Unip – Universidade Paulista

Linguagem de Programação Orientada a Objetos - LPOO

Aula 1 – Introdução e Impressão

Profº Ms Gustavo Molina

Agenda

- Algoritmos
- Introdução ao JAVA
  - Histórico
  - JVM- Java Virtual Machine
    - Características
  - Estudo do Ambiente para o desenvolvimento de programas;
  - Edição, compilação e enlace;
  - Execução e Configuração;

Introdução ao JAVA [I/II]

- Histórico
  - Primeiros passos 1991, primeira versão disponível para *download*: 1995.
  - Originalmente desenvolvido para dispositivos eletrônicos inteligentes de consumo popular.
  - Depois utilizado para criar páginas da Web com conteúdo dinâmico.

Agora também utilizado para:

Desenvolver aplicativos corporativos de larga escala.

Aprimorar funcionalidades de servidores Web.

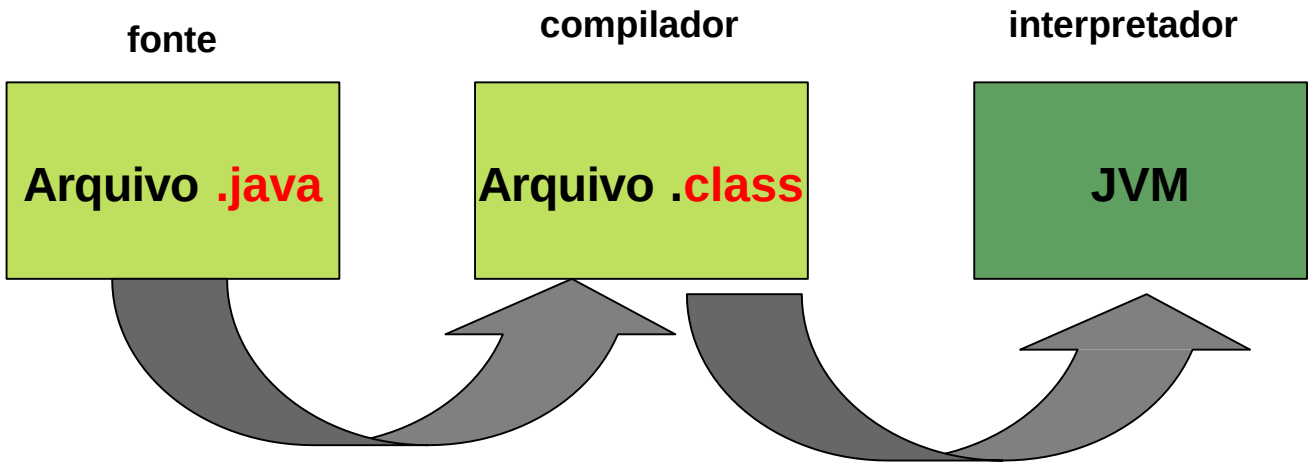
Fornecer aplicativos para dispositivos de consumo popular (telefones celulares etc.)

Introdução ao JAVA [II/II]

- Programas Java
  - Consistem em partes chamadas *classes*, as quais incluem *métodos* que realizam tarefas e retornam informações ao concluir.
  - Programadores podem criar classes e métodos para construir programas Java.
- O Java oferece bibliotecas de classe
  - Conhecidas como Java APIs (Application Programming Interfaces) ou APIs do Java.

Java Virtual Machine (JVM)

- É uma abstração, provendo, um ambiente multi plataforma para publicação de aplicativos JAVA;
- Recebe chamadas *bytecodes*;
- Chamadas são independentes do SO;
- *Bytecodes são gerados a partir de um compilador JAVA;*

