

# Técnica de Programação I



Curso Superior de Tecnologia em Desenvolvimento de  
Software Multiplataforma

Aula 01

Prof. Claudio Benossi



# Exercícios

01-) Crie um diagrama de classes UML para abstrair os atributos dos seguintes objetos:

- a) Eletrodoméstico
- b) Carro
- c) Caixa de Diálogo

02-) Implemente as classes do exercício acima (cada uma em um arquivo).

01-) Crie uma classe com o método main e instancie um objeto de cada uma das classes acima, coloque valores nos atributos e os mostre na tela.

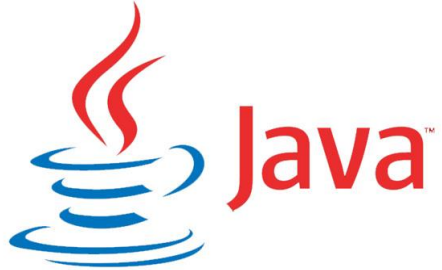




# 1. Unidade

## **Programação Orientada a Objetos**

# Softwares



- Java SE SDK  
<http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>



- NetBeans  
[www.netbeans.org](http://www.netbeans.org)



- IntelliJ  
<https://www.jetbrains.com/pt-br/idea/>



- Eclipse  
[www.eclipse.org](http://www.eclipse.org)



- JCreator  
[www.jcreator.com](http://www.jcreator.com)



- VS Code  
<https://code.visualstudio.com/download>



# NetBeans



O NetBeans é uma IDE gratuita e de código fonte aberto para desenvolvimento Java, porém extensível para diversas outras linguagens, como PHP, Python, JavaScript, etc.

Lançada em dezembro de 2000, o NetBeans é uma das principais IDEs para o desenvolvimento Java. Inicialmente desenvolvido como um software proprietário, em 2010, ao ser adquirido pela Oracle, o NetBeans se tornou parte do ecossistema Java, alavancando ainda mais sua utilização e popularidade.





# NetBeans



Porém, em 2016, a Oracle propôs mover o projeto NetBeans para um projeto aberto dentro da Apache, o chamando de Apache NetBeans.

Dentre suas principais características, podemos citar:

Multiplataforma: Podemos utilizar o NetBeans nos principais sistemas operacionais do mercado (Windows, Linux e macOS);





# NetBeans



Melhor suporte ao Java: Por fazer parte do ecossistema do Java, é a IDE oficial e recomendada pela própria Oracle;

Criação de interfaces: Possui suporte para criação de interfaces para aplicações web, desktop e mobile.

IDE oficial para o desenvolvimento Java, seja ela Desktop ou Web.





O IntelliJ IDE é uma das principais IDEs do mercado.

Criada pela JetBrains, uma empresa especializada no desenvolvimento de IDEs, o IntelliJ teve um crescimento impressionante nos últimos anos.

Apesar de ter sido lançada em 2001, foi a partir de 2010 que a IDE começou a ser reconhecida no mercado.





Em 2014, a Google anunciou que o Android Studio, uma IDE baseada no IntelliJ IDE para criação de aplicações Android, seria a IDE oficial para o desenvolvimento Android.

Com este anúncio, as IDEs desenvolvidas pela JetBrains ganharam mais visibilidade.

Dentre suas principais características, podemos citar:

- ▶ Assistente de código: Possui um ótimo assistente de código, autocompletando trechos de sentenças para facilitar a criação de aplicações;
- ▶ Uso de plugins: É possível desenvolver em diferentes tecnologias com o IntelliJ (Python, Dart, etc) com o uso de plugins;

Dentre suas principais características, podemos citar:

- Suporte nativo ao Kotlin: Podemos desenvolver aplicações utilizando o Kotlin, linguagem baseada no Java criada pela própria JetBrains.



O IntelliJ cresceu muito nos últimos anos, se tornando uma das principais IDEs para o desenvolvimento de aplicações Java.

O IntelliJ possui duas versões, a “Ultimate” que possui diversos recursos, como ferramentas de bancos de dados, suporte nativo ao Spring e detecção de duplicidades.



# Eclipse



Lançada em 2001, possuindo como autor a IBM, sobre a licença EPL (Eclipse Public Licence), o Eclipse é uma IDE para desenvolvimento em Java que também suporta diversas outras linguagens apenas com a instalação de plugins (C/C++, PHP, Python, Kotlin, entre outras).



Dentre suas principais características podemos citar:

- ▶ Multiplataforma: Pode ser executado nos diferentes sistemas operacionais (Windows, Linux e macOS);
- ▶ Tecnologia baseada em plugins: Através da instalação de plugins, o desenvolvedor poderá incrementar as funcionalidades do Eclipse;

Dentre suas principais características podemos citar:

- ▶ Pacotes de desenvolvimento: Podemos utilizar diversos pacotes de desenvolvimento para criar diferentes tipos de aplicações com Java (Web e Desktop);
- ▶ Uso de SWT (Standard Widget Toolkit): Widget toolkit para uso com a plataforma Java;

Dentre suas principais características podemos citar:

- Criação de aplicações gráficas multiplataforma: Com o Eclipse podemos criar interfaces gráficas para aplicações Java.

O Eclipse é uma excelente IDE, muito utilizada no mercado. Desta forma, seu uso facilita a criação de aplicações Java tanto para Desktop ou Web.



O JCreator é uma IDE Java criado pela Xinox Software e sua interface lembra a do Microsoft Visual Studio. Por ela ser programada totalmente em C++, a Xinox afirma que o JCreator é mais rápido que as IDE(s) concorrentes baseadas na linguagem JAVA.

Essa IDE está disponível apenas para o sistema operacional Windows, porém, tanto a versão LE quanto a versão Pro do JCreator funciona adequadamente no Linux usando o Wine.

O conjunto de recursos disponíveis para a versão Pro é comparável à de outros IDE(s) no que diz respeito a recursos de gerenciamento e edição de projetos, mas não tem recursos avançados, tais como refatoração, dentre outros, que pode ser encontrada em outras IDE(s) que acercam a linguagem JAVA como Eclipse e Netbeans, por exemplo.

Já a versão gratuita LE carece de mais alguns recursos como conclusão de código, que são incluídas com outras IDEs livres e também não tem o nível de extensibilidade através de plug-ins de terceiros que é comum em outras IDE(s) populares do JAVA.

Dentre as características do JCREATOR, podemos destacar:

- Diferentes perfis do JDK podem ser utilizados;
- Fácil visualização do projeto com navegador de classe;
- Depuração com uma interface fácil e intuitiva;
- Assistentes ajudam a ir direto ao ponto de escrever o projeto de uma forma ágil e facilitada;
- Interface de usuário muito parecida com a Microsoft Visual Studio;

A IDE JCreator não requer um ambiente de tempo de execução JAVA para executar, o que pode torná-lo rápido no que comparado em outras IDE(s) baseadas em JAVA.

Além disso, temos como vantagens do JCREATOR uma interface um modo simples e facilitado e ele ocupa pouco tamanho em disco, depois de instalado, comparado com outras IDE(s).

Entre as desvantagens podemos destacar que a IDE JCreator está disponível apenas para o sistema operacional Windows, embora o Linux Wine pode ser usado em sistemas Unix para executar o JCreator.

Além disso, na maioria das vezes, o JCreator tem o navegador Internet Explorer como sendo o padrão, as configurações de impressoras usadas para impressões não são confiáveis e suas versões são pagas.



# VS Code



Lançado pela Microsoft em 2015, o Visual Studio Code (VSCode) é um editor de código para desenvolvimento de aplicações web, tendo uma grande adoção pelas comunidades de diversas linguagens e tecnologias, não atendendo somente a projetos ASP.NET como também projetos em Node.js.





# VS Code



Dentre suas principais características podemos citar:

- Multiplataforma: Pode ser executado nos diferentes sistemas operacionais (Windows, Linux e macOS);
- Open source: seu código foi disponibilizado no GitHub, o que permite à comunidade contribuir com a criação de extensões e novas funcionalidades.
- Integração com o git





# Git e GitHub



**Git = Sistema de controle de versão.**

- Lugar onde posso guardar diferentes versões de um arquivo, que pode ser compartilhada por uma ou mais pessoas, controlando as alterações realizadas, indicando as alterações realizadas e quem as realizou.



**GitHub = Rede social de desenvolvedores.**

- Você pode criar seus projetos (seu portfólio) e compartilhar com outros desenvolvedores.

