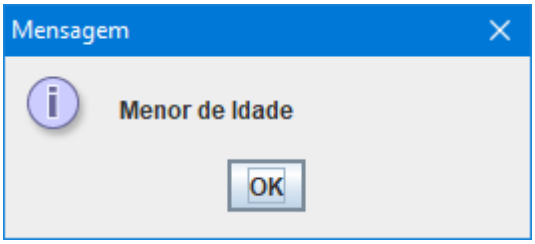
Observamos que é praticamente idêntico ao exemplo anterior, porém caso o teste condicional falhe (resultado falso – não) executamos um comando que exibe a mensagem “menor de idade” para o usuário. Isso é feito através da utilização da cláusula **Senão** no Pseudocódigo e **else** no Java.

Programa completo codificado em Java:

*ifComposto.java

```
1  import javax.swing.JOptionPane;
2
3  public class ifComposto {
4
5      public static void main(String[] args) {
6          //declaraçao de variáveis
7          int idade; // armazena a idade
8          String aux; //variável auxiliar
9
10         //entrada de dados
11         aux = JOptionPane.showInputDialog("Entre com a sua idade");
12         //conversão de tipos
13         idade = Integer.parseInt(aux);
14
15         //Decisão
16         if (idade >=18) {
17             //comandos se a condição for verdadeira
18             JOptionPane.showMessageDialog(null, "Maior de Idade");
19         } else {
20             //comandos se a condição for falsa
21             JOptionPane.showMessageDialog(null, "Menor de Idade");
22         } // fim do if
23     }//fim do main
24
25 }//fim da classe
26
```

O resultado se digitarmos 17 na caixa de diálogo que pede a idade será:



Estrutura de decisão aninhada

O que aconteceria se precisássemos de mais de duas alternativas para resolver um problema?

Outro fato muito comum de se acontecer é a necessidade de utilizarmos mais de uma estrutura de decisão simultaneamente. Pode-se notar que em uma estrutura de decisão podemos somente ter duas saídas: verdadeiro e falso. Mas e quando necessitamos de uma saída com mais de duas alternativas?

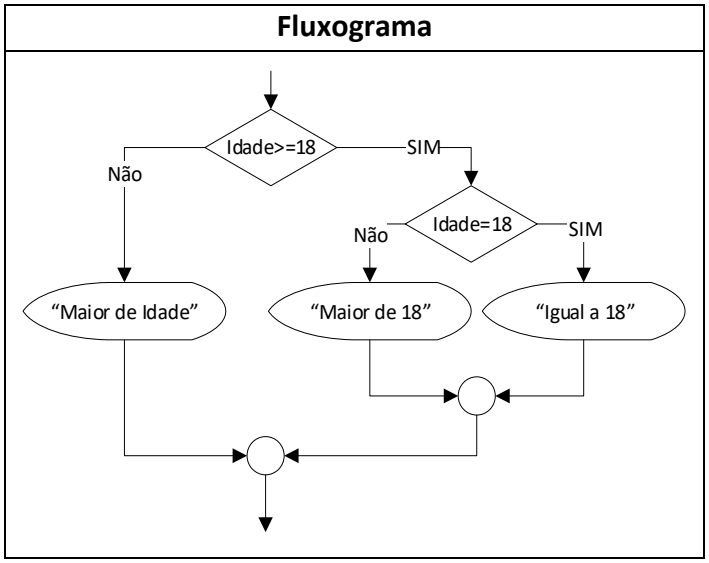
Para isso usamos as estruturas de decisão aninhadas que consistem em utilizar um comando SE encadeado no interior de outro. Parece confuso à primeira vista, mas explicando com um exemplo o entendimento será rápido. Partindo do mesmo exemplo de classificação de idade anterior. Podíamos classificar somente em maior de idade e menor de idade. Mas e se quiséssemos classificar também como igual a 18 anos? A codificação do programa ficaria da seguinte forma:

Pseudocódigo	Java

SE (idade >=18) **Então**
  SE (idade = 18) **Então**
  Escreva (“Igual a 18”)
  Senão
  Escreva (“Maior de 18”)
  Fim-Se
Senão
  Escreva (“Menor de idade”)
Fim-Se
	if (idade >=18)**{**
	  if (idade==18)**{**
	  JOptionPane.showMessageDialog(null, “igual a 18”)**;**
	  }
	else {
	  JOptionPane.showMessageDialog(null, “Maior de 18”);
	  }
	}
	else {
	  JOptionPane.showMessageDialog(null, “Menor de idade”)**;**
	}

Note que ao executar a primeira tomada de decisão **se (idade>=18)** em caso verdadeiro sabe-se somente que a idade é maior ou igual a 18. Para saber se a idade é igual a 18, é necessária a execução de outra estrutura de decisão **se (idade=18)**. Em caso afirmativo sabemos que é igual e

em caso negativo sabemos que é maior de 18 anos (maior de 18). Isso que chamamos de estrutura de decisão aninhada.
Achou que faltou algo? O fluxograma? Lá vem ele logo a seguir:



Código em Java:

*ifAninhado.java

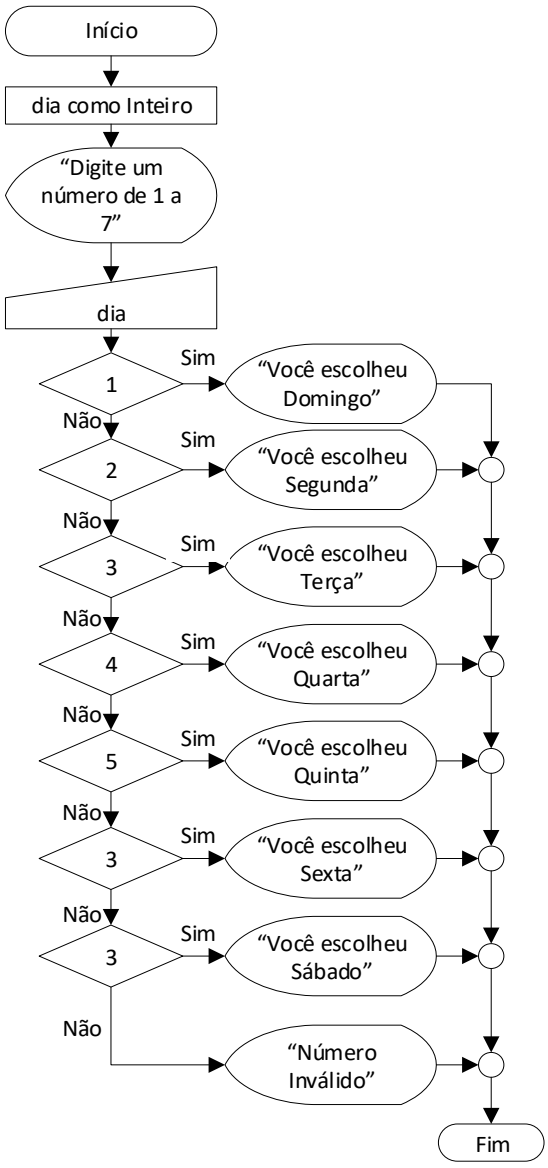
```
1  import javax.swing.JOptionPane;
2
3  public class ifAninhado {
4
5      public static void main(String[] args) {
6          //declaraçao de variáveis
7          int idade; // armazena a idade
8          String aux; //variável auxiliar
9
10         //entrada de dados
11         aux = JOptionPane.showInputDialog("Entre com a sua idade");
12         //conversão de tipos
13         idade = Integer.parseInt(aux);
14
15         //Decisão
16         if (idade >=18) { // primeiro if
17             //comandos se a condição for verdadeira
18             if (idade == 18) { // segundo if
19                 JOptionPane.showMessageDialog(null, "igual a 18");
20             }else {
21             JOptionPane.showMessageDialog(null, "Maior de 18");
22             }//fim do segundo if
23         } else {
24             //comandos se a condição for falsa
25             JOptionPane.showMessageDialog(null, "Menor de Idade");
26         } // fim do primeiro if
27     }//fim do main
28
29 }//fim da classe
30
```

Um exemplo de exercício com programa completo: suponha que precisamos fazer um programa para o usuário inserir um número de 1 a 7 para que este exiba qual é o dia da semana

correspondente. Domingo é o início da semana e, portanto, o número 1 e assim sucessivamente. Como resolveríamos esse programa?

Primeiramente devemos pensar nas variáveis necessárias. Bom, tem-se que inserir um número de 1 a 7, logo precisamos de uma variável que vamos chamar de entrada. Qual seria o tipo da variável entrada? Acertou se você respondeu inteiro, pois ela irá armazenar somente números inteiros. Mais alguma variável é necessária? A resposta é não, porque vamos fazer a saída diretamente imprimindo na tela.

Vamos pensar no fluxograma primeiro pois é mais fácil de entender e visualizar:



Feito o fluxograma, podemos escrever o Pseudocódigo:

```
Programa Semana
Declare
        dia como inteiro
Início
        Escreva(“Digite um Número de 1 a 7”)
        Leia(dia)
        Se (dia = 1) Então
                Escreva (“Você escolheu domingo”)
        senão
                Se (dia = 2) Então
                        Escreva (“Você escolheu segunda”)
                senão
                        Se (dia = 3) Então
                                Escreva (“Você escolheu terça”)
                        senão
                                Se (dia = 4) Então
                                        Escreva (“Você escolheu quarta”)
                                senão
                                        Se (dia = 5) Então
                                                Escreva (“Você escolheu quinta”)
                                        senão
                                                Se (dia = 6) Então
                                                        Escreva (“Você escolheu sexta”)
                                                senão
                                                        Se (dia = 7) Então
                                                                Escreva (“Você escolheu sábado”)
                                                        senão
                                                                Escreva(“Número Inválido”)
                                                        Fim-Se
                                                Fim-Se
                                        Fim-Se
                                Fim-Se
                        Fim-se
                Fim-Se
        Fim-Se
Fim.
```

E por final a codificação em Java:

ifAninhadoSemana.java

```
1  import javax.swing.JOptionPane;
2
3  public class ifAninhadoSemana {
4
5      public static void main(String[] args) {
6          //declaração de variáveis
7          int dia; // variável para armazenamento da semana
8
9          //entrada de dados com conversão de tipos juntas
10         dia = Integer.parseInt(JOptionPane.showInputDialog("Entre com um número de 1 a 7"));
11
12         //processamento
13
14         if (dia == 1) { //if 1
15             JOptionPane.showMessageDialog(null, "Você escolheu Domingo");
16         } else {
17             if (dia == 2) { //if 2
18                 JOptionPane.showMessageDialog(null, "Você escolheu Segunda");
19             } else {
20                 if (dia == 3) { //if 3
21                     JOptionPane.showMessageDialog(null, "Você escolheu Terça");
22                 } else {
23                     if (dia == 4) { //if 4
24                         JOptionPane.showMessageDialog(null, "Você escolheu Quarta");
25                     } else {
26                         if (dia == 5) { //if 5
27                             JOptionPane.showMessageDialog(null, "Você escolheu Quinta");
28                         } else {
29                             if (dia == 6) { //if 6
30                                 JOptionPane.showMessageDialog(null, "Você escolheu Sexta");
31                             } else {
32                                 if (dia == 7) { //if 7
33                                     JOptionPane.showMessageDialog(null, "Você escolheu Sábado");
34                                 } else {
35                                     JOptionPane.showMessageDialog(null, "Número Inválido");
36                                 } // fim do if 7
37                             } // fim do if 6
38                         } // fim do if 5
39                     } // fim do if 4
40                 } // fim do if 3
41             } // fim do if 2
42         } // fim do if 1
43     } // fim do método main
44
45 } // fim da classe
46
```

VOCÊ NO COMANDO

Observando os exemplos apresentados sobre estruturas de decisão aninhadas, existe alguma relação entre o número de alternativas e o número necessário de comparações a serem efetuadas? Reflita sobre o assunto.

Sim, existe uma relação. O número de comparações necessárias em um programa é o número de alternativas menos uma unidade. Ou seja, se temos 6 alternativas, teremos 5 comparações em nosso programa.

Como realizar comparações com String no Java

Já pensou como poderíamos realizar comparações com uma sequência de caracteres em Java? Seria a mesma forma que realizamos com tipos numéricos?

Para realizarmos uma comparação de um conteúdo de uma variável com uma String – sequência de caracteres - no Java temos que utilizar um método especial o **.equals()**.

Mas como utilizamos o **.equals()**? Vejamos o exemplo abaixo:

*IfEquals.java

```
1  import javax.swing.JOptionPane;
2
3  public class IfEquals {
4
5      public static void main(String[] args) {
6          //declaração de variáveis
7          String nome;
8
9          //entrada de dados
10         nome = JOptionPane.showInputDialog("Entre com um nome");
11
12         //Processamento e saída
13         if (nome.equals("Jose")) {
14             JOptionPane.showMessageDialog(null, "O Nome Digitado é Jose");
15         } else {
16             JOptionPane.showMessageDialog(null, "O nome digitado foi " + nome);
17         }
18
19     }
20
21 }
22
```

Note que na linha 13 utilizamos a expressão **nome.equals(“Jose”)**, ou seja, estamos comparando se o valor armazenado na variável **nome** é igual a Jose. Se o usuário digitar qualquer outro nome, ou inclusive jose ou José o resultado da comparação é falso.
Então o **.equals()** é utilizado como: <nomedavariável>**.equals(**valor a comparar**)**.

VOCÊ NO COMANDO

Como poderíamos elaborar um programa que faça a leitura de um nome de usuário e uma senha e libere o acesso ao aplicativo somente se ambos estiverem corretos?

LoginSenha.java

```java
import javax.swing.JOptionPane;

public class LoginSenha {

    public static void main(String[] args) {
        // Login e senha
        // login = aluno
        // senha = aluno

        //declaração de variáveis
        String login, senha; // variáveis para armazenar o login e senha

        //entrada de dados
        login = JOptionPane.showInputDialog("Entre com o Login");
        senha = JOptionPane.showInputDialog("Entre com a senha");

        if( login.equals("aluno") && senha.equals("aluno")) {
            JOptionPane.showMessageDialog(null, "Acesso liberado");
        }else {
            JOptionPane.showMessageDialog(null, "Login ou senha incorretos");
        } // fim do if

    }//fim do main
} // fim da classe
```

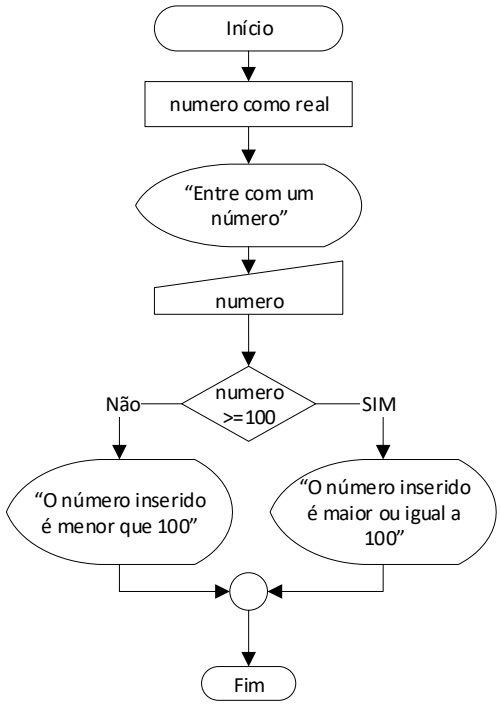
Elabore um programa em Java que leia um número e compare se ele é maior ou igual a 100.

1. Faça um programa que leia dois números do tipo Double, realize a sua soma e informe se ele é menor que 10.000.
2. Elabore um programa que leia um número e o classifique como par ou ímpar.
   Obs.: para utilizar o resultado do resto da divisão utilize o comando **mod <numero>**.
   Ex.: 10 mod 3. O resultado será 1.
   No Java basta substituir o “**/**” (divisão) por “**%**” (resto da divisão). Ex.: 10 % 3.
3. Escreva um programa que leia um nome de um cliente e o valor da compra dela e caso a compra seja superior a R$1.000,00 retorne uma mensagem contendo o nome do cliente e o valor da compra e informe que ele é um cliente VIP e ganhou um desconto de 10% sobre o valor da próxima compra
4. Gustavo deseja determinar se um número está compreendido entre 10 e 100. Escreva um programa que realize a leitura de um número inteiro e informe a Gustavo se este está compreendido entre 10 e 100

5. Desenvolva um programa que verifique se um número é divisível simultaneamente por 2 e por 5 e exiba a mensagem correspondente para o usuário ao final da execução deste.

Respostas:

1.

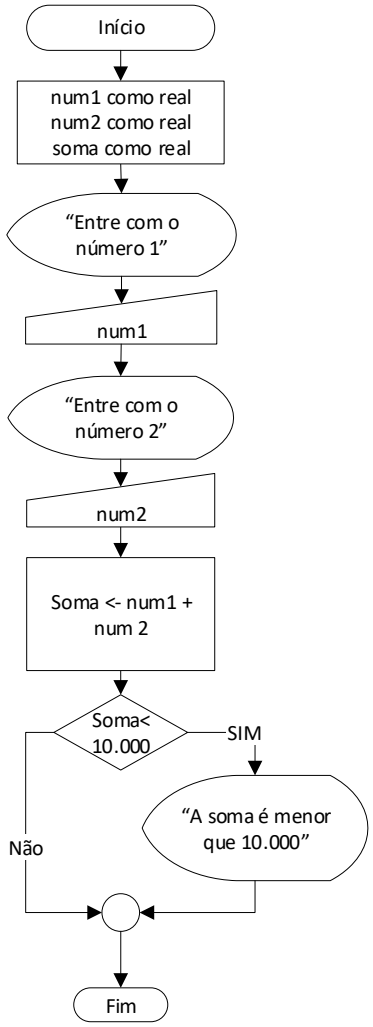
Pseudocódigo	Fluxograma
Programa ex1 Declare numero como real Início Escreva (“Entre com um número”) Leia (numero) Se (numero>=100) Então Escreva (“ O número inserido é maior ou igual a 100”) Senão Escreva (“O número inserido é menor que100”) Fim-Se Fim.	

Java

JOptionPane_ex5.java | JOptionPane_ex6.java | if_Ex1.java

```
1  import javax.swing.JOptionPane;
2
3  public class if_Ex1 {
4
5      public static void main(String[] args) {
6          // exercício 1
7
8          //declaração de variáveis
9          double numero;
10         String aux;
11
12         //entrada de dados
13         aux =JOptionPane.showInputDialog("Entre com um número");
14         numero = Double.parseDouble(aux);
15
16         //processamento e saída
17         if (numero >=100) {
18             JOptionPane.showMessageDialog(null, "O número inserido é maior ou igual que 100");
19         } else {
20             JOptionPane.showMessageDialog(null, "O número inserido é menor que 100");
21         }
22
23     }
24
25 }
26
```

2.

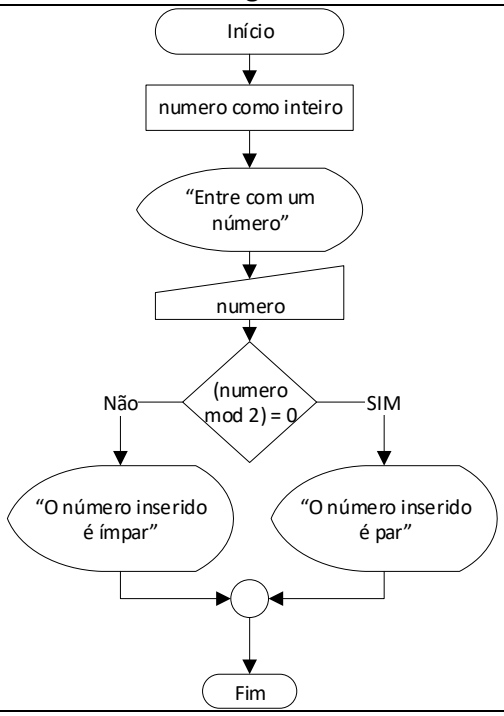
Pseudocódigo	Fluxograma
Programa ex2 Declare num1 como real num2 como real soma como real Início Escreva (“Entre com o número 1”) Leia (num1) Escreva (“Entre com o número 2”) Leia (num2) soma <- num1 + num2 Se (soma<10000) Então Escreva (“ A soma é menor que 10000”) Fim-Se Fim.	

Java

*if_Ex2.java

```
1  import javax.swing.JOptionPane;
2
3  public class if_Ex2 {
4
5      public static void main(String[] args) {
6          // exercício 2
7
8          //declaração de variáveis
9          double num1, num2,soma;
10         String aux;
11
12         //entrada de dados
13         aux =JOptionPane.showInputDialog("Entre com o número 1");
14         num1 = Double.parseDouble(aux);
15
16         aux =JOptionPane.showInputDialog("Entre com o número 2");
17         num2 = Double.parseDouble(aux);
18
19         //processamento e saída
20         soma = num1 +num2;
21         if (soma < 10000) {
22             JOptionPane.showMessageDialog(null, "A soma é menor que 10000");
23         }
24     }
25
26 }
27
```

3.

Pseudocódigo	Fluxograma
Programa ex3 Declare numero como inteiro Início Escreva (“Entre com um número”) Leia (numero) Se ((numero mod 2)=0) Então Escreva (“ O número inserido é par”) Senão Escreva (“O número inserido é ímpar”) Fim-Se Fim.	

Java

if_Ex3.java

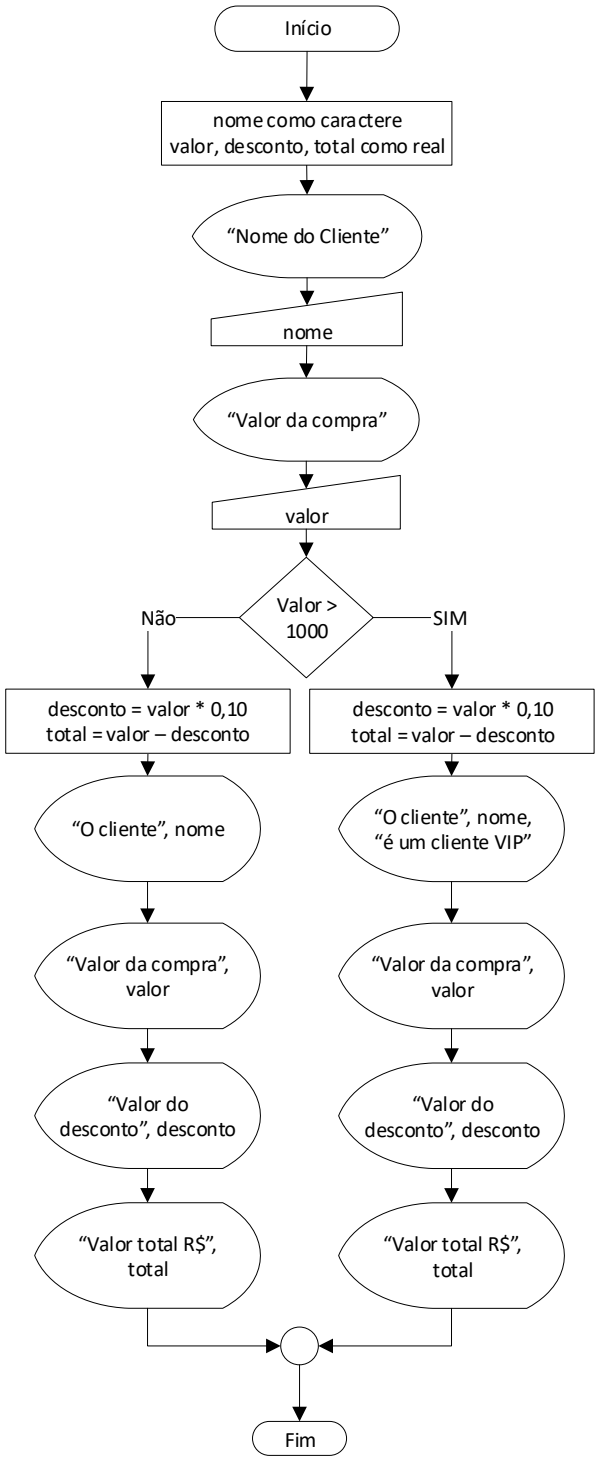
```java
import javax.swing.JOptionPane;

public class if_Ex3 {

    public static void main(String[] args) {
        // exercício 3

        //declaração de variáveis
        int numero;

        //entrada de dados
        numero = Integer.parseInt(JOptionPane.showInputDialog("Entre com um número"));

        //processamento e saída
        if ((numero % 2) ==0) {
            JOptionPane.showMessageDialog(null, "O número inserido é par");
        } else {
            JOptionPane.showMessageDialog(null, "O número inserido é ímpar");
        }

    }

}
```

4.

Pseudocódigo

```
Programa ex4

Declare
   nome como caractere
   valor, desconto, total como real
Início
   Escreva (“Nome do Cliente”)
   Leia (nome)
   Escreva (“Valor da compra R$”)
   Leia (valor)

   Se ( valor > 1000 ) Então
      desconto = valor * 0,10
      total = valor – desconto
      Escreva (“ O cliente”, nome, “é um
cliente

VIP”)
      Escreva (“Valor da compra R$”, valor)
      Escreva (“Valor do desconto R$”,

desconto)
      Escreva (“Valor Total R$”, total)
   Senão
      desconto = 0
      total = valor – desconto
      Escreva (“ O cliente”, nome)
      Escreva (“Valor da compra R$”, valor)
      Escreva (“Valor do desconto R$”,

desconto)
      Escreva (“Valor Total R$”, total)
   Fim-Se
Fim.
```

Fluxograma

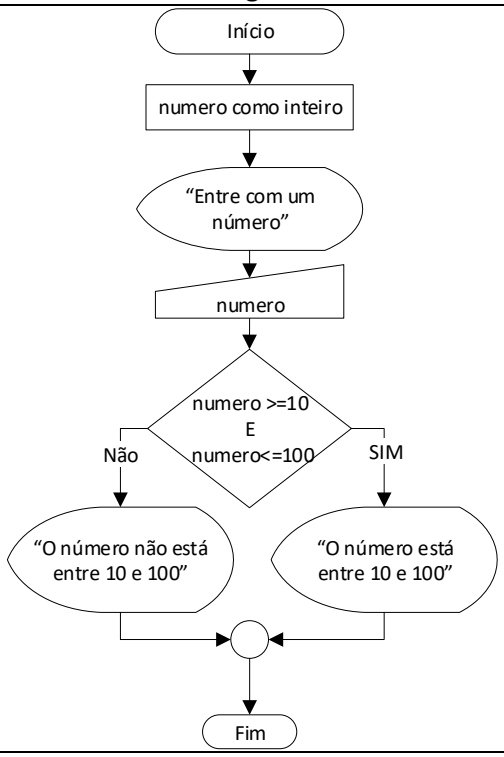


Java

if_Ex3.java *if_Ex4.java

```
1  import javax.swing.JOptionPane;
2
3  public class if_Ex4 {
4
5      public static void main(String[] args) {
6          // exercício 4
7          //declaração de variáveis
8          String nome;
9          double valor,desconto, total;
10
11         //entrada de dados
12         nome = JOptionPane.showInputDialog("Nome do Cliente");
13         valor = Integer.parseInt(JOptionPane.showInputDialog("Valor da Compra R$"));
14
15         //processamento e saída
16         if (valor > 1000) {
17             desconto = valor * 0.10; //calcula um desconto de 10%
18             total = valor - desconto;
19             JOptionPane.showMessageDialog(null, "O cliente" + nome + " é um cliente VIP" +
20                     "\n Valor da compra R$" + valor +
21                     "\n Valor do desconto R$" + desconto +
22                     "\n Total a pagar R$" + total);
23         } else {
24             desconto = 0;
25             total = valor - desconto;
26             JOptionPane.showMessageDialog(null, "O cliente" + nome +
27                     "\n Valor da compra R$" + valor +
28                     "\n Valor do desconto R$" + desconto +
29                     "\n Total a pagar R$" + total);
30         }
31     }
32 }
33
```

5.

