



## Introdução à Linguagem Java

Prof. Mário Augusto Pazoti  
pazoti@icmc.usp.br



## P1: Meu primeiro programa Java

```
1 // Fig. 2.1: Exemplo1.java
2 //
3
4 public class Exemplo1 {
5
6     // método principal
7     public static void main( String args[] )
8     {
9         System.out.println( "Meu primeiro programa java!" );
10    } // final do método principal
11
12 } // final da classe Exemplo1
13
14 Meu primeiro programa Java!
```



## P1: Modificando meu programa Java

```
1 // Fig. 2.3: Exemplo2.java
2 //
3
4 public class Exemplo2 {
5
6     // método principal
7     public static void main( String args[] )
8     {
9         System.out.print( "Meu primeiro " );
10        System.out.println( " programa Java" );
11    } // final do método principal
12
13 } // final da classe Exemplo2
14
```

System.out.print mantém o cursor na mesma linha, enquanto, System.out.println coloca o cursor na próxima linha



## P1: Modificando meu programa Java

```
1 // Fig. 2.4: Exemplo3.java
2 //
3
4 public class Exemplo3 {
5
6     // método principal
7     public static void main( String args[] )
8     {
9         System.out.println( "Meu\nprimeiro\nprograma\njava!" );
10    } // end method main
11
12 } // end class Welcome3
13
```

Meu  
primeiro  
programa  
java!

Uma nova linha é inserida na saída quando o operador '\n' é utilizado.

### Caractere Especiais

\n nova linha  
\r enter  
\t tabulação (tab)  
\b retrocesso  
\" aspas  
\\ barra



## Fluxo de Controle

- **Objetivo:** Determinar o próximo comando a ser executado
- Fluxo de controle puramente **seqüencial**
  - Próximo comando na seqüência
- Fluxo de controle **condicional**
  - Comandos de seleção
- Fluxo de controle **iterativo**
  - Comandos de repetição



## Operadores Relacionais: IF

Padrões algébricos de equivalência ou Operadores relacionais	Operadores relacionais em Java	Exemplo of Java condition	Significado da condição em Java
Equivalência			
=	==	x == y	x é igual a y
	!=	x != y	x é diferente de y
Operadores relacionais			
>	>	x > y	x é maior que y
<	<	x < y	x é menor que y
>=	>=	x >= y	x é maior ou igual a y
<=	<=	x <= y	x é menor ou igual a y



## Comandos de Seleção simples e dupla

- Seleção simples: if (teste) comando  

```
if ((arg1 == 20) && (arg2 != 10))
    System.out.println(" algo ");
```
- Seleção dupla: if (teste) comando else comando  

```
if (teste == true){// comandos}
else{//comandos}
```
- Importante:** a seleção também pode ser usada como expressão condicional  

```
String ms = i==1 ? "um objeto" : "objetos";
```

teste

true

false



## Seleção múltipla

- Seleção múltipla: switch (seletor){casos}  

```
switch(variavel){
    case 1: //trecho de código
        break;
    case 2: //trecho de código
        return 0;
    default: // nenhuma opção anterior
        return -1;
}
```



## Comandos de iteração

- for: valor inicial, teste de fim, passo ( +, -)  

```
for(k=0; k<100; k++){//faz algo}
```
- while: executa o teste no início do laço  

```
while(k<100){ //faz algo}
```
- do-while: executa o teste no final do laço  

```
do{ //faz algo}
while(k<100);
```



