



# DOMAIN DRIVEN DESIGN

Prof. Rafael Desiderio

03 – IDE ECLIPSE E TIPOS DE DADOS

# | Integrated Development Enviroment - IDE

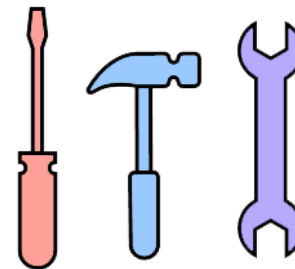
- **Integrated Development Environment (IDE)** ou **Ambiente Integrado de Desenvolvimento**, é um programa de computador que reúne características e ferramentas de **apoio ao desenvolvimento de software** com o objetivo de agilizar este processo;
- Geralmente os IDEs **facilitam** a técnica de RAD (Rapid Application Development, ou "**Desenvolvimento Rápido de Aplicativos**"), que visa a maior **produtividade** dos desenvolvedores;
- Exemplos de **IDEs** para desenvolvimento na plataforma **Java**:
  - Eclipse
  - NetBeans
  - IntelliJ



# | Integrated Development Enviroment - IDE

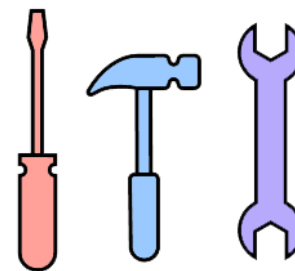
As características e ferramentas mais comuns encontradas nas **IDEs** são:

- **Editor** - edita o código-fonte do programa escrito nas linguagens suportadas pela IDE;
- **Compilador** – compila o código-fonte do programa;
- **Debugger (Depurador)** – auxilia no processo de encontrar e corrigir defeitos no código-fonte do programa;
- **Modelagem** – criação do modelo de classes, objetos, interfaces, associações e interações dos artefatos envolvidos no software com o objetivo de solucionar as necessidades do software final;
- **Geração de código** – geração de código a partir de templates de código comumente utilizados para solucionar problemas rotineiros;



# | Integrated Development Enviroment - IDE

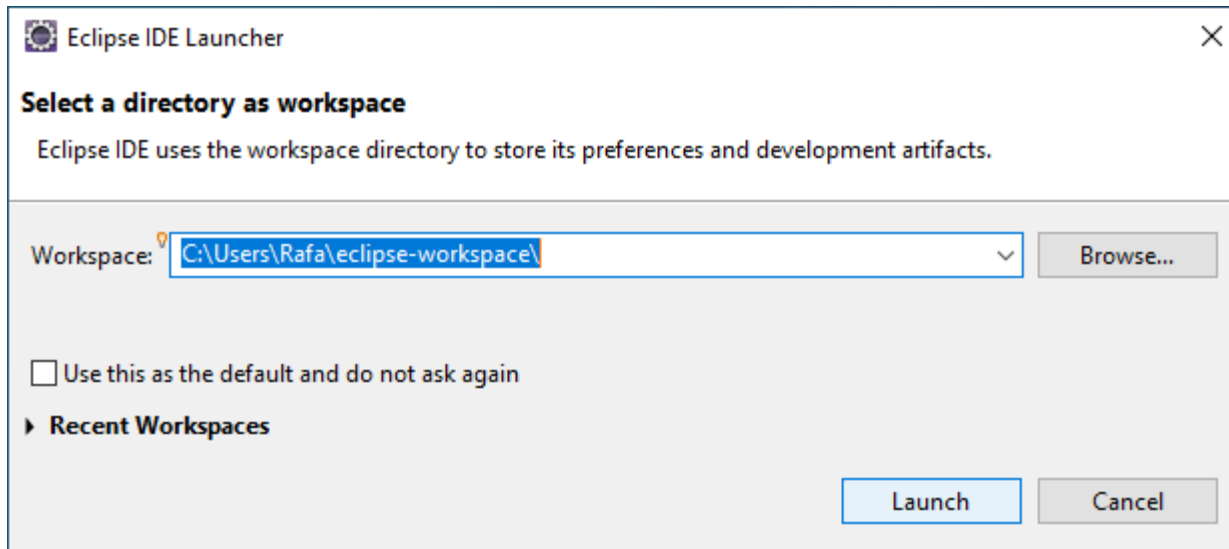
- **Deploy (Distribuição)** - auxilia no processo de criação do instalador do software ou outra forma de distribuição;
- **Testes automatizados** – realiza testes no software de forma automatizada, com base em scripts ou programas de testes previamente especificados, gerando um relatório, assim auxiliando na análise do impacto das alterações no código fonte;
- **Refactoring (Refatoração)** – consiste na melhoria constante do código-fonte do software, seja na construção de código mais otimizado, mais limpo e/ou com melhor entendimento pelos envolvidos no desenvolvimento de software;



