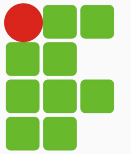
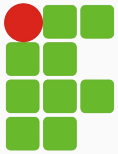


Programação orientada a objetos

Maximilian Jaderson de Melo

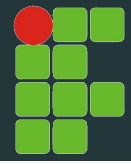
Aula 2

Adaptado do material do prof. Rafael Liberato (UTFPR-CM)

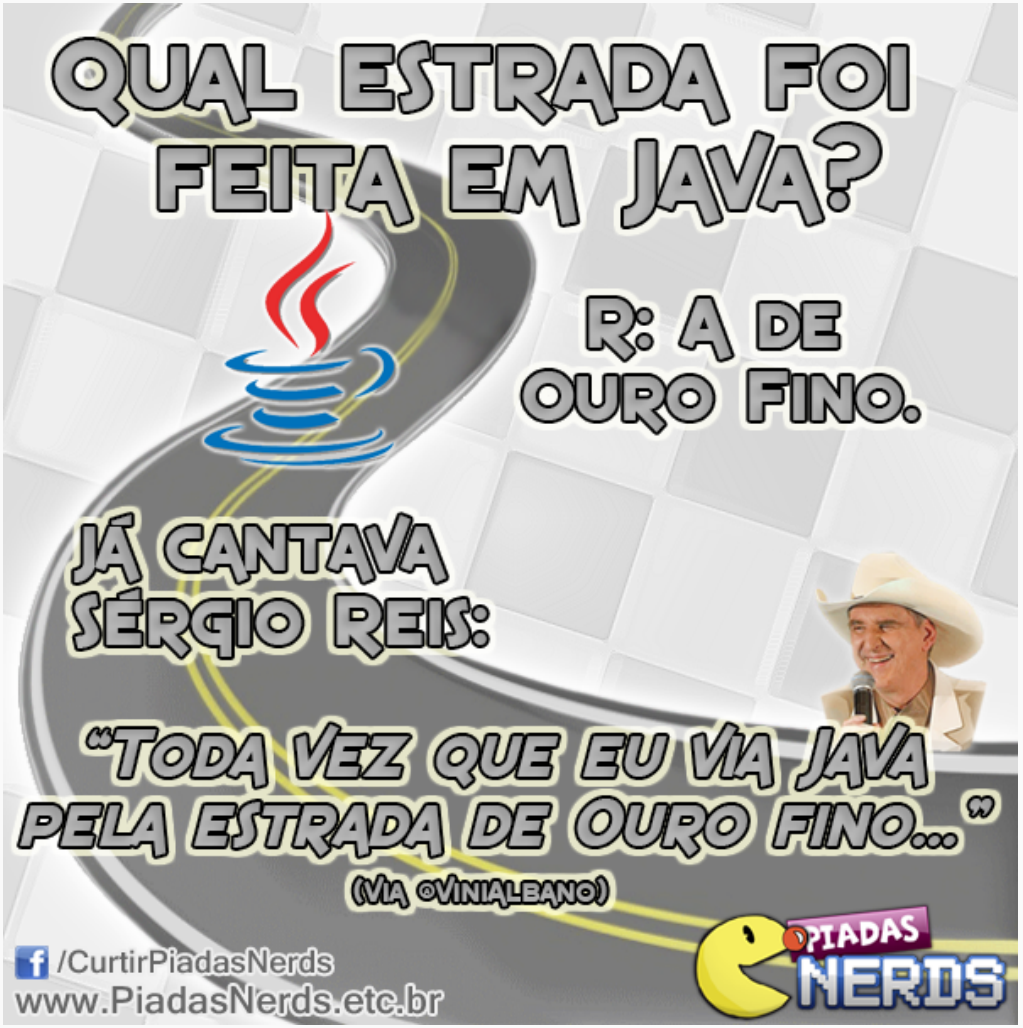


Introdução ao *Java* e ao *Netbeans*

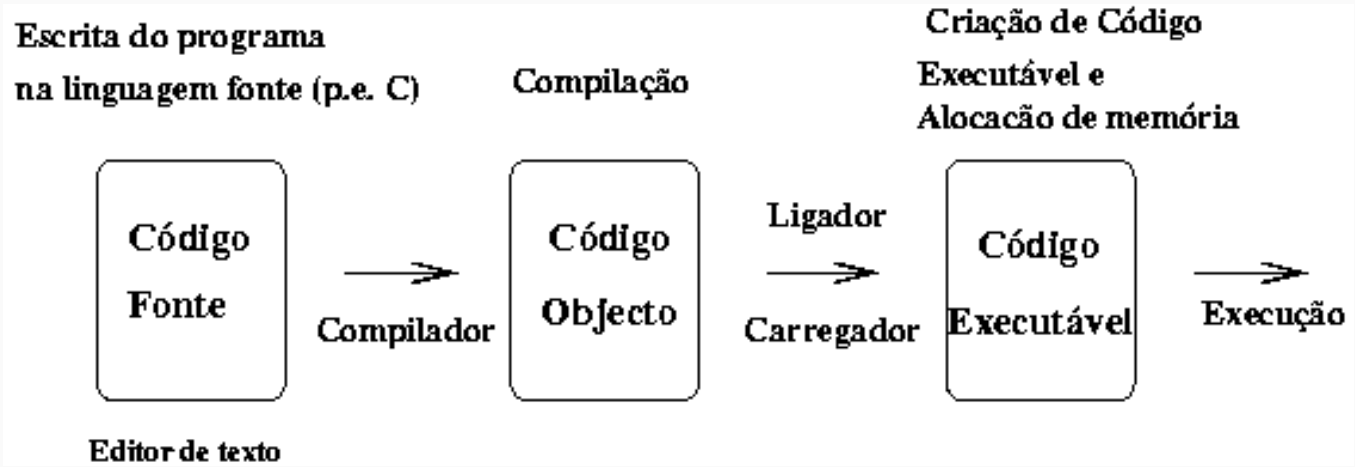
Conteúdo de hoje



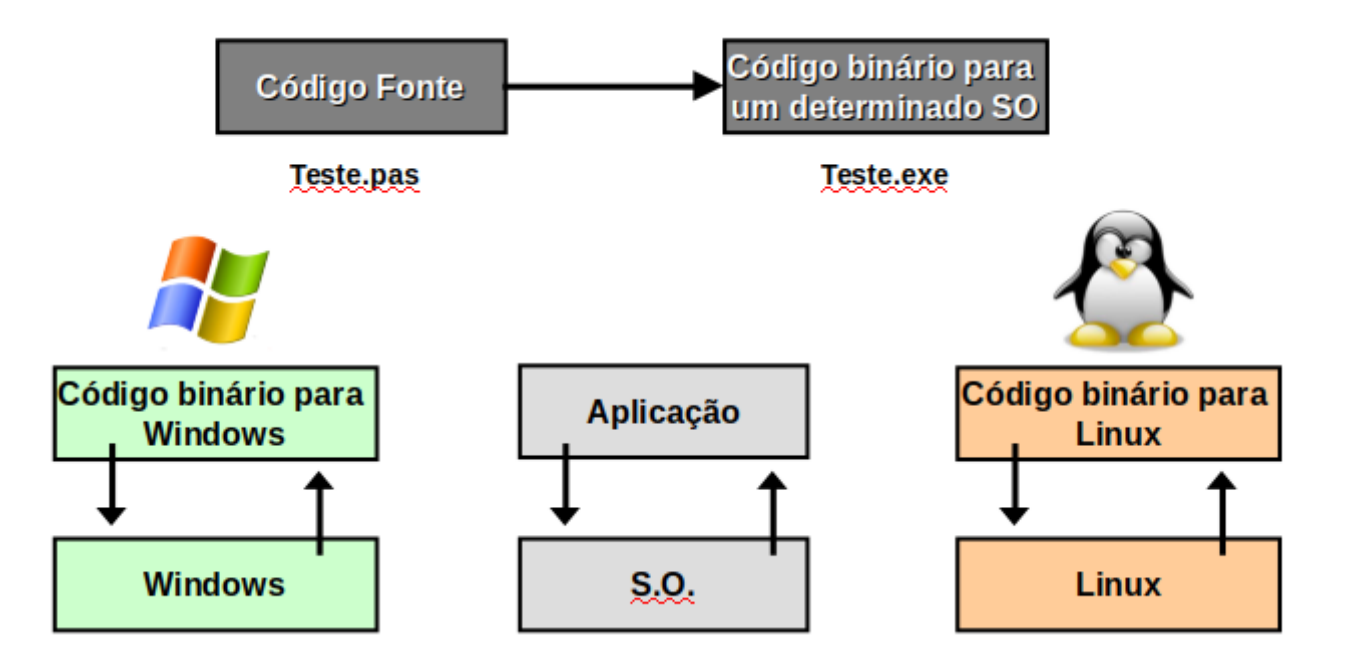
- Java.
  - Linguagens compiladas vs interpretadas.
  - Tecnologias em *Java*.
- Netbeans.
  - Uso básico da IDE.