## Tipos Primitivos de Dados

Type	Size in bits	Values	Standard
boolean		true or false <small>[Note: The representation of a boolean is specific to the Java Virtual Machine on each computer platform.]</small>	
char	16	'\u0000' to '\uFFFF' (0 to 65535)	(ISO Unicode character set)
byte	8	-128 to +127 ( $-2^7$ to $2^7 - 1$ )	
short	16	-32,768 to +32,767 ( $-2^{15}$ to $2^{15} - 1$ )	
int	32	-2,147,483,648 to +2,147,483,647 ( $-2^{31}$ to $2^{31} - 1$ )	
long	64	-9,223,372,036,854,775,808 to +9,223,372,036,854,775,807 ( $-2^{63}$ to $2^{63} - 1$ )	
float	32	Negative range: -3.4028234663852886E+38 to -1.40129846432481707E-45 Positive range: 1.40129846432481707E-45 to 3.4028234663852886E+38	(IEEE 754 floating point)
double	64	Negative range: -1.7976931348623157E+308 to -4.94065645841246544E-324 Positive range: 4.94065645841246544E-324 to 1.7976931348623157E+308	(IEEE 754 floating point)



## Convenção para nomes

- Java, como C/C++ distingue entre letras maiúsculas e minúsculas  
 – Exemplo: *fruta difere de Fruta*
- Nomes de classes iniciam com maiúscula  
 – Exemplo: *class Fruta*
- Nomes de variáveis iniciam com minúsculas  
 – Exemplo: *int calorias*
- Nomes de métodos são verbos que iniciam com minúscula e após usam maiúsculas  
 – Exemplo: *alteraPeso*



## Classe Java – apenas o método principal

```
class Benvindo {
    public static void main (String[] args) {
        System.out.println("Benvindo a Java.");
    }
}
```

Cuidar maiúsculas e minúsculas!

Método main: onde inicia a execução



## Classe Java – com dois métodos

```
class Benvindo {
    static void imprime(){
        System.out.println
            ("Benvindo a Java.");
    }
    public static void main(String[] args) {
        imprime();
    }
}
```



## Exercícios práticos

- Criar um diretório para seus programas
- Localizar o ambiente
- Copiar/abrir o o arquivo com a lista de exercícios
- Fazer os exercícios seguindo o roteiro



## P2: Exibindo Texto em uma caixa de dialogo

- Exibição
  - A maioria das aplicações Java utilizam windows ou caixas de dialogo (dialog box)
    - Até agora nós utilizamos as command window
  - A classe JOptionPane permite utilizar dialog boxes
- Packages
  - Conjunto pré-definido de classes para utilização
  - Grupos de classes relacionadas são chamadas de packages
    - Grupos de todos os packages são conhecidos como Java class library ou Java applications programming interface (Java API)
  - A classes JOptionPane esta no package javax.swing
    - Essa classe é utilizada na construção de Graphical User Interfaces (GUIs)



## P2: Exibindo Texto em uma caixa de dialogo

```
1 // Exemplo4.java
2 // Imprimindo multiplas linhas em uma caixa de dialogo.
3
4 // Java packages
5 import javax.swing.JOptionPane; // programa utiliza JOptionPane
6
7 public class Exemplo4 {
8
9     // método principal
10    public static void main( String args[] )
11    {
12        JOptionPane.showMessageDialog(
13            null, "Meu\nprimeiro\nprograma\nem\nJava!" );
14
15        System.exit( 0 ); // fim da aplicação
16
17    } // fim do método principal
18
19 } // fim da classe Exemplo4
```



## P2: Exibindo Texto em uma caixa de dialogo

- ```
4 // Java packages
```
- Grupos de packages da Java API
  - Core packages (principais)
    - Começam com java
    - Inclusos no Java 2 Software Development Kit
  - Extension packages (extensões)
    - Começam com javax
    - Novos Java packages
- ```
5 import javax.swing.JOptionPane; // program uses JOptionPane
```
- declaração import
    - Utilizadas para o compilador localizar as classes externas utilizadas pelo programa java.
    - Faz o compilador ler a classes JOptionPane do package javax.swing



## P2: Exibindo Texto em uma caixa de dialogo

```
12 JOptionPane.showMessageDialog(
13     null, "welcome\nto\nJava\nProgramming!" );
```

- Chamada do método showMessageDialog da classes JOptionPane
  - São necessários dois argumentos
  - Múltiplos argumentos são separados por virgulas (,)
  - Por enquanto, o primeiro argumento será sempre null
  - O segundo argumento é a string que será exibida
- showMessageDialog é um static método da classes JOptionPane
  - Static: chamada do método utilizando o nome classe, ponto (.), o nome do método.

