1. Orientações gerais

Os roteiros foram desenvolvidos com o intuito de não apenas permitir a prática da programação de computadores, como também apresentar um material de apoio a disciplina teórica. Dessa forma, cada roteiro tem uma descrição da matéria que será praticada (estrutura da linguagem JAVA), bem como, exemplos práticos da linguagem.

Assim sendo, espera-se que todo aluno inicie o roteiro a partir da leitura do material e, em seguida, tente compilar e entender os exemplos. Dessa forma, muitas dúvidas serão sanadas. Mesmo assim, caso tenha necessidade solicite a ajuda do professor e/ou monitor da disciplina.

2. Instruções para Elaboração/ Entrega dos roteiros

- O desenvolvimento dos roteiros deverá seguir o cronograma apresentado.
- O aluno deverá entregar seu roteiro individualmente, podendo realizar um trabalho em equipe, mas cada aluno deve ser capaz de fazer e apresentar o seu trabalho.
- **Cópias dos roteiros práticos e listas de exercícios serão penalizadas com a nota ZERO para ambos os alunos (o que fez e o que copiou).**
- Os roteiros práticos deverão ser entregues via classroom da disciplina prática SEMPRE na data previamente agendada pelo professor. O professor poderá sortear alguns alunos para apresentar sua solução de alguns exercícios específicos.
- **IMPORTANTE:** os nomes dos arquivos a serem entregues deve seguir a seguinte regras: "nome_aluno_numero_do_exercicio", exemplo: O exercício número 1 do aluno José deverá ser nomeado como Jose1, e assim por diante...
- Para desenvolver os roteiros o aluno precisará instalar o JGRASP, NetBeans, eclipse ou qualquer outra IDE de desenvolvimento em seu computador.
- Instaladores disponibilizados no classroom da disciplinas.

3. Referências Bibliográficas

FARRER, H., BECKER, C. G., FARIA, E. C., MATOS, H. F. M, MAIA, M. L. Programação Estruturada de Computadores - Algoritmos Estruturados. 3º Edição. Editora LTC, 2010.MIZRAHI, V. V. Treinamento em Linguagem C++ - Módulo 1. 2 Edição. Editora Pearson, 2006. ISBN 9788576050452

ASCENCIO, A. F. G.; CAMPOS, E. A. V. Fundamentos da programação de computadores: algoritmos, Pascal e C/C++ e Java. 3 Edição. São Paulo: Pearson Prentice Hall, 2012. ISBN 9788564574168.

CORMEN, T., LEISERSON, C. E., RIVEST, R. L., STEIN, C. Algoritmos: Teoria e Prática. 3 Edição. Editora Campus, 2012. ISBN 978-85-352-3699-6

DEITEL, H. M., DEITEL, P.J. C++ – Como Programar. 5ºEd. Pearson, 2006. ISBN 9788576050568

MIZRAHI, V. V. Treinamento em Linguagem C++ - Módulo 1. 2 Edição. Editora Pearson, 2006. ISBN 9788576050469

KNUTH, D. E. The art of computer programming: fundamental algorithms. 3 ed. vol. 01-04. Editora Pearson, 2011. ISBN 9780321751041

ZIVIANI, N. Projeto de Algoritmos com Implementações em Pascal e C. Editora Cengage Learning, 2011. ISBN 8522110506

ZIVIANI, N. Projeto de Algoritmos com Implementações em Java e C++. Editora Cengage Learning, 2006. ISBN 8522105251

Roteiro 1 - Como iniciar uma aplicação em JGRASP ou NetBeans

Estrutura do código JAVA

```
import nome_do_pacote_das_classes;
public class nome
{
      public static void main (String args[])
      {
        bloco_de_comandos;
      }
}
```

Os pacotes de classes são arquivos contendo diferentes classes que possuem vários métodos, ou seja, funções, os quais podem ser utilizados nos programas escritos em Java. A diretiva import permite que o programa reconheça as classes do pacote e, consequentemente, a utilização de seus métodos.

É importante salientar que a linguagem Java é sensível a letras maiúsculas e minúsculas, ou seja, considera letras maiúsculas diferentes de minúsculas (por exemplo, a é diferente de A). Sendo assim, cada comando tem a própria sintaxe, que, às vezes, é somente com letras minúsculas e outras vezes com letras maiúsculas e minúsculas.

- JGRASP

Uma aplicação do tipo console é uma aplicação que roda no Prompt de comando. Ela não utiliza **interface gráfica** do Windows. Este documento tem o objetivo de ensinar passo-a-passo como iniciar o desenvolvimento de uma aplicação de linha de comando usando o JGRASP e NetBeans.

1. Inicie o JGRASP clicando no ícone na área de trabalho ou no menu iniciar. Ao iniciar, o programa abrirá uma tela parecida com a exibida na **Figura 1.**

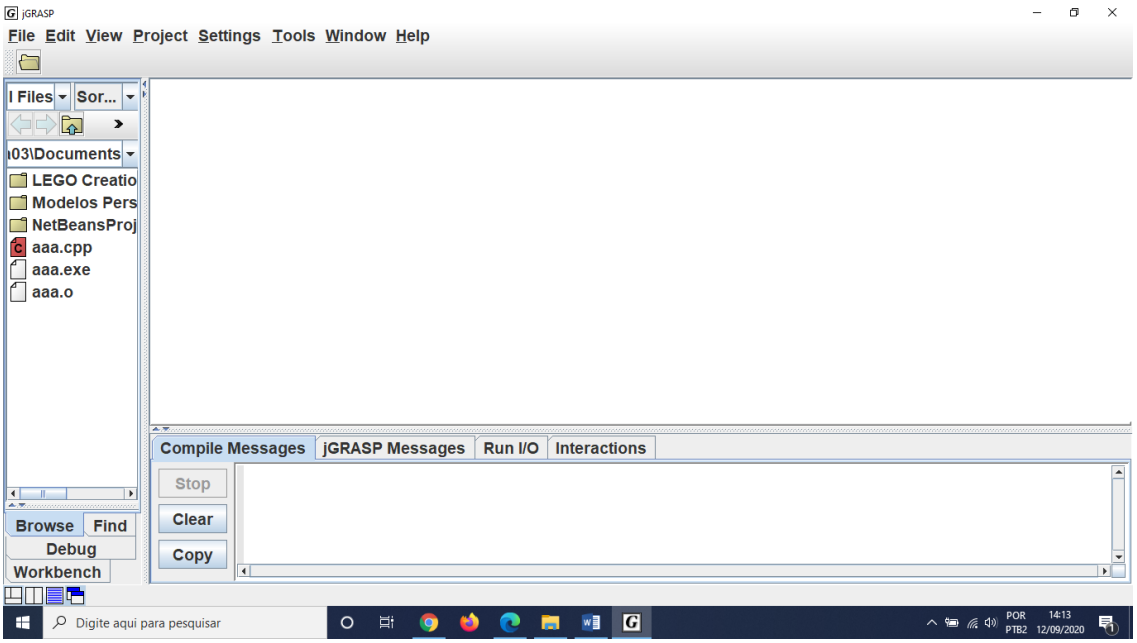


Figura 1 – Tela inicial da IDE JGRASP

2. Para criar um novo projeto, pressione o botão **FILE – New - JAVA.** Em seguida aparecerá a tela mostrada na **Figura 2.**

