



PROGRAMAÇÃO ORIENTADA A OBJETOS [POO]

Material 03 – POO_03

Prof. Mestre Marcos Roberto de Moraes [Maromo]

Recursos da Linguagem Java

Recursos Básicos da Linguagem Java

- ❑ Variáveis e Atributos
- ❑ Tipos primitivos e Tipos de dados Referência
- ❑ Operadores Aritméticos
- ❑ Operadores Unários
- ❑ Operadores Relacionais e Operadores Lógicos
- ❑ Operadores de Atribuição
- ❑ Estruturas de Controle de Fluxo
- ❑ Estruturas de Repetição
- ❑ Comandos break, continue, entrada e saída
- ❑ Exercícios



Variáveis em Java

- ❑ Representa um endereço de memória.
- ❑ Quando definimos um tipos de variável consideramos o seu nome, o tipo e um valor de inicialização.
- ❑ É definida dentro de um método, enquanto um atributo é definido dentro de uma Classe.
- ❑ Uma variável definida internamente a um método só será visível dentro do método.

Atributos em Java

- ❑ Atributo deve ser criado após a definição da classe, fora de qualquer método. [Campos].
- ❑ Podem ser:
 - ▣ Estáticos (enquanto a classe estiver carregada na JVM.
 - ▣ e não estáticos (enquanto o objeto estiver ativo)

Programa Exemplo:

Atributos e Métodos

- ❑ Crie um novo projeto com o nome de **ProjetoContas**
- ❑ Crie as Classe de acordo com a Representação Abaixo:

ContasCorrente		
f	conta	int
f	agencia	int
f	saldo	double
f	nome	String
f	cpmf	double
m	efetuarSaque(double)	void
m	efetuarDeposito(double)	void

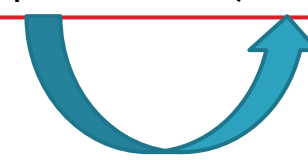
ContasTeste		
m	main(String[])	void

Classe: ContasCorrente

```
public class ContasCorrente {  
    //Atributos  
    public int conta;  
    public int agencia;  
    public double saldo;  
    public String nome;  
    public static double cpmf; //Atributo estatico  
    //Metodos  
    public void efetuarSaque(double valor){  
        this.saldo = this.saldo - valor;  
    }  
    public void efetuarDeposito(double valor){  
        this.saldo = this.saldo + valor;  
    }  
}
```

Na classe ContasCorrente foi definido o atributo cpmf como estático.

Para atribuir valor ao campo, faz-se por meio do nome da Classe. (ContasCorrente.cpmf=0.0038)



Classe: PrincipalConta
Método: main

```
3 public static void main(String[] args) {
4     ContasCorrente conta1 = new ContasCorrente ();
5     //acesso ao atributo estatico por meio do nome da classe
6     ContasCorrente.cpmf = 0.0038;
7     conta1.saldo = 1000;
8     ContasCorrente conta2 = new ContasCorrente ();
9     conta2.saldo = 2000;
10    System.out.println("Manipulando Objetos em Java\u2122");
11    System.out.println("Oracle & Sun Microsystem, Inc.\u00A9");
12    //atributo estatico sendo acessado por um objeto
13    System.out.println("Objeto conta1 - atributo est\u00e1tico: " + conta1.cpmf);
14    System.out.println("Objeto conta1 - atributo n\u00e3o est\u00e1tico: " + conta1.saldo);
15    //atributo estatico sendo acessado por um objeto
16    System.out.println("Objeto conta2 - atributo est\u00e1tico: " + conta2.cpmf);
17    System.out.println("Objeto conta2 - atributo n\u00e3o est\u00e1tico: " + conta2.saldo);
18    conta2.cpmf = 0.0010;
19    //atributo estatico sendo acessado por um objeto
20    System.out.println("Apos altera\u00e7\u00e3o....");
21    System.out.println("Objeto conta1 - atributo est\u00e1tico: " + conta1.cpmf);
22    System.out.println("Objeto conta1 - atributo n\u00e3o est\u00e1tico: " + conta1.saldo);
23    //atributo estatico sendo acessado por um objeto
24    System.out.println("Objeto conta2 - atributo est\u00e1tico: " + conta2.cpmf);
25    System.out.println("Objeto conta2 - atributo n\u00e3o est\u00e1tico: " + conta2.saldo);
26    //Acessando o atributo estatico por meio da classe
27    System.out.println("Objeto conta1 Acessado pelo nome da classe: " + ContasCorrente.cpmf);
28    System.out.println("Objeto conta1 - atributo n\u00e3o est\u00e1tico: " + conta1.saldo);
29 }
```

