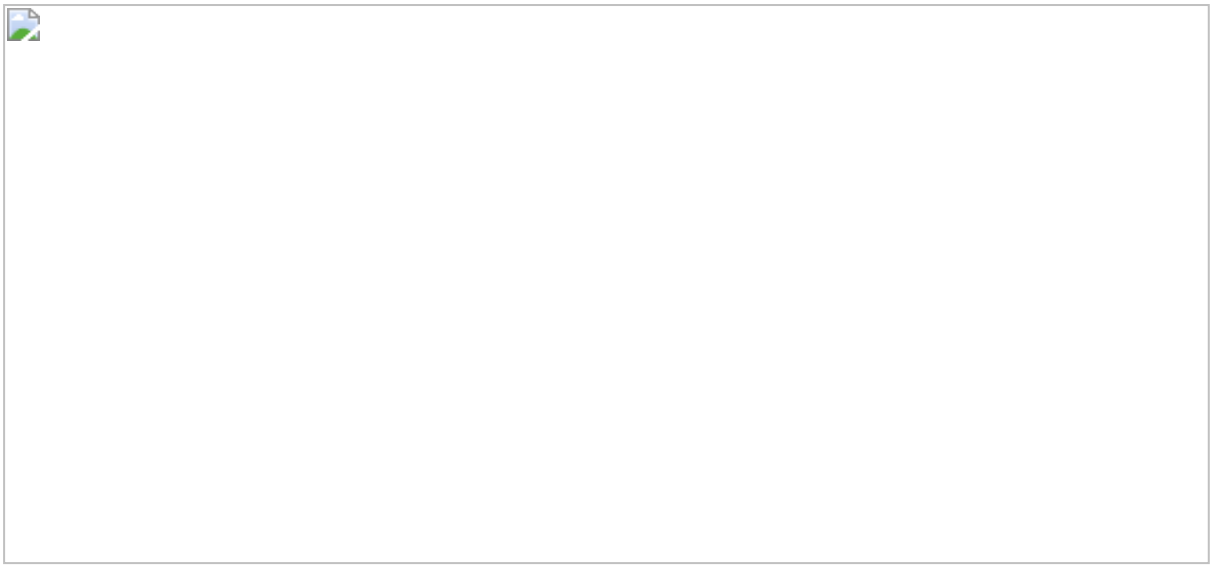

Introdução aos Aplicativos Java

O Java tornou-se a linguagem preferida para implementar aplicativos baseados na Internet e software para dispositivos que se comunicam por uma rede. Equipamentos de som estéreo e outros dispositivos domésticos muitas vezes são conectados em rede pela tecnologia Java. Atualmente, existem bilhões de celulares e dispositivos portáteis compatíveis com Java! O Java é a linguagem preferida para atender às necessidades de programação de muitas organizações.

Java Standard Edition (Java SE)

Java Enterprise Edition (Java EE) é adequada para desenvolver aplicativos distribuídos em rede em larga escala e aplicativos baseados na Web.

Java Micro Edition (Java ME) é voltada para o desenvolvimento de aplicativos de pequenos dispositivos com limitações de memória.

Em geral, programas Java passam por cinco fases - edição, compilação, carregamento, verificação e execução.

Fonte: Deitel, Java: Como Programar.

A máquina virtual (***Virtual Machine*** - ***VM***) é um aplicativo de software que simula um computador, mas oculta o sistema operacional e o hardware subjacentes dos programas que interagem com ela. A JVM é umas das máquinas virtuais mais amplamente utilizadas.

Ao contrário da linguagem de máquina, que é dependente do hardware específico de computador, os *bytecodes* são independentes de plataforma - eles não dependem de uma plataforma de hardware particular. Portanto, os *bytecodes* do Java são portáteis - sem recompilar o código-fonte, os mesmos *bytecodes* podem executar em qualquer plataforma contendo uma JVM que entende a versão do Java em que os *bytecodes* foram compilados. A JVM é invocada pelo comando java.

