Introdução a Java

Mário Meireles Teixeira
UFMA – DEINF

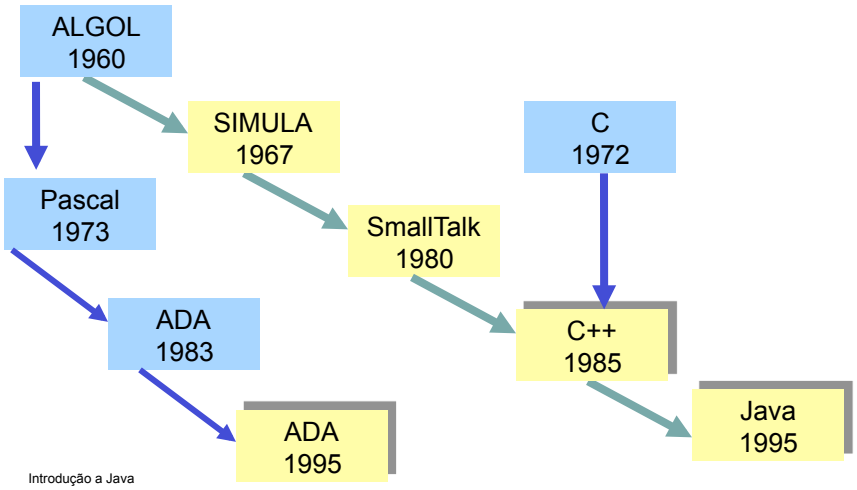
Tecnologia Java

- Java = linguagem de programação de alto nível + plataforma de execução de aplicações
- Java como linguagem de programação:
  - Aplicativos locais (programas Java) e distribuídos (sockets, RMI)
  - Aplicativos que executam em browsers (Applets)
  - Aplicativos corporativos de grande porte e Páginas Web dinâmicas (Servlets, JSP, JSF → WebApps)
  - Aplicativos para dispositivos móveis (Java ME, Android)
- Java como ambiente de execução:
  - ambiente neutro (JRE – Java Runtime Environment) para diferentes plataformas: SO's, browsers, celulares, tablets...

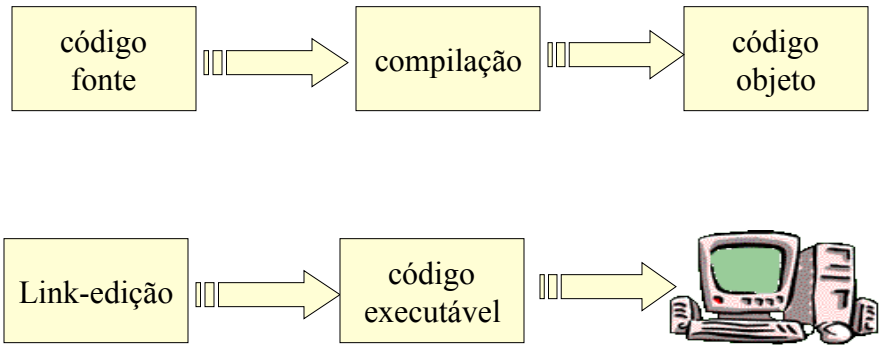
Linguagem Java

- Linguagem de programação orientada a objetos:
  - Familiar (sintaxe parecida com C)
  - Simples e robusta (minimiza bugs, aumenta produtividade)
  - Suporte nativo a threads (+ simples, maior portabilidade)
  - Dinâmica (módulos, acoplamento em tempo de execução)
  - Com coleta de lixo (menos bugs, mais produtividade)
  - Independente de plataforma
  - Segura (vários mecanismos para controlar segurança)
  - Código intermediário de máquina virtual interpretado (compilação rápida, execução + "lenta", + produtividade no desenvolvimento)
  - Sintaxe uniforme, rigorosa quanto a tipos (código mais consistente)