QUAL ESTRADA FOI FEITA EM JAVA?
R: A DE OURO FINO.
JÁ CANTAVA SÉRGIO REIS:
"TODA VEZ QUE EU VIA JAVA PELA ESTRADA DE OURO FINO..."
(VIA @VINIALBANO)
/CurtirPiadasNerds
www.PiadasNerds.etc.br
PIADAS NERDS

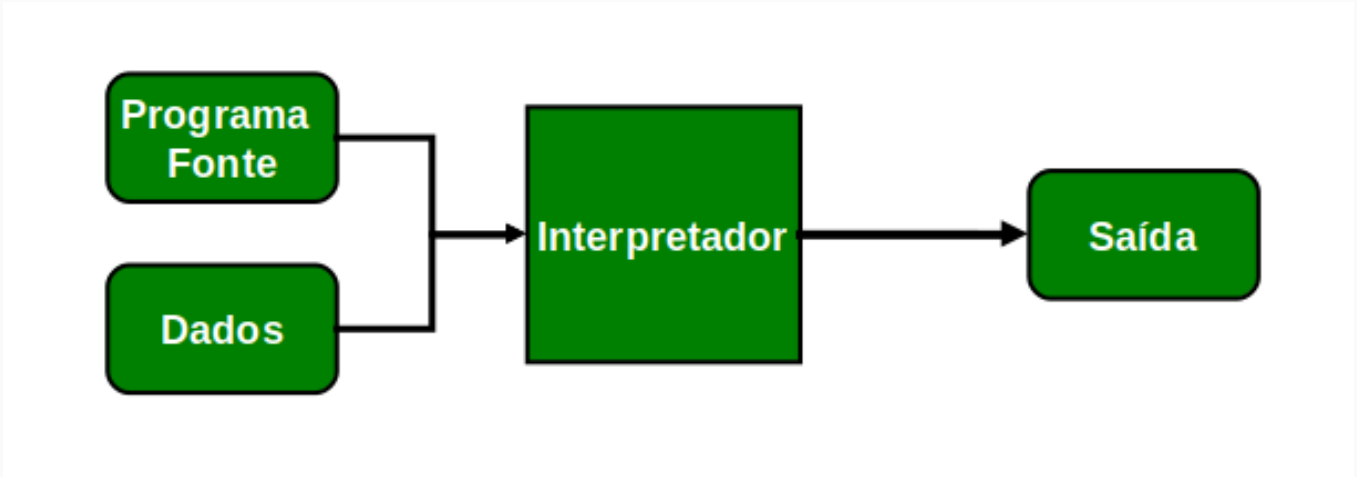
Compilação


Escrita do programa
na linguagem fonte (p.e. C)
Compilação
Criação de Código
Executável e
Alocacão de memória
Código
Fonte
Compilador
Código
Objecto
Ligador
Carregador
Código
Executável
Execução
Editor de texto

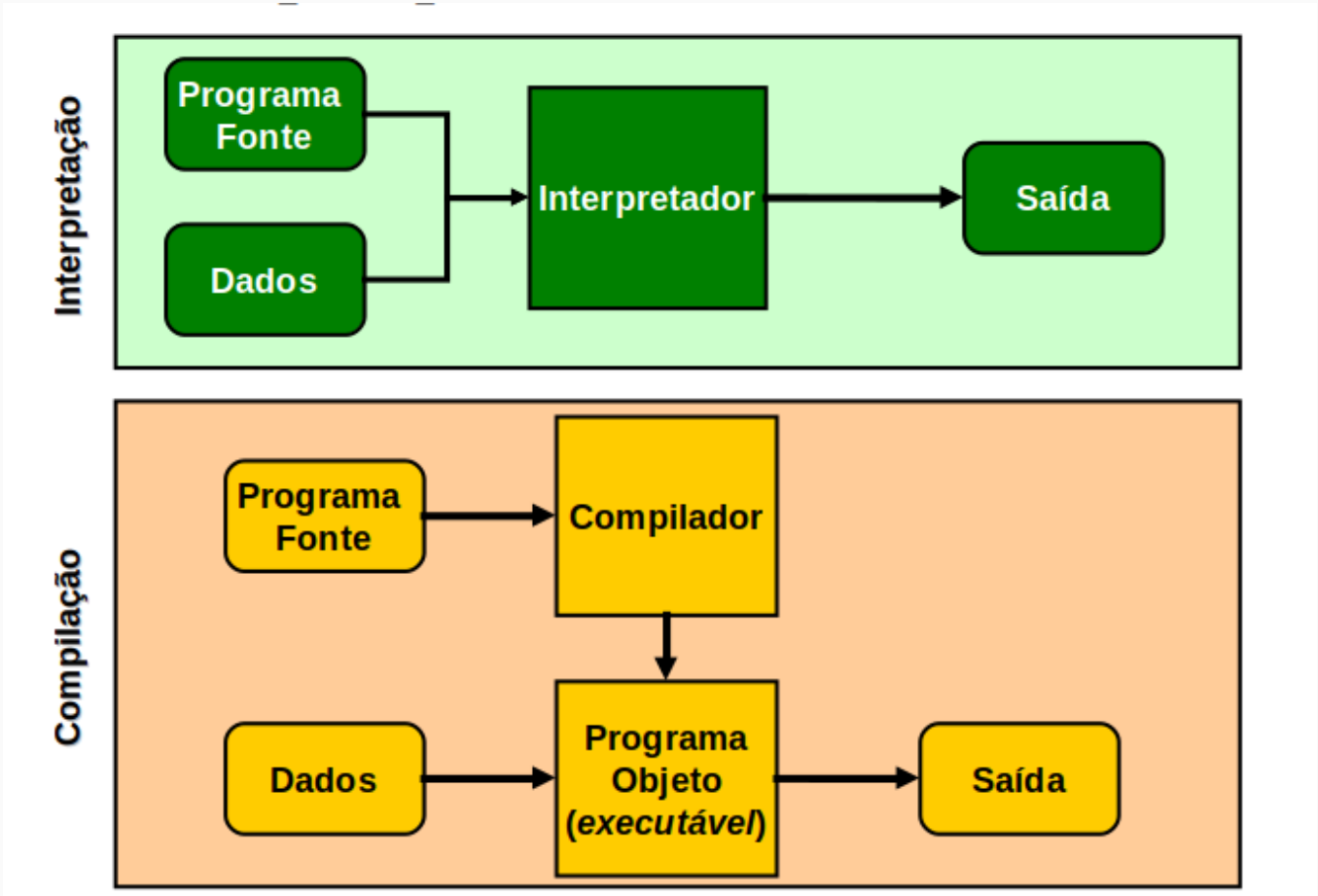
Compilação


Código Fonte
Código binário para um determinado SO
Teste.pas
Teste.exe
Código binário para Windows
Aplicação
Código binário para Linux
Windows
S.O.
Linux

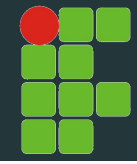
Interpretação



Compilação x Interpretação

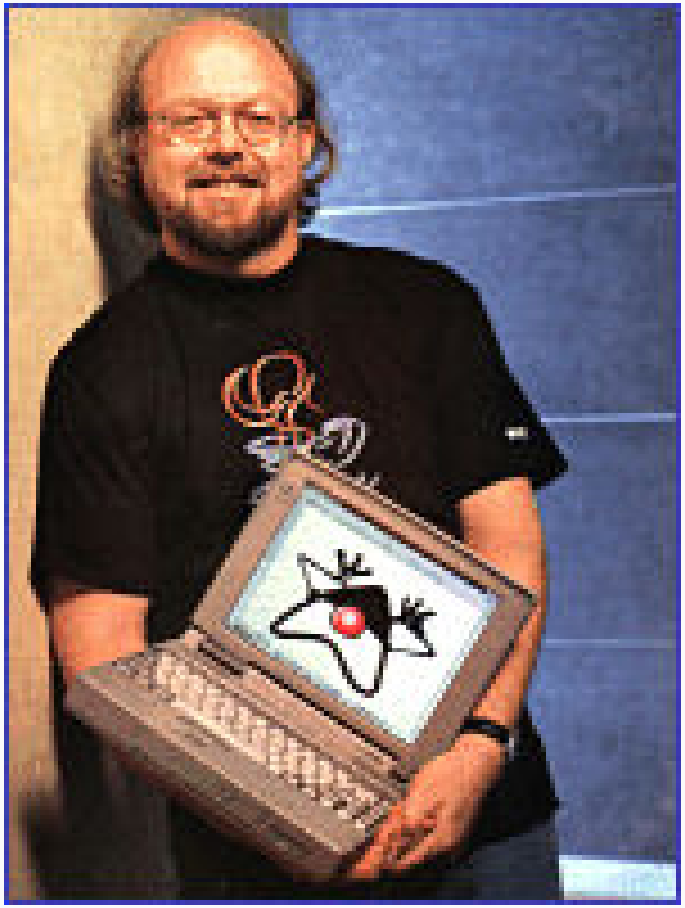


Como tudo começou...

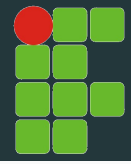


- Em 1991 Sun financiou um projeto de pesquisa interno de codinome **Green**, chefiado por James Gosling e que criou uma linguagem baseada em C e C++ chamada Oak (carvalho) para dispositivos eletrônicos inteligentes.
- Como o projeto não alcançou os resultados de mercado esperados, e em 1993 a World Wide Web explodiu em popularidade nos EUA, a Sun resolveu adaptar o projeto para ser usado no desenvolvimento de conteúdo dinâmico para web.

