Aula 2 – Introdução ao Java

Prof. Me. Rodrigo Brito Battilana

Roteiro

- Configurando Variáveis de Ambiente Java
- Introdução a linguagem Java
  - Introdução
  - Estrutura do programa
  - Comentários no código fonte
  - Identificadores e classes
  - Executando o primeiro programa em Java
  - Modificando o programa em Java

Roteiro

- Operadores
  - Aritméticos
  - Lógicos
  - Relacionais
- Referências

Configurando variáveis de ambiente para o Java

- O JDK (Java Development Kit) possui todo o ambiente necessário para desenvolver e executar aplicativos em Java.

- O JRE (Java Runtime Envirorment) é o ambiente de execução Java, é o mínimo que você precisa ter instalado para poder rodar um aplicativo Java.

Configurando variáveis de ambiente para o Java

- Para configurar as variáveis de ambiente no java, são necessários realizar alguns procedimentos:

1 - Clique com o botão direito em cima do ícone **Meu Computador**

2- Vá em **Propriedades**;

3 – Escolha a aba **Configurações avançadas do sistema**, depois na aba **Avançado**

Configurando variáveis de ambiente para o Java

4 - Clique no botão **Variáveis de ambiente**;

5 - Clique no botão **Nova** em **Variáveis do sistema**;

6 - Escolha o Nome da variável: “JAVA_HOME”

7 - Valor da variável: coloque aqui o endereço de instalação C:\Arquivos de programas\Java\jdkX.X.XXX

8 - Clique em **OK**

Introdução à linguagem Java

Introdução

- Um aplicativo Java é um programa de computador que é executado quando você utiliza o comando java para carregar a Java Virtual Machine (JVM).

- Para trabalhar com a estrutura do Java, geralmente é necessário criar algum projeto dentro da hierarquia de pacotes do NetBeans.

Introdução

- Consideremos um aplicativo simples que exibe uma linha de texto.

```
1   // Figura 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     } // fim do método main
11  } // fim da classe Welcome1
```

```
Welcome to Java Programming!
```

Figura 2.1 | Programa de impressão de texto.

Comentários no código fonte

- **Comentários tradicionais**

- Podem ser distribuídos ao longo de várias linhas. Eles começam e terminam com delimitadores, /* e */, como em:

```
/* Esse é um comentário tradicional. Ele
   pode ser dividido em várias linhas */
```

- O compilador ignora todo o texto entre os delimitadores.
- O Java incorporou comentários tradicionais e comentários de fim de linha das linguagens de programação C e C++, respectivamente.

Comentários no código fonte

- **Comentários Javadoc**

- São delimitados por //** e */. O compilador ignora todo o texto entre os delimitadores.
- São o formato de documentação Java preferido na indústria.
- O programa utilitário javadoc (parte do JDK) lê comentários Javadoc e os usa para preparar a documentação do programa no formato HTML.

Comentários no código fonte

- Linhas em branco, caracteres de espaço e tabulações tornam os programas mais fáceis de ler. Juntos, eles são conhecidos como **espaços em branco**.
- O compilador ignora espaços em branco.
- Utilize linhas e espaços em branco para aprimorar a legibilidade do programa.

Erros de programação

Erro comum de programação 2.1

Esquecer um dos delimitadores de um comentário tradicional no estilo Javadoc causa um erro de sintaxe. A ***sintaxe*** *de uma linguagem de programação especifica as regras para criar programas apropriados nessa linguagem, assim como as regras de gramática de uma língua natural especificam a estrutura da frase. Um* ***erro de sintaxe*** *ocorre quando o compilador encontra o código que viola as regras da linguagem do Java (isto é, sua sintaxe). Nesse caso, o compilador emite uma mensagem de erro e impede o programa de compilar. Erros de sintaxe também são chamados* ***erros de compilador***, ***erros em tempo de compilação*** *ou* ***erros de compilação***, *porque o compilador detecta-os durante a fase de compilação.*

Identificadores e classes

- • As **palavras-chave** são reservadas para uso pelo Java e sempre são escritas com todas as letras minúsculas.
- • A palavra-chave `class` introduz uma **declaração de classe**.
- • Por convenção, todos os **nomes de classes** em Java começam com uma letra maiúscula e apresentam a letra inicial de cada palavra que eles
- incluem em maiúscula (por exemplo, `SampleClassName`).