Linguagens OO: genealogia



Desenvolvimento de programas

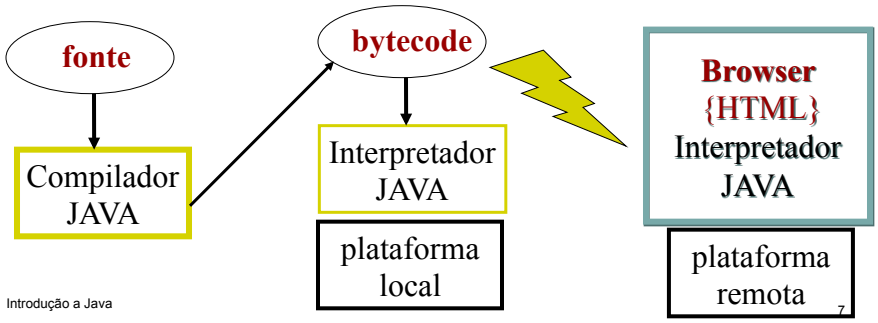


Implementação de LP

- Compilação
  - geração de código executável
  - dependente da plataforma de execução
  - tradução lenta X execução rápida
- Interpretação pura
  - sem geração de código
  - execução lenta, independente de plataforma
- Híbrida
  - geração de código intermediário
  - independente de plataforma de execução
  - tradução rápida X execução não muito rápida

Compilação e Execução

- **Compilação:** Programa Java é compilado para um código intermediário conhecido como bytecode
- **Execução:** Bytecodes executam em qualquer máquina que possua uma JVM (Java Virtual Machine) → Independência de plataforma!



Compilação x Interpretação

- Problema:
  - Para ter flexibilidade e segurança, abre-se mão do tempo de execução
  - Um programa Java típico costumava ser 10 vezes mais lento que um programa em C (na década de 90)
- Solução: Compiladores JIT (*Just-in-time)*
  - Convertem *bytecodes* Java (.class) para linguagem de máquina nativa otimizada à medida que os mesmos são lidos
  - Penalidade: a primeira leitura dos programas é mais lenta
  - Porém, execuções subsequentes são mais rápidas
  - Atualmente, invocações de métodos Java almejam ser tão rápidas quanto chamadas de funções virtuais puras em C++ (binding dinâmico)

Ambientes de execução e desenvolvimento

- **JSE (Java Platform; Standard Edition)** - JDK
  - Coleção de ferramentas para desenvolvimento, depuração e execução de aplicações Java (JRE)
  - Destinado a desenvolvedores
- **Java Runtime Environment** (JRE)
  - Ambiente para execução de aplicações Java
  - Voltado a usuários finais
  - Faz parte do JDK e das principais distribuições Linux, MacOS X, Windows 98/ME/2000/XP/7/8...
- Download: http://java.com

Plataformas Java

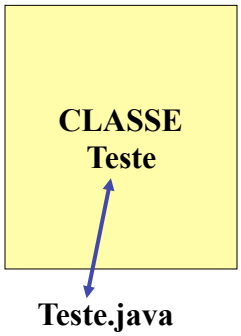
- Java possui diversas plataformas para desenvolvimento e execução de aplicações (www.java.com), logo é importante conhecê-las e entender qual delas deve ser utilizada em cada caso
- As principais APIs Java são distribuídas pela Oracle juntamente com os kits para desenvolvimento de aplicações:
  - **Java Standard Edition (JSE)**: ferramentas e APIs para desenvolvimento de aplicações Java para PCs e servidores
  - **Java Enterprise Edition (JEE):** ferramentas e APIs para aplicações corporativas e de Internet (distribuídas, Web Services...)
  - **Java Micro Edition (Java ME):** ferramentas e APIs para aplicações em dispositivos móveis e embarcados
    - **Android SDK:** API de propósito específico, para desenvolvimento de aplicativos para dispositivos com sistema operacional Android (API do Google, não da Oracle!)

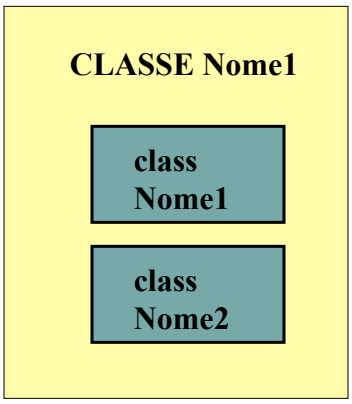
Ambientes Integrados de Desenvolvimento – IDEs

- Existem diversos IDEs construídos a partir das ferramentas básicas de desenvolvimento:
  - ambientes baseados em janelas
  - editores, visualizadores de classes, prototipação ...
- Exemplos:
  - Eclipse, NetBeans (Oracle)
  - Visual Studio (Microsoft), BlueJ, GreenFoot, Alice
  - JBuilder (Borland), Visual Café (Symantec)...