Resultado da Execução

```
run:
Manipulando Objetos em Java™
Oracle & Sun Microsystem, Inc.®
Objeto conta1 - atributo estático: 0.0038
Objeto conta1 - atributo não estático: 1000.0
Objeto conta2 - atributo estático: 0.0038
Objeto conta2 - atributo não estático: 2000.0
Apos alteração....
Objeto conta1 - atributo estático: 0.0010
Objeto conta1 - atributo não estático: 1000.0
Objeto conta2 - atributo estático: 0.0010
Objeto conta2 - atributo não estático; 2000.0
Objeto conta1 Acessado pelo nome da classe: 0.0010
Objeto conta1 - atributo não estático: 1000.0
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```


Nota sobre o programa anterior

- ❑ No exemplo estamos acessando o atributo `cpmf` por meio de um objeto (`conta2.cpmf=0.0010`) e do nome da classe (`ContasCorrente.cpmf=0.0038`).
- ❑ O acesso por meio do objeto foi usado apenas para demonstrar que, quando alterarmos o valor de um objeto, os outros objetos passam a visualizar o valor alterado.
- ❑ **Boa prática:** devemos sempre acessar um atributo estático por meio do nome da classe.

Tipos Primitivos em Java:

Relembrando

- Tipos primitivos
 - ▣ **boolean**: um valor indicando verdadeiro ou falso.
 - ▣ **byte**: um inteiro de 8 bits (signed).
 - ▣ **char**: um caracter unicode (16-bit unsigned).
 - ▣ **double**: um número de ponto flutuante de 64 bits (signed).
 - ▣ **float**: um número de ponto flutuante de 32 bits (signed).
 - ▣ **int**: um inteiro de 32 bits (signed).
 - ▣ **long**: um inteiro de 64 bits (signed).
 - ▣ **short**: um inteiro de 32 bits (signed)

Tipos de Dados Referência

- ❑ Toda classe criada pelo programador ou pela Sun representa um tipo por referência.
- ❑ Ou seja, quando criamos objetos do tipo *ContasCorrente* estamos criando uma variável do tipo referência.
- ❑ Neste tipo caso o parâmetro seja alterado no método chamado, este terá seu valor alterado também quando retornar ao método que o invocou.

Operadores

Função	Sinal
Adição	+
Subtração	-
Multiplicação	*
Divisão	/
Resto da divisão	%
Operadores Unários	
Incremento	++
Decremento	--

Operadores Relacionais

Função	Sinal
Igual	==
Diferente	!=
Maior que	>
Maior ou igual a	>=
Menor que	<
Menor ou igual a	<=

Operadores Lógicos

Função	Sinal
E	&&
OU	
Não	!

Conversão de tipos

Supondo a variável x	Converter em	y recebe o valor convertido
✓ Entre tipos numéricos		
int x = 10	float	float y = (float) x
int x = 10	double	double y = (double) x
float x = 10.5	int	int y = (int) x
✓ De string para numéricos		
String x = "10"	int	int y = Integer.parseInt(x)
String x = "20.5"	float	float y = Float.parseFloat(x)
String x = "20.5"	double	double y = Double.parseDouble(x)
✓ De numéricos para string		
int x = 10	String	String y = Integer.toString(x) ou String y = String.valueOf(x)
float x = 10.5	String	String y = Float.toString(x) ou String y = String.valueOf(x)
double x = 10.5	String	String y = Double.toString(x) ou String y = String.valueOf(x)

Inserção de Comentários

```
// Comentários em uma única linha
```

```
/* Comentários em  
* várias linhas  
*/
```

```
/** Comentários inseridos no formato reconhecido  
* por um utilitário de documentação chamado javadoc  
* fornecido pela Sun junto com o JDK  
*/
```


Estruturas de Controle de Fluxo

```
if (num1 >= 10) {  
    System.out.println("Condição verdadeira!");  
} else {  
    System.out.println("Condição falsa!");  
}
```

■ Desvios condicionais

