UNIDADE I

Linguagem de Programação Orientada a Objetos

Prof. Me. Ricardo Veras

Características da Linguagem Java

- Simples e Dinâmica.
- É Orientada a Objetos - programam-se classes, e executam-se instâncias dessas classes.
- Baseia-se na Abstração, Encapsulamento e Herança, que permite:
  - A representatividade do mundo real.
  - A manipulação "segura" das informações.
  - O reaproveitamento de código.
- É "Portável" (independente de plataforma):
    - O Java trabalha com programas em "linguagem de máquina" (*bytecodes* - ou arquivos compilados) que podem ser lidos e interpretados por uma "Máquina Virtual" (a JVM - *Java Virtual Machine*).
    - Para cada Plataforma (Windows, IOS, Linux, Android, etc) existe uma JVM específica que pode ser instalada.

Características da Linguagem Java

- É uma linguagem fortemente tipada: nela as variáveis precisam ser declaradas antes de serem utilizadas, com um tipo "bem definido" que não poderá ser alterado.
- É *case-sensitive*: verifica as diferenças entre caracteres maiúsculos e minúsculos.
- Os Arquivos Fonte (com o código) possuem a extensão ".java“
- Os bytecodes possuem a extensão ".class“
    - A JVM contém um mecanismo de liberação automática de memória (*Garbage Collector* - "Coletor de Lixo") que limpa automaticamente a memória à medida que os objetos não estão mais sendo utilizados.

Características da Linguagem Java

Existem 3 Edições do Java:

- Java SE - Java *Standard Edition* – APIs essenciais para qualquer aplicação java: *Core* Java, Aplicações *Desktop*.

- Java EE - Java *Enterprise Edition* - APIs para o desenvolvimento de aplicações distribuídas: JSP (*Java Server Page* - Sistemas Web), EJB (*Enterprise Java Beans* - Sistemas em Redes Corporativas)

- Java ME - Java *Micro Edition* - APIs para o desenvolvimento de aplicações para portáteis (*mobiles*): PDAs (*Personal Digital Assistant*), Palms, Celulares, TVs digitais, etc.

- Nesta disciplina vamos trabalhar com a Edição SE da linguagem Java.

Programando na Linguagem Java

Instalando o Java:

- Pode-se instalar Somente o JRE, ou instalar o JDK, que possui o JRE.
- .JRE – Java *Runtime Environment* (ambiente de execução Java)
- .JDK – Java *Development Kit* (conjunto de ferramentas para desenvolvimento Java)
  - Obs.: basicamente, o JDK instala, além de todo JRE, todo o código fonte do Java, e mais o JavaDoc (offline).

Para baixar o JRE, basta entrar no site:

- https://www.java.com/pt-BR/download/
- ...e fazer o download.

Para baixar o JDK, basta entrar no site:

- https://www.oracle.com/java/technologies/javase-jdk15-downloads.html
- ...e fazer o download.

Programando na Linguagem Java

Instalando o Eclipse:

- Para instalar a versão 64 bits, basta entrar no na página web do link abaixo:
  - https://www.eclipse.org/downloads/
  - ...- clicar no botão "Download x86_64"
  - - na página seguinte, clicar no botão "Download"
  - ...e em seguida executar o arquivo de instalação.
- Para instalar a versão 32 bits (micros mais antigos), basta entrar na página web do link abaixo:
  - https://www.eclipse.org/downloads/packages/release/luna/sr2
  - ...- clicar no link "32-bit" (de "Eclipse IDE for Java Developers"). Este é um arquivo compactado que deve ser baixado em qualquer diretório, mas descompactado no diretório raiz de seu micro (c:\).

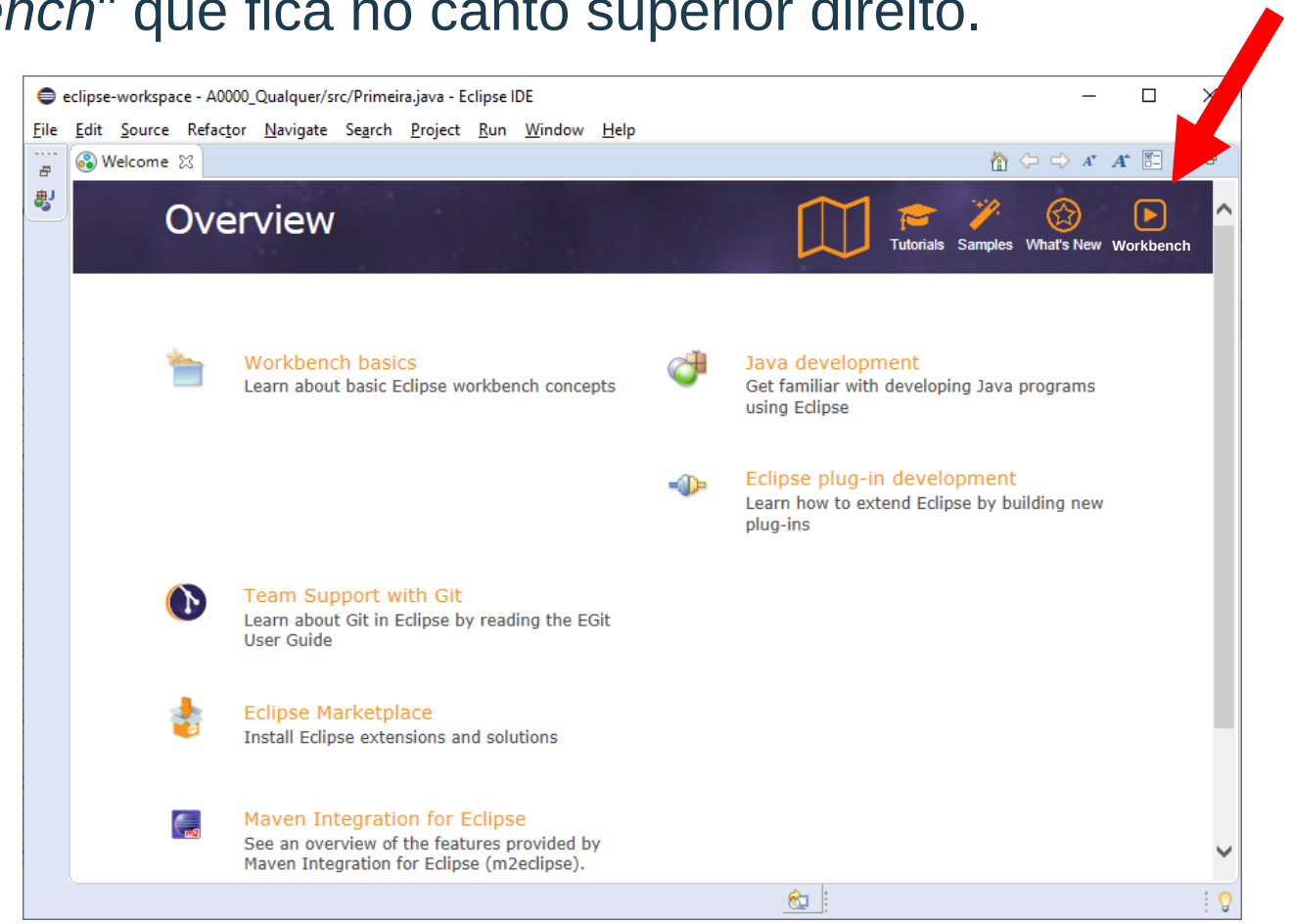
Fonte: autoria própria.

