
Conceitos Básicos da Linguagem Java

Sumário

- Introdução;
- Histórico do Java;
- Características do Java;
- Compilação de um Programa em Java;
- Modificando o Primeiro Programa em Java;
- Declaração e Leitura de Variáveis;
- Estruturas de Seleção;
 - if;
 - switch;

Sumário

- Estruturas de Repetição;
 - while;
 - do...while
 - for;
- Os comandos `break` e `continue`;
- Array;
- Métodos;
- Bibliografia.

Histórico do Java

- A **Sun Microsystems**, em 1991, percebeu que os microprocessadores iriam modificar profundamente a vida das pessoas;
- Dessa forma, financiou um projeto de pesquisa (codinome Green) que **resultou na linguagem de programação baseada no C++** que seu criador, James Gosling, chamou de Oak;
- Mais tarde descobriu-se que já existia uma linguagem de programação **chamada Oak**;

Histórico do Java

- Quando uma equipe da Sun visitou uma cafeteira local, o nome **Java** (cidade de origem de um tipo de café importado) foi sugerido; e o nome pegou;
- O projeto Green previa o **avanço dos dispositivos eletrônicos inteligentes** mas no início dos anos 90 isso não ocorreu;
- Felizmente, a World Wide Web explodiu em popularidade em 1993 e a equipe da Sun viu o imediato **potencial de utilizar Java para:**
 - Adicionar conteúdo dinâmico;
 - Interatividade;
 - Animações às páginas Web...

Histórico do Java

- A Sun anunciou o Java formalmente em uma importante conferência em maio de 1995;
- O Java logo chamou atenção da comunidade de negócios por causa do enorme interesse na WWW;
- Hoje o Java é utilizado para:
 - Desenvolver aplicativos corporativos de grande porte;
 - Aprimorar a funcionalidade de servidores Web;
 - Fornecer aplicativos para dispositivos voltados para o consumo popular (celulares, PDAs, etc);

Características da Linguagem Java

■ **Compilada:**

- Um programa em Java é compilado para o chamado "bytecode", que é próximo as instruções de máquina. O bytecode é um código de uma máquina virtual idealizada pelos criadores da linguagem;

■ **Portável:**

- Java foi criada para ser portável. O "bytecode" gerado pelo compilador para a sua aplicação específica pode ser transportado entre plataformas distintas que suportam Java;

Características da Linguagem Java

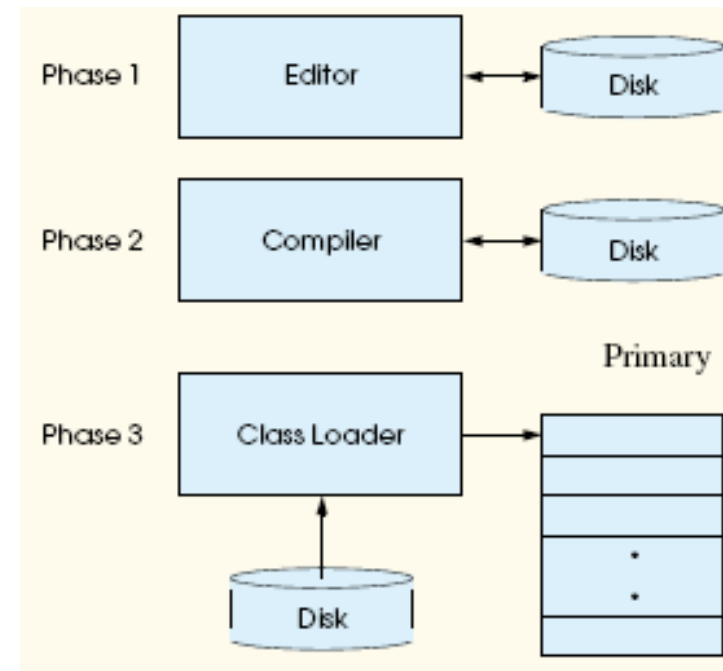
- **Orientada a Objetos:**
 - Java suporta herança, mas não herança múltipla compensada pelo uso de interfaces;
- **Segurança:**
 - Presença de coleta automática de lixo, que evita erros comuns ao acessar diretamente a memória.
 - Presença de mecanismos de tratamento de exceções que torna as aplicações mais robustas, não permitindo que elas abortem, mesmo quando rodando sob condições anormais;