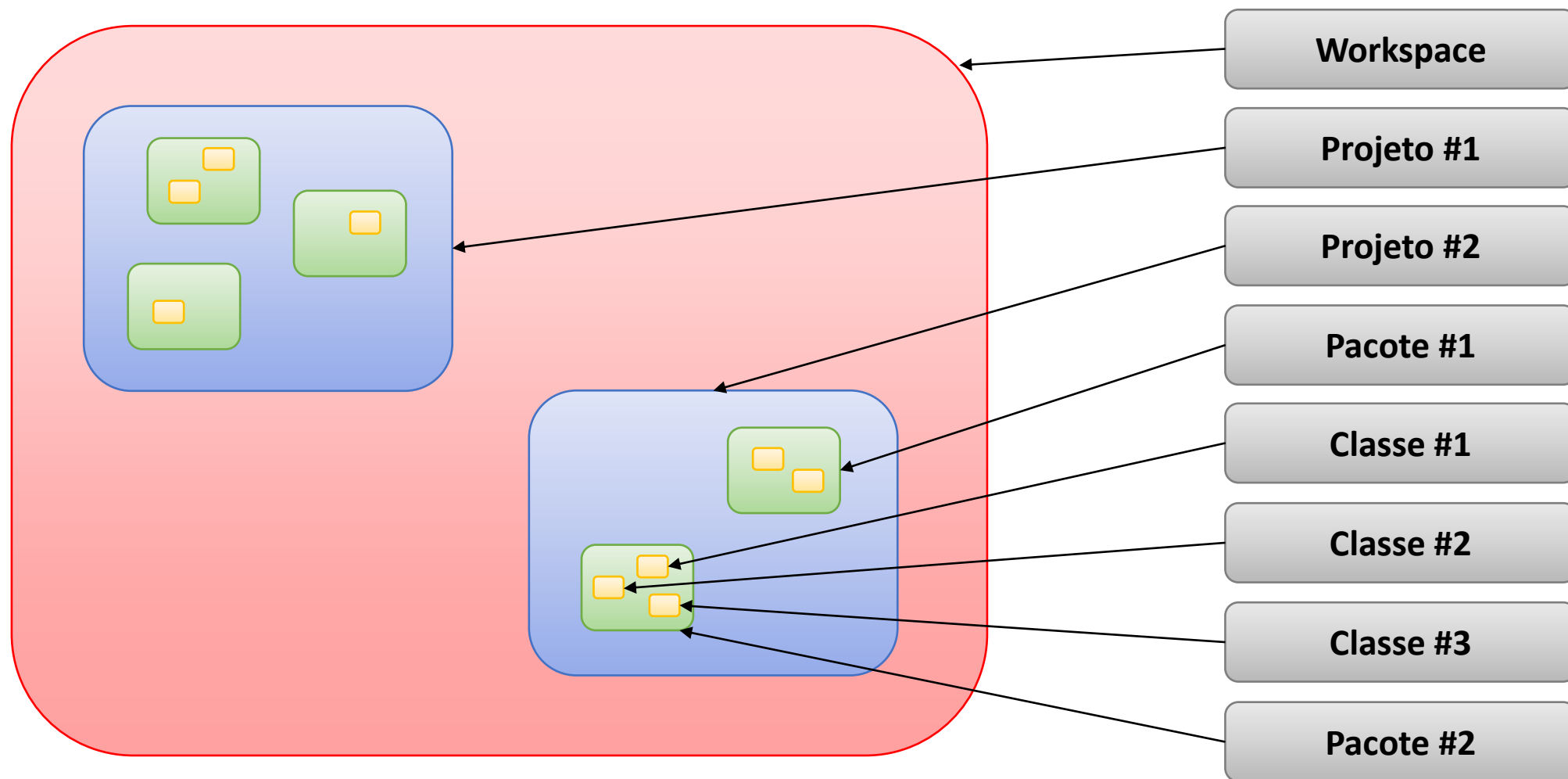
- **Eclipse** é um **IDE** desenvolvido em **Java**, seguindo o modelo **open source** de desenvolvimento de software;
- O projeto **Eclipse** foi iniciado na **IBM** que desenvolveu a primeira versão do produto e doou-o como **software livre para a comunidade**;
- O **Eclipse** possui como características marcantes o uso da **SWT** e não do **Swing** como biblioteca gráfica, a forte orientação ao desenvolvimento baseado em plug-ins e o amplo suporte ao desenvolvedor com centenas de plug-ins que procuram atender as diferentes necessidades de diferentes programadores;
- Com o uso de **plugins**, pode ser usado não só para desenvolver em **Java**, mas também em **C/C++**, **PHP**, **ColdFusion** e até mesmo **Python**;

# | Eclipse - Workspace

- Ao abrir o **eclipse** é preciso **selecionar o diretório** onde serão gravados os arquivos/diretórios criados na ferramenta;
- Esse diretório é chamado de **workspace**;
- Um **workspace** pode conter vários projetos, cada projeto pode ter vários pacotes e classes;
- Nos laboratórios da **FIAP** utilize sempre **um diretório** dentro do local **D:/**, pois caso a máquina desligue/reinicie os arquivos serão **preservados**;