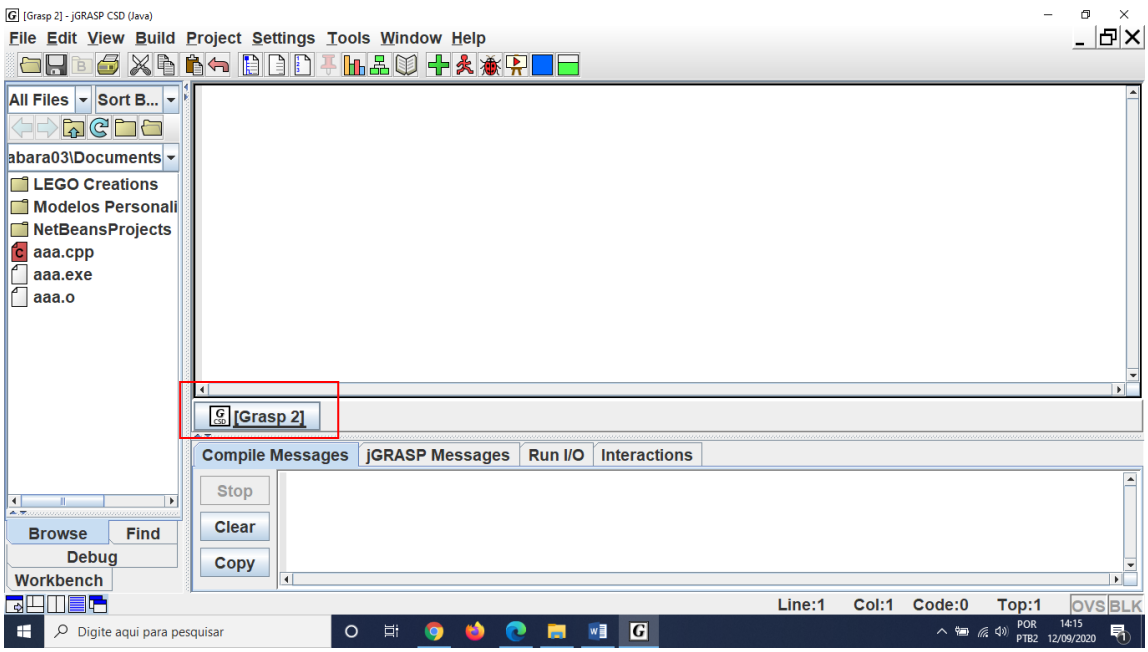


Figura 2 – Criação do arquivo fonte no JGRASP

Repare que somente criou-se um arquivo editável, conforme marcado na caixa em vermelho. Este é o arquivo principal que conterá o código fonte.

3. Na Figura 3, apresentamos um código inicial com uma mensagem de boas-vindas.

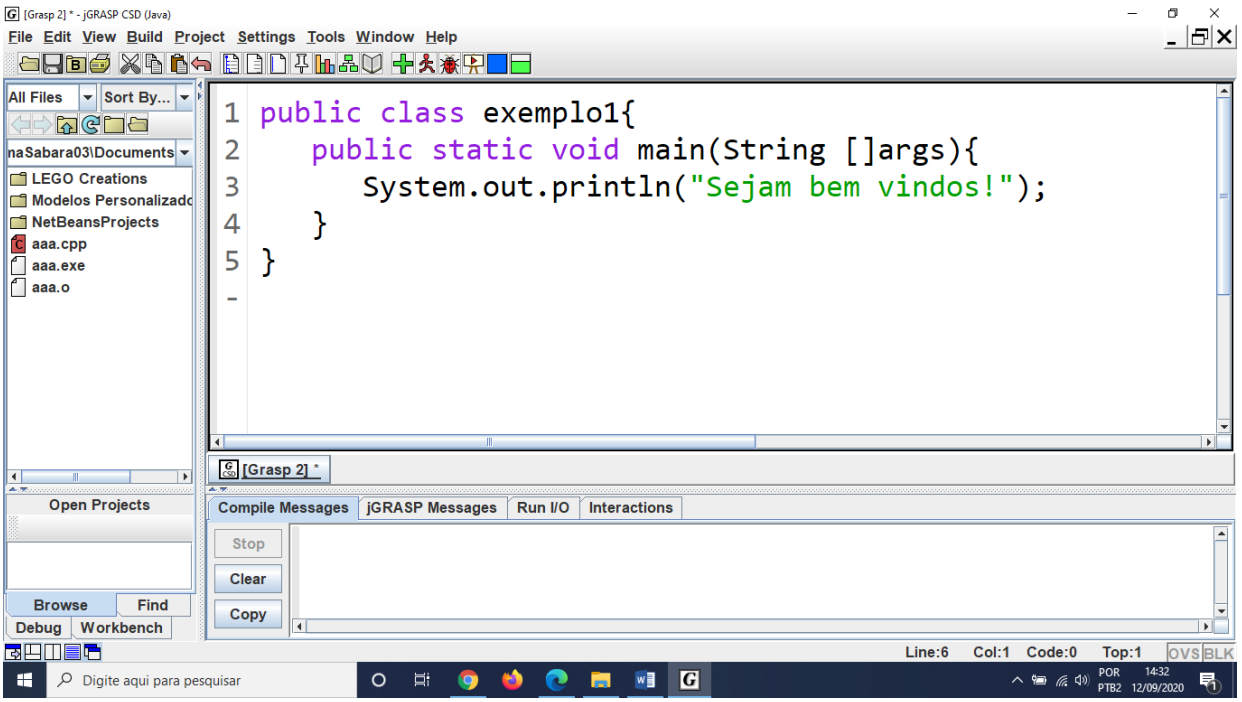


Figura 3 – Código inicial

Após digitar o código, é necessário salvá-lo para compilar. Para salvar o código, vá em **File – Save as.** Após isso aparecerá a seguinte tela (Figura 4). Nesta tela você deverá escolher o local para salvar o arquivo **(Vermelho)** e o nome do arquivo (**Verde**). O nome do arquivo deve ser exatamente o mesmo nome da classe, neste exemplo, o nome do arquivo deve era exemplo1.java .

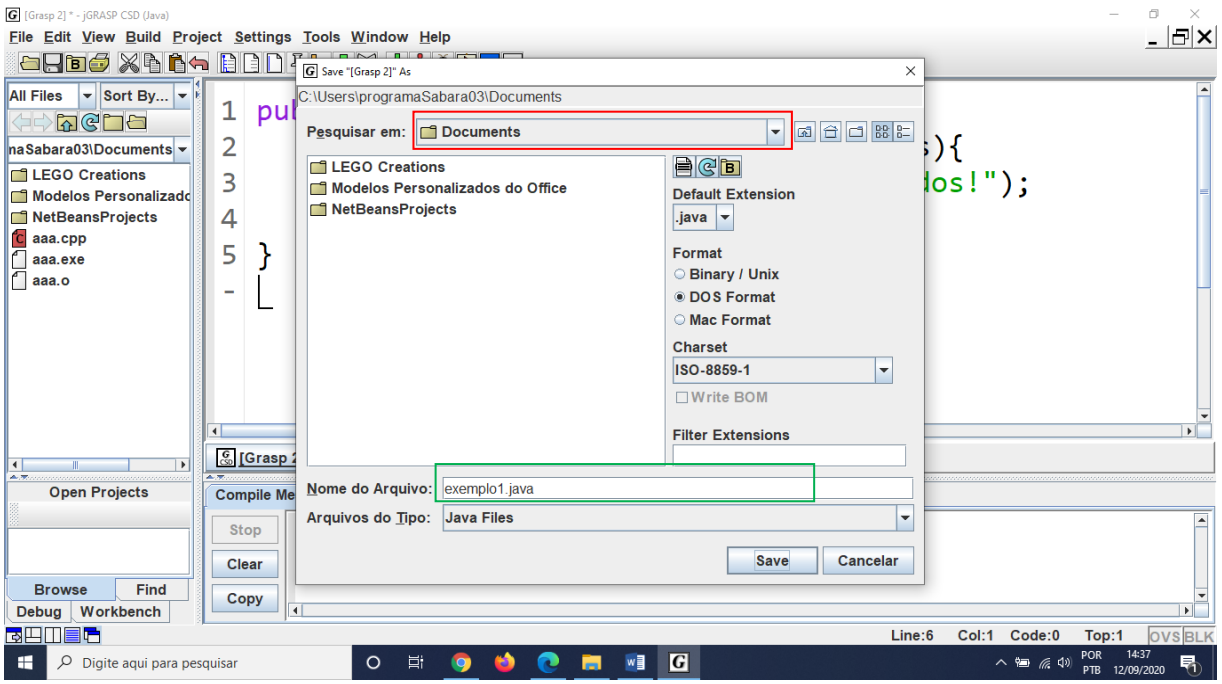


Figura 4 – Salvando arquivo

Após salvar, o arquivo fonte aparecerá na aba ao lado (vermelho) e pode ser compilado e executado (seta). Para isso clique no ícone do bonequinho vermelho correndo. Com isso, a mensagem é exibida na aba de execução (verde).