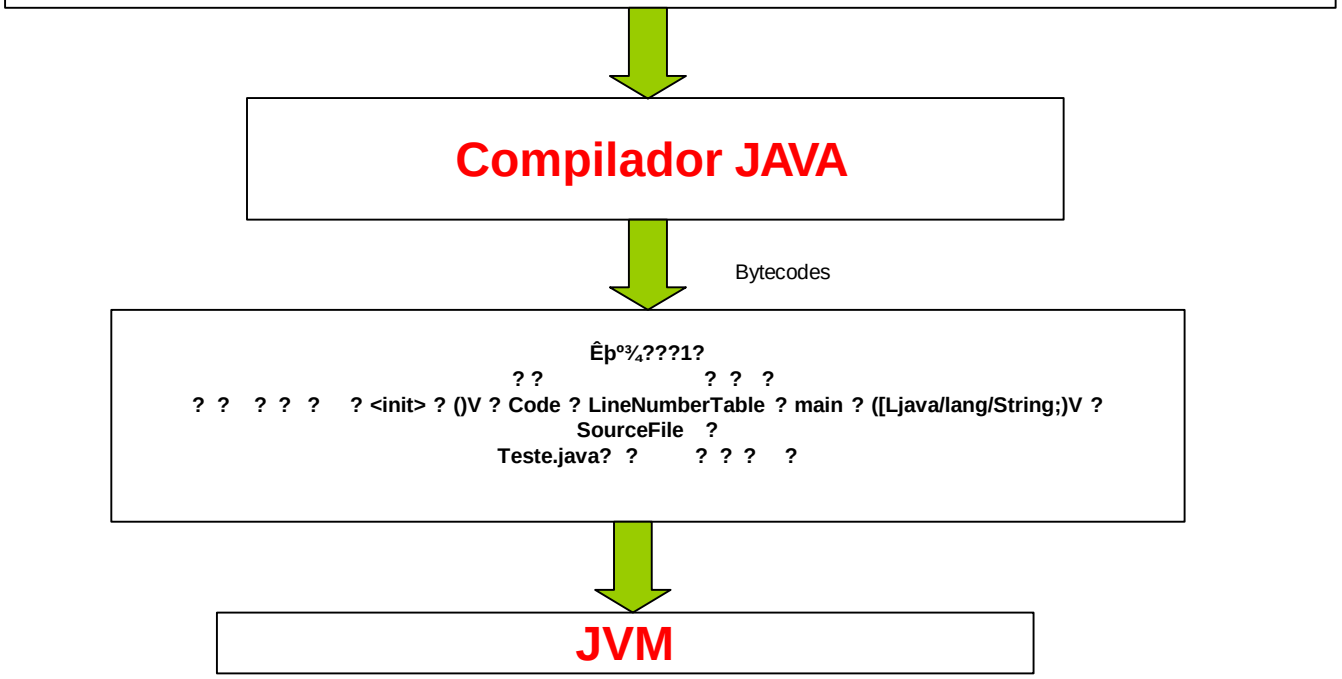


Exemplo JVM

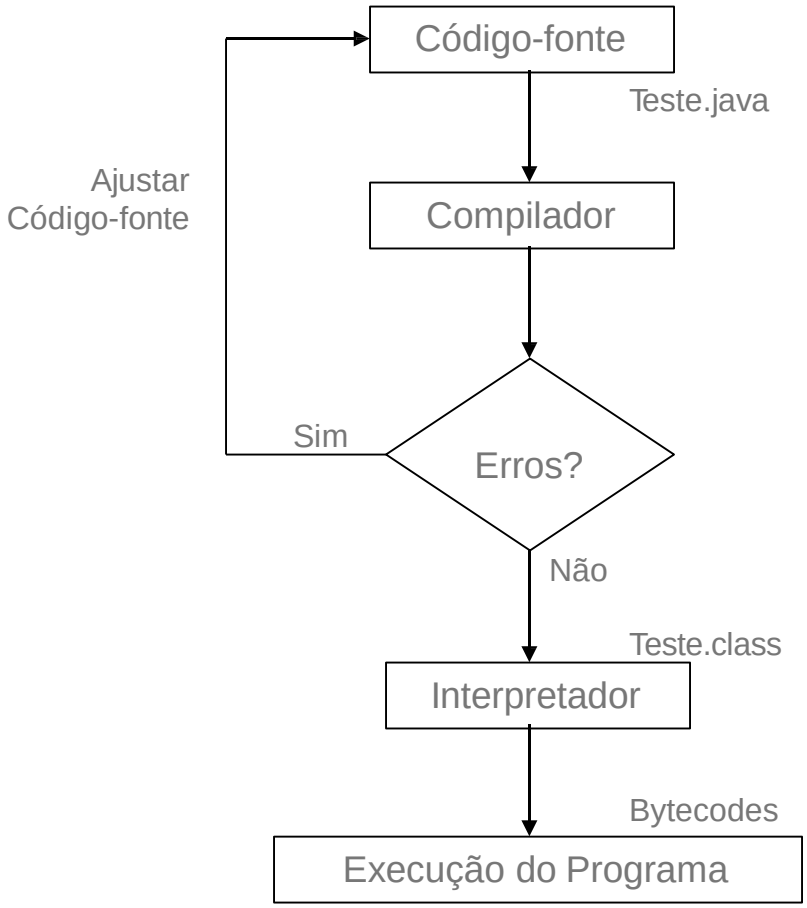
Teste.java

```
class Teste {

public static void main (String args[ ]) {
        System.out.print(“Primeira aula”);
        System.out.print(“Linguagem Java”);
}
```



Sequência de criação/execução



Na fase de execução é necessário que haja a Máquina Virtual Java (MVJ)

A MVJ interpreta os *bytecodes* gerados pelo interpretador.