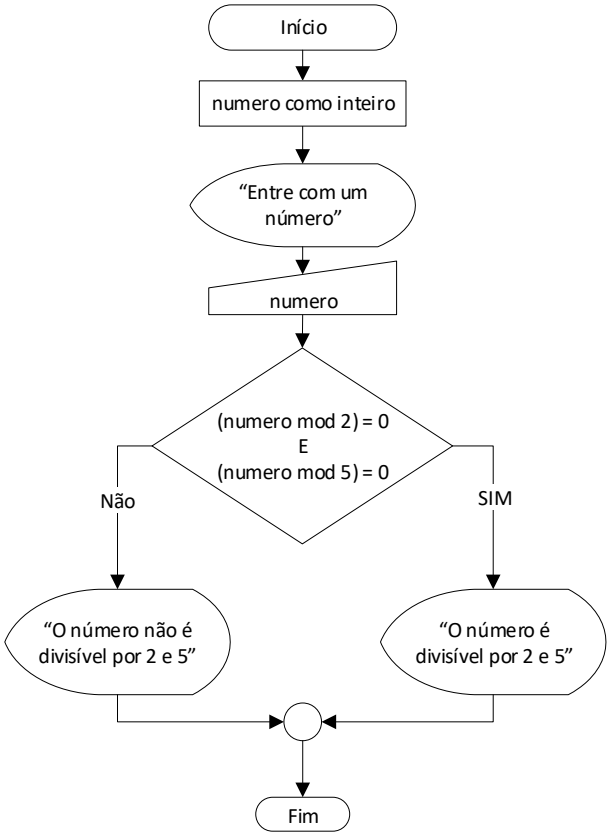
Pseudocódigo	Fluxograma
Programa ex5 Declare numero como inteiro Início Escreva (“Entre com um número”) Leia (numero) Se (numero >=10 **E** numero <=100) Então Escreva (“ O número”, numero, “está entre 10 e 100”) Senão Escreva (“ O número”, numero, “ não está entre 10 e 100”) Fim-Se Fim.	

Java

if_Ex4.java | if_Ex5.java

```
1  import javax.swing.JOptionPane;
2
3  public class if_Ex5 {
4
5      public static void main(String[] args) {
6          // exercício 5
7          //declaração de variáveis
8          int numero;
9
10         //entrada de dados
11         numero = Integer.parseInt(JOptionPane.showInputDialog("Entre com um número inteiro"));
12
13         //processamento e saída
14         if ((numero >=10) && (numero <=100)) {
15             JOptionPane.showMessageDialog(null, "O Número " + numero +
16                     " está entre 10 e 100");
17         } else {
18             JOptionPane.showMessageDialog(null, "O Número " + numero +
19                     " não está entre 10 e 100");
20         }
21     }
22
23 }
24
```

6.

Pseudocódigo	Fluxograma
Programa ex6 Declare numero como inteiro Início Escreva (“Entre com um número”) Leia (numero) Se ((numero mod 2)=0 **E** (numero mod 5)=0) Então Escreva (“ O número é divisível por 2 e 5”) Senão Escreva (“O número não é divisível por 2 e 5”) Fim-Se Fim.	

Java

*if_Ex6.java

```
1  import javax.swing.JOptionPane;
2
3  public class if_Ex6 {
4
5      public static void main(String[] args) {
6          // exercício 6
7
8          //declaração de variáveis
9          int numero;
10
11         //entrada de dados
12         numero = Integer.parseInt(JOptionPane.showInputDialog("Entre com um número"));
13
14         //processamento e saída
15         if ((numero % 2) == 0 && (numero % 5) == 0) {
16             JOptionPane.showMessageDialog(null, "O número é divisível por 2 e 5");
17         } else {
18             JOptionPane.showMessageDialog(null, "O número não é divisível por 2 e 5");
19         }
20     }
21
22 }
23
```

Estes exercícios devem ser entregues de forma on-line como atividades da agenda.

Exercício 1

A empresa NewInfo está desenvolvendo um sistema para classificar a prioridade na fila de espera de atendimento de um de seus clientes. A classificação da prioridade consiste em perguntar a idade do usuário a ser atendido e encaminhá-lo para uma fila prioritária caso este seja maior de 60 anos.

Elabore um programa que peça para o utilizador inserir a sua idade e exiba na tela a mensagem “fila prioritária” caso este tenha mais de 60 anos ou exiba “fila comum” caso contrário. Apresente a resposta em pseudocódigo, fluxograma e linguagem de programação Java.

Exercício 2

Recentemente o Estatuto do idoso foi alterado pela lei n. 13.466 de 12/07/2017 que acrescenta uma nova classificação de prioridade de atendimento dentre os idosos. O Artigo 2º m inciso 2 diz que “dentre os idosos, é assegurada prioridade especial aos maiores de 80 anos, atendendo-se suas necessidades sempre preferencialmente em relação aos demais idosos.”[1] Com isso é necessária uma alteração do programa que foi desenvolvido pela empresa NewInfo acrescentando mais essa situação disposta em lei.

Reescreva o programa do exercício 1 acrescentando essa nova situação de classificação. Agora temos a “fila comum”, a “fila prioritária” e dentro da fila prioritária teremos a “fila 80+”. Apresente a resposta em pseudocódigo, fluxograma e linguagem de programação Java.

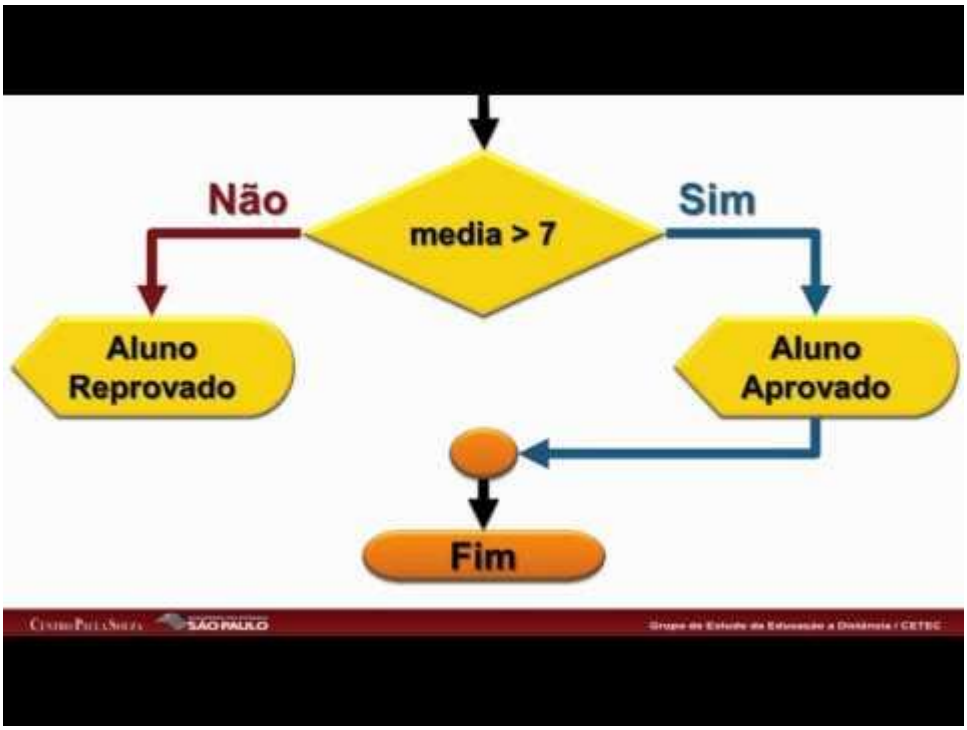
1. Fonte: http://www.planalto.gov.br/ccivil_03/_Ato2015-2018/2017/Lei/L13466.htm#art2. Acessado em 29/10/2017

Exercício 3

Joaquina deseja escrever um programa pseudocódigo, fluxograma e em linguagem Java que ao se entrar com um número inteiro de 1 a 12 ele exiba o nome do mês do ano correspondente. Caso coloque um número fora desta faixa o programa deve exibir uma mensagem de erro: “mês inválido”. Como ela fará este desafio?

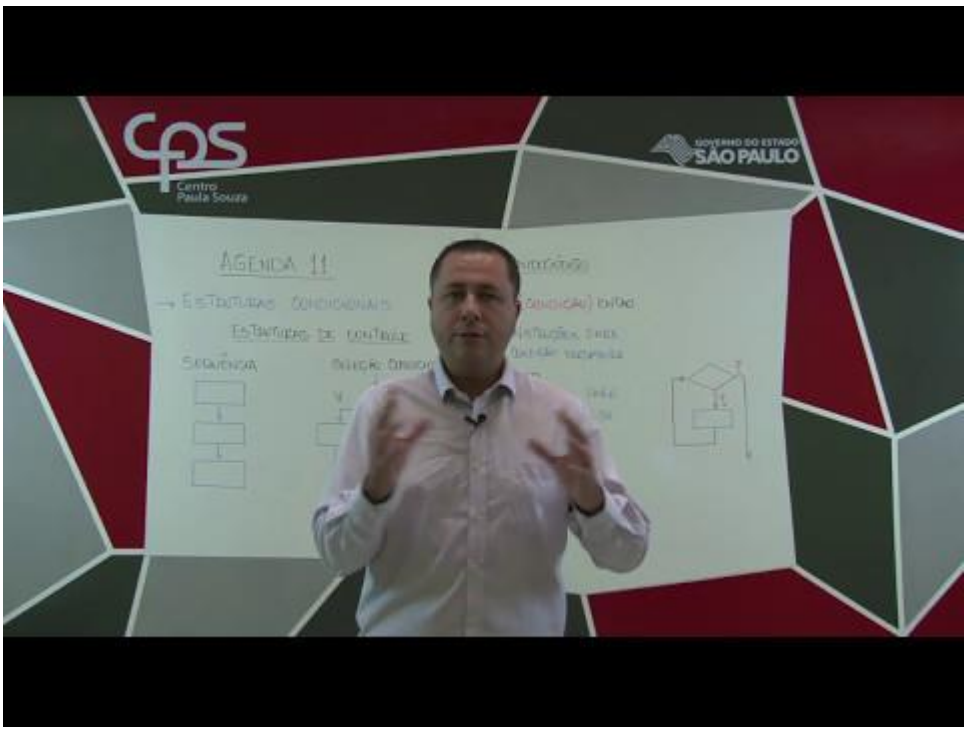
Não deixe também de assistir aos vídeos:

Informática - Módulo I - Agenda 12 - Estruturas de Decisão I



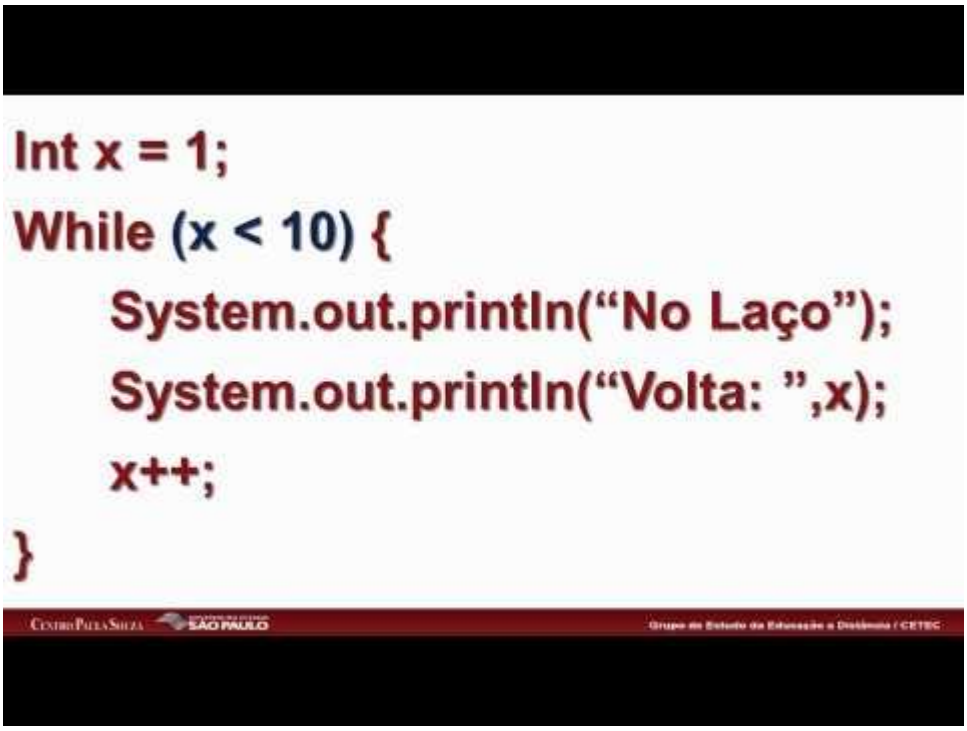
Link: https://www.youtube.com/watch?v=Y_wLEhL1Ha4 – Acessado em 14/12/2017

Informática -Módulo ! – Agenda 11



Link: https://youtu.be/H3PrZTZfVA4 - Acessado em 14/12/2017

Programação em Java e comando if



Link: https://www.youtube.com/watch?v=gVhcgdRFW78 – Acessado em 14/12/2017

Para aprofundamento dos temas discutidos nesta aula, seguem abaixo algumas dicas de filmes, livros e artigos que se relacionam com o conteúdo estudado. Estas dicas são muito importantes para você!

Vídeo:

Procure no YouTube um vídeo denominado ;“EstruturaCondicional VisuALG”, disponível em : http://www.youtube.com/watch?v=mJzVSmv5vn8. Acessado em 29/10/2017.

Procure no Youtube um vídeo denominado “Curso de Java #09 – Estruturas Condicionais (Parte1)”, disponível em : https://www.youtube.com/watch?v=wW3eve4vTMc. Acessado em 29/10/2017.

Procure no Youtube um vídeo denominado “Curso de Java #10 – Estruturas Condicionais (Parte2)”, disponível em : https://www.youtube.com/watch?v=oNSrBld06qs. Acessado em 29/10/2017.

Livros:

FORBELLONE, André L. V.; ELBERSPACHER, Henri Frederico. Lógica de Programação: A Construção de Algoritmos e Estrutura de Dados. Editora Pearson, 2000.

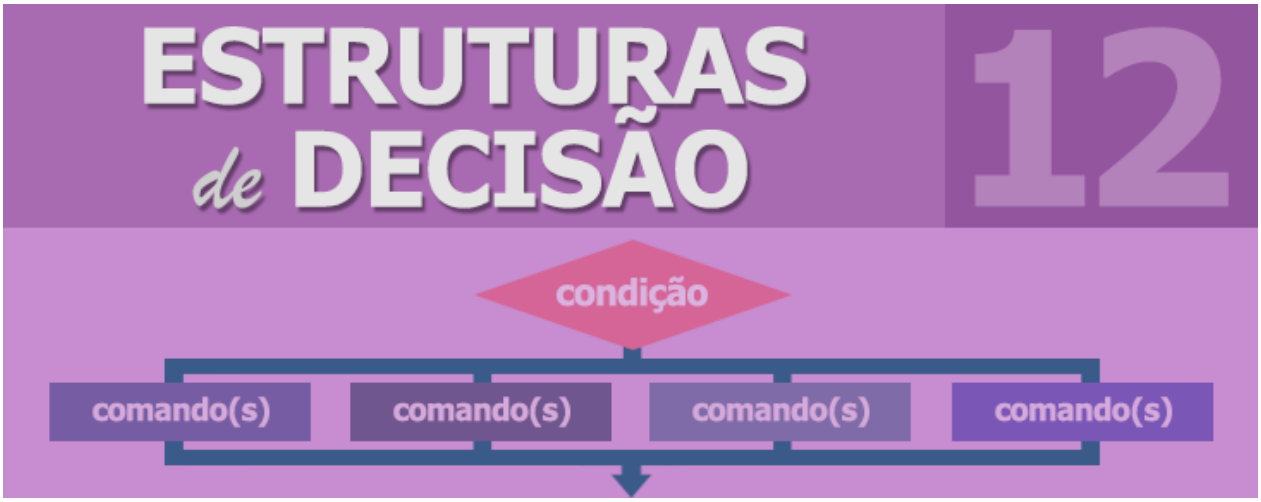
MANZANO, José Augusto N. G; OLIVEIRA, Jayr Figueiredo. Algoritmos: Lógica para Desenvolvimento de Programação. Editora Érica, 2007.

PUGA, SANDRA; RISSETTI, GERSON. Lógica de Programação e Estruturas de Dados com Aplicações em Java. Editora Pearson. 2009

SCHILDT, HEBERT. Java para Iniciantes. Editora Bookman.2015

Artigos:

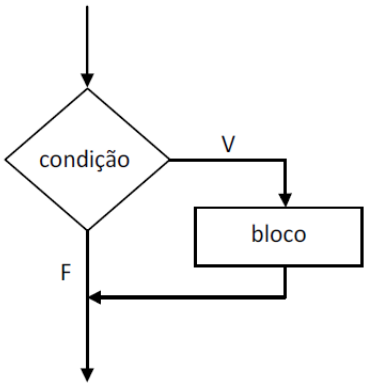
MORAES, Paulo Sérgio de. ;“Lógica de Programação”, 2000, disponível em ;http://www.siban.com.br/destaque/21_carta.pdf. Acessado em 29/10/2017



AGENDA 12

ESTRUTURAS DE DECISÃO "SELECIONE...CASO...SENÃO...FIM_SELECIONE"

MOMENTO DE REFLEXÃO



Você já percebeu quantas escolhas tem de fazer diariamente? Você começa de manhã escolhendo a sua roupa, escolhe também qual o caminho para a escola ou para o trabalho, na hora do almoço escolhe o que vai comer e por aí vai... As decisões fazem parte do nosso cotidiano e temos de conviver com a responsabilidade de decidir desde o momento em que acordamos até ao deitar novamente. Para complicar um pouco, nem sempre podemos fazer escolhas simples, com apenas duas opções. Quando vamos ao cinema, por exemplo, precisamos escolher um dentre os cinco ou seis disponíveis naquela temporada, considerando que você não vá para assistir um filme especificamente.

No mundo da computação não é diferente. As escolhas e as decisões também são necessárias e influenciam no sistema como um todo. Assim, nesta aula, você verá como esta ideia de escolhas múltiplas pode ser desenvolvida utilizando a lógica de programação. Verá, também, quais são os comandos utilizados e como fica sua estrutura dentro do algoritmo.

Pense em uma situação onde vamos comprar um automóvel. Já imaginou quantas opções de marcas e modelos existem? Imagine como seria se tivéssemos que listar todas essas opções somente utilizando um comando que nos permite selecionar apenas duas alternativas por vez.

Nem sempre os algoritmos envolvem apenas fórmulas ou regras matemáticas. Em determinados momentos, a lógica requer que um programador reproduza em um programa soluções que envolvam decisões desenvolvendo algoritmos e fluxogramas. Ou seja, é necessário que o programador consiga se antecipar às situações que podem exigir que sejam tomadas decisões e para que ocorram ações que as suportem utilizando modelos, pseudocódigos na representação da solução de problemas. Neste sentido, essas decisões precisam ser interpretadas pelo programa que precisa “entender” o que fazer. E fazer o computador “entender” que ações executar e como interpretar as instruções seguintes são tarefas do programador.
Nesta agenda, além das estruturas de decisão “selecione...caso...senão...fim_selecione”, iremos abordar como realizamos um tratamento básico de erros de entrada de dados de um programa a fim de que o usuário possa ter uma ideia do que aconteceu de errado durante a execução do programa.

Nem sempre os algoritmos envolvem apenas fórmulas ou regras matemáticas. Em determinados momentos, a lógica requer que um programador reproduza em um programa soluções que envolvam decisões. Ou seja, é necessário que o programador consiga se antecipar às situações que podem exigir que sejam tomadas decisões e para que ocorram ações que as suportem. Neste sentido, essas decisões precisam ser interpretadas pelo programa que precisa “entender” o que fazer. E fazer o computador “entender” que ações executar e como interpretar as instruções seguintes são tarefas do programador.

Imagem 20

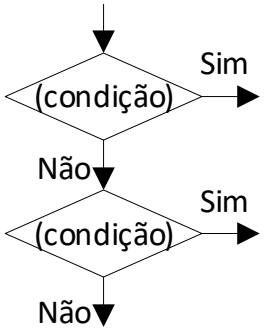
Já sabemos que a lógica de programação possui mecanismos que nos permitem tomar decisões dentro de um algoritmo. Sabemos também que estes mecanismos são denominados “estruturas de decisão”. A novidade então é que estas estruturas não se restringem a apenas o “Se...senão...fim_se”. E se tivéssemos uma decisão a ser tomada entre dez opções? Será que o “Se...senão...fim_se” seria o mais apropriado para esta situação? Será que existe alguma outra estrutura mais adequada para este tipo de ocorrência? Existe sim! Esta estrutura é chamada de “Selecione...caso... senão...fim_selecione” e sua função principal é

facilitar a escrita do algoritmo quando se têm muitos caminhos a serem seguidos a partir de uma decisão. Assim como a estrutura “Se...senão...fim_se”, é necessário saber quando e como utilizar o “Selecione...caso...senão...fim_selecione”.

Se recapitularmos o último exemplo de Estrutura de decisão aninhada da agenda anterior no qual o programa exibia o dia da semana correspondente ao usuário inserir um número de 1 a 7, notamos que o código do programa fica relativamente confuso devido a grande quantidade de comandos de seleção (Se). A tendência dessa confusão é aumentar conforme o número de comandos de decisão aninhados for crescendo. Isso, é claro, levando em conta que estamos realizando a comparação sempre com a mesma variável.

Para estas situações podemos utilizar a estrutura “selecione caso...senão...fim_selecione” do pseudocódigo ou a estrutura switch-case do Java.

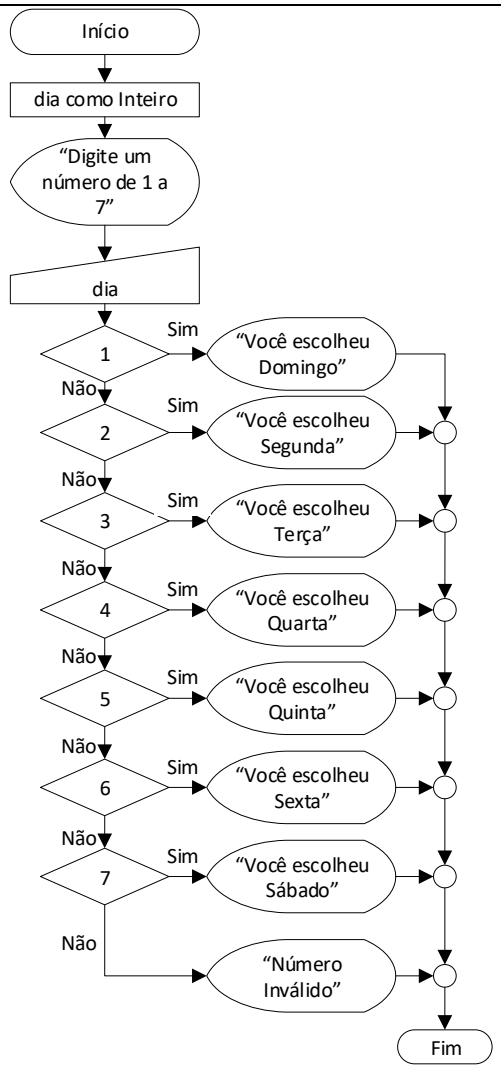
Essas estruturas permitem que a seleção correta seja feita a partir da comparação do valor do conteúdo da variável com uma lista definida durante a programação. Ao encontrar a correspondência correta os respectivos comandos são executados e as demais opções ignoradas. Tanto as sintaxes em pseudocódigo quanto em Java são semelhantes como temos a seguir:

Pseudocódigo	Fluxograma	Java
Selecione Caso {variável} **Caso** condição 01 {comando(s)} **Caso** condição 02 {comando(s)} **Caso** condição 03 {comando(s)} ... **Senão** {comando(s)} **Fim-Selecione**		**switch** (variável)**{** **case** condição 01: {comando(s)}; **break;** **case** condição 02: {comando(s)}; **break;** **case** condição 03: {comando(s)}; **break;** ... **default:** {comando(s)}; **}**

O Funcionamento é muito simples. Vamos ver um exemplo para entender o funcionamento.

Pseudocódigo	Fluxograma

```
Programa semana
Declare
        dia como inteiro
Início
   Escreva (“Digite um Número de 1 a 7”)
   Leia(dia)
   Selecione Caso (dia)
        Caso 1
                Escreva (“Você Escolheu Domingo”)
        Caso 2
                Escreva (“Você Escolheu Segunda”)
        Caso 3
                Escreva (“Você Escolheu Terça”)
        Caso 4
                Escreva (“Você Escolheu Quarta”)
        Caso 5
                Escreva (“Você Escolheu Quinta”)
        Caso 6
                Escreva (“Você Escolheu Sexta”)
        Caso 7
                Escreva (“Você Escolheu Sábado”)
        Senão
                Escreva(“Número Inválido”)
   Fim-Selecione
Fim.
```



O pseudocódigo inicia-se com a leitura da variável **dia** e a estrutura de seleção **Selecione Caso** utiliza a variável **dia** para comparar com as diversas opções de 1 a 7. **Caso** o número inserido corresponda a um dos critérios o programa escreve o dia da semana correspondente para o usuário. Por exemplo, se o usuário digitar 6, o programa irá escrever somente “você escolheu Sexta”, pois é a única comparação válida. Se o programa não achar nenhuma comparação válida a cláusula **Senão** é executada e é exibida a mensagem “número inválido” para o usuário.

Você acha que já viu o programa em algum lugar? Não é impressão, não! É exatamente o mesmo programa de exemplo de Estrutura de decisão aninhada da agenda anterior. Com essa nova estrutura que foi apresentada o código foi escrito de maneira muito mais elegante. O fluxograma permaneceu o mesmo sem alteração nenhuma.