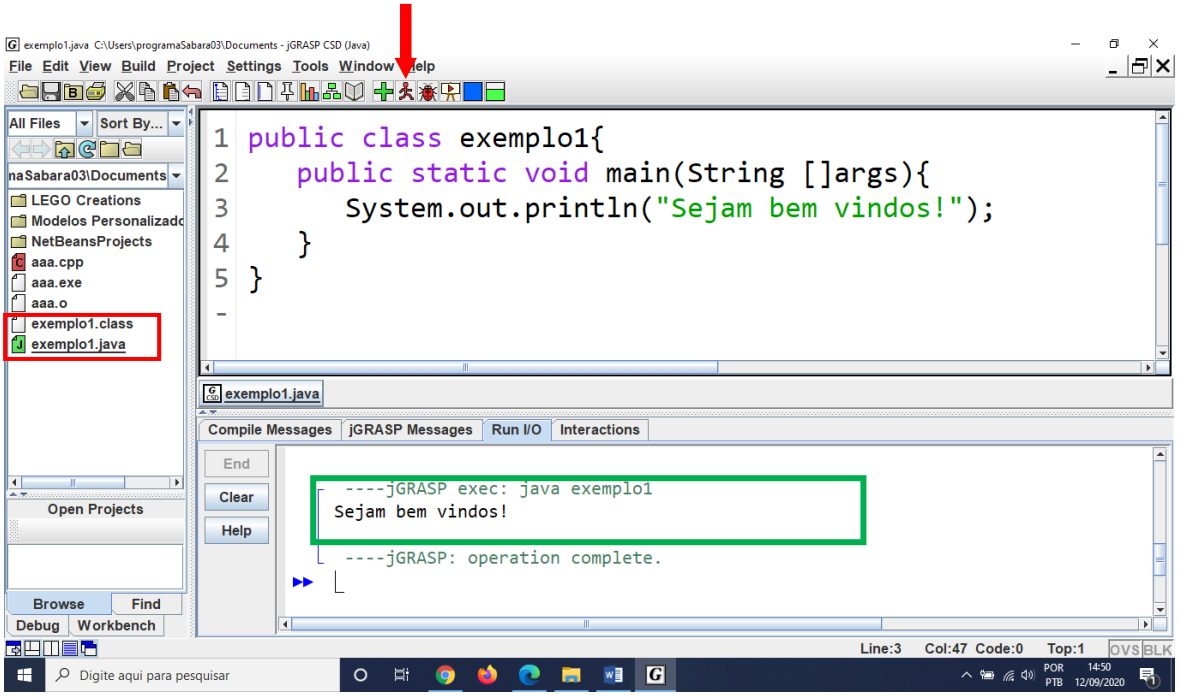


Figura 5 – Compilando código

4. Identificamos no código acima os elementos: **public class** e o método (função) **main**.
   - Todos os comandos da linguagem JAVA devem estar dentro de classes, o que implica, portanto, que um programa deve ter no mínimo uma classe e um método (função main). A função main tem o nome de **main**, sendo esse nome que a identifica.
   - Todas as declarações e comandos da linguagem devem ser terminados por **";" (ponto e vírgula)**. Esse sinal serve não apenas como separador nas declarações, mas também como identificadores da composição de sequência entre os comandos, isto é, primeiro é executado um e depois o outro.
5. Diferente de outras IDE's mais avançadas, como NetBeans ou eclipse), no JGRASP devemos digitar todo o código necessário para execução da implementação. Nas outras IDE's existe um código padrão ao se iniciar um novo projeto contendo somente o "esqueleto" de um programa básico (veremos a seguir).
6. No código exemplo que utilizamos, inserimos uma ação simples ao programa. Esse comando faz a impressão de uma mensagem na tela. O comando citado é **System.out.println("Sejam bem vindos!");** que está digitado entre as chaves de início e fim do método/função **main**.

- NetBeans

Diferentemente do JGRASP, o NetBeans é um IDE‘s mais avançada. Ela é dita intelesense, pois autocompleta vários comandos, além de mostrar os erros de sintaxe “sem a necessidade de compilação” do código.
O Editor Java do NetBeans ajuda a completar e gerar código rapidamente usando a funcionalidade autocompletar código "inteligente". Em um senso geral, o recurso autocompletar código é muito útil quando você deseja preencher o código faltantes, examinar as opções disponíveis no contexto de sua aplicação, e gerar blocos de código quando necessário. A seguir apresentamos um passo a passo para iniciar uma aplicação básica na referida IDE

1. Inicie o NetBeans clicando no ícone na área de trabalho ou no menu iniciar. Ao iniciar, o programa abrirá uma tela parecida com a exibida na **Figura 6.**

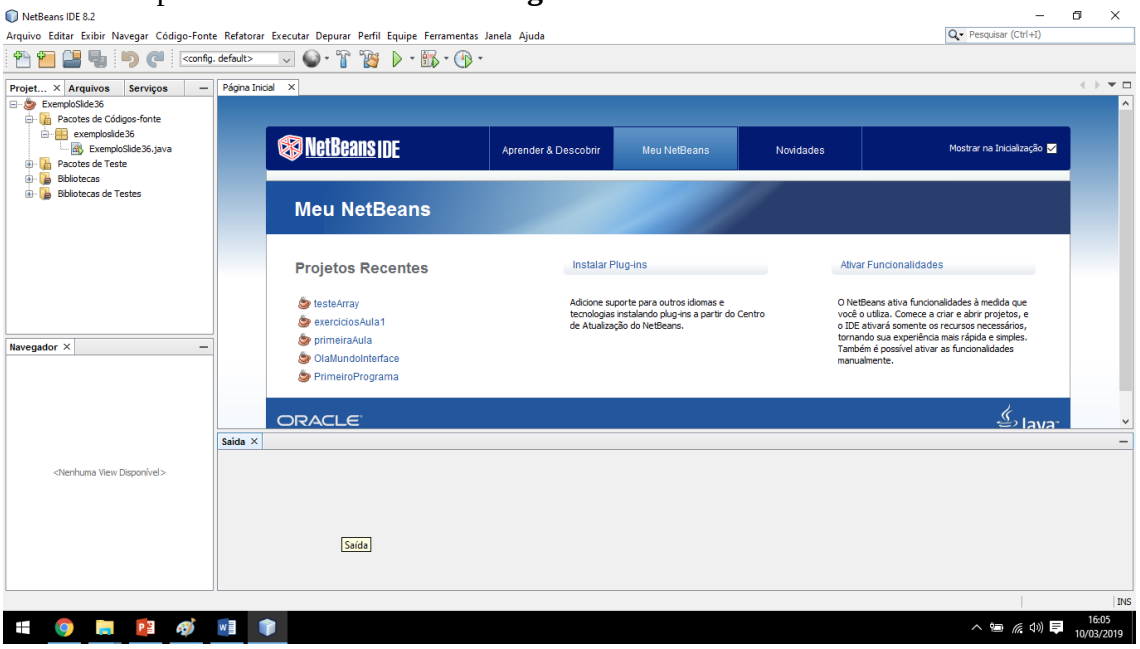


Figura 6 - Tela Inicial do NetBeans

2. Para criar um novo projeto, pressione o botão **Arquivo – Novo Projeto.** Em seguida aparecerá a tela mostrada na **Figura 7.**