**P3:**

**Somando números Inteiros**

```

1 // Soma.java
2 // Programa que soma dois números inteiros
3
4 // Java packages
5 import javax.swing.JOptionPane; // program uses JOptionPane
6
7 public class Soma{
8
9     // método principal
10    public static void main( String args[] )
11    {
12        String firstNumber; // 1a. String digitada pelo usuário
13        String secondNumber; // 2a. String digitada pelo usuário
14
15        int number1; // primeiro número para soma
16        int number2; // segundo número para soma
17        int sum; // soma de number1 e number2
18
19        // leitura do primeiro número
20        firstNumber = JOptionPane.showInputDialog( "Primeiro número: " );
21
22        // leitura do segundo número
23        secondNumber =
24            JOptionPane.showInputDialog( "Segundo número: " );
25
26        // conversão dos números para string
27        number1 = Integer.parseInt( firstNumber );
28        number2 = Integer.parseInt( secondNumber );
29
30        // soma dos números
31        sum = number1 + number2;
32
33    }
34 }

```

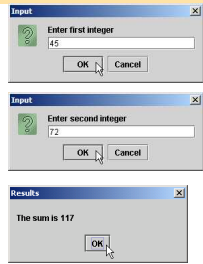
**P3:**

**Somando números Inteiros**

```

33 // exibir resultado
34 JOptionPane.showMessageDialog( null, "A soma é " + sum,
35     "Resultado", JOptionPane.PLAIN_MESSAGE );
36
37 System.exit( 0 ); // terminate application with window
38
39 } // end method main
40
41 } // end class Addition

```



**P3: Somando números inteiros**

**JAVA**

```

5 import javax.swing.JOptionPane; // program uses JOptionPane

```

- Import da classe `JOptionPane`

```

7 public class Soma {

```

- Começo `public` da classe `Soma`
  - Nome do arquivo deve ser `Soma.java`
- Linhas 10-11: `main`

```

12 String firstNumber; // first string entered by user
13 String secondNumber; // second string entered by user

```

- Declarações
  - `firstNumber` e `secondNumber` são variáveis `String`

**P3: Somando números inteiros**

**JAVA**

```

12 String firstNumber; // primeira string digitada pelo usuário
13 String secondNumber; // segunda string digitada pelo usuário

```

- Variáveis
  - Armazenam informações na memória
    - Declaração : <tipo> depois <nome>
  - `firstNumber` e `secondNumber` são do tipo `String` (package `java.lang`)
  - Nome das variáveis: qualquer identificador válido
  - Declaração termina com ;

```

String firstNumber, secondNumber;

```

- É possível declarar múltiplas variáveis do mesmo tipo ao mesmo tempo
- Utiliza-se vírgula (,) para separar
- Pode ser adicionada comentário para descrever o propósito das variáveis

**P3: Somando números inteiros**

**JAVA**

```

15 int number1; // primeiro número
16 int number2; // segundo número
17 int sum; // soma de number1 e number2

```

- Declaração das variáveis `number1`, `number2`, e `sum` do tipo `int`
  - `int` armazena valores inteiros. Ex: 0, -4, 97
  - Tipos `float` e `double` também podem armazenar valores decimais
  - Tipo `char` pode armazenar caracteres simples como: x, \$, \n

**P3: Somando números inteiros**


**JAVA**

```

20 firstNumber = JOptionPane.showInputDialog( "Enter first integer" );

```

- Leitura da `String` digitada pelo usuário, representando o primeiro número a ser somado
  - Método `JOptionPane.showInputDialog` exibe o seguinte:



```

20 firstNumber = JOptionPane.showInputDialog( "Enter first integer" );

