

Programação Orientada a Objetos (PO24CP)

Aula #02 - Introdução ao Ambiente Java

Prof^a Luciene de Oliveira Marin
lucienemarin@utfpr.edu.br

Introdução ao Ambiente Java

- Em 1991 um pequeno grupo de engenheiros da *Sun Microsystems*, chamado de “*Green Team*”, acreditava que a nova onda computacional seria a união dos dispositivos eletrônicos portáteis com os computadores
 - **Nessa época o MS-DOS era o sistema operacional dominante e um telefone celular pesava meio kilo**
- Surge a linguagem Java, nome inspirado durante uma reunião do grupo em uma cafeteria.



História da linguagem Java

O foco inicial do *Green Team*:

produtos para entretenimento ligados a indústria de **TV digital**

- O conceito estava muito avançado para a época e o projeto começava a ruir.

A Internet:

Em 1993 o projeto toma um novo rumo com o surgimento do primeiro navegador gráfico para web, o Mosaic 1.0 da NCSA. Em 1995 a Sun lança oficialmente o ambiente Java e sua incorporação no Netscape Navigator trouxe vida as páginas web, antes estáticas.

História da linguagem Java

O foco inicial: **StartSeven** - *7



O Java continua a estar presente:

- em páginas web:
 - Em sua forma original (applets) ou em uma forma mais moderna (Java Server Pages - JSP)
- Aplicações para computadores de mesa - IRPF
- Aplicações servidoras - Apache Tomcat
- Dispositivos móveis (telefone celular, PDA, GPS, videogame)
- Em sistemas embarcados - Ginga (SBTVD interativa), SmartTVs

Características da linguagem Java

Orientada a objetos

- Paradigma que surgiu na década de 60 que tem como foco dados, ou objetos, e suas interfaces.
- Recursos de OO do Java são comparáveis aos recursos do C++

Robustez

- Ênfase na verificação preliminar de possíveis problemas, verificação dinâmica (em tempo de execução) e eliminação de situações propensas a erros
- Apresenta uma solução elegante para os principais pontos fracos do C++
 - **Alocação dinâmica de memória e ponteiros**

Características da linguagem Java

Neutro em relação à arquitetura

O compilador Java gera um código intermediário, chamado de *bytecode*, que pode ser executado em qualquer arquitetura de máquina e S.O que tenha um ambiente de execução Java (Máquina virtual Java)

Portável

- Na especificação da linguagem não existem pontos como “dependente de implementação”, como ocorre em C e C++
- Em Java o tipo primitivo `int` sempre consiste de um número inteiro de 32 bits

```
#include<stdio.h>
```

```
#include<limits.h>
```

```
int main(void){
```

```
    printf("%d\n", INT_MAX); /* Qual o valor? Depende da tua  
                             arquitetura, 32bits, 64bits...*/
```

```
    return 0;
```

```
}
```

Características da linguagem Java

Independente de plataforma

Escreva uma única vez e rode em qualquer lugar que possua uma máquina virtual Java (JVM)

Biblioteca completa para concepção de aplicações complexas

- Programação concorrente - *Multi-thread*
- Programação distribuída

Alto desempenho

- Os bytecode são interpretados pela JVM resultando em um desempenho inferior quando comparado com códigos compilados para um CPU específico;
- Os compiladores de bytecode "*just-in-time*" surgem como uma solução para este problema, impondo em alguns casos um desempenho superior

O Java é interpretado, portanto é muito mais lento

- Compiladores *just-in-time* permitem que códigos Java sejam executados com tanta rapidez como códigos C++
- A inicialização da JVM e as interfaces gráficas em Java (GUI) são sim lentas

C# é uma linguagem mais nova, deixando o Java obsoleto

- C# incorporou muitas boas ideias do Java, como máquina virtual, linguagem limpa, coleta de lixo
- Mas deixou para trás a segurança e independência de plataforma (feita para Windows, apesar de haver máquinas virtuais de terceiros para outros S.O, ex: Mono)

Javascript é uma versão simplificada do Java

Javascript foi criada pela Netscape para criação de scripts que podem ser usada em páginas Web

Ferramentas para desenvolvimento

O kit de desenvolvimento Java (*Java Development Kit - JDK*) é distribuído gratuitamente pela Oracle

- Compilador, máquina virtual Java, código fonte, documentação das APIs
- <http://www.oracle.com/technetwork/java/javase/downloads>

Para executar uma aplicação Java é necessário possuir uma máquina virtual Java

A Oracle disponibiliza gratuitamente o “ambiente de execução Java” (Java Runtime Environment - JRE) para diversos sistemas operacionais e arquiteturas de máquina

Algumas tecnologias Java

Java SE - Standard Edition

Para o desenvolvimento de aplicações desktop

Java EE - Enterprise Edition

Para o desenvolvimento de aplicações empresariais

Java ME - Mobile Edition

Para o desenvolvimento de aplicações para dispositivos móveis

Java Embedded

Para o desenvolvimento de aplicações que conectam dispositivos eletrônicos (como aparelhos eletrodomésticos, eletroportáteis, máquinas industriais, meios de transporte etc.) à Internet