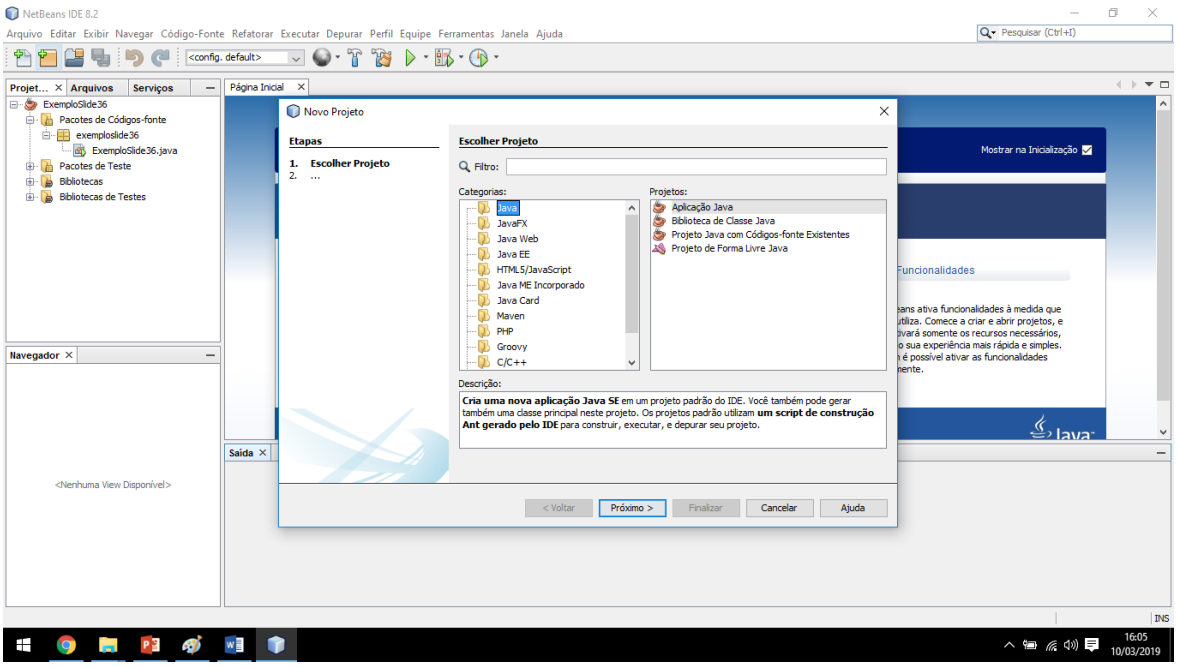


Figura 7 - Escolhendo tipo de aplicação

3. Selecione a opção "Java"->"Aplicação Java" conforme indicado na **Figura 7**, e clique em **Próximo**. Em seguida aparecerá a tela da **Figura 8.**

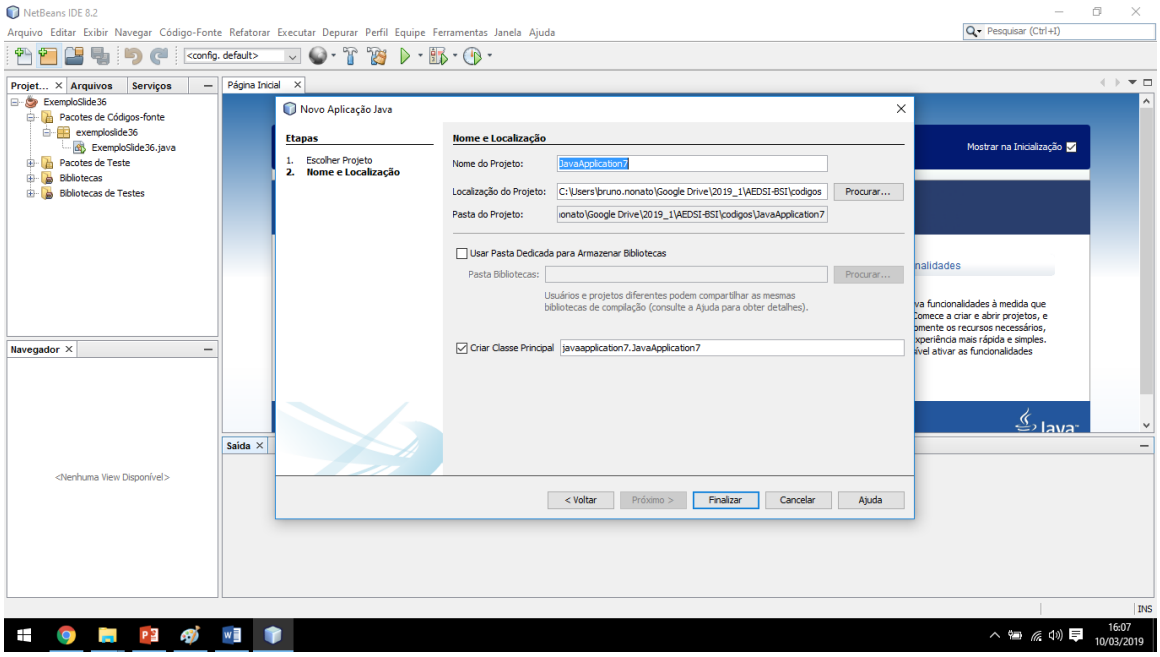


Figura 8 – Nome do Projeto

4. Nessa tela, preencha o campo **Nome do Projeto** com um **"nome para o projeto"**. O nome do projeto não pode conter caracteres especiais (conforme veremos nas regras de formação de identificadores). No campo **Localização do Projeto** escolha o local em que será salvo o projeto. Deixe marcado a opção de Criar

Classe Principal. Clique em **Finalizar**. Executando tais passos, o projeto será criado e a tela da **Figura 9** será gerada.