Vamos agora ao código em Java:

switchCase.java

```
1  import javax.swing.JOptionPane;
2
3  public class switchCase {
4
5      public static void main(String[] args) {
6          //declaração de variáveis
7          int dia; // variável para armazenamento da semana
8
9          //entrada de dados com conversão de tipos juntas
10         dia = Integer.parseInt(JOptionPane.showInputDialog("Entre com um número de 1 a 7"));
11
12         switch (dia) {
13
14         case 1:
15             JOptionPane.showMessageDialog(null, "Você escolheu Domingo");
16             break;
17         case 2:
18             JOptionPane.showMessageDialog(null, "Você escolheu Segunda");
19             break;
20         case 3:
21             JOptionPane.showMessageDialog(null, "Você escolheu Terça");
22             break;
23         case 4:
24             JOptionPane.showMessageDialog(null, "Você escolheu Quarta");
25             break;
26         case 5:
27             JOptionPane.showMessageDialog(null, "Você escolheu Quinta");
28             break;
29         case 6:
30             JOptionPane.showMessageDialog(null, "Você escolheu Sexta");
31             break;
32         case 7:
33             JOptionPane.showMessageDialog(null, "Você escolheu Sábado");
34             break;
35
36         default:
37             JOptionPane.showMessageDialog(null, "Número Inválido");
38             break;
39         }
40     } // fim do método main
41
42 } // fim da classe
43
```

Analogamente ao pseudocódigo, em Java, após fazermos a entrada do dado pelo usuário, o comando **switch (dia)** irá comparar o valor armazenado na variável **dia** com um valor de 1 a 7 em cada comando **case**.

Exemplo:

se o usuário entrar com o valor 3, teremos **dia = 3** e ao executarmos o comando **case 3**: (dia=3), como o resultado da comparação será verdadeiro, ele executará o comando **JOptionPane.showMessageDialog(null, “Você escolheu Terça”);** e o comando **break;** e por fim finalizará o programa.

O comando **break** tem a finalidade de parar a execução do comando switch, uma ver que já foi executada a ação necessária (comparação verdadeira) e não há a necessidade de se continuar com o comando switch.

Caso nenhuma comparação resulte em uma resposta verdadeira, o comando **default:** é executado gerando uma mensagem de “número inválido” para o usuário.

VOCÊ NO COMANDO

Pense em como poderíamos elaborar um aplicativo em pseudocódigo utilizando a estrutura selecione caso...senão...fim_selecione em que o usuário possa selecionar entre 6 cores distintas. Reflita como podemos resolver essa questão antes de prosseguir a leitura.

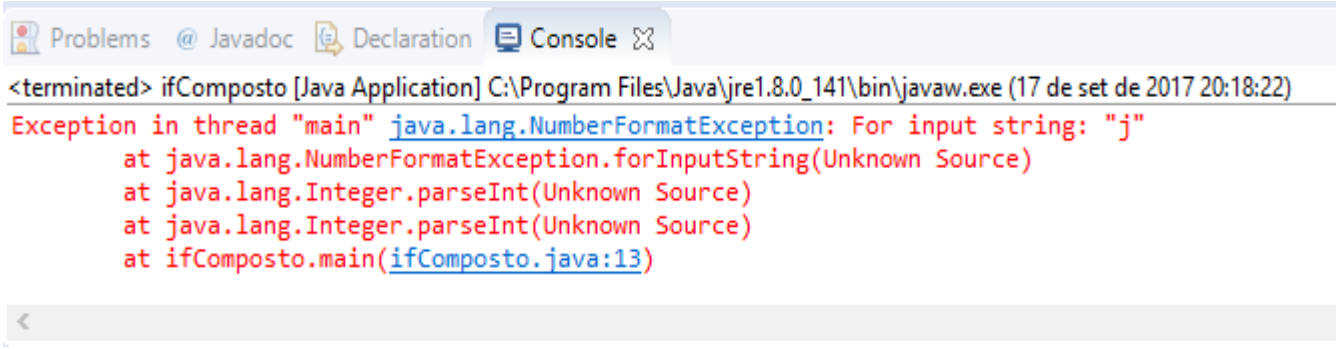
Programa de seleção de cores:

```
Programa cor
Declare
        cor como caractere
Início
   Escreva (“Digite o nome de uma cor”)
   Leia(cor)
   Selecione Caso (cor)
        Caso “Azul”
                Escreva (“Você Escolheu a cor Azul”)
        Caso “Vermelho”
                Escreva (“Você Escolheu a cor Vermelha”)
        Caso “Branco”
                Escreva (“Você Escolheu a cor Branca”)
        Caso “Preto”
                Escreva (“Você Escolheu a cor Preta”)
        Caso “Verde”
                Escreva (“Você Escolheu a cor Verde”)
        Caso “Amarelo”
                Escreva (“Você Escolheu a cor Amarela”)
        Senão
                Escreva(“Cor Inválida”)
   Fim-Selecione
Fim.
```

Tratamento de erros com o comando Try-Catch

Até agora não foi realizado nenhum tratamento de erros quando o usuário insere algo errado no sistema como, por exemplo, insere um caractere no lugar de um número. Quando isso acontece sem o tratamento de erro, do ponto de vista do usuário o programa simplesmente fecha. Do

ponto de vista do programador uma mensagem de erro é gerada no IDE (ambiente de desenvolvimento integrado) para alertá-lo. A figura abaixo ilustra um erro de entrada de dados:



Parece uma coisa desnecessária realizarmos tratamento de erros à primeira vista. Mas reflita: quantas vezes não ficamos com raiva quando ao utilizar algum software ou app e ele simplesmente fechou sem dar nenhuma explicação? Temos que pensar sempre no usuário do programa.

Mas, o problema realmente é com o usuário que fica sem saber o que aconteceu. Com certeza algum programa de computador ou app de celular já fechou inesperadamente sem nenhum aviso de erro e perdemos o que já estava sendo feito. Isso dá uma raiva imensa.

Para evitar ou minimizar aborrecimentos, no Java existem várias técnicas e rotinas de tratamento de erros. Vamos apresentar aqui o comando try-catch. Ele consiste em capturar erros na conversão de tipos na entrada de dados para os nossos programas. O comando trata mais tipos de erro, mas não é o escopo desta explicação.

A sintaxe é:

```
try{
 comando(s);
} cacth (NumberFormatException e){
        Comando(s);
}
```

Vamos explicar com um exemplo:

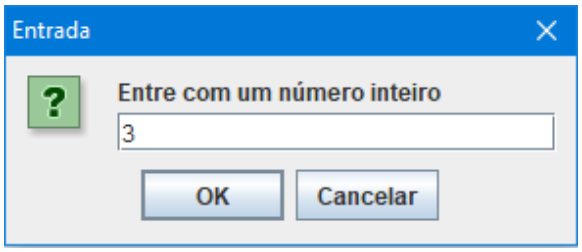
*tryCatch.java

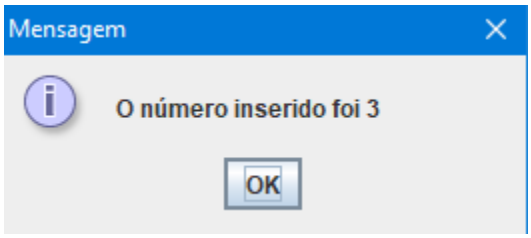
```
1  import javax.swing.JOptionPane;
2
3  public class tryCatch {
4
5      public static void main(String[] args) {
6          // declaração de variáveis
7          int numero=0;
8          String aux;
9
10         //entrada de dados
11         try {
12             aux = JOptionPane.showInputDialog("Entre com um número inteiro");
13             numero = Integer.parseInt(aux);
14             JOptionPane.showMessageDialog(null, "O número inserido foi " + numero);
15         } catch (NumberFormatException e) {
16             JOptionPane.showMessageDialog(null, "Entre somente com um número Inteiro",
17                 "E R R O", JOptionPane.ERROR_MESSAGE);
18         }//fim do try-catch
19     }//fim do método main
20 }//fim da classe
21
```

O try-catch funciona da seguinte forma:

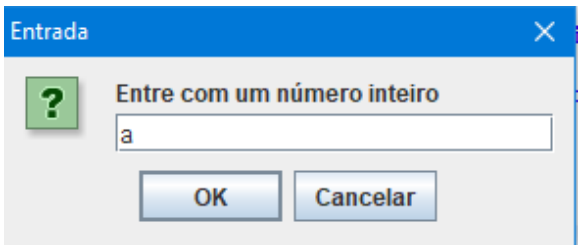
Na linha 11 o comando **try** tenta executar os comandos das linhas 12 a 14 que estão dentro das chaves, que é fechada na linha 15. Caso não consiga, ele executa a cláusula **catch (NumberFormatException e)** e os comandos dentro das chaves das linhas 15 a 18. Isto Exibirá uma mensagem de erro.

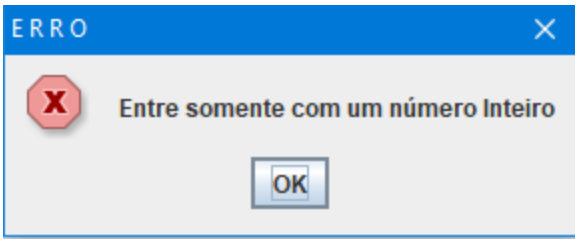
Quando não ocorre nenhum erro de inserção de dados por parte do utilizador, ou seja, é inserido um número inteiro o programa exibe a mensagem: “O número inserido foi <numero>” como nas figuras abaixo:





Quando um erro de inserção de dados ocorre e, por exemplo, um caractere ou número real é inserido a seguinte mensagem de erro aparece:





Note que o try-catch capturou o erro de conversão, afinal o caractere “a” não é um número inteiro. E exibiu a mensagem de erro correspondente.

Um detalhe interessante: não sei se repararam, mas a caixa de diálogo da **Erro! Fonte de eferência não encontrada.** é bem diferente da **Erro! Fonte de referência não encontrada.**. Isso porque durante a escrita da caixa de diálogo de saída, esta foi formatada. Vamos analisar o comando da linha 16 e 17:

```
JOptionPane.showMessageDialog(null, "Entre somente com um número Inteiro",
                                    "E R R O", JOptionPane.ERROR_MESSAGE);
```

O comando **JOptionJane.showMessageDialog(null, “Entre com um número Inteiro”** exibe a caixa de diálogo que já conhecemos. Note que o número de argumentos do comando aumentou:

,”E R R O”, JOptionPane.ERROR_MESSAGE);

É isso que inclui a formação da caixa de diálogo. A parte em **vermelho** (“E R R O”), insere o título da janela e a parte em verde (JOptionPane.ERROR_MESSAGE) altera o ícone para uma cruz vermelha para indicar erro.

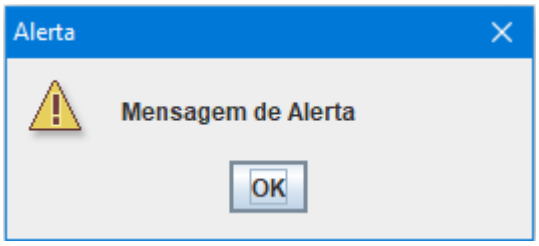
Uma observação: os argumentos são sempre separados por vírgula

Exemplo:

Se quisermos exibir uma mensagem de alerta para o usuário com um ponto de exclamação o código será:

mensagem_de_alerta.java

```
1  import javax.swing.JOptionPane;
2
3  public class mensagem_de_alerta {
4
5      public static void main(String[] args) {
6          JOptionPane.showMessageDialog(null, "Mensagem de Alerta",
7                  "Alerta",JOptionPane.WARNING_MESSAGE);;
8
9      }
10
11 }
12
```



Perceba que o título da janela foi definido como Alerta.

VOCÊ NO COMANDO

Acabamos de ver como podemos modificar a exibição de uma caixa de diálogo de saída de dados. Explore um pouco na IDE Eclipse algumas outras personalizações desta caixa de diálogo. Utilize a função de auto completar comandos da IDE. Para usar esta função basta digitar o início de um comando e pressionar as teclas Control (CRTL) + barra de espaços.

Exercícios sobre estrutura de seleção:

Os exercícios de 1 a 4 devem ser desenvolvidos elaborando o pseudocódigo, fluxograma e linguagem Java:

1. Vanessa precisa desenvolver um programa em que ao se digitar o código de um produto cadastrado da papelaria Lápis Colorido este retorne o nome do produto. Se o código do produto não estiver cadastrado o programa deve exibir a mensagem produto não cadastrado. A tabela a seguir descreve os códigos e os produtos cadastrados:

   Como Vanessa resolveria essa questão?

Código	Produto
100	Lápis preto N.2
150	Borracha branca
200	Caneta Azul
230	Caneta Vermelha
256	Giz de Cera 12un.
300	Cartolina Branca
310	Resma de Sulfite branco A4
400	Estojo Escolar Verde
470	Caderno universitário 100fls.
500	Caderno brochura 50 fls.

2. Eliberto deseja fazer um menu de seleção de um programa utilizando o comando de seleção múltipla. O menu consiste nas opções de:
   1. Cadastrar usuário
   2. Alterar dados
   3. Excluir Usuário

   Como Eliberto faria esse programa? Elabore uma mensagem diferente para ser exibida para o usuário ao selecionar cada um dos itens dos menus.

3. Além do menu apresentado na questão anterior Eliberto precisa fazer alguns submenus com a seguinte estrutura:

    1. Cadastrar usuário
    2. Alterar dados
        1. Alterar nome
        2. Alterar Endereço
        3. Alterar telefone
        4. Alterar RG.
    3. Excluir usuário

    Altere o programa do exercício anterior para incluir este submenu e exiba uma mensagem correspondente de acordo com o menu escolhido pelo usuário.

4. Crie um programa de uma calculadora com cinco operações básicas (soma, subtração, multiplicação, divisão e resto da divisão) onde dois números reais devem ser inseridos, e posteriormente em um menu o usuário escolha qual operação deve ser realizada. Após a execução da operação selecionada o resultado deve ser apresentado na tela do computador.

Exercícios para serem resolvidos somente em linguagem Java (5 a 7):

5. Elabore um programa em que o usuário deva inserir um número do tipo double e a seguir o programa exiba este número. O programa deve conter uma rotina de tratamento de erros quanto a inserção de dados de tipos errados.
6. Refaça o exercício 4 sobre estruturas de seleção (calculadora) adicionando a rotina de tratamento de erros try-catch. Não se esqueça de alterar a caixa de diálogo quando houver um erro na execução do programa
7. Denis decidiu, por curiosidade, elaborar um programa que realiza a conversão de quilômetros para milhas. Sabe-se que uma milha tem 1,609km. Como Denis resolveria este problema? O programa deve utilizar o tratamento de erros do tipo try-catch.

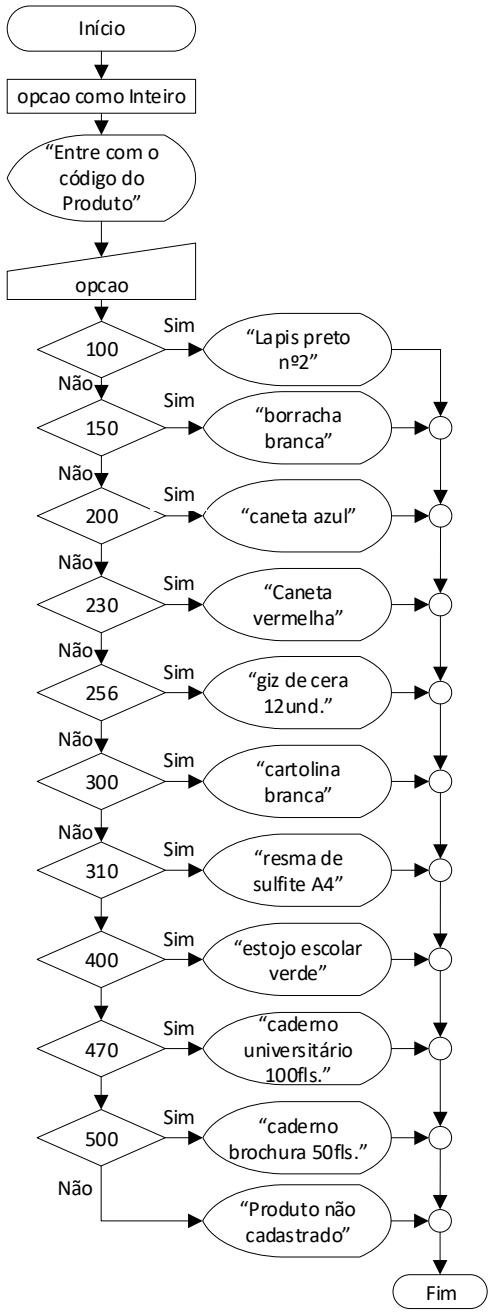
Respostas:

Comando de Seleção de Múltipla

1.

Pseudocódigo	Fluxograma

```
Programa ex1
Declare
   opcao como inteiro
Início
   Escreva (“Entre com o código do produto”)
   Leia (opcao)
   Selecione Caso (opcao)
     Caso condição 100
       Escreva (“Lápis preto nº2”)
     Caso condição 150
       Escreva (“borracha branca”)
     Caso condição 200
       Escreva (“caneta azul”)
     Caso condição 230
       Escreva (“caneta vermelha”)
     Caso condição 256
       Escreva (“giz de cera 12 unid.”)
     Caso condição 300
       Escreva (“cartolina branca”)
     Caso condição 310
       Escreva (“resma de sulfite A4”)
     Caso condição 400
       Escreva (“estojo escolar verde”)
     Caso condição 470
       Escreva (“caderno universitário 100fls.”)
     Caso condição 500
       Escreva (“caderno brochura 50fls.”)
     Senão
       Escreva (“Produto não cadastrado”)
   Fim-selecione
Fim.
```

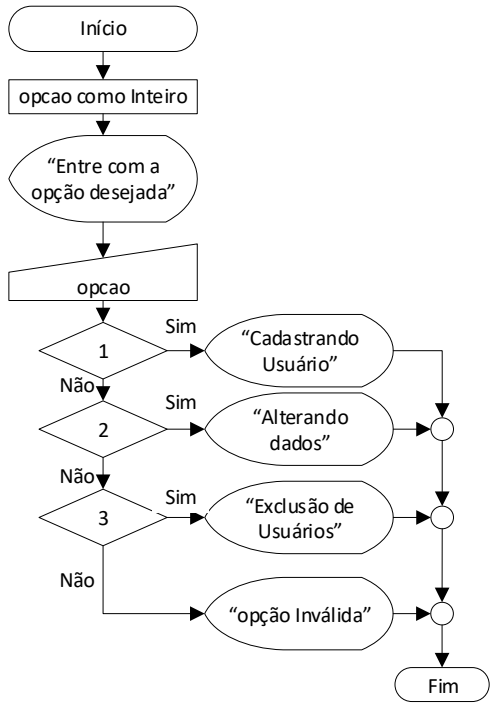


Java

Switch_ex1.java

```java
import javax.swing.JOptionPane;

public class Switch_ex1 {

    public static void main(String[] args) {
        // Exercício 1

        //declaração de variáveis
        int opcao;

            //entrada de dados
            opcao = Integer.parseInt(JOptionPane.showInputDialog("Entre com o código "
                    + "do produto"));

            //processamento e saida
            switch (opcao) {
            case 100:
                JOptionPane.showMessageDialog(null, "O produto selecionado é Lapis preto n.2");
                break;

            case 150:
                JOptionPane.showMessageDialog(null, "O produto selecionado é Borracha branca");
                break;

            case 200:
                JOptionPane.showMessageDialog(null, "O produto selecionado é Caneta Azul");
                break;

            case 230:
                JOptionPane.showMessageDialog(null, "O produto selecionado é Caneta Vermelha");
                break;

            case 256:
                JOptionPane.showMessageDialog(null, "O produto selecionado é Giz de cera 12und.");
                break;

            case 300:
                JOptionPane.showMessageDialog(null, "O produto selecionado é Cartolina Branca");
                break;

            case 310:
                JOptionPane.showMessageDialog(null, "O produto selecionado é Resma de "
                        + "sulfite branco A4");
                break;

            case 400:
                JOptionPane.showMessageDialog(null, "O produto selecionado é Estojo "
                        + "escolar verde");
                break;

                  ---

            case 470:
                JOptionPane.showMessageDialog(null, "O produto selecionado é caderno "
                        + "universitário 100fls.");
                break;

            case 500:
                JOptionPane.showMessageDialog(null, "O produto selecionado é caderno"
                        + "brochura 50fls.");
                break;

            default:
                JOptionPane.showMessageDialog(null, "Produto não cadastrado");
                break;
            }//fim do switch-case
    } // fim do método main
} // fim da classe
```

2.

Pseudocódigo	Fluxograma
Programa ex2 Declare opcao como inteiro Início Escreva (“Entre com a opção desejada:”) Escreva (“1 – Cadastrar usuário”) Escreva (“2 – Alterar dados”) Escreva (“3 – Excluir usuário”) Leia (opcao) Selecione Caso (opcao) Caso condição 1 Escreva (“Cadastrando Usuário”) Caso condição 2 Escreva (“Alterando dados”) Caso condição 3 Escreva (“Exclusão de usuários”) Senão Escreva (“Opção Inválida”) Fim-selecione Fim.	

Java

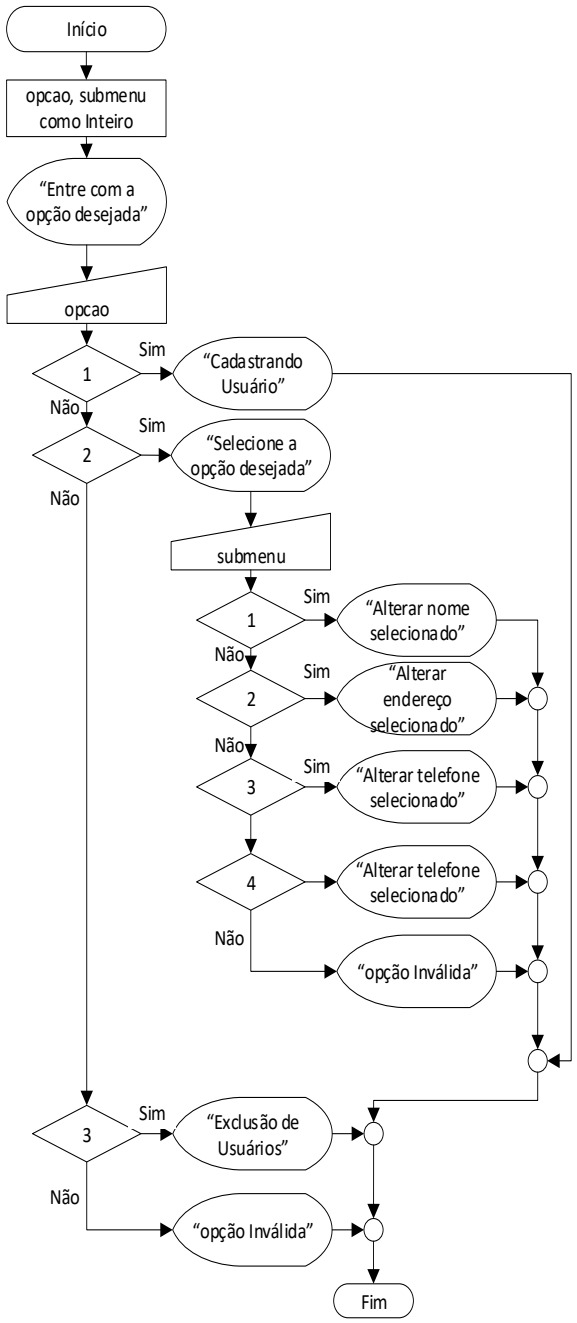
*Switch_ex2.java

```java
import javax.swing.JOptionPane;

public class Switch_ex2 {

    public static void main(String[] args) {
        // Exercício 2

        //declaração de variáveis
        int opcao;

            //entrada de dados
            opcao = Integer.parseInt(JOptionPane.showInputDialog("Entre com a opção desejada:" +
                    "\n1 - Cadastrar usuário" +
                    "\n2 - Alterar dados"+
                    "\n3 - Excluir usuário"));


            //processamento e saida
            switch (opcao) {
            case 1:
                JOptionPane.showMessageDialog(null, "Cadastrando usuário");
                break;

            case 2:
                JOptionPane.showMessageDialog(null,"Alterando dados");
                break;

            case 3:
                JOptionPane.showMessageDialog(null, "Exclusão de usuários");
                break;

            default:
                JOptionPane.showMessageDialog(null, "Opção Inválida");
                break;
            }//fim do switch-case

    } // fim do método main

} // fim da classe

```

3.