O objetivo da MVJ é permitir que qualquer sistema operacional possa executar uma aplicação Java

JAVA - Principais Características

- É uma linguagem de desenvolvimento orientada a objetos;
- É Multiplataforma;
- Uma coleção de APIs (classes, componentes, frameworks) para o desenvolvimento de aplicações multiplataforma;
- Um ambiente de execução presente em browsers, mainframes, SOs, celulares, palmtops, cartões inteligentes, eletrodomésticos;
- A evolução da linguagem é controlada pelo Java Community Process (www.jcp.org) formado pela Sun e usuários Java

Versões do JAVA

- As principais APIs são distribuídas juntamente com os produtos para desenvolvimento de aplicações
  - Java 2 Standard Edition *(J2SE)*: ferramentas e APIs essenciais para qualquer aplicação Java (inclusive GUI)
  - Java 2 Enterprise Edition *(J2EE)*: ferramentas e APIs para o desenvolvimento de aplicações distribuídas
  - Java 2 Micro Edition *(J2ME)*: ferramentas e APIs para o desenvolvimento de aplicações para aparelhos portáteis

Primeiro programa Java

Welcome1.java

```
1  // Fig. 2.1: Welcome1.java
2  // Programa de impressão de texto.
3
4  public class Welcome1
5  {
6     // método main inicia a execução do aplicativo Java
7     public static void main( String args[] )
8     {
9        System.out.println( "Welcome to Java Programming!" );
10
11    } // fim do método main
12
13 } // fim da classe Welcome1
```

```
Welcome to Java Programming!
```

Explicação [I/VI]

```
1  // Fig. 2.1: Welcome1.java
2  // Programa de impressão de texto.
```

Comentários iniciam com: //

- Comentários ignorados durante a execução do programa.
- Documentam e descrevem o código.
- Fornecem legibilidade do código.

Comentários tradicionais: /* ... */

```
/* Este é um comentário
 tradicional. Ele pode se estender
 por várias linhas */
```

Nota: os números de linha não fazem parte do programa; eles foram adicionados para referência.

Explicação [II/VI]

```
3
```

- Linha em branco.
  - Torna o programa mais legível.
  - Linhas em branco, espaços e tabulações são caracteres de espaço em branco.
    - Ignorados pelo compilador

```
4 public class Welcome1
```

- Inicia a declaração de classe para a classe Welcome1
  - Cada programa Java tem pelo menos uma classe definida pelo usuário.
  - Palavra-chave: palavras reservadas para uso pelo Java.
    - Palavra-chave class seguida pelo nome da classe.
  - Atribuindo nomes de classes: coloque a inicial de cada palavra em maiúscula.
    - ExemploDeNomeDeClasse

Explicação [III/VI]

```
4   public class Welcome1
```

– Salvando arquivos:
  - O nome do arquivo deve possuir a extensão `.java`.
  - `Welcome1.java`

```
5   {
```

– Chave esquerda {
  - Inicia o corpo de toda classe.
  - A chave esquerda termina as declarações (linha 13).

Explicação [IV/VI]

– Parte de todo aplicativo Java:

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

- Aplicativos começam a executar em `main`.
  - Parênteses indicam que `main` é um método .
  - Aplicativos Java contêm um ou mais métodos.
- Exatamente um método deve se chamar `main`.

– Métodos podem realizar tarefas e retornar informações:

- `void` significa que `main` não retorna nenhuma informação.
- Por enquanto, simula a primeira linha de `main`.

```
8      {
```

– A chave esquerda, `{`, inicia o corpo de declaração do método.

– A chave direita, `}`, termina o corpo (linha 11 do programa).

Explicação [V/VI]

```
9         System.out.println( "Welcome to Java Programming!" );
```

- Instrui o computador a realizar uma ação:
  - Imprime strings de caracteres.
    - String: série de caracteres entre aspas duplas.
  - Espaços em branco em strings não são ignorados pelo compilador.
- `System.out`:
  - Objeto de saída-padrão.
  - Imprime na janela de comando (isto é, Prompt do MS-DOS).
- Método `System.out.println`:
  - Exibe uma linha de texto.
- Isso é conhecido como uma declaração:
  - Instruções terminam com ponto-e-vírgula (`;`).

Explicação [VI/VI]

```
11     } // fim do método main
```

– Termina a declaração de método.

```
13  } // fim da classe Welcome1
```

– Termina a declaração da classe.

– É possível adicionar comentários para monitorar chaves finais.

Erros comuns em Java

- **O Java diferencia letras maiúsculas de minúsculas [é *"case sensitive"*]. Não diferenciar as letras maiúsculas e minúsculas adequadas para um identificador normalmente causa um erro de compilação, onde `a1` e `A1` são diferentes.**

Referências Bibliográficas

- Deitel , H.M. Java: Como Programar -6. ed- São Paulo: Pearson Prentice Hall, 2005.
- Site da oracle: http://www.oracle.com/technetwork/pt/java/javase/overview/index.html