Como tudo começou...

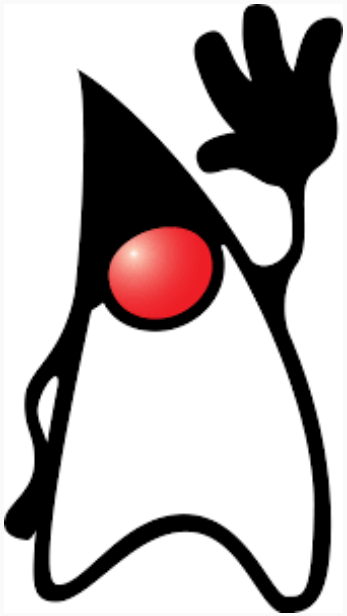


Como tudo começou...

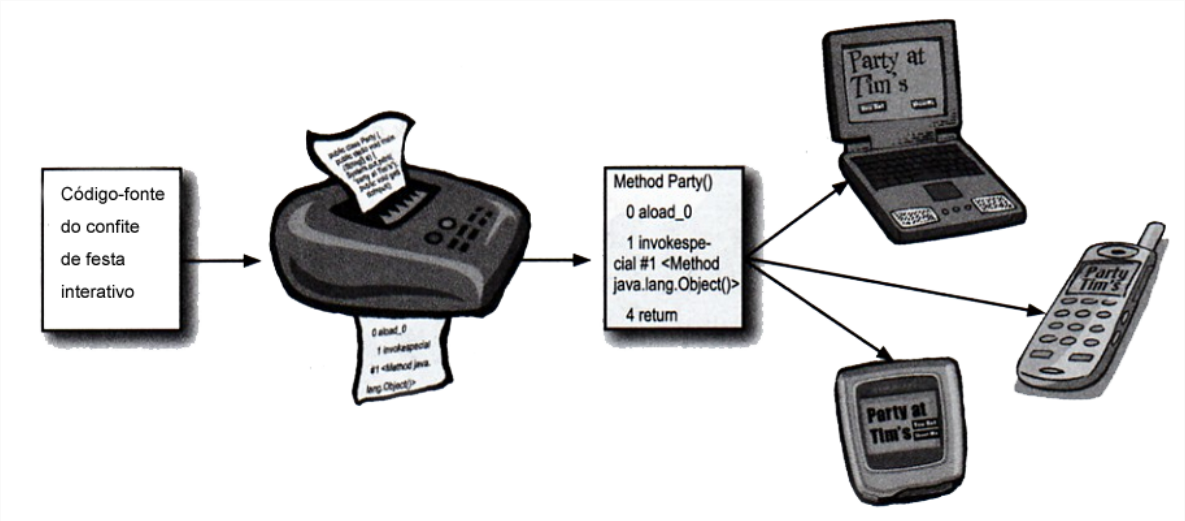


- Como já existia uma linguagem chamada *Oak* o nome foi trocado por *Java* (em homenagem a um tipo de café importado que a equipe tomava numa cafeteria perto da empresa)
- Em 23 de maio de 1995 a Sun apresentou oficialmente a linguagem *Java* ao mercado, na conferência SunWorld. E o seu mascote *duke*.

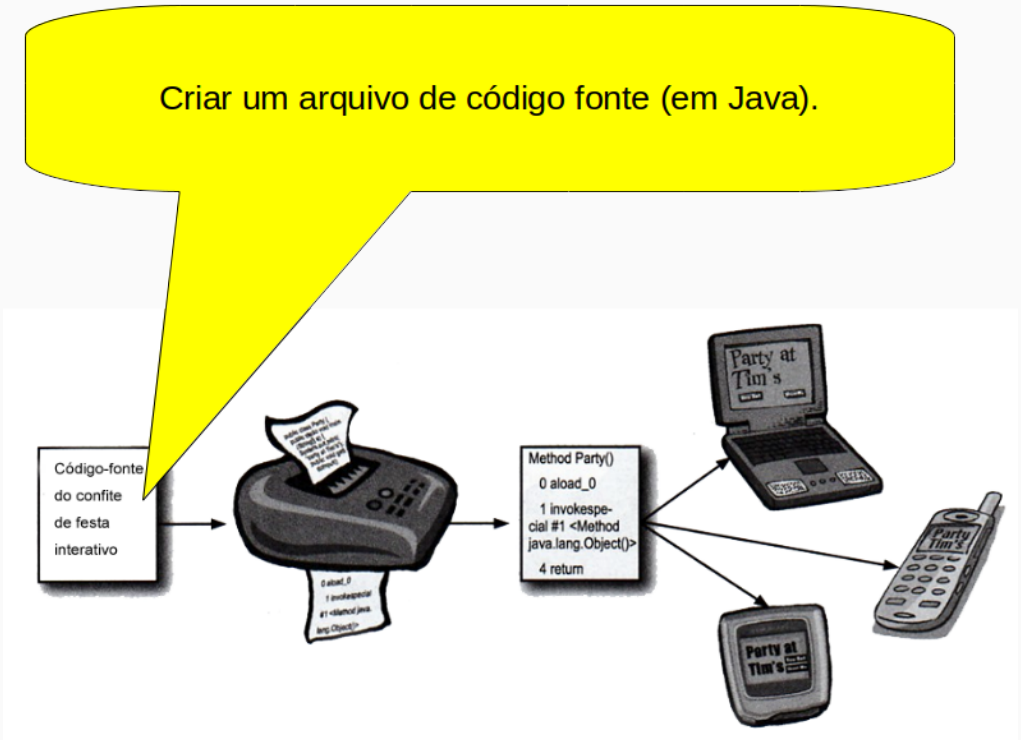
Como tudo começou...