Programa Java = Classe

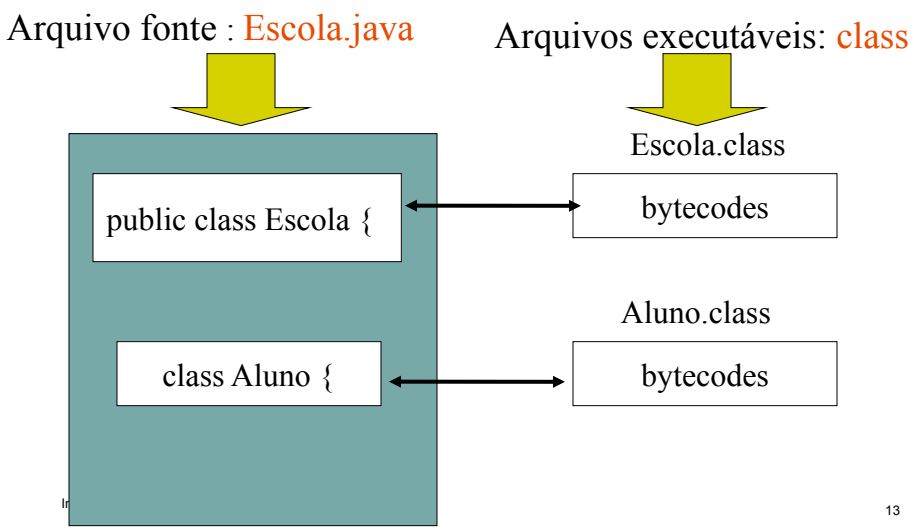
• Uma classe contém código-fonte em Java (texto simples)





Um arquivo .java pode conter várias classes, porém deve ter o **mesmo nome** de uma de suas classes.

Arquivos fonte e executáveis



Convenção para Nomes

- Java, como C/C++, distingue entre letras maiúsculas e minúsculas
  - Exemplo: escola é diferente de Escola
- Nomes de classes iniciam com maiúscula
  - Exemplo: class Bemvindo
- Nomes de variáveis iniciam com minúsculas
  - Exemplo: int peso;
- Nomes de métodos são verbos que iniciam com minúscula e depois usam maiúsculas
  - Exemplo: alteraPeso
- Representação: Unicode (16 bits - 65.536 caracteres)

Compilação e Execução: JDK – linha de comando

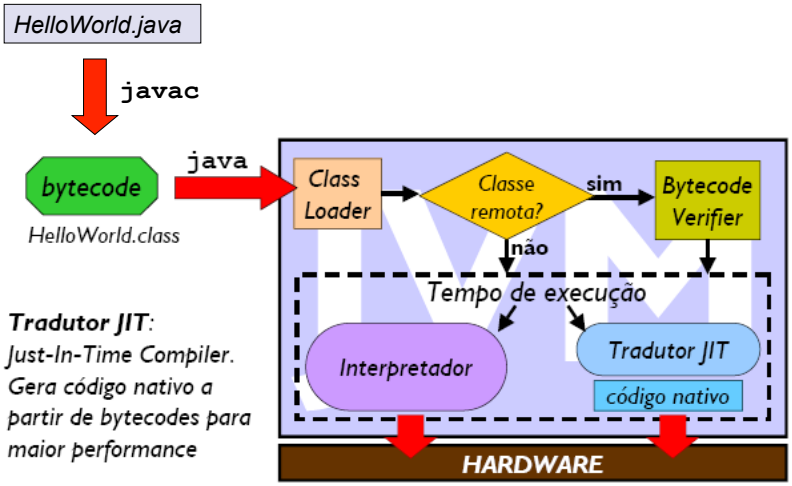
- Compilação: javac
- A partir do diretório local:
  javac Nome.java
- Vai produzir:
  - arquivos .class separados para cada classe no arquivo .java
  - coloca arquivos no diretório corrente

- Execução: java
- A partir do diretório local:
  java Nome

onde: Nome = arquivo .class que contém main()

- Executa a partir de main()
- ou com argumentos:
  java Nome arg1,arg2, ...