Identificadores e classes

- Nomes de classe

  Por convenção, iniciam com uma letra maiúscula e apresentam a letra inicial de cada palavra que eles incluem em maiúscula (por exemplo, SampleClassName).

  O nome de uma classe Java é um **identificador** — uma série de caracteres que consiste em letras, dígitos, sublinhados (_) e sinais de cifrão ($) que não iniciem com um dígito e não contenham espaços.

  O Java faz **distinção entre maiúsculas e minúsculas** — letras maiúsculas e letras minúsculas são diferentes — portanto, a1 e A1 são identificadores diferentes (mas ambos válidos).

Identificadores e classes

- O nome de uma classe Java é um **identificador** – uma série de caracteres consistindo em letras, dígitos, sublinhados (_) e sinais de cifrão ($) que não iniciem com um dígito nem contenham espaços.

- O Java faz **distinção entre maiúsculas e minúsculas**.

- O **corpo** de cada declaração de classe é delimitado por chaves, { e }.

Identificadores e classes

- Os métodos realizam tarefas e retornam informações ao concluí-las.
- A palavra-chave `void` indica que um método executará uma tarefa, mas não retornará nenhuma informação.
- As instruções instruem o computador a realizar ações.
- Uma string entre aspas duplas é às vezes chamada de string de caracteres ou string literal.

Identificadores e classes

- O objeto de saída padrão (`System.out`) exibe caracteres na janela de comando.

- O método `System.out.println` exibe seu argumento na janela de comando seguido por um caractere de nova linha para posicionar o cursor de saída no começo da próxima linha.

Executando o primeiro programa em Java

- ***Compilando e executando seu primeiro Aplicativo Java.***

  Abra uma janela de prompt de comando e mude para diretório onde programa está armazenado.

  Muitos sistemas operacionais utilizam o comando `cd` para mudar de diretório.

  Para compilar o programa, digite

  ```
  javac Welcome1.java
  ```

  Se o programa não contiver nenhum erro de sintaxe, o comando anterior cria um novo arquivo chamado `.class` (conhecido como o **arquivo de classe**) contendo os bytecodes Java independentes de plataforma que representam o aplicativo.

  Quando utilizamos o comando `java` para executar o aplicativo em uma dada plataforma, esses bytecodes serão traduzidos pela JVM em instruções que são entendidas pelo sistema operacional subjacente.

Executando o primeiro programa em Java

- Para executar o programa, digite `java Welcome1`.
- Isso carrega a JVM, que carrega o arquivo `.class` para a classe `Welcome1`.
- Observe que a extensão do nome de arquivo `.class` é omitida do comando precedente; caso contrário, a JVM não executará o programa.
- A JVM chama o método `main` para executar o programa.

Modificando o programa em Java

- • `System.out.print` exibe seu argumento e posiciona o cursor de saída imediatamente após o último caractere exibido.

- • Uma barra invertida (`\`) em uma string é um caractere de escape. O Java combina-o com o próximo caractere para formar uma sequência de escape. A sequência de escape `\n` representa o caractere de nova linha.

Modificando o programa em Java

- A classe `welcome2`, mostrada na Figura 2.3, utiliza duas instruções para produzir a mesma saída mostrada na Figura 2.1.
- Novos recursos e os principais recursos em cada listagem de código são destacados em amarelo.
- O método `print` de System.out exibe uma string.
- Diferente de `println`, `print` não posiciona o cursor de saída no início da próxima linha na janela de comando.

  O próximo caractere que o programa exibe aparecerá imediatamente depois do último caractere que `print` exibe.