Como o Java funciona

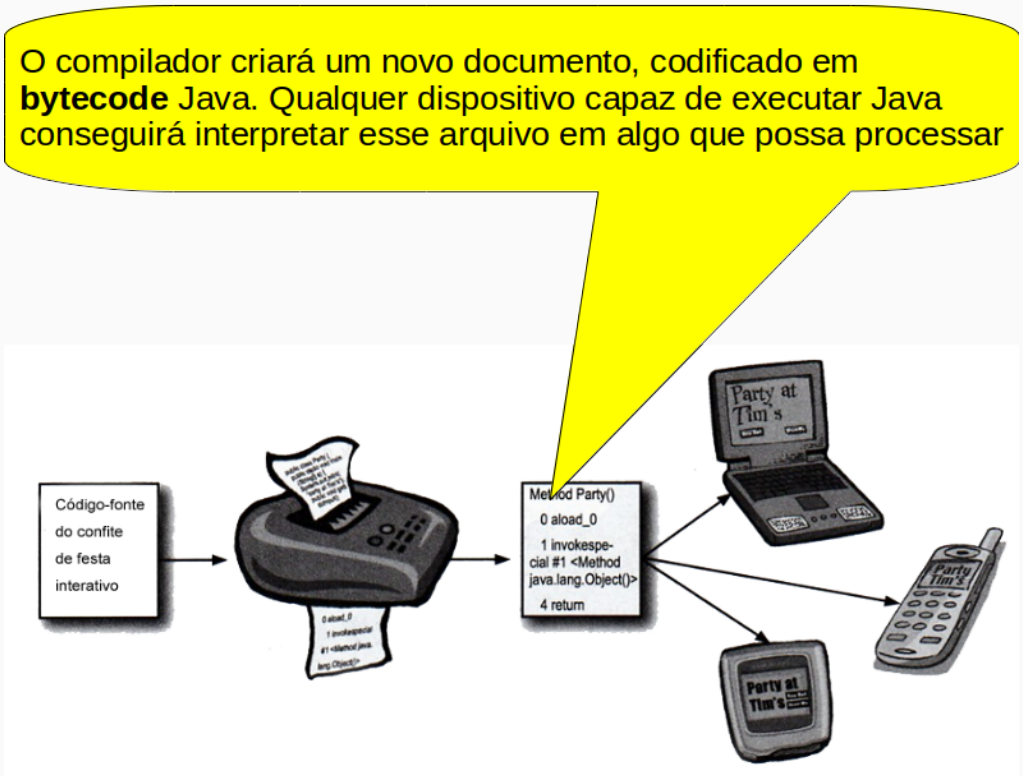


Como o Java funciona

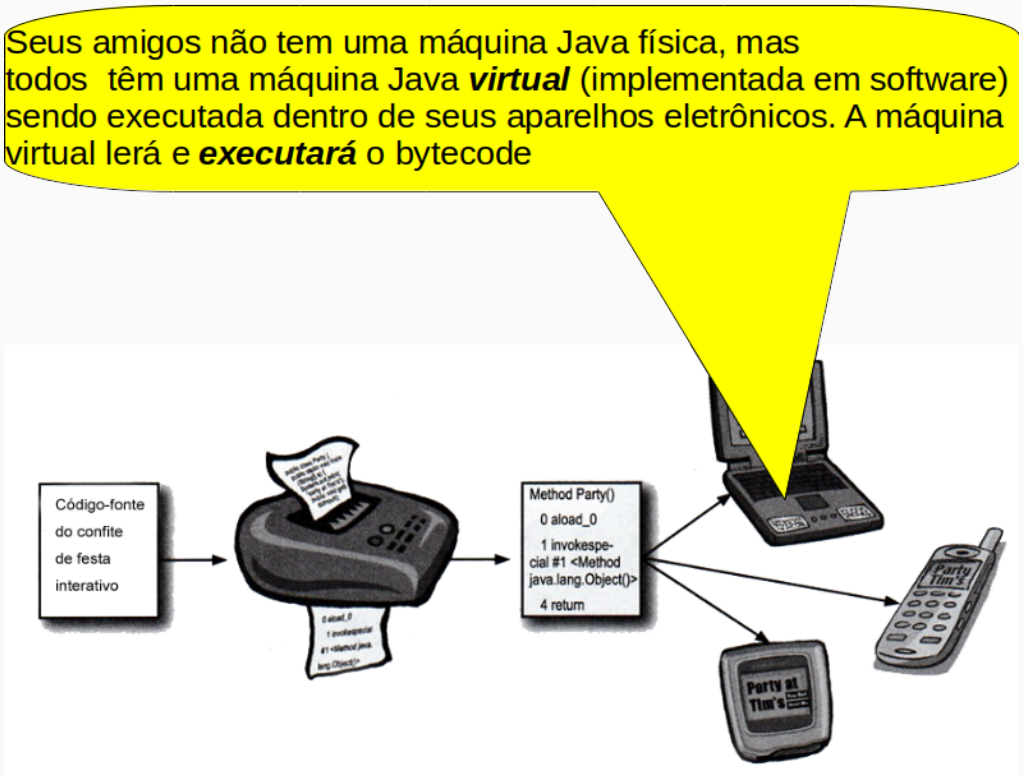


Como o Java funciona

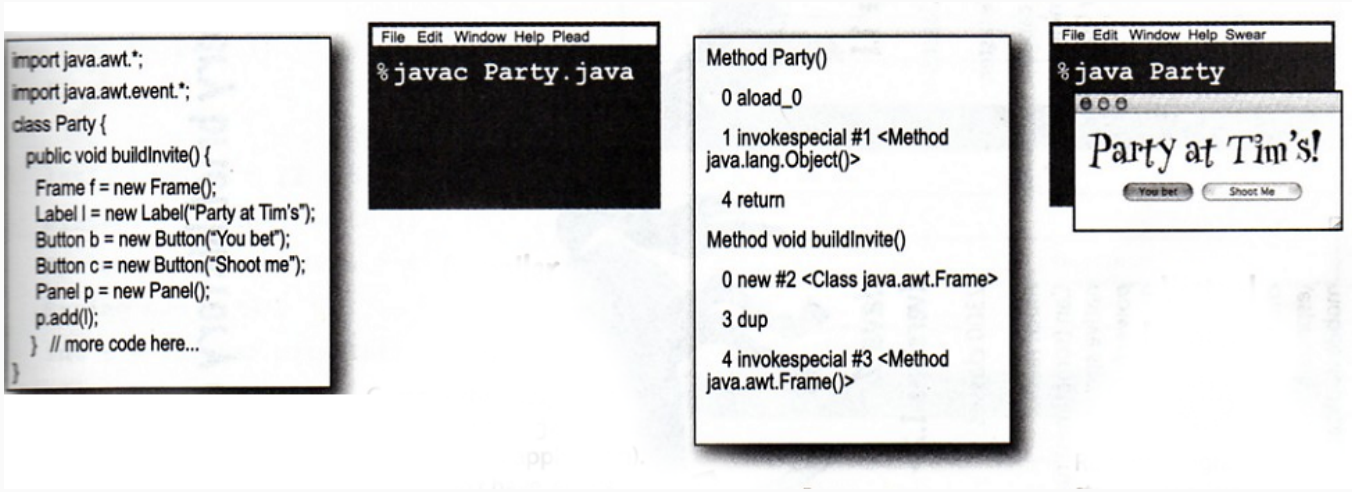
Como o Java funciona



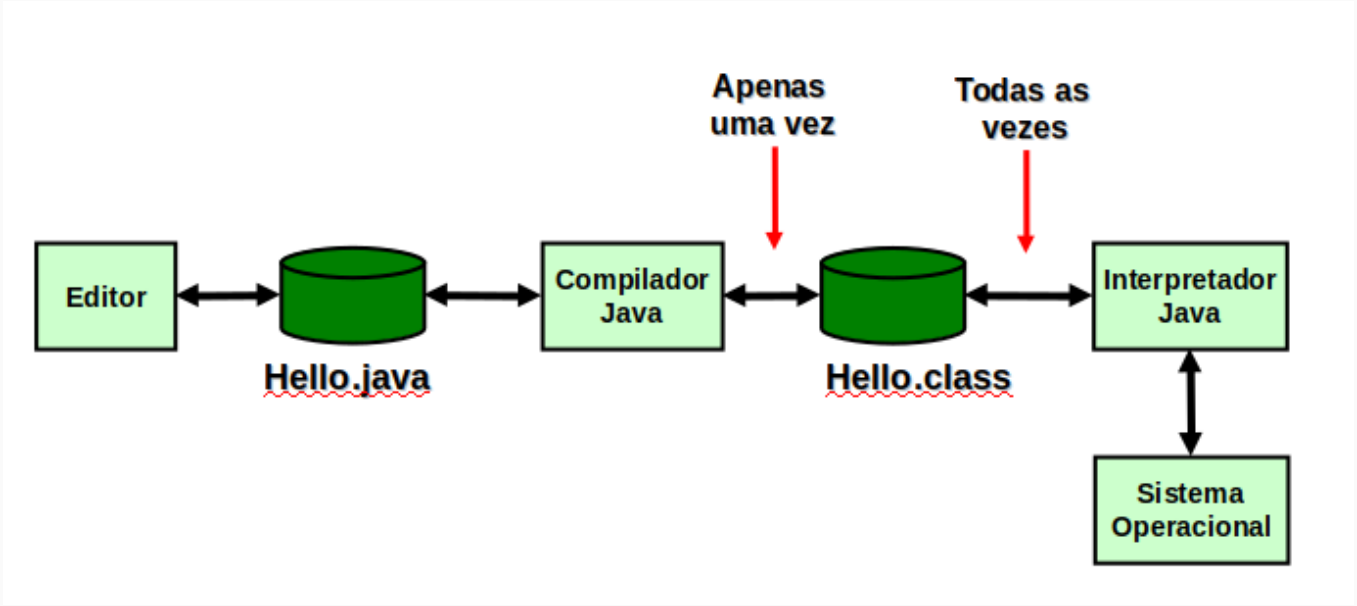
Como o Java funciona



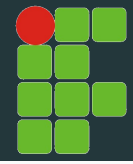
Como o Java funciona



Fases de um programa em Java



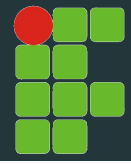
Características de Java



- Máquina virtual.
  - Interpretador.
- Checagem de erros em tempo de execução.
- *Garbage Collector*.
  - Libera memória alocada automaticamente.
- A *VM* torna *Java* independente de plataforma.
- *Write once, run anywhere*.

Plataformas

Plataformas



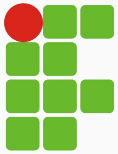
- *Java Standard Edition (SE)*: destinada ao desenvolvimento de aplicações *Desktop*;
- *Java Enterprise Edition (EE)*: destinada ao desenvolvimento de aplicações *WEB*, para servidores.
- *Java Micro Edition (ME)*: destinada ao desenvolvimento de aplicações com pouca memória, como celulares, PDA's, etc.
- *Java Card*: destinada ao desenvolvimento de aplicações com MUITO pouca memória, como cartões inteligentes.

Kits

- *Java Development Kit (JDK)*: kit de desenvolvimento *Java*, com compilador e *VM*, além de outras ferramentas.
- *Java Runtime Environment (JRE)*: conjunto de bibliotecas e interpretador (*VM*) para a execução do *Java*.
- *Java Documentation (Javadocs)*: documentação e ferramentas de criação de documentação padronizada e robusta. Exporta em formato *HTML*.