```

- Resultado da chamada `showInputDialog` é armazenada na variável `firstNumber` utilizando o operador =

**P3: Somando números inteiros**

```

27 number1 = Integer.parseInt( firstNumber );
28 number2 = Integer.parseInt( secondNumber );

```

- Método `Integer.parseInt`
  - Converte uma `String` em um número inteiro (tipo `int`)
    - Classe `Integer` do `java.lang`
  - Inteiro retornado em `Integer.parseInt` é associado a variável `number1` (linha 27)
    - Lembrando: `number1` é declarada como sendo do tipo `int`

```

34 JOptionPane.showMessageDialog( null, "The sum is " + sum,
35 "Results", JOptionPane.PLAIN_MESSAGE );

```

- Utilização do método `showMessageDialog` para exibir o resultado
- "The sum is " + `sum`
  - Usa o operador `+` to concatenar a string "The sum is " com a variável `sum`
  - Se `sum` contém 117, então "The sum is " + `sum` exibirá a string "The sum is 117"

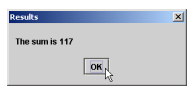
**P3: Somando números inteiros**

```





34 JOptionPane.showMessageDialog( null, "The sum is " + sum,
35 "Results", JOptionPane.PLAIN_MESSAGE );

```

- Versão diferente do `showMessageDialog`
  - São necessários quatro argumentos
  - Primeiro argumento é: `null`
  - Segundo: string de exibição
  - Terceiro: string da barra de título
  - Quarto: tipo do ícone da caixa de diálogo
    - Linha 35: `JOptionPane.PLAIN_MESSAGE`



**P3: Somando números inteiros**

Tipo da caixa de diálogo	Ícone	Descrição
<code>JOptionPane.ERROR_MESSAGE</code>		Ícone para mensagens de erro
<code>JOptionPane.INFORMATION_MESSAGE</code>		Ícone para mensagens de informação
<code>JOptionPane.WARNING_MESSAGE</code>		Ícone para alertar o usuário sobre potenciais problema
<code>JOptionPane.QUESTION_MESSAGE</code>		Ícone para questionar o usuário. Normalmente utilizado como resposta, com os botões de Sim ou Não.
<code>JOptionPane.PLAIN_MESSAGE</code>	no icon	Não exibe nenhum ícone

**P4: Utilizando o operador if**

```

1 // Comparacao.java
2 // Compara numeros inteiros utilizando o comando if
3 // e operadores relacionais.
4
5 // Java packages
6 import javax.swing.JOptionPane;
7
8 public class Comparacao {
9
10 // método principal
11 public static void main( String args[] )
12 {
13     String firstNumber; //
14     String secondNumber; //
15     String result; // string de resultado que contém a saída
16
17     int number1; // 1o. Numero para comparacao
18     int number2; // 2o. Numero para comparacao
19
20 // leitura do primeiro numero
21 firstNumber = JOptionPane.showInputDialog( "Enter first integer:" );
22
23 // leitura do segundo numero
24 secondNumber =
25     JOptionPane.showInputDialog( "Enter second integer:" );
26
27 // conversao de string para int
28 number1 = Integer.parseInt( firstNumber );
29 number2 = Integer.parseInt( secondNumber );
30
31 // inicializa a variavel result como vazio
32 result = "";
33


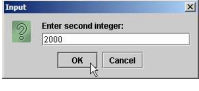
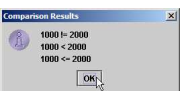
```

```

34 if ( number1 == number2 )
35     result = result + number1 + " == " + number2;
36
37 if ( number1 != number2 )
38     result = result + number1 + " != " + number2;
39
40 if ( number1 < number2 )
41     result = result + "\n" + number1 + " < " + number2;
42
43 if ( number1 > number2 )
44     result = result + "\n" + number1 + " > " + number2;
45
46 if ( number1 <= number2 )
47     result = result + "\n" + number1 + " <= " + number2;
48
49 if ( number1 >= number2 )
50     result = result + "\n" + number1 + " >= " + number2;
51
52 // exibir resultados
53 JOptionPane.showMessageDialog( null, result, "Resultado ",
54     JOptionPane.INFORMATION_MESSAGE );
55
56 System.exit( 0 ); // fim da aplicacao
57
58 } // fim do metodo principal
59
60 } // fim da classe comparacao