# Como Funciona o Java



```
import java.io.*;
import java.net.*;

public class Iluminar_Ambiente implements Actuador {

    @Override
    public Boolean setValue(String value){
        try {
            URL address = new URL(value);
            URLConnection connection = address.openConnection();
            BufferedReader in = new BufferedReader(new InputStreamReader(
                connection.getInputStream()));

            String inputLine;
            while ((inputLine = in.readLine()) != null)
                System.out.println(inputLine);
            in.close();
        } catch (Exception e) {}
        return true;
    }

    public static void main(String args[]){
        Iluminar_Ambiente IA = new Iluminar_Ambiente();
        //SystemBox address: 10.1.1.7
        //Device light bulb: unit 119 and Commands: ON = 1 / OFF = 0
        String value = "http://10.1.7.30/monitor/"
            + "monitor.cgi?ref_page=cmd&unit=119&newvalue=1";
        Boolean command = IA.setValue(value);
    }
}
```

Código Fonte



Não Funciona!

# Como Funciona o Java

```
import java.io.*;
import java.net.*;

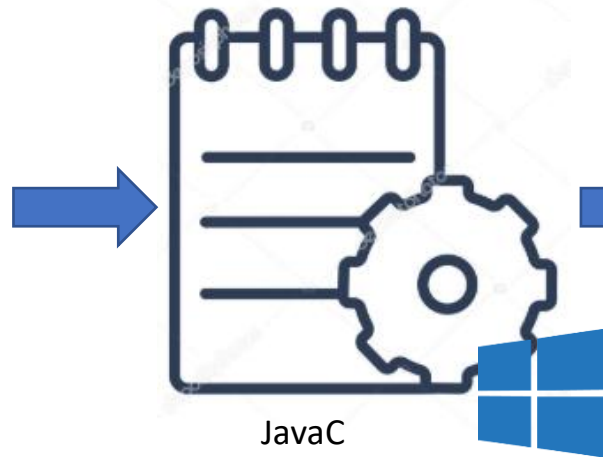
public class Iluminar_Ambiente implements Actuador {

    @Override
    public Boolean setValue(String value){
        try {
            URL address = new URL(value);
            URLConnection connection = address.openConnection();
            BufferedReader in = new BufferedReader(new InputStreamReader(
                connection.getInputStream()));

            String inputLine;
            while ((inputLine = in.readLine()) != null)
                System.out.println(inputLine);
            in.close();
        } catch (Exception e) {}
        return true;
    }

    public static void main(String args[]){
        Iluminar_Ambiente IA = new Iluminar_Ambiente();
        //SystemBox address: 10.1.1.7
        //Device light bulb: unit 119 and Commands: ON = 1 / OFF = 0
        String value = "http://10.1.7.30/monitor/"
            + "monitor.cgi?ref_page=cmd&unit=119&newvalue=1";
        Boolean command = IA.setValue(value);
    }
}
```

Código Fonte



JavaC



ByteCode



JVM

Funciona!



# Como Funciona o Java

```
import java.io.*;
import java.net.*;

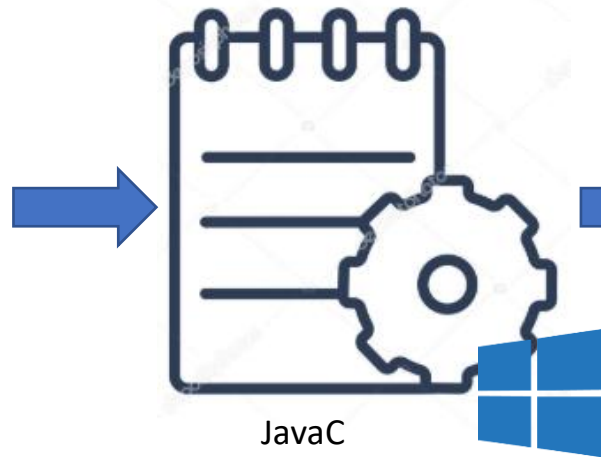
public class Iluminar_Ambiente implements Actuador {

    @Override
    public Boolean setValue(String value){
        try {
            URL address = new URL(value);
            URLConnection connection = address.openConnection();
            BufferedReader in = new BufferedReader(new InputStreamReader(
                connection.getInputStream()));

            String inputLine;
            while ((inputLine = in.readLine()) != null)
                System.out.println(inputLine);
            in.close();
        } catch (Exception e) {}
        return true;
    }

    public static void main(String args[]){
        Iluminar_Ambiente IA = new Iluminar_Ambiente();
        //SystemBox address: 10.1.1.7
        //Device light bulb: unit 119 and Commands: ON = 1 / OFF = 0
        String value = "http://10.1.7.30/monitor/"
            + "monitor.cgi?ref_page=cmd&unit=119&newvalue=1";
        Boolean command = IA.setValue(value);
    }
}
```

Código Fonte



JavaC



ByteCode



JVM



Funciona!

# Como Funciona o Java

```
import java.io.*;
import java.net.*;

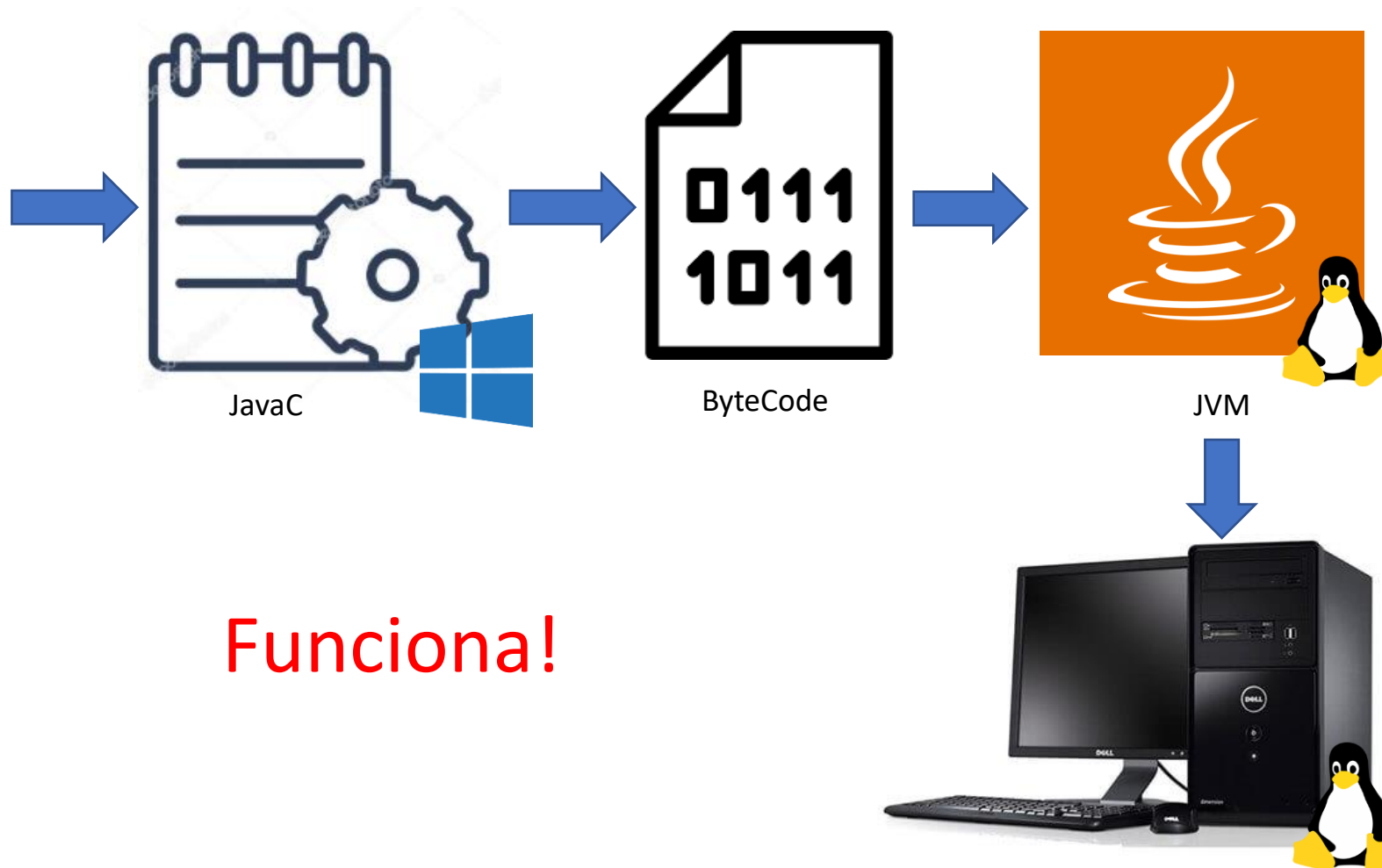
public class Iluminar_Ambiente implements Actuador {

    @Override
    public Boolean setValue(String value){
        try {
            URL address = new URL(value);
            URLConnection connection = address.openConnection();
            BufferedReader in = new BufferedReader(new InputStreamReader(
                connection.getInputStream()));

            String inputLine;
            while ((inputLine = in.readLine()) != null)
                System.out.println(inputLine);
            in.close();
        } catch (Exception e) {}
        return true;
    }

    public static void main(String args[]){
        Iluminar_Ambiente IA = new Iluminar_Ambiente();
        //SystemBox address: 10.1.1.7
        //Device light bulb: unit 119 and Commands: ON = 1 / OFF = 0
        String value = "http://10.1.7.30/monitor/"
            + "monitor.cgi?ref_page=cmd&unit=119&newvalue=1";
        Boolean command = IA.setValue(value);
    }
}
```