Características da Linguagem Java

- **Concorrência:**
 - Permite, de maneira fácil, a criação de vários "threads" de execução;
- **Eficiente:**
 - Exige pouco espaço, pouca memória comparando-se com outras linguagens de scripting existentes;
 - Ao passo que é 20 vezes mais lenta que C (caindo gradativamente a cada ano);
- **Suporte para Programação de Sistemas Distribuídos:**
 - Java fornece facilidade para programação com sockets, chamada remota de procedimentos, tcp-ip, etc;

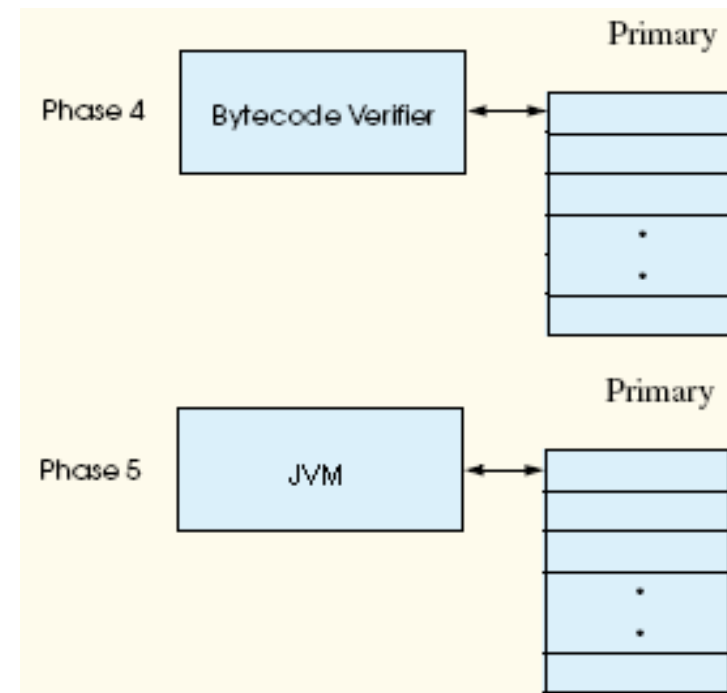
Compilação de um Programa em Java

- **Fase 1:** Consiste em editar um arquivo com um programa ou ambiente de desenvolvimento integrado (*Integrated Development Environments - IDE*)
- **Fase 2:** Compilação do programa. O compilador converte o código-fonte Java em bytecodes
- **Fase 3:** O programa é carregado na memória para ser executado posteriormente;



Compilação de um Programa em Java

- **Fase 4:** Verificação do bytecode. O verificador de bytecode examina seus bytecodes para assegurar que eles são válidos e que não violam restrições de segurança do Java;
- **Fase 5:** Execução. A máquina virtual do Java executa os bytecodes do programa. Combinação de interpretação e a chamada compilação just-in-time(JIT);



Primeiro Programa em Java

- Analisemos, agora, um simples programa em Java:

Nome do programa.
Obrigatoriamente
deverá ter uma classe
do tipo `public` com o
mesmo nome no arquivo.

Declaração de classe
definida pelo
programador.

```
1  /*
2  * Main.java
3  *
4  * Created on 3 de Setembro de 2007, 18:19
5  *
6  * To change this template, choose Tools | Template Manager
7  * and open the template in the editor.
8  */
9
10 package helloworld;
11
12 /**
13  *
14  * @author Leonardo
15  */
16 public class Main {
17
18     /** Creates a new instance of Main */
19     public Main() {
20     }
21
22     /**
23      * @param args the command line arguments
24      */
25     public static void main(String[] args) {
26         System.out.println("!!!Hello World!!!");
27         // TODO code application logic here
28     }
29
30 }
31
```

Comentário de múltiplas
linhas `/* comentário */`
gerado pela ferramenta
NetBeans para auxiliar a
documentação do arquivo.

Pacote. Usado para ajudar os
programadores a gerenciar
os códigos e facilitar a
reutilização de software.

Primeiro Programa em Java

- Analisemos, agora, um simples programa em Java:

```
1  /*
2  * Main.java
3  *
4  * Created on 3 de Setembro de 2007, 18:19
5  *
6  * To change this template, choose Tools | Template Manager
7  * and open the template in the editor.
8  */
9
10 package helloworld;
11
12 /**
13  *
14  * @author Leonardo
15  */
16 public class Main {
17
18     /** Creates a new instance of Main */
19     public Main() {
20     }
21
22     /**
23      * @param args the command line arguments
24      */
25     public static void main(String[] args) {
26         System.out.println("!!!Hello World!!!");
27         // TODO code application logic here
28     }
29
30 }
31
```

Ponto de partida de cada aplicativo Java. Os parênteses indicam que a linha define um método. Em Java, um dos métodos deve ter o nome main.