Na Fase 5, a JVM executa os *bytecodes* do programa, realizando assim as ações especificadas pelo programa. Nas primeiras versões do Java, a JVM era simplesmente um interpretador para *bytecodes* Java. Isso fazia com que a maioria dos programas Java executasse lentamente porque a JVM interpretava e executava um *bytecode* por vez. Em geral, as JVMs atuais executam *bytecodes* utilizando uma combinação de interpretação e a chamada **compilação Just-In-Time (JIT)**. Nesse processo, a JVM analisa os *bytecodes* à medida que eles são interpretados, procurando **hot spots (pontos ativos)** — partes dos *bytecodes* que executam com frequência. Para essas partes, um **compilador Just-In-Time** (**JIT**) — conhecido como **compilador Java HotSpot** — traduz os *bytecodes* para a linguagem de máquina de um computador subjacente. Quando a JVM encontra novamente essas partes compiladas, o código de linguagem de máquina mais rápido é executado. Portanto, os programas Java na realidade passam por duas fases de compilação — uma em que código-fonte é traduzido em bytecodes (para a portabilidade entre JVMs em diferentes plataformas de computador) e uma segunda em que, durante a execução, os bytecodes são traduzidos em linguagem de máquina para o computador real em que o programa é executado.

Nosso primeiro programa Java: Imprimindo uma linha de texto

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

```
// Programa de impressão de texto
// Isso é um comentário
/*Isso também é um comentário
mas pode cobrir várias linhas*/
```

```
public class Welcome
{

    // método principal inicia a execução do aplicativo Java
    public static void main(String[] args)
    {
        System.out.println("Hello World!");
    } // fim do método main
} // fim da classe Welcome
```

Declaração de classe:

- A palavra-chave class introduz uma declaração de classe;
- Os nomes de classes iniciam com uma letra maiúscula e apresentam a letra inicial de cada palavra que eles incluem em maiúscula (ex.: ExemploNomeClasse);
- É um identificador, não inicia com um dígito e não contém espaços (ex.: BemVindo1, $valor, _valor, botao7);
- Distinção entre letras maiúsculas e minúsculas (a1 é diferente de A1).

Declaração do Método:

- Um dos métodos deve ser chamado main e deve ser definido como está aqui. Caso contrário, a JVM não executará o aplicativo;
- A palavra-chave void indica que esse método não devolverá nenhuma informação;
- `String[] args` entre parênteses é uma parte requerida da declaração do método main.

O objeto `System.out` é conhecido como o objeto de saída padrão. O método `System.out.println` exibe uma linha de texto na janela de comando. A String entre parênteses é o argumento para o método.

Estamos agora pronto para compilar e executar o nosso programa. Abra uma janela de comando e vá para o diretório onde o programa está armazenado. Se o programa não contiver nenhum erro de sintaxe, o comando anterior cria um novo arquivo chama **Welcome.class** que

contém os *bytecodes* Java independentes de plataforma que representam nosso 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.

Exibindo uma linha de texto com múltiplas instruções

```
// Imprimindo uma linha de texto com múltiplas instruções

public class Welcome
{
    // método principal inicia a execução do aplicativo Java
    public static void main(String[] args)
    {
        System.out.print("Hello World ");
        System.out.println("UnDF");
        } //fim do método main
} // fim da classe Welcome
```

Exibindo múltiplas linhas de texto com uma única instrução

```
// Imprimndo múltiplas linhas de texto com uma única instruçã
public class Welcome
{
    // método principal inicia a execução do aplicativo Java
    public static void main(String[] args)
    {
        System.out.println(" Hello\n World\n e\n UnDF\n");
    } //fim do método main
} // fim da classe Welcome
```

Sequência de escape	Descrição
\n	Nova linha.
\t	Tabulação horizontal.

Sequência de escape	Descrição
\r	Posiciona o cursor da tela no início da linha atual.
\\	Barras invertidas.
\”	Aspas duplas.