Código Fonte



# Como Funciona o Java

```
import java.io.*;
import java.net.*;

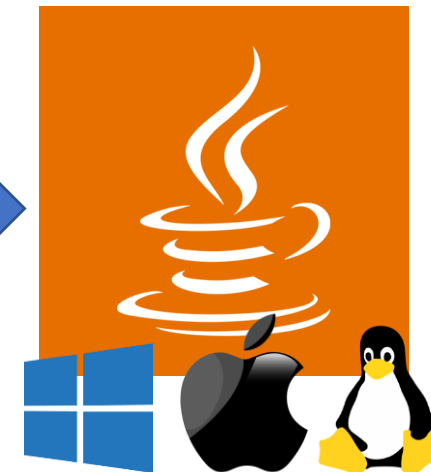
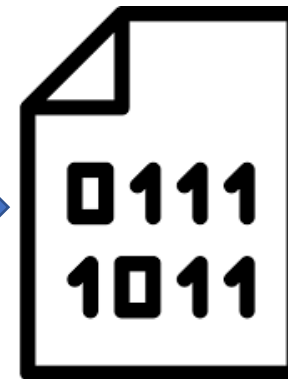
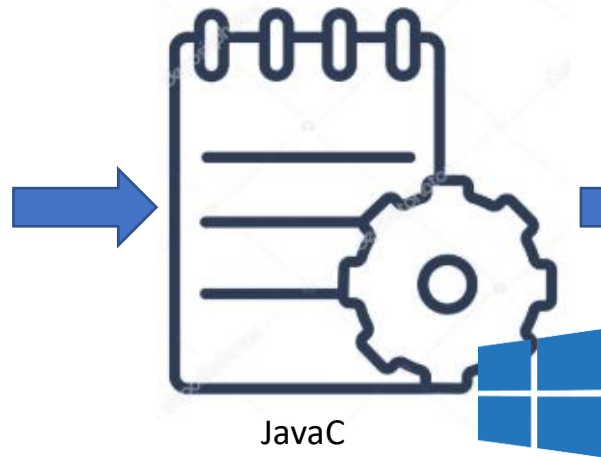
public class Iluminar_Ambiente implements Actuador {

    @Override
    public Boolean setValue(String value){
        try {
            URL address = new URL(value);
            URLConnection connection = address.openConnection();
            BufferedReader in = new BufferedReader(new InputStreamReader(
                connection.getInputStream()));

            String inputLine;
            while ((inputLine = in.readLine()) != null)
                System.out.println(inputLine);
            in.close();
        } catch (Exception e) {}
        return true;
    }

    public static void main(String args[]){
        Iluminar_Ambiente IA = new Iluminar_Ambiente();
        //SystemBox address: 10.1.1.7
        //Device light bulb: unit 119 and Commands: ON = 1 / OFF = 0
        String value = "http://10.1.7.30/monitor/"
            + "monitor.cgi?ref_page=cms&unit=119&newvalue=1";
        Boolean command = IA.setValue(value);
    }
}
```

Código Fonte



Funciona!



# Como Funciona o Java

## WORA

W<sub>rite</sub> O<sub>nce</sub> R<sub>un</sub> A<sub>nywhere</sub>

(Escreva um vez e execute em qualquer lugar)



# Como Funciona o Java

## JDK

Java  
Development  
Kit



Quem quer Programar

## JRE

Java  
Run  
Environment



Quem que Usar



# Como Funciona o Java

**JDK**

Java  
Development  
Kit



Quem quer Programar

**JRE**

Java  
Run  
Environment



Quem que Usar

J  
R  
E

JVM

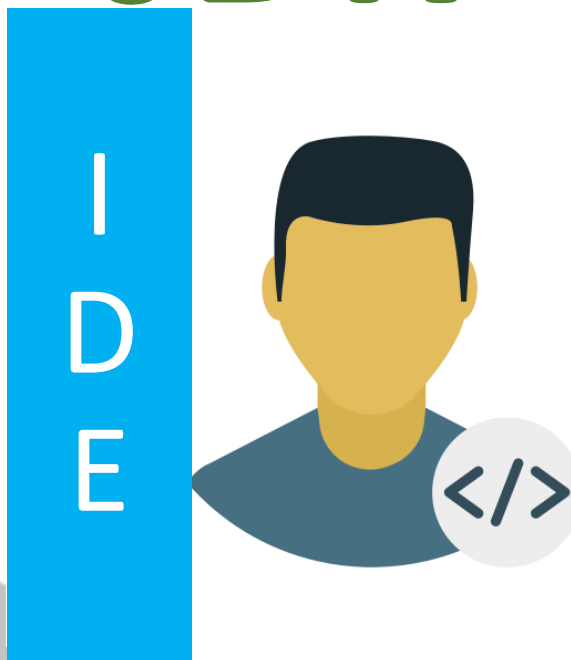
Loader / Verificador  
Interpretador / Gerenciador  
Compilador JIT / ...

Bibliotecas

# Como Funciona o Java

## JDK

Java  
Development  
Kit



Java Lang

Java Tools

JavaC  
Debugger  
APIs

Quem quer Programar

IDE não vem instalado no JDK – Ambiente de Desenvolvimento Integrado

## JRE

Java  
Run  
Environment



Quem que Usar

# Como Funciona o Java

- Editor Completo
- Compilador Linker Debugger
- Gerador de Código
- Ambiente de Testes
- Distribuição Simplificada

I  
D  
E

Integrate  
Development  
Environment



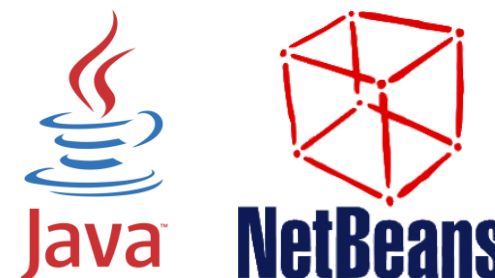
IDE não vem instalado no JDK – Ambiente de Desenvolvimento Integrado



# NetBeans

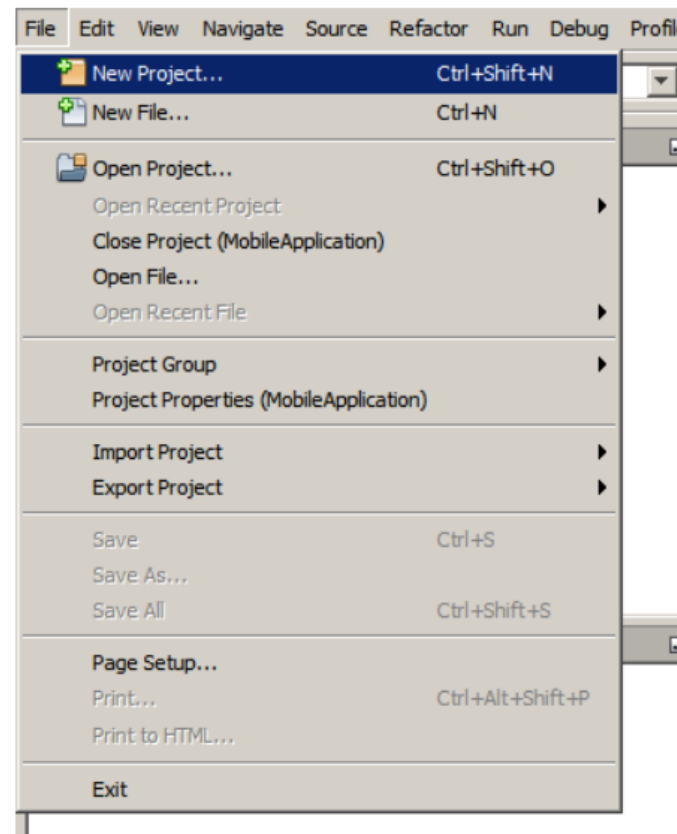
Este conteúdo oferece uma introdução simples e rápida ao fluxo de trabalho do NetBeans IDE, orientando você na criação de uma aplicação de console simples do "Hello World" de Java.