```
switch (op) {  
    case 1:  
        System.out.println("Caso op igual a 1...");  
        break;  
    case 2:  
        System.out.println("Caso op igual a 2...");  
        break;  
    case 3:  
        System.out.println("Caso op igual a 3...");  
        break;  
    default:  
        System.out.println("Caso op não seja 1, 2 ou 3");  
        break;  
}
```

Estruturas de Controle de Repetição

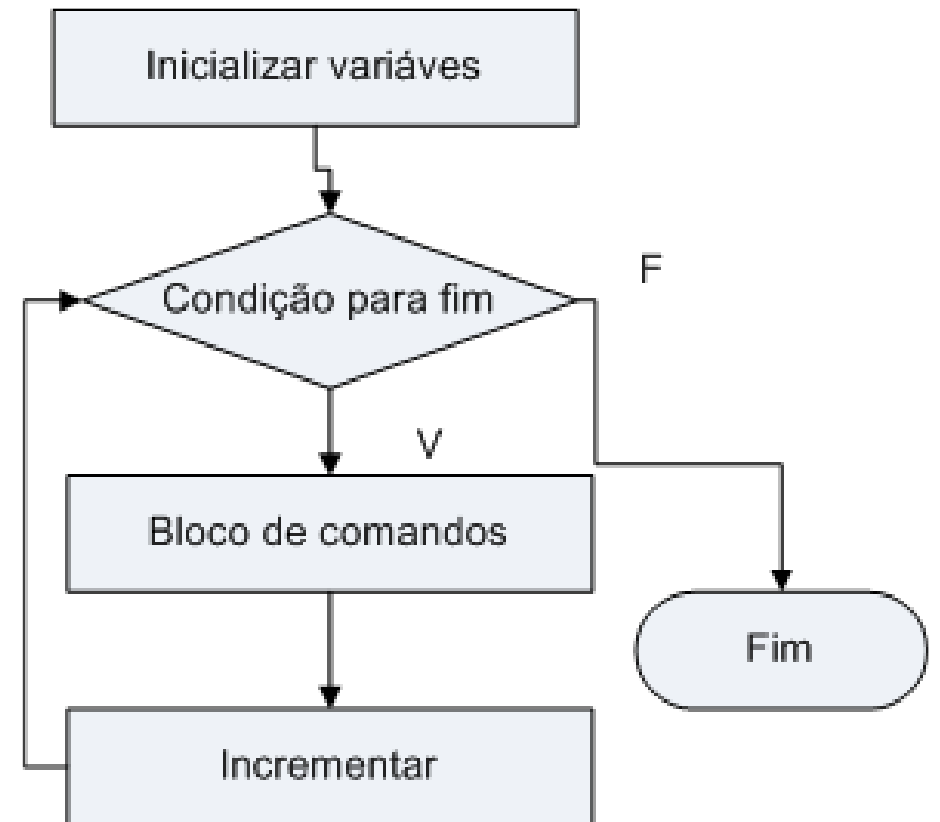
□ Comandos:

- ▣ for
- ▣ while
- ▣ do
- ▣ break
- ▣ continue

Comando for

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

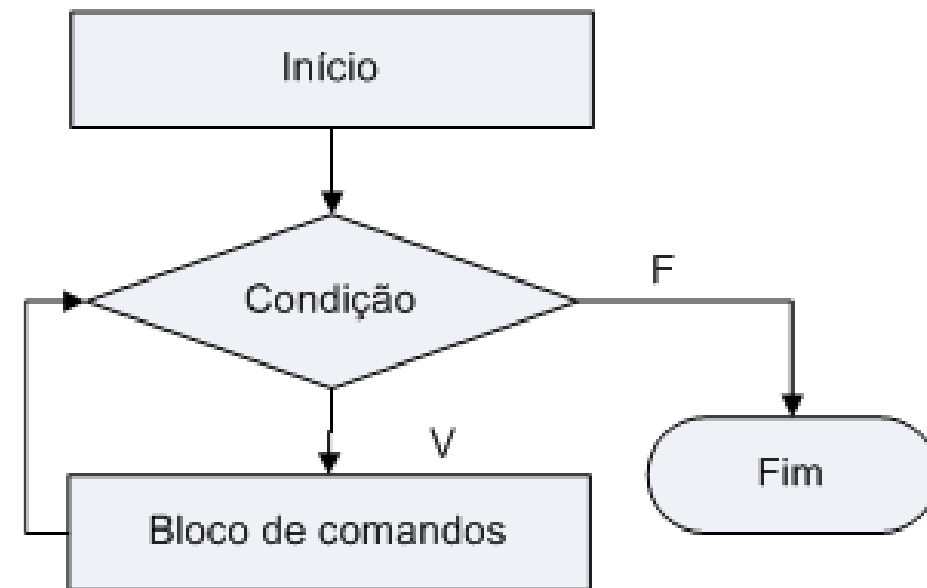
```
for(int i=0; i<10; i++)  
{  
    System.out.println(i);  
}
```



Comando While

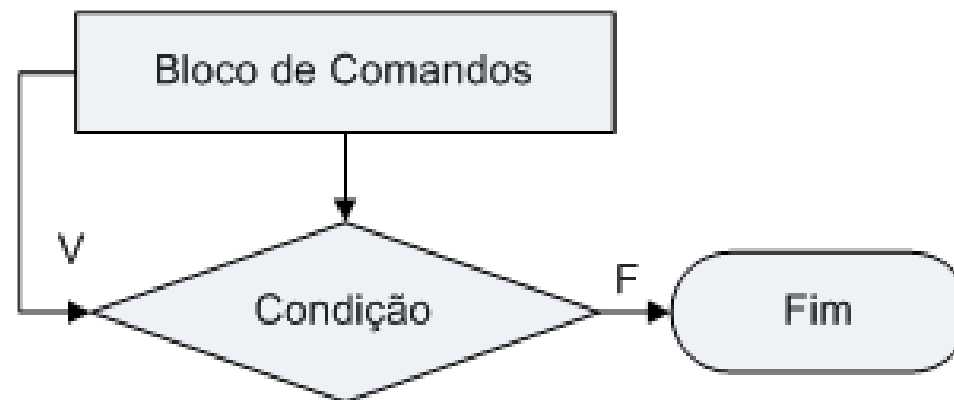
```
while(condição)  
{  
    bloco de comandos  
}
```