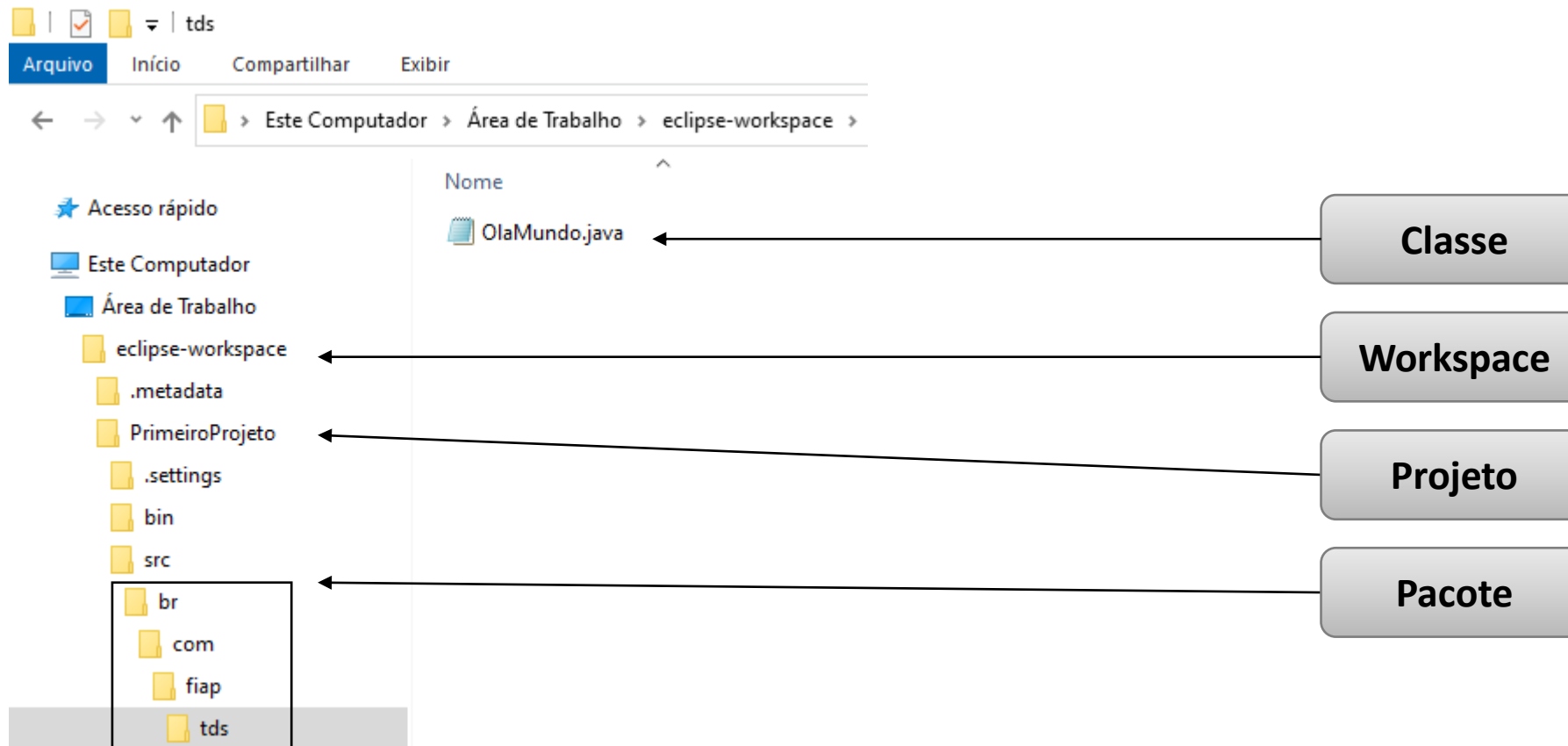


# | Eclipse - Workspace



# | Workspace – Visão física

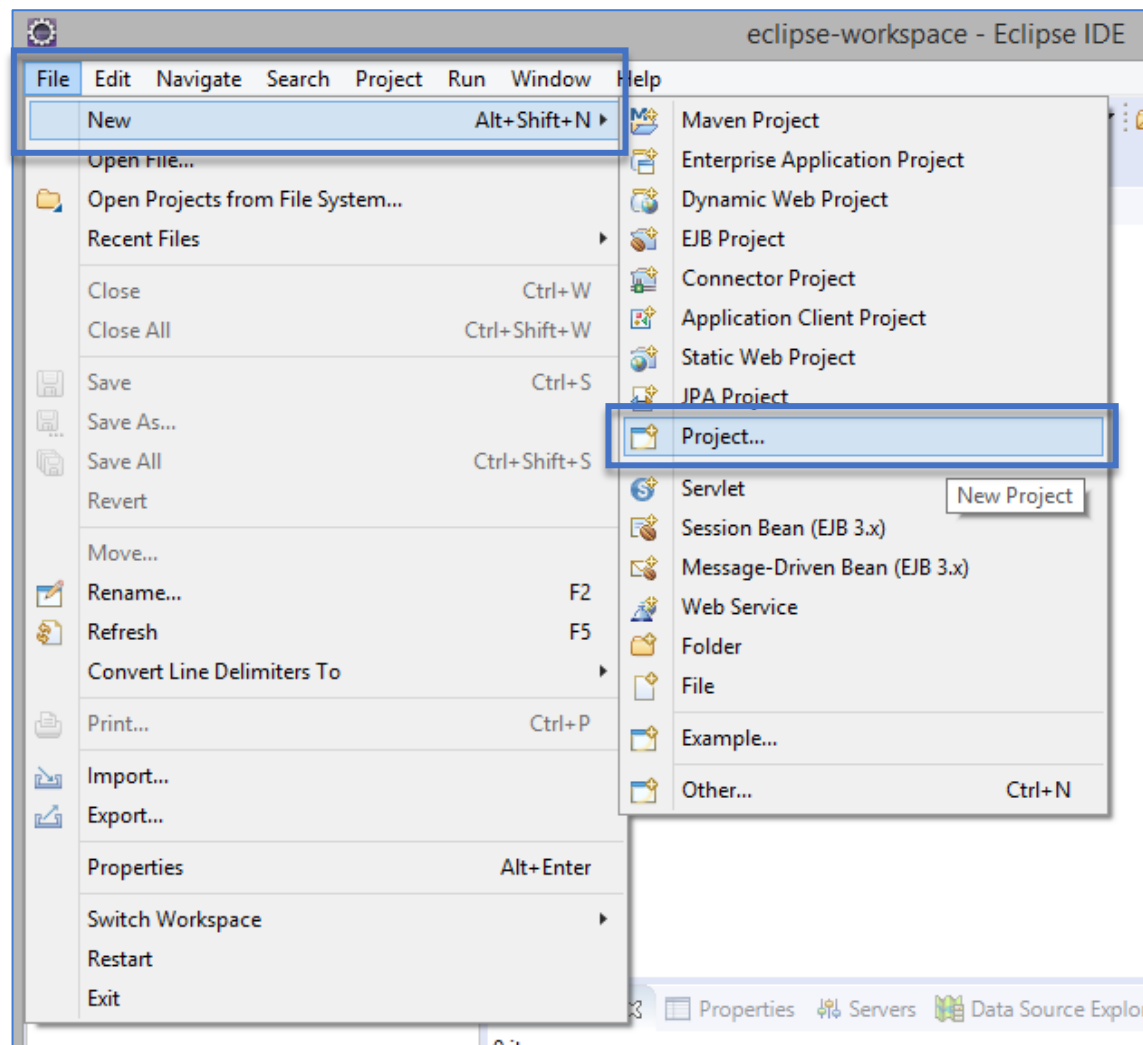
- No **Windows Explorer** é possível ver os diretórios e arquivos que foram gerados dentro do **eclipse**:





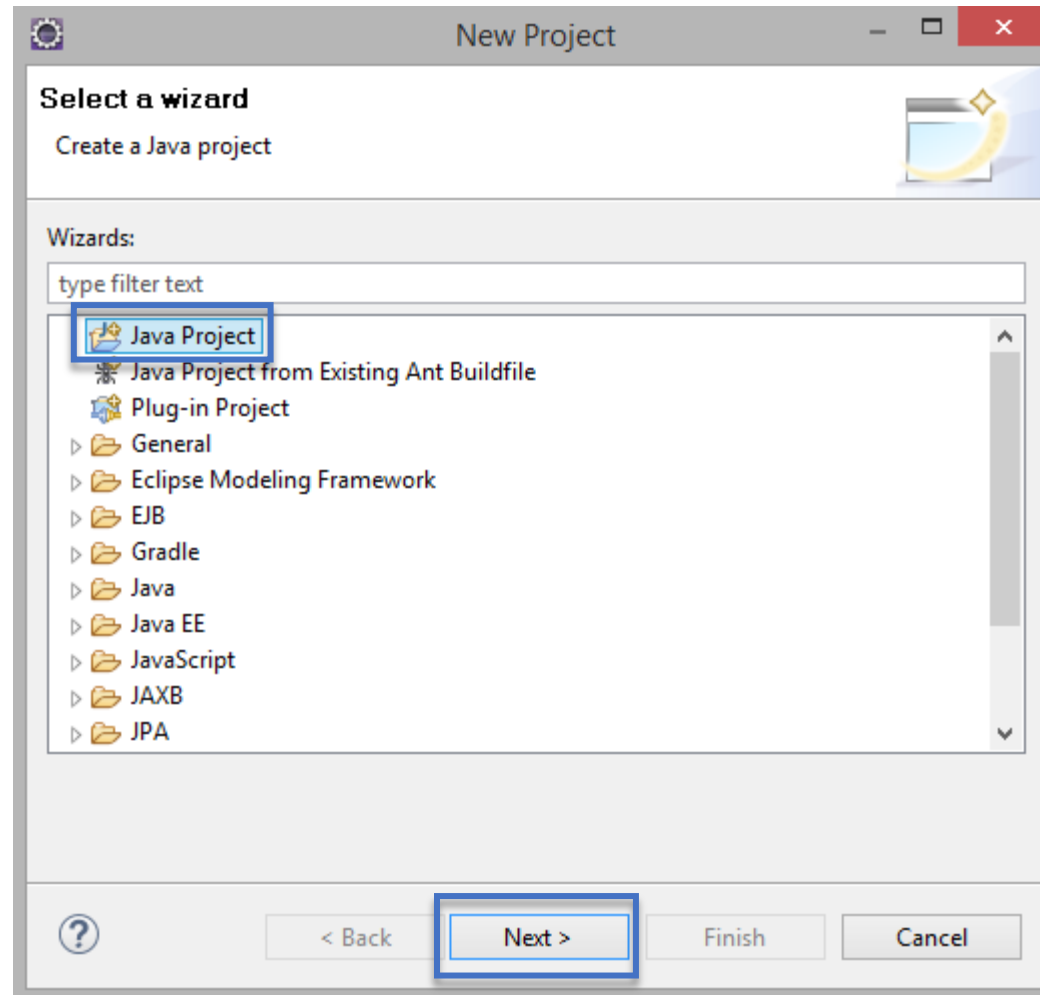
# Eclipse – Criar um Projeto

- Para criar um **projeto** utilize o menu **File > New > Project**;



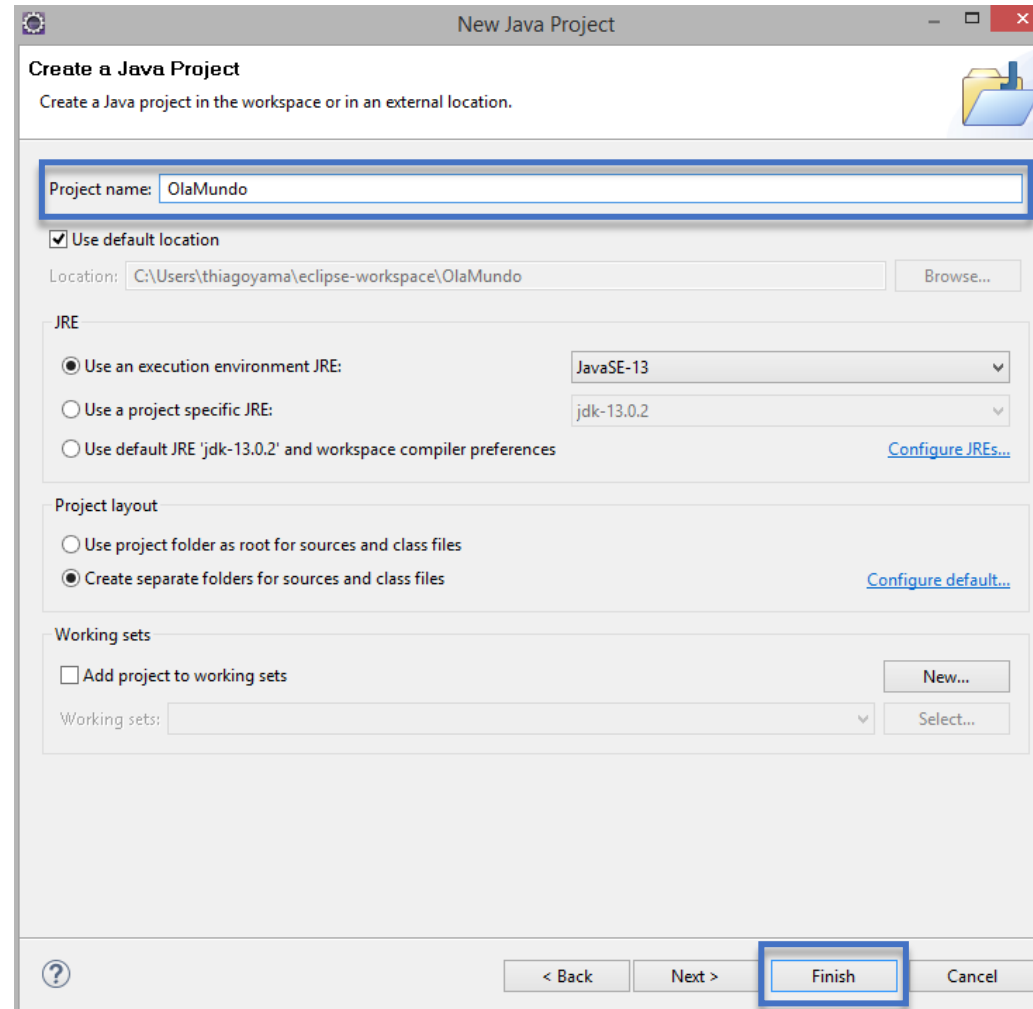
# Eclipse – Criar um Projeto

- Crie um projeto **JSE**, escolhendo a opção **Java > Java Project**;