Com isso você terá adquirido um conhecimento geral sobre como criar e executar aplicações no IDE.



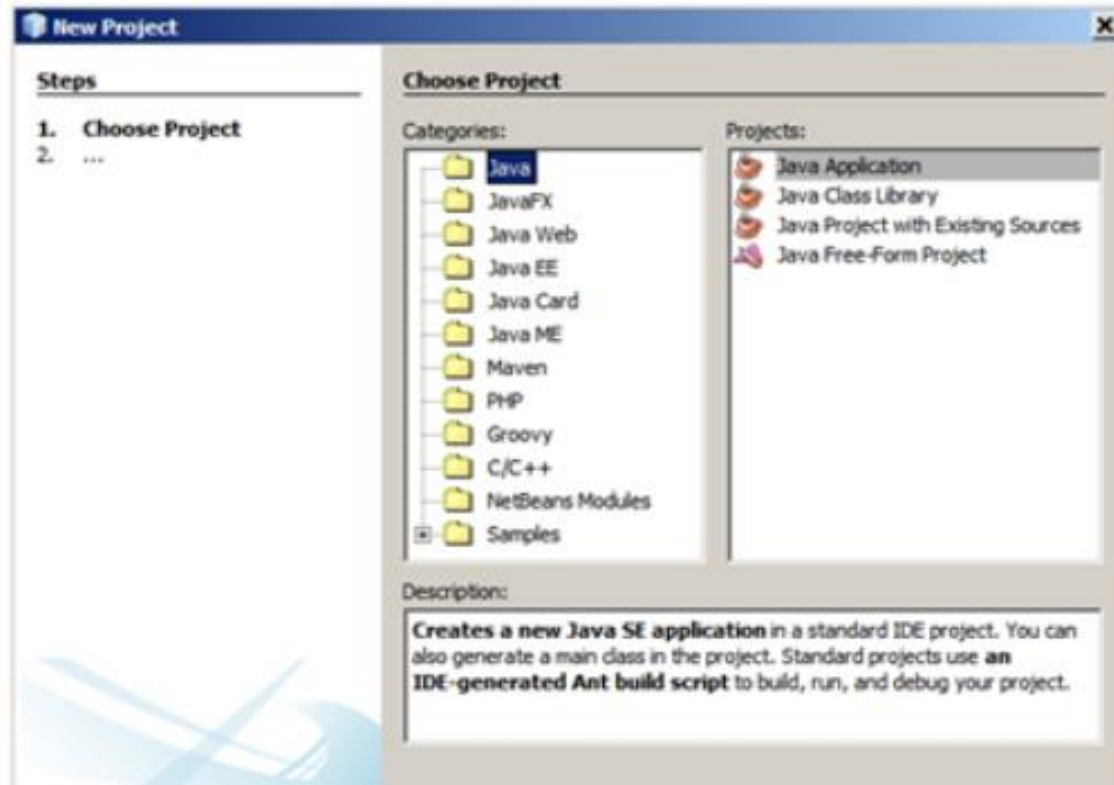
# Configurando o Projeto

Para criar um projeto do IDE: Inicie o NetBeans IDE.  
No IDE, escolha Arquivo > Novo Projeto.



# Configurando o Projeto

No assistente **Novo Projeto**, expanda a categoria Java e selecione **Aplicação Java**, em seguida, clique em **Próximo**.



# Configurando o Projeto

Na página Nome e Localização do assistente, adote o procedimento a seguir: no campo Nome do Projeto, digite **HelloWorldApp**.

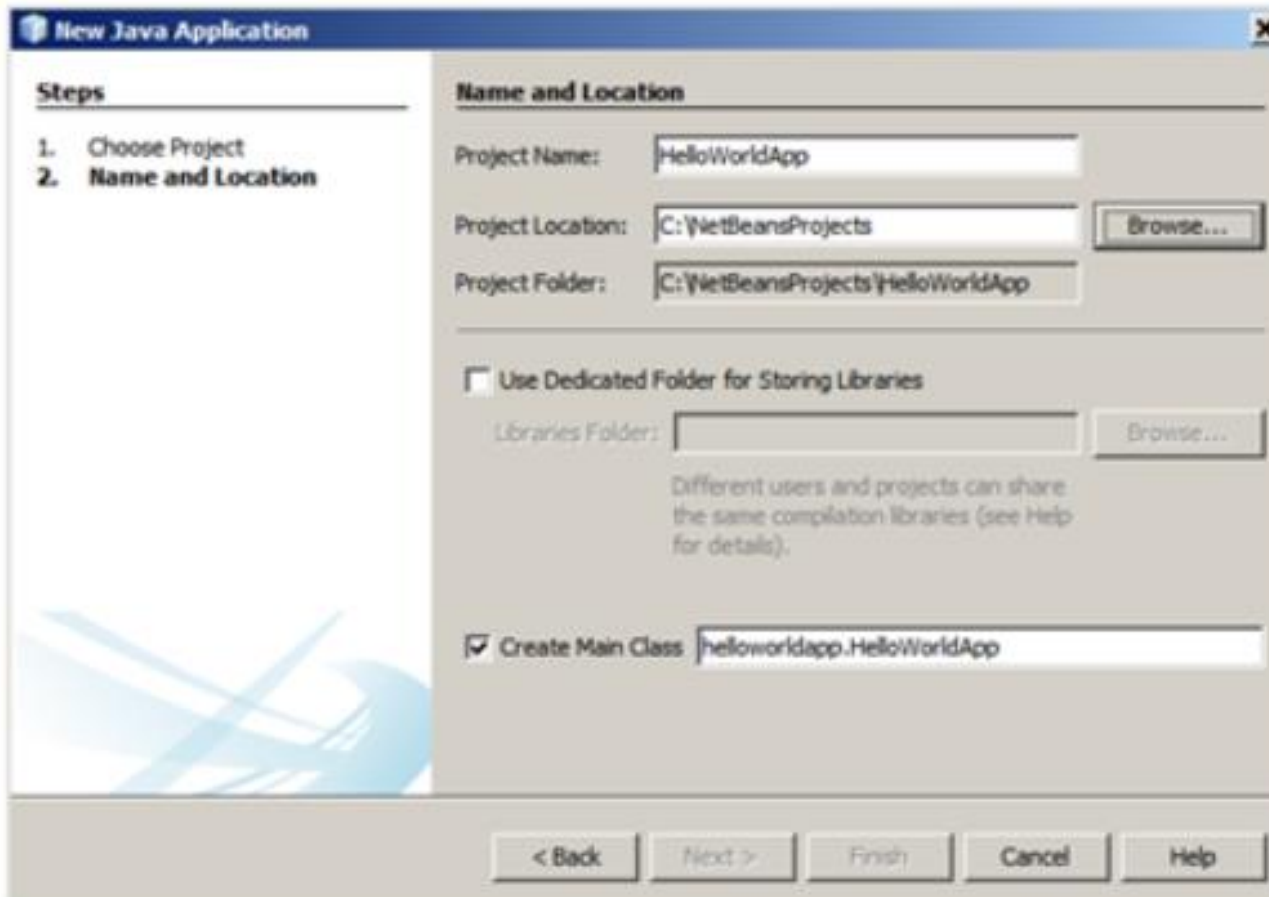
Deixe desmarcada a caixa de seleção **Utilizar Pasta Dedicada para Armazenar Bibliotecas**.

No campo Criar Classe Principal, digite **helloworldapp>HelloWorldApp**.