```
// Teste condicional no inicio  
while(op != 0){  
    // Instruções  
}
```



Comando do

do
{
 bloco de comandos
} **(condição)**



```
// Teste condicional no fim  
do{  
    // Instruções  
}while (op != 0);
```

Comando break tem a função de interromper a execução de um Loop. Ex:

```
public static void main(String[] args) throws IOException {  
    System.out.println("Digite f para terminar: ");  
    int letra = ' ';  
    while(true) {  
        letra = System.in.read(); //Lê do teclado um caractere  
        if((char)letra=='f')  
            break; //força saída do loop  
    }  
    System.out.println("O loop foi encerrado");  
}
```

Comando continue tem como função fazer com que a condição do comando loop seja novamente testada. Ex.

```
3 public static void main(String[] args) {  
4     for(int i=0;i<=30;i++){  
5         if((i>10)&&(i<20))  
6             continue;  
7         System.out.print(i + " ");  
8     }  
9 }  
  
run:  
0 1 2 3 4 5 6 7 8 9 10 20 21 22 23 24 25 26 27 28 29 30
```

- Representaram uma grande dificuldade nas primeiras versões de Java.
- A Classe Scanner, disponível em **java.util** a partir da versão J2SE 5.0 resolveu esse problema.
- Para usá-la, deve-se incluir o comando **import**
java.util.Scanner

Comandos de Entrada

Exemplo

```
2 import java.util.Scanner;
3 public class Main {
4     public static void main(String[] args) {
5         System.out.println("Digite um número inteiro: ");
6         Scanner vInt = new Scanner(System.in);
7         int num = vInt.nextInt();
8         System.out.println("Digite seu nome completo [string composta]:");
9         Scanner sc = new Scanner(System.in);
10        String nomeCompleto = sc.nextLine();
11        System.out.println("Número Registro.....: " + num);
12        System.out.println("Nome Completo.....: " + nomeCompleto);
13    }
14 }
```

A classe **Scanner** permite a leitura de tipos de dados primitivos e String. A fonte usada para a leitura do teclado foi (System.in) e o tipo primitivo lido, o int.

Para leitura de uma String composta usamos o método `nextLine()` que substitui o delimitador “\n”.

Comandos de Saída

- Usamos o comando:
 - ▣ `System.out.println("texto");`
- Sua função: retornar determinado resultado na tela.
- O atributo estático **out**, definidor na classe **System**, representam um **stream** de saída padrão e é um atributo da classe **PrintStream** [Veremos em breve].