Programando na Linguagem Java

Abrindo o Eclipse:

- Ao iniciar (ao dar um duplo clique no arquivo eclipse.exe), o mesmo abre em sua tela de "*Welcome*".
- Deve-se clicar no ícone "*Workbench*" que fica no canto superior direito.

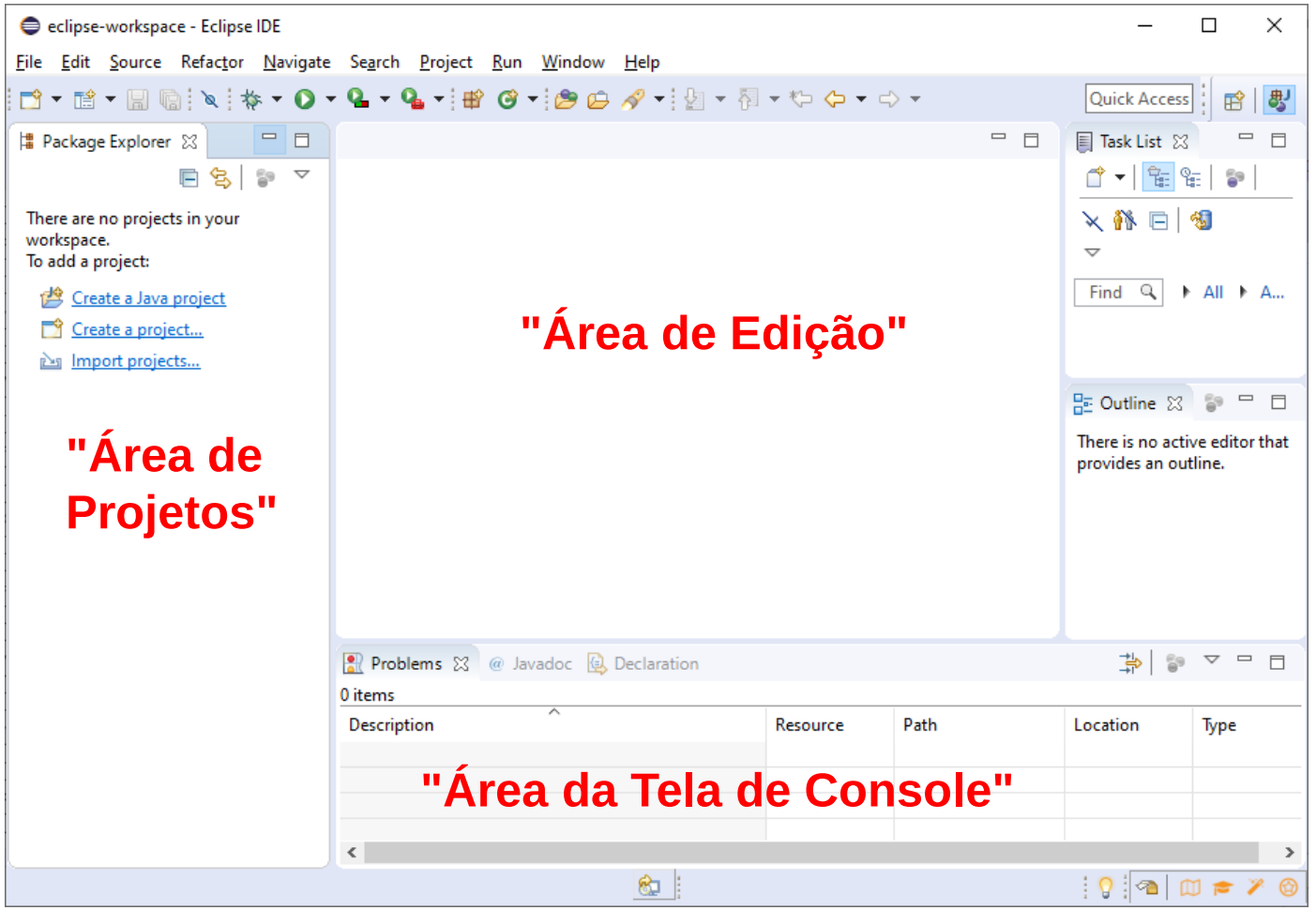
Fonte: autoria própria.



Tela do Workbench:

- Uma vez aberta, sempre que se iniciar o Eclipse, abrirá na tela da imagem abaixo.
- Áreas principais da tela do Workbench:

Fonte: autoria própria.



Programando na Linguagem Java

Programa "Olá Mundo":

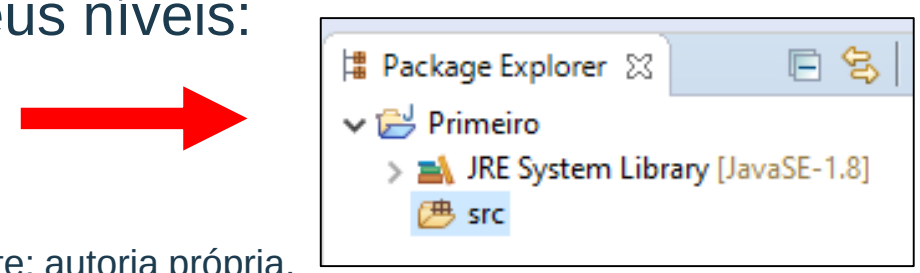
- Todo programa criado em Java, quando criado em alguma IDE (como o Eclipse, por exemplo) deve ser criado em um "Projeto"
- Assim, no Eclipse deve-se:

1º -

- selecionar os menus: "*File*" – "*New*" – "*Project*..."
- abrir (clicando em +) a pasta "Java" – selecionar "*Java Project*" – clicar "*Next* >"
- inserir um nome para o Projeto (como por exemplo "Primeiro") no campo "*Project name*:" e clicar em "*Finish*".

2º -

- No "*Package Explorer*" abrir o projeto e verificar os seus níveis:



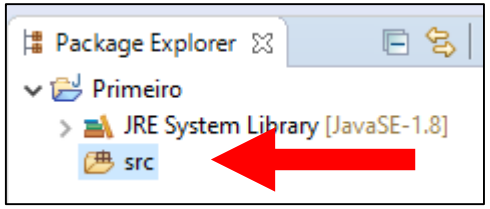
Fonte: autoria própria.

Programando na Linguagem Java

Programa "Olá Mundo“ (continuação):

3º

- com o botão direito do mouse sobre a pasta (*package*) "src“



Fonte: autoria própria.

- ...selecionar "*New*" – "*Class*“
  - na tela que se abre, no campo "*Name*:" inserir o nome da Classe (obs.: para nomear uma Classe, deve-se seguir as proibições para nomes de variáveis, mas por convenção sempre iniciar com letra Maiúscula – exemplo: Olamundo)
  - perceba que o arquivo criado ("Olamundo.java") sempre terá exatamente o mesmo nome da Classe.