# Configurando o Projeto

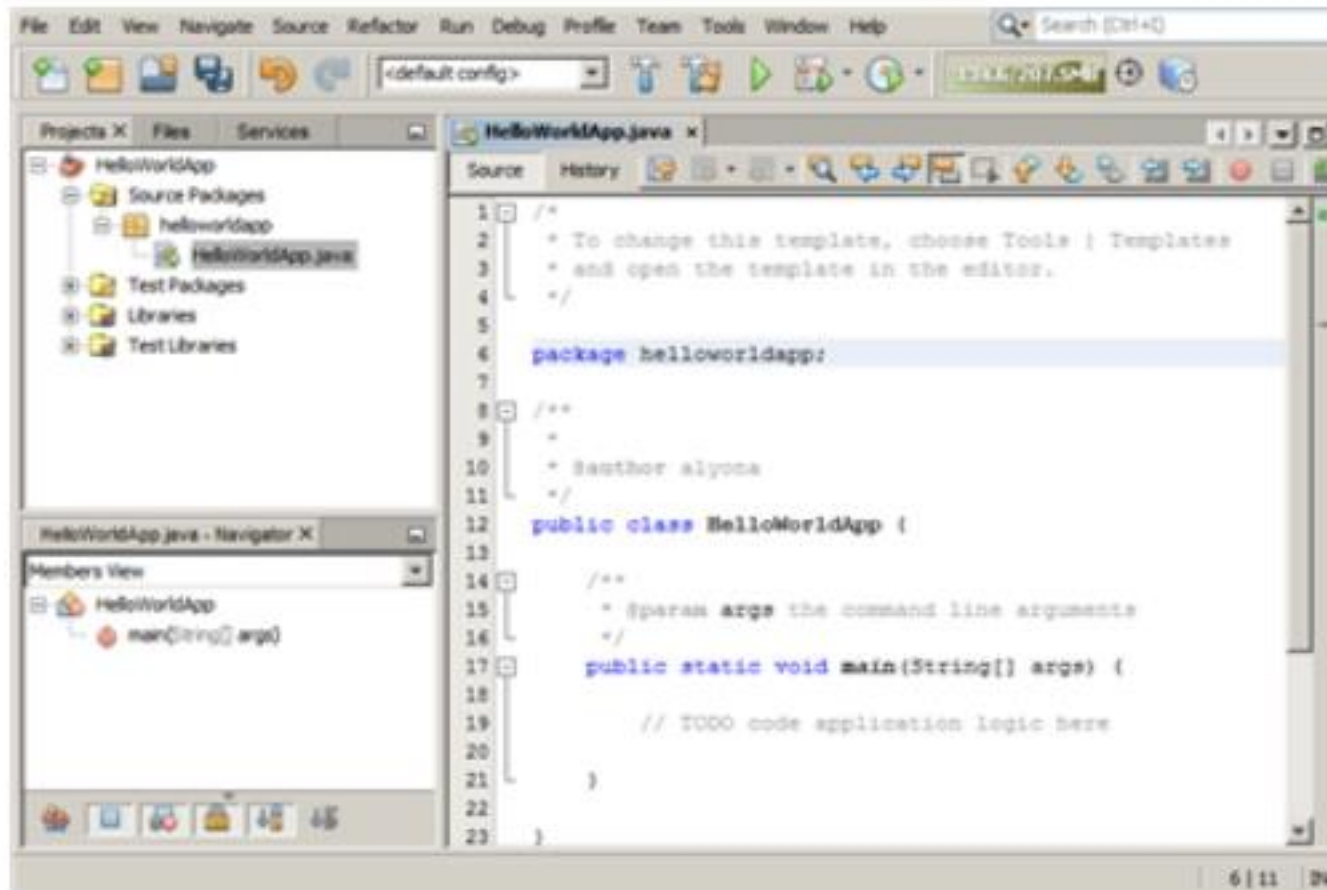
Clique em finalizar:





# Configurando o Projeto

O projeto é criado e aberto no IDE.



# Configurando o Projeto

Agora você deve ver os seguintes componentes:

- A janela Projetos, que contém uma view em árvore dos componentes do projeto, incluindo arquivos de código-fonte, bibliotecas de que seu código depende, e assim por diante.

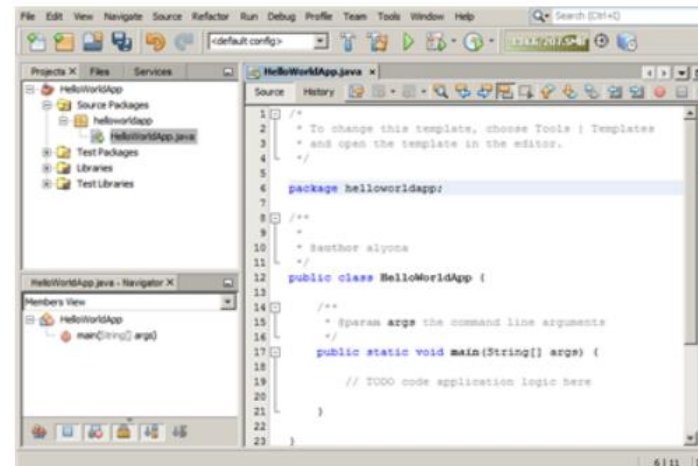


# Configurando o Projeto

O projeto é criado e aberto no IDE.

Agora você deve ver os seguintes componentes:

- ▶ A janela Editor de Código-fonte com um arquivo chamado **HelloWorldApp** é aberta.

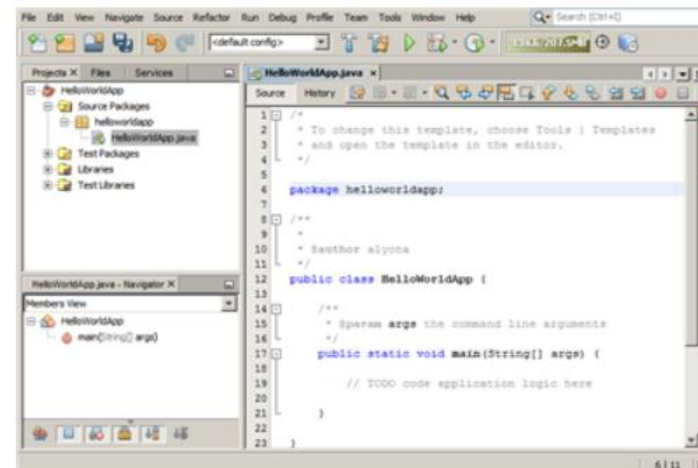


# Configurando o Projeto

O projeto é criado e aberto no IDE.

Agora você deve ver os seguintes componentes:

- ▶ A janela Navegador, que você pode utilizar para navegar rapidamente entre elementos dentro da classe selecionada.



## Adicionando Código ao Arquivo de Origem Gerado

Como a caixa de seleção Criar Classe Principal foi marcada no assistente de Novo Projeto, o IDE criou uma classe principal de esqueleto.

Você pode adicionar a mensagem "**Hello World!**" ao código de esqueleto substituindo a linha:

pela linha:

```
// TODO code application logic here
```

```
System.out.println("Hello World!");
```



## Adicionando Código ao Arquivo de Origem Gerado

Salve a alteração escolhendo Arquivo > Salvar.

O arquivo deve ter uma aparência semelhante à seguinte amostra de código.

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package helloworldapp;

/**
 *
 * @author <your name>
 */
public class HelloWorldApp {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }

}
```



# Compilando e Executando o Programa

Devido à funcionalidade Compilar ao Salvar do IDE, não é necessário compilar manualmente o projeto para que seja executado no IDE.

Quando um arquivo de código-fonte Java é salvo, ele é compilado automaticamente pelo IDE.



# Compilando e Executando o Programa

A funcionalidade Compilar ao Salvar pode ser desativado na janela Propriedades do Projeto.

Clique com o botão direito do mouse no projeto e selecione Propriedades.



# Compilando e Executando o Programa

Na janela Propriedades, escolha a guia Compilação.

A caixa de seleção Compilar ao Salvar está na parte superior.

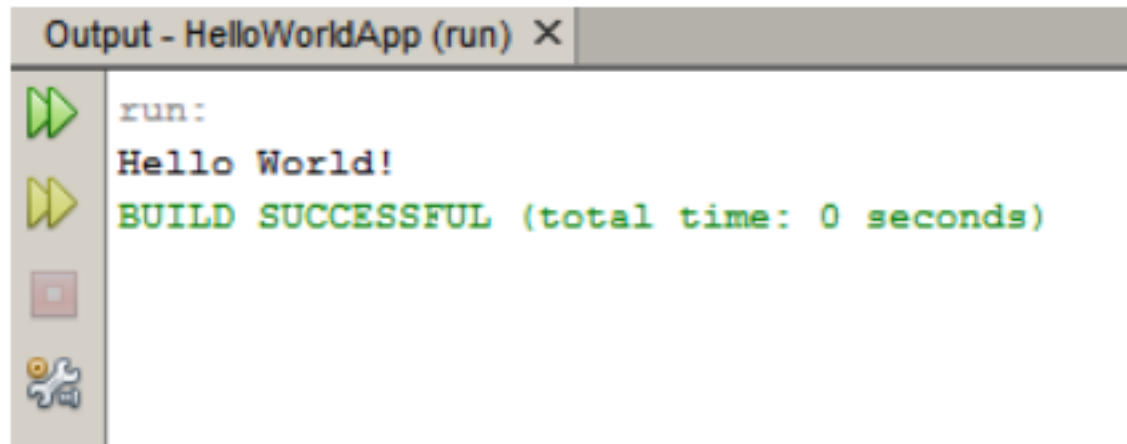
Observe que, na janela Propriedades do Projeto, é possível configurar várias definições para o projeto: bibliotecas do projeto, encapsulamento, construção, execução, etc.



# Compilando e Executando o Programa

Para executar o programa:

Escolha Executar > Executar Projeto.



```
Output - HelloWorldApp (run) X
run:
Hello World!
BUILD SUCCESSFUL (total time: 0 seconds)
```

Parabéns!

# Compilando e Executando o Programa

Se houver erros de compilação, eles são marcados com glifos vermelhos nas margens esquerda e direita do Editor de Código-fonte.

Os glifos da margem esquerda indicam os erros das linhas correspondentes. Os glifos da margem direita mostram todas as áreas do arquivo que apresentam erros, incluindo os erros das linhas que não estão visíveis.



# Compilando e Executando o Programa

É possível passar o mouse sobre a marca do erro para ver a descrição deste erro.

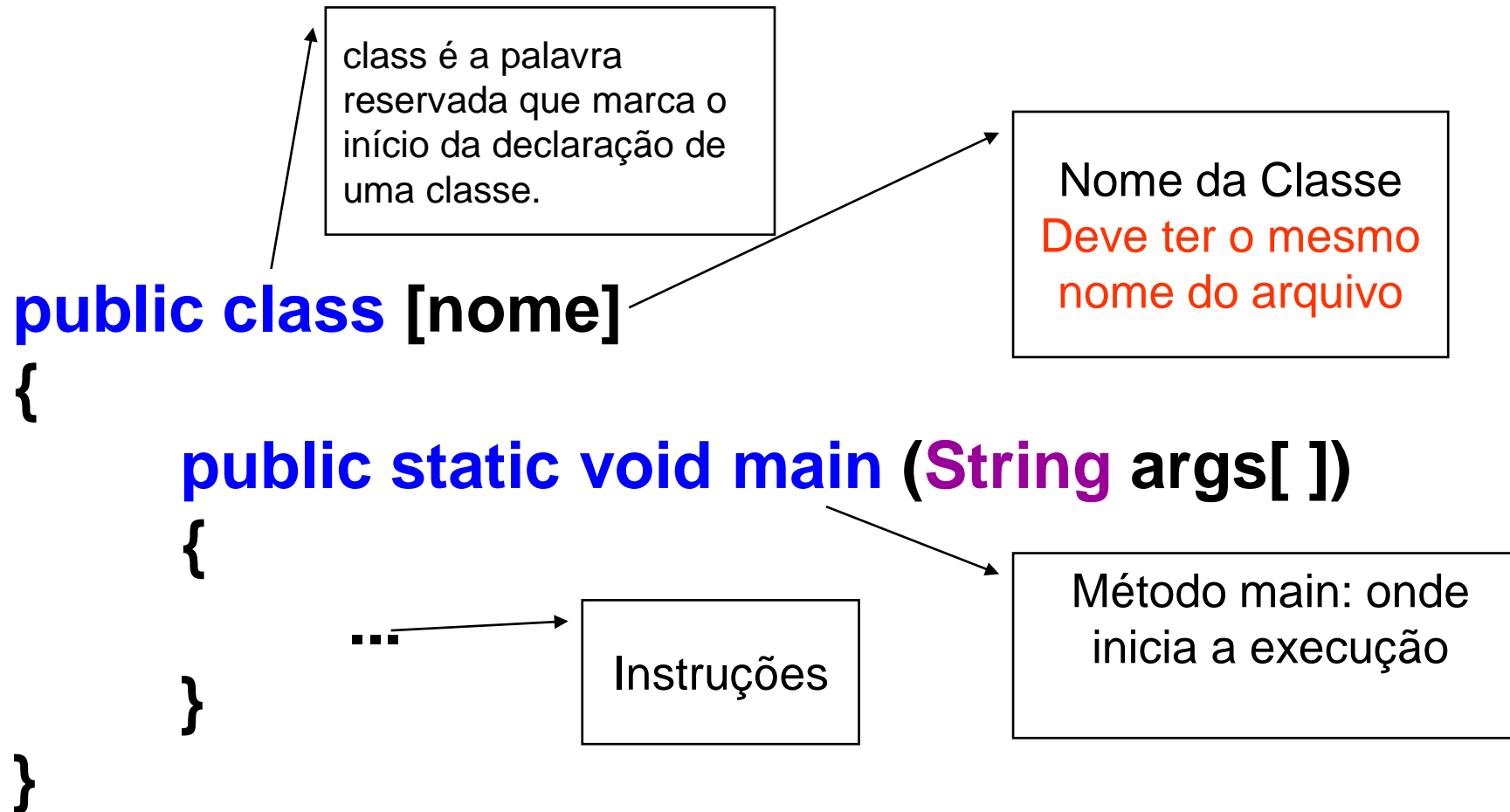
É possível clicar em um glifo da margem direita para ir para a linha que apresenta o erro.

## Referência:

[https://netbeans.org/kb/docs/java/quickstart\\_pt\\_BR.html](https://netbeans.org/kb/docs/java/quickstart_pt_BR.html)



# Estrutura de uma classe Java executável



Essa estrutura estará em todos os programa desenvolvidos em java

Exemplo1.java

Nome do  
arquivo

```

1  /*+++++++
2  Programa: Exibe mensagem na tela
3  Autor: Cristiane RGM: 29801-8
4
5  ++++++
6
7  public class Exemplo1{
8  //o método main inicia a execução do aplicativo Java
9      public static void main (String args[ ]){
10         System.out.println("Primeiro exemplo");
11     } //fim do método main
12 } //fim da classe Exemplo1

```

O mesmo nome  
do arquivo

/\* Esse é um  
comentário  
com várias  
Linhas\*/

Inicia-se com o delimitador /\* e  
termina com \*/. O comentário  
é ignorado pelo compilador.

# Dicas

- Termine (quase...) todas as linhas com ;
- Quando ocorrer um erro de compilação, dê um duplo clique sobre a mensagem de erro para destacar o comando errado no programa e verifique também a linha anterior, que pode ser a responsável pelo erro, especialmente se faltar o ;
- Como convenção, todos os nomes de classe (identificador) em Java iniciam com uma letra maiúscula.
- Use comentários, iniciados por // ou /\* ... \*/

letras, dígitos, sublinhados(\_) e sinais de cifrão (\$) que não iniciem com dígito e não contenham espaços em branco nem ponto. Em Java o nome Exemplo1 é diferente de exemplo1.

# Tipos de Dados

- Em Java, os tipos são oito: **byte, short, int, long, float, double, char** e **boolean**.
- Esses tipos são portáveis entre todas as plataformas.
- Podem ser agrupados em quatro categorias:

<b>Inteiros</b>	<b>Reais</b>	<b>Literal</b>	<b>Lógico</b>
byte	float	char	boolean
short	double		
int			
long			



# Tipos de dados

Tipo de dado	Tamanho em bytes e comentários
boolean	1 (valores true ou false)
char	2 (valores de 0 a 65535, conjunto de caracteres Unicode ISO)
byte	1 (valores de -128 a +127)
short	2 (valores inteiros de -32768 a 32767)
int	4 (valores inteiros de -2.xx bilhões a 2.xx bilhões)
long	8 (valores inteiros neg. e positivos de 19 dígitos)
float	4 (valores com casas decimais entre $\approx -3 \times 10^{38}$ e $+3 \times 10^{38}$ )
double	8 (valores com casas decimais entre $\approx -1 \times 10^{308}$ e $+1 \times 10^{308}$ )



# String (conjunto de caracteres)

Armazena um conjunto de dados alfanuméricos.

Ele é sempre representado por aspas duplas. Exemplos:

- “Hoje”
- “Aula de Java”
- “Amanhã é sexta-feira, dia de *happy hour*.”





# Variáveis

São espaços ou alocações nas quais os dados são armazenados.

A declaração de uma variável instrui o programa a reservar um espaço na memória, para que seja possível armazenar um dado de determinado tipo.

A quantidade de memória reservada para uma variável é definida pelo seu tipo.

`int`  $\Rightarrow$  4 bytes (32 bits)

# Declaração de Variáveis

Pode ser feita em qualquer parte do **corpo de uma classe**.

- Pode ser composta por:
  - Um tipo
  - Um identificador (nome)
  - Um valor (opcional)
  - Ponto-e-vírgula

# Exemplos de Declaração de Variáveis