Construtor do objeto. Utilizado para iniciar o objeto da classe Main quando ele for criado.

Instrução para impressão de string (contida entre aspas duplas) na janela de comando na qual o aplicativo Java é executado.

Primeiro Programa em Java

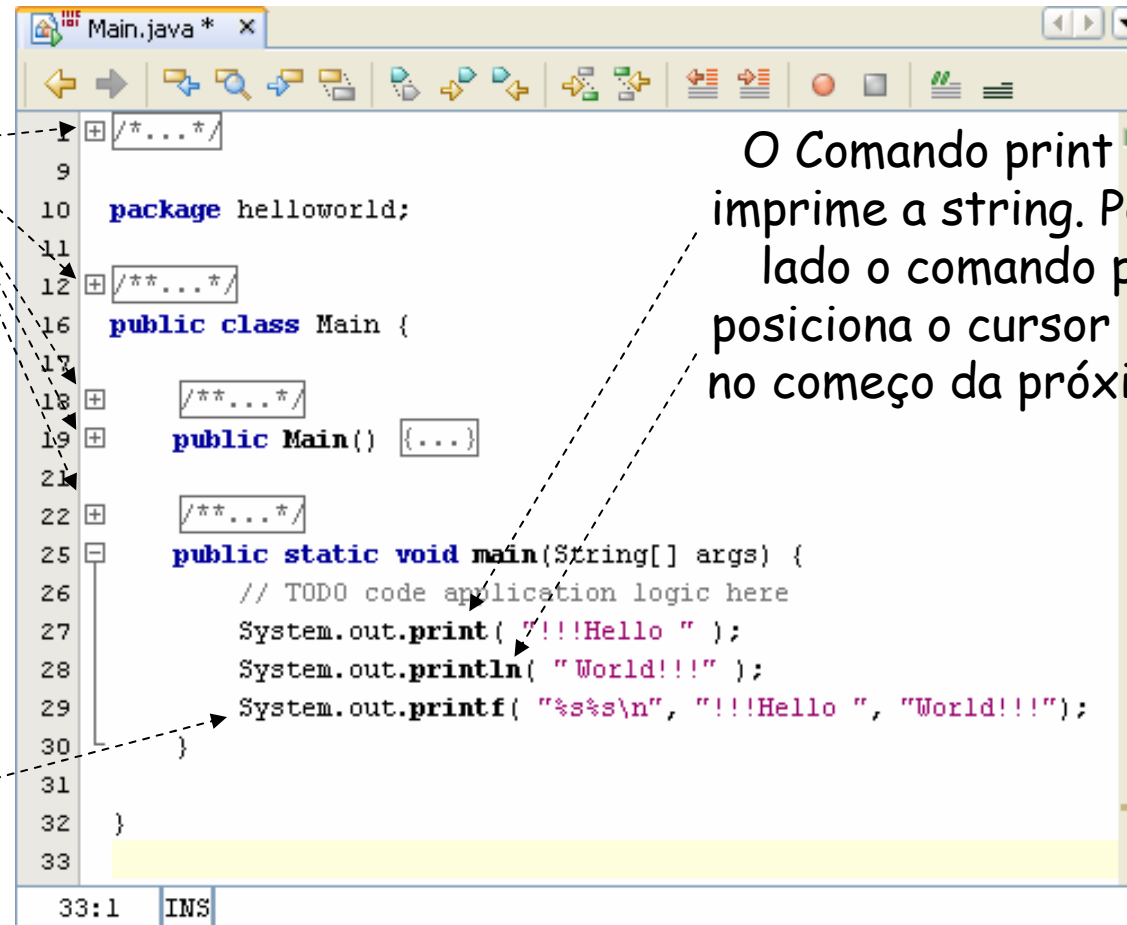
■ Considerações:

- ❑ O nome de uma classe Java é um identificador - uma série de letras, dígitos, sublinhados (_) e sinais de cifrão (\$) que não iniciem com um dígito e não contenham espaços;
- ❑ **Por convenção**, todos os nomes de classes em Java iniciam com uma letra maiúscula e apresentam a letra inicial de cada palavra que eles incluem em maiúsculo (ex: ImprimeNome);
- ❑ Java faz distinção entre **letras maiúsculas e minúsculas**, assim a1 e A1 são identificadores diferentes;
- ❑ Caso o arquivo .java não contenha um método chamado main a JVM **não executará o aplicativo**;
- ❑ A chave esquerda, {, inicia um bloco. A chave direita, }, finaliza um bloco;

Modificando o Primeiro Programa em Java

Utilitário da ferramenta que suprime o código em cada bloco.

