

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.

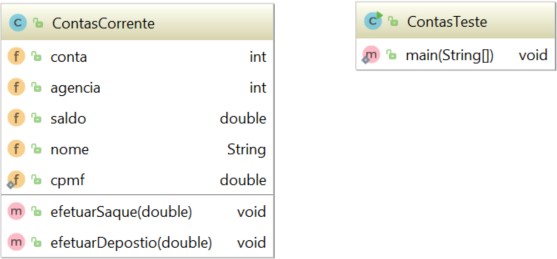
### Atributos em Java

* 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)

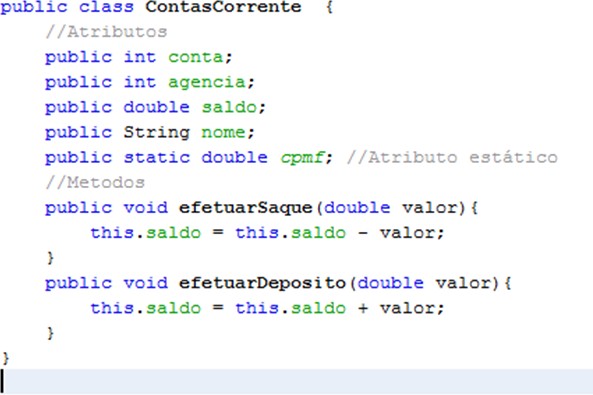
### Programa Exemplo:

Atributos e Métodos

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



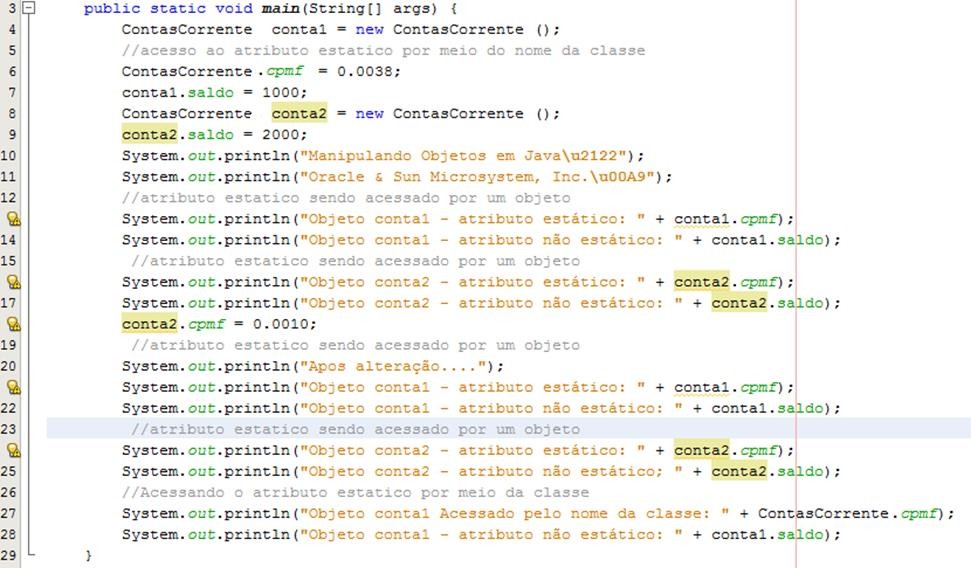
### Classe: ContasCorrente



Na classe ContasCorrente foi definido o atributo cpmf como estático.

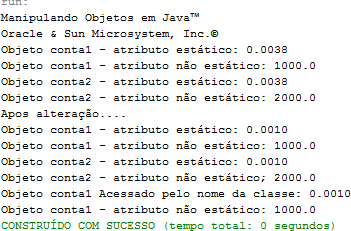
Para atribuir valor ao campo, faz- se por meio do nome da Classe. (ContasCorrente.cpmf=0.0038)

Classe: PrincipalConta



Método: main

### Resultado da Execução



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

Aritméticos e Unários

|  |  |
| --- | --- |
| **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 | <= |

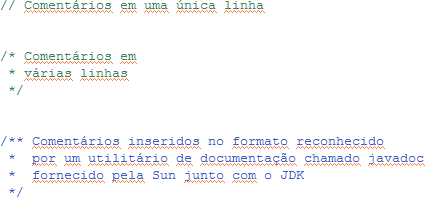
Operadores Lógicos

|  |  |
| --- | --- |
| **Função** | **Sinal** |
| E | && |
| OU | | | |
| Não | ! |

# Conversão de tipos

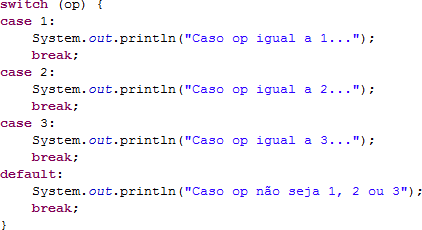
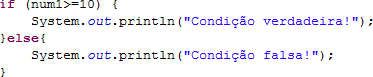
|  |  |  |
| --- | --- | --- |
| **Supondo a variável x** | **Converter em** | **y recebe o valor convertido** |
| * Entre tipos numéricos | | |
| int x = 10 | float | float y = (float) x |
| int x = 10 | double | double y = (double) x |
| float x = 10.5 | int | int y = (int) x |
| * De string para numéricos | | |
| 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