Pseudocódigo	Fluxograma

```
Programa ex3
Declare
  opcao como inteiro
Início
   Escreva (“Entre com a opção desejada:”)
   Escreva (“1 – Cadastrar usuário”)
   Escreva (“2 – Alterar dados”)
   Escreva (“3 – Excluir usuário”)
   Leia (opcao)
   Selecione Caso (opcao)
    Caso condição 1
      Escreva (“Cadastrando Usuário”)
    Caso condição 2
      Escreva (“Menu de alteração de dados”)
      Escreva (“Selecione a opção desejada:”)
      Escreva (“1 – Alterar nome”)
      Escreva (“2 – Alterar Endereço”)
      Escreva (“3 – Alterar Telefone”)
      Escreva (“4 – Alterar RG”)
      Leia (submenu)
      Selecione Caso (submenu)
        Caso condição 1
          Escreva (“Alterar nome selecionado”)
        Caso condição 2
          Escreva (“Alterar Endereço selecionado”)
        Caso condição 3
          Escreva (“Alterar telefone selecionado”)
        Caso condição 4
          Escreva (“Alterar RG selecionado”)
        Senão 1
          Escreva (“Opção Inválida”)
     Fim-selecione
    Caso condição 3
      Escreva (“Exclusão de usuários”)
    Senão
      Escreva (“Opção Inválida”)
   Fim-selecione
Fim.
```

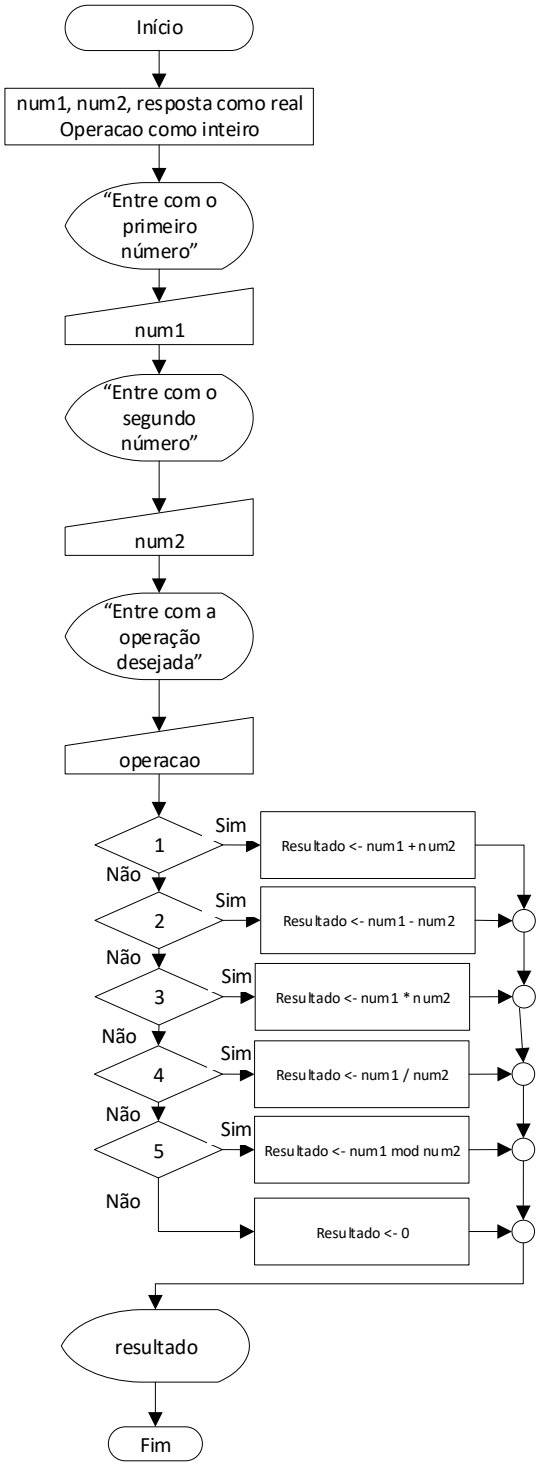


Java

Switch_ex3.java

```java
import javax.swing.JOptionPane;

public class Switch_ex3 {

    public static void main(String[] args) {
        // Exercício 3

        // declaração de variáveis
        int opcao, submenu;

        // entrada de dados
        opcao = Integer.parseInt(JOptionPane.showInputDialog("Entre com a opção desejada:" +
                "\n1 - Cadastrar usuário" +
                "\n2 - Alterar dados" +
                "\n3 - Excluir usuário"));

        // processamento e saida
        switch (opcao) {
        case 1:
            JOptionPane.showMessageDialog(null, "Cadastrando usuário");
            break;

        case 2:
            submenu = Integer.parseInt(JOptionPane.showInputDialog(
                    "Menu de alteração de dados" +
                    "\nSelecione a opção desejada:" +
                    "\n1 - Aletar nome" +
                    "\n2 - Alterar endereço" +
                    "\n3 - Alerar Telefone" +
                    "\n4 - Alterar RG"));
            // início do 2ºswitch
            switch (submenu) {
            case 1:
                JOptionPane.showMessageDialog(null, "Alterar nome selecionado");
                break;
            case 2:
                JOptionPane.showMessageDialog(null, "Alterar endereço selecionado");
                break;
            case 3:
                JOptionPane.showMessageDialog(null, "Alterar telefone selecionado");
                break;
            case 4:
                JOptionPane.showMessageDialog(null, "Alterar RG selecionado");
                break;
            default:
                JOptionPane.showMessageDialog(null, "Opção inválida");
                break;
            }// fim do 2º switch
            break;

        case 3:
            JOptionPane.showMessageDialog(null, "Exclusão de usuários");
            break;

        default:
            JOptionPane.showMessageDialog(null, "Opção Inválida");
            break;
        }// fim do switch-case
    } // fim do método main
} // fim da classe
```

4.

Pseudocódigo	Fluxograma
Programa ex4 Declare num1, num2 e resultado como real operacao como inteiro Início Escreva (“Entre com o primeiro número”) Leia (num1) Escreva (“Entre com o segundo número”) Leia (num2) Escreva (“Digite a operação desejada: ”) Escreva (“1 –soma”) Escreva (“2 – subtração”) Escreva (“3 – multiplicação”) Escreva (“4 – divisão”) Escreva (“5 – resto da divisão”) Leia (operacao) Selecione Caso (operacao) Caso condição 1 Resultado <- num1 + num2 Caso condição 2 Resultado <- num1 - num2 Caso condição 3 Resultado <- num1 * num2 Caso condição 4 Resultado <- num1 / num2 Caso condição 5 Resultado <- num1 mod num2 Senão Resultado <- 0 Fim-selecione Escreva (“O resultado da operação é “, resultado) Fim.	

Java

*Switch_ex4.java

```java
import javax.swing.JOptionPane;

public class Switch_ex4 {

    public static void main(String[] args) {
        // Exercício 4

        //declaração de variáveis
        double num1, num2, resultado;
        int operacao;

            //entrada de dados
            num1 = Double.parseDouble(JOptionPane.showInputDialog("Entre com o primeiro número"));
            num2 = Double.parseDouble(JOptionPane.showInputDialog("Entre com o segundo número"));

            operacao = Integer.parseInt(JOptionPane.showInputDialog(
                    "Digite a operaçao desejada: " +
                    "\n1 - Soma" +
                    "\n2 - Subtração" +
                    "\n3 - Multiplicação" +
                    "\n4 - Divisão" +
                    "\n5 - Resto da divisão"));

            //processamento
            switch (operacao) {
            case 1: //soma
                resultado = num1 + num2;
                break;

            case 2: //subtração
                resultado = num1 - num2;
                break;

            case 3: //multiplicação
                resultado = num1 * num2;
                break;

            case 4: //divisão
                resultado = num1 / num2;
                break;

            case 5: //resto da divisão
                resultado = num1 % num2;
                break;

            default:
                resultado = 0;
                break;
            }//fim do switch-case

            //saída de dados
            JOptionPane.showMessageDialog(null, "O resultado da operação é " + resultado);
    } // fim do método main

} // fim da classe

```

Try-Catch

Observação: resolução dos exercícios somente em Java.

5.

TryCatch_Ex1.java

```
1  import javax.swing.JOptionPane;
2
3  public class TryCatch_Ex1 {
4
5      public static void main(String[] args) {
6          // Exercício 1
7
8          //declaração de variáveis
9          double num;
10
11         try { //tratamento de erros
12             //entrada de dados
13             num = Double.parseDouble(JOptionPane.showInputDialog("Entre com um número"));
14             //saída de dados
15             JOptionPane.showMessageDialog(null, "O número digitado é " + num);
16
17         }catch (NumberFormatException e) {
18             JOptionPane.showMessageDialog(null, "Entre somente com números",
19                     "E R R O", JOptionPane.ERROR_MESSAGE);
20         }//fim do try-catch
21     }
22 }
23
```

6.

TryCatch_Ex2.java

```
import javax.swing.JOptionPane;

public class TryCatch_Ex2 {

    public static void main(String[] args) {
        // Exercício 1

        //declaração de variáveis
        double num1, num2, resultado;
        int operacao;

        try { //tratamento de erros
            //entrada de dados
            num1 = Double.parseDouble(JOptionPane.showInputDialog("Entre com o primeiro número"));
            num2 = Double.parseDouble(JOptionPane.showInputDialog("Entre com o segundo número"));

            operacao = Integer.parseInt(JOptionPane.showInputDialog(
                    "Digite a operaçao desejada: " +
                    "\n1 - Soma" +
                    "\n2 - Subtração" +
                    "\n3 - Multiplicação" +
                    "\n4 - Divisão" +
                    "\n5 - Resto da divisão"));

            //processamento
            switch (operacao) {
            case 1: //soma
                resultado = num1 + num2;
                break;

            case 2: //subtração
                resultado = num1 - num2;
                break;

            case 3: //multiplicação
                resultado = num1 * num2;
                break;

            case 4: //divisão
                resultado = num1 / num2;
                break;

            case 5: //resto da divisão
                resultado = num1 % num2;
                break;

            default:
                resultado = 0;
                break;
            }//fim do switch-case

            //saída de dados
            JOptionPane.showMessageDialog(null, "O resultado da operação é " + resultado);

        }catch (NumberFormatException e) {
            JOptionPane.showMessageDialog(null, "Entre somente com números",
                    "E R R O", JOptionPane.ERROR_MESSAGE);
        }//fim do try-catch
    }
}
```

7.

TryCatch_Ex3.java

```
1  import javax.swing.JOptionPane;
2
3  public class TryCatch_Ex3 {
4
5      public static void main(String[] args) {
6          // Exercício 3
7
8          //declaração de variáveis
9          double km, milha;
10
11         try { //tratamento de erros
12             //entrada de dados
13             km = Double.parseDouble(JOptionPane.showInputDialog("Entre com um número"
14                     + " em quilômetros"));
15
16             //processamento
17             milha = km * 1.609;
18
19             //saída de dados
20
21             JOptionPane.showMessageDialog(null, "O valor de " + km +
22                     " quilômetros convertido para milhas é " + milha + " milhas");
23
24         }catch (NumberFormatException e) {
25             JOptionPane.showMessageDialog(null, "Entre somente com números",
26                     "E R R O", JOptionPane.ERROR_MESSAGE);
27         }//fim do try-catch
28     }
29 }
30
```

ATIVIDADE ONLINE

Estes exercícios devem ser entregues de forma on-line como atividades da agenda.

Exercício 1

Na agenda anterior, no último exercício, Joaquina desejava escrever um programa pseudocódigo, fluxograma e em linguagem Java que ao se entrar com um número inteiro de 1 a 12 ele exiba o nome do mês do ano correspondente. Caso coloque um número fora desta faixa o programa deve exibir uma mensagem de erro: “mês inválido”.

Bem agora que nessa agenda vocês estudaram o comando selecione caso...senão...fim_selecione, reescreva o pseudocódigo, o fluxograma e o programa em Java utilizando os novos conhecimentos construídos ao longo dessa semana.

Exercício 2

Imagem 21

Maurício está programando em linguagem Java quando se deparou com um problema de entrada de dados. O programa que ele estava desenvolvendo fechava inesperadamente quando o usuário inseria no sistema um tipo de dado inconsistente com o tipo de dado declarado na variável utilizada para o armazenamento da informação. Isso pode vir a causar um grande transtorno pois o usuário fica sem saber qual o motivo do programa encerrar-se inesperadamente.

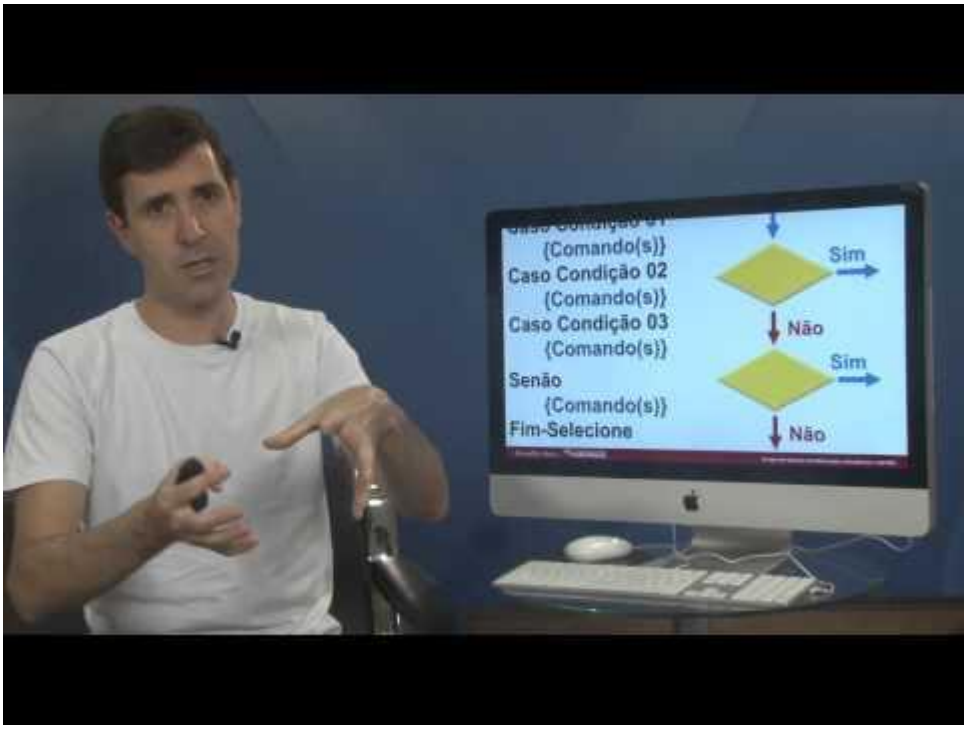
O programa que Maurício está desenvolvendo consiste na leitura do número de um voo e no peso de 10 de bagagens em quilogramas a serem despachadas. Posteriormente o programa deve calcular o peso total da carga e o peso médio de cada volume de bagagem.

Considerando que o número do voo consiste em uma sequência alfanumérica e uma bagagem pode ter um peso que inclua casas decimais, como Maurício pode resolver esse problema utilizando tratamento de erros?

AMPLIANDO HORIZONTES

Não deixe também de assistir ao vídeo:

Programação em Java e comando if



Link: https://www.youtube.com/watch?v=8b9S3fvFLis. Acessado em 02/11/2017.

Para aprofundamento dos temas discutidos nesta aula, seguem abaixo algumas dicas de filmes, livros e artigos que se relacionam com o conteúdo estudado. Estas dicas são muito importantes para você!

Vídeo:

Procure no YouTube um vídeo denominado ;“ Seleção de Múltipla Escolha (Escolha...Caso)”, disponível em : ;http:// www.youtube.com/watch?v=UG27GMtvzQM. Acessado em 02/11/2017.

Procure no Youtube um vídeo denominado “Curso de Java #10 – Estruturas Condicionais (Parte2)”, disponível em : https://www.youtube.com/watch?v=oNSrBld06qs. Acessado em 02/11/2017.

Livros:

FORBELLONE, André L. V.; ELBERSPACHER, Henri Frederico. Lógica de Programação: A Construção de Algoritmos e Estrutura de Dados. Editora Pearson, 2000.

MANZANO, José Augusto N. G; OLIVEIRA, Jayr Figueiredo. Algoritmos: Lógica para Desenvolvimento de Programação. Editora Érica, 2007.

PUGA, SANDRA; RISSETTI, GERSON. Lógica de Programação e Estruturas de Dados com Aplicações em Java. Editora Pearson. 2009

SCHILDT, HEBERT. Java para Iniciantes. Editora Bookman.2015

AGENDA 13
ESTRUTURAS DE REPETIÇÃO "PARA...FIM-PARA"

MOMENTO DE REFLEXÃO

"O homem não teria alcançado o possível se, repetidas vezes, não tivesse tentado o impossível."

(Max Weber)

A frase acima sinaliza o objeto de estudo desta aula: a Repetição. Mas antes que você ache estranho falar de repetição na lógica de programação, e também para entender melhor o conceito, vamos fazer uma analogia com o nosso cotidiano. Imagine a troca de uma lâmpada que está queimada em um determinado cômodo da casa. Imagine, também, que no momento da troca dispomos de uma caixa com várias lâmpadas que podem estar queimadas ou podem ser novas, mas não dá para saber só com olhar. Qual seria então o nosso procedimento? Obviamente, teríamos que experimentar colocando cada lâmpada individualmente no bocal até que uma delas acenda, não é mesmo? Da mesma forma, na lógica de programação, também temos a necessidade de "experimentar" várias opções na busca pela opção mais acertada. A forma como isso é feito, a estrutura utilizada e em quais momentos, é exatamente o tema de que trata esta aula.

POR QUE APRENDER?

Pense numa situação em que você está em uma festa e precisa reproduzir uma lista de 10 músicas durante toda a balada. Você pode escolher em repetir manualmente esta lista ou deixar programado para que as músicas fiquem tocando até o momento que você pare a reprodução dessa lista.

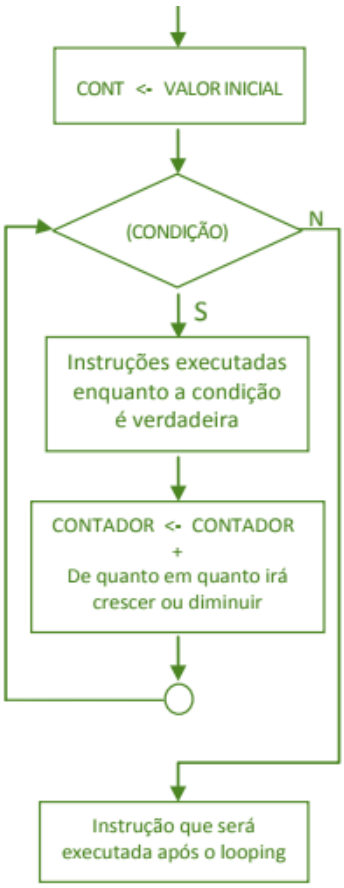
Os programas computacionais estão por toda parte. Praticamente todos os estabelecimentos, sejam eles comerciais, industriais, acadêmicos ou qualquer outro, fazem uso de programas que automatizem suas rotinas. Para ser um indivíduo competitivo no mercado de trabalho e se tornar “empregável” no desenvolvimento de softwares computacionais, é extremamente importante que você conheça o modo como o computador entende e implementa as estruturas de repetição dentro da lógica de programação.

PARA COMEÇAR O ASSUNTO...

Vimos na última aula que é possível “instruir” o computador a prosseguir em um determinado programa a partir de uma condição. Ele pode identificar se a condição é verdadeira e executar uma(s) determinada(s) instrução(ões); ou falsa e seguir pelo caminho contrário.

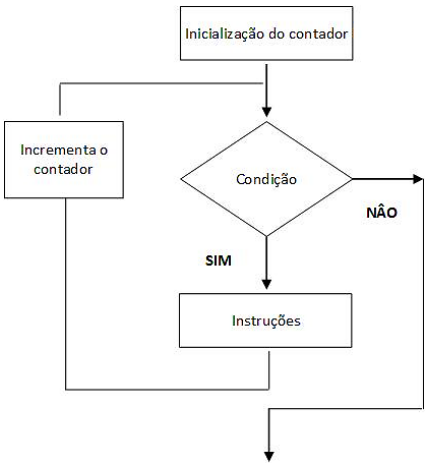
Esta rotina, porém, acontece uma única vez. O programa seleciona a opção (falso ou verdadeiro) e executa as instruções uma vez.

Mas, e se for preciso executar estas instruções mais de uma vez? Para responder esta pergunta, utilizamos as estruturas de repetição que, como o próprio nome diz, repetem um bloco de instruções enquanto uma condição for verdadeira; ou então, repetem um bloco de instruções enquanto uma dada condição for falsa. Você achou complicado? Não! É só dar uma olhada no “mergulhando no tema” para entender direitinho como funciona.



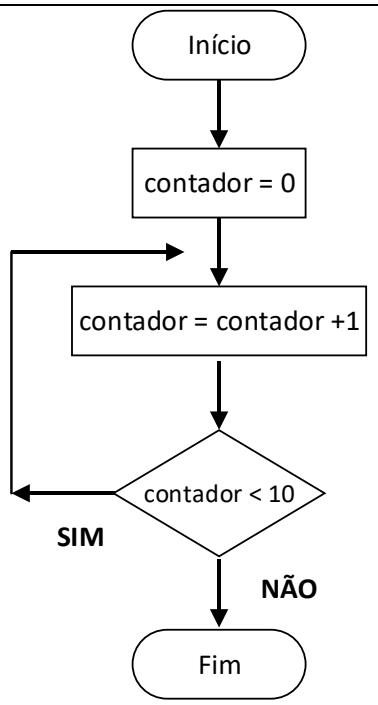
MERGULHANDO NO TEMA...

Além dos desvios sequenciais, é possível criar desvios em loop ou repetição, ou seja, repetir trechos do algoritmo sobre determinada condição e controlar a forma com que serão executados. O comando "para...fim-para" permite que uma variável realize a contagem do número de repetição a executar, conforme a indicação inicial e final dessa contagem e também indique o formato em que essa tarefa será realizada. Observe o quadro a seguir e veja as sintaxes em pseudocódigo quanto em Java são parecidas:

Pseudocódigo	Fluxograma	Java
Para {variável} = <valor inicial> **até** <valor final> **passo** <argumento> **faça** {comando(s)} **fim-para**		**for (int i=0; i <10; i++)** { System.out.println(i**);** **}**

Observe que no Fluxograma aparece a palavra contador e tanto no Pseudocódigo quanto no Java não aparecem. Calma! Ela aparece, porém de outra forma. Veja que no Pseudocódigo tem {variável} e no Java aparece i. Ambos são os contadores que aparece no Fluxograma.

A execução é muito fácil. Veja um exemplo para melhor compreensão:

Pseudocódigo	Fluxograma
Programa mostrar números de 1 até 10 Declare numero como inteiro Início Leia(numero) Para numero ←1 até 10 passos 1 repetir Mostrar numero Fim para Fim.	

Apenas para você assimilar melhor, o pseudocódigo inicia-se com a leitura da variável **numero** com tipo de dados inteiro, ou seja, esta variável irá receber apenas números inteiros. Após esta leitura desta variável, inicia-se a execução da estrutura de repetição **PARA** onde realizará a “contagem” dos números um por vez (está para repetir 1 por vez) até chegar no número 09. Ao chegar no limite declarado (número 9) irá mostrar os números da sequência de 1 até 10.

No exemplo anterior, tanto o Pseudocódigo quanto no Fluxograma, não fazemos a contagem até o número 10. Por que? Você já deve ter visto este assunto em algumas agendas anteriores em que foi informado sobre os operadores (aritméticos, comparação e lógicos).

Como a condição é:

Para numero ← 0 até 09 (Pseudocódigo) ou contador < 10 (Fluxograma)
então irá contar do 0 até 09, totalizando 10 contagens.
Porém se estivesse o operador = (igual),
então, estaria contando do 0 até 10 (contador <=10) , totalizando 11 contagens.

Veja agora o código em Java:

lacofor.java

```java
package lacos;

import javax.swing.JOptionPane;

public class lacofor {

    public static void main(String[] args) {
        // declaração das variáveis.

        int i; //variável que será utilizada para o laço de repetição for.

        for (i=0; i<10; i++) { //uso da estrutura do for.
            JOptionPane.showMessageDialog(null, i); // saída de dados.
        } // fecha o laço de repetição for.

    } // fecha o método.

} // fecha a classe.

```

Correspondendo ao Pseudocódigo informado anteriormente, no Java, declaramos as variáveis que iremos utilizar. Neste caso, apenas o i como inteiro. Em seguida iniciaremos o laço de repetição FOR com a variável inicial 0 (zero), variável final 9 (nove) e o incremento com 1. Ou seja, o resultado irá mostrar as sequências dos números 0, 1, 2, 3, 4, 5, 6, 7, 8 e 9.

VOCÊ NO COMANDO

Pense em como poderíamos elaborar um programa utilizando a estrutura para...fim-para em que o usuário mostre os números de 01 até 10 na tela. Reflita como podemos resolver essa questão antes de prosseguir a leitura.

Pseudocódigo:

Programa mostrar números

```
Programa mostrar numeros
Declare
    num como inteiro
Início
    Para num ← 1 até 10 faça
        passo  1 repetir
        Mostrar num
    Fim para
Fim.
```

No Java:

*ex02.java

```
1
2  public class ex02 {
3
4      public static void main(String[] args) {
5
6          System.out.println("Os números de 0 a 10 são:");
7              for (int i=0; i<=10; i++){ //laço de repetição FOR
8          System.out.println (i); // saída de dados
9              }
10     }
11 }
12
```

Resultado:

Problems @ Javadoc Declaration Console

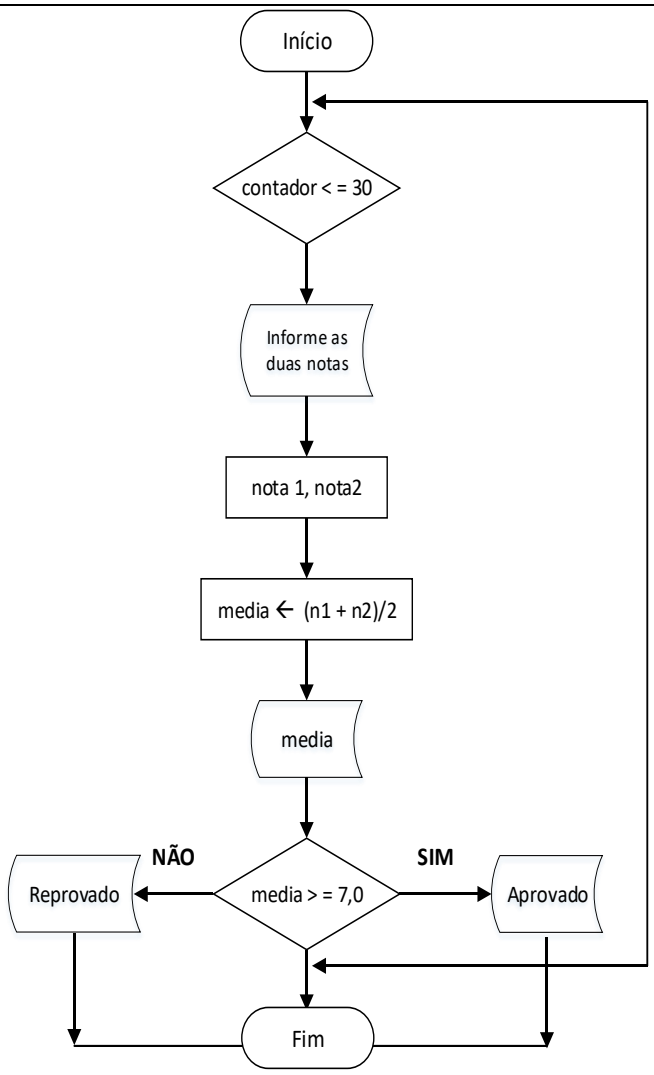
```
<terminated> ex02 [Java Application] C:\Program Files\Java\jre1.8.0_151\
Os números de 0 a 10 são:
0
1
2
3
4
5
6
7
8
9
10
```

Observe que apenas usamos a sintaxe System.out.println nas linhas 6 e 8 do código para que a informação apareça no Console. Além disso, a estrutura do FOR iniciou com valor 0 e foi até 10 com passos 1 (linha 7 do código), ou seja, ele andou de um em um, como mostra a tela do resultado.

O Laço de Repetição “para...fim-para” é muito simples e fácil de usar. Porém se você quiser incrementar o seu uso conseguimos utilizá-lo junto com a estrutura “se...senão...fim_se”.

VOCÊ NO COMANDO

Pense em uma situação em que você precise calcular a média de 30 alunos e informar a situação de cada um deles, ou seja, informar a média como também se cada um dos 30 alunos estão “Aprovados” ou “Reprovados”. Reflita sobre este novo conceito antes de prosseguir com a leitura.