Programando na Linguagem Java

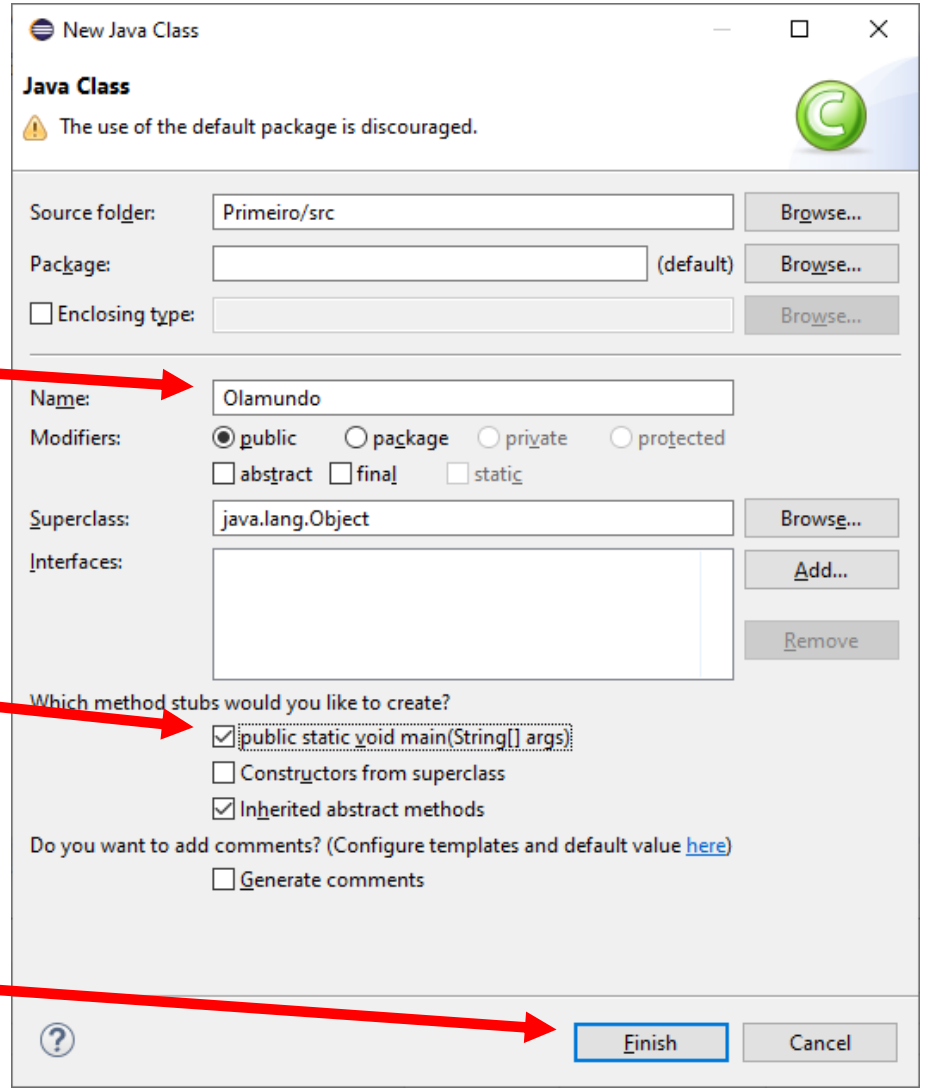
Programa "Olá Mundo“ (continuação):

3º (continuação)

A - inserir o nome da Classe:

B - selecionar a opção de inclusão do método main:

C – clicar no botão "Finish":



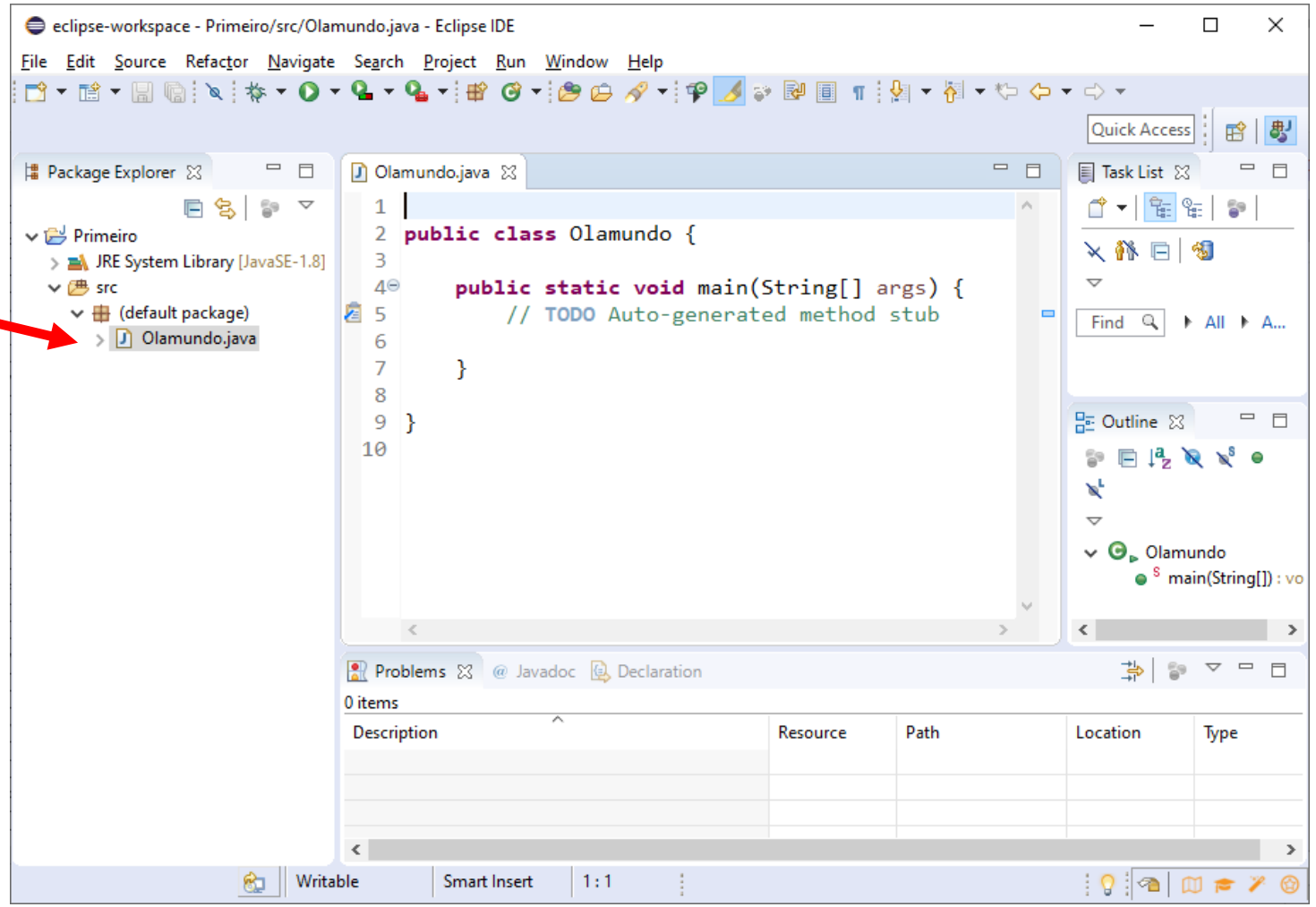
Fonte: autoria própria.

Programando na Linguagem Java

Programa "Olá Mundo“ (continuação):

4º - A tela de Edição de Programa: com o botão direito do mouse sobre a pasta (*package*) "src“

Nome do arquivo:



Fonte: autoria própria.

