Aula 1 – POO 1
Prática

Profa. Elaine Faria

UFU - 2020

Introdução

- Java
  - Linguagem de programação poderosa
  - Utiliza o paradigma Orientado a Objetos
  - Muito utilizada em aplicativos para Internet e para redes
  - Difundida em aplicações corporativas
  - Utilizada em programas para *smartphones*

Histórico

- Projeto financiado pela SUN em 1991
  - Desenvolvimento de uma linguagem baseada no C++
  - Nome dado: *OAK* (árvore)
  - Novo nome: Java (cidade de origem de um tipo de café importado)
- 1993: WWW explodiu em popularidade
  - Java tinha potencial para trabalhar com WEB

Histórico

- 2009: o Java pertence à Oracle

- Atualmente o Java vem sendo utilizado em aplicações de grande porte, principalmente na WEB

Ambiente de desenvolvimento Java

- Programas Java passam por 5 fases
  - Edição
  - Compilação
  - Carga
  - Verificação
  - Execução

Ambiente de desenvolvimento Java

- Edição
  - Consiste em editar um arquivo em um programa editor
  - Digitação do código fonte utilizando o editor
  - Nome do código fonte java termina com .java
  - Exemplo de editores: bloco de notas, *NetBeans, Eclipse, JBuilder, BlueJ,* etc.

Ambiente de desenvolvimento Java

- Compilação
  - Compilar o programa usando o *javac*
    - javac nomearquivo.java
  - O compilador produz um arquivo .*class* que contém a versão compilada do programa
  - O compilador converte o código fonte em *bytecodes*
    - Os *bytecodes* são executados pela *Java Virtual Machine* (JVM)

Ambiente de desenvolvimento Java

- Compilação
  - JMV
    - Executa as instruções do programa na plataforma de hardware nativa para a qual a JVM foi escrita
    - Simula um computador e oculta o SO e o hardware subjacentes dos programas que interagem com a VM
      - Os *bytecodes* independem da plataforma, podem ser executados em qualquer plataforma contendo a JVM
    - É invocada por: java nomearq

Ambiente de desenvolvimento Java

- Carga
  - Alocação do programa na memória antes de ser executado
  - O carregador transfere os arquivos *java.class* (*bytecodes*) para a memória principal
  - Também carrega os arquivos *.class* que seu programa utiliza

Ambiente de desenvolvimento Java

- Verificação
  - Enquanto as classes são carregadas o verificador examina os *bytecodes* para assegurar que eles são válidos e não violam restrições de segurança do java
  - O Java impõe restrições de segurança contra vírus, por exemplo

Ambiente de desenvolvimento Java

- Execução
  - A JVM executa os *bytecodes* do programa realizando ações especificadas
  - Antigamente as JVM interpretavam o *bytecode* resultando em execuções lentas
  - Atualmente as JVM executam os *bytecodes* usando uma combinação de interpretação com compilação *just-in-time*(JIT)

Ambiente de desenvolvimento Java

- Execução
  - JIT
    - A JVM analisa *bytecodes* à medida que são interpretados procurando *hot spots* (pontos ativos), que são partes que executam com freqüência
    - Para as partes freqüentes, um compilador JIT traduz os *bytecodes* para linguagem de máquina
    - Quando a JVM encontra as partes compiladas em linguagem de máquina, estas são mais rapidamente executadas

Java

- Estrutura:
  - Pacotes
    - Mecanismo de namespace
  - Classes
    - Ficam dentro dos pacotes
  - Métodos
    - Ficam dentro das classes

Java - Ferramentas

- *Eclipse*
  - Ferramenta utilizada para edição e compilação de códigos Java
  - IDE de Software Livre
  - Será usada na nossa disciplina
- Exemplos de outras ferramentas
  - NetBeans e BlueJ

Instalação do Java

- Eclipse
  - https://www.eclipse.org/downloads/
  - Para usar o Eclipse é preciso ter o JDK instalado
  - Se você fizer o download de um dos pacotes configuráveis Eclipse, ele já virá com o JDK.

Instalação do Java

- JDK
  - https://www.oracle.com/technetwork/java/javase/downloads/index.html

Java *Development Kit*

- JDK (*Java Development Kit*)
  - É um grande "pacote" com tudo o que o desenvolvedores necessitam para trabalhar com Java
  - Contém: o compilador Java (*javac*), *Java Debugger* (*javadb*) e a JVM (*Java Virtual Machine*)
  - Contém: biblioteca de classes completa de utilitários de pré-construção que ajuda o desenvolvedor a realizar tarefas de desenvolvimento de aplicativo mais comuns.