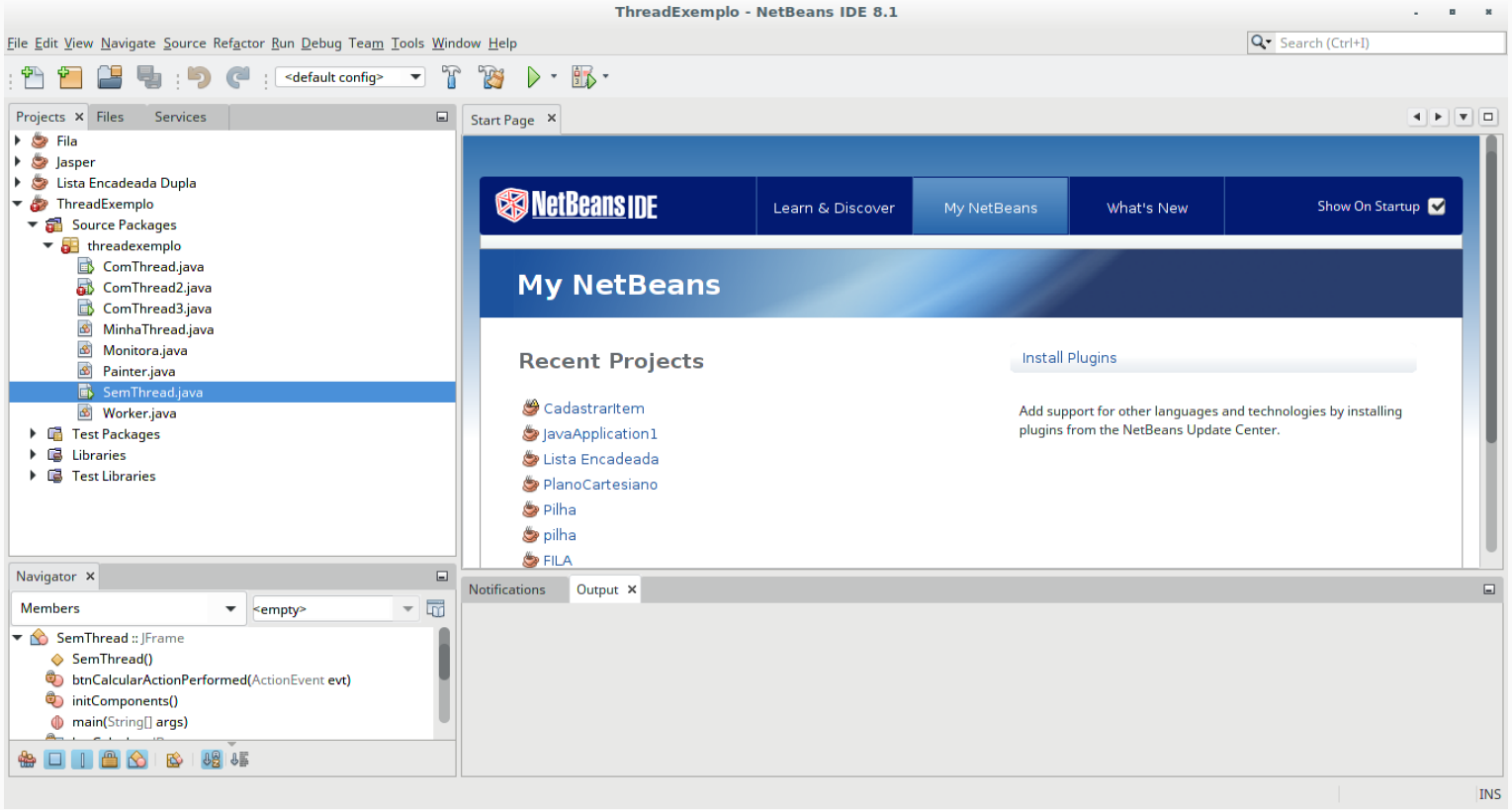
Tipos de arquivos

- .java: códigos fonte.
- .class: *bytecode*, o executável java.
- .jar (*Java Archive*): agrupamento de classes executáveis. Semelhante a um arquivo ZIP.
- .jad (*Java Archive Descriptor*): arquivo que descreve o conteúdo de um .jar.