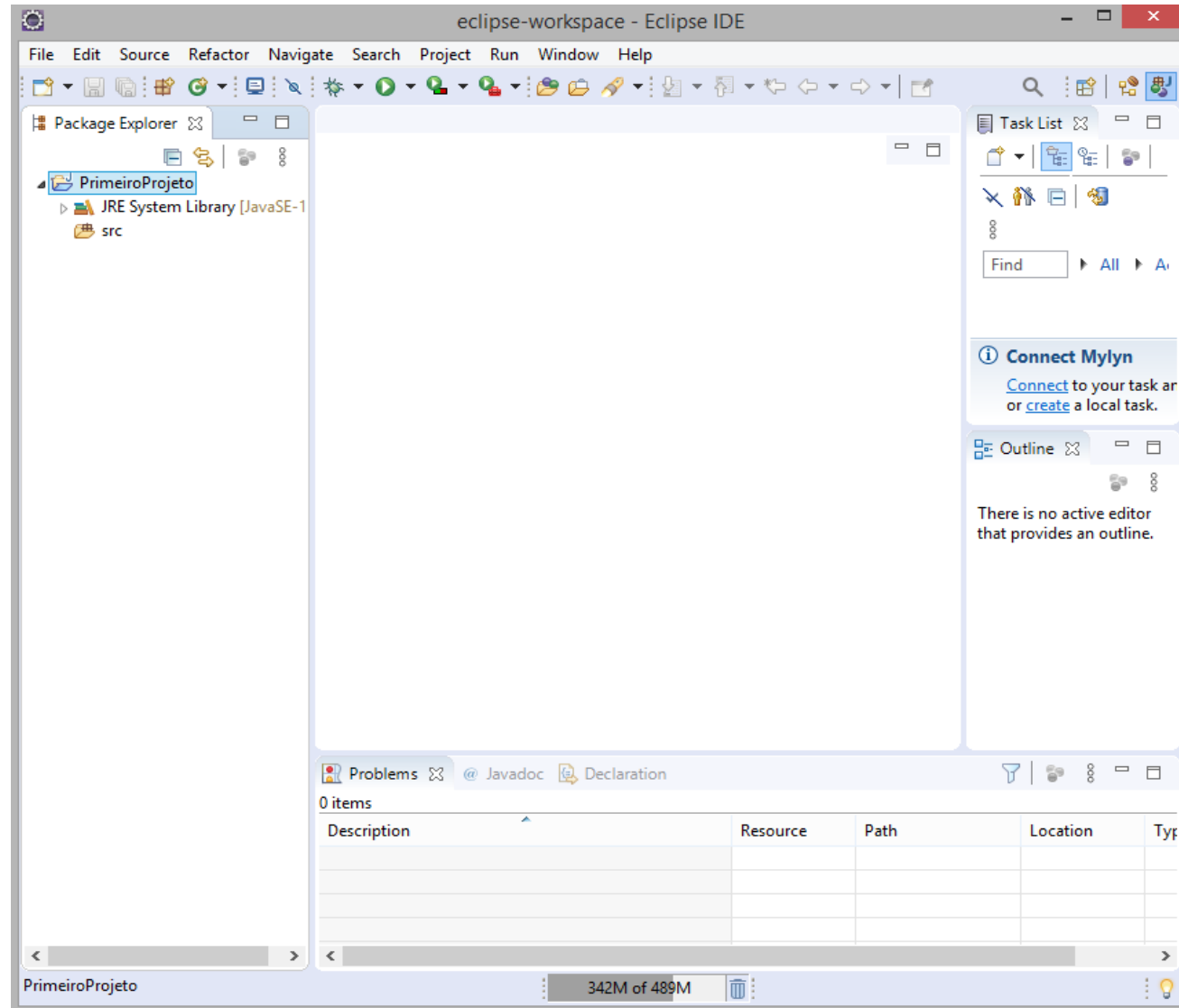


# | Eclipse – Criar um Projeto

- Dê um **nome ao projeto** e finalize com o botão **Finish**;