Execução de programas Java

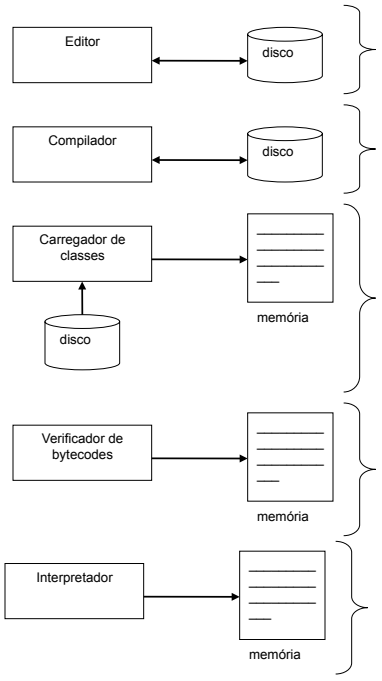


***Tradutor JIT**:*
Just-In-Time Compiler. Gera código nativo a partir de bytecodes para maior performance

Verificação de bytecode

- Etapa que antecede a execução do código em classes carregadas através da rede
  - *Class Loader* distingue classes locais (seguras) de classes remotas (potencialmente inseguras)
- Verificação garante:
  - Aderência à especificação da JVM
  - Não-violação de políticas de acesso estabelecidas pela aplicação
  - Não-violação da integridade do sistema
  - Ausência de estouros de pilha
  - Tipos de parâmetros corretamente especificados
  - Ausência de conversões ilegais de tipos

Processo de Autoria de Aplicativos Java



Programas JAVA

- **Aplicações** (puras)
  - são programas carregados e executados localmente pelo interpretador Java
  - possuem acesso a todos os recursos do ambiente local: diretórios, arquivos
  - sempre contêm um método principal main(), de onde se inicia a execução do programa
  - podem fazer chamadas a programas em código nativo (outras linguagens, como C, C++)

Programas Cliente/Servidor

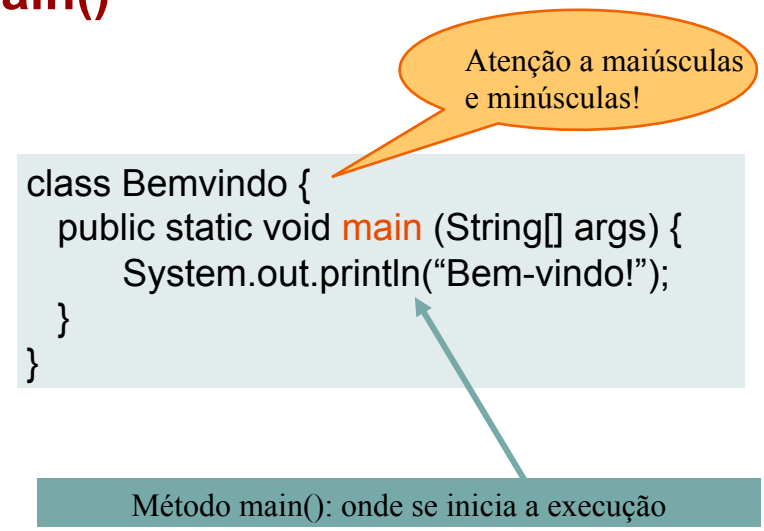
- **Applets**: cliente
  - são programas inseridos em páginas HTML e executados pelo browser (interpretador Java)
  - programas (classes) podem ser carregados remotamente
  - restrições de segurança: não podem acessar recursos locais, só podem comunicar-se com o servidor de onde foram "baixados"
- **Servlets**: servidor
  - executados no ambiente do servidor de aplicações (p.ex., TomCat)
  - classe Java carregada no servidor, dinamicamente, por requisição do cliente
  - permite páginas web dinâmicas

Applets Java X JavaScript

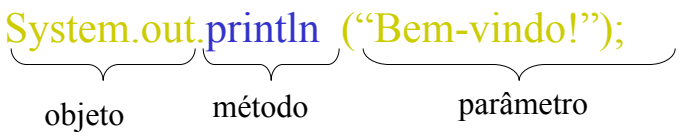
- Ambos são códigos executáveis inseridos em páginas HTML
- Código Java: classes compiladas e carregadas remotamente
- Código JavaScript: instruções fisicamente dispersas ao longo da página HTML
- Java é uma LP de uso geral, orientada a objetos: classes, objetos, encapsulamento, herança e polimorfismo
- JavaScript: linguagem script não OO, sem suporte à herança ou polimorfismo