```

public static void main(String[] args) {
    System.out.printf("|%8d|\n", 820);
    System.out.printf("|%2.6f|\n", 1223.4432);
    System.out.printf("|%2.2f|\n", 1223.4432);
    System.out.printf("|%10.2f|\n", 1223.4432);
    System.out.printf("|%010.2f|\n", 1223.4432);
    System.out.printf("|%20f|\n", 1223.4432);
    System.out.printf("|%.2f|\n", 1223.4432);
    System.out.printf("|%10s|\n", "abcdefghijklmnopqrstuvwxywz");
    System.out.printf("|%10.8s|\n", "abcdefghijklmnopqrstuvwxywz");
    System.out.printf("|%10s|\n", "abcde");
    System.out.printf("|%-10s|\n", "abcde"); //Alinh. a esquerda
}

```



```
|      820|
```



```
|1223,443200|
```



```
|1223,44|
```



```
|    1223,44|
```

```
|0001223,44|
```

```
|              1223,443200|
```

```
|1223,44|
```

```
|abcdefghijklmnopqrstuvwxywz|
```

```
|  abcdefgh|
```

```
|    abcde|
```

```
|abcde    |
```




```
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

Exercício 1

Projeto: ProjetoAcampamento

- ❑ Crie um projeto com o nome acima.
- ❑ Como será o projeto:

		Acampamento	
		nome	String
		equipe	char
		idade	int
		imprimir()	void
		separarGrupo()	void

		AcampamentoTeste	
		main(String[])	void

- Parte a) Desenvolver uma classe chamada **Acampamento** com os seguintes atributos: nome, equipe, idade. Em seguida implementar os seguintes métodos:

Método	Descrição
imprimir()	Este método não retorna valor e deve exibir os valores dos atributos na tela.
separarGrupo()	Este método não retorna valor e deverá verificar as seguintes condições: se a idade estiver entre 6 e 10 anos, atribuir A ao atributo equipe; e a idade estiver entre 11 e 20, atribuir B ao atributo equipe; se a idade for superior a 21 anos, atribuir C ao atributo equipe.

- Parte b) Na segunda classe Java chamada **AcampamentoTeste** com a seguinte estrutura, realize no método main():
 - ▣ Criar um objeto chamado membro da classe Acampamentos e atribuir valores aos seus atributos nome e idade.
 - ▣ Executar o método imprimir() e analisar o resultado na tela.
 - ▣ Executar o método separarGrupo().
 - ▣ Executar o método imprimir() novamente e analisar o que será exibido na tela.

Exercício 2

Projeto Computador

- ❑ Crie um projeto com o nome acima.
- ❑ Como será o projeto:

Computador		
f	marca	String
f	cor	String
f	modelo	String
f	numeroSerie	long
f	preco	double
<hr/>		
m	imprimir()	void
m	calcularValor()	void
m	alterarValor(double)	int

ComputadorTeste		
m	main(String[])	void

- Parte a) Desenvolver uma classe chamada **Computador** com os seguintes atributos: marca, cor, modelo, numeroSerie, preco. Implementar os seguintes métodos:

Método	Descrição
imprimir()	Este método não retorna valor e deve exibir os valores dos atributos na tela.
calcularValor()	Não retorna valor e deverá verificar as seguintes condições: caso a marca seja HP, acrescentar 30% ao preço; caso seja IBM, acrescentar 50% ao preço; caso seja qualquer outra marca, manter o preço original.
alterarValor()	Este método recebe um valor como parâmetro. Atribuir este valor ao atributo preço, caso o valor do parâmetro recebido seja maior do que 0. Neste caso, o método alterarValor() deverá além de alterar o valor, retornar 1. Caso contrário não atribuir o valor ao atributo preço e retornar 0.