Novo recurso do J2SE 5.0 que exibe dados formatados.



```
9
10 package helloworld;
11
12 /**...*/
13
14 public class Main {
15
16     /**...*/
17
18     public Main() {...}
19
20     /**...*/
21
22     public static void main(String[] args) {
23         // TODO code application logic here
24         System.out.print( "!!!Hello " );
25         System.out.println( " World!!!" );
26         System.out.printf( "%s%s\n", "!!!Hello ", "World!!!");
27     }
28
29 }
30
31
32
33
```

The screenshot shows a code editor window titled 'Main.java *'. The code is as follows:

```
9
10 package helloworld;
11
12 /**...*/
13
14 public class Main {
15
16     /**...*/
17
18     public Main() {...}
19
20     /**...*/
21
22     public static void main(String[] args) {
23         // TODO code application logic here
24         System.out.print( "!!!Hello " );
25         System.out.println( " World!!!" );
26         System.out.printf( "%s%s\n", "!!!Hello ", "World!!!");
27     }
28
29 }
30
31
32
33
```

Annotations in the image include:

- A dashed arrow pointing to the 'package helloworld;' line from the text 'Utilitário da ferramenta que suprime o código em cada bloco.'
- A dashed arrow pointing to the 'System.out.printf' line from the text 'Novo recurso do J2SE 5.0 que exibe dados formatados.'
- A dashed arrow pointing to the 'System.out.println' line from the text 'O Comando print apenas imprime a string. Por outro lado o comando println posiciona o cursor de saída no começo da próxima linha'.

O Comando print apenas imprime a string. Por outro lado o comando println posiciona o cursor de saída no começo da próxima linha

Modificando o Primeiro Programa em Java

- Saída no console para o programa anterior:

■ Saída - HelloWorld (run)

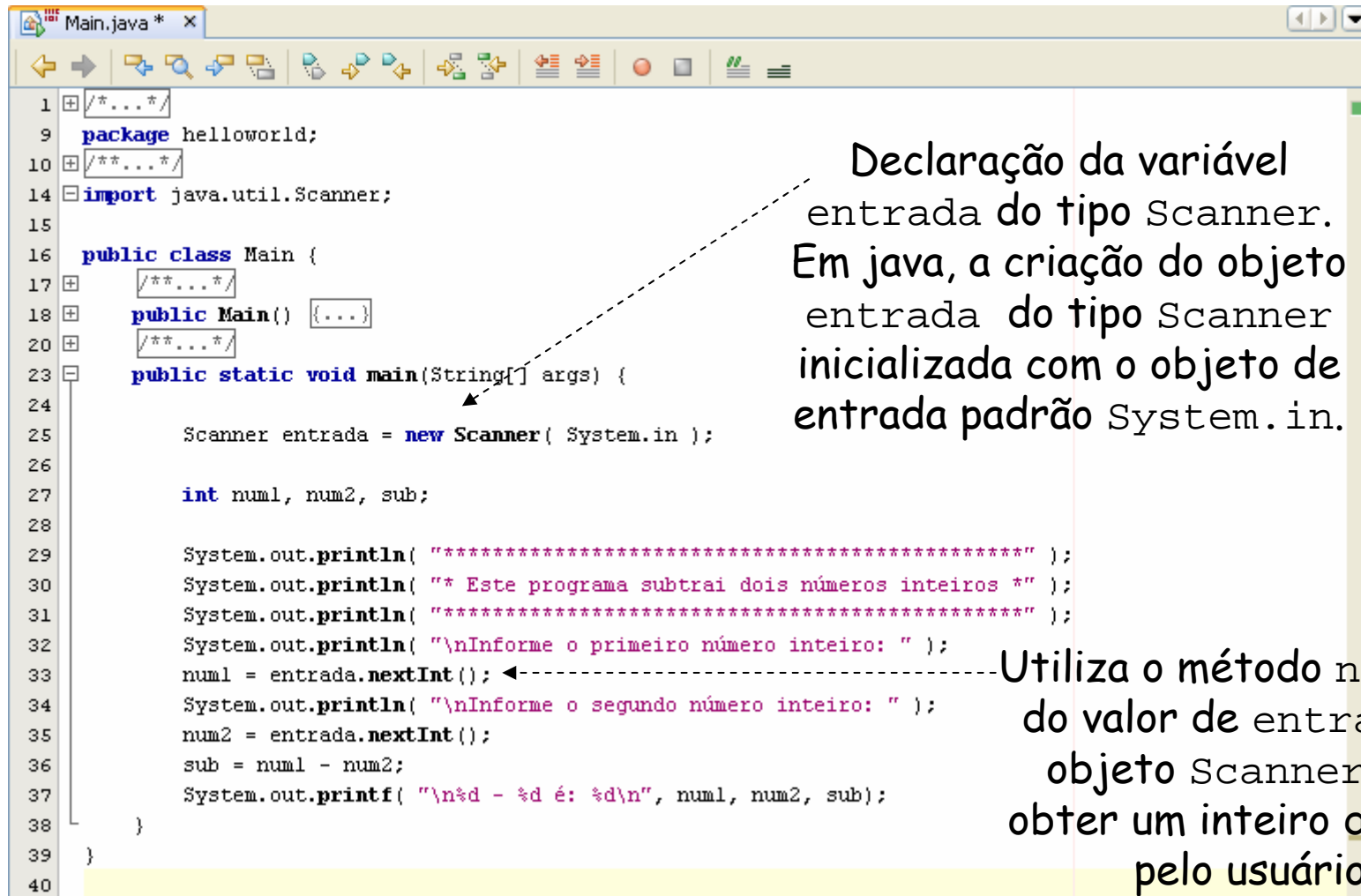
```
init:  
deps-jar:  
compile:  
run:  
!!!Hello World!!!  
!!!Hello World!!!  
EXECUTADO COM SUCESSO (tempo total: 0 segundos)|
```


