fusberti@ic.unicamp.br

Introdução aos Aplicativos Java

prof. Fábio Luiz Usberti

MC322 - Programação Orientada a Objetos

Instituto de Computação - UNICAMP - 2014

Sumário

Histórico

História do Java

- Java (1991) evoluiu a partir do C++ (1980), que evoluiu do C (1972), que evoluiu do B (1970), que evoluiu do BCPL (1966).
- A popularização da linguagem Java teve dois principais fatores:
  - A portabilidade dos aplicativos, em virtude da introdução de uma máquina virtual.
  - A possibilidade de desenvolver aplicativos web (applets) permitiu introduzir dinamicidade às páginas da internet, logo isso fez com que os principais navegadores incorporassem a funcionalidade de execução desses aplicativos.

Histórico

Linguagem BCPL

- BCPL (1966): Basic Combined Programming Language (ou *Before C Programming Language*).
- Martin Richards (University of Cambridge).
- Desenvolvida para escrever compiladores de outras linguagens.
- Programa “Hello world”:

```
GET "LIBHDR"

LET START () BE
$(
    WRITES ("Hello world!*N")
$)
```

Histórico

Linguagem B

- B (1969): nome supostamente advindo da religião tibetana *Bön*.
- Ken Thompson e Dennis Ritchie (Bell Labs).
- A linguagem B consiste essencialmente da remoção de muitos componentes da linguagem BCPL, diminuindo os requisitos de memória para tornar a linguagem compatível com os computadores da época.
- Programa “Hello world”:

```
main( ) {
  extrn a, b, c;
  putchar(a); putchar(b); putchar(c); putchar('!*n');
}
a 'Hell';
b 'o wo';
c 'rld!';
```

Histórico

Linguagem C

- C (1972): extensão da linguagem B.
- Dennis Ritchie (Bell Labs).
- Linguagem estruturada de propósito geral, utilizada para o desenvolvimento do sistema operacional UNIX. Linguagem mais utilizada nos dias de hoje.
- Programa “Hello world”:

```c
#include <stdio.h>

int main(void)
{
    printf("Hello world!\n");
}
```

Histórico

Linguagem C++

- C++ (1979): extensão da linguagem C.
- Bjarne Stroustrup (Bell Labs).
- Linguagem orientada a objetos, que integra atributos de linguagens de baixo e alto níveis.
- Programa “Hello world”:

```
# include <iostream>

int main()
{
  std::cout << "Hello world!\n";
}
```

Histórico

Linguagem Java

- Em 1991, a empresa Sun Microsystems financiou um projeto de pesquisa que resultou em uma linguagem baseada em C++ que seu criador, James Gosling, chamou de *Oak* em homenagem à árvore vista da janela de sua sala.
- Descobriu-se mais tarde que já havia uma linguagem de computador com esse nome.

Histórico

Linguagem Java

- Ao visitar uma cafeteria local, uma equipe da Sun sugeriu o nome Java, cidade exportadora de café.

- Programa “Hello world”:

```
public class HelloWorld {

    public static void main(String[] args) {
        System.out.println("Hello world!");
    }

}
```

Características

Objetivos dos criadores da linguagem

Durante a concepção da linguagem Java, seus desenvolvedores tinham em mente os objetivos listados abaixo:

- Simples, Familiar e Orientada a Objetos
- Robusta e Segura
- Independente de arquiteturas e Portável
- Bom desempenho
- Interpretada e Paralelizável

Características

Simples, Familiar e Orientada a Objetos

- Os conceitos fundamentais são simples o suficiente para que sejam facilmente absorvidos pelos programadores.
- A semelhança com outras linguagens como C++ torna a transição para Java mais fácil e familiar.
- A necessidade crescente de sistemas distribuídos e complexos, baseados em cliente-servidor, coincide com os conceitos de linguagem orientada a objetos, especialmente a possibilidade de encapsulamento fornecida pela linguagem.

Características

Robusta e Segura

- A linguagem Java foi construída para permitir o desenvolvimento de aplicativos de alta confiabilidade.
- O gerenciamento de memória é bastante simples: não há ponteiros explícitos (criados pelo programador) e as memórias que deixaram de ser utilizadas pelo aplicativo são automaticamente liberadas pelo *coletor de lixo* (garbage collector).
- Ela provê extensivas checagens de segurança, em tempo de compilação, para impedir invasão externa de sistemas, invasão de arquivos e criação de vírus.

Características

Independente de arquiteturas e Portável

- Os aplicativos Java suportam plataformas, hardwares e redes heterogêneas.
- Para acomodar essa diversidade de ambientes, o compilador Java produz instruções, denominadas bytecodes, que são interpretadas por uma máquina virtual, um software que simula um computador
- A portabilidade também é alcançada pela convenção dos tamanhos dos tipos de dados e dos comportamentos dos operadores aritméticos.

Características

Bom desempenho

- Os aplicativos Java atingem desempenhos competitivos com outras linguagens.
- O *garbage collector* é executado somente como uma thread de baixa prioridade.
- Aplicações que requerem melhor desempenho podem ter parte de seus códigos reescritos em linguagem de máquina (Hot Spot compiler).

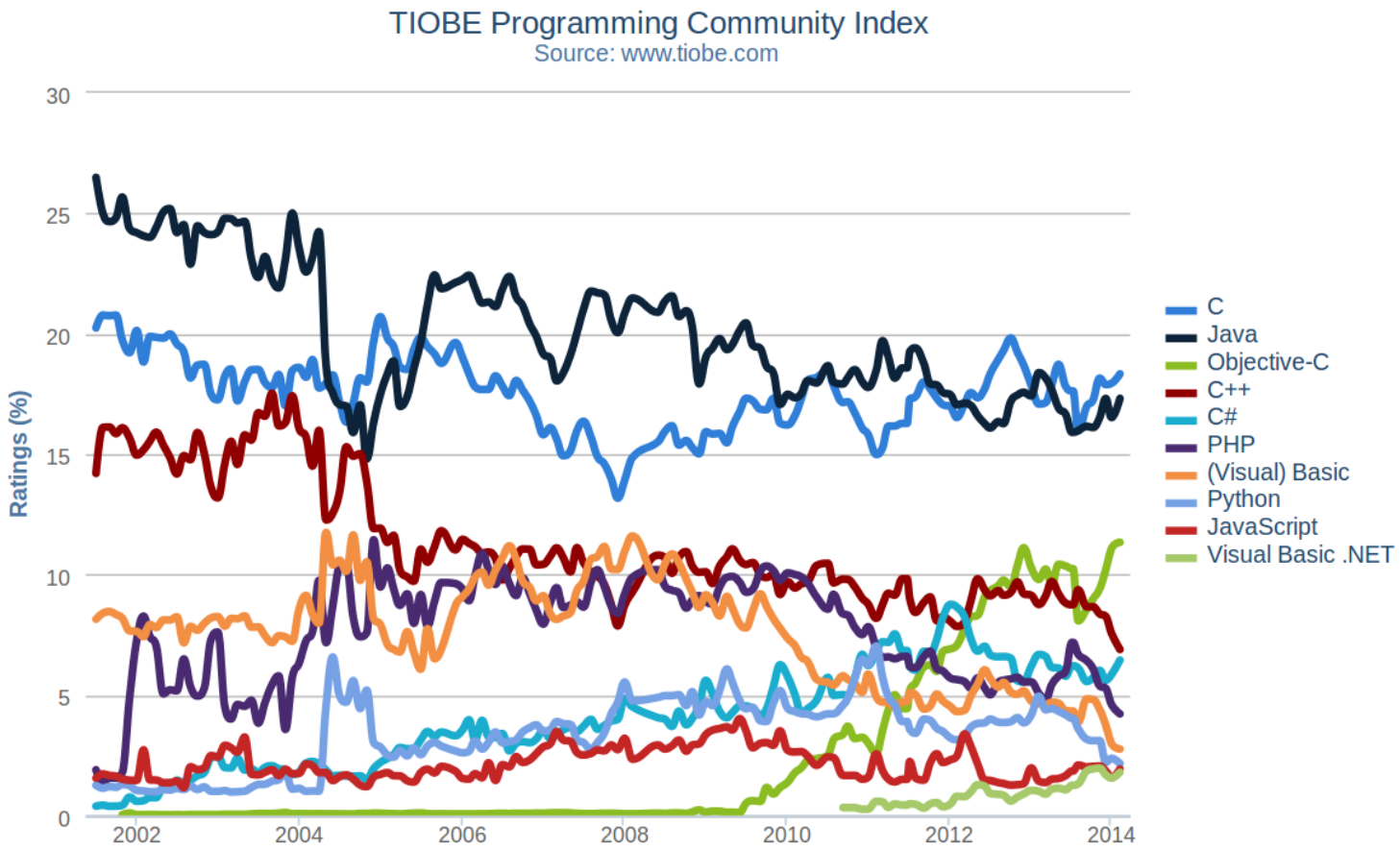
Características

Interpretada e Paralelizável

- Os bytecodes podem ser interpretados em qualquer máquina contanto que a máquina virtual esteja devidamente instalada.
- A plataforma Java permite multithreading com a adição de métodos para sincronização. Muitas bibliotecas em Java já foram elaboradas para suportar multithreading e concorrência.

fusberti@ic.unicamp.br

Percentual de uso das linguagens de programação



Desenvolvimento passo-a-passo

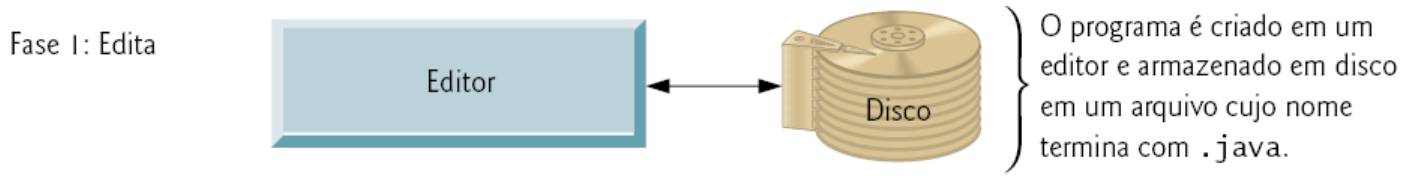
Fases de desenvolvimento

- Programas Java passam por cinco fases até sua execução:
  1. Edição
  2. Compilação
  3. Carregamento
  4. Verificação
  5. Execução

Desenvolvimento passo-a-passo

Fase 1. Edição

- Esta fase consiste em digitar um código-fonte em um arquivo utilizando um programa editor de texto.
- Esse arquivo é salvo em disco com extensão `.java`



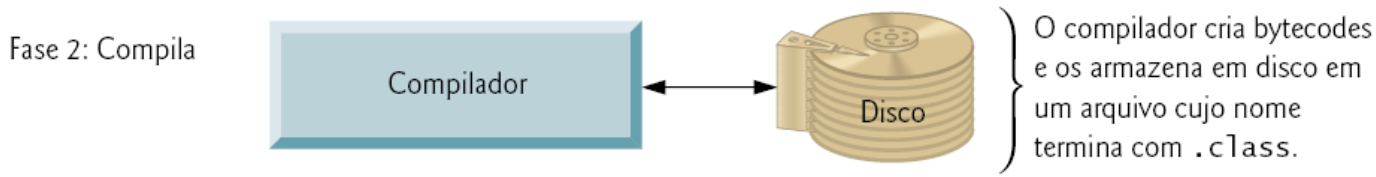
Desenvolvimento passo-a-passo

Fase 2. Compilação

- Nesta fase o compilador Java é executado para o código-fonte utilizando o comando `javac` em um terminal:

```
javac HelloWorld.java
```

- O arquivo compilado terá extensão `.class` (HelloWorld.class).
- O compilador Java converte o código-fonte Java em bytecodes que representam as instruções a serem executadas pela JVM (Java Virtual Machine).



Desenvolvimento passo-a-passo

Fase 2. Compilação

- A máquina virtual é um aplicativo de software que simula um computador, ocultando o sistema operacional e hardware dos programas que interagem com ela.
- Ao contrário de uma linguagem de máquina, que é dependente do hardware, os bytecodes são independentes da plataforma.
- Isso implica que os bytecodes são portáveis, ou seja, sem recompilar o código-fonte, os mesmos bytecodes podem ser executados em qualquer plataforma (obs: a JVM precisa ser compatível).

Desenvolvimento passo-a-passo

Fase 3. Carregamento

- A JVM é chamada pelo comando `java` em um terminal:

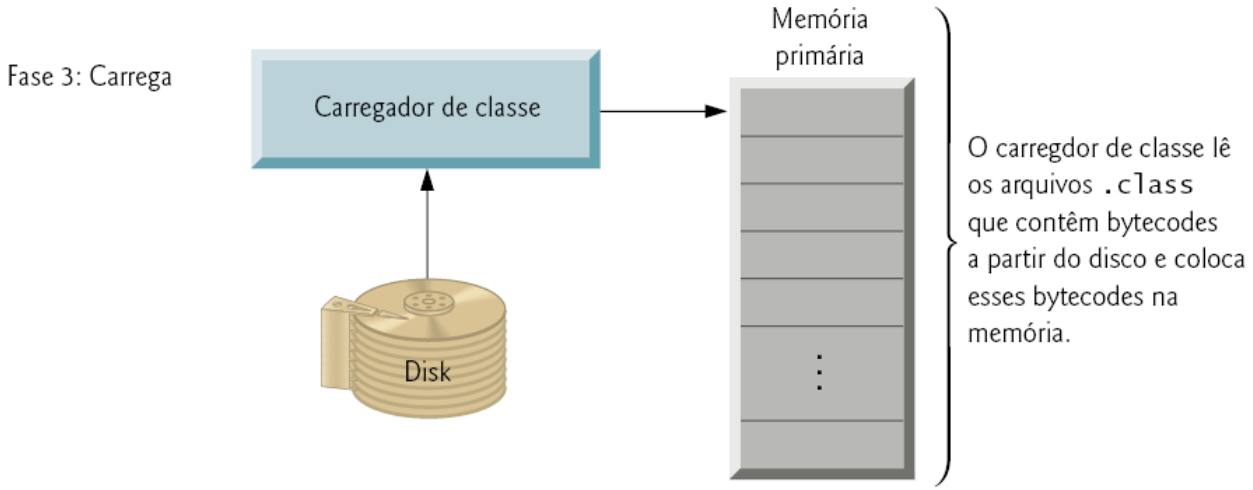
```
java HelloWorld.class
```

- Essa chamada inicia a fase de carregamento, onde o carregador de classe da JVM transfere os bytecodes do programa para a memória principal.

Desenvolvimento passo-a-passo

Fase 3. Carregamento

- São carregados todos os arquivos `.class`, inclusive as Java APIs[1] (Application Programming Interfaces) utilizadas pelo programa.
- As APIs consistem em uma rica biblioteca de classes Java rigorosamente implementadas para executar de modo eficiente.

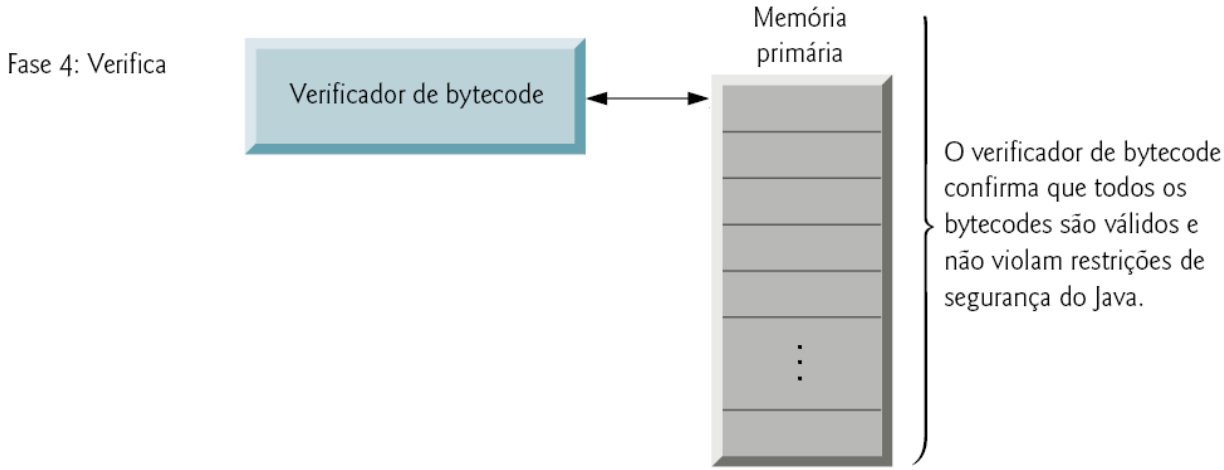


[1] http://docs.oracle.com/javase/7/docs/api/

Desenvolvimento passo-a-passo

Fase 4. Verificação

- Nesta fase, enquanto as classes são carregadas, o verificador de bytecodes da JVM examina os bytecodes para assegurar que são válidos e não violam restrições de segurança.
- Os códigos Java passam por diversas certificações de segurança para reduzir ataques a arquivos e sistemas por vírus de computador.



Desenvolvimento passo-a-passo

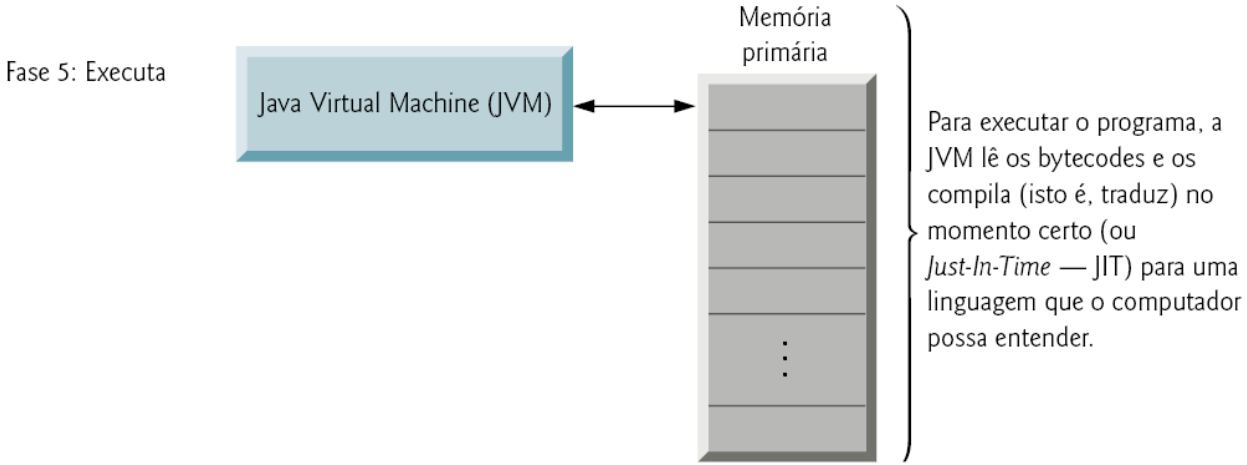
Fase 5. Execução

- Na fase 5, a JVM executa as intruções definidas pelos bytecodes.
- Nas primeiras versões de Java, a JVM era simplesmente um interpretador de bytecodes, o que deixava a execução dos programas lenta.
- As atuais JVMs executam bytecodes combinando interpretação e compilação Just-In-Time (JIT).
- Na compilação JIT, a JVM analisa os bytecodes enquanto são interpretados, procurando por hot spots, parte dos bytecodes executados com frequência.

Desenvolvimento passo-a-passo

Fase 5. Execução

- Os programas Java passam então por duas fases de compilação:
  - pré-execução: o código-fonte é traduzido em bytecodes pelo compilador Java.
  - durante execução: os bytecodes são traduzidos por um compilador JIT em linguagem de máquina para o computador real em que o programa é executado.



Java IDEs

Integrated Development Environment (IDE)

- Uma IDE é um software que fornece facilidades para programadores e desenvolvedores de software.
- As principais funcionalidades fornecidas são um editor de código-fonte, ferramentas de montagem automática e um depurador.
- Dois importantes IDEs para Java são:

https://www.eclipse.org/

NetBeans

https://netbeans.org/

Olá Mundo!

Dissecando o programa "Olá Mundo!"

- Um aplicativo Java é um programa executado quando utilizamos o comando `java` para carregar a JVM.
- Vamos examinar em detalhes um aplicativo simples que imprime o texto "Olá Mundo!" na tela.

```
// Programa HelloWorld.java
/* Programa que exemplifica a impressão de texto em Java.
   Autor: Fábio L. Usberti. Data da última modificação: 23/02/2014. */

public class HelloWorld {

  public static void main(String[] args) {
    System.out.println("Olá Mundo!");
  } // fim do método main

} // fim da classe HelloWorld
```

Olá Mundo!

Comentário de fim de linha

- O símbolo `//` indica um comentário de fim de linha, que tem esse nome por terminar no fim da linha em que aparece.
- Boa prática de programação: coloque depois da chave de fechamento do corpo de uma declaração de método ou de classe um comentário que indica a qual método ou classe a chave pertence.

```
// Programa HelloWorld.java
/* Programa que exemplifica a impressão de texto em Java.
   Autor: Fábio L. Usberti. Data da última modificação: 23/02/2014. */

public class HelloWorld {

  public static void main(String[] args) {
    System.out.println("Olá Mundo!");
  } // fim do método main

} // fim da classe HelloWorld
```

Olá Mundo!

Comentário tradicional

- Um comentário tradicional inicia com `/*` e termina com `*/` .
- Todo o texto entre esses delimitadores são ignorados pelo compilador.
- Comentários no estilo Javadoc são delimitados por `/**` e `*/` permitindo incorporar a documentação no próprio código-fonte e posterior exportação em formato HTML.
- Boa prática de programação: comece o programa com um comentário que informa o objetivo e o autor do programa, a data e a hora em que o programa foi modificado pela última vez.

```
// Programa HelloWorld.java
/* Programa que exemplifica a impressão de texto em Java.
   Autor: Fábio L. Usberti. Data da última modificação: 23/02/2014. */

public class HelloWorld {

  public static void main(String[] args) {
    System.out.println("Olá Mundo!");
  } // fim do método main

} // fim da classe HelloWorld
```

Olá Mundo!

Declaração de uma classe

- Todo programa Java possui pelo menos uma classe definida pelo programador.
- A palavra-chave `class` introduz uma declaração de classe e é imediatamente seguida pelo nome da classe (`HelloWorld`).
- Convenção: O nome de classe deve começar por letra maiúscula assim como a inicial de cada palavra incluída (`HelloWorld`).
- A palavra-chave `public` indica que a classe é acessível por qualquer outra classe definida no programa.
- Importante: O arquivo `.java` deve possuir o mesmo nome da classe pública declarada.

```
// Programa HelloWorld.java
/* Programa que exemplifica a impressão de texto em Java.
   Autor: Fábio L. Usberti. Data da última modificação: 23/02/2014. */

public class HelloWorld {

  public static void main(String[] args) {
    System.out.println("Olá Mundo!");
  } // fim do método main

} // fim da classe HelloWorld
```

Olá Mundo!

Declaração de um método

- Os parênteses depois do identificador `main` indicam que trata-se de um método.
- O método `main()` é o ponto de partida para todas as aplicações Java.
- Um aplicativo precisa possuir um método `main()` como declarado no exemplo para ser executado pela JVM.
- A palavra-chave `public` indica que o método é acessível por qualquer outra classe definida no programa.
- A palavra-chave `static` indica que o método `main()` não pertence a um objeto instanciado da classe, mas pertence à própria classe.

```
// Programa HelloWorld.java
/* Programa que exemplifica a impressão de texto em Java.
   Autor: Fábio L. Usberti. Data da última modificação: 23/02/2014. */

public class HelloWorld {

  public static void main(String[] args) {
    System.out.println("Olá Mundo!");
  } // fim do método main

} // fim da classe HelloWorld
```

Olá Mundo!

Argumentos de um método

- O argumento de um método `main()` deve ser declarado como `String[] args`
- Esse argumento trata-se de um vetor de objetos da classe `String`, que representam cadeias de caracteres.
- Objetos do tipo `String` são imutáveis, ou seja, depois de inicializados eles não podem mais ser modificados.
- O vetor `args` armazena em cada posição um dos argumentos passados na execução de um aplicativo.

```
// Programa HelloWorld.java
/* Programa que exemplifica a impressão de texto em Java.
   Autor: Fábio L. Usberti. Data da última modificação: 23/02/2014. */

public class HelloWorld {

  public static void main(String[] args) {
    System.out.println("Olá Mundo!");
  } // fim do método main

} // fim da classe HelloWorld
```

Olá Mundo!

Argumentos de um método

- Por exemplo, executando a classe `HelloWorld` com os argumentos `3.14` e `pi`:

```
java HelloWorld 3.14 pi
```

- Nesse caso, tem-se que `args[0]` conterá `"3.14"` e `args[1]` conterá `"pi"`.

```
// Programa HelloWorld.java
/* Programa que exemplifica a impressão de texto em Java.
   Autor: Fábio L. Usberti. Data da última modificação: 23/02/2014. */

public class HelloWorld {

  public static void main(String[] args) {
    System.out.println("Olá Mundo!");
  } // fim do método main

} // fim da classe HelloWorld
```

Olá Mundo!

Impressão de texto

- A classe `System` fornece ao programador entrada e saída padrões.
- O objeto `System.out` é conhecido como o objeto de saída padrão, que permite aos aplicativos Java exibirem caracteres no terminal.
- O método `System.out.println` imprime uma cadeia de caracteres (fornecida como argumento) no terminal.
- A linha completa `System.out.println("Olá Mundo!");` representa uma instrução, logo deve ser finalizada com `;` ponto-e-vírgula.

```
// Programa HelloWorld.java
/* Programa que exemplifica a impressão de texto em Java.
   Autor: Fábio L. Usberti. Data da última modificação: 23/02/2014. */

public class HelloWorld {

  public static void main(String[] args) {
    System.out.println("Olá Mundo!");
  } // fim do método main

} // fim da classe HelloWorld
```

Referências

1. Java: Como Programar, Paul Deitel & Heivey Deitel; Pearson; 7a. Ed. (no. chamada IMECC – 05.133 D368j)
2. Data Structures and Algorithms with Object Oriented Design Patterns in Java, Bruno Preiss; (http://www.brpreiss.com/books/opus6/)
3. The Java Tutorials (Oracle) (http://docs.oracle.com/javase/tutorial/)
4. Guia do Usuário UML, Grady Booch et. al.; Campus(1999)
5. Java Pocket Guide - Robert Liguori & Patricia Liguori; O'Reilley, 2008.