Java *Runtime Environment*

- JRE (*Java Runtime Environment*)
  - É um pacote mais restrito, utilizado apenas para executar aplicações Java.
  - É composto principalmente pela JVM
  - Está incluído na JDK

Eclipse

- Principais componentes:
  - Área de trabalho (*workspace*)
    - Contém todos os seus *projetos*
  - Projetos
  - Perspectivas
    - É uma forma de consulta a cada projeto
  - Visualizações
    - *Package* Explorer e o *Outline* são algumas das visualizações muito usadas

Eclipse

- Principais componentes:
- *Visualizaçãoes*
  - *Package* Explorer
    - Projeto
    - Pacotes
    - Classes
  - *Console*
  - *Error log*
  - Ambiente para codificação

Primeiro Programa

- Criar um projeto chamado POO1
- Criar uma classe chamada Inicial

```
public class Inicial {
    public static void main(String args[]){
        System.out.println(“Hello World! ”);
    }
}
```

Primeiro Programa

- Cada programa Java consiste em pelo menos uma declaração de classe definida pelo programador
- A palavra *class* introduz uma classe e deve ser seguida pelo nome da classe (no exemplo a classe chama-se Inicial)
- Por convenção o nome das classes começam com letra maiúscula

Primeiro Programa

- O comando *public static void main (String args[])* é o ponto de partida de cada aplicativo Java
- Para um aplicativo Java ser executado ele deve conter um método *main*
- A palavra-chave *void* indica que o método realizará uma tarefa, mas não retornará nenhuma informação complementar

Primeiro Programa

- *String args[]* é uma parte requerida da declaração do método *main*
- O comando System.out.println(“Hello World”) instrui o computador a imprimir a *string* de caracteres contidas entre aspas duplas
- *System.out* é conhecido como objeto de saída padrão

Primeiro Programa

- Usar ponto-e-vírgula (;) no final de cada instrução
- Comentário em Java
  - Bloco: /* */
  - Linha: //

Variáveis

- Correspondem a posições na memória do computador
- Possuem: nome, tipo, tamanho e valor
- O Java é *Case Sensitive*

Variáveis

- No Java existem os seguintes tipos de dados
  - Tipo lógico: *boolean*
  - Tipo textual: *char* e *String*
  - Tipo inteiro: *byte, short, int e long*
    - 8 bits byte $-2^7 \ldots . 2^7 - 1$
    - 16 bits short $-2^{15} \ldots . 2^{15} - 1$
    - 32 bits int $-2^{31} \ldots . 2^{31} - 1$
    - 64 bits long $-2^{63} \ldots . 2^{63} - 1$
  - Tipo ponto flutuante: *float* ou *double*

Palavras Reservadas

abstract	do	implements	private	throw
boolean	double	import	protect	throws
break	else	instanceof	public	transient
byte	extends	int	return	true
case	false	interface	short	try
catch	final	long	static	void
char	finally	native	super	volatile
class	float	new	switch	while
continue	for	null	synchronized	
default	if	package	this	

Operadores Aritméticos

- Adição: +
- Subtração: -
- Multiplicação: *
- Divisão: /
  - A divisão de inteiros produz um inteiro: Exemplo 7/4 = 1
- Resto: %

Obs.: Parênteses podem ser usados assim como na álgebra

As regras de precedência de operadores são as mesmas da álgebra

Operadores de Igualdade

- Igual: ==
  - Ex: x == y
- Diferente:!=
  - Ex: x!=y

Operadores Relacionais

- Maior que: >
- Menor que: <
- Maior que ou igual a: >=
- Menor que ou igual a: <=

Operadores Lógicos

- Conjunção: &&
- Disjunção: ||
- Negação: !
- Disjunção exclusiva (XOR): ^

Desvio Condicional

Sintaxe

```
if (condição) {
      codigo-se-condição-true;
}
else {
      código-se-condição-false;
}
```

Exemplo

```
if ( x < y)
      System.out.println(“ x e menor do que y”);
else
      System.out.println(“ y e maior);
```

Desvio Condicional

Exemplo

```
if (x> w)
{ // inicio do bloco
   int y=50;
   System.out.println(“dentro do bloco”);
   System.out.println(“x:” + x);
   System.out.println(“y:” + y);
} // final do bloco
```

Desvio Condicional

```
switch (variável){
    case valor 1: {
        bloco-de-código;
        break;
    }
    case valor 2: {....}
    ....
    default: {...}
}
```

Loop

- O loop For em Java tem a sintaxe:

```
for(inicialização; teste;
incremento){
     bloco de comandos;
}
```

- O loop While tem a sintaxe:

```
while (condição){
     bloco de comandos;
}
```

Arrays

- Declarando um Array:

```
String difficult[];
Point hits[];
int temp[];
```

- Criando Objetos Arrays:
  - O operador new para cria uma nova instância de um array,

```
int[] temps = new int[99];
```

Arrays

- Acessando os Elementos do Array
  - Os arrays em Java sempre iniciam-se na posição 0 como no C++

```
String[] vet = new String[10];
vet[10]=”erro...”;
```

  - Descobrir tamanho do array vet:

```
vet.length;
```

Arrays Multidimensionais

```
int coords[][]= new int[12][12];
coords[0][0] = 1;
coords[0][1] = 2;
```

Mais detalhes da sintaxe da linguagem serão vistos nas próximas aulas

Exercício

- Crie uma classe java chamada Operadores.
- Crie duas variáveis, atribua valores a cada uma delas
- Teste cada um dos operadores vistos nesta aula
- Teste os tipos de dados vistos nesta aula
- Teste os comando *if, for, while*