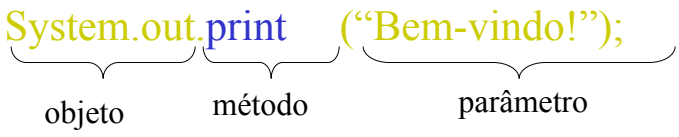
Primeiros passos

Classe Java: apenas o método main()



Exemplo de chamada de métodos





Chamada parametrizada

```
class Bemvindo {
        public static void main(String[] args) {
            System.out.println("Bem-vindo " + args[0]);
        }
}
```

Linha de comando: **`java Bemvindo Pedro`**

Método main(): aceita argumentos para execução

Classe Java: usando um método estático

```
class Bemvindo2 {
        static void imprime() {
            System.out.println("Bem-vindo!");
        }

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

Não recomendável

Método static: método de classe, não aplicável sobre objetos

Classe Java: instanciando um objeto

```
class Bemvindo3 {
  void imprime() {
        System.out.println("Bem-vindo!");
  }

  public static void main(String[] args) {
        Bemvindo3 obj = new Bemvindo3();
        obj.imprime();
  }
}
```

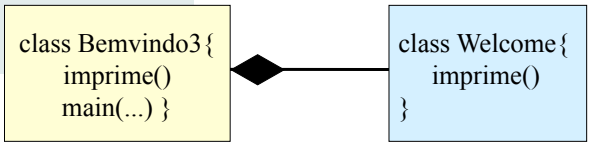
Criação do objeto

Chamada do método

Método de instância, acessado **somente** através de um objeto

Duas classes: composição

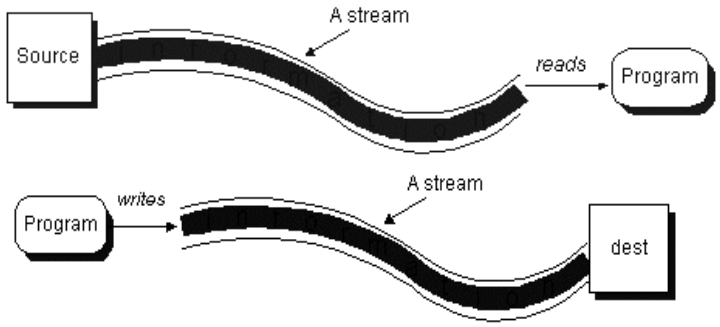
```
class Bemvindo3{
void imprime(){
  System.out.println("Bem-vindo!");
}
public static void main(String[] args){
  Bemvindo3 obj=new Bemvindo3();
  Welcome outro=new Welcome();
  obj.imprime();
  outro.imprime();
}}
```