Modificando o programa em Java

```
1  // Figura 2.3: Welcome2.java
2  // Imprimindo uma linha de texto com múltiplas instruções.
3
4  public class Welcome2
5  {
6     // método principal inicia a execução do aplicativo Java
7     public static void main( String[] args )
8     {
9        System.out.print( "Welcome to " );
10       System.out.println( "Java Programming!" );
11    } // fim do método main
12 } // fim da classe Welcome2
```

Imprime `welcome to` e deixa o cursor na mesma linha

Imprime `Java Programming!` iniciando onde o cursor estava posicionado e, então, um caractere de nova linha

```
Welcome to Java Programming!
```

Figura 2.3 | Imprimindo uma linha de texto com múltiplas instruções.

Caracteres de Escape

Sequência de escape	Descrição
`\n`	Nova linha. Posiciona o cursor de tela no início da próxima linha.
`\t`	Tabulação horizontal. Move o cursor de tela para a próxima parada de tabulação.
`\r`	Retorno de carro. Posiciona o cursor da tela no início da linha atual — não avança para a próxima linha. Qualquer saída de caracteres depois do retorno de carro sobrescreve a saída de caracteres anteriormente gerados na linha atual.
`\\`	Barras invertidas. Utilizada para imprimir um caractere de barra invertida.

`\"`	Aspas duplas. Utilizada para imprimir um caractere de aspas duplas. Por exemplo,
	`System.out.println( "\"in quotes\"" );`
	exibe
	`"in quotes"`

Figura 2.5 | Algumas sequências de escape comuns.

Declaração import

- Uma declaração `import` ajuda o compilador a localizar uma classe que é usada em um programa.

- O rico conjunto do Java de classes predefinidas é agrupado em pacotes – chamados de grupos de classes. Esses são referidos como biblioteca de classes Java, ou Interface de Programação de Aplicativo Java (API Java).

Operadores

- **Operadores aritméticos**
- **O asterisco** (*) indica a multiplicação.
- O sinal de porcentagem (**%**) é o **operador de resto.**
- Os operadores aritméticos são operadores binários porque cada um deles opera em dois operandos.
- A **divisão de inteiros** produz um quociente inteiro.

  Qualquer parte fracionária na divisão de inteiros é simplesmente descartada (isto é, truncada) — nenhum arredondamento ocorre.
- O operador de módulo, %, produz o resto depois da divisão.

Operadores Aritméticos

Operação Java	Operador	Expressão algébrica	Expressão Java
Adição	+	$f + 7$	`f + 7`
Subtração	-	$p - c$	`p - c`
Multiplicação	*	bm	`b * m`
Divisão	/	x/y ou $\frac{x}{y}$ ou $x \div y$	`x / y`
Resto	%	$r \bmod s$	`r % s`

Figura 2.11 | Operadores aritméticos.

Operadores

- Operadores de igualdade (== e !=)
- Operadores relacionais (>, <, >= e <=)
- Os dois operadores de igualdade têm o mesmo nível de precedência, que é mais baixo que o dos operadores relacionais.
- Os operadores de igualdade são associados da esquerda para a direita.
- Todos os operadores relacionais têm o mesmo nível de precedência e também são associados da esquerda para a direita.

Operadores

Operador algébrico	Operador de igualdade ou relacional Java	Exemplo de condição em Java	Significado da condição em Java
Operadores de igualdade			
=	==	x == y	x é igual a y
≠	!=	x != y	x é não igual a y
Operadores relacionais			
>	>	x > y	x é maior que y
<	<	x < y	x é menor que y
≥	>=	x >= y	x é maior que ou igual a y
≤	<=	x <= y	x é menor que ou igual a y

Figura 2.14 | Operadores de igualdade e operadores relacionais.

Referências

- Deitel, Harvey M. **Java C o m o Programar**, 8ªed tradução Edson Furmankiewicz ; São Paulo, Editora: Pearson Prentice Hall, 2010.