Pseudocódigo	Fluxograma
Programa mostrar media e situacao Declare n1, n2, media como real contador como inteiro Início Para contador ← 1 até 30 faça passo 1 repetir escreva (“Informe sua primeira nota:”) ler (nota1) escreva (“Informe sua primeira nota:”) ler (nota2) media ← (n1 + n2)/2 escreva (“A sua média é: “ , media) se (media >=7,0) entao escreva (“Aprovado”) senao escreva (“Reprovado”) Fim se Fim para Fim.	

No Java:

*ex03.java *ex06.java

```
1  import java.util.Scanner;
2  public class ex06 {
3
4      public static void main(String[] args) {
5
6          double n1, n2, media;
7          n1 = 0;
8          n2 = 0;
9          //utilizado Scanner como visto nas agendas anteriores.
10         Scanner ler = new Scanner(System.in);
11         for(int i=1; i<=30; i++){ //laço de repetição para os 30 alunos.
12             System.out.printf("Digite a primeira nota:" , n1);
13             n1 = ler.nextDouble();
14         //usuário irá digitar a nota e este comando irá ler e armazenar na variável.
15             System.out.printf("Digite a segunda nota:" , n2);
16             n2 = ler.nextDouble();
17             media = (n1 + n2)/2; //cálculo da média.
18                 if(media >= 7.0) {
19             //coloca a decisão para dizer se está aprovado ou reprovado.
20                     System.out.println("Sua situação é: Aprovada " + media);
21                 }
22                 else {
23                     System.out.println("Sua situação é: Reprovada " + media);
24                 }
25             }
26         }
27     }
```

Observe que utilizamos o Laço de Repetição PARA (linha 11) e o Laço de Decisão (linha 18) para que resolvesse o problema informado anteriormente. Além disso, utilizamos o Scanner (linhas 1, 10, 13 e 16) para que o usuário pudesse digitar as notas e o programa realizar todo cálculo.

Repare que estamos utilizando diversos conceitos aprendidos nas agendas anteriores, ou seja, tudo que você está estudando é acumulativo e poderá ser usado tudo de uma vez para que realize um melhor Programa para solução do problema.

Teste de Mesa

Para que possamos nos certificar de que o algoritmo realizado está correto, antes de passar para a linguagem de programação (no nosso caso, Java) podemos testá-lo simulando valores e verificar se o resultado é o esperado. Esta simulação não é realizada no computador utilizando nenhum software. É realizada no papel.

Calma! O Teste de Mesa é muito simples de realizar. Basta montar uma pequena tabela e começar a simular os valores utilizando o seu pseudocódigo.

Para exemplificar, vamos voltar no algoritmo anterior (Mostrar os números de 01 até 09 com passo 01) e aplicar o Teste de Mesa.

contador	resultado
0	0
0+1	1
1+1	2
2+1	3
3+1	4
4+1	5
5+1	6
6+1	7
7+1	8
8+1	9

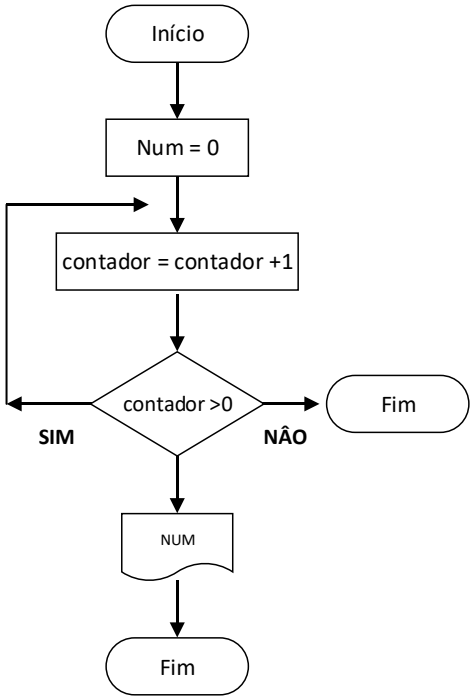
Veja que você iniciou com zero e como o exercício pede o incremento, ou seja, precisa somar 1 ao resultado anterior.

E como foi dito anteriormente, como o valor final é i<10, então iremos até o valor 9.

1. Elabore o Algoritmo, o Fluxograma e um Programa em Java que some todos os números no intervalo de 0 até 100.
2. Elabore o Algoritmo e um Programa em Java que exiba a Tabuada dos números 1 até 8.
3. Criar o Algoritmo e um Programa em Java que mostre o quadrado dos números de 1 até 5.

Confira se você conseguiu resolver os desafios....

Respostas:1.

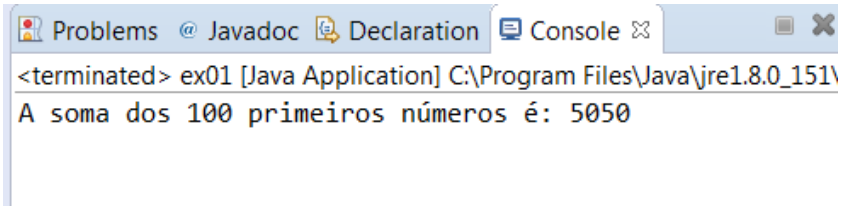
Pseudocódigo	Fluxograma
Programa ex1 Declare num como inteiro Início Escreva (“Entre com um número”) Leia (numero) Para (num de 1 até 1000 repita passo 1) Faça (mostrar num) fim-para	

No Java:

*ex01.java

```
1
2  public class ex01 {
3
4      public static void main(String[] args) {
5
6      int soma = 0; //declarando a variável como inteira e
7                    //inicializando com valor zero.
8
9      System.out.print("A soma dos 100 primeiros números é: ");
10     //A frase que está aparecendo na Saída de dados.
11
12          for(int i = 1; i<=100; i++) // Laço de repetição FOR
13              soma += i; //soma = soma + i
14             System.out.println(soma); // saída de resultados.
15
16         }
17     }
```

Resultado:



2.

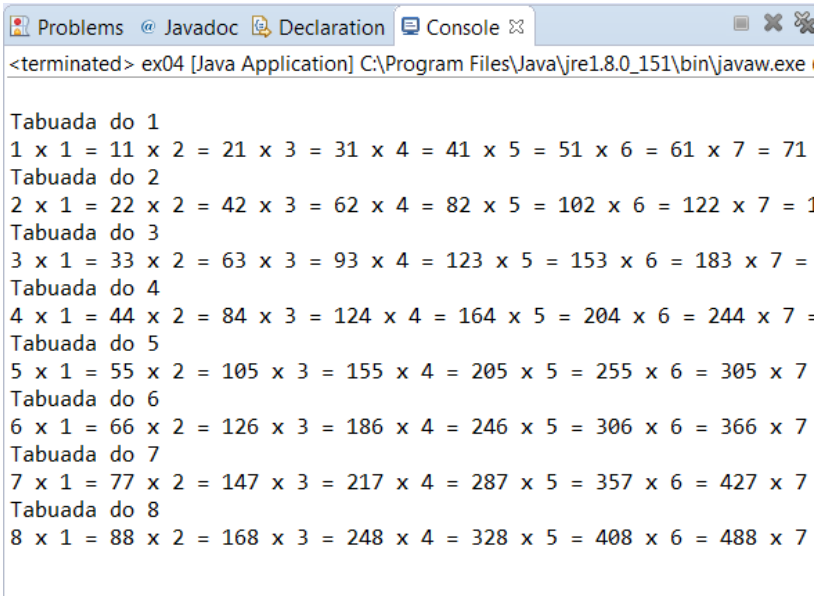
```
Programa ex4

Declare
   X, i, res como inteiro
Início
   Leia (x)
   Para (i de 1 até 10 repita passo 1)
       Res ← x * i
       Escreva (x, “*”, i, “=”, res)
    fim-para
fim
```

No Java:

ex01.java | *ex04.java

```
1
2  public class ex04 {
3
4      public static void main(String[] args) {
5  // Laço de repetição que determinará qual número será a tabuado. Ex.:
6  // Tabuada do 2 ou do 3 e assim por diante até o 8.
7          for (int I = 1; I < 9; I++){
8              System.out.println("\nTabuada do "+I);
9              // Mostra qual número será a Tabuada.
10             for (int N = 1; N <= 10; N++){
11 //Laço de repetição que determinará qual o número que será multiplicado.
12                 System.out.print(I+" x "+N+" = "+I*N);// saída de dados.
13
14 }
15   }
16     }
17 }
```

Resultado:



3.

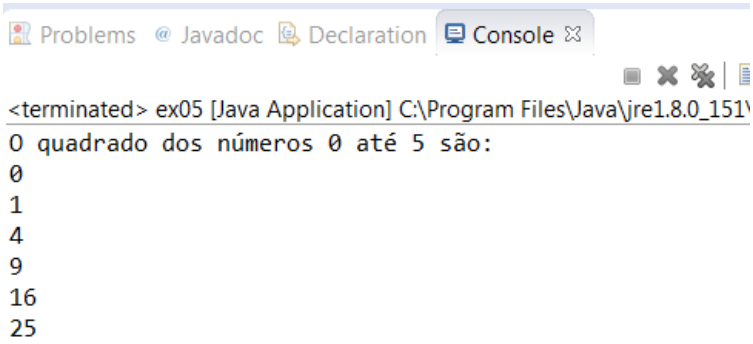
```
Programa ex5
Declare
   x, i  como inteiro
Início
   Para (i de 1 até 5 repita passo 1)
       Escreva (“Informe o”, i, “° número: “)
       Leia (x)
       Escreva (x, “^2 = ”, x ^2)
    fim-para
fim
```

No Java:

*ex05.java

```
1
2  public class ex05 {
3
4      public static void main(String[] args) {
5
6          int x; //declaração da variável x (resultado) como inteira.
7          System.out.println("O quadrado dos números 0 até 5 são:");
8          //apenas informando ao usuário a frase acima.
9              for (int i=0; i<=5; i++){ //laço de repetição FOR
10             x= i*i;
11 //elevar o número ao quadrado, ou seja, multiplicar ele por ele mesmo
12             System.out.println (x); // saída de dados
13                 }
14         }
15     }
```

Resultado:



Estes exercícios devem ser entregues de forma on-line como atividades da agenda.

Exercício 1

Joaquim tem uma Padaria e seu aniversário está próximo. Ele e sua família decidiram além de fazer a tradicional comemoração, irão selecionar 10 produtos na Padaria para fazer a promoção da semana de 50% desses produtos.

Escreva um Algoritmo, faça o Fluxograma e um Programa em Java solicitando a entrada de 10 números inteiros e apresentando a metade de cada número.

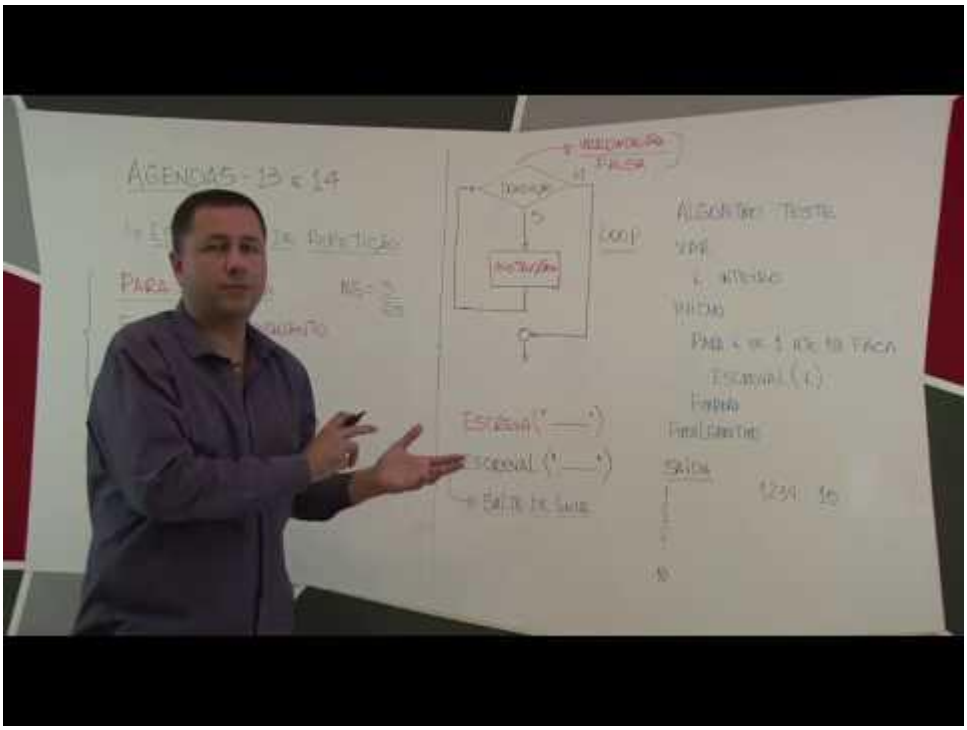
Exercício 2

Giovanna adora usar computador, navegar na internet e matemática. Ela decidiu estudar tabuada devido sua dificuldade durante o período de aula (ela apenas sabe as tabuadas dos números 0 até 05).

Ao navegar na internet aprendeu a desenvolver alguns algoritmos e decidiu desenvolver algo para ajudá-la em seus estudos com tabuada.

Escreva para Giovanna um Algoritmo e um Programa em Java da Tabuada dos números 6 até 10.

Não deixe também de assistir ao vídeo:
Informática - Módulo I - Agendas 13 e 14



Link: https://www.youtube.com/watch?v=MGZtsa9U58Q. Acessado em 25/11/2017.

Para aprofundamento dos temas discutidos nesta aula, seguem abaixo algumas dicas de filmes, livros e artigos que se relacionam com o conteúdo estudado. Estas dicas são muito importantes para você!

Vídeo:

Procure no Youtube um vídeo denominado “Curso de Java #13 – Estruturas de Repetição (Parte3)”, disponível em : https://www.youtube.com/watch?v=XLqPZh6n8IA. Acessado em 25/11/2017.

Livros:

FORBELLONE, André L. V.; ELBERSPACHER, Henri Frederico. Lógica de Programação: A Construção de Algoritmos e Estrutura de Dados. Editora Pearson, 2000.

PUGA, SANDRA; RISSETTI, GERSON. Lógica de Programação e Estruturas de Dados com Aplicações em Java. Editora Pearson. 2009

SCHILDT, HEBERT. Java para Iniciantes. Editora Bookman.2015

AGENDA 14
ESTRUTURAS DE REPETIÇÃO “ENQUANTO... FIM-ENQUANTO” E “REPITA... ATÉ QUE”

MOMENTO DE REFLEXÃO

Imagem 22

(Fonte: http://bichinhosdejardim.com/repeticao/ acesso em dezembro/2016)

Você reparou que fazemos diversas atividades repetidas ao longo da vida? E no seu cotidiano? Todo dia você acorda em um determinado horário, come algo, pega ônibus ou metrô ou trem ou os três para chegar na escola, assisti a aula, faz as tarefas, volta da escola e assim vai. Se pararmos um pouco e pensarmos quantas atividades repetimos... imagine o computador!

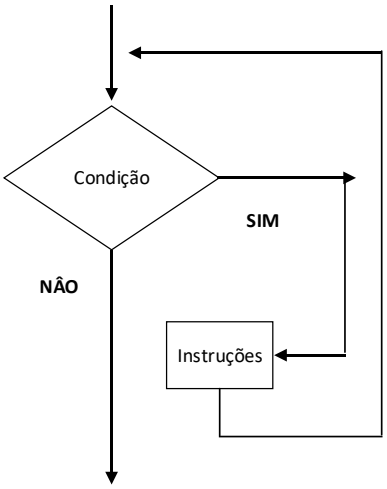
Ainda bem que para o computador, tem a Lógica de Programação e as Estruturas de Repetição que somadas ajudam a desenvolver diversos algoritmos para que estas atividades rotineiras fiquem mais fáceis e práticas.

Na agenda passada você conheceu uma das Estruturas de Repetição: **“PARA...FIM-PARA”** e agora irá conhecer outras duas que também são importantes e com certeza nas próximas agendas e módulos ajudará bastante na Programação.

POR QUE APRENDER?

Na última agenda estudamos o Laço de Repetição **“PARA...FIM-PARA”** e dando sequência, você estudará nesta agenda os outros dois tipos de Laço de Repetição: **“ENQUANTO... FIM-ENQUANTO” E “REPITA... ATÉ QUE”** que tem características diferentes e irão facilitar bastante na solução de diversos problemas que você encontrará ao longo da Programação. Com isso finalizamos esta estrutura que é muito usual e prática na programação.

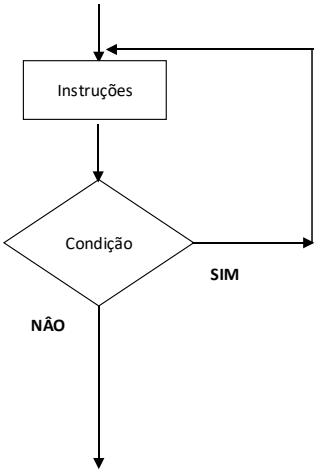
PARA COMEÇAR O ASSUNTO...



Já que estudaremos os dois últimos tipos do Laço de Repetição nesta agenda, que tal começar a separar antes que façamos uma pequena bagunça?

Primeiro vamos para o **“ENQUANTO... FIM-ENQUANTO”** conhecida também como somente **“ENQUANTO”** (While).

Esta estrutura habitualmente utilizamos quando não sabemos o número de repetições que ocorrerá e só encerra a sua execução quando a condição é satisfeita. A imagem ao lado representa exatamente esta informação.



Agora o **“REPITA... ATÉ QUE”** conhecida também como Do...While. Esta estrutura é muito parecida com a primeira e por conta disso existem vários Programadores que não a usam.

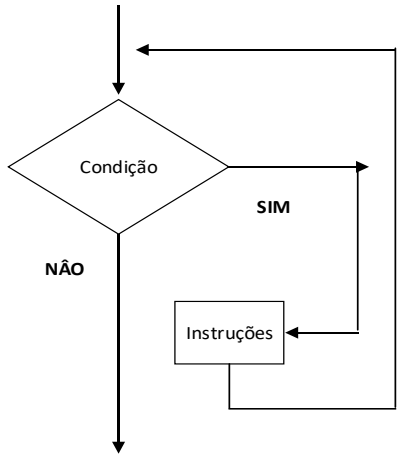
Esta estrutura é utilizada quando, também, não conhecemos o número de repetições que ocorrerá e só encerra, também, quando uma condição é satisfeita. A diferença desta para a primeira é que a condição é testada por último fazendo com que, pelo menos, uma vez este trecho seja executado.

Mas fique tranquilo, pois agora em “Mergulhando no Tema” iremos praticar um pouco mais sobre estes laços.

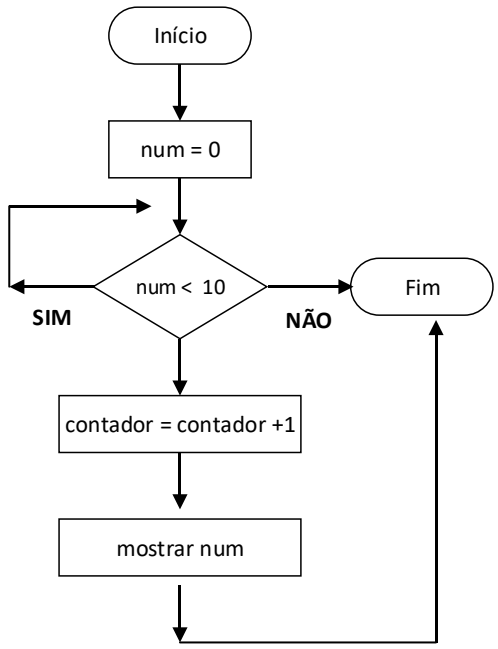
MERGULHANDO NO TEMA...

“ENQUANTO... FIM-ENQUANTO”

Este Laço de Repetição, como informado anteriormente, trabalha enquanto a condição for verdadeira e vai executando as instruções. Porém a condição sendo falsa, ele sai do loop e vai para o próximo comando na programação.

Pseudocódigo	Fluxograma	Java
Enquanto <condição> **faça** <Comando> <Comando> <Comando> **Fim enquanto**		**while (condição) {** instrução **}**

Sabendo da definição e estrutura deles, vamos ver um exemplo?
Elabore um Algoritmo, Fluxograma e um Programa em Java que mostre todos os números menores que 10.

Pseudocódigo	Fluxograma
Programa ex01 Declare num como inteiro Início Enquanto (num < 10) faça Mostre num fim-enquanto	

No Java:

*ex01.java

```
1  public class ex01 {
2
3      public static void main(String[] args) {
4          int num = 0;
5          while (num < 10) {
6              System.out.println("Número " + num);
7              num++;
8          }
9      }
10 }
```

Observe que na linha 5 temos a estrutura de repetição Enquanto (while) tendo como condição a situação que o exercício colocou (números menores que 10). As linhas 6 e 7 são exatamente os comandos que são executados dentro desta estrutura.

Resultado:

Problems Javadoc Declaration Console

```
<terminated> ex01 (1) [Java Application] C:\Program Files\Java\jre1.8.0_151\
Número 0
Número 1
Número 2
Número 3
Número 4
Número 5
Número 6
Número 7
Número 8
Número 9
```

Dica! A condição colocada acima foi num < 10 e por isso você deve ter percebido no resultado que os números que saíram foram do 0 até o 9.

VOCÊ NO COMANDO

Você viu que o comando ENQUANTO é muito simples. Como no exemplo acima, foi mostrado apenas os números de 0 até 9, será que você consegue fazer com que mostre no Java os números de 0 até 10?

Dica: este assunto já foi abordado na agenda anterior!

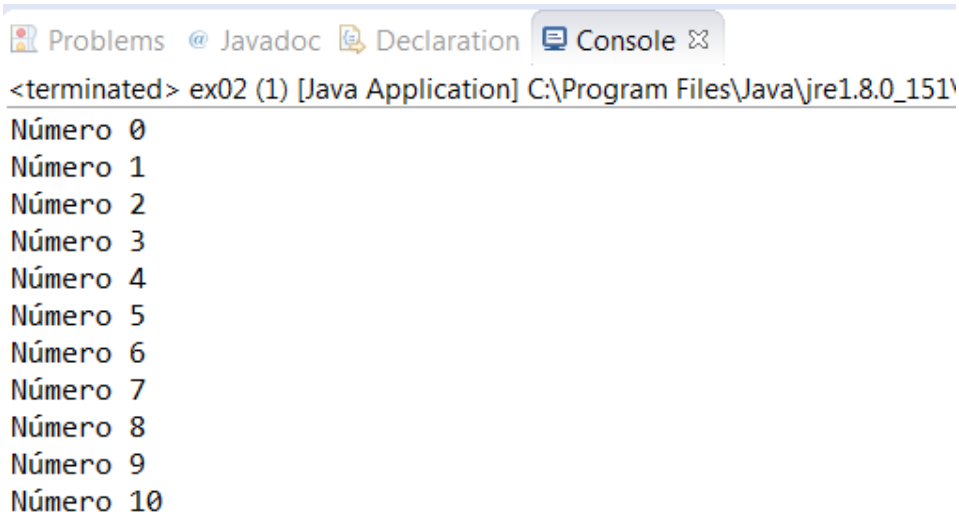
No Java:

*ex02.java

```
1  public class ex02 {
2
3      public static void main(String[] args) {
4              int num = 0;
5              while (num <= 10) {
6                  System.out.println("Número " + num);
7                  num++;
8              }
9          }
10     }
11
```

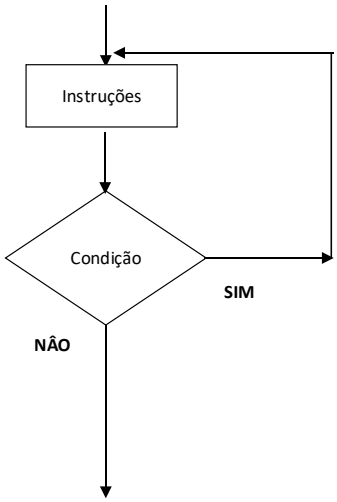
A diferença deste código no Java com o anterior é apenas o operador igual (=). Ao colocar a condição < = 10 você incluiu o 10 na condição e o resultado (mostrando a seguir) aparecem os números de 0 a 10.

Resultado:

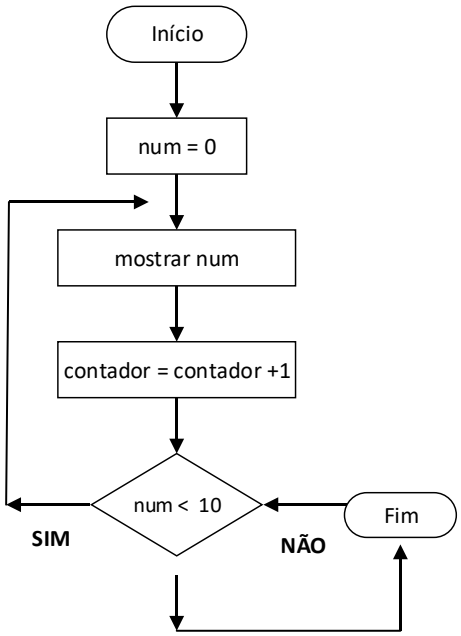


“REPITA... ATÉ QUE”

Esta estrutura, como informado, anteriormente, é parecida com a primeira. A única diferença é que a condição dela é executada por último. Vamos aos detalhe para entender melhor?

Pseudocódigo	Fluxograma	Java
Repetir <Comando> <Comando> <Comando> **até** <condição>		**do {** instrução **}** **while** (condição)

Agora que você já conhece um pouco mais esta estrutura, vamos voltar ao exercício anterior e fazer nesta estrutura: Elabore um Algoritmo, Fluxograma e um Programa em Java que mostre todos os números menores que 10.

Pseudocódigo	Fluxograma
Programa ex01 Declare num como inteiro Início repetir Mostre num até (num < 10) fim-repetir	

No Java:

*ex03.java *ex06.java ex1.java ex01.java

```
1
2  public class ex1 {
3
4      public static void main(String[] args) {
5          int num = 0;
6          do{
7            System.out.println("Número " + num);
8            num++;
9          }
10         while (num < 10);
11
12         }
13     }
14
```

Observe que estamos com o mesmo exercício e o resultado é o mesmo, porém com outro Laço de repetição. Nas linhas 6 até 10 é a sintaxe do REPITA ATÉ.

Na agenda passada foi informado que podemos misturar o PARA (for) com o SE (if) para que possamos atingir a solução de um problema na programação. Assim, como no PARA, podemos, também, usar o ENQUANTO (while) e o REPITA ATÉ (do while) com o SE (if) com o mesmo propósito.

Exemplo: Desenvolver um Programa em Java onde é solicitado ao usuário o seu nome, cargo (Diretor, Gerente ou Operacional) e salário. Seguindo a tabela abaixo é feito um cálculo para verificar se o funcionário tem direito ao empréstimo e qual o valor.