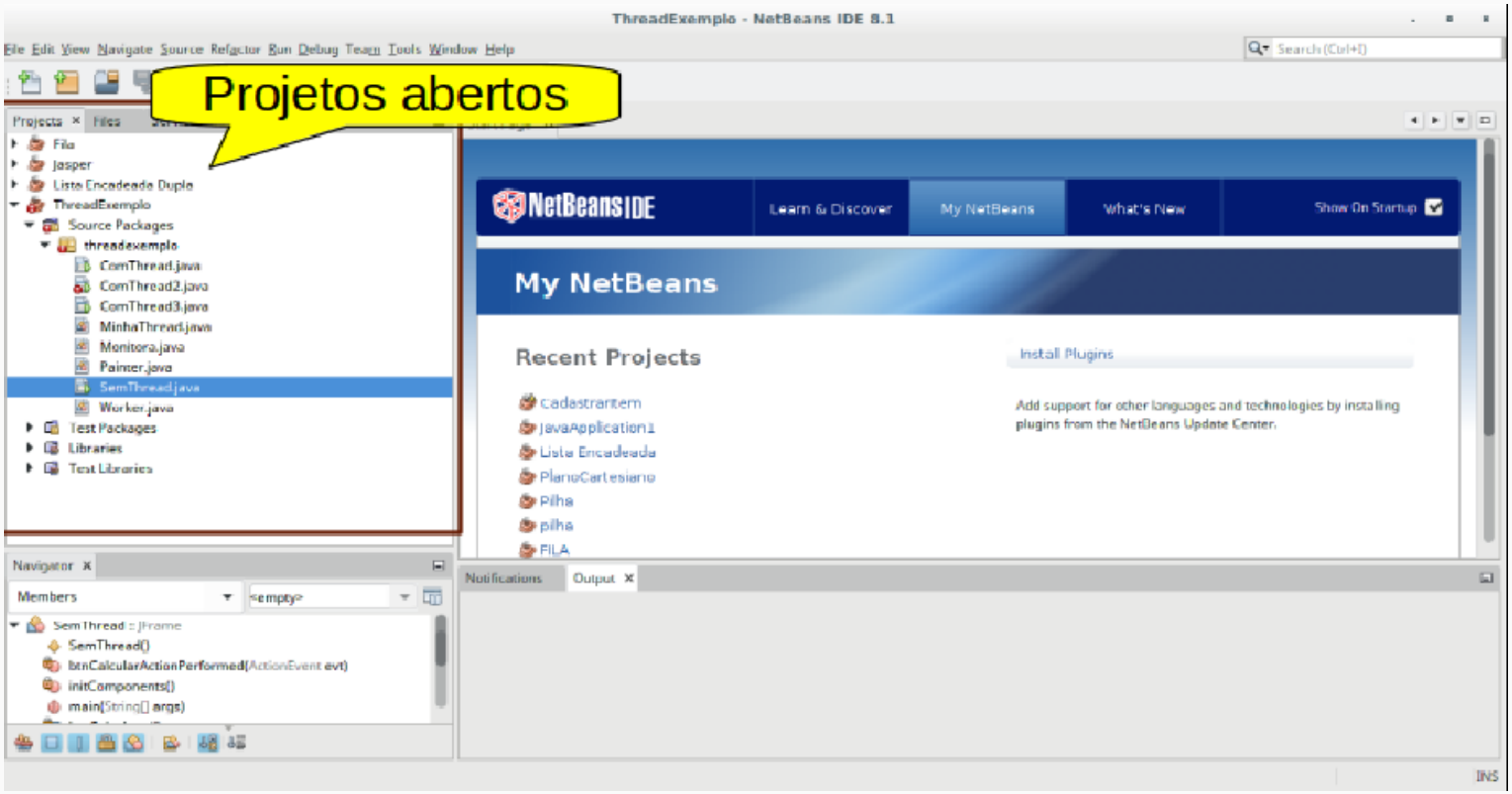


Netbeans

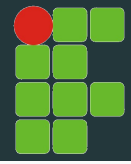
Netbeans IDE



Netbeans IDE

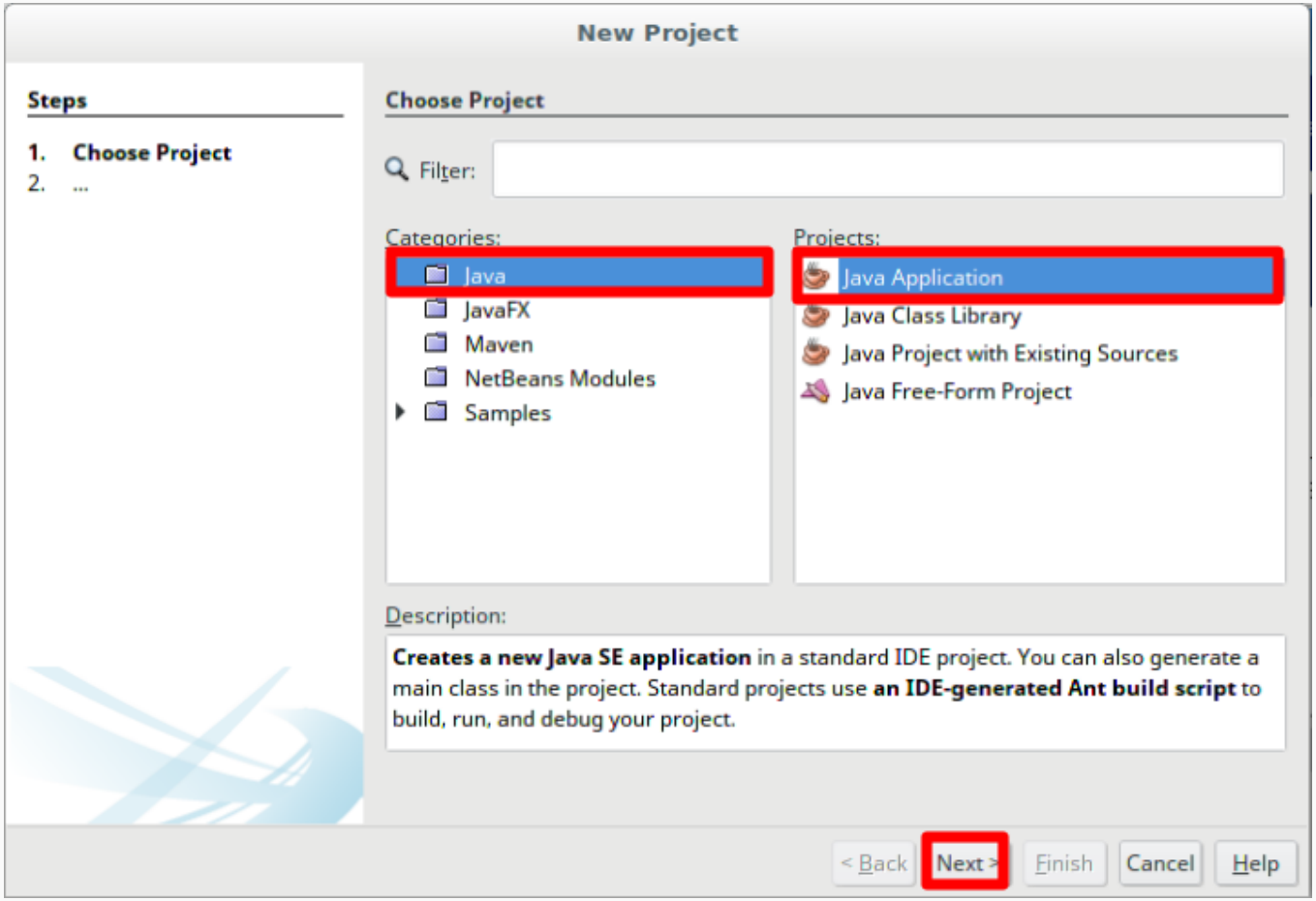


Netbeans IDE

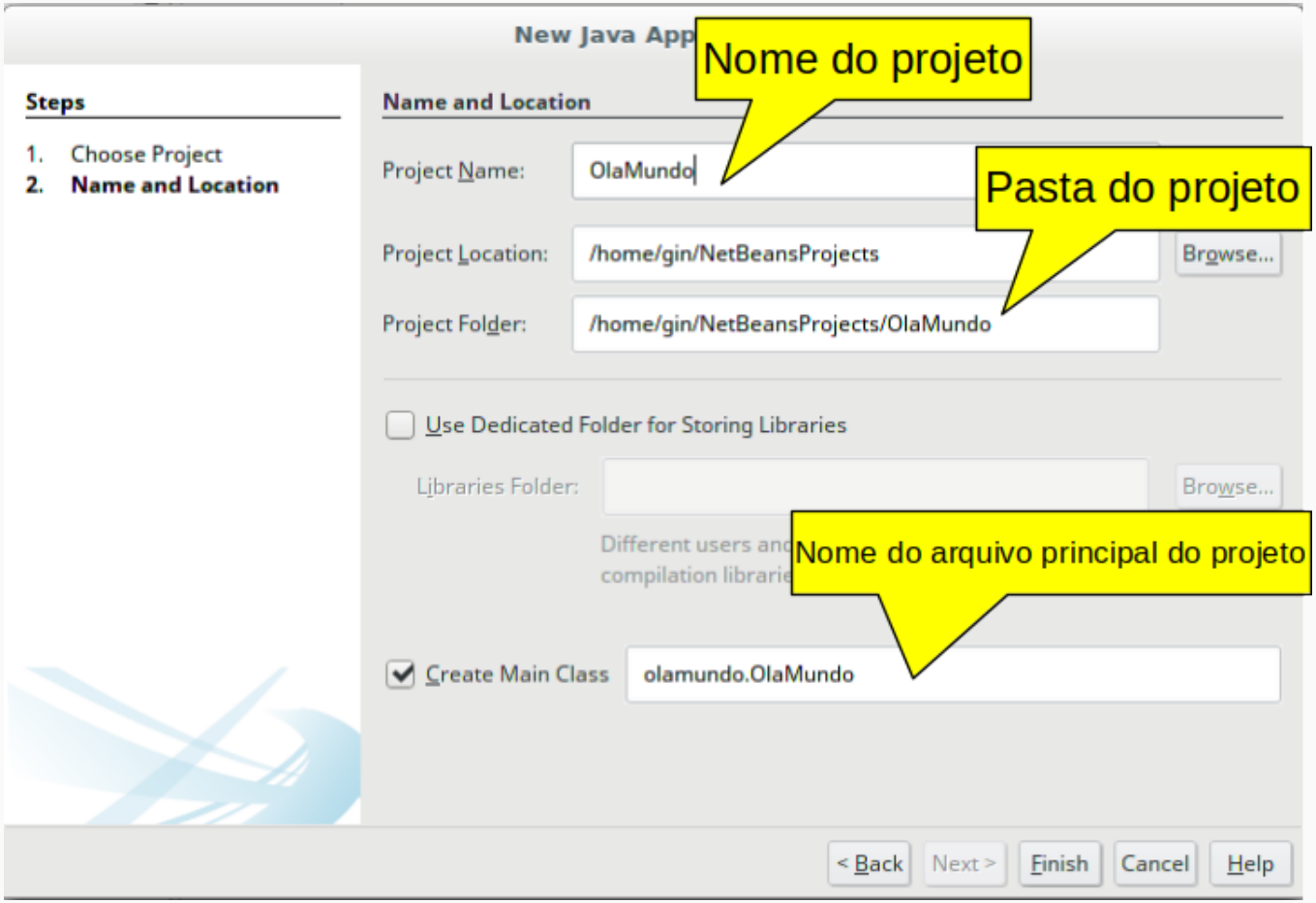


- Para criar um novo projeto vá em:
  - Arquivo $>$ Novo projeto (CTRL + SHIFT + N)

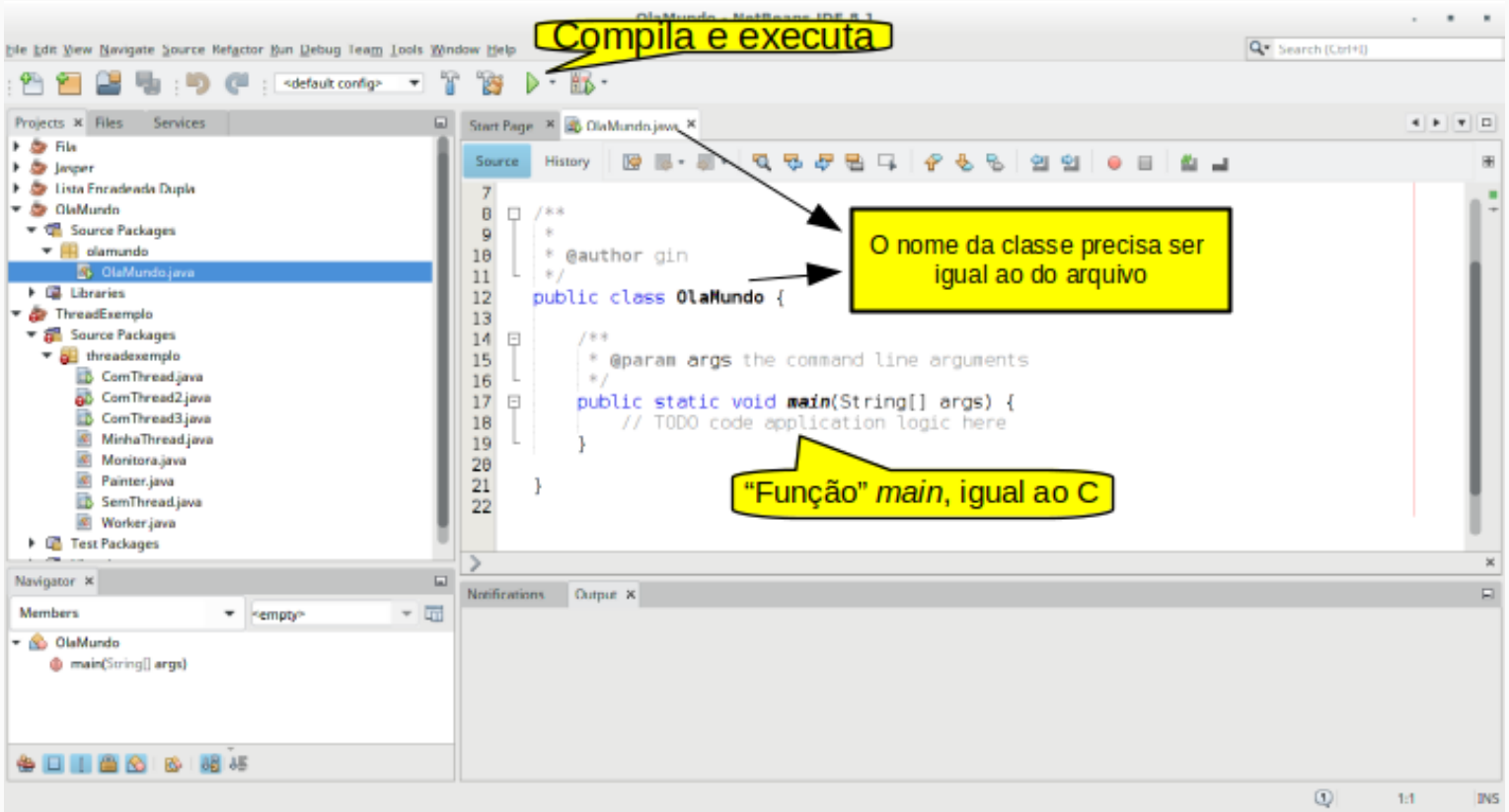
Netbeans IDE

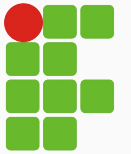


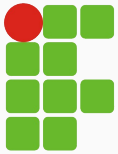
Netbeans IDE



Netbeans IDE

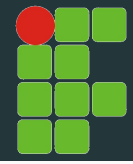






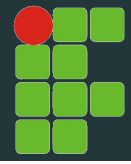
Entrada e saída de dados e tipos

Canais de entrada e saída



- System.out: saída padrão de fluxo de dados.
- System.in: entrada padrão de fluxo de dados.
- System.err: saída padrão de fluxo de dados para erros.
- Os atributos in, out, err indicam objetos dos tipos PrintStream.