- ❑ Parte b) Na segunda classe Java chamada **ComputadorTeste** realize:
 - ❑ Criar um objeto da classe **Computador** e atribuir valores a seus atributos. Atribuir HP ao atributo marca.
 - ❑ Executar o método imprimir() e analisar o que será exibido na tela.
 - ❑ Executar o método calcularValor().
 - ❑ Executar o método imprimir() e analisar o que será exibido na tela.
 - ❑ Criar um segundo objeto e atribuir valores a seus atributos. Atribuir IBM ao atributo marca do novo objeto.
 - ❑ Executar o método imprimir() do novo objeto e analisar o que será exibido na tela.
 - ❑ Executar o método calcularValor() do novo objeto.
 - ❑ Executar o método imprimir() do novo objeto e analisar o que será exibido na tela.
 - ❑ Executar para o novo objeto o método alterarValor() com um valor positivo.
 - ❑ Verificar no método main() o retorno do método alterarValor() e mostrar a mensagem de “Valor alterado” caso este retorne 1, e “Valor NÃO alterado” caso retorne 0.
 - ❑ Executar o método imprimir() deste objeto e analisar o que será exibido na tela.

Exercício 3

Projeto: ProjetoConta

- ❑ Crie um projeto com o nome acima.
- ❑ Como será o projeto:

Conta		
f	conta	String
f	agencia	String
f	saldo	double
f	nomeCliente	String
m	sacar(double)	int
m	depositar(double)	void
m	imprimir()	void

ContaTeste		
f	cc	Conta
m	main(String[])	void
m	execCadastrar()	void
m	execConsultar()	void
m	execSacar()	void
m	execDepositar()	void

- Parte a) Desenvolver uma classe Java chamada **Conta** com a seguinte estrutura: conta, agencia, saldo e nomeCliente. Em seguida implementar os seguintes métodos:

Método	Descrição
sacar()	Retorna valor 1 caso o saque seja realizado ou 0 se não houver saldo suficiente na conta. Deverá receber como parâmetro o valor a ser sacado.
depositar()	Realiza o depósito do valor recebido como parâmetro. Não deve retornar valor.
imprimir()	Exibe na tela os atributos da classe. Este método não retorna nada.

□ Parte b) Na segunda classe java chamada **ContaTeste** defina:

- ▣ Criar um atributo público da classe **Conta** para ser usado pelos métodos da classe para realizar saques e depósitos (fora do método main()). Não se esquecer de executar o operador new para o atributo criado.
- ▣ OBS: Ao executar o programa só poderemos fazer um saque se já tivermos realizado um depósito.
- ▣ Implemente os métodos dispostos no próximo slide.

Método	Descrição
main()	Implementá-lo conforme padrão da linguagem Java. O método main() deverá criar um loop para o usuário escolher entre as opções cadastrar, depositar, sacar, consultar. Se for selecionada a opção sacar, executar o método execSacar , Se for selecionado depositar, executar o método execDepositar . Para a opção consultar, executar o método execConsultar . Para a opção cadastrar, executar o método execCadastrar .
execSacar()	Solicitar ao usuário que digite um valor e executar o método sacar() da classe ContasCorrentes usando o atributo criado. Testar o retorno do método sacar() . Se for retornado 1, exibir "Saque realizado", caso contrário, exibir "Saque não realizado".
execDepositar()	Solicitar ao usuário que digite um valor e executar o método depositar() da classe ContasCorrentes usando o objeto criado anteriormente.
execConsultar()	Apresentar os atributos na tela executando o método imprimir() da classe ContasCorrentes .
execCadastrar()	Solicitar que o usuário realize a leitura dos dados via teclado e em seguida realize a atribuição dos valores lidos do teclado aos atributos do objeto classe ContasCorrentes , criado como atributo dessa classe.

Referências

□ Bibliográficas:

- ▣ Mendes – Java com Ênfase em Orientação a Objetos [Exercícios do Capítulo 1]
- ▣ Deitel – Java, como programar – 6º edição.
- ▣ Arnold, Gosling, Holmes – A linguagem de programação Java – 4º edição.
- ▣ Apostilas Caelum
- ▣ Material do Curso de Capacitação Java do CPS

□ Internet

- ▣ <http://java.sun.com>
- ▣ <http://www.guj.com.br>
- ▣ <http://www.portaljava.com>

FIM

Maromo