```

**P4: Utilizando o operador if**



## Um pouco de Java e Orientação a Objetos (OO)



### Orientação a Objetos em Java

```
classe Pessoa
{
    String nome;
    String endereço;
    int cep;
    int cpf;
    Public cadastraPessoa(String Nome, String Endereço,
                          int cep, int cpf);
    Public exibeDadosPessoa();
}
```



### Orientação a Objetos em Java

- **Classe:** Pessoa
- **Objetos:** são instâncias da classe Pessoa
- **Atributos:** nome, endereço, cep, cpf
- **Métodos:** cadastraPessoa, exibeDadosPessoa
- **Encapsulamento:** operadores utilizados para trabalhar com o objeto. Por exemplo, o usuário não precisa saber o que acontece com o método exibeDadosPessoa, apenas como utilizá-lo.



### Orientação a Objetos em Java

- **Herança:** simplifica a declaração de novas classes pois é baseada em definições de classes já definidas.

```
classe Estudante (Pessoa)
{
    int RA;
    String Curso;
}

classe Funcionário(Pessoa)
{
    String função;
    String departamento;
}
```



### Orientação a Objetos em Java

- **Polimorfismo:** possibilita instanciar um mesmo objeto de várias formas. Por exemplo:

*CriarJanela()*  
→ Cria uma janela com tamanho pré-definido

*CriarJanela(100,100)*  
→ Cria uma janela com tamanho de 100x100

*CriarJanela(100,100, center)*  
→ Cria uma janela com tamanho de 100x100 e alinhamento no centro



### POO 01: Exemplo OO em Java

```
class ExemploOO_01
{
    public static void main (String arg[])
    {
        Pessoa aux;

        aux = new Pessoa();

        aux.cadastraPessoa("João", "Silva", 40);

        //aux.nome = "João ";
        //aux.sobrenome = "Roberto ";
        //aux.idade = 40;

        aux.exibeDadosPessoa();

        //System.out.println( "Nome: " + aux.nome);
        //System.out.println( "Sobrenome: " + aux.sobrenome);
        //System.out.println( "Idade: " + aux.idade);
    }
}
```



## POO 01: Exemplo OO em Java

```
class Pessoa
{
    String nome;
    String sobrenome;
    int idade;

    public void cadastraPessoa(String no,String sob, int id)
    {
        nome = no;
        sobrenome = sob;
        idade = id;
    }

    public void exibedadosPessoa()
    {
        System.out.println( "Nome: " + aux.nome);
        System.out.println( "Sobrenome: " + aux.sobrenome);
        System.out.println( "Idade: " + aux.idade);
    }
}
```



## POO 02: Exemplo OO em Java

```
class ExemploOO_02
{
    public static void main (String arg[])
    {
        Contador MeuCont;

        // criar objeto contador (alocacao)
        MeuCont = new Contador();

        // inicializar contador
        MeuCont.init(0);

        MeuCont.imprime();

        MeuCont.incrementa();
        MeuCont.incrementa();
        MeuCont.imprime();

        MeuCont.decrementa();
        MeuCont.decrementa();
        MeuCont.imprime();
    }
}
```



## POO 02: Exemplo OO em Java

```
class Contador
{
    int nro;

    public void init(int ini)
    {
        nro = ini;
    }

    public void incrementa()
    {
        // incremento
        nro = nro + 1; //nro++;
    }

    public void decrementa()
    {
        // decremento
        nro = nro -1; //nro--;
    }

    public void imprime()
    {
        System.out.println("Contador: " + nro );
    }
}
```