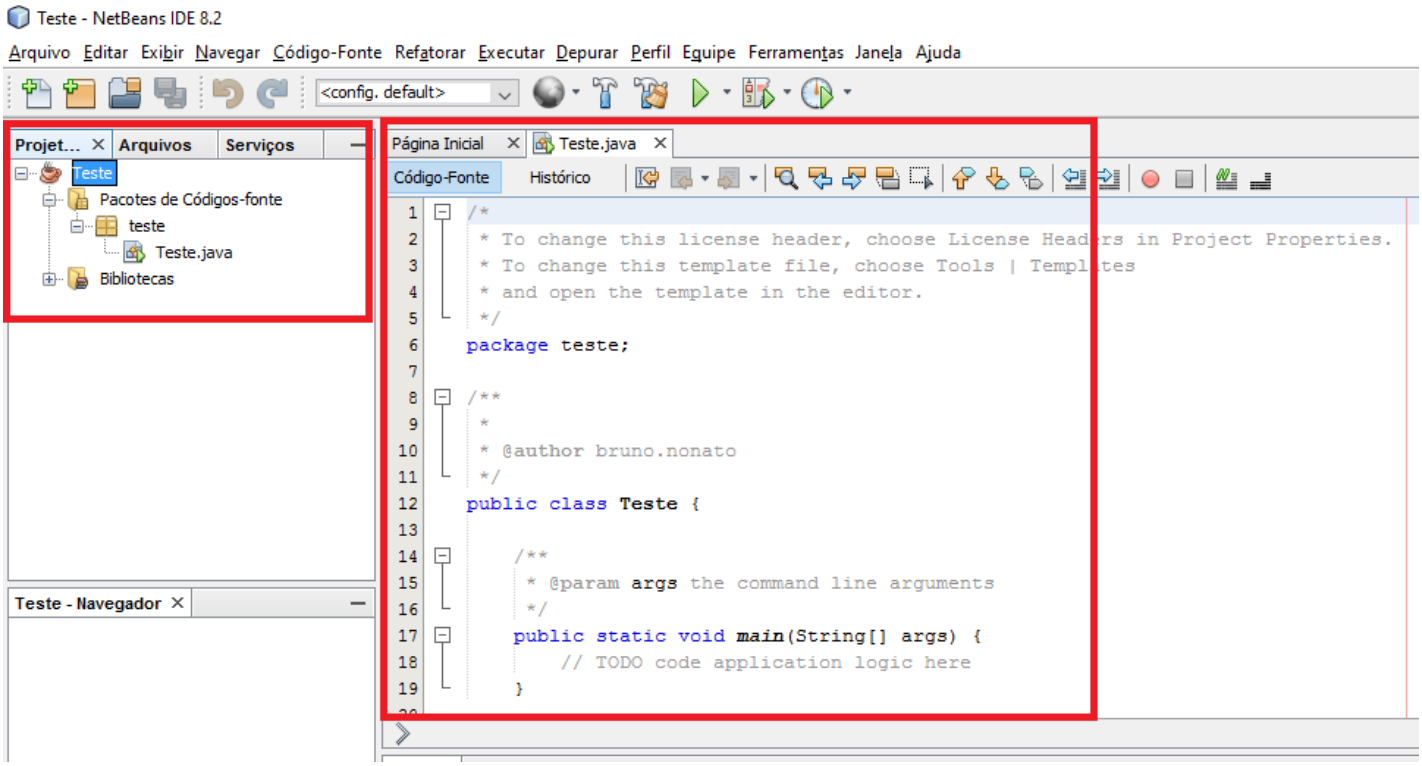


Figura 9 - Tela final de criação de projeto

5. Após a criação do projeto, a Figura 9 representa os componentes do mesmo. Na parte esquerda da figura estão as informações relacionadas aos pacotes e bibliotecas do projeto e na parte direita está o código fonte do mesmo. Como pode-se perceber o projeto já cria um código padrão.

6. Identificamos no código acima os elementos: **package**, **public class**, a função **main**.
   - Todos os comandos da linguagem JAVA devem estar dentro de classes, o que implica, portanto, que um programa deve ter no mínimo uma classe e um método (função main). A função main tem o nome de **main**, sendo esse nome que a identifica. Note que o NetBeans já adiciona automaticamente uma função de nome **main** por onde a execução inicia.
   - Todas as declarações e comandos da linguagem devem ser terminados por **";" (ponto e vírgula)**. Esse sinal serve não apenas como separador nas declarações, mas também como identificadores da composição de sequência entre os comandos, isto é, primeiro é executado um e depois o outro.
7. O NetBeans já nos dá um código padrão, é necessário a compilação e execução do mesmo. Para isso, clique no botão limpar e construir  e clicar no botão  (este processo compila e executa o programa).
8. Repare que o programa padrão criado pelo NetBeans possui somente o "esqueleto" de um programa básico, ou seja, caso seja executado não fará nenhuma ação explícita. Para que você veja alguma ação do programa, experimente inserir um código de impressão na tela. Para isso adicione o comando **System.out.println("Olá pessoal, sejam bem vindos à aula de programação!");** entre as chaves de

início e fim da função **main**. Após isso, compile e execute o código novamente. O programa mostrará a mensagem na tela. Nesse caso, "Olá pessoal, sejam bem vindos à aula de programação!".

```
1   package teste;
2
3   public class Teste {
4       public static void main(String[] args) {
5           System.out.println("Olá pessoal, sejam bem vindos à aula de programação");
6       }
7
8   }
9
```

Figura 10 – Programa básico – "Boas Vindas"

Exemplos

Como exercício, vamos digitar o seguinte código do exemplo 1 da Figura 11 e tentar entender o que está ocorrendo.

```
1   package teste;
2   /*
3    * @author bruno.nonato
4    */
5   public class Teste {
6       public static void main(String[] args) {
7           System.out.println("IFMG - SABARÁ");//Comando para imprimir na tela passando cursor para a linha de baixo
8           System.out.print("Aula prática de programação");//Comando para imprimir na tela matendo cursor na linha de atual
9           System.out.print("PROGRAMA 1");//Comando para imprimir na tela matendo cursor na linha de atual
10      }
11  }
12
```

Figura 11 - Exemplo 1

- Nesse programa aparecem os comandos para se colocar comentários no código, sã eles: **/*** **e** */ para comentários de blocos (mais de uma linha), conforme pode ser visto das linhas 2 à 4 e // para comentários de apenas uma linha como nas linhas 7, 8 e 9.
- Os textos colocados nos comentários não são considerados pelo compilador, eles servem somente para acrescentar informações a respeito do programa ou de algum trecho de código.
- Repare que o comando de impressão na tela foi utilizado de 2 formas diferentes:
  - System.out.print**ln**("MENSAGEM");
  - System.out.print("MENSAGEM");

  Na primeira forma, o comando imprime a mensagem na tela e continua na linha abaixo; já na segunda forma o comando imprime a mensagem na tela e mantém o "cursor" na linha atual.
- Compile e execute o código para ver o resultado!
- Experimente trocar os comandos das linhas 8 e 9 para o System.out.print**ln**

Agora digite o exemplo 2 tente entender o que acontece.

```java
package teste;
import java.util.*;
/*
@author bruno.nonato
Data: 10/02/2019
Descrição: Programa que imprime na tela o triplo de um número digitado pelo usuário
Entrada: numero; Processamento: triplo = numero*3; Memoria: numero, triplo; Saída: triplo
 */
public class Teste {
    public static void main(String[] args) {
        int numero, triplo; //declarando duas variáveis inteiras
        Scanner teclado = new Scanner(System.in); //adicionando leitura pelo teclado
        System.out.println("Digite um número: ");//Solicitando usuário para digitar um número
        numero = teclado.nextInt(); //lendo a entrada do teclado e armazenando na variável "numero"
        triplo = numero*3; //multiplicando o número lido por 3 e armazenando na variável "triplo"
        System.out.println("O triplo de " + numero + " é: "+triplo); //imprimindo o resultado na tela
    }
}
```

Figura 12 - Exemplo 2

- Nesse exemplo, já se introduziu a utilização de **variáveis do programa**. Essas e outras questões serão vistas com mais detalhes mais à frente na próxima seção. As **variáveis**, a grosso modo, nada mais são que espaços de memória reservados pelo compilador para utilização do programa.

Estrutura Sequencial

Para prosseguirmos com a disciplina alguns conceitos devem ser dados para melhor entendimento do conteúdo.

Características da Linguagem

- Composta basicamente por **variáveis** e **funções**;
- A diretiva **import** **"avisa"** ao compilador que serão usados procedimentos de uma determinada biblioteca. As bibliotecas são arquivos contendo várias funções que podem ser incorporadas no programa sem a necessidade de implementação.

```
import java.utili.*;
import java.swing.*;
etc...
```

- A linguagem JAVA é **sensível ao caso (case sensitive),** ou seja, considera que letras maiúsculas e minúsculas são diferentes

MDC != Mdc != mdc

main != Main != maiN

- Todos os comandos da linguagem devem ser escritos com letras minúsculas.

Declaração de Variáveis

Variáveis são objetos que não possuem valor fixo. Representam uma região na memória e possuem um **tipo** e um **identificador** associado. Além disso, variáveis simples podem armazenar somente um valor a cada momento. Portanto, declarar uma variável significa dizer, a grosso modo, que estamos reservando um espaço de memória que possui um identificador (nome) para armazenar valores de um determinado tipo. É importante notar que as variáveis são declaradas após a especificação do seu tipo.