Estruturas de Controle de Fluxo

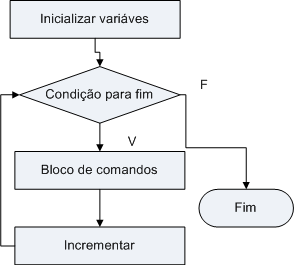
◼ Desvios condicionais



### Estruturas de Controle de Repetição

* Comandos:
  + for
  + while
  + do
  + break
  + continue

### Comando for

**for(inicialização;condição de fim;incremento)**

**{**

**bloco de comandos**

**}**

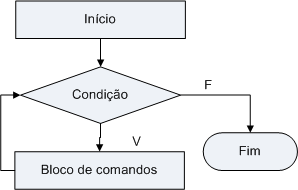
**for(int i=0; i<10; i++)**

**{**

**System.out.println(i);**

**}**

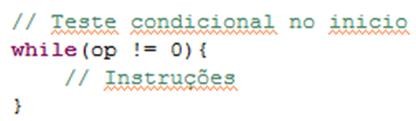
### Comando While

**while(condição)**

**{**

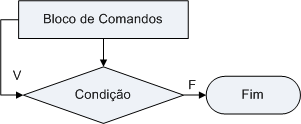
**bloco de comandos**

**}**



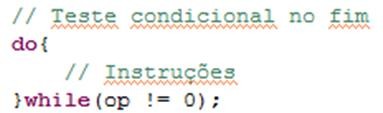
### Comando do

**do**

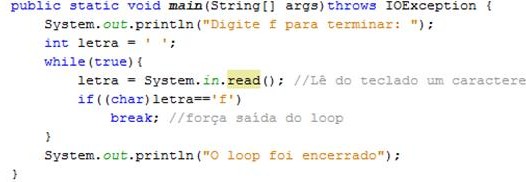
**{**

**bloco de comandos**

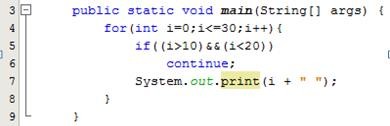
**} (condição)**



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



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

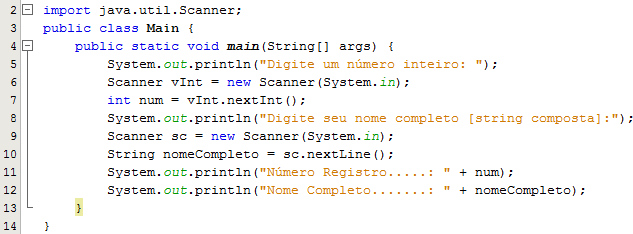


Comandos de Entrada

* 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



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”.

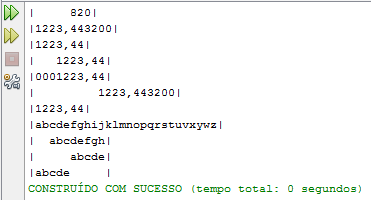
Comandos de Saída

* 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].



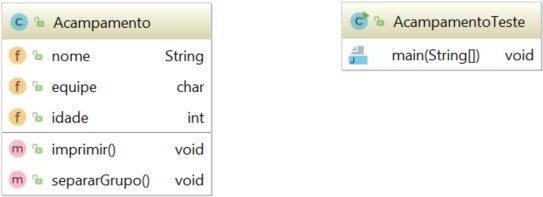


## Crie um projeto com o nome acima.

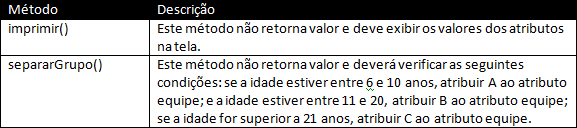
* Como será o projeto:

### Exercício 1

#### Projeto: ProjetoAcampamento



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



* + 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.

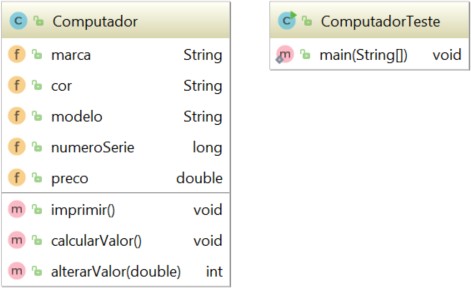
Executar o método separarGrupo().

Executar o método imprimir() novamente e analisar o que será exibido na tela.

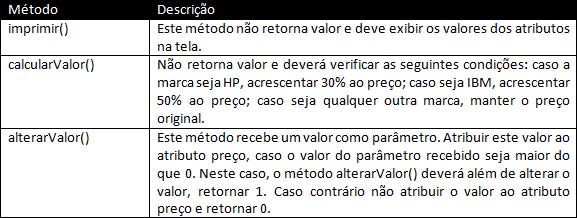
### Exercício 2

#### ProjetoComputador

* Crie um projeto com o nome acima.
* Como será o projeto:



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



* 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.

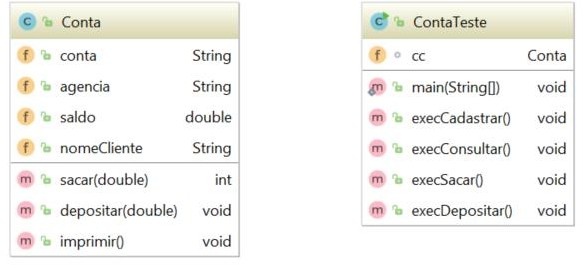
 Verificar no método main() o retorno do método alterarValor() e mostrar a mensagem de “Valor alterado” caso este retorne 1, e “Valor NÃO alterado” caso retorne 0.

Executar o método imprimir() deste objeto e analisar o que será exibido na tela.

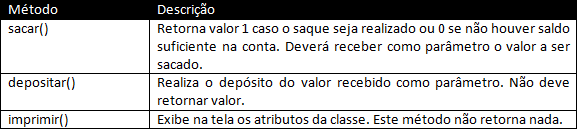
### Exercício 3

**Projeto:** ProjetoConta

* Crie um projeto com o nome acima.
* Como será o projeto:



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