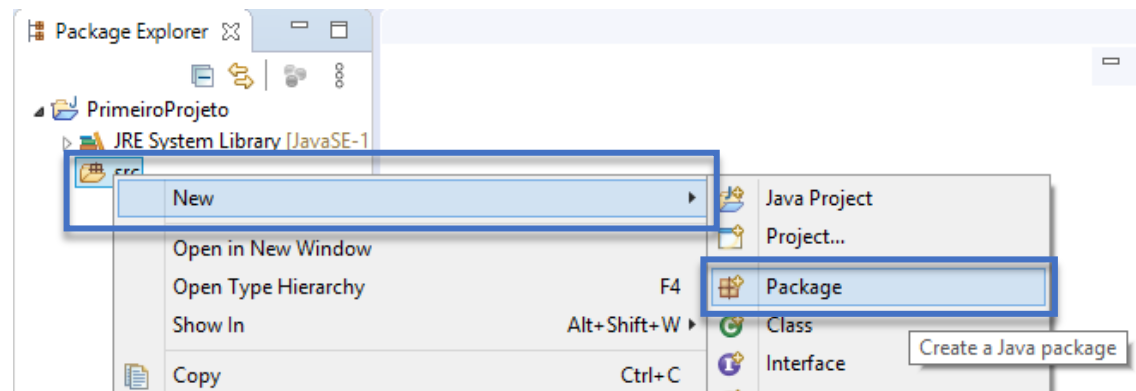


# | Eclipse – Ambiente de Desenvolvimento



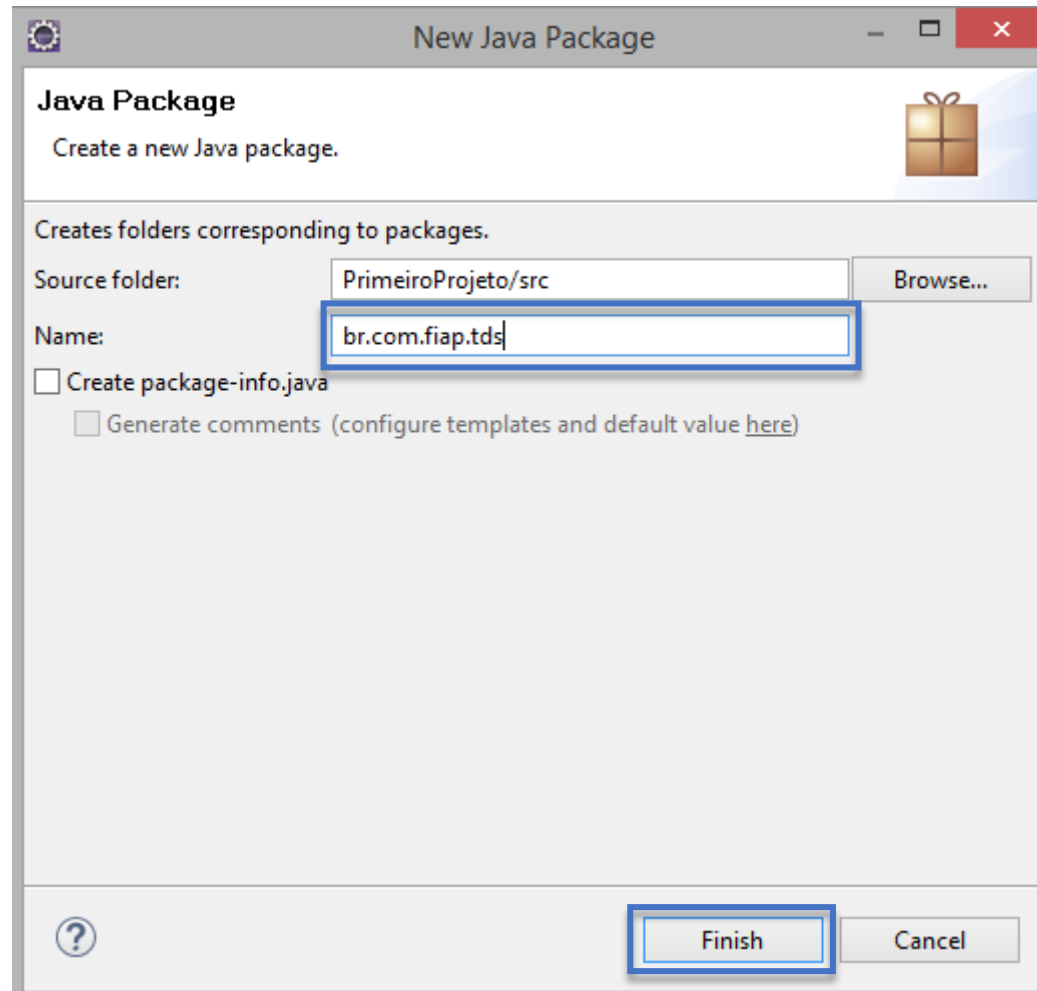
# | Eclipse – Criar um Pacote

- Crie um **pacote** para organizar as classes;
- A boa prática para **nomenclatura** de pacotes é utilizar a URL ao contrário, como por exemplo: **br.com.fiap.projeto**;
- Para isso, clique com o botão direito do mouse na pasta **src** e escolha a opção **New > Package**:



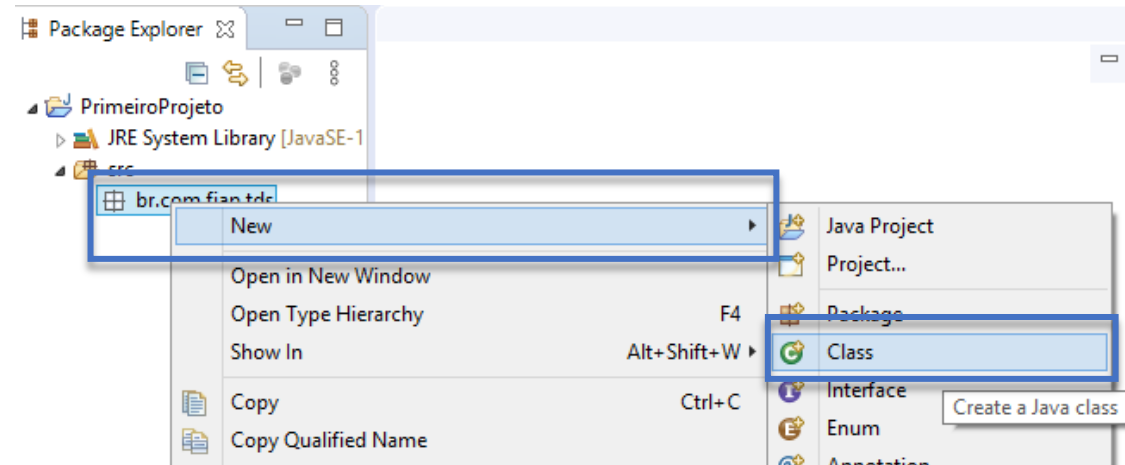
# | Eclipse – Criar um Pacote

- Configure o **nome do pacote** e **finalize** o processo:



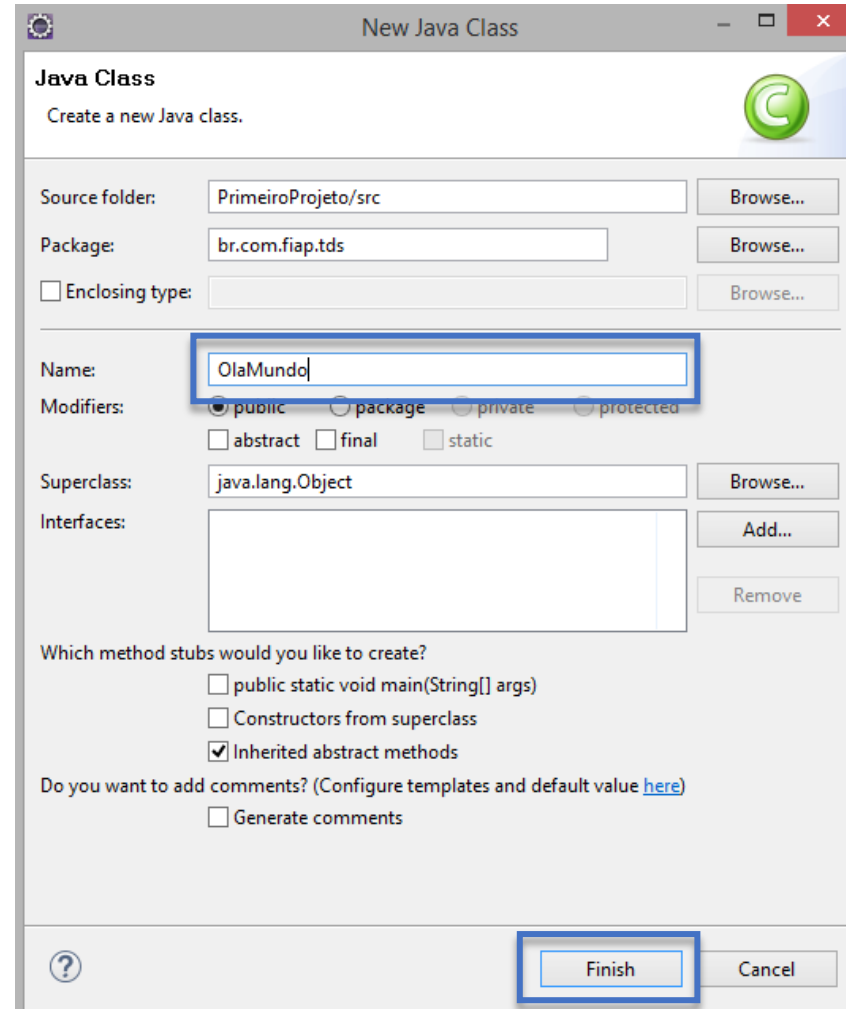
# | Eclipse – Criar uma Classe

- Para criar uma **classe**, clique com o **botão direito do mouse** no pacote e escolha a opção **New > Class**;



# | Eclipse – Criar uma Classe

- Configure o **nome** e **finalize** o processo:





# | Implemente a Classe

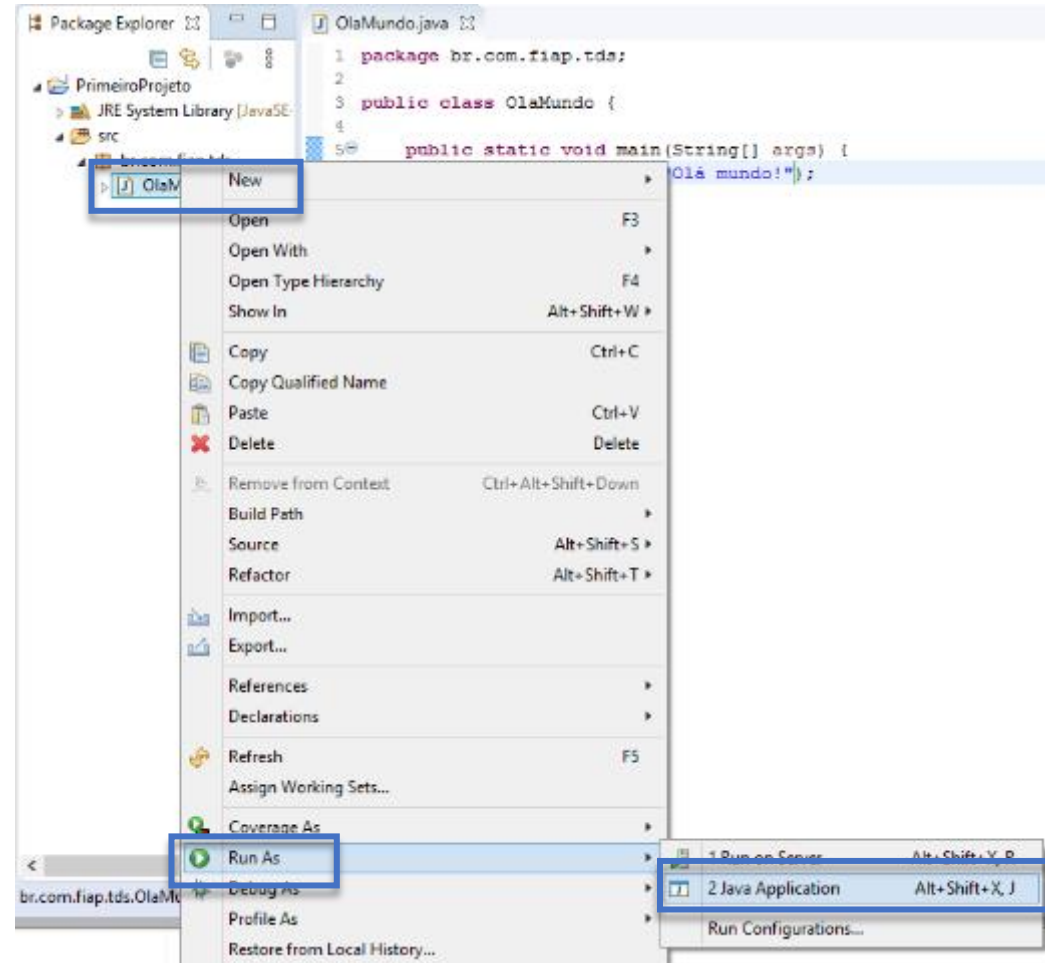
- Implemente o método **main** para executar o projeto **Java**:

```
public class OlaMundo {  
  
    public static void main(String[] args){  
        System.out.println("Olá Mundo");  
    }  
}
```



## | Executar o programa

- Para **executar**, clique com o botão direito do mouse na classe e escolha a opção: **Run**  
**As > Java Application;**



# Tipos Primitivos

# | Tipos de Dados

- A linguagem **Java** oferece diversos **tipos de dados** com os quais podemos trabalhar;
- Há basicamente **duas categorias** em que se encaixam os tipos de dados:
  - **Tipos primitivos** – correspondem aos tipos de dados mais básicos e usuais, ou seja, que possuem uma maior frequência de utilização;
    - **Exemplos:** int, boolean, double;
  - **Tipos por referência** - consistem em arrays (vetores, matrizes), classes e interfaces;
    - **Exemplos:** Casa, Carro, String, ContratoAluguel;