Uso de Strings



- Em *Java* o tipo *String* é nativo.
- Serve para armazenar qualquer sequência alfanumérica.
  - Letras e números.
  - Serve para textos.

```
String var = "Olá mundo";
```

- No exemplo acima, uma variável chamada var é declarada como *String* e possui o valor “olá mundo”.

Concatenação de Strings

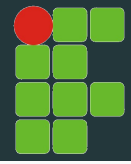
- É feita usando o operador $+$.

```
String s1 = “Java”;
String s2 = “ é uma linguagem OO”;
double real = 1.6;
```

s1 + s2 Java é uma linguagem OO

s1 + real Java 1.6

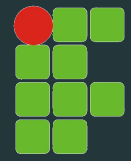
Concatenação de Strings - Exemplo



- Qual a saída?

```
public class ExemploConcatenacao {

   public static void main(String[] args) {

      String texto1 = ">> O operador de concatenação (+) ";
      String texto2 = "é muito prático ";
      String texto3 = texto1 + texto2 + "!";

      System.out.println(texto3 + " <<");
      System.out.println(">> 2 + 5 = " + 2 + 5);   // Uso errado
      System.out.println(">> 2 + 5 = " + (2 + 5)); // Uso Correto
   }
}
```

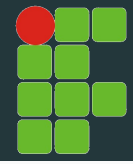
Importando classes



- As classes nativas de Java estão organizadas hierarquicamente em pacotes.
- Para usar uma funcionalidade é necessário indicar qual o pacote e a classe onde o recurso está implementado.

```
import java.io.BufferedReader;
```

Importando classes



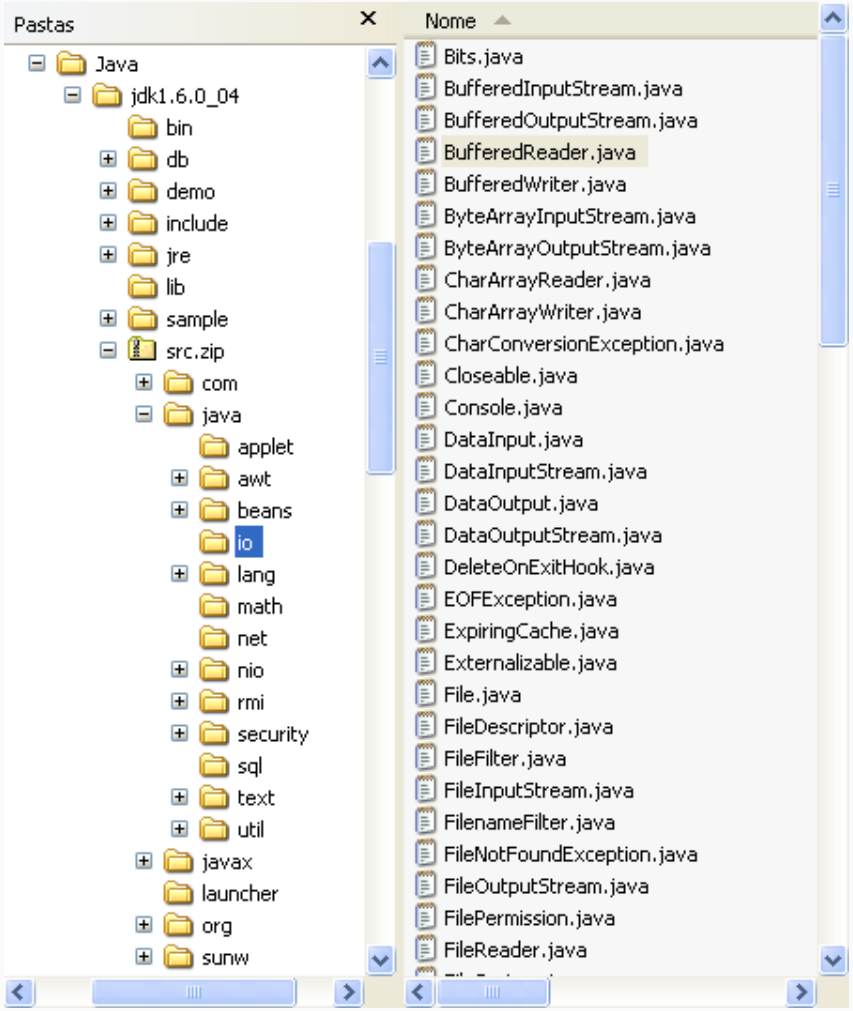
```
import pacote.classe; //importa uma classe
import pacote.*
//importa todas as classes de um pacote
```

- O segundo caso acima é semelhante a

```
using namespace std;
```

- O *Java* busca as classes a partir de uma variável de sistema chamada CLASSPATH

Importando classes

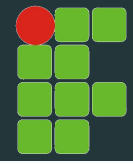


Importando classes

Leitura pelo teclado

- Scanner
- JoptionPane

Scanner



- No início do programa:

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

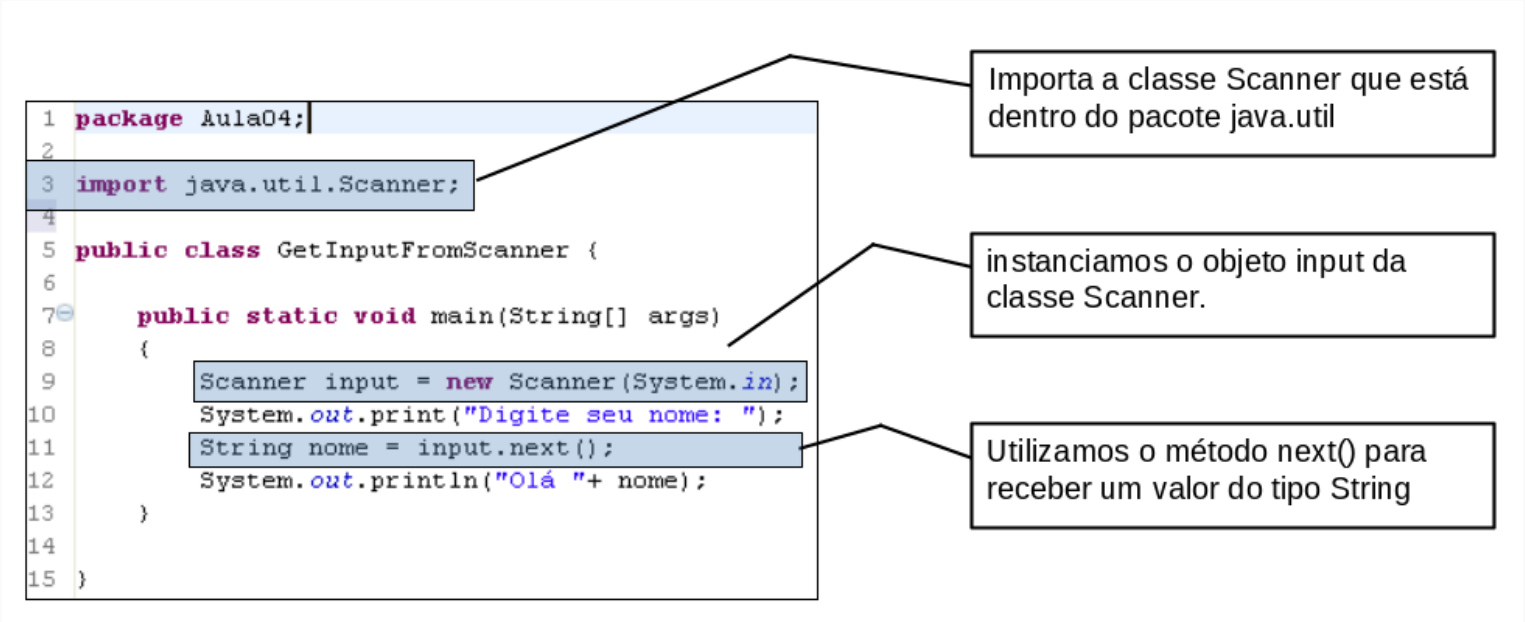
- Dentro da main:

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

Scanner

Método	Finalidade
next()	Aguarda uma entrada no formato String
nextInt()	Aguarda uma entrada no formato inteiro
nextByte()	Aguarda uma entrada no formato byte
nextShort()	Aguarda uma entrada no formato short
nextLong()	Aguarda uma entrada no formato long
nextFloat()	Aguarda uma entrada no formato float
nextDouble()	Aguarda uma entrada no formato double

Scanner


1 package Aula04;
2
3 import java.util.Scanner;
4
5 public class GetInputFromScanner {
6
7 public static void main(String[] args)
8 {
9 Scanner input = new Scanner(System.in);
10 System.out.print("Digite seu nome: ");
11 String nome = input.next();
12 System.out.println("Olá "+ nome);
13 }
14
15 }
Importa a classe Scanner que está dentro do pacote java.util
instanciamos o objeto input da classe Scanner.
Utilizamos o método next() para receber um valor do tipo String

Scanner - Exemplo

- Qual a saída?