Modificando o Primeiro Programa em Java

- Quando aparece uma barra invertida em uma string de caracteres, o Java combina o próximo caractere com as barras invertidas para formar uma **seqüência de scape**:

| Seqüência de Scape | Descrição |
|--------------------|----------------------|
| \n | Nova linha |
| \r | Retorno de carro |
| \t | Tabulação horizontal |
| \" | Aspas duplas |
| \\ | Barra invertida |

Declaração e Leitura de Variáveis



```
1  /*...*/
9  package helloworld;
10 /*...*/
14 import java.util.Scanner;
15
16 public class Main {
17     /*...*/
18     public Main() { ... }
19     /*...*/
23     public static void main(String[] args) {
24
25         Scanner entrada = new Scanner( System.in );
26
27         int num1, num2, sub;
28
29         System.out.println( "*****" );
30         System.out.println( "* Este programa subtrai dois números inteiros *" );
31         System.out.println( "*****" );
32         System.out.println( "\nInforme o primeiro número inteiro: " );
33         num1 = entrada.nextInt();
34         System.out.println( "\nInforme o segundo número inteiro: " );
35         num2 = entrada.nextInt();
36         sub = num1 - num2;
37         System.out.printf( "\n%d - %d é: %d\n", num1, num2, sub);
38     }
39 }
40
```

Declaração da variável entrada do tipo Scanner. Em java, a criação do objeto entrada do tipo Scanner inicializada com o objeto de entrada padrão System.in.

Utiliza o método nextInt do valor de entrada do objeto Scanner para obter um inteiro digitado pelo usuário.

Declaração e Leitura de Variáveis

- Saída no console para o programa anterior:

```
Saída - HelloWorld (run)
init:
deps-jar:
compile:
run:
*****
* Este programa subtrai dois números inteiros *
*****

Informe o primeiro número inteiro:
90

Informe o segundo número inteiro:
80

90 - 80 é: 10
EXECUTADO COM SUCESSO (tempo total: 8 segundos)
```

Declaração e Leitura de Variáveis

- Os tipos de dados primitivos da linguagem Java são:

| Tipo de Dados | Tamanho em bits | Considerações |
|---------------------|-----------------|--|
| <code>double</code> | 64 | |
| <code>float</code> | 32 | |
| <code>long</code> | 64 | -2^{63} a $+2^{63}-1$ |
| <code>int</code> | 32 | -2^{31} a $+2^{31}-1$ |
| <code>short</code> | 16 | -32768 a $+32767$ |
| <code>byte</code> | 8 | -128 a $+127$ |
| <code>char</code> | 16 | '\u0000' a '\uFFFF' (0 a 65535) |
| <code>bool</code> | | false ou true a depender da JVM em cada plataforma |

Estruturas de Seleção - if

- Uma condição é uma expressão que pode ser verdadeira ou falsa;
- As condições nas instruções if pode, ser formadas utilizando os operadores de igualdade (== e !=) e os operadores relacionais (<, >, <= e >=);
- Vejamos um exemplo de sua utilização:

Estruturas de Seleção - if

```
12 import javax.swing.JOptionPane;
13 /**...*/
17 public class Main {
18     /**...*/
21     public static void main(String[] args) {
22         int faltasInt;
23
24         String apresentacao =
25             String.format( "Este programa informa a situação das faltas de um aluno em P00" );
26         JOptionPane.showMessageDialog( null, apresentacao );
27
28         String faltas =
29             JOptionPane.showInputDialog( "Informe o número de faltas do aluno" );
30
31         faltasInt = Integer.parseInt( faltas );
32
33         if( faltasInt > 15)
34         {
35             JOptionPane.showMessageDialog( null, "O aluno atingiu o limite de 15 falta, portanto, " +
36                 "está REPROVADO por falta" );
37         }
38         else
39         {
40             JOptionPane.showMessageDialog( null, "Não atingiu o limite de 15 faltas, portanto, " +
41                 "permanece apto a cursar a disciplina" );
42         }
43     }
44 }
45
```

Pacote javax.swing que contém inúmeras classes e métodos destinadas a interface.

Exibe uma caixa de diálogo que contenha uma mensagem.

Estruturas de Seleção - if

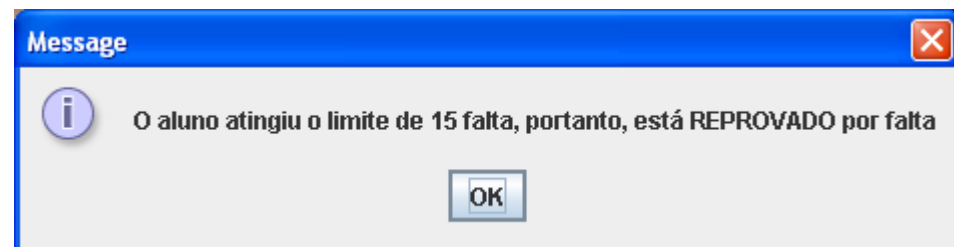
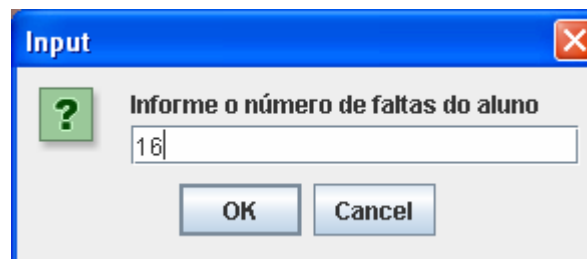
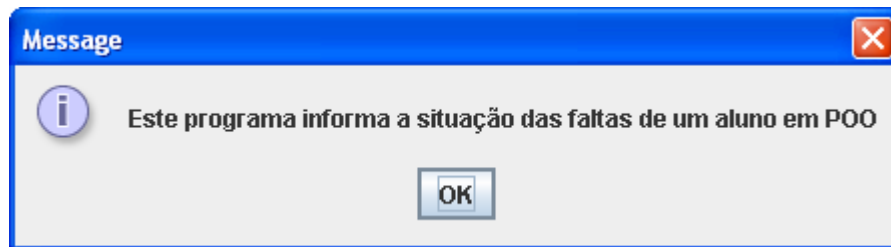
```
12 import javax.swing.JOptionPane;
13 /**...*/
17 public class Main {
18     /**...*/
21     public static void main(String[] args) {
22         int faltasInt;
23
24         String apresentacao =
25             String.format( "Este programa informa a situação das faltas de um aluno em P00" );
26         JOptionPane.showMessageDialog( null, apresentacao );
27
28         String faltas =
29             JOptionPane.showInputDialog( "Informe o número de faltas do aluno" );
30
31         faltasInt = Integer.parseInt( faltas );
32
33         if( faltasInt > 15)
34         {
35             JOptionPane.showMessageDialog( null, "O aluno atingiu o limite de 15 falta, portanto, " +
36                 "está REPROVADO por falta" );
37         }
38         else
39         {
40             JOptionPane.showMessageDialog( null, "Não atingiu o limite de 15 faltas, portanto, " +
41                 "permanece apto a cursar a disciplina" );
42         }
43     }
44 }
45
```

Exibe um diálogo de entrada que permite ao usuário inserir dados.

Método parseInt utilizado para converter a String em um int.