```
class Welcome{
   void imprime(){
     System.out.println("Welcome!");
   }}
```

Streams de entrada/saída

- Em um programa, dados são lidos/gravados através de *streams* (fluxos) de E/S
- Fontes e destinos: console, arquivos, rede...
- Um programa pode gerenciar diversos streams



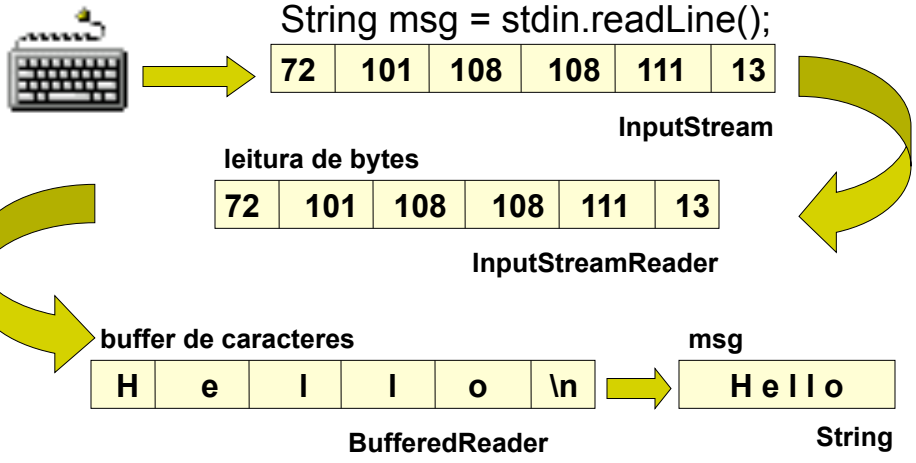
Chamada Parametrizada X Entrada/Saída via Console

- Chamada parametrizada:
  - programa inicia com valores armazenados em um array de Strings
  - não é uma forma usual de entrada de dados
- Ao iniciar a execução de um programa, são criados automaticamente os seguintes streams:
  - System.in - standard input (teclado)
  - System.out - standard output (vídeo)
  - System.err - standard error (vídeo)
  - Podem ser redirecionados para disco, por exemplo

Entrada via Console

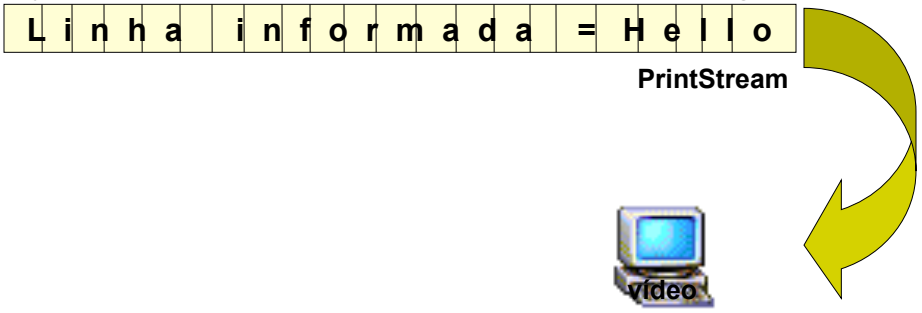
- Classe BufferedReader
  - usada para leitura de um *stream* de caracteres para um buffer
  - definida no pacote java.io
- Método readLine()
  - definido em BufferedReader
  - faz a leitura de uma linha de texto até um caracter **'\n'**
  - causa uma exceção se ocorrer um erro enquanto lê os dados da entrada

Entendendo Entrada via Console



Entendendo Saída via Console

System.out.println("Linha informada = "+ msg);



Entrada via Console: String

```
import java.io.*;

class Leitura {
  public static void main(String[] args) throws
                                          IOException{

      BufferedReader stdin = new BufferedReader(
                new InputStreamReader(System.in));

      System.out.println("Entre uma linha:");
      String msg = stdin.readLine();
      System.out.println("Linha informada = "+ msg);
  }
}
```

Entrada via Console: int

```
import java.io.*;

class Leitura {
   public static void main(String[] args) throws
                                                IOException{

        BufferedReader stdin = new BufferedReader(
                    new InputStreamReader(System.in));

        System.out.println(“Entre um número:”);
        int i = Integer.parseInt(stdin.readLine());
        System.out.println(“Número = ”+ i);
   }
}
```

Usando Scanner() e printf()

```
import java.io.*;
import java.util.Scanner;

public class LeScanner {
   public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Digite seu nome: ");
        String nome = input.next();
        System.out.print("Digite sua idade: ");
        int idade = input.nextInt();

        System.out.printf("%s tem %d anos\n", nome, idade);
   } }
```

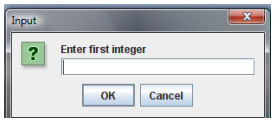
Usando Caixas de Diálogo

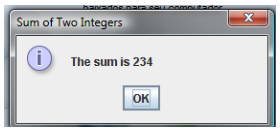
```
public class Addition
{
  public static void main( String args[] )
  {
    // obtain user input from JOptionPane input dialogs
    String firstNumber =
      JOptionPane.showInputDialog( "Enter first integer" );
    String secondNumber =
      JOptionPane.showInputDialog( "Enter second integer" );

    // convert String inputs to int values for use in a calculation
    int number1 = Integer.parseInt( firstNumber );
    int number2 = Integer.parseInt( secondNumber );

    int sum = number1 + number2; // add numbers

    // display result in a JOptionPane message dialog
    JOptionPane.showMessageDialog( null, "The sum is " + sum,
      "Sum of Two Integers", JOptionPane.INFORMATION_MESSAGE );
  } // end method main
} // end class Addition
```





Tutoriais de Java

- Para quem está começando e além:
  - The Java Tutorials (Sun/Oracle oficial)
    - https://docs.oracle.com/javase/tutorial/
  - Learn Java Programming (Tutorials Point)
    - http://www.tutorialspoint.com/java/
  - Eclipse and Java (Vídeo-aulas)
    - http://eclipsetutorial.sourceforge.net/totalbeginner.html