```
//Tipos inteiros
```

```
byte idade;
```

```
short nro;
```

```
int cod;
```

```
long qtde;
```

```
//Tipos reais
```

```
float = preco;
```

```
double = receber;
```

```
//Tipo caracter
```

```
char = letra;
```

```
//Tipo lógico
```

```
boolean = escolha;
```

# Identificador (nome):

É através dele que você irá se referir a variável no código para lhe atribuir algum dado ou para recuperar um dado que foi armazenado.

Regras para os nomes das variáveis:

- Podem começar com letra minúscula ou maiúscula, underscore (\_), ou símbolo de dólar (\$);
- Não podem conter pontuação nem espaços;
- Não podem ser utilizadas palavras reservadas.

# Palavras Reservadas

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

# Variáveis

As variáveis são *sensitive* → Ar ≠ ar

Nomes válidos:	Nomes inválidos:
codigo	1teste
quantidade	nome cliente
\$nome	cli.nome
_nro	x - y
indice1	código



# Atribuição de valores a variáveis

Existem várias maneiras de atribuir valores a variáveis:

- Dizendo no código qual o valor a variável deve assumir:

```
int cod = 123;  
float preco = 12.35F;  
char letra = 'A';
```

Identifica que o  
valor é do tipo  
float

- Definir que uma variável assuma o valor de uma outra variável:

```
int cod1, cod2;  
cod1 = 54;  
cod2 = cod1;
```

# Atribuição de valores a variáveis

Atribuir uma variável o resultado de uma expressão:

```
float = preco, qtde_comprada;  
double = receber;  
receber = preco*qtde_comprada;
```

Entre outros.

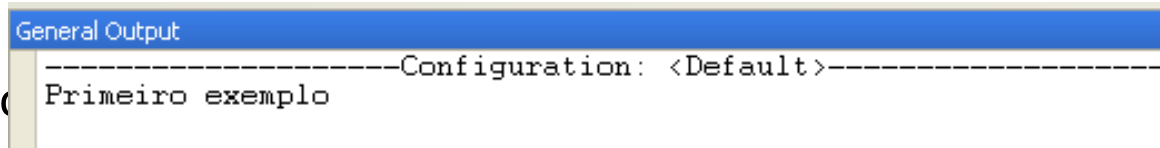
# Comandos de entrada e saída

Saída:

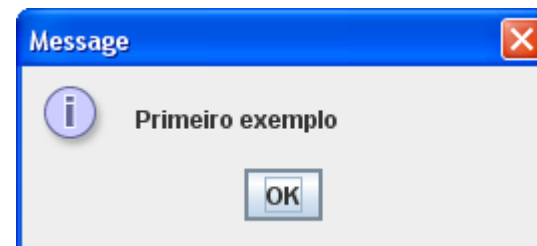
- Modo texto:

```
System.out.println("Primeiro exemplo");
```

- Modo Gráfico



```
JOptionPane.showMessageDialog(null, "Primeiro exemplo");
```



# Comandos de saída – Modo Gráfico

```
JOptionPane.showMessageDialog(null, "Primeiro exemplo");
```

- Já está pronta, porém precisamos indicar que iremos utilizar e para isso temos que *importar* um pacote.

```
import javax.swing.JOptionPane;
```

- Carrega a classe **JOptionPane** do pacote **javax.swing**
- Ajuda a definir interfaces gráficas com o usuário para seu aplicativo
- Sempre coloquem esta diretiva no início do programa.

# Detalhes do comando

```
JOptionPane.showMessageDialog(null,  
"Primeiro: "+valor1+ " Segundo: "+valor2);
```

- Indica uma chamada para o método ***showMessageDialog*** da classe **JOptionPane**, exige dois argumentos, separados por vírgula.
- **null** → ajuda o Java a posicionar a caixa de diálogo (caixa de diálogo aparece no centro da tela).

# Entrada de dados no Modo Texto

- Para utilizarmos o comando de entrada no modo texto, é necessário importar o seguinte pacote:

```
import java.util.Scanner;
```

- A classe Scanner ajuda na leitura dos dados informados. Para isso, é necessário criar um objeto do tipo **Scanner** que recebe o objeto System.in, do seguinte modo:

```
Scanner leia = new Scanner(System.in);
```

faz a leitura do  
que se escreve  
no teclado

## Entrada de dados no Modo Texto

Para cada tipo de dado primitivo existe uma chamada do método para retornar o valor especificado na entrada de dados, sempre seguindo o formato `nextTipoDado()`.

```
Scanner leia= new Scanner(System.in);
```

```
float peso= leia.nextFloat();
```

```
double salario = leia.nextDouble();
```

```
int idade= leia.nextInt();
```

```
byte valor1= leia.nextByte();
```

```
long valor2= leia.nextLong();
```

```
boolean b1 = leia.nextBoolean();
```

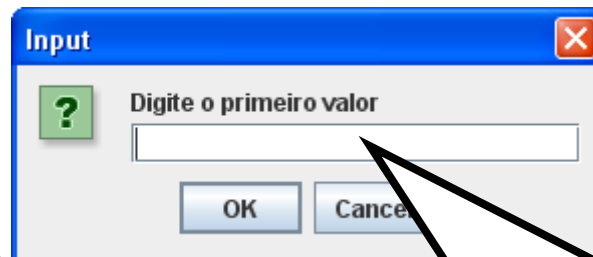
```
String nome = leia.nextLine();
```

# Comando de entrada – Modo Gráfico

- Modo Gráfico:

```
valor = JOptionPane.showInputDialog("Digite o primeiro valor");
```

A variável que recebe conteúdo é uma String



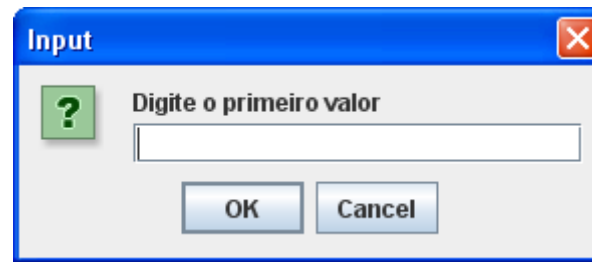
Mesmo que o usuário digite um valor numérico, o conteúdo é uma String

Também precisamos *importar* o pacote...



## Comando de entrada – Modo Gráfico

```
valor = JOptionPane.showInputDialog("Digite o primeiro valor");
```



Podemos converter uma String para um dado do tipo numérico para realizar cálculos matemáticos, para isso:

```
n1 = Integer.parseInt(valor);
```

Declarada como  
int

## Conversões de tipos – String para outros tipos

**Para o tipo int:**

`Integer.parseInt (variável)`

**Para o tipo double:**

`Double.parseDouble(variável)`

**Para o tipo float:**

`Float.parseFloat(variável)`

**Para o tipo long:**

`Integer.parseInt(variável)`

**Para o tipo char:**

`entrada.charAt(0);`

# Construtores

São trechos de código invocados automaticamente quando um objeto é instanciado.

**NUNCA** retornam nada e não possuem tipo.

Os construtores sempre tem o mesmo nome da classe a que se refere.

Para cada objeto, o construtor é chamado exatamente uma vez, na sua criação.

Exemplo:

```
Objeto obj = new Objeto();
```

Alguns podem requerer parâmetros

```
Objeto obj1 = new Objeto(35, "Nome");
```

# Construtores

Se não colocarmos nenhum construtor na nossa classe, o compilador do Java irá inserir o construtor padrão em nosso código antes de gerar o bytecode, assim como outras alterações que veremos no decorrer o semestre.

**Código fonte**

**Compilador  
Java**

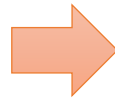
**bytecode**

O compilador processa nosso código e insere comandos na nossa classe antes de criar o bytecode.

# Construtores

Se não tem construtor, o Java insere um.