Estruturas de Seleção - if

- Saída para o programa anterior:



Estruturas de Seleção - if

■ Considerações:

- ❑ O Pacote `javax.swing` contém inúmeras classes que ajudam os programadores em Java a criar interfaces gráficas com usuário (*graphical user interfaces - GUIs*) para aplicações;
- ❑ `showMessageDialog` é um método especial da classe `JOptionPane` chamado método `static`. Dessa forma, não exige a criação de um objeto;
- ❑ O método `format` retorna uma `String` formatada, mas não exibe na tela;

Estruturas de Seleção - switch

```
14 import java.util.Scanner;
15 public class selecao {
16     public static void main( String args[] )
17     {
18         int faltas;
19         Scanner entrada = new Scanner ( System.in );
20
21         System.out.println( "Este Programa informa o conceito (A, B, C ou D) de um aluno em P00" );
22         System.out.println( "Informe o nome do aluno: " );
23         String nome = entrada.nextLine();
24         System.out.println( "Informe a média final do aluno (0 -100): " );
25         faltas = entrada.nextInt();
26         switch(faltas/10)
27         {
28             case 9:
29             case 10:
30                 System.out.printf( "O conceito do aluno %s foi A!!!!", nome);
31                 break;
32             case 8:
33                 System.out.printf( "O conceito do aluno %s foi B!!!!", nome);
34                 break;
35             case 7:
36                 System.out.printf( "O conceito do aluno %s foi C!!!!", nome);
37                 break;
38             default:
39                 System.out.printf( "O conceito do aluno %s foi D!!!!", nome);
40                 break;
41         }
42     }
43 }
```

O método `nextLine` lê caracteres até que um caractere de nova linha seja encontrado (Enter).

Estruturas de Seleção - switch

- Saída para o programa anterior:

```
Este Programa informa o conceito (A, B, C ou D) de um aluno em POO
Informe o nome do aluno:
Luis
Informe a média final do aluno (0 -100):
85
O conceito do aluno Luis foi B!!!!
EXECUTADO COM SUCESSO (tempo total: 13 segundos)|
```

Estrutura de Repetição - While

```
1 public class repeticao
2 {
3     public static void main( String args[] )
4     {
5         int count = 1;
6
7         while ( count <= 10 )
8         {
9             System.out.println( count % 2 == 1 ? "*****" : "++++++" );
10            ++count;
11        } // fim do while
12    } // fim de main
13
14 } // fim da classe repeticao
15
```

Operador Ternário

run-single:

```
****
++++++
****
++++++
****
++++++
****
++++++
****
++++++
EXECUTADO COM SUCESSO (tempo total: 0 segundos)
```

Estrutura de Repetição - do...while

```
1 public class repeticao
2 {
3     public static void main( String args[] )
4     {
5         int counter = 1; // inicializa o contador
6
7         do
8         {
9             System.out.printf( "%d ", counter );
10            ++counter;
11        } while ( counter <= 10 ); // fim da instrução do...while
12
13        System.out.println(); // gera a saída de um caractere nova linha
14    } // fim de main
15 } // fim da classe repeticao
```

↳ Saída - Aula_02 (run-single)

```
init:
deps-jar:
compile-single:
run-single:
1 2 3 4 5 6 7 8 9 10
EXECUTADO COM SUCESSO (tempo total: 0 segundos)
```

Estrutura de Repetição - for

```
1 public class repeticao
2 {
3     public static void main( String args[] )
4     {
5         double amount; // quantia em depósito ao fim de cada ano
6         double principal = 1000.0; // quantidade inicial antes dos juros
7         double rate = 0.05; // taxa de juros
8
9         // exibe cabeçalhos
10        System.out.printf( "%s%20s \n", "ANO", "Valor do Depósito" );
11
12        // calcula quantidade de depósito para cada um dos dez anos
13        for ( int year = 1; year <= 10; year++ )
14        {
15            // calcula nova quantidade durante ano especificado
16            amount = principal * Math.pow( 1.0 + rate, year );
17
18            // exibe o ano e a quantidade
19            System.out.printf( "%4d%,20.2f\n", year, amount );
20        } // for final
21    } // fim de main
22 } // fim da classe repeticao
```

Método static
chamado
especificando apenas
o nome da classe

Estrutura de Repetição - for

- Saída para o programa anterior:

```
run-single:
ANO  Valor do Depósito
  1      1.050,00
  2      1.102,50
  3      1.157,63
  4      1.215,51
  5      1.276,28
  6      1.340,10
  7      1.407,10
  8      1.477,46
  9      1.551,33
 10      1.628,89
EXECUTADO COM SUCESSO (tempo total: 0 segundos)|
```