Entrada de dados pelo usuário

```
// Programa de adição que exibe a soma de dois números.
import java.util.Scanner; //programa utiliza a classe Scanner

public class Addition
{
    // método principal inicia a execução do aplicativo Java
    public static void main(String[] args)
    {
        // cria um Scanner para obter entrada da janela de co
        Scanner entrada = new Scanner(System.in);

        int num1;
        int num2;
        int soma;

        System.out.print("Entre com o primeiro número: ");
        num1 = entrada.nextInt(); //lê o primeiro o número fo

        System.out.print("Entre com o segundo número: ");
        num2 = entrada.nextInt(); //lê o segundo o número for

        soma = num1 + num2;

        System.out.println("A soma é " + soma);
    } // fim do método main
} // fim da classe Addition
```

`Import java.util.Scanner;` é uma declaração ***import*** que ajuda o compilador a localizar uma classe utilizada nesse programa. Indica que usa a classe ***Scanner*** predefinida do Java do pacote ***java.util.***

A declaração `Scanner entra = new Scanner(System.in);` especifica que a variável nomeada entrada seja do tipo ***Scanner***. Um ***Scanner*** permite a um programa ler os dados para utilização em um programa. O sinal de igual (=) indica que a variável ***Scanner*** entrada deve ser *inicializada* na sua declaração com o resultado da expressão ***new Scanner(System.in)*** à direita do sinal de igual. A expressão utiliza a palavra-chave new para cria um objeto Scanner que lê caracteres digitados pelo usuário no teclado.

`num1 = entrada.nextInt();` utiliza o método ***nextInt*** do valor de entrada do objeto ***Scanner*** para obter um inteiro digitalizado pelo usuário.

Para cada nova classe da Java API que usamos, indicamos o pacote em que ela está localizada. Uma versão baseada na Web dessa documentação pode ser obtida em:

`java.sun.com/javase/6/docs/api/`

Aritmética

Operação Java	Operador	Ordem de procedência
Adição	+	Avaliado segundo. Elas serão aplicadas da esquerda para a direita;
Subtração	-	
Multiplicação	*	Avaliado primeiro. Elas serão aplicadas da esquerda para a direita;
Divisão	/	
Resto	%	
Atribuição	=	Avaliado por último.Uma condição é uma expressão que pode ser true ou false.

Operador de igualdade e operadores relacionais

Operador relacional algébrico	Operador relacional Java
=	==
≠	!=
<	<
≥	>==
≤	<==

```
// Programa que compara números entradas do usuário
public class Comparison
{
    public static void main(String[] args)
    {
        // cria um Scanner para obter entrada da janela de co
        Scanner entrada = new Scanner(System.in);

        int num1;
        int num2;

        System.out.print("Entre com o primeiro número: ");
        num1 = entrada.nextInt(); //lê o primeiro o número fo

        System.out.print("Entre com o segundo número: ");
        num2 = entrada.nextInt(); //lê o segundo o número for

        if (num1 > num2)
        {
            System.out.println("O primeiro número é maior que
        }
        if (num1 < num2)
        {
            System.out.println("O segundo número é maior que
        }
        if (num1 == num2)
        {
            System.out.println("O primeiro número é igual ao
        }
```

```
    }
}
```

Exercícios da Aula

1. Escreva um aplicativo que solicita ao usuário inserir dois inteiros, obtém do usuário esses números e imprime sua soma, produto, diferença e divisão. Resolução.
2. Escreva um aplicativo que exibe uma caixa, uma oval, uma seta e um losango utilizando asteriscos (*). Resolução.
3. Escreva um aplicativo que lê um inteiro, determina e imprime se ele é ímpar ou par. Resolução.
4. Escreva um aplicativo que insere um número consistindo em cinco dígitos do usuário, separa o número em seus dígitos individuais e imprime os dígitos separados uns dos outros por três espaços cada. Por exemplo, se o usuário digitar o número 42339, o programa deve imprimir:

   4   2   3   3   9. Resolução.