```
public class Pessoa {  
    String nome;  
    int idade;  
    double renda;  
}
```



```
public class Pessoa extends  
Object{  
    String nome;  
    int idade;  
    double renda;  
  
    Pessoa(){  
  
    }  
}
```



# Exemplo

Construtor padrão

Construtor com  
parâmetros

Pessoa
nome: String idade: int renda: double
Pessoa() Pessoa(n: String, i: int, r: double)



# Exemplo

```
public class Pessoa {
    String nome;
    int idade;
    double renda;

    Pessoa() {
        System.out.println("O construtor foi acionado");
    }
}
```

O construtor obrigatoriamente possui o mesmo nome da classe

```
public class UsaPessoa {
    public static void main(String[] args) {
        Pessoa p = new Pessoa();
    }
}
```

O construtor é acionado somente quando a classe é instanciada

```
-----Configuracao
O construtor foi acionado
```

A mensagem é exibida porque o construtor **Pessoa** foi acionado na instanciacao da classe

# Construtores – O que está errado?

```
public class UsaPessoa {  
    public static void main(String[] args) {  
        Pessoa p = new Pessoa();  
        p.Pessoa();  
    }  
}
```

Gera erro chamar um construtor como se ele fosse um método

```
public class Pessoa {  
    String nome;  
    int idade;  
    double renda;  
    void Pessoa() {  
        System.out.println("O construtor foi acionado");  
    }  
}
```

Colocar um tipo de retorno no construtor, implica em alterar a assinatura dele e com isso, ele não será acionado quando a classe for instanciada



# Construtores com parâmetros / Argumentos

Alterando o construtor para inicializar os atributos da classe

```
public class Pessoa {  
    String nome;  
    int idade;  
    double renda;  
  
    Pessoa(String n, int i, double r) {  
        nome=n;  
        idade=i;  
        renda=r;  
    }  
}
```

```
public class UsaPessoa {  
    public static void main(String[] args) {  
        Pessoa p = new Pessoa("Fulano", 40,1000.0);  
  
        System.out.println("Nome: "+p.nome);  
        System.out.println("Idade: "+p.idade);  
        System.out.println("Renda: "+p.renda);  
    }  
}
```

Um construtor pode ser alterado para receber tantos parâmetros quanto o programador quiser, no entanto, na instanciação da classe têm que ser passadas estas informações

```
-----  
Nome: Fulano  
Idade: 40  
Renda: 1000.0  
  
Process completed.
```

# Construtores com parâmetros / Argumentos

- O que é errado ao ter um construtor que recebe parâmetros

```
public class UsaPessoa {  
    public static void main(String[] args) {  
        Pessoa p = new Pessoa();  
    }  
}
```

Gera erro: criar uma instância da classe sem passar os argumentos requeridos

```
public class UsaPessoa {  
    public static void main(String[] args) {  
        Pessoa p = new Pessoa("Fulano");  
    }  
}
```

Gera erro: criar uma instância da classe sem passar todos os argumentos requeridos

```
public class UsaPessoa {  
    public static void main(String[] args) {  
        Pessoa p = new Pessoa("Fulano", 12.0, 1000.0);  
    }  
}
```

Gera erro: criar uma instância da classe e passar argumentos de tipos incompatíveis, no exemplo, float no lugar de int

# Classes com mais de um construtor (sobrecarga)

```
public class Pessoa {  
    String nome;  
    int idade;  
    double renda;  
  
    Pessoa(){  
        System.out.println("Construtor acionado");  
    }  
  
    Pessoa(String n, int i, double r) {  
        nome=n;  
        idade=i;  
        renda=r;  
    }  
}
```

Uma classe pode ter mais de um construtor, desde que, exista diferença na quantidade e tipos de parâmetros recebidos.

Neste exemplo, um construtor não recebe nenhum **parâmetro** e o outro recebe três, logo, o sistema pode diferenciar um do outro no momento da instanciação

Todos os construtores obrigatoriamente tem de ter o mesmo nome da classe e não ter tipo de retorno

# Classes com mais de um construtor

```
public class UsaPessoa {
    public static void main(String[] args) {

        Pessoa p = new Pessoa("Fulano", 40, 1000.0);

        System.out.println("Nome: "+p.nome);
        System.out.println("Idade: "+p.idade);
        System.out.println("Renda: "+p.renda);

        Pessoa p1 = new Pessoa();

        System.out.println("\nNome: "+p1.nome);
        System.out.println("Idade: "+p1.idade);
        System.out.println("Renda: "+p1.renda);
    }
}
```

A classe pode ser instanciada de duas maneiras, uma não passando nenhum **parâmetro** e outra passando três (nesta ordem: String, int e double)

```
-----
Nome: Fulano
Idade: 40
Renda: 1000.0
Construtor acionado

Nome: null
Idade: 0
Renda: 0.0

Process completed.
```

Em cada uma das maneiras de instanciar é chamado o construtor correspondente

# Exemplos

Triangulo
base: float altura: float
Triangulo() Triangulo(b: float, a: float)

```
public class Triangulo {  
    float base;  
    float altura;  
  
    Triangulo(){  
  
    }  
  
    Triangulo(float b, float a){  
        base = b;  
        altura = a;  
    }  
}
```

Data
dia: int mes: int ano: int
Data() Data(d: int, m: int, a: int)

```
public class Data {  
    int dia;  
    int mes;  
    int ano;  
  
    Data(){ }  
    Data(int d, int m, int a){  
        dia = d;  
        mes = m;  
        ano = a;  
    }  
}
```

# Usando a classe

```
public class TesteClasses {  
  
    public static void main(String[] args) {  
        Triangulo t = new Triangulo(2.5f, 3f);  
        float area = (t.base * t.altura)/2;  
        System.out.println("Área: " + area);  
  
        Data d1 = new Data(2, 9, 2015);  
        Data d2 = new Data();  
        System.out.println(d1.dia+"/"+d1.mes+"/"+d1.ano);  
        System.out.println(d2.dia+"/"+d2.mes+"/"+d2.ano);  
    }  
}
```

# Outro Exemplo

ContaCorrente
nome: string saldo: float limite: float tipo: char
ContaCorrente(n: string, s: float, l: float, t: char) ContaCorrente(n: string, s: float, t: char)

Declare dois objetos da classe ContaCorrente, solicite os dados para o usuário e instancie um objeto com o primeiro construtor, repita o processo para instanciar o segundo objeto com o segundo construtor

# Outro Exemplo

```
public class ContaCorrente {  
    String nome;  
    float saldo;  
    float limite;  
    char tipo;  
  
    ContaCorrente(String n, float s, float l, char t){  
        nome = n;  
        saldo = s;  
        limite = l;  
        tipo = t;  
    }  
  
    ContaCorrente(String n, float s, char t){  
        nome = n;  
        saldo = s;  
        tipo=t;  
    }  
}
```



## Exercícios

- 1) Considere os diagramas UML dos seguintes itens, observe que cada item possui 2 construtores distintos. Construa o código das respectivas classes e crie 2 objetos usando os construtores criados e os preencha com dados fornecidos pelo usuário. Exiba os dados dos objetos.

Paciente
nome: string rg: string endereco: string telefone: string dataNascimento: string profissao: string
Paciente() Paciente(nome: string)

## Exercícios

- 2) Considere os diagramas UML dos seguintes itens, observe que cada item possui 2 construtores distintos. Construa o código das respectivas classes e crie 2 objetos usando os construtores criados e os preencha com dados fornecidos pelo usuário. Exiba os dados dos objetos.

Produto
marca: string fabricante: string cod_barras: string preco: float
Produto() Produto (m: string, f:string, c:string, p:float)

“Saber muito não lhe torna  
inteligente.

A inteligência se traduz na  
forma que você reconhece,  
julga, maneja e, sobretudo,  
onde e como aplica esta  
informação”



**Carl Sagan**

# Obrigado!

**Se precisar ...**

Prof. Claudio Benossi

**[Claudio.benossi@fatec.sp.gov.br](mailto:Claudio.benossi@fatec.sp.gov.br)**