Programando na Linguagem Java

Programa "Olá Mundo“ (continuação):

5º - Na área de Edição, completar o programa para que fique com o seguinte código:

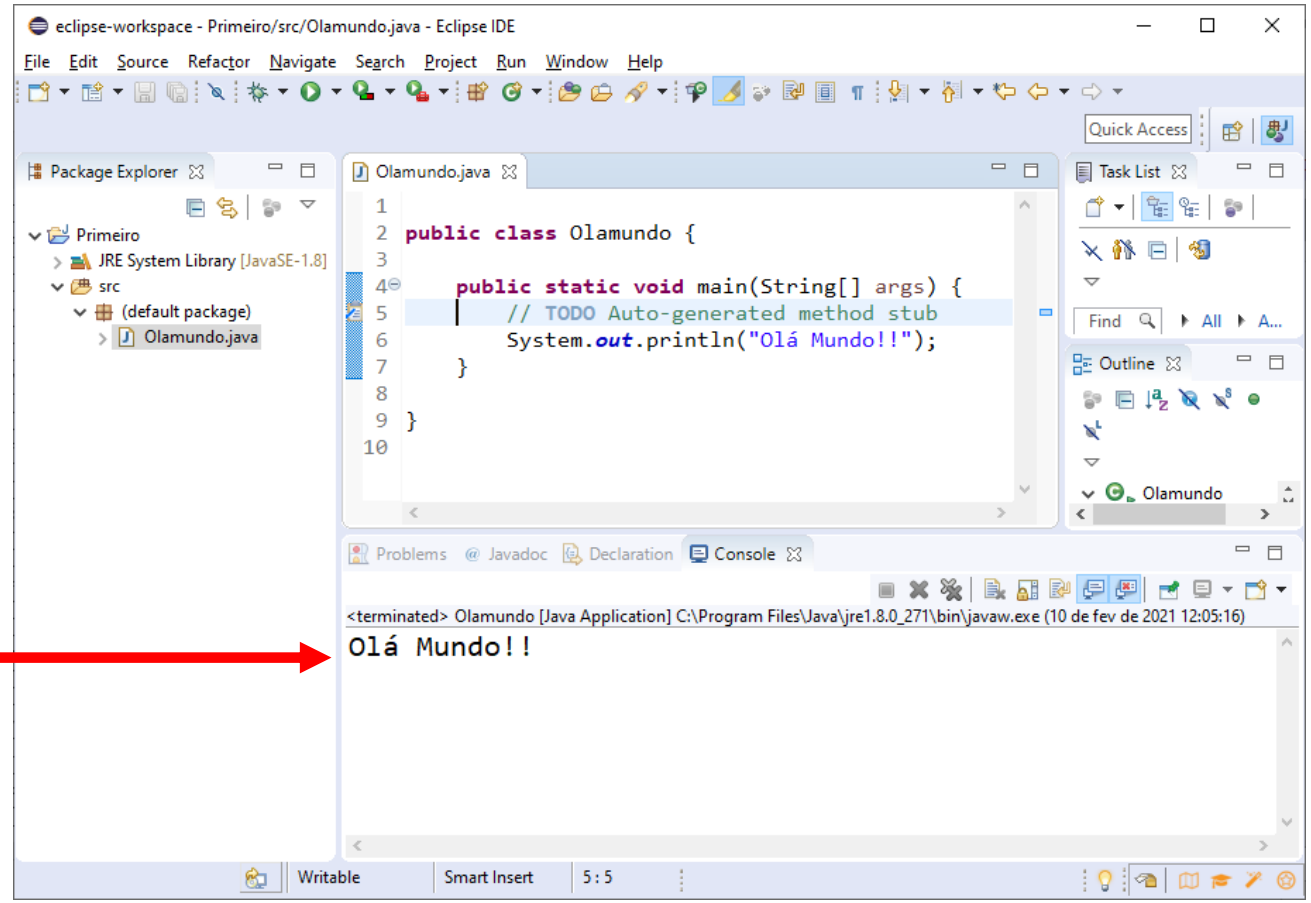
```
public class Olamundo {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.println("Olá Mundo!!");
    }

}
```

Programando na Linguagem Java

Programa "Olá Mundo“ (continuação):

6º - Para rodar o código criado, clicar com o botão direito do mouse sobre ele (sobre o código na própria área de "Edição de Código"), e selecionar "*Run As*" – "*Java Application*", de forma que na Área da Console aparecerá a frase desejada:

Fonte: autoria própria.



Interatividade

Sabendo que em programação com Java, cada Classe criada busca representar uma entidade do mundo real, o que significa que se estivéssemos criando um programa para uma empresa, e precisássemos criar uma Classe que representasse um funcionário, qual das opções abaixo melhor representará os nomes respectivamente: da classe, do arquivo do código fonte desta classe, do arquivo compilado desta classe.

a) funcionário, área.class, empresa.java
b) Funcionário.class, Funcionario.fonte, Funcionario.comp.
c) Funcionario, Funcionario.java, Funcionario.class.
d) funcionario, funcionario.font, funcionario.class.
e) Pessoas , Funcionario.fonte, Funcionario.java.

Resposta

Sabendo que em programação com Java, cada Classe criada busca representar uma entidade do mundo real, o que significa que se estivéssemos criando um programa para uma empresa, e precisássemos criar uma Classe que representasse um funcionário, qual das opções abaixo melhor representará os nomes respectivamente: da classe, do arquivo do código fonte desta classe, do arquivo compilado desta classe.

a) funcionário, área.class, empresa.java
b) Funcionário.class, Funcionario.fonte, Funcionario.comp.
c) Funcionario, Funcionario.java, Funcionario.class.
d) funcionario, funcionario.font, funcionario.class.
e) Pessoas , Funcionario.fonte, Funcionario.java.

Classes

- As Classes, na Programação Orientada a Objeto, são modelos (projetos) de um elemento "real" existente em memória (um objeto).
- Uma Classe possui dois elementos básicos: atributos (características - dados) e métodos (comportamentos - ações).
- Uma Classe é idealizada e codificada visando a representação do "mundo real", no momento da criação do Projeto (uma classe é um elemento de programação).
- Características das Classes:
  - Toda Classe possui um nome (iniciando com Maiúsculo);
  - Possuem características de "visibilidade" (*public*, *private*, *protected* e *default*);
    - A sintaxe de uma Classe é:

```
public class NomeDaClasse {
  //atributos (ou propriedades)
  //métodos (ou possíveis ações)
}
```

Exemplo de Classe

```
public class Pessoa {
    //atributos (ou propriedades)
    public String nome;
    public int idade;
    public double altura;
    //métodos
    public void cadastrarPessoa() {
        //...lógica de cadastro de uma Pessoa
    }
    public void calcularIMC() {
        //...lógica do cálculo do IMC...
    }
    //... demais métodos
}
```