Diretor = 30%	Gerente = 25%	Operacional = 20%

*ex03.java

```
1  import java.util.Scanner;
2  public class ex03 {
3
4      public static void main(String[] args) {
5
6          String nome, cargo;
7          float sal, emprestimo=0;
8
9          System.out.println("Por favor informe seu nome");
10         nome = new Scanner(System.in).nextLine();
11         System.out.println("E agora informe seu cargo (Diretor, Gerente ou Operacional");
12         cargo = new Scanner(System.in).nextLine();
13
14         while (cargo == "Diretor" || cargo == "Gerente" || cargo =="Operacional"){
15           System.out.println("Cargo incorreto, por favor informe o cargo novamente");
16           cargo = new Scanner(System.in).nextLine();
17           }
18
19         System.out.println("Agora informe o seu salário");
20         sal = new Scanner(System.in).nextFloat();
21             if (cargo.equals("Gerente"))
22               emprestimo = sal*25/100;
23             else if (cargo.equals("Diretor"))
24               emprestimo = sal*30/100;
25             else
26               emprestimo = sal*20/100;
27
28         System.out.println("Olá "+nome);
29         System.out.print("Seu cargo é "+cargo+", ");
30         System.out.println("Seu sálario é "+sal);
31         System.out.println("E você tem direito a pegar R$"+emprestimo+" de empréstimo.");
32     }
33 }
34
```

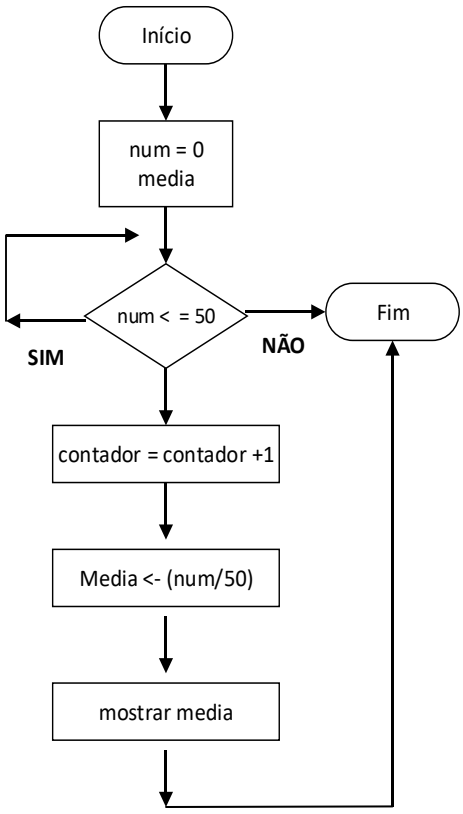
No exemplo acima, observamos que além dos usos do Laço de repetição WHILE (linhas 14 até 17), da Estrutura de seleção IF (linhas 21 até 26), foi utilizado, também, o Scanner (linhas 1, 10, 12, 16 e 21).

1. Faça o Fluxograma e um Programa em Java que leia 50 números, calcule e exiba a média aritmética dele.
2. Faça um Programa em Java para calcular a soma dos dígitos de um número. Por exemplo: N°: 21 -> 2+1 = 3
3. Faça um Programa em Java para realizar comparação de números e informar o número maior. Para este código será necessário solicitar ao usuário digitar a quantidade de números que ele quer comparar e ele deverá digitar os números.

Respostas:

1.

Fluxograma:



Java:

ex03.java *ex04.java

```java

public class ex04 {

    public static void main(String[] args) {
        int num = 0;
        float media;

        while (num <= 50) {
            //System.out.println("Número " + num);
            num++;
            }
        media = (num/50);
        System.out.println("A média é: " + media);
    }
}
```

2.

ex1.java *ex05.java

```java
import java.util.Scanner;
public class ex05 {

    public static void main(String[] args) {

        int num;
        int soma = 0;
        Scanner ler = new Scanner(System.in);
        System.out.println("Digite um número qualquer:");

        num = ler.nextInt();
        while (num > 0) {
            num = num /10;
            soma = soma + (num % 10);
        }
    System.out.printf ("A soma dos dígitos do número que você digitou é: " , soma);
    }
}
```

3.

*ex06.java

```java
import java.util.Scanner;
public class ex06 {

    public static void main(String[] args) {
        int x=2; //iniciando a variável com valor.
        int num;
        int maior=0;
        int compara;

        Scanner entrada = new Scanner (System.in); //utilizando Scanner para capturar dados.
        System.out.println("Digite quantos números você quer comparar: ");
        compara = entrada.nextInt();
        System.out.printf("Insira o primeiro número: ", maior);
            maior = entrada.nextInt();

            while (x <= compara) { //Laço de repetição While.
                System.out.println("Digite " + x + " numero");
                num = entrada.nextInt();

              if (num > maior) { //Decisão
                 maior = num;
              }
               x++;
              }
                  System.out.println("O maior número digitado foi " + maior );
          }
      }
```

Estes exercícios devem ser entregues de forma on-line como atividades da agenda.

Exercício 1

Na Agenda 11, você realizou este exercício com a Estrutura de Decisão. Agora, realize com a Estrutura de Repetição: while ou do while (você pode escolher).

A empresa NewInfo está desenvolvendo um sistema para classificar a prioridade na fila de espera de atendimento de um de seus clientes. A classificação da prioridade consiste em perguntar a idade do usuário a ser atendido e encaminhá-lo para uma fila prioritária caso este seja maior de 60 anos.
Elabore um programa que peça para o utilizador inserir a sua idade e exiba na tela a mensagem “fila prioritária” caso este tenha mais de 60 anos ou exiba “fila comum” caso contrário. Apresente a resposta em pseudocódigo, fluxograma e linguagem de programação Java.

Exercício 2

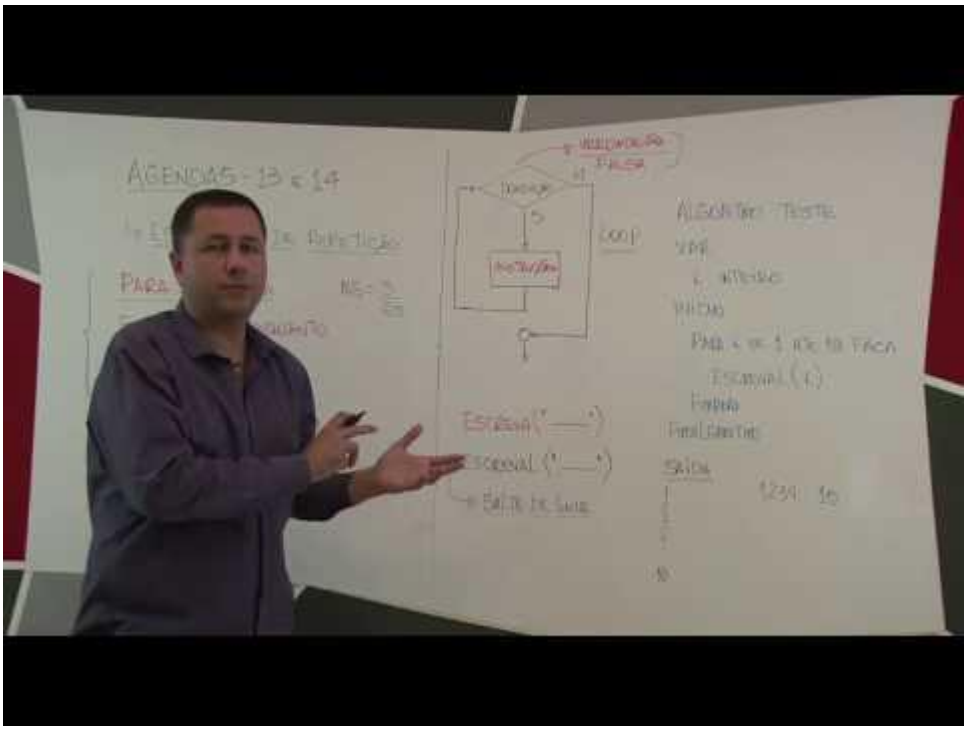
Na Agenda 13, você realizou este exercício com a Estrutura de Repetição PARA (for). Agora, realize com a Estrutura de Repetição: while ou do while (você pode escolher).

Giovanna adora usar computador, navegar na internet e matemática. Ela decidiu estudar tabuada devido sua dificuldade durante o período de aula (ela apenas sabe as tabuadas dos números 0 até 05). Ao navegar na internet aprendeu a desenvolver alguns algoritmos e decidiu desenvolver algo para ajuda-la em seus estudos com tabuada.
Escreva para Giovanna um Algoritmo e um Programa em Java da Tabuada dos números 6 até 10.

AMPLIANDO HORIZONTES

Não deixe também de assistir ao vídeo:

Informática - Módulo I - Agendas 13 e 14



Link: https://www.youtube.com/watch?v=MGZtsa9U58Q. Acessado em 25/11/2017.

Curso de Java - Correção Exercícios Aula 17 (while, do-while, for) Parte 1

Link: https://www.youtube.com/watch?time_continue=2&v=7ccdc5Vkf7Q. Acessado em 02/12/2017

Para aprofundamento dos temas discutidos nesta aula, seguem abaixo algumas dicas de filmes, livros e artigos que se relacionam com o conteúdo estudado. Estas dicas são muito importantes para você!

Vídeo:

Procure no Youtube um vídeo denominado “Curso de Java #11 – Estruturas de Repetição (Parte1)”, disponível em : https://www.youtube.com/watch?v=2fawKjR8d4c. Acessado em 30/11/2017.

Procure no Youtube um vídeo denominado “Curso de Java #12 – Estruturas de Repetição (Parte2)”, disponível em : https://www.youtube.com/watch?v=ojLALwmvQIU. Acessado em 30/11/2017.

Livros:

FORBELLONE, André L. V.; ELBERSPACHER, Henri Frederico. Lógica de Programação: A Construção de Algoritmos e Estrutura de Dados. Editora Pearson, 2000.

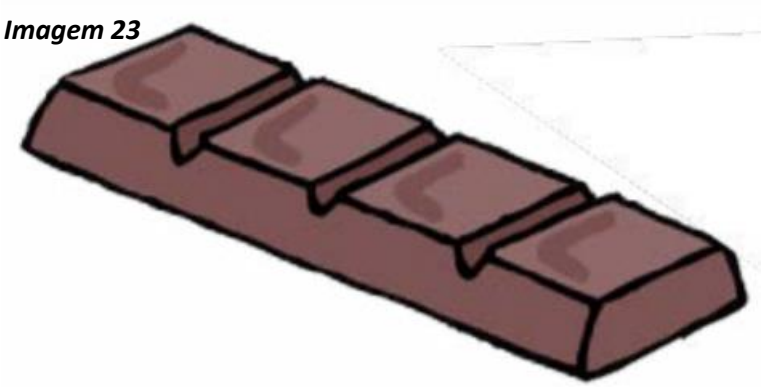
PUGA, SANDRA; RISSETTI, GERSON. Lógica de Programação e Estruturas de Dados com Aplicações em Java. Editora Pearson. 2009

SCHILDT, HEBERT. Java para Iniciantes. Editora Bookman.2015

AGENDA 15
VETORES

MOMENTO DE REFLEXÃO

Imagem 23



Você certamente já viu uma barra de chocolate não é mesmo? Mais do que isso – se tiver o mesmo apreço por chocolate que a maioria das pessoas, será capaz até de citar marcas, formatos e texturas desta guloseima. Entre estes formatos está o mais conhecido – aquele em que podemos visualizar os “quadradinhos” que formam uma barra. Assim, quando pensamos em uma barra de chocolate, entendemos que uma mesma barra é composta por pedaços menores.

Esses pedaços podem ter recheios que, via de regra, são de um único sabor (normalmente uma calda de morango). Antes que você corra para o supermercado mais próximo da sua casa a fim de comprar uma barra de chocolate, vamos entender o que isso tem a ver com esta aula. Similarmente à barra de chocolate, variáveis também podem armazenar vários valores, como se fossem o recheio armazenado nos “pedaços” desta barra.

Seguindo esta mesma analogia, esses valores devem obrigatoriamente ser do mesmo tipo, como o recheio sempre do mesmo sabor, lembra-se?.

Às variáveis que possuem estas particularidades chamamos vetores, as quais são o foco desta agenda.

Até então trabalhamos com variáveis simples, que só recebem um dado por vez. Para atender às exigências da programação, precisamos aprender a lidar com uma técnica de programação que nos permitirá trabalhar com um agrupamento de várias informações dentro de uma mesma variável, tornando a programação muito mais rápida e eficiente. É por isto que nesta agenda estudaremos sobre a prática do uso de Vetores.

Vamos imaginar uma Barra de chocolate, destas que são divididas em vários quadradinhos. Você compra os quadradinhos do chocolate individualmente, cada um em sua própria embalagem? Provavelmente não. Você certamente faria a opção por comprar uma barra completa de uma única vez, pela comodidade e também pelo preço, que acaba saindo mais barato não é mesmo?

Quando trabalhamos com vetores a ideia é praticamente a mesma. Juntamos vários valores em uma única variável, cada uma com o seu espaço, similar a barra de chocolate, com o intuito de facilitar o acesso aos dados e economia de espaço em memória.
Com esta facilidade, o processo de leitura e gravação de um dado em uma variável, torna-se simplificado graças as estruturas de repetição. Como esta simplificação será feita? É o que veremos durante este capítulo. Vamos lá?

PARA COMEÇAR O ASSUNTO...

Imagem 24

Jonas, um adepto da tecnologia e um estudante de Lógica de programação, recebeu uma importante tarefa de seus pais, a de criar um pequeno programa, onde fosse possível inserir os nomes dos 5 amigos restantes de seu avô Marco e o programa gerar uma pequena mensagem, homenageando-os por serem amigos de Marco por tanto tempo.

Confeccionando o Programa, Jonas percebeu que o que parecia simples se tornará muito extenso, uma vez que devido ao fato de cada nome ficar em uma variável, será impossível utilizar uma estrutura de repetição para satisfazer o problema.

Neste caso, como Jonas poderá entregar ao seu avô a homenagem aos seus amigos em tempo hábil e de forma eficiente?

MERGULHANDO NO TEMA...

Durante seus estudos de Lógica de Programação, você já passou por alguma situação onde seja necessária a declaração de muitas variáveis para o mesmo propósito?

Alguma vez você já se perguntou se para calcular a média entre 10 notas de um aluno seria necessário declarar 10 variáveis?

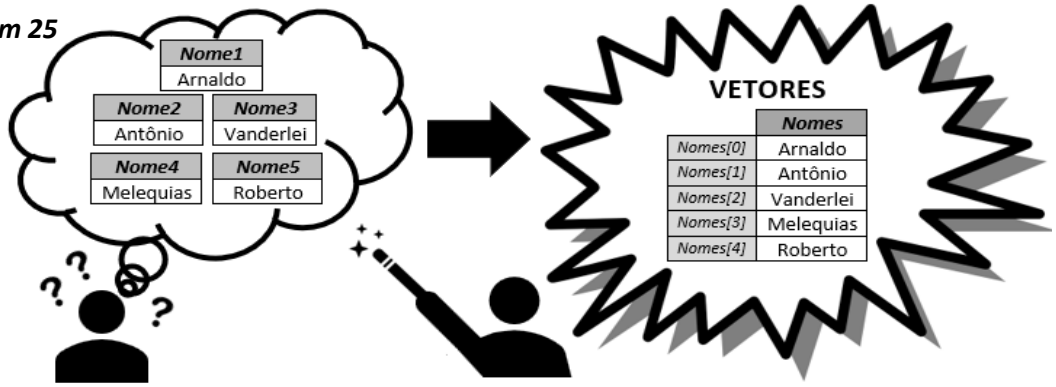
Até então trabalhamos com variáveis simples, que só recebem um dado por vez. Para atender às exigências da programação, precisamos aprender a lidar com uma técnica de programação que nos permitirá trabalhar com um agrupamento de várias informações dentro de uma mesma variável, tornando a programação muito mais rápida e eficiente. É por isto que neste capítulo estudaremos sobre a prática do uso de Vetores, que são variáveis especiais, com a capacidade de receber diversos valores.

Agora que você já aprendeu a manipular variáveis e as principais estruturas de todas as linguagens de programação, chegou a hora de aprender a trabalhar com um tipo especial de variável: os vetores.

Vetores, ou Matrizes Unidimensionais são variáveis diferentes, nela podemos receber mais de um valor, desde que todos os valores sejam do mesmo tipo (int, double, String etc).

Vetores foram desenvolvidos para que leituras e gravações de dados repetitivos sejam simplificados através das estruturas de repetição, uma vez que valores individuais devem sempre ser lidos individualmente, não sendo possível ler N variáveis dentro de uma única estrutura de repetição.

Imagem 25



VOCÊ NO COMANDO

Agora que já vimos que vetores são variáveis que podem receber mais de um valor, em quais casos já estudados podemos substituir variáveis repetidas por vetores? Reflita sobre este novo conceito antes de prosseguir com a leitura.

Para entender melhor a importância deste tipo de variável, vamos resolver a seguinte situação problema:

Jennifer, uma professora do Curso Técnico em Eventos e iniciada em Programação de Computadores, decidiu criar um programa onde seja confeccionada automaticamente uma lista de convidados para o próximo evento da turma do 1º Módulo do Curso. Para testar se o programa funcionará corretamente, Jennifer deseja primeiramente cadastrar apenas 10 convidados e exibi-los na mesma ordem na qual foram cadastrados.

Desenvolvendo seu programa, foi criado o seguinte programa em Java:

Principal.java

```
import javax.swing.JOptionPane;

public class Principal {

    public static void main(String[] args) {
        String convidado1 = "";
        String convidado2 = "";
        String convidado3 = "";
        String convidado4 = "";
        String convidado5 = "";
        String convidado6 = "";
        String convidado7 = "";
        String convidado8 = "";
        String convidado9 = "";
        String convidado10 = "";
        String listafinal = "";
        convidado1 = JOptionPane.showInputDialog("Digite o nome do 1º Convidado");
        convidado2 = JOptionPane.showInputDialog("Digite o nome do 2º Convidado");
        convidado3 = JOptionPane.showInputDialog("Digite o nome do 3º Convidado");
        convidado4 = JOptionPane.showInputDialog("Digite o nome do 4º Convidado");
        convidado5 = JOptionPane.showInputDialog("Digite o nome do 5º Convidado");
        convidado6 = JOptionPane.showInputDialog("Digite o nome do 6º Convidado");
        convidado7 = JOptionPane.showInputDialog("Digite o nome do 7º Convidado");
        convidado8 = JOptionPane.showInputDialog("Digite o nome do 8º Convidado");
        convidado9 = JOptionPane.showInputDialog("Digite o nome do 9º Convidado");
        convidado10 = JOptionPane.showInputDialog("Digite o nome do 10º Convidado");
        listafinal = "Os convidados da festa são: "+ convidado1 + ", "+ convidado2 + ", "+ convidado3 + ", "
                + convidado4 + ", "+ convidado5 + ", "+ convidado6 + ", "+ convidado7 + ", "+ convidado8 + ", "
                + convidado9 + ", "+ convidado10;
        JOptionPane.showMessageDialog(null,listafinal);
    }

}
```

Apesar do programa funcionar sem problemas, a codificação ficou muito repetitiva, não acha?

Isto que o programa está fazendo apenas uma função básica, sem muitos incrementos.
Com vetores, podemos reorganizar este código para utilizar menos da metade dos comandos, e com uma vantagem: Não é necessário acrescentar nenhuma linha de código extra, seja necessário cadastrar 10, 100 ou até 1000 convidados.

Como trabalhar com um Vetor

Em primeiro lugar para trabalhar com um vetor, devemos declara-lo, assim como fazemos com uma variável, a diferença é que na declaração devemos especificar a quantidade de “casas” que o vetor terá.
Cada casa do vetor, para o computador, é um espaço onde poderá ser escrito um valor. A primeira casa do vetor, sempre será a casa de número zero devido a numeração binária utilizada em qualquer meio computacional, onde o primeiro número é zero.

Vale lembrar que assim como uma variável, você nunca deve declarar um vetor sem necessidade. Um problema comum para quem está iniciando os estudos com vetores é a declaração de mais casas no vetor do que realmente necessita, ou o estouro do

vetor, que é o fato de tentar alocar mais valores no vetor do que ele suporta. Tome cuidado!

Declarando um vetor

Para declarar um vetor é muito simples, basta seguir o seu padrão conforme a tabela abaixo:

Pseudocódigo	Java
nomevariavel **como conjunto**[inicio..fim] **de** tipo	tipo[] nomevariavel = **new** tipo[10];

Repare que tanto o Pseudocódigo, quanto o Fluxograma utilizam a mesma forma para declarar um Vetor. Isto acontece porque ambas as notações utilizam a mesma linguagem como base. O Português Estruturado.

Praticando a Lógica com Vetores.

Dando continuidade ao nosso Vetor, vamos verificar como ficará o problema da Jennifer utilizando Vetores?

As únicas diferenças entre uma variável comum e um vetor é que ela sempre deverá ser lida acompanhada da "casa" em que o dado deverá ser lido/gravado, além disto, em Java, devemos nos atentar que sempre a primeira casa do vetor é nomeada como a casa Número 0.

Na prática, se declaramos um vetor de 10 casas em Java, ele alocará na memória as casas numeradas de 0 a 9.

```
System.out.println("O convidado é " + convidados[4]);
```

	Convidados[10]
convidados[0]	Zenaide
convidados[1]	Julio
convidados[2]	Luci
convidados[3]	Melvin
convidados[4]	Marta
convidados[5]	Blaino
convidados[6]	Hilda
convidados[7]	Kirla
convidados[8]	Lourdes
convidados[9]	Clovis

→ O convidado é Marta

VOCÊ NO COMANDO

Seguindo o vetor de convidados da imagem, como deve ser feito para exibir que a Kirla foi convidada? E para exibir todos os convidados? Você tem alguma ideia? Reflita sobre isto antes de dar continuidade a leitura.

Seguindo as particularidades do algoritmo e da Linguagem de Programação Java, Vamos desenvolver a aplicação para o problema da professora Jennifer utilizando Vetor:

Pseudocódigo

```
Programa convidaEvento
Declare
    convidados como conjunto[1..10] de caractere
    contador como inteiro
    listafinal como caractere
Inicio
    listafinal <- “Os convidados são: “
    Para contador <- 1 até 10 faça
        Escreva(“Digite o convidado número ”, contador,
“: “)
        Leia(convidados[contador])
     Fim-Para
     Para contador <- 1 até 10 faça
         listafinal <- listafinal +  convidados[contador]+ “ “
     Fim-Para
     Escreva(listafinal)
Fim
```

Em Java:

PincipalVetor.java

```
1  import javax.swing.JOptionPane;
2
3  public class PincipalVetor {
4
5      public static void main(String[] args) {
6          String[] convidados = new String[10];
7          int contador = 0;
8          //Definir a frase que antecede o nome dos convidados
9          String listafinal = "Os convidados da festa são: ";
10         //Laço de repetição para ler cada um dos convidados.
11         for (contador =0;contador<10;contador ++) {
12             convidados[contador]=JOptionPane.showInputDialog("Digite o " + (contador+1) + " convidado");
13             //Como o vetor inicia-se com zero, utilizamos contador+1 para identificar ao usuário qual
14             //é o número do convidado no qual ele está digitando corretamente.
15         }
16         //laço de repetição para alocar em uma variável o nome de cada um dos candidatos
17         for(contador = 0; contador < 10; contador ++) {
18             //Alocação dos dados na variável, adicionando novos valores a cada passagem do laço.
19             listafinal = listafinal + convidados[contador] + " ";
20         }
21         //exibição dos dados armazenados em listafinal.
22         JOptionPane.showMessageDialog(null, listafinal);
23
24
25     }
26
27 }
```

Você percebeu o quanto o código fonte diminuiu de tamanho após a aplicação do vetor? Um vetor pode ser um pouquinho mais trabalhoso para ser aplicado no início, mas ele consegue fazer o trabalho de muitas variáveis utilizando apenas uma! Este é um ótimo aliado para todo o bom programador.

1. Observe o exemplo de declaração de vetor a seguir e em seguida declare o vetor para as situações solicitadas:

a) Conforme a lista de convidados de Jennifer, vamos declarar o vetor correspondente?

Pseudocódigo	Java
convidados como conjunto[1..10] de caractere	String[] convidados = new String[];

b) Declarar um vetor para possibilitar o cálculo de uma média aritmética simples com base em 15 notas.

Pseudocódigo	Java

c) Declarar um vetor de 15 números inteiros para ordena-los em ordem crescente.

Pseudocódigo	Java

d) Declarar um vetor com 4 senhas diferentes para ser lidas e comparadas posteriormente.

Pseudocódigo	Java

2. Resolva as seguintes situações problemas com auxílio de Vetores:

a) Luana necessita de um programa para o cálculo das médias dos alunos de sua escola de música. Para cada módulo do curso, o aluno efetua 5 atividades. Caso a média destas atividades seja maior que 5, o aluno poderá passar para o próximo módulo. Desenvolva um programa em Java utilizando Vetores que calcule a média para os alunos de Luana corretamente.

b) Desenvolva um programa que leia 20 valores inteiros. Após a leitura, o programa deverá exibir os números armazenados no vetor, porém, os números que estiverem em casas pares do vetor, o valor deverá ser multiplicado por 2.

c) Marcos, Fibula e Jurema, alunos do primeiro módulo Curso Técnico em Informática, decidiram criar um novo jogo para a VI Feira de Informática da Escola onde estudam, afim de demonstrar o conhecimento adquirido e também entreter os visitantes.
Neste jogo, serão cadastradas previamente em um vetor (dentro do próprio código fonte), 5 palavras secretas, que terão algo em comum entre elas. Durante a execução do programa, será exibido para o usuário, o que as palavras tem em comum e o usuário digitará uma palavra. Se a palavra estiver no vetor, será exibida uma mensagem para o usuário, avisando-o que ele acertou o palpite e parabenizando-o. Caso contrário será exibida uma mensagem de erro.
Desafio: Para que o jogo fique mais empolgante, o usuário poderá ter até 3 chances de acertar uma das palavras.

d) Desenvolva um programa que leia 10 números do tipo inteiro, porém exiba apenas os números pares que foram cadastrados.

e) Desenvolva um programa que leia 5 valores de um vetor e os exiba na ordem inversa da qual foram cadastrados.