Exemplo:

```
1  import java.util.Scanner;
2  public class ExemploTipoPrimitivo2 {
3    public static void main(String[] args) {
4        int idade;
5        float peso;
6        double salario;
7        char sexo;
8        String nome;
9    }
10 }
```

Figura 1 – Declaração de variáveis

Tipos Básicos

		Valores possíveis				
Tipos	**Primitivo**	**Menor**	**Maior**	**Valor Padrão**	**Tamanho**	**Exemplo**
Inteiro	byte	-128	127	0	8 bits	byte ex1 = (byte)1;
	short	-32768	32767	0	16 bits	short ex2 = (short)1;
	int	-2.147.483.648	2.147.483.647	0	32 bits	int ex3 = 1;
	long	-9.223.372.036.854.770.000	9.223.372.036.854.770.000	0	64 bits	long ex4 = 1l;
Ponto Flutuante	float	-1,4024E-37	3.40282347E + 38	0	32 bits	float ex5 = 5.50f;
	double	-4,94E-307	1.79769313486231570E + 308	0	64 bits	double ex6 = 10.20d; ou double ex6 = 10.20;
Caractere	char	0	65535	\0	16 bits	char ex7 = 194; ou char ex8 = 'a';
Booleano	boolean	false	true	false	1 bit	boolean ex9 = true;

Figura 2 – Tipos de dados primitivos

Regras básicas para formação de identificadores

- Permitido: números, letras e caractere sublinhado (underline) e $
- Primeiro caractere não pode ser um valor numérico
- Não permite-se uso de espaço em branco e caracteres especiais (@,#,%, etc)
- Não é permitido a utilização de palavras reservadas (pertencem a uma linguagem de programação específica)

Exemplos corretos:

float notaMedia;
int numero;
char nome;

Exemplo incorretos:

int nota Media;
float @numero;
char 2nome;

Comandos de Saída

Os comandos de saída são utilizados para imprimirmos algo da memória do computador para o usuário. A grosso modo, o comando "pega" algo da memória do computador e mostra de alguma maneira para o usuário (normalmente a tela do computador).

Vamos abordar em nosso curso 4 tipos de saídas de dados utilizadas em JAVA:

- System.out.**print :** imprime na tela e mantém o "cursor" na linha corrente

```
1  public class ExemploTipoPrimitivo2 {
2    public static void main(String[] args) {
3        double salarioMinimo = 999.54;
4        System.out.print("O salario mínimo é "+salarioMinimo);
5    }
6  }
```

```
ExemploTipoPrimitivo2 > main >
Saída - Teste (run) ×
run:
O salario mínimo é 999.54CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

Figura 3 – Exemplo de impressão com .print

- System.out.**println:** imprime na tela e passa o "cursor" para a próxima linha

```
1  public class ExemploTipoPrimitivo2 {
2    public static void main(String[] args) {
3        double salarioMinimo = 999.54;
4        System.out.println("O salario mínimo é "+salarioMinimo);
5    }
6  }
```

```
ExemploTipoPrimitivo2 > main >
Saída - Teste (run) ×
run:
O salario mínimo é 999.54
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

Figura 4 – Exemplo de impressão com .println

Tanto o comando System.out.print quanto o System.out.println utilizam o caractere + para concatenar as impressões, ou seja, juntar as mensagens com as variáveis a serem impressas.

- System.out.**printf:** imprime na tela seguindo formatação pré-definida no comando.

```
1  public class ExemploTipoPrimitivo2 {
2    public static void main(String[] args) {
3        double salarioMinimo = 999.54;
4        System.out.printf("O salario mínimo é %.4f ", salarioMinimo);
5    }
6  }
```

```
ExemploTipoPrimitivo2 > main >
Saída - Teste (run)
run:
O salario mínimo é 999,5400 CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

Figura 5 – Exemplo de impressão com .prinf

```
1  public class ExemploTipoPrimitivo2 {
2    public static void main(String[] args) {
3        double salarioMinimo = 999.54;
4        System.out.printf("O salario mínimo é %.4f \n", salarioMinimo);
5    }
6  }
```

```
ExemploTipoPrimitivo2 > main >
Saída - Teste (run)
run:
O salario mínimo é 999,5400
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

Figura 6 – Exemplo de impressão com .prinf e \n

- System.out.**format:** imprime na tela seguindo formatação pré-definida no comando

```
1  public class ExemploTipoPrimitivo2 {
2    public static void main(String[] args) {
3        double salarioMinimo = 999.54;
4        System.out.format("O salario mínimo é %.1f \n", salarioMinimo);
5    }
6  }
```

```
ExemploTipoPrimitivo2 > main >
Saída - Teste (run)
run:
O salario mínimo é 999,5
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

Figura 7 – Exemplo de impressão com .format

Repare que os comandos System.out.printf e System.out.format possuem a mesma sintaxe. Neles toda a mensagem é escrita primeiro colocando nos devidos lugares o formato em que as variáveis serão impressas. Após a mensagem completa (conteúdo entre aspas duplas), segue-se com as variáveis esperadas na mensagem. Exemplo:

String nome = “Bruno”; int idade = 33; float peso = 72.5;

System.out.printf(“O professor %s tem idade %d e pesa %.2f ”, nome, idade, peso);

Caracteres especiais para impressão:

Os “códigos” citados acima podem ser utilizados na mensagem para melhor formatação do texto. Existem diversas outras que podem ser utilizadas.

Comando de entrada

O comando de entrada serve para passarmos informações externas para a memória do computador. A grosso modo, o comando que "pega" dados que estão fora do computador e "inserem dentro da memória do computador" (a forma mais comum é o teclado).

O JAVA carrega por padrão a biblioteca java.lang, apesar de já possuir os comandos para saída de dados, ela não possui comandos para entrada de dados. Uma das formas de entrada utilizada na linguagem JAVA é por meio da classe **Scanner**, que requer a importação do pacote java.util

import java.util.Scanner;

Sintaxe:

Scanner "nomeObjeto" = new Scanner(System.in);
"variavel" = "nomeObjeto".next"tipo"();

Exemplos:

double salarioMinimo;
Scanner teclado = new Scanner(System.in);
salarioMinimo = teclado.nextDouble();

```
1  import java.util.Scanner;
2  public class ExemploTipoPrimitivo2 {
3    public static void main(String[] args) {
4        Scanner teclado = new Scanner(System.in);
5        String nomeProfessor;
6        nomeProfessor = teclado.nextLine(); //Lendo String
7        int idade;
8        idade = teclado.nextInt(); //Lendo Inteiro
9        float salario;
10       salario = teclado.nextFloat(); //Lendo float
11       double peso;
12       peso = teclado.nextDouble(); //Lendo double
13       System.out.println("Nome "+nomeProfessor+" idade "+idade+
14                " salario "+salario+" peso " +peso);
15   }
16 }
```

Figura 8 – Exemplo de entrada de dados com objeto Scanner

É importante salientar que todas as entradas são recebidas pela linguagem Java como um conjunto de caracteres. Assim, esses caracteres deverão ser convertidos por funções de conversão de tipos. Seguem algumas dessas funções.

Função	Funcionalidade
next()	Aguarda uma entrada em formato String com uma única palavra.
nextLine()	Aguarda uma entrada em formato String com uma ou várias palavras.
nextInt()	Aguarda uma entrada em formato inteiro.
nextByte()	Aguarda uma entrada em formato inteiro.
nextLong()	Aguarda uma entrada em formato inteiro.
nextFloat()	Aguarda uma entrada em formato número fracionário.
nextDouble()	Aguarda uma entrada em formato número fracionário.

Outra questão importante é que a leitura de dados através do objeto Scanner tem que ser cuidadosa, principalmente quando se faz leituras consecutivas. Por exemplo, na Figura seguinte tem-se a leitura da variável String consecutiva à leitura de uma variável numérica. Digite e compile o código da Figura 9 e veja que a instrução de leitura da String é "desconsiderada", pois o nextline( ) "pega" o lixo buffer do teclado e atribui para a variável. Nesse caso é necessário um comando para limpeza do buffer do teclado.