Objetos

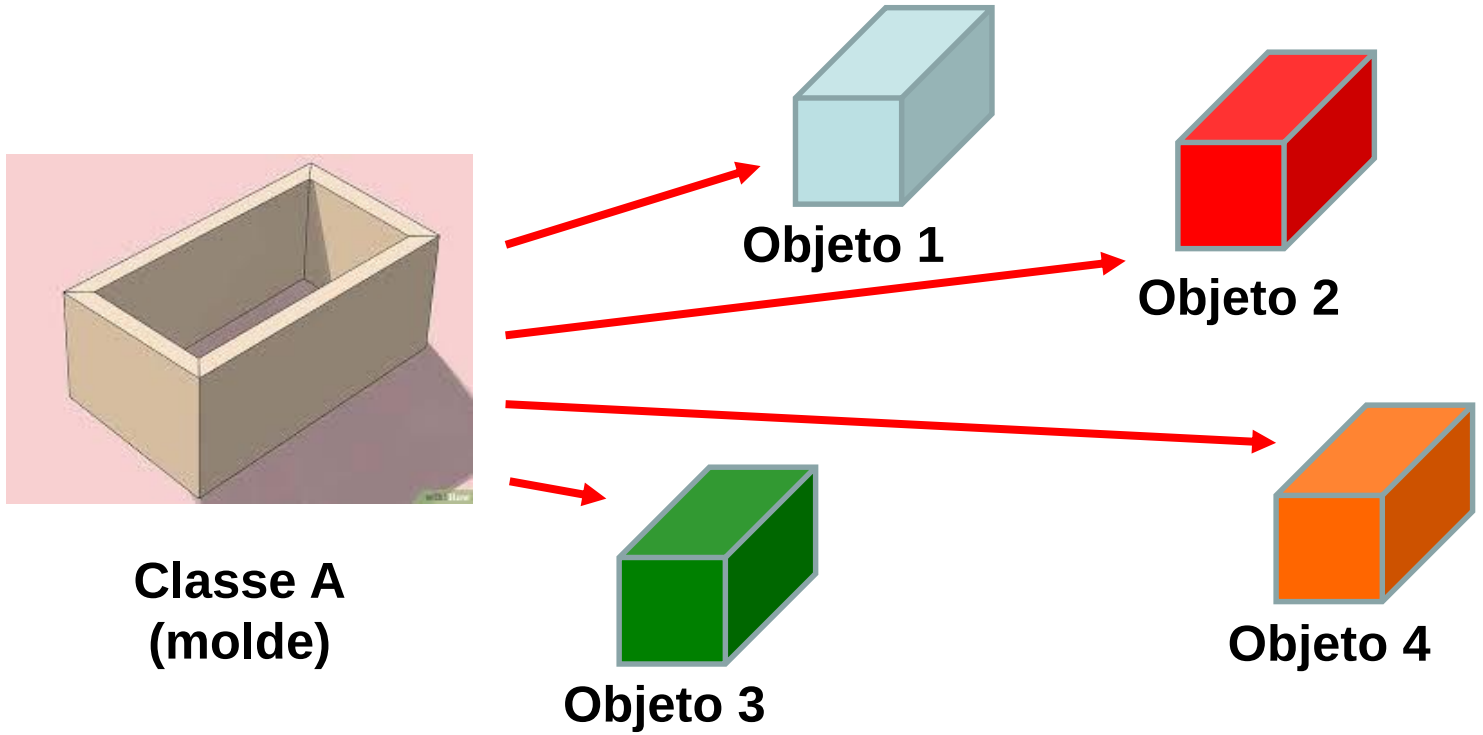
- Os objetos são elementos definidos pelas Classes.
- Eles são as Classes em "funcionamento", ou Classes instanciadas (geradas em memória).
- Neles podemos "popular" (inserir valores) seus atributos, e "invocar" (acionar ou chamar) seus métodos.
- Sintaxe de uma "instanciação" de uma Classe:

```
NomeClasse nomeObjeto = new NomeClasse();
```

- Quando a JVM executa uma linha de comando deste tipo, ela cria em memória um objeto a partir da Classe "NomeClasse", e cuja identificação será "nomeObjeto".

Representação Classe - Objeto

- A Classe é como um "molde" de objetos.
- A partir de uma Classe, podemos gerar em memória vários objetos que possuirão a mesma estrutura, que poderão realizar as mesmas ações, mas que serão independentes entre si e que, no geral, possuirão informações (dados) diferentes.

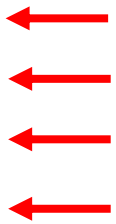


Fonte: autoria própria.

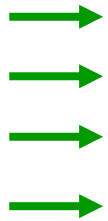
Representação Classe - Objeto

```
public class Pessoa {
        public String nome;
        public int idade;
        public void calcularIMC() {
                //... cálculo do IMC
        }
}
```

...código...



...memória...
(RAM)



```
objeto p1
nome = José
idade = 62
```

```
objeto p2
nome = Maria
idade = 57
```

```
objeto p3
nome = Ana
idade = 25
```

Representação Classe - Objeto

- Instanciando uma Classe:
  - Para se gerar os objetos do exemplo anterior, fazemos:

```
//..dentro de um método qualquer:

Pessoa p1 = new Pessoa();
p1.nome = "José";
p1.idade = 62;

Pessoa p2 = new Pessoa();
p2.nome = "Maria";
p2.idade = 57;

Pessoa p3 = new Pessoa();
p3.nome = "Ana";
p3.idade = 25;

//... Chamando o método de cálculo:
p3.calcularIMC; // chamado a partir do objeto p3

//... continuação do código ...
```

Atributos

- Os atributos são os dados de um objeto (valores que o caracterizam), e também podem ser definidos como variáveis ou campos. Esses valores definem o estado de um objeto (podem sofrer alterações).
- Os atributos devem possuir: nome, tipo e valor (obs.: o nome de um atributo deve, por convenção iniciar com minúsculo).
- Sintaxe de declaração de um atributo:

```
modificadores tipo nomeAtributo;

    ou ainda...

modificadores tipo nomeAtributo = valor;
```

Trabalhando com Atributos

- Mesmo que se atribua um valor inicial a um atributo, se o mesmo não for uma constante, seu valor poderá ser alterado a qualquer momento.
- Para se atribuir um valor a um atributo fazemos:

```
objeto.atributo = valor;
```

- Obs.: ao se trabalhar com valores de um atributo, deve-se indicar qual o objeto a que ele pertence.

```
Pessoa p1 = new Pessoa();
Pessoa p2 = new Pessoa();
Pessoa p3 = new Pessoa();
// alterando-se o valor do atributo “nome”
p1.nome = "José";
p2.nome = "Maria";
p3.nome = "Ana";
```

Trabalhando com Atributos

- Também quando queremos utilizar o valor de um atributo, seja para mostrar seu valor ou para utilizar em um processamento (como um cálculo, por exemplo), partimos da mesma ideia, identificando o objeto a qual o atributo pertence:

```
Pessoa p1 = new Pessoa();
// atribuindo valores
p1.peso = 70.45;
p1.altura = 1.75;
// utilizando-se de seus valores
double imc = p1.peso / (p1.altura * p1.altura);
String txt = "Características da pessoa:\n";
txt += "Peso: " + p1.peso + "\n";
txt += "Altura: " + p1.altura + "\n";
txt += "...terá IMC = " + imc;
System.out.println(txt);
```

Métodos

- Os métodos são as ações ou procedimentos com os quais um objeto pode interagir e se comunicar com outros objetos. A execução dessas ações, assim como a resposta dessas ações, se dá através de "mensagens", tendo como objetivo o envio de uma solicitação ao objeto para que seja efetuada a rotina (ação) desejada, e para que seja devolvida uma resposta, quando necessário.
- Em geral, um método é representado por um verbo (ex: criarNovoElemento(), gerarLista(), acelerarPersonagem(), etc.).
- Sintaxe de declaração de um método:

```
modificadores tipoDeRetorno nomeDoMetodo (parâmetros) {
    //codificação – a lógica do método
}
```

Métodos

Características dos Métodos:

- Na sintaxe de declaração de um método temos:

```
modificadores tipoRetorno nomeMetodo (parâmetros) {
    //codificação com a lógica do método
}
```

- O tipo de retorno indica qual o tipo do dado que o método "devolverá" (retornará) a quem o chamou (neste caso, a palavra reservada "void" define um método que não terá retorno).
    - Os parâmetros são as informações que enviamos ao método para que ele as processe (obs.: quando o método não possui parâmetros, o mesmo possui ao menos os parênteses).
    - Sintaxe dos "(parâmetros)":

      ```
      (tipo1 param1, tipo2 param2, tipo3 param3, etc...)
      ```

    - Ao invocarmos, ou chamarmos, um método, deve-se enviar os valores de todos os seus parâmetros, seguindo a ordem definida na declaração.