Respostas

1.

b)

Pseudocódigo	Java
media como conjunto [1..10] de real	double[] media = new Double[10];

c)

Pseudocódigo	Java
numero como conjunto [1..15] de inteiro	int[] media = new int[15];

d)

Pseudocódigo	Java
convidados como conjunto[1..4] de caractere	String[] convidados = new String[4];

2.

a) **Pseudocódigo**

```
declare
  nota como conjunto[1..5] de real
  soma como real
  media como real
  contador como inteiro
inicio
   para contador de 1 ate 5 passo 1 faca
     leia(nota[contador])
   fim-para
   para contador de 1 ate 5 passo 1 faca
     soma <- soma + nota[contador]
   fim-para
   media <- soma/5
   se media >= 5 entao
     escreva("O aluno tirou "+ media+ ", portanto está APROVADO")
   senao
     escreva("O aluno tirou "+ media+ ", portanto está REPROVADO")
   fim-se
fim
```

No Java:

Principal.java

```java
import javax.swing.JOptionPane;

public class Principal {
    public static void main(String args[]) {
        double nota[] = new double[5];
        double soma = 0;
        double media;
        int contador;
        //laço para ler as notas do aluno
        for (contador = 0; contador < 5; contador ++) {
            nota[contador] = Double.parseDouble(JOptionPane.showInputDialog("Digite a " + (contador+1) +" nota do aluno."));
        }
        //laço para somar as notas dos alunos
        for(contador = 0; contador <5; contador ++) {
            soma = soma + nota[contador];
        }
        media = soma / 5;
        if (media >= 5 ) {
            JOptionPane.showMessageDialog(null, "O aluno tirou " + media + ", portando está APROVADO");
        }
        else{
            JOptionPane.showMessageDialog(null, "O aluno tirou " + media + ", portando está REPROVADO");
        }

    }
}
```

b) **Pseudocódigo**

```
declare
  valor como conjunto[1..20] de inteiro
  contador como inteiro
  mult como inteiro
  numeros como caractere
inicio
   para contador de 1 ate 20 passo 1 faca
     leia(valor[contador])
   fim-para
   numeros <- "Os números digitados foram: "
   para contador de 1 ate 20 passo 1 faca
     se valor[contador] mod 2 = 0 entao
        mult <- valor[contador] * 2
        numeros <- numeros + mult + " "
     senao
         numeros <- numeros + valor[contador]
     fim-se
   fim-para
        escreva(numeros)
fim
```

No Java:

Principal.java

```java
import javax.swing.JOptionPane;

public class Principal {
    public static void main(String args[]) {
        int valor[] = new int[20];
        int contador;
        String numeros;
        int mult;

        for (contador = 0; contador < 20; contador ++) {
            valor[contador]= Integer.parseInt(JOptionPane.showInputDialog("Digite o "+ (contador +1) +" valor"));
        }
        numeros = "Os números digitados foram: ";
        for (contador = 0; contador < 20; contador ++) {
            if(valor[contador] % 2 == 0) {
                mult = valor[contador] * 2;
                numeros = numeros + String.valueOf(mult) + " ";
            }
            else {
                numeros = numeros + String.valueOf(valor[contador]) + " ";
            }
        }
        JOptionPane.showMessageDialog(null, numeros);
    }
}
```

c) **Pseudocódigo**

```
declare
  resposta como conjunto[1..5] de caractere
  palavra como caractere
  acerto como caractere
  contador como inteiro
inicio
    resposta[1] = "Futebol"
    resposta[2] = "Natação"
    resposta[3] = "Ciclismo"
    resposta[4] = "Arremesso de peso"
    resposta[5] = "Remo"
    escreval("São esportes Olímpicos")
    escreva("A palavra é: ")
    leia(palavra)
    acerto = "nao"
    para contador de 1 ate 5 passo 1 faca
       se resposta[contador] = palavra entao
         escreva("Parabéns, você acertou!!!")
         acerto = "sim"
       fim-se
    fim-para
        se acerto = "nao" entao
          escreva("Poxa! Você errou!")
        fim-se
fim
```

No Java:

ProjetoPalavraSecreta.java

```java
import javax.swing.JOptionPane;

public class ProjetoPalavraSecreta {
    public static void main(String args[]) {
           String resposta[] = new String[5];
           String palavra;
           String acerto;
           int contador;
              resposta[0] = "futebol";
              resposta[1] = "natação";
              resposta[2] = "ciclismo";
              resposta[3] = "arremesso de peso";
              resposta[4] = "remo";
              JOptionPane.showMessageDialog(null, "São esportes Olímpicos");
              palavra = JOptionPane.showInputDialog("A palavra é: ");
              palavra = palavra.toLowerCase(); //linha para padronizar a palavra em caixa baixa (letras minúsculas).
              acerto = "nao"; // a variável acerto faz o controle para que a mensagem de erro não apareça em caso de acerto.
              for (contador = 0; contador < 5; contador ++) {
                  if(resposta[contador].equals(palavra)) {
                      JOptionPane.showMessageDialog(null, "Parabéns, você acertou!!!");
                      acerto = "sim";
                  }
              }
              if(acerto.equals("nao")) {
                  JOptionPane.showMessageDialog(null, "Poxa! Você errou!");
              }

    }
}
```

Pseudocódigo:

```
declare
  resposta como conjunto[1..5] de caractere
  palavra como caractere
  acerto como caractere
  contador como inteiro
  tentativa como inteiro
inicio
    resposta[1] = "Futebol"
    resposta[2] = "Natação"
    resposta[3] = "Ciclismo"
    resposta[4] = "Arremesso de peso"
    resposta[5] = "Remo"
    escreval("São esportes Olímpicos")
    acerto = "nao"
para tentativa de 1 ate 3 passo 1 faca
    escreva("A palavra é: ")
    leia(palavra)
    para contador de 1 ate 5 passo 1 faca
        se resposta[contador] = palavra entao
          escreva("Parabéns, você acertou!!!")
          acerto = "sim"
            tentativa <-3
        fim-se
    fim-para
         se acerto = "nao" entao
           escreva("Poxa! Você errou!")
         fim-se
fim-para
se acerto = "nao" entao
   escreva("Poxa! Você errou!")
fim-se

fim
```

No Java:

*ProjetoPalavraSecreta.java

```java
import javax.swing.JOptionPane;
public class ProjetoPalavraSecreta {
    public static void main(String args[]) {
            String resposta[] = new String[5];
            String palavra;
            String acerto;
            int tentativa;
            int contador;
               resposta[0] = "futebol";
               resposta[1] = "natação";
               resposta[2] = "ciclismo";
               resposta[3] = "arremesso de peso";
               resposta[4] = "remo";
               JOptionPane.showMessageDialog(null, "São esportes Olímpicos");
               acerto = "nao"; // a variável acerto faz o controle para que a mensagem de erro não apareça em caso de acerto.
            for(tentativa = 0; tentativa < 3; tentativa ++) {//obrigatório para as 3 tentativas
                palavra = JOptionPane.showInputDialog("A palavra é: ");
                palavra = palavra.toLowerCase(); //linha para padronizar a palavra em caixa baixa (letras minúsculas).
               for (contador = 0; contador < 5; contador ++) {
                   if(resposta[contador].equals(palavra)) {
                       JOptionPane.showMessageDialog(null, "Parabéns, você acertou!!!");
                       acerto = "sim";
                       tentativa = 3; //caso você iguale a um valor maior do que o previsto na estrutura, ela será finalizada.
                   }
               }
               if(acerto.equals("nao")) {
                   JOptionPane.showMessageDialog(null, "Poxa! Você errou!");
               }
            }
            if(acerto.equals("nao")) {
                JOptionPane.showMessageDialog(null, "Você errou as 3 tentativas!");
            }
    }
}
```

d) **Pseudocódigo**

```
declare
  valor como conjunto[1..10] de inteiro
  contador como inteiro
  numeros como caractere
inicio
   para contador de 1 ate 10 passo 1 faca
     leia(valor[contador])
   fim-para
   numeros <- "Os números pares digitados foram: "
   para contador de 1 ate 10 passo 1 faca
     se valor[contador] mod 2 = 0 entao
        numeros <- numeros + valor[contador] + " "
     fim-se
   fim-para
        escreva(numeros)
fim
```

No Java:

*Principal.java

```
1  import javax.swing.JOptionPane;
2
3      public class Principal {
4      public static void main(String args[]) {
5          int valor[] = new int[10];
6          int contador;
7          String numeros;
8          for (contador = 0; contador < 10; contador ++) {
9              valor[contador]= Integer.parseInt(JOptionPane.showInputDialog("Digite o "+ (contador +1) +" valor"));
10         }
11         numeros = "Os números pares digitados foram: ";
12         for (contador = 0; contador < 10; contador ++) {
13             if(valor[contador] % 2 == 0) {
14                 numeros = numeros + String.valueOf(valor[contador]) + " ";
15             }
16
17         }
18         JOptionPane.showMessageDialog(null, numeros);
19     }
20 }
```

Outra solução para o mesmo exercício:

Pseudocódigo

```
declare
  numeros como conjunto[1..5] de inteiro
  contador como inteiro
  resultado como caractere
inicio
    para contador de 1 ate 5 passo 1 faca
        escreva ("Digite o " + contador + " numero")
        leia(numeros[contador])
    fim-para
    resultado<- "Os números em ordem inversa são: "
    para contador de 5 ate 1  passo -1 faca
       resultado <- resultado + numeros[contador]
    fim-para
    escreva(resultado)
fim
```

No Java:

Principal.java

```
1  import javax.swing.JOptionPane;
2
3  public class Principal {
4      public static void main(String[] args) {
5          int[] numeros = new int[5];
6          int contador;
7          String resultado;
8          resultado = "Os números em ordem inversa da digitada são: ";
9          for (contador =0;contador<5;contador ++) {
10             numeros[contador]=Integer.parseInt(JOptionPane.showInputDialog("Digite o " + (contador+1) + " número"));
11
12         }
13
14         for(contador = 4; contador >= 0; contador --) {
15
16             resultado = resultado + String.valueOf(numeros[contador]) + " ";
17         }
18         JOptionPane.showMessageDialog(null, resultado);
19
20
21     }
22
23 }
```

Como você já sabe, cada programa, cada código fonte gerado, sempre há mais de uma forma de se estruturar o programa para receber os mesmos resultados ao final. Ao se Tratar de Vetores, não é diferente. Nesta aula vimos a sua forma básica, abordando apenas uma dimensão. Dependendo da circunstância na qual se encontra o problema, podemos utilizar também as Matrizes Bidimensionais e Heterogêneas para resolver nossos problemas computacionais, basta saber utiliza-los adequadamente.

Estes exercícios devem ser entregues de forma on-line como atividades da agenda.

Exercício 1

Imagem 25

A empresa LotoCom, uma empresa especializada em sorteios para promoções, decidiu iniciar a programação de um novo software para trabalhar com uma nova forma de testar a sorte dos participantes de seus sorteios. Neste programa, os participantes deverão inserir 3 números da sorte, de 0 a 38, em um vetor de três posições.O sistema deverá somar todos os números do vetor que forem divisíveis por 4. Após a soma, deverá dividir o número encontrado por 3. Se o resultado da conta do vetor for

acima de 25, o sistema mostrará que o usuário terá prêmio máximo, se estiver entre 20 e 25, prêmio comum e abaixo disso avisará que ele não receberá prêmio nenhum.
Como deve ficar o programa codificado em Java?
Fonte da imagem: https://mx.depositphotos.com/88907452/stock-illustration-abstract-dark-grey-random-numbers.html. Acessado em 21/12/2017.

Não esqueça que para saber se um número é divisível por outro, deve-se utilizar o operador responsável por calcular o resto da divisão.

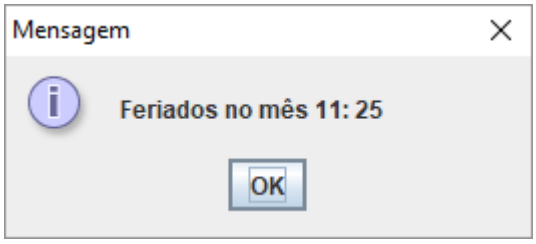
Exercício 2

Monito, um funcionário aplicado da empresa Vetor Participações, está desenvolvendo um módulo em Java para o software principal de sua empresa, onde será exibido os dias dos feriados nacionais, sempre que solicitado. Como resultado, Monito produziu o seguinte programa em Java:

Feriado.java

```
1  import javax.swing.JOptionPane;
2
3  public class Feriado {
4
5      public static void main(String[] args) {
6          int contador = 0;
7          String mensagem = "";
8          String[] feriados = new String[12];
9          feriados[0]="01";
10         feriados[1]="28";
11         feriados[2]="01";
12         feriados[3]="14, 21";
13         feriados[4]="01";
14         feriados[5]="15";
15         feriados[6]="Nenhum";
16         feriados[7]="Nenhum";
17         feriados[8]="07";
18         feriados[9]="12";
19         feriados[10]="02, 15";
20         feriados[11]="25";
21         for(contador = 0; contador <=11; contador ++) {
22             mensagem =  "Feriados no mês "+ contador + ": " + feriados[contador] + "\n" ;
23         }
24         JOptionPane.showMessageDialog(null, mensagem);
25     }
26 }
```

Como Monito não tem um conhecimento tão vasto em Java, ao executar seu programa o sistema retornou:



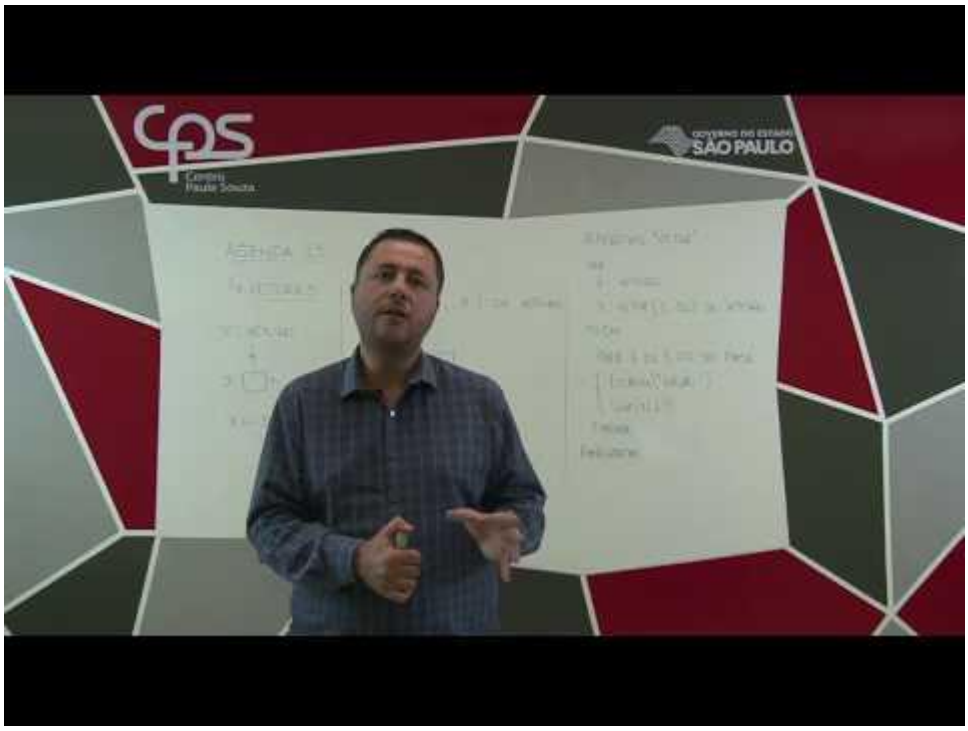
Como foi observado, não deu certo o programa construído por ele. Para resolver este problema, como Monito deverá alterar seu código para que o número do mês apareça corretamente? Replique o programa criado por Monito no seu computador, e corrija suas falhas.

AMPLIANDO HORIZONTES

Para aprofundamento dos temas discutidos nesta aula, seguem abaixo algumas dicas de filmes, livros e artigos que se relacionam com o conteúdo estudado. Estas dicas são muito importantes para você!

Não deixe também de assistir ao vídeo:

Informática - Módulo I - Agenda 15 - Vetores



Link: https://www.youtube.com/watch?v=C7HgTq2bE-E. Acessado em 21/12/2017.

Apostila:

Apostila de Algoritmo Estruturado de Salete Buffoni – Capítulo 5. Disponível em: http://www.dainf.ct.utfpr.edu.br/~pbueno/Arquivos/Algoritmos.pdf. Acessado em 18/10/2017.

Livro:

Lógica de Programação e Estruturas de Dados com Aplicações em Java de Sandra Puga e Gerson Risseti. Editora Pearson, 2009.

AGENDA 16
MATRIZES

MOMENTO DE REFLEXÃO

Imagem 26

Ninguém entende as mulheres! Sapatos e bijuterias, um de cada cor, são algumas das prioridades na vida delas. Só quem é mulher pode entender muito bem isso!

Vive comprando e, quando procura e não acha, compra mais ainda porque pensa que não tem. Isso só acontece porque na maioria das vezes as coisas não estão muito bem organizadas. E quando encontra o brinco azul, mas só acha o anel vermelho... Nada combina!

Amanda resolveu colocar um fim na bagunça das suas bijuterias e comprou uma caixa com 3 fileiras, cada uma com 3 repartições e assim conseguiu organizar certinho cada conjunto de cores, um em cada fileira, colocando os brincos, anéis e colares, cada um numa repartição.

Aí, pronto! Ficou tudo guardado em ordem, sem perder nenhum conjunto de cores!

Em cada fileira tem uma repartição correspondente à cor da bijuteria.

Assim acontece com os dados no mundo da programação, quando precisam ser armazenados ou lidos de acordo com a relação entre eixos, representados por mais de um valor dentro de uma variável. É o que chamamos de matrizes que são variáveis que armazenam múltiplos valores com mais de uma dimensão. Assim como as gavetas com as fileiras que contém mais de uma repartição!

Até agora você aprendeu a trabalhar com o uso de uma única variável indexada com apenas uma dimensão (uma coluna com várias linhas), que são os vetores que acabamos de estudar. No entanto, existem situações onde precisamos trabalhar com mais de um tipo de informação sendo necessário utilizar mais de uma coluna, onde devemos ter variáveis no sentido horizontal e vertical. Por este motivo é muito importante você estudar matrizes de mais de uma dimensão!

Nas agendas anteriores foram explicados o conceito e a prática de vetores, que nada mais é do que uma matriz com uma dimensão – linha ou coluna - e posteriormente o conceito de uma matriz. Vale ressaltar que uma matriz também pode ser chamada de Array, termo originário do inglês e também muito utilizado.
Um modo fácil de entender o conceito de como os elementos de uma matriz são ordenados é realizar uma comparação com o jogo de batalha naval, onde as linhas são representadas por números e as colunas por letras. Para identificarmos um elemento no “campo de batalha” do jogo utilizamos uma combinação de letra e número, por exemplo A10 ou D5. Nas matrizes a identificação é exatamente idêntica, porém utilizamos somente números para a representação e para não realizamos confusão usamos duas variáveis distintas, uma para a representação das linhas e outra das colunas, como foi visto na agenda anterior.

	1	**2**	**...**	**100**
1	10	7	...	
2	34	78	...	9
...	...	...	...	18
100	6	3	...	54

Pense na matriz representada acima: na primeira linha e na primeira coluna (em negrito) estão a representação dos índices para a identificação de cada elemento da matriz. Por exemplo, para endereçarmos o elemento de número 9 teremos: linha[2], coluna[100] = 9.

Matrizes são utilizadas constantemente em nosso cotidiano sem nos darmos conta. Vamos a um exemplo: um professor para realizar o fechamento das notas bimestrais de uma de suas turmas possui notas de três atividades de cada um de seus alunos e resolve colocar tudo em uma tabela. Considerando que todas as notas podem ser números reais teremos tipos de dados idênticos para todos os campos, inclusive para a média das notas. Assim cada uma das notas individuais de seus alunos da tabela representa um elemento da matriz. E esses elementos podem ser facilmente referenciados para a realização de cálculos ou outra operação necessária.

$$\begin{bmatrix} 5 & 7 & 6 & 6 \\ 8 & 8 & 8 & 8 \\ ... & ... & ... & ... \\ 10 & 8 & 9 & 9 \end{bmatrix}$$

Pense na matriz representada acima: nas três primeiras colunas temos as notas das 3 atividades e na última coluna a média bimestral. É um modo muito simples e eficaz de representamos os dados em programação.

O conceito de como uma matriz é declarada e utilizada já foi explicado anteriormente. Neste capítulo focaremos na parte prática da utilização delas em pseudocódigo e em Java.
A sintaxe da declaração de uma matriz bidimensional é ilustrada na tabela a seguir:

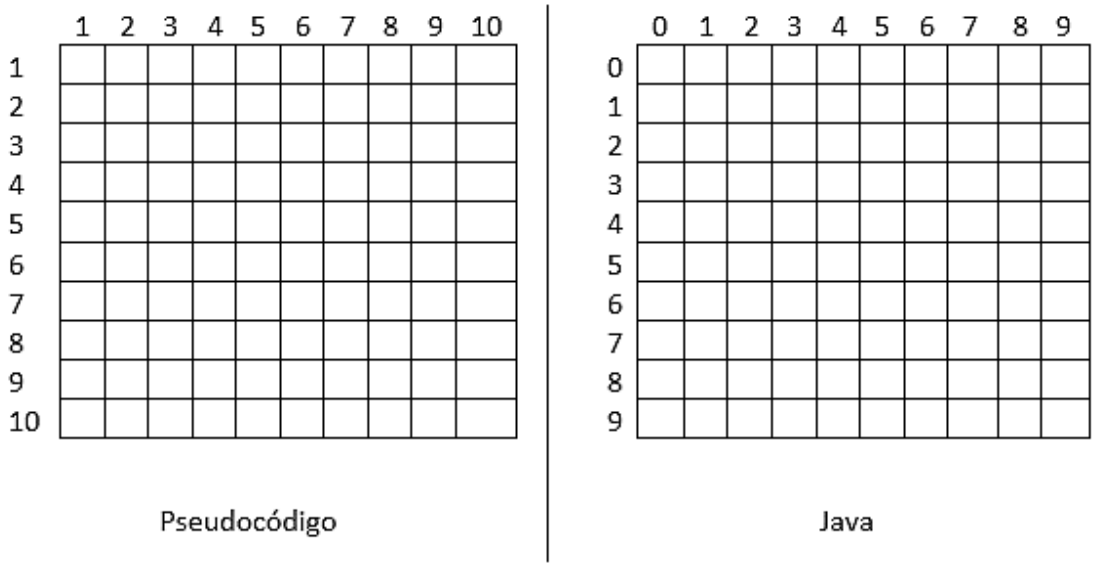
Pseudocódigo	Java
Declare <nome> como **conjunto** [1..n][1..m] de <tipo>	<tipo> <nome>[ ][ ] = new <tipo>[n][m];

Onde temos que **<nome>** é o nome da matriz, **<tipo>** é o tipo de variável a ser armazenada na matriz e **n** o número de linhas e **m** o número de colunas.

Exemplos de declaração de matrizes bidimensionais:

Pseudocódigo	Java
Declare mat como **conjunto** [1..10][1..10] de inteiro	int mat[ ][ ] = new int [10][10];

Em ambos os casos temos a declaração de uma matriz **mat** com duas dimensões (linha e coluna) com 100 posições no total (10 linhas x 10 colunas).



Aqui cabe uma observação muito importante: note que foram ilustradas duas matrizes acima. Na matriz de **pseudocódigo a contagem inicia-se pelo número 1** (um). Já na matriz do **Java a contagem inicia-se pelo número 0** (zero). **Lembre-se disso**!

Não se esqueça quando utilizamos Arrays em Java, sejam eles unidimensionais – vetores – ou bidimensionais – matrizes – a contagem sempre se inicia do zero. Isso mesmo! Vocês podem utilizar estes termos: Arrays Unidimensionais para informar o uso de vetores ou Arrays bidimensionais para utilizar matrizes. Esta é a diferença!

Outros exemplos:

Pseudocódigo	Java

Declare
nomes como **conjunto** [1..10][1..5] de caractere
num como **conjunto** [1..4][1..4] de real
	String nomes[ ][ ] = new String [10][5];
	double num[ ] [ ] = new double [4][4];
	float num1[ ] [ ] = new float [4][4];

Observe que o número de linhas e o número de colunas representados pelos [ ] (colchetes) não precisam ser iguais.

VOCÊ NO COMANDO

Até agora foram declaradas matrizes sem valores inicializados. Pesquise e Pense em como poderíamos declarar uma matriz 2x2 (duas linhas por duas colunas) com os seus valores inteiros já inicializados em Java.

MatrizIncializada.java

```
1  import javax.swing.JOptionPane;
2
3  public class MatrizIncializada {
4
5      public static void main(String[] args) {
6          // Matriz 2x2 inicializada
7
8          //declaração da matriz e inicialização
9          int mat[][] = { {1,2}, {3,4} };
10
11         //saída de valores da matriz
12         for (int linha= 0; linha<2;linha++) {
13             for(int coluna = 0; coluna<2; coluna++) {
14                 JOptionPane.showMessageDialog(null, "Matriz[" + linha + "] coluna["
15                         + coluna + "] = " + mat[linha][coluna]);
16             }//fim do segundo for
17         }//fim do primeiro for
18     }//fim do main
19 }//fim da classe
20
```

Na agenda passada, você estudou Vetores. A mesma situação que você viu agora com Matriz, também, pode acontecer com os Vetores.

O acesso aos elementos de uma matriz é feito utilizando-se comandos de repetição. Bem, foram vistos durante o curso, três tipos de comandos de repetição. O comando de repetição “Para...Fim-

Para", FOR no Java é o que mais se adequa as necessidades pois executa uma repetição por um número fixo de vezes. Lembrando que uma matriz possui um número fixo de linhas e colunas.

Os comandos "Para...Fim-Para" e FOR ficariam com a seguinte sintaxe para permitir o acesso a um elemento da matriz, considerando uma matriz 4x4 de números inteiros, com as variáveis linha e coluna controlando o acesso a linha e a coluna das matrizes:

Pseudocódigo	Java

Programa MatrizExemplo
Declare
num como **conjunto** [1..4][1..4] de real
linha, coluna como inteiro
Início
Escreva("inserindo os dados na Matriz")
Para linha = 1 **Até** 4 **Faça**
Para coluna = 1 **Até** 4 **Faça**
Escreva ("Entre com um número")
Escreva ("linha", linha)
Escreva ("coluna, coluna)
Leia num[linha, coluna]
Fim-Para
Fim-Para
Escreva("Mostrando os dados na Matriz")
Para linha = 1 **Até** 4 **Faça**
Para coluna = 1 **Até** 4 **Faça**
Escreva ("linha", linha)
Escreva ("coluna, coluna)
Escreva ("valor lido", num[linha, coluna])
Fim-Para
Fim-Para
Fim
	double num [] [] = new double [4][4];
	int linha, coluna;
	JOptionPane.showMessageDialog (null, "Inserindo os dados na Matriz");
	for (linha = 0; linha < 4; linha++) {
	for (coluna = 0; coluna < 4; coluna++) {
	num[linha][coluna] = Double.parseDouble(
	JOptionPane.showInputDialog("Entre com o número" + "\nlinha " + "" + linha + "\ncoluna " + coluna));
	}
	}
	JOptionPane.showMessageDialog (null, "Mostrando
	os dados na Matriz");
	for (linha = 0; linha < 4; linha++) {
	for (coluna = 0; coluna < 4; coluna++) {
	JOptionPane.showMessageDialog(null,"linha " + linha + "\ncoluna " + coluna + "\nNúmero " + num[linha][coluna])
	}
	}

O pseudocódigo inicia-se com a declaração da matriz **num 4x4** e das variáveis linha e coluna que servirão para acessarmos os elementos da matriz. Logo depois temos dois comandos de repetição encadeados, um para controlarmos a linha e outro para a coluna. Dessa forma os elementos da matriz serão inseridos na primeira linha uma coluna por vez e ao chegar ao final (coluna 4) passa-se para a linha seguinte.