```
1  import java.util.Scanner;
2  public class ExemploTipoPrimitivo2 {
3    public static void main(String[] args) {
4        Scanner teclado = new Scanner(System.in);
5        int idade;
6        idade = teclado.nextInt(); //Lendo Inteiro
7        String nomeProfessor;
8        nomeProfessor = teclado.nextLine(); //Lendo String
9        float salario;
10       salario = teclado.nextFloat(); //Lendo float
11       double peso;
12       peso = teclado.nextDouble(); //Lendo double
13       System.out.println("Nome "+nomeProfessor+" idade "+idade+
14                " salario "+salario+" peso " +peso);
15   }
16 }
```

Figura 9 – Exemplo de problema com leitura consecutiva Número String pelo Scanner

Diante do problema, é necessário ter um comando para limpeza do buffer do teclado, conforme Figura seguinte.

```
1  import java.util.Scanner;
2  public class ExemploTipoPrimitivo2 {
3    public static void main(String[] args) {
4        Scanner teclado = new Scanner(System.in);
5        int idade;
6        idade = teclado.nextInt(); //Lendo Inteiro
7        String nomeProfessor;
8        teclado.nextLine(); //Limpa buffer do teclado para leitura da String
9        nomeProfessor = teclado.nextLine(); //Lendo String
10       float salario;
11       salario = teclado.nextFloat(); //Lendo float
12       double peso;
13       peso = teclado.nextDouble(); //Lendo double
14       System.out.println("Nome "+nomeProfessor+" idade "+idade+
15                " salario "+salario+" peso " +peso);
16   }
17 }
```

Figura 10 – Exemplo utilizando o comando de limpeza do buffer do teclado

Atribuição de Valores às variáveis

- A atribuição de valores é utilizada para atribuir valores às variáveis, sendo representada pelo sinal = (igualdade).
- Pode ser feita na declaração (**int num = 2;**) ou através de um comando fora da declaração (**num = 2; ou num = x + y; ou num = 2 + 2; etc**).

Constantes

Em ocasiões específicas pode ser que seja requisitada uma variável que não altera o valor durante a execução do programa, como por exemplo o valor de **PI**. Nesses casos a utilização de constantes é indicada.

Sintaxe de declaração de constante
final tipo nome_constante;

Exemplos:
final double PI = 3.1415;
final int idadeMaxima = 50;

Operadores e Funções predefinidas

A linguagem JAVA possui alguns operadores e funções predefinidas destinadas a cálculos matemáticos e à manipulação de caracteres.

Operadores matemáticos:

Operador	Exemplo	Comentário
+	x + y	Soma o conteúdo de X e de Y.
-	x – y	Subtrai o conteúdo de Y do conteúdo de X
*	x * y	Multiplica o conteúdo de X pelo conteúdo de Y
/	x / y	Obtém o quociente da divisão de X por Y
%	x % y	Obtém o resto da divisão de X por Y
++	x ++	Aumenta o conteúdo de X em uma unidade (é o mesmo que x = x + 1)
- -	x - -	Diminui o conteúdo de X em uma unidade (é o mesmo que x = x - 1)

Operadores matemáticos de atribuição:

Operador	Exemplo	Comentário
+ =	x + = y	Equivale a X = X + Y.
- =	x – = y	Equivale a X = X - Y.
* =	x * = y	Equivale a X = X * Y.
/ =	x / = y	Equivale a X = X / Y.
% =	x % = y	Equivale a X = X % Y.

- Devem ser utilizados somente com valores numéricos;
- Operador **%(resto)** só pode ser usado com variáveis do tipo inteiro;
- Operador **/ (divisão)** quando utilizado com operandos inteiros retorna valores inteiro;
- Divisão por **zero** pode interromper a execução do programa.

Funções matemáticas predefinidas

A linguagem JAVA possui algumas funções matemáticas prontas para serem usadas. Todas elas podem ser observadas detalhadamente na documentação da "biblioteca" **Math**. Para se utilizar as funções dessa basta seguir a seguinte sintaxe:

Math."nomeFunção"(argumentos)

Algumas das funções disponíveis nessa biblioteca são:

Função	Finalidade
abs(i)	**Retorna** o valor absoluto de **i**.
ceil(d)	Arredonda para cima, para o próximo valor inteiro maior que **d**.
cos(d)	**Retorna** o cosseno de **d**.
floor(d)	Arredonda para baixo, para o próximo valor inteiro menor que **d**.
log(d)	Calcula o logaritmo neperiano log(**d**).
pow(d1, d2)	Retorna **d1** elevado a **d2**.
rand()	**Retorna** um inteiro positivo aleatório.
sin(d)	**Retorna** o seno de d.
sqrt(d)	**Retorna** a raiz quadrada de d.
tan(d)	**Retorna** a tangente de d.

Exemplos:

double resultado = Math.pow(numero, 2); // eleva a variável número ao quadrado e armazena o valor na variável resultado

Observação

Os métodos `sin`, `cos` e `tan` esperam receber argumentos no formato de radianos; para receberem argumentos em graus, siga o próximo exemplo.

```
dado = new Scanner(System.in);
x = dado.nextDouble();
y = Math.sin(Math.toRadians(x));
```

A linguagem JAVA possui muitas outras funções matemáticas que podem ser observadas detalhadamente na documentação da classe `Math`.

Exemplo

Teorema de Pitágoras: O programa exemplificado calcula a hipotenusa de um triângulo retângulo, dados os seus catetos, pelo Teorema de Pitágoras.