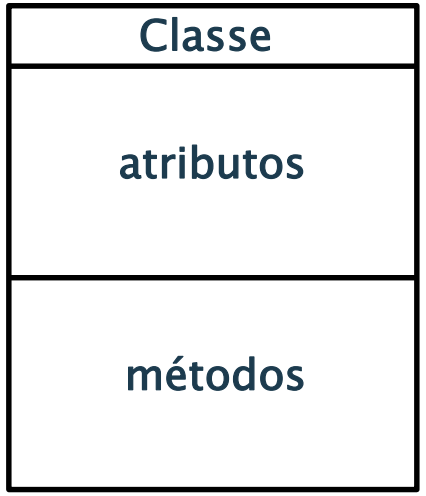
Exemplo de Métodos

- A Classe Calculadora:

```
public class Calculadora {

	public double res = 0;

	//---------------------------

	public void multiplicar (double v1, double v2) {
		res = v1 * v2;
	}
	public void subtrair (double v1, double v2) {
		res = v1 - v2;
	}
	public double retornarSoma (double v1, double v2) {
		return v1 + v2;
	}
	public void verResultado () {
		System.out.print(res);
	}
}
```

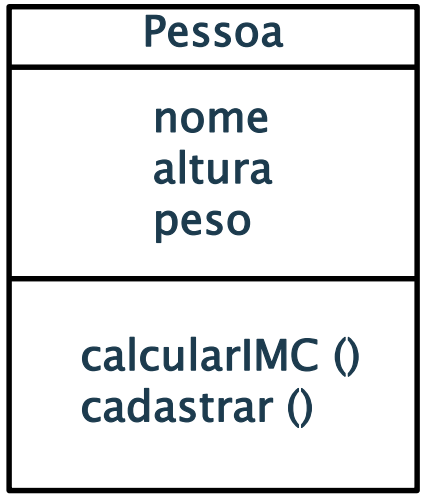
Representação UML de uma Classe:

- Obs.: UML – "*Unified Modeling Language*" ou Linguagem de Modelagem Unificada
- Na UML uma Classe é representada da seguinte forma:



Fonte: autoria própria.

Exemplo:



Fonte: autoria própria.

Interatividade

Num programa, instanciou-se a classe “Aviao” com o objeto “c2”. Sabendo que esta Classe contém o atributo “public double velocidade;”, determine qual das opções abaixo representa o comando que insere corretamente um valor neste atributo daquele objeto.

a) inserir 80.5 em “velocidade”;
b) set 80.5 to velocidade;
c) velocidade = 80,5;
d) c2.velocidade = 80.5;
e) velocidade.do.objeto = 1.000,20;

Resposta

Num programa, instanciou-se a classe “Aviao” com o objeto “c2”. Sabendo que esta Classe contém o atributo “public double velocidade;”, determine qual das opções abaixo representa o comando que insere corretamente um valor neste atributo daquele objeto.

a) inserir 80.5 em “velocidade”;
b) set 80.5 to velocidade;
c) velocidade = 80,5;
d) c2.velocidade = 80.5;
e) velocidade.do.objeto = 1.000,20;

O Método main

- O método "main" é o método responsável por acionar inicialmente um programa Java, de forma que este método precisa existir em apenas 1(uma) Classe do Sistema.
- Uma vez acionado o método "main", nas linhas de código do sistema criado, podemos acionar uma infinidade de outros métodos, pertencentes a uma infinidade de outras Classes.
- Sua sintaxe (do método “main”), dentro de uma Classe, será:

```
public class ClasseA {
    public static void main (String[] args) {
        // linhas de código do método – sua lógica
    }
}
```

Tipo de Variáveis

- A linguagem Java é "fortemente tipada", e depende que se defina o Tipo de uma variável sempre antes de utilizá-la.

- Uma vez definido no programa o tipo de uma variável, o mesmo não pode mais ser alterado.

- O Tipo de uma variável não só define qual tipo de informação será guardada nela, como também define o tamanho do espaço em memória que será reservado para que o compilador guarde o valor daquela variável.

Tipo de Variáveis (tipos primitivos)

- Tipos numéricos Inteiros:
  - **byte**: números de -128 a 127
  - **short**: números de -32.768 a 32.767 (...mil)
  - **int**: números de -2.147.483.648 a 2.147.483.647 (...bilhão)
  - **long**: números de -9.223.372.036.854.775.808 a 9.223.372.036.854.775.807 (...quintilhão)
- Tipos numéricos Reais:
  - **float**: ótima precisão para números reais com até 7 casas decimais.
  - **double**: ótima precisão para números reais com até 15 casas decimais.
    - Tipo Lógico:
      - **boolean**: guarda um valor lógico: *true* ou *false*
    - Tipo Alfanumérico:
      - **char**: guarda um valor de caractere: ex.: 't', ou 'x', ou '%' ...

Tipo de Variáveis

- O tipo String: este tipo, em Java, não é considerado um "tipo primitivo", mas sim uma Classe (percebam que o nome do tipo inicia com letra maiúscula).
- Este tipo pode guardar desde letras, palavras, e até frases inteiras.
- Ao contrário do que ocorre em algumas linguagens, Strings em Java não devem ser tratadas como sequências (ou matrizes) de caracteres.
- Para se definir um valor (literal) a uma variável do tipo String, deve-se utilizar "aspas-duplas".

```
String txt = "Um texto qualquer...";
```

- Obs.: percebam a diferença entre o valor de uma "String" (que deve estar entre aspas duplas) e o valor de um `'char'` (que deve estar entre aspas simples)

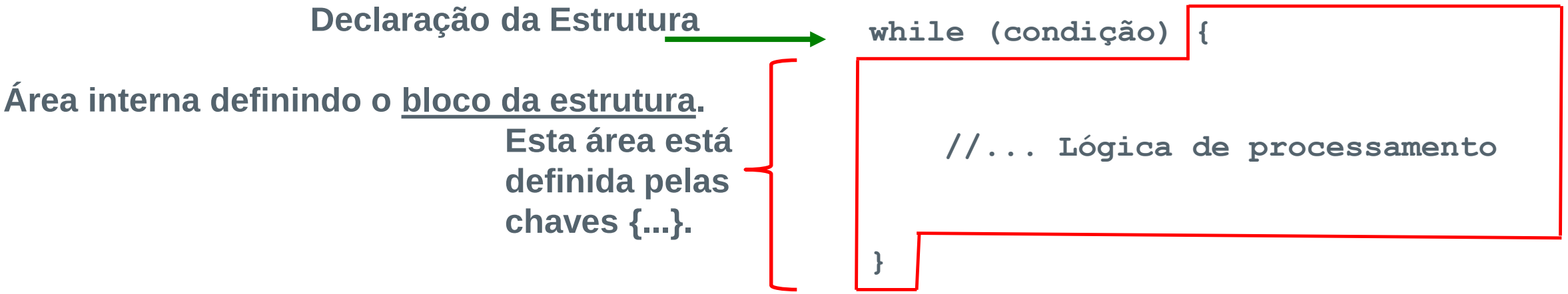
```
String texto = "Outro texto qualquer...";
char c = '@';
```

Estruturas de Controle

Em Java, toda estrutura é definida pela sua "declaração", e mais os seus delimitadores, sendo:

- o Parêntesis (...) determinando um conteúdo de controle de processamento;
- as Chaves {...} definindo a extensão do bloco da estrutura.

- Exemplo: a estrutura while:



Estruturas Condicionais

- Em Java, existem duas estruturas condicionais:
  - if – else
  - switch – case

- A Estrutura "if":

```
if (cond1) {
   //...
} else if (cond2) {
   //...
} else if (condN) {
   //...
} else {
   //...
}
```

Lembrando que desta estrutura, apenas um dos blocos será executado: aquele que primeiro tiver sua condição verdadeira, de modo que se nenhuma condição for verdadeira, executa-se o bloco definido pelo “else” (que, apesar de não ser um bloco obrigatório, é sempre colocado por último).