Note que no interior desses dois comandos de repetição temos uma entrada de dados para inserir o valor do elemento **num[linha][coluna]** da matriz. Exemplo quando a linha = 1 e a coluna = 2, teremos acesso ao elemento num[1][2].

O mesmo ocorre na etapa "Mostrando dos dados da matriz", porém ao invés de realizarmos a leitura do valor **num[linha][coluna]** realizamos a escrita deste.

No Java ocorre algo semelhante. Primeiro realizamos a declaração da matriz e das variáveis de controle, para na primeira dupla do comando FOR realizarmos a entrada dos dados e na segunda dupla realizarmos a exibição dos dados inseridos.

VOCÊ NO COMANDO

Quando executamos o exemplo acima em Java, várias janelas com a exibição dos dados são geradas. São pelo menos 16 janelas para a entrada e mais 16 para a saída de dados. Desafio: Tente modificar o programa de modo que ele fique com as 16 janelas de entrada e somente 1 janela de saída. Obs.: a resposta estará na resolução dos exercícios deste capítulo na seção Exercitando e Aprimorando.

A instrução "linha++" dentro do comando for é igual a conta: linha = linha +1

Vamos a mais um exemplo: vamos retomar o exemplo do início do capítulo onde o professor desejava fazer uma matriz em um programa para calcular as notas bimestrais dos seus estimados alunos. Considerando que ele tem dez alunos nessa classe e três notas de atividades mais a média, teremos uma matriz com 10 linhas e 4 colunas. Uma linha para cada aluno, 3 colunas para as notas dos alunos e uma coluna para o cálculo da média. Como codificaríamos esse programa?

Pseudocódigo

```
Programa MatrizExemplo
Declare
   notas como conjunto [1..10][1..4] de real
   linha, coluna como inteiro
   media como real
   Início
   Escreva("inserindo os dados na Matriz")
   Para linha = 1 Até 10 Faça
      Para coluna = 1 Até 3 Faça
         Escreva ("Entre com um número")
         Escreva ("linha", linha)
         Escreva ("coluna, coluna)
         Leia notas[linha, coluna]
      Fim-Para
   Fim-Para

   Escreva("calculando...")
   Media = 0
   Para linha = 1 Até 10 Faça
      Para coluna = 1 Até 3 Faça
```

```
        media = media + notas[linha, coluna]
      Fim-Para
      notas[linha,4] = media/3
      media = 0
    Fim-Para

    Escreva("Mostrando os dados na Matriz")
    Para linha = 1 Até 10 Faça
      Para coluna = 1 Até 3 Faça
        Escreva ("linha", linha)
        Escreva ("coluna, coluna)
        Escreva ("valor lido", notas[linha, coluna])
      Fim-Para
        Escreva ("Média" ,notas[linha,4])
    Fim-Para
Fim
```

No Java:

Matriz.java | *MatrizExemploMedia.java

```
1  import javax.swing.JOptionPane;
2
3  public class MatrizExemploMedia {
4
5      public static void main(String[] args) {
6          //declaração de variáveis
7          double notas [] [] = new double [10][4];
8          int linha, coluna;
9          double media = 0;
10
11         //entrada de dados
12         JOptionPane.showMessageDialog (null, "Inserindo os dados na Matriz");
13         for (linha = 0; linha < 10; linha++) {
14             for (coluna = 0; coluna < 3; coluna++) {
15                 notas[linha][coluna] = Double.parseDouble(JOptionPane.showInputDialog("Entre com o número" +
16                  "\nlinha " + linha +
17                  "\ncoluna " + coluna));
18             }//fim do for
19         }//fim do for
20
21         //cálculo da média
22         for (linha = 0; linha < 10; linha++) {
23             for (coluna = 0; coluna < 3; coluna++) {
24                 media = media + notas[linha][coluna];
25             }//fim do for
26             notas[linha][3] = media/3;
27         }//fim do for
28
29         //saída de dados
30         JOptionPane.showMessageDialog (null, "Mostrando os dados na Matriz");
31         for (linha = 0; linha < 10; linha++) {
32             for (coluna = 0; coluna < 3; coluna++) {
33                 JOptionPane.showMessageDialog(null,"linha " + linha +
34                  "\ncoluna " + coluna + "\nNúmero " + notas[linha][coluna]);
35             }//fim do for
36             //exibe a média de cada aluno
37             JOptionPane.showMessageDialog(null, "Média: " + notas[linha][3]);
38         }//fim do for
39     }//fim do método main
40 }
41
```

Geralmente utilizamos até matrizes bidimensionais, mas podemos utilizar Arrays tridimensionais também. Para isso teremos que acrescentar mais um for para o acesso dos elementos e a declaração desta com 2 x 2 x 2 elementos ficará assim em Java: double matriz [ ] [ ] [ ] = new double [2][2][2];

VOCÊ NO COMANDO

Pense em como poderíamos inserir o tratamento de erros Try-Catch no exemplo acima.

Exemplo com o Try-Catch em Java:

*MatrizExemploMedia.java

```
1  import javax.swing.JOptionPane;
2
3  public class MatrizExemploMedia {
4
5      public static void main(String[] args) {
6          double notas [] [] = new double [10][4];
7          int linha, coluna;
8          double media = 0;
9          try {
10             JOptionPane.showMessageDialog (null, "Inserindo os dados na Matriz");
11             for (linha = 0; linha < 10; linha++) {
12                 for (coluna = 0; coluna < 3; coluna++) {
13                     notas[linha][coluna] = Double.parseDouble(JOptionPane.showInputDialog(
14                         "Entre com o número" + "\nlinha " + linha + "\ncoluna " + coluna));
15                 }//fim do for
16             }//fim do for
17             for (linha = 0; linha < 10; linha++) {
18                 for (coluna = 0; coluna < 3; coluna++) {
19                     media = media + notas[linha][coluna];
20                 }//fim do for
21                 notas[linha][3] = media/3;
22             }//fim do for
23
24             JOptionPane.showMessageDialog (null, "Mostrando os dados na Matriz");
25             for (linha = 0; linha < 4; linha++) {
26                 for (coluna = 0; coluna < 4; coluna++) {
27                     JOptionPane.showMessageDialog(null,"linha " + linha +
28                             "\ncoluna " + coluna + "\nNúmero " + notas[linha][coluna]);
29                 }//fim do for
30                 JOptionPane.showMessageDialog(null, "Média: " + notas[linha][3]);
31             }//fim do for
32         }catch (NumberFormatException e) {
33             JOptionPane.showMessageDialog(null, "Entre somente com números\nEncerrando",
34                     "ERRO", JOptionPane.ERROR_MESSAGE);
35         } //fim do try-catch
36     }//fim do método main
37 }//fim da classe
```

Os exercícios devem ser desenvolvidos elaborando o pseudocódigo e linguagem Java:

1. Como estudante do ensino médio Paulo deseja fazer um software que realize a soma de duas matrizes 4x4. Ele sabe que para realizar a soma de duas matrizes, segundo a matemática, cada elemento da matriz A deve ser somado ao seu elemento correspondente da matriz B, gerando o resultado em uma terceira matriz C. Soma de matrizes: A[1,1] + B[1,1] = C[1,1]. Como Paulo resolverá essa questão?
2. Paulo agora quer um outro programa que realize a soma de todos os elementos de uma matriz 10x10 contendo números inteiros. Como Paulo deve elaborar o programa?
3. Altere o programa do exercício 2 para que este calcule também a média de todos os valores da matriz 10x10.
4. Roberta deseja fazer um programa que armazene somente três dados de cada um dos visitantes de seu restaurante: o nome completo, a cidade e o estado aonde residem. O programa deve ser capaz de armazenar 100.000 pessoas. Como Roberta resolveria esse problema?

Dica: Durante os testes faça com somente 10 entradas e depois altere para o que foi pedido.

Respostas:

1.

Pseudocódigo

```
Programa MatrizEx1
Declare
   a como conjunto [1..4][1..4] de inteiro
   b como conjunto [1..4][1..4] de inteiro
   c como conjunto [1..4][1..4] de inteiro
   linha, coluna como inteiro
   Início
   Escreva("inserindo os dados na Matriz A")
   Para linha = 1 Até 4 Faça
      Para coluna = 1 Até 4 Faça
         Escreva ("Entre com um número")
         Escreva ("linha", linha)
         Escreva ("coluna, coluna)
         Leia a[linha,coluna]
      Fim-Para
   Fim-Para

   Escreva("inserindo os dados na Matriz B")
   Para linha = 1 Até 4 Faça
      Para coluna = 1 Até 4 Faça
         Escreva ("Entre com um número")
         Escreva ("linha", linha)
         Escreva ("coluna, coluna)
         Leia b[linha,coluna]
      Fim-Para
   Fim-Para

   Escreva("calculando matriz c")
   Para linha = 1 Até 4 Faça
      Para coluna = 1 Até 4 Faça
         c[linha,coluna] = a[linha,coluna] + b[linha,coluna]
      Fim-Para
   Fim-Para

   Escreva("Mostrando os resultado da matriz C")
   Para linha = 1 Até 10 Faça
      Para coluna = 1 Até 3 Faça
         Escreva ("linha", linha)
         Escreva ("coluna, coluna)
         Escreva ("valor calculado", c[linha,coluna])
      Fim-Para
   Fim-Para
Fim
```

No Java:

```
1  import javax.swing.JOptionPane;
2
3  public class MatrizEx1 {
4
5      public static void main(String[] args) {
6          // exercício 1
7
8          //declaração de variáveis
9
10         int a[][] = new int [4][4];
11         int b[][] = new int [4][4];
12         int c[][] = new int [4][4];
13         int linha, coluna;
14
15         //entrada de dados
16         //matriz A
17         for (linha = 0; linha < 4; linha++) {
18             for (coluna = 0; coluna <4; coluna++) {
19                 a[linha][coluna]= Integer.parseInt(JOptionPane.showInputDialog
20                         ("Entre com o elemento [" + linha +"][" + coluna + "] da matriz A"));
21             }//fim do for
22         }//fim do for
23
24         //matriz B
25         for (linha = 0; linha < 4; linha++) {
26             for (coluna = 0; coluna <4; coluna++) {
27                 b[linha][coluna]= Integer.parseInt(JOptionPane.showInputDialog
28                         ("Entre com o elemento [" + linha +"][" + coluna + "] da matriz B"));
29             }//fim do for
30         }//fim do for
31
32         //cálculo
33         for (linha = 0; linha < 4; linha++) {
34             for (coluna = 0; coluna <4; coluna++) {
35                 c[linha][coluna]= a[linha][coluna] + b[linha][coluna];
36             }//fim do for
37         }//fim do for
38
39         //saída de dados
40         for (linha = 0; linha < 4; linha++) {
41             for (coluna = 0; coluna <4; coluna++) {
42                 JOptionPane.showMessageDialog(null, "C[" + linha + "][" + coluna +
43                         "] = " + c[linha][coluna]);
44             }//fim do for
45         }//fim do for
46     }
47 }
48
```

2.

Pseudocódigo

```
Programa MatrizEx2
Declare
   a como conjunto [1..10][1..10] de inteiro
   linha, coluna, soma como inteiro
   Início
   Escreva("inserindo os dados na Matriz")
   Para linha = 1 Até 10 Faça
      Para coluna = 1 Até 10 Faça
         Escreva ("Entre com um número")
         Escreva ("linha", linha)
         Escreva ("coluna, coluna)
         Leia a[linha, coluna]
      Fim-Para
   Fim-Para

   Escreva("calculando...")
   soma = 0
   Para linha = 1 Até 4 Faça
      Para coluna = 1 Até 4 Faça
         soma = soma + a[linha, coluna]
      Fim-Para
   Fim-Para

  Escreva ("soma", soma)
Fim
```

No Java:

```
1  import javax.swing.JOptionPane;
2
3  public class MatrizEx2 {
4
5      public static void main(String[] args) {
6          // exercício 2
7
8          //declaração de variáveis
9
10         int linha = 10, coluna = 10;
11         int a[][] = new int [linha][coluna];
12         int i, j, soma;
13
14         //entrada de dados
15         for (i = 0; i < linha; i++) {
16             for (j  = 0; j < coluna; j++) {
17                 a[i][j]= Integer.parseInt(JOptionPane.showInputDialog
18                         ("Entre com o elemento [" + i +"][" + j + "] da matriz"));
19             }//fim do for
20         }//fim do for
21
22         //cálculo
23         soma = 0;
24         for (i = 0; i < linha; i++) {
25             for (j  = 0; j < coluna; j++) {
26                 soma = soma + a[i][j];
27             }//fim do for
28         }//fim do for
29
30         //saída de dados
31         JOptionPane.showMessageDialog(null, "A soma de todos os elementos é " + soma);
32     }//fim do método main
33
34 }//fim da classe
35
```

3.

Pseudocódigo
Programa MatrizEx3
Declare
a como **conjunto** [1..10][1..10] de inteiro
linha, coluna, soma como inteiro
media como real
Início
Escreva(“inserindo os dados na Matriz”)
Para linha = 1 **Até** 10 **Faça**
Para coluna = 1 **Até** 10 **Faça**
Escreva (“Entre com um número”)
Escreva (“linha”, linha)
Escreva (“coluna, coluna)
Leia a[linha, coluna]
Fim-Para

```
   Fim-Para

   Escreva("calculando...")
   soma = 0
   Para linha = 1 Até 4 Faça
      Para coluna = 1 Até 4 Faça
         soma = soma + a[linha, coluna]
      Fim-Para
   Fim-Para
   media = soma / (10 *10)

  Escreva ("soma", soma)
  Escreva ("média", media)
Fim
```

No Java:

```
1  import javax.swing.JOptionPane;
2
3  public class MatrizEx3 {
4
5      public static void main(String[] args) {
6          // exercício 1
7
8          //declaração de variáveis
9
10         int linha = 10, coluna = 10;
11         int a[][] = new int [linha][coluna];
12         int i, j, soma;
13         double media;
14
15         //entrada de dados
16         for (i = 0; i < linha; i++) {
17             for (j  = 0; j < coluna; j++) {
18                 a[i][j]= Integer.parseInt(JOptionPane.showInputDialog
19                         ("Entre com o elemento [" + i +"][" + j + "] da matriz"));
20             }//fim do for
21         }//fim do for
22
23         //cálculo
24         soma = 0;
25         for (i = 0; i < linha; i++) {
26             for (j  = 0; j < coluna; j++) {
27                 soma = soma + a[i][j];
28             }//fim do for
29         }//fim do for
30         media = soma / ( linha * coluna);
31
32         //saída de dados
33         JOptionPane.showMessageDialog(null, "A soma de todos os elementos é " + soma
34                 + "\nMédia dos elementos: " + media);
35     }//fim do método main
36
37 }//fim da classe
38
```

4.

Pseudocódigo

```
Programa MatrizEx2
Declare
   dados como conjunto [1..100000][1..3] de caractere
   linha, coluna como inteiro
   Início
   Escreva("inserindo os dados na Matriz")
   Para linha = 1 Até 100000 Faça
         Escreva ("Entre com o nome")
         Leia dados[linha, 1]
         Escreva ("Entre com a Cidade")
         Leia dados[linha, 2]
         Escreva ("Entre com o Estado")
         Leia dados[linha, 3]
      Fim-Para
   Fim-Para

   Escreva("dados...")
   Para linha = 1 Até 100000 Faça
      Para coluna = 1 Até 3 Faça
         Se (coluna = 1) Então
            Escreva ("Nome ", dados[linha,coluna])
         Se (coluna = 2) Então
            Escreva ("Cidade ", dados[linha,coluna])
         Senão
            Escreva ("Estado ", dados[linha,coluna])
      Fim-Para
   Fim-Para
Fim
```

No Java:

```java
import javax.swing.JOptionPane;

public class MatrizEx4 {

    public static void main(String[] args) {
        //Exercício4

        //declaração de variáveis
        String dados[][] = new String [100000][3];
        int linha, coluna;
        String saida = "Nome: ";

        //Entrada de dados
        JOptionPane.showMessageDialog(null, "Entrada de dados");

        for (linha = 0; linha < 100000; linha++) {
            //não usaremos o for para a coluna enderecaremos a coluna diretamente
            dados[linha][0] = JOptionPane.showInputDialog("Entre com o Nome:");
            dados[linha][1] = JOptionPane.showInputDialog("Entre com a cidade:");
            dados[linha][2] = JOptionPane.showInputDialog("Entre com o Estado:");
        }//fim do for

        //exibição dos dados

        for(linha = 0; linha <100000; linha++) {
            for (coluna = 0; coluna < 3; coluna++) {
                saida = saida.concat(dados[linha][coluna]);
                if (coluna == 0)
                    saida = saida.concat("\nCidade: ");
                if (coluna == 1)
                    saida = saida.concat("\nEstado: ");
            }//fim do for
            JOptionPane.showMessageDialog(null, "Dados:\n" + saida);
            saida = "Nome: ";
        }//fim do for
    }
}
```

Resposta do desafio:

```java
import javax.swing.JOptionPane;

public class Matriz {

    public static void main(String[] args) {
        //declaração de variáveis
        double num [] [] = new double [4][4];
        int linha, coluna;
        String saida = "";

        //entrada de dados
        JOptionPane.showMessageDialog (null, "Inserindo os dados na Matriz");
        for (linha = 0; linha < 4; linha++) {
            for (coluna = 0; coluna < 4; coluna++) {
                num[linha][coluna] = Double.parseDouble(JOptionPane.showInputDialog("Entre com o número" +
                 "\nlinha " + linha +
                 "\ncoluna " + coluna));
            }//fim do for
        }//fim do for

        //saída de dados
        JOptionPane.showMessageDialog (null, "Mostrando os dados na Matriz");
        for (linha = 0; linha < 4; linha++) {
            for (coluna = 0; coluna < 4; coluna++) {
                saida = saida.concat("    " + num[linha][coluna]);
                if (coluna == 3) {
                    saida = saida.concat("\n");
                }//fim do if
            }//fim do for
        }//fim do for
        JOptionPane.showMessageDialog(null, saida);
    }//fim do método main
}
```

Estes exercícios devem ser entregues de forma on-line como atividades da agenda.

Exercício 1

Angélica precisa escrever um programa no qual a primeira linha de uma matriz bidimensional 5 x 5 deve ser colocada na última e vice-versa. Faça a codificação do programa em pseudocódigo e em Java.

Exercício 2

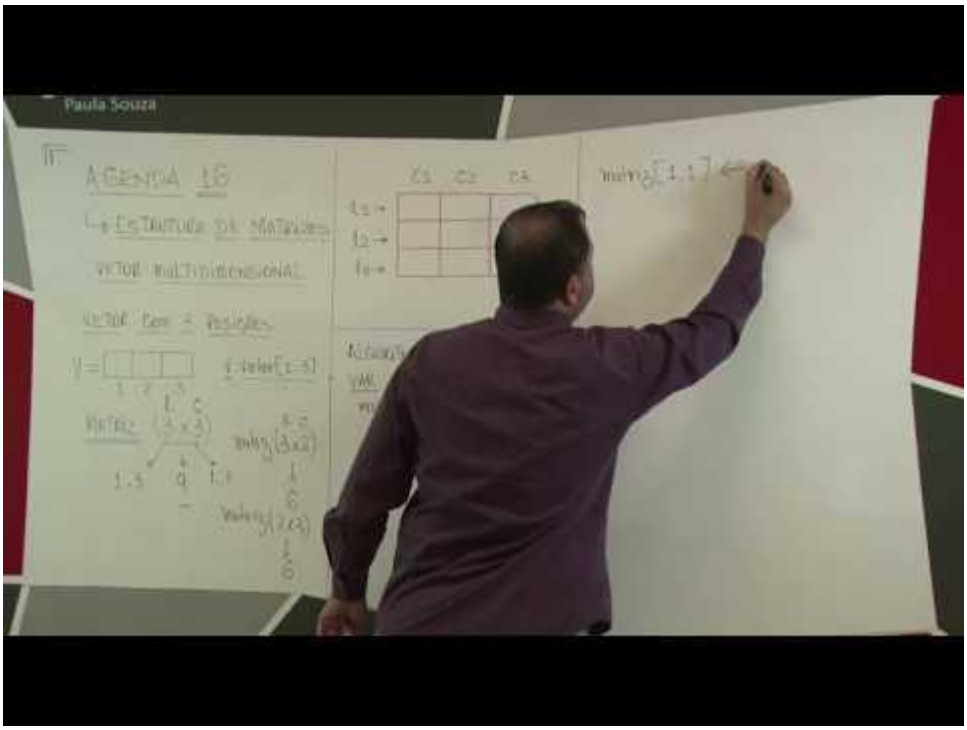
Imagem 27

José trabalha em uma empresa se classificação de dados estatísticos e necessita saber qual é o maior valor encontrado em uma tabela. A tabela consiste em uma matriz bidimensional de 10x10 elementos reais. Elabore um programa que realize a leitura destes elementos e exiba com resultado final o maior elemento contido na matriz analisada.

AMPLIANDO HORIZONTES

Não deixe também de assistir aos vídeos:

Informática – Módulo I – Agenda 16



Link: https://www.youtube.com/watch?time_continue=2&v=sgvfbApx7Xw. Acessado em 21/11/2017

Lógica de Programação – Vetores e Matrizes

Link: https://www.youtube.com/watch?v=7i3YfiLy1vE. Acessado em 02/11/2017.

Informática - Módulo I – Revisão

Link: https://www.youtube.com/watch?v=mV4GGC5UARQ. Acessado em 21/12/2017.

Para aprofundamento dos temas discutidos nesta aula, seguem abaixo algumas dicas de filmes, livros e artigos que se relacionam com o conteúdo estudado. Estas dicas são muito importantes para você!

Vídeo:

Procure no YouTube um vídeo denominado ;“ Matrizes + VisuAlg”, disponível em : http://www.youtube.com/watch?v=C0fqsjFF1YE. Acessado em 02/11/2017.

Procure no Youtube um vídeo denominado “Matrizes – Curso de algoritmos #15 – Gustavo Guanabara”. disponível em : https://www.youtube.com/watch?v=hkE9WrjpAAk. Acessado em 02/11/2017.

Livros:

FORBELLONE, André L. V.; ELBERSPACHER, Henri Frederico. Lógica de Programação: A Construção de Algoritmos e Estrutura de Dados. Editora Pearson, 2000.

MANZANO, José Augusto N. G; OLIVEIRA, Jayr Figueiredo. Algoritmos: Lógica para Desenvolvimento de Programação. Editora Érica, 2007.

PUGA, SANDRA; RISSETTI, GERSON. Lógica de Programação e Estruturas de Dados com Aplicações em Java. Editora Pearson. 2009.

SCHILDT, HEBERT. Java para Iniciantes. Editora Bookman.2015.

FONTES IMAGÉTICAS:

Imagem 01: Livro Telecurso Tec: Módulo 1 – Andrea Ramal; Silvina Ramal.

Imagem 02: Arquivo GEEaD

Imagem 03: Livro Comércio Telecurso Tec: Módulo 2 – Andrea Ramal; Silvina Ramal.

Imagem 04: Livro Telecurso Tec: Módulo 1 – Andrea Ramal; Silvina Ramal.

Imagem 05: Arquivo GEEaD

Imagem 06: Livro Telecurso Tec: Módulo 1 – Andrea Ramal; Silvina Ramal.

Imagem 07: http://vidadeprogramador.com.br/2011/03/22/logica-de-programacao/

Imagem 08: Arquivo GEEaD

Imagem 09: Adaptado de http://profissoes.colorir.com/informatica/computador-2.html. Acessado em 29/12/2017.

Imagem 10: Adaptado de http://dimassantos.com.br/fichas-sujas-de-olho-nas-brechas-da-lei-da-ficha-limpa/. Acessado em 29/12/2017.

Imagem 11: Livro Telecurso Módulo 2 do Curso de Comércio

Imagem 12: Arquivo GEEaD

Imagem 13: https://producaodejogos.com/fazendo-jogos-e-aplicativos-com-unity-3d/. Acessado em 29/12/2017.

Imagem 14: http://kellyreceitasdebrigadeiros.blogspot.com.br/2013/06/brigadeiro-de-panela.html. Acessado em 29/12/2017.

Imagem 15: https://www.devmedia.com.br/alo-mundo-entendendo-o-java-de-uma-forma-diferente/24032. Acessado em 29/12/2017.

Imagem 16: https://www.youtube.com/watch?v=Efd3GUAyGVM. Acessado em 29/12/2017.

Imagem 17: Arquivo GEEaD

Imagem 18: Arquivo GEEaD

Imagem 19: Arquivo GEEaD

Imagem 20: Arquivo GEEaD

Imagem 21: https://ogimg.infoglobo.com.br/in/21586653-a6d-038/FT1086A/420/bagagemproteste.jpg. Acessado em 02/11/2017.

Imagem 22: http://bichinhosdejardim.com/repeticao. Acesso em 07/08/2017.

Imagem 23: Arquivo GEEaD

Imagem 24: http://www.acheienderecos.com.br/imagens-e-homenagem-do-dia-do-amigo-20-de-julho/. Acessado em 02/11/2017.

Imagem 25: Arquivo GEEaD

Imagem 26: Arquivo GEEaD

Imagem 27: http://www.jaguarutilidades.com.br/_temas/t_site/imagem/estatistica-presente-em-todas-as-areas-da-vida.jpg . Acessado em 13/11/2017.

REFERÊNCIAS

DELLA CROCE FILHO, Ralfe; RIBEIRO Carlos Eduardo. Informática - Programação de Computadores. Manual de Informática Centro Paula Souza, V. 4. Fundação Padre Anchieta, 2010.

FORBELLONE, André L. V.; ELBERSPACHER, Henri Frederico. Lógica de Programação: A Construção de Algoritmos e Estrutura de Dados. Editora Pearson, 2000.

MANZANO, José Augusto N. G; OLIVEIRA, Jayr Figueiredo. Algoritmos: Lógica para Desenvolvimento de Programação. Editora Érica, 2007.

PUGA, SANDRA; RISSETTI, GERSON. Lógica de Programação e Estruturas de Dados com Aplicações em Java. Editora Pearson. 2009.

SCHILDT, HEBERT. Java para Iniciantes. Editora Bookman.2015.