Os comandos break e continue

- A instrução break pára a execução de um bloco, vejamos:

```
1 public class repeticao
2 {
3     public static void main( String args[] )
4     {
5         int count; // variável de controle também utilizada depois que loop
6
7         for ( count = 1; count <= 10; count++ ) // faz o loop 10 vezes
8         {
9             if ( count == 5 ) // se contagem for 5,
10                break;        // termina o loop
11
12                System.out.printf( "%d ", count );
13            } // for final
14
15            System.out.printf( "\nBroke out of loop at count = %d\n", count );
16        } // fim de main
17    } // fim da classe repeticao com break
18
```

run-single:

1 2 3 4

Broke out of loop at count = 5

EXECUTADO COM SUCESSO (tempo total: 0 segundos)

Os comandos break e continue

- Por outro lado, o comando continue pula as instruções restantes de um bloco para executar o próximo loop;

```
1 public class repeticao
2 {
3     public static void main( String args[] )
4     {
5         for ( int count = 1; count <= 10; count++ ) // faz o loop 10 vezes
6         {
7             if ( count == 5 ) // se contagem for 5,
8                 continue; // pula o código restante no loop
9
10            System.out.printf( "%d ", count );
11        } // for final
12
13        System.out.println( "\nUsed continue to skip printing 5" );
14    } // fim de main
15 } // fim da classe repetição com continue
16
```

```
run-single:
1 2 3 4 6 7 8 9 10
Used continue to skip printing 5
EXECUTADO COM SUCESSO (tempo total: 0 segundos)
```

Array

- Um array é um grupo de variáveis (elementos ou componentes) que contém valores que são todos do mesmo tipo;

```
1 public class exemploArray
2 {
3     public static void main( String args[] )
4     {
5         int array[]; // declara o array identificado
6         array = new int[ 10 ]; // cria o espaço para o array
7
8         System.out.printf( "%s%8s\n", "Index", "Value" ); // títulos de colu
9
10        // gera saída do valor de cada elemento do array
11        for ( int counter = 0; counter < array.length; counter++ )
12            System.out.printf( "%5d%8d\n", counter, array[ counter ] );
13    } // fim de main
14 } // fim da classe exemploArray
```

Declaração do array

Criação do objeto

run-single:

| Index | Value |
|-------|-------|
| 0 | 0 |
| 1 | 0 |
| 2 | 0 |
| 3 | 0 |
| 4 | 0 |
| 5 | 0 |
| 6 | 0 |
| 7 | 0 |
| 8 | 0 |
| 9 | 0 |

EXECUTADO COM S

Array

- Outras formas de declarar e inicializar um Array:

```
1 public class exemploArray
2 {
3     public static void main( String args[] )
4     {
5         // array de respostas da pesquisa
6         int responses[] = { 1, 2, 6, 4, 8, 5, 9, 7, 8, 10, 1, 6, 3, 8, 6,
7             10, 3, 8, 2, 7, 6, 5, 7, 6, 8, 6, 7, 5, 6, 6, 5, 6, 7, 5, 6,
8             4, 8, 6, 8, 10 };
9         int frequency[] = new int[ 11 ]; // array de contadores de frequência
10
11         // para cada resposta, seleciona o elemento da respostas e utiliza esse valor
12         // como índice de frequência para determinar o elemento a incrementar
13         for ( int answer = 0; answer < responses.length; answer++ )
14             ++frequency[ responses[ answer ] ];
15
16         System.out.printf( "%s%10s", "Rating", "Frequency" );
17
18         // gera saída do valor de cada elemento do array
19         for ( int rating = 1; rating < frequency.length; rating++ )
20             System.out.printf( "%6d%10d\n", rating, frequency[ rating ] );
21     } // fim de main
22 } // fim da classe exemploArray
```

Método que retorna o tamanho do array

Array

- A saída para o programa anterior:

```
┆ Saída - Aula_02 (run-single)
run-single:
RatingFrequency
    1          2
    2          2
    3          2
    4          2
    5          5
    6         11
    7          5
    8          7
    9          1
   10          3
EXECUTADO COM SUCESSO (
```

Métodos

- Há três tipos de módulos em Java - métodos, classes e pacotes; (Próxima Aula)
- Os métodos, mais conhecidos como funções ou procedimentos em outras linguagens de programação, permitem a **modularização do código**;
- Há vários motivos para modularizar um código, os principais são:
 - Dividir para conquistar;
 - Reutilização;

Bibliografia

- Deitel, H. M. & Deitel, P. J. *Java: como programar*, Editora Bookman. 6ª ed. São Paulo: 2005.