```
1  import java.util.Scanner;
2  public class ExemploTipoPrimitivo2 {
3    public static void main(String[] args) {
4        double cat1, cat2, somaCatetos, hipotenusa;
5        Scanner teclado = new Scanner(System.in);
6        System.out.println("Digite o valor do cateto 1:");
7        cat1 = teclado.nextDouble(); //Lendo double
8        System.out.println("Digite o valor do cateto 2:");
9        cat2 = teclado.nextDouble(); //Lendo double
10       somaCatetos = Math.pow(cat1,2) + Math.pow(cat2,2);
11       hipotenusa = Math.sqrt(somaCatetos);
12       System.out.printf("O triângulo com catetos %.2f e %.2f "
13               + "tem hipotenusa %.2f\n", cat1, cat2, hipotenusa);
14   }
15 }
```

Digite e compile o código acima. Observe a utilização das funções matemáticas predefinidas.

Exercícios

1. Crie um algoritmo que calcule a soma e a média de 3 números passados pelo usuário.

2. Implemente um algoritmo que leia um número inteiro e imprima seu antecessor e seu sucessor.

3. Implemente um algoritmo que receba 3 números reais de entrada. Calcule e mostre o resultado da multiplicação dos dois primeiros números dividido pelo terceiro número fornecido pelo usuário. Sabe-se que o denominador não pode ser zero, mas neste momento não se preocupe com as validações.

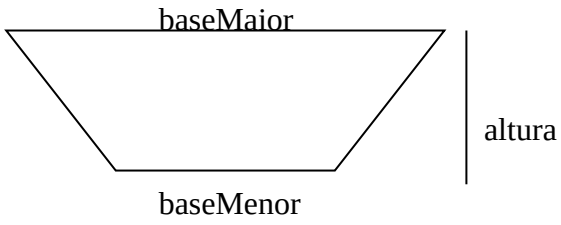
4. Faça um algoritmo que receba 3 notas e seus respectivos pesos, calcule e mostre a media ponderada dessas notas.

5. Faça um programa que receba o salário base de um funcionário, calcule e mostre o salário a receber, sabendo que esse funcionário possui uma gratificação de 10% sobre o salário base e paga 5% de imposto sobre o valor acumulado (salário base + gratificação).

6. Funcionários da IFVende tem como benefício a receber ao final de cada mês um salário fixo mais 4% de comissão sobre as vendas realizadas pelo mesmo. Assim, faça um algoritmo que receba o salário fixo e o valor de vendas realizadas por um funcionário, calcule e mostre o benefício a ser recebido pelo mesmo.

7. Faça um algoritmo que calcule a área e o perímetro de um retângulo recebendo de entrada os valores dos lados.

8. Zé Borba Gato é dono de um terreno na cidade de Sabará e deseja saber qual seria o preço médio de venda desse terreno. Conforme visto na planta do mesmo, nota-se que o lote possui um formato de trapézio (Figura abaixo).



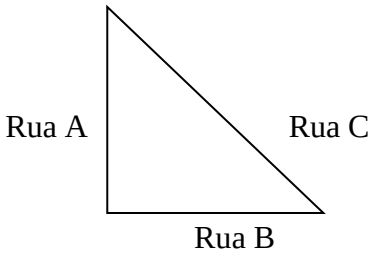
Assim sendo, dados os valores das medidas da base maior, base menor e da altura do terreno (em metros), e sabendo o valor médio (R$) pago por metro quadrado no local onde se encontra o lote, calcule e mostre a área total e o valor médio de venda (R$) que Zé Borba Gato pode pedir pelo terreno.

9. Implemente um algoritmo que receba o número de lados de um polígono convexo regular, calcule e mostre o número de diagonais desse polígono. Sabe-se que ND = N*(N – 3)/2, em que N representa o número de lados do polígono.

10. Uma pessoa depositou R$2000,00 em um fundo de investimento que rende 0.5% ao mês. Essa pessoa gostaria de saber qual o total acumulado após 2 anos. Faça um programa que forneça tais informações. (Obs. desconsidere correção monetária e utilize a fórmula de juros compostos).

11. João recebeu seu salário e precisa pagar 2 contas atrasadas. Em razão do atraso, ele deverá pagar multa de 2% sobre cada conta. Faça um programa que receba o salário do João e o valor de cada conta, calcule e mostre quanto restará de salário após o pagamento das duas contas.

12. Dona Maria das Couve é uma mulher muito preocupada com sua saúde e busca a prática de exercícios físicos constantes. Devido a crise financeira, Dona Maria está sem dinheiro para pagar academia, e teve como alternativa fazer caminhada diária ao redor do quarteirão de sua casa. O quarteirão da casa dela possui formato de um triângulo retângulo (figura abaixo), sabendo que ela deve caminhar um valor fixo de km por dia e dadas as medidas das ruas A e B (em metros) da figura abaixo, calcule e mostre quantas voltas Dona Maria precisa fazer no quarteirão para que ela atinja sua meta.



13. A copa do mundo de futebol da FIFA é uma competição internacional que iniciou em 1930, acontecendo desde então de 4 em 4 anos, exceto durante o período das guerras mundiais que impossibilitaram a realização 2 copas mundiais. Sabendo dessas informações, faça um algoritmo em JAVA que dado o ano atual calcule o número de copas já realizadas e exiba o resultado ao usuário.

14. Sabe-se que, para iluminar corretamente os cômodos de uma casa, para cada m², deve-se usar 18 W de potência. Faça um programa que receba as dimensões de um cômodo retangular (em metros), calcule e mostre a área de mesmo (em m²) e a potência de iluminação necessária para iluminar corretamente o cômodo.

15. Faça um algoritmo que calcule a área de uma circunferência, recebendo o valor do raio. Obs.: Defina **PI** como constante de valor 3.1416.

16. Faça um algoritmo que receba uma quantidade qualquer em minutos e converta em horas e minutos (utilize divisão inteira e resto da divisão inteira).

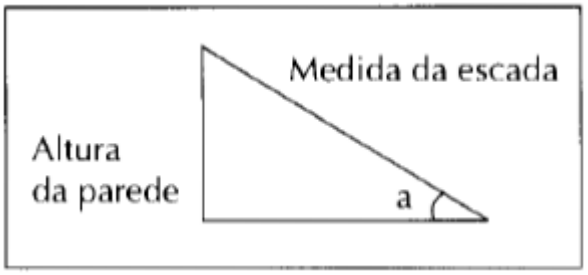
17. Faça um programa que receba de entrada um número real, encontre e mostre:
    a. A parte inteira desse número;
    b. A parte fracionária desse número.

18. Sabe-se que o valor pago por quilowatt de energia custa um quinto do valor do salário mínimo. Faça um algoritmo que receba o valor atual do salário mínimo e a quantidade de quilowatt consumida em uma residência, calcule e mostre:
    a. O valor pago por quilowatt;
    b. O valor a ser pago pelo consumo nessa residência;
    c. O valor a ser pago considerando um desconto de 15%.

19. Implemente m programa que receba um número positivo, calcule e mostre:
    - O número digitado elevado ao quadrado;
    - O número digitado elevado ao cubo;
    - A raiz quadrada do número;
    - A raiz cúbica do número.

20. Faça um algoritmo que receba o número de horas trabalhadas, o valor do salário mínimo e o número de horas extras trabalhadas. Calcule e mostre o salário a receber seguindo as seguintes regras:
    a) O valor a ser pago por hora trabalhada é 0.125 do salário mínimo;
    b) O valor a ser pago por hora extra vale 0.25 do salário mínimo;
    c) o salário bruto equivale ao número de horas trabalhadas vezes o valor pago por hora;
    d) a quantia a receber por horas extras equivale à horas extras realizadas multiplicado pelo valor pago por hora extra;
    e) o salário a receber equivale à soma do salário bruto mais a quantia a receber pelas horas extras.

21. Faça um programa que receba a medida do ângulo formado por uma escada apoiada e a altura da parede. Calcule e mostre a medida da escada para que a ponta da parede possa ser alcançada.



22. Faça um algoritmo que leia dois valores numéricos e armazene nas variáveis **A** e **B**, após isso, efetue a troca dos valores de forma que **A** passe a possuir o valor de **B** e **B** passe a possuir o valor de **A**. Imprima os valores após troca.

23. Num dado momento, 3 canais de TV tinham, em sua programação, novelas em seu horário nobre: canal A, novela A, canal B, novela B, canal C novela C. Numa pesquisa com 3000 pessoas, perguntou-se quais novela agradavam. A tabela a seguir mostra o resultado da pesquisa:

Novelas	**Nº de telespectadores**
A	1450
B	1150
C	900
A e B	350
A e C	400

B e C	300
A, B e C	100

Implemente um algoritmo que encontre o número de telespectadores que nenhuma das novelas os agradam. Receba as informações de preferências na entrada de dados e utilize as fórmulas de teoria dos conjuntos.

24. Considere uma equação do segundo grau na forma genérica ($ax^2 + bx + c$) e calcule o valor das raízes da mesma. Sabe-se que os coeficientes a, b, e c devem ser fornecidos pelo usuário. Utilize as equações abaixo como teste. Teste também para alguns valores aleatórios de coeficientes (exemplo a = 3, b = 2 e c =4) e veja que em alguns casos não se retornam as raízes de forma correta. Identifique o porquê desse problema e indique uma possível solução.

    Equações para teste:

    $1x^2 + 2x + 1 = 0$

    $1x^2 + 3x + 2 = 0$

    $1x^2 + 4x + 3 = 0$

    $1x^2 + 4x + 4 = 0$

    $2x^2 + 3x - 2 = 0$