Estruturas Condicionais

- Exemplo de um programa com a estrutura "if":

```
int a = 150;
if (a < 10) {
      System.out.println("o Valor de a é menor que dez");
} else if (a < 100) {
      System.out.println("o Valor de a é menor que cem");
} else if (a < 1000) {
      System.out.println("o Valor de a é menor que mil");
} else {
      System.out.println("o Valor de a é muito grande");
}
```

O Programa acima, quando executado, imprimirá na console o seguinte texto:

o Valor de a é menor que mil

Estruturas Condicionais

A Estrutura "switch-case":

- A estrutura condicional "switch-case" compara o valor da variável (determinada no parênteses da declaração) com o valor descrito em cada um dos blocos "case", de modo que se este valor for igual ao da variável, aquele bloco é executado. Sua sintaxe é:

```
switch (variavel) {
   case valor1:
      //...
      break;
   ...
   case valorN:
      //...
      break;
   default:
      //...
      break;
}
```

Assim como na estrutura “if”, nesta estrutura apenas um dos blocos será executado: aquele que primeiro tiver seu valor igual ao da variável, de modo que se nenhum valor for igual ao da variável, executa-se o bloco definido pelo “default” (que, apesar de não ser um bloco obrigatório, é sempre colocado por último).

Obs.: os blocos “case” (ou “default”) iniciam-se logo após o “:” e terminam sempre com “break;”

Estruturas Condicionais

Exemplo da Estrutura "switch-case":

```
int a = 1;
switch (a) {
    case 1:
        System.out.println("o Valor de a é igual a um");
        break;
    case 2:
        System.out.println("o Valor de a é igual a dois");
        break;
    case 3:
        System.out.println("o Valor de a é igual a três");
        break;
    default:
        System.out.println("Valor fora da tabela");
        break;
}
```

Interatividade

Num programa, precisamos gerar uma lógica que, de acordo com o valor dado à variável “ctrl” (que é uma variável de controle do tipo double), o código deverá verificar se este valor é maior que o valor de uma única outra variável, de forma que caso ela seja maior, um cálculo deverá ser realizado, e caso não seja maior, nada deve mudar na execução do programa. Neste caso, qual das estruturas abaixo, é a mais adequada para realizar este tipo de comparação.

a) Estutura “if” simples (sem o bloco do “else”).
b) Estutura “switch-case” (com apenas um “case”).
c) Estutura “switch-case” (com vários “cases”).
d) Estutura “if” composta (com o bloco do “else”).
e) Estutura “if” encadeada (com mais de um “else if”).

Resposta

Num programa, precisamos gerar uma lógica que, de acordo com o valor dado à variável “ctrl” (que é uma variável de controle do tipo double), o código deverá verificar se este valor é maior que o valor de uma única outra variável, de forma que caso ela seja maior, um cálculo deverá ser realizado, e caso não seja maior, nada deve mudar na execução do programa. Neste caso, qual das estruturas abaixo, é a mais adequada para realizar este tipo de comparação.

a) Estutura “if” simples (sem o bloco do “else”).
b) Estutura “switch-case” (com apenas um “case”).
c) Estutura “switch-case” (com vários “cases”).
d) Estutura “if” composta (com o bloco do “else”).
e) Estutura “if” encadeada (com mais de um “else if”).

Estruturas de Repetição

Em Java, existem três estruturas de repetição:

- for
- while
- do – while

- Essas estruturas permitem que um mesmo bloco de código seja executado repetidas vezes. A quantidade de iterações (execução completa do bloco da estrutura) dependerá dos controles definidos na declaração da estrutura.

Estruturas de Repetição

Sintaxe da Estrutura "for":

```
for (var = valorInicial; condição de repetição; variação da variável de controle) {
   ...
}
```

- A estrutura “for” deve ser utilizada sempre que se é possível saber qual a quantidade de iterações que a estrutura de repetição executará. Como a linguagem Java é “fortemente tipada”, o tipo da variável de controle deve ser definido antes da (ou na) declaração da estrutura (e sempre como tipo inteiro).

```
int var;
for (var = valorInicial; condição; variação) {
   //... bloco a ser executado repetidas vezes
}
```

...ou ainda:

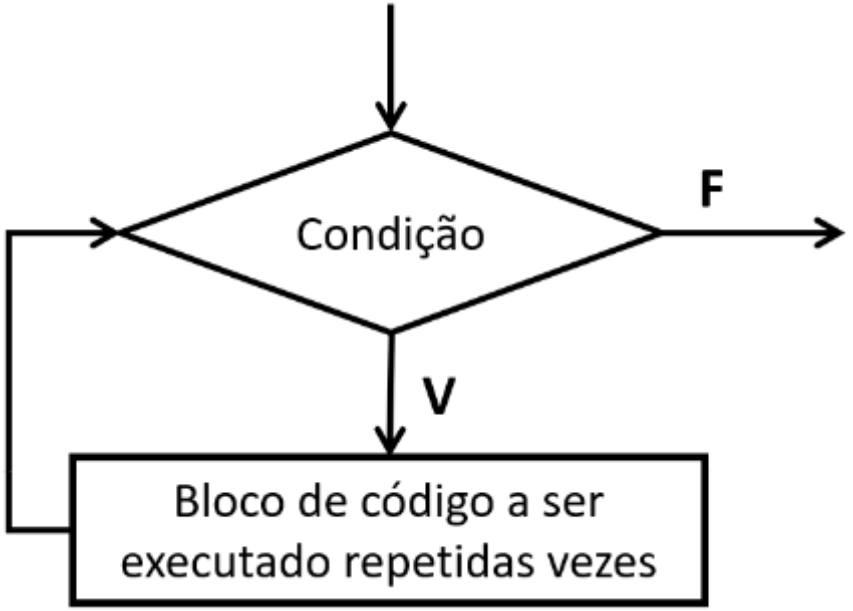
```
for (int var = valorInicial; condição; variação) {
   //... bloco a ser executado repetidas vezes
}
```

Estruturas de Repetição

Sintaxe da Estrutura "while":

```
while (condição) {
     ...
}
```

- A estrutura “while” pode ser utilizada quando não se sabe a quantidade exata de iterações que a estrutura de repetição executará. Nesta estrutura, a condição é sempre verificada no início da iteração.



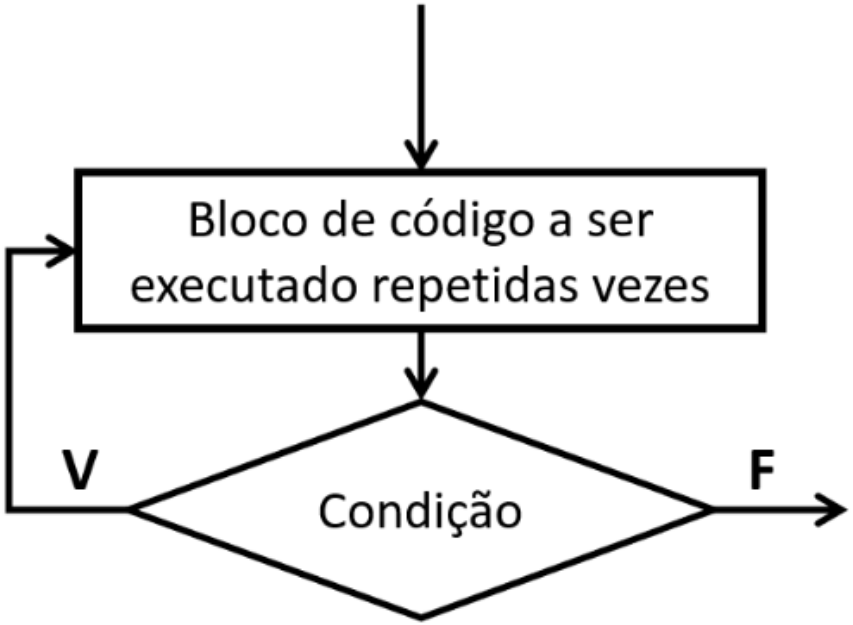
Fonte: autoria própria.