# | Tipos Primitivos

- **Java** tem **8 tipos primitivos**, agrupados em **4 categorias**:
  - **Lógico**: boolean (8 bits);
  - **Texto**: char (16 bits);
  - **Inteiro**: byte (8 bits), short (16 bits), int (32 bits), long (64 bits);
  - **Ponto flutuante**: float (32 bits) e double (64 bits);



# | Tipos de Dados

Tipo	Tamanho em bits	Valores (min. e max.)	Valor Padrão
boolean	8	true ou false	false
char	16	0 a 65.535	0
byte	8	-128 a +127	0
short	16	-32.768 a +32.767	0
int	32	-2.147.483.648 a +2.147.483.647	0
long	64	-9.223.372.036.854.775.808 a +9.223.372.036.854.775.807	0L
float	32	1,40129846432481707E-45 a 3,40282346638528860E+38	0.0f
double	64	4,94065645841246544E-324 a 1,79769313486231570E+308	0.0d

# Variáveis

# | Declaração de Variáveis

**Variáveis** são nomes **atribuídos à endereços na memória** de um computador onde se guardam dados. A **declaração de uma variável** consiste em **dar um nome para a posição de memória** a ser usada e especificar qual tipo de dado que será armazenado na memória.

- Para **declarar uma variável**, utiliza-se a seguinte sintaxe:
  - **<tipo> <nome da variável>**
- É possível **atribuir um valor a variável no momento da declaração**, utiliza-se a seguinte sintaxe:
  - **<tipo> <nome da variável> = <valor da variável>**
- No **Java**, também é possível **declarar mais de uma variável** do mesmo tipo de uma só vez, utiliza-se a seguinte sintaxe:
  - **<tipo> <nome da variável 1>, <nome da variável 2>**



# | Variáveis - Palavras Reservadas

- **Palavras reservadas**, ou palavras-chave, são **palavras que não podem ser utilizadas como identificadores**, ou seja, não podem ser usadas como **nome de variáveis**, **nome de métodos**, **nome de classes** e etc.;
- Na **linguagem Java** as seguintes **palavras são reservadas**:

abstract	continue	for	new	switch
assert	default	goto	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
const	float	native	super	while

# | Variáveis tipo String

- Para declarar uma **String**, utilizaremos a seguinte sintaxe:
  - **String <nome da variável> = new String();**
- É possível **atribuir** o valor da **String** no momento da sua declaração, utilizando a seguinte sintaxe:
  - **String <nome da variável> = <valor da variável>**

```
String endereco = new String();  
String complemento = "";  
String nome = "Pedro";
```

# | Operadores Aritméticos

Operador	Função
+	Adição
-	Subtração
*	Multiplicação
/	Divisão
%	Módulo (resto de divisão)
++	Incremento
--	Decremento
+=	Atribuição aditiva
-=	Atribuição subtrativa
*=	Atribuição multiplicativa
/=	Atribuição de divisão
%=	Atribuição de módulo

# | Operadores Aritméticos

```
public class Operadores {  
  
    public static void main(String args[]){  
        int m = 2 * 10;  
        System.out.println("Multiplicando 2*10 = " + m);  
        double d = 10 / 2;  
        System.out.println("Dividindo 10 por 2 = " + d);  
        int r = 10 % 3;  
        System.out.println("Resto da divisão 10/3=" + r);  
        int i = m++;  
        System.out.println("Incrementando " + i + " em 1 = " + m);  
    }  
}
```

# Entrada e Saída de dados

## | Comandos de Entrada de Dados

- Por enquanto, nossos programas terão apenas **entrada via console** (Command do Windows ou Shel do Linux);
- Para ler uma entrada no console, utilize a classe **Scanner**;
- A classe Scanner está na **biblioteca (java.util)** do Java. Para usá-la, você deve **importar** esta **biblioteca**;
- Agora, dentro do programa em si (dentro do método **main**), basta fazer a declaração abaixo no começo de seu código:
  - **Scanner entrada = new Scanner(System.in);**

# | Comandos de Entrada de Dados

- Sempre que precisar ler uma entrada do seu programa, use:

```
entrada.nextByte(); //para ler um byte  
entrada.nextShort(); //para ler um short entrada.nextInt(); //para ler  
um inteiro entrada.nextLong(); //para ler um long entrada.nextFloat();  
//para ler um float entrada.nextDouble(); //para ler um double  
entrada.nextBoolean(); //para ler um boolean entrada.next(); //para  
ler um uma palavra
```

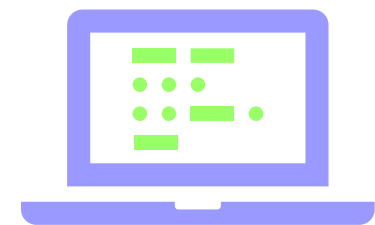
## | Comandos de Saída de Dados

- Por enquanto, nossos programas terão apenas **saída via console** (Command do Windows ou Shel do Linux);
- Para apresentar a **saída** no **console**, vamos usar um comando já conhecido:  
**System.out.println("texto");**
- A saída pode ser uma variável ou **concatenada** com **variáveis**:
  - **String msg = "Ola, como vai?";**
  - **System.out.println(msg);** //Perceba que não usa aspas
  - ou
  - **String nome = "Pedro";**
  - **System.out.println("Ola" + nome);**
  - O "+" é usado como operador de **concatenação** entre **Strings**.



## | Vamos a prática

- Dentro de um **pacote**, crie uma **classe** para representar uma **pessoa**, com os atributos **nome**, **idade** e **altura**.
- Em seguida, em um **outro pacote**, crie uma **classe de leitura de dados**, usando os mesmos atributos da classe **pessoa**, mas agora vamos usar o **Scanner** para que possamos **inserir** e logo após faça **exibir** o resultado com os dados digitados.



**Copyright © 2022 - 2023**  
**Prof. Rafael Desiderio**

Todos os direitos reservados. Reprodução ou divulgação total ou parcial deste documento é expressamente proibido sem o consentimento formal, por escrito do Professor (autor).