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

    int i1;
    float f1;

    Scanner input = new Scanner(System.in);

    System.out.print("Digite um número do tipo INTEIRO: ");
    i1 = input.nextInt();

    System.out.print("Digite um número do tipo FLOAT: ");
    f1 = input.nextFloat();

    int prod = (int) (i1 * f1);

    System.out.println("A parte inteira de " + i1 +" * "+ f1 +" = "+ prod);

}
```

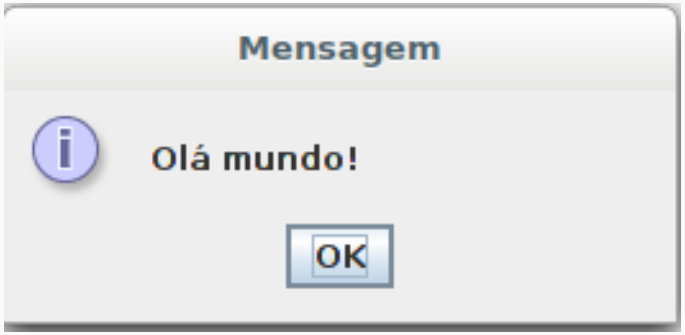
JOptionPane

- Importação:

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

- Na main:

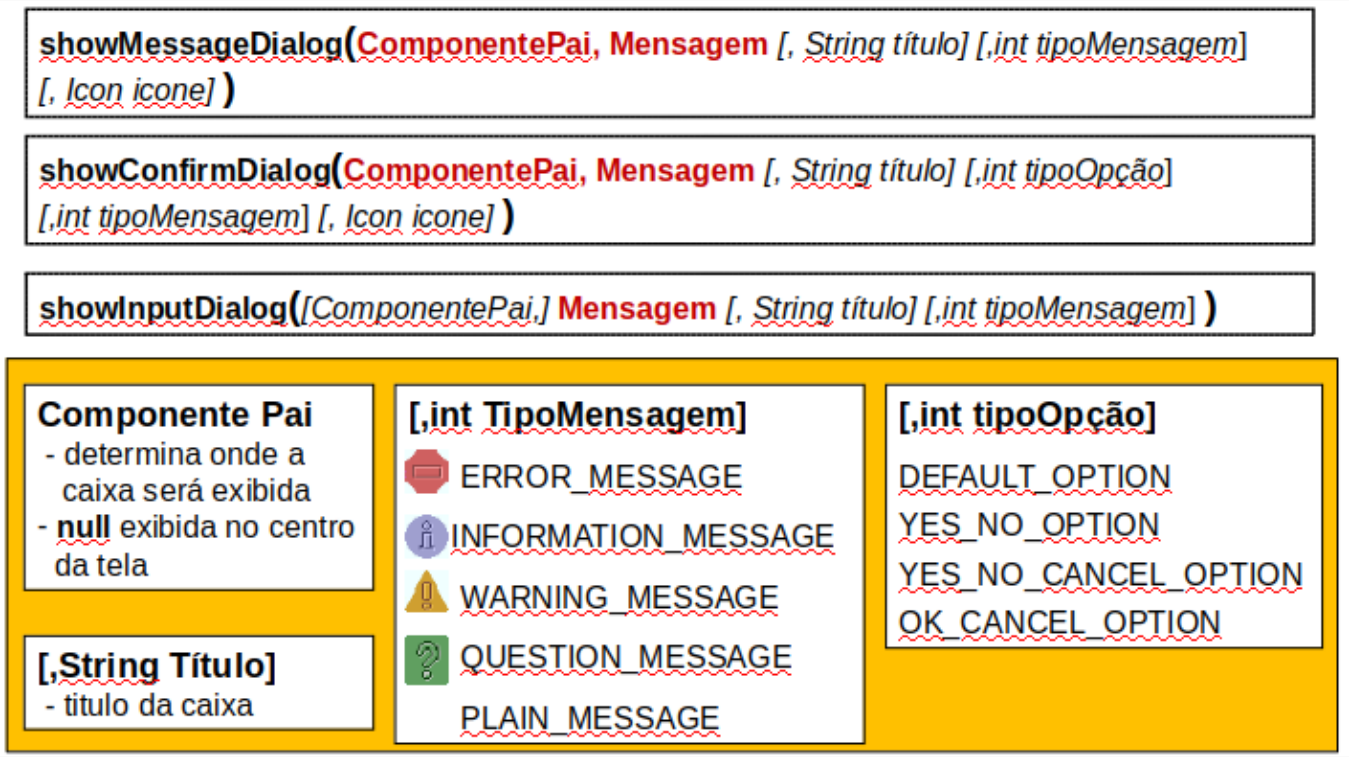
```
JOptionPane.
showMessageDialog(null, "Olá mundo!");
```



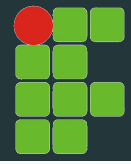
JOptionPane

Nome do método	Descrição
showConfirmDialog	Mostra uma caixa de confirmação para o usuário, no formato yes/no/cancel.
showInputDialog	Mostra uma caixa de entrada de dados, com um campo para leitura (field).
showMessageDialog	Mostra uma caixa de diálogo com uma mensagem de texto, e um botão OK para fechar.

JOptionPane



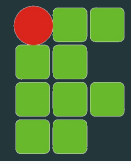
JOptionPane - Exemplo 1



- Qual a saída?

```
package aula04;

import javax.swing.JOptionPane;

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

        JOptionPane.showMessageDialog(null, "Mensagem de AVISO", "Titulo da Caixa",JOptionPane.WARNING_MESSAGE);
        JOptionPane.showMessageDialog(null, "Mensagem de ERRO", "Titulo da Caixa",JOptionPane.ERROR_MESSAGE);
        JOptionPane.showMessageDialog(null, "Mensagem de INFORMATIVA", "Titulo da Caixa",JOptionPane.INFORMATION_MESSAGE);
        JOptionPane.showMessageDialog(null, "Mensagem de INTERROGATIVA", "Titulo da Caixa",JOptionPane.QUESTION_MESSAGE);
        JOptionPane.showMessageDialog(null, "Mensagem SIMPLES", "Titulo da Caixa",JOptionPane.PLAIN_MESSAGE);

        System.exit(0);
    }
}
```

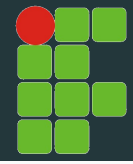
JOptionPane - Exemplo 2



- Qual a saída?

```
public static void main(String[] args) {
    String nome = JOptionPane.showInputDialog(null, "Digite seu nome", "Nome completo", JOptionPane.INFORMATION_MESSAGE );

    String escola = JOptionPane.showInputDialog(null, "Digite a escola onde estuda", "Nome da escola", JOptionPane.PLAIN_MESSAGE);

    String aux = JOptionPane.showInputDialog(null, "Digite o período atual", "Período", JOptionPane.QUESTION_MESSAGE);

    int periodo = Integer.parseInt(aux);

    JOptionPane.showMessageDialog(null, "Nome: "+ nome+"\nEscola: "+escola+"\nPeríodo: "+periodo,
            "Valores informados", JOptionPane.INFORMATION_MESSAGE);

}
```

JOptionPane - Exemplo 3



- Qual a saída?

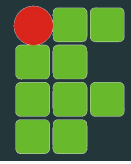
```
package aula04;

import javax.swing.JOptionPane;

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

        int resp = JOptionPane.showConfirmDialog(null, "Iniciar Download",
                   "Caixa de Download",
                   JOptionPane.YES_NO_OPTION,
                   JOptionPane.QUESTION_MESSAGE);

        if (resp == JOptionPane.YES_OPTION){
            JOptionPane.showMessageDialog(null,"Download CONCLUÍDO","Download",JOptionPane.INFORMATION_MESSAGE);
        }else{
            JOptionPane.showMessageDialog(null,"Download ABORTADO","Download",JOptionPane.ERROR_MESSAGE);
        }
        System.exit(0);
    }
}
```

Exercícios

- Utilize a leitura de dados via *Scanner*/showInputDialog e mostre a saída utilizando *JOptionPane*.

1. Faça um programa em Java para solicitar um número pelo teclado e imprimir a tabuada deste.
2. Faça um programa em Java para solicitar dois números pelo teclado e imprimir a média aritmética.
3. Faça um programa em Java para solicitar um número pelo teclado e imprimir o fatorial deste.

Próxima aula



- Revisão de sub-rotinas/funções e modularização.
- Resolução de exercícios selecionados da lista de revisão de lógica.
- Introdução ao ambiente gráfico *SWING* do *Java*.
- Introdução a orientação a objetos.

Dúvidas, críticas ou sugestões?

maximilian.melo@ifms.edu.br

max.mjm.melo@gmail.com