Estruturas de Repetição

Sintaxe da Estrutura “do – while":

```
do {
    ...
} while (condição);
```

- Nesta estrutura, a condição é sempre verificada no final da iteração.



Fonte: autoria própria.

Exemplo com as Estruturas de Repetição

- (todos os exemplos abaixo gerados com cada uma das estruturas de repetição, imprimem na tela da console os valores inteiros de 1 a 10)

```
// ---- Estrutura for -------
for (int w = 1; w <= 10; w++) {
        System.out.print(w + " ");
}
// ---- Estrutura while -----
System.out.println(" ");
int x = 1;
while (x <= 10) {
        System.out.print(x + " ");
        x++;
}
// ---- Estrutura do-while -----
System.out.println(" ");
int y = 1;
do {
        System.out.print(y + " ");
        y++;
} while (y <= 10);
```

Packages (pacotes)

- Num projeto Java, utilizamos pacotes para organizarmos as classes em categorias (o equivalente a pastas, quando queremos organizar nossos arquivos no computador).
- Fisicamente (caso se procure no *Explorer* do Windows), um pacote é identificado como um ou mais diretórios que contêm internamente uma ou mais classes Java.
- Por exemplo,

```
package model;
```

- é o diretório "model" que fica na raiz do projeto Java.

```
package com.mysql.jdbc;
```

- é o diretório com/mysql/jdbc (três diretórios, um dentro do outro) que fica na raiz do projeto Java.

Packages (pacotes)

- Quando uma classe está localizada em um pacote, internamente, na classe, essa situação precisa estar definida (em código), identificando o pacote em que ela se localiza, por meio da palavra reservada "package".

```
package nome.do.pacote;
//significa que esta Classe pertence ao pacote "nome.do.pacote"
```

Packages (pacotes)

- Quando vamos utilizar, ao longo da classe, uma outra classe que está em outro pacote, somos obrigados a importar aquela classe, utilizando a palavra reservada "import", e indicando o caminho até ela:

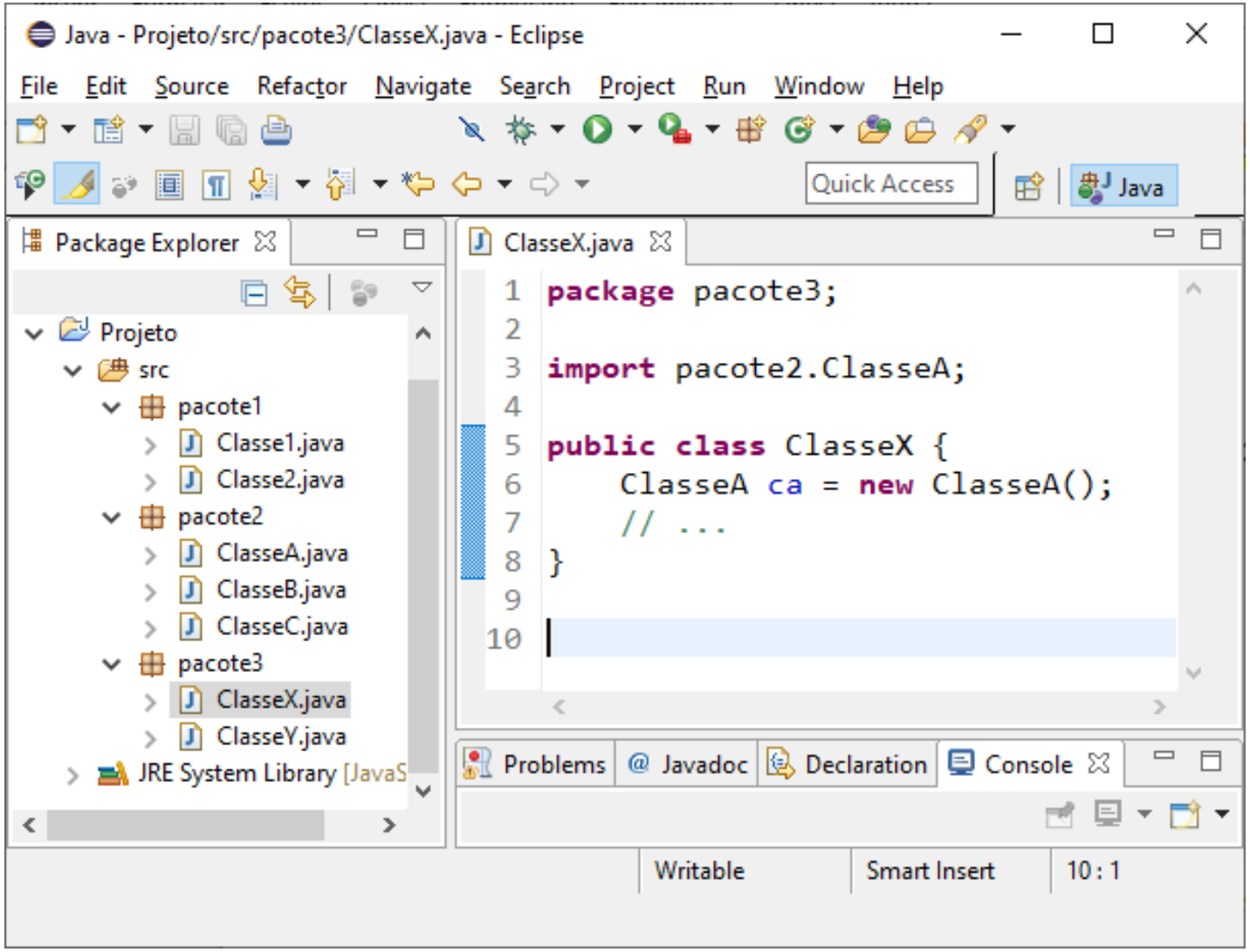
```
import nome.do.pacote.Classe;
```

- Caso se queira importar todas as classes de um pacote, ao invés de importar classe por classe, colocamos o caractere * (asterisco).

```
import nome.do.pacote.*;
```

Packages (pacotes)

Verificando a imagem abaixo:



Fonte: autoria própria.

Interatividade

Em uma, classe tem-se o seguinte código:

```
public static void main(String[] args) {
	int x = 2;
	for (int a = 1; a < 6; a++) {
		if (a != 4) System.out.print((a*x) + " ");
	}
}
```

Qual das seguintes alternativas mostra o texto escrito na tela da console após se executar o código acima?

a) 1 2 3 4 5

b) 1*2 2*2 3*2 4*2 5*2

c) 2 4 6 8 10

d) 1 2 3 5 6

e) 2 4 6 10

Resposta

Em uma, classe tem-se o seguinte código:

```
public static void main(String[] args) {
        int x = 2;
        for (int a = 1; a < 6; a++) {
                if (a != 4) System.out.print((a*x) + " ");
        }
}
```

Qual das seguintes alternativas mostra o texto escrito na tela da console após se executar o código acima?

a) 1 2 3 4 5

b) 1*2 2*2 3*2 4*2 5*2

c) 2 4 6 8 10

d) 1 2 3 5 6

e) 2 4 6 10

ATÉ A PRÓXIMA!