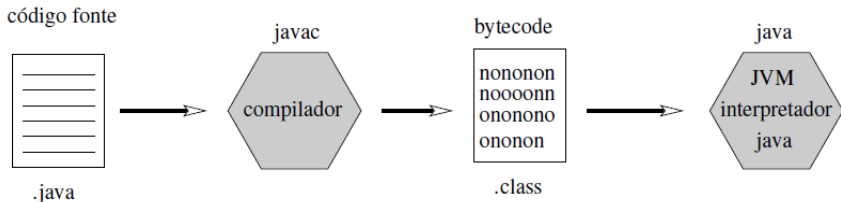
Ambientes integrado de desenvolvimento

Os ambientes integrados de desenvolvimento (AID) (ou *Integrated Development Environment* - IDE)

Tornam mais agradável e produtivo o desenvolvimento de aplicações:

- Netbeans - Apoiado pela Sun (agora Oracle)
 - <http://www.netbeans.org>
- Eclipse - Bem usado pela comunidade
 - <http://www.eclipse.org>

Criando e executando um aplicativo Java



- Compilando

```
javac Arquivo.java
```

- Executando

```
java Arquivo
```

Definições iniciais

- Um programa em Java consiste em uma coleção de classes
- Geralmente cada classe possui seu respectivo arquivo `.java`
- O nome do arquivo deve ser idêntico ao nome da classe
- O conteúdo do método `main` é a primeira parte de uma classe a ser executada, quando pretende-se que esta seja executada como um aplicativo

Primeiro código em Java - OlaMundo.java

```
public class OlaMundo{  
    public static void main(String[] args){  
        // imprimindo a mensagem na tela  
        System.out.println("Ola mundo!");  
    }  
}
```

- Compilando e executando

```
$ javac OlaMundo.java
```

```
$ java OlaMundo
```

Referências sobre a linguagem

```
double d;  
// algumas funcoes matematicas  
d = Math.sqrt(25); // obtem a raiz quadrada  
d = Math.pow(4,2); // 4 elevado a 2  
d = Math.sin(45); // Math.cos(45), Math.tan(45)...  
  
// obtendo numeros pseudo-aleatorios de 0 a 9  
Random r = new Random();  
int i = r.nextInt(10);  
  
// formatando a saida  
// largura de campo de 8 caracteres e precisao de 2 caracteres  
System.out.printf("%8.2f", d);  
  
// criando um vetor de inteiros com 10 posicoes  
int[] vet = new int[10];  
vet[0] = 5;  
vet[9] = 4;
```

Referências sobre a linguagem

Lendo informações pelo teclado

```
import java.util.Scanner;

public class Segundo{

    public static void main(String[] args){

        Scanner teclado = new Scanner(System.in);

        int i = teclado.nextInt(); // lendo inteiro
        double r = teclado.nextDouble(); // lendo real
        String s = teclado.nextLine(); // lendo cadeia de caracteres
        System.out.println("inteiro: " + i + ", real: " + r);
        System.out.println("Frase: " + s);
    }
}
```

Referências sobre a linguagem

Lendo informações de outra forma

```
import javax.swing.JOptionPane;

public class Terceiro{

    public static void main(String[] args){
        String texto;
        texto = JOptionPane.showInputDialog("Entre com um numero");
        //convertendo String para int
        int numero = Integer.parseInt(texto);
        JOptionPane.showMessageDialog(null, numero);
    }
}
```



Caelum Ensino e Soluções em Java

Apostila Caelum FJ-11 Java e Orientação a Objetos

<http://www.caelum.com.br/download/caelum-java-objetos-fj11.pdf>

⇒ Capítulo 3 - Leitura obrigatória

⇒ Capítulo 2 - Leitura recomendada


Exercícios - Introdução ao Ambiente Java

Revisão de conceitos básicos de programação
(declaração de variáveis, if, else, for, while, vetores).

Exercícios de fixação (1/4)

1. Leia um número do teclado e imprima todos os números ímpares de 0 até o número lido;
2. Leia um número do teclado e informe se este número é primo ou não;
3. Leia um número do teclado e informe se este é um número perfeito. Número perfeito é um número inteiro cuja soma de todos os seus divisores positivos, excluindo ele próprio, é igual ao próprio número;
4. Leia um número do teclado e informe todos os números primos entre 0 e este número;

5. Preencha um vetor com 10 números pseudo-aleatórios e imprima este vetor de forma ordenada. Faça uso do **algoritmo de ordenação bolha**.¹

¹https://pt.wikipedia.org/wiki/Bubble_sort 

6. Leia um número inteiro, positivo e ímpar n e imprima uma matriz com $2 * n - 1$ colunas, onde a última linha tem um 1 no centro com 0's em volta, a penúltima linha tem três 1's no centro com 0's em volta, a antepenúltima linha tem cinco 1's, e assim por diante (os 1's formarão um triângulo com a base para cima), até a primeira linha ser composta apenas por 1's. A condição de parada do programa é n igual a -1. A leitura de n deve ser validada de forma que o programa apenas aceite números ímpares positivos como entrada válida para produzir a matriz.

Exercícios de fixação (4/4)

Exemplos:

$n = 3$

1	1	1	1	1
0	1	1	1	0
0	0	1	0	0

$n = 5$

1	1	1	1	1	1	1	1	1
0	1	1	1	1	1	1	1	0
0	0	1	1	1	1	1	0	0
0	0	0	1	1	1	0	0	0
0	0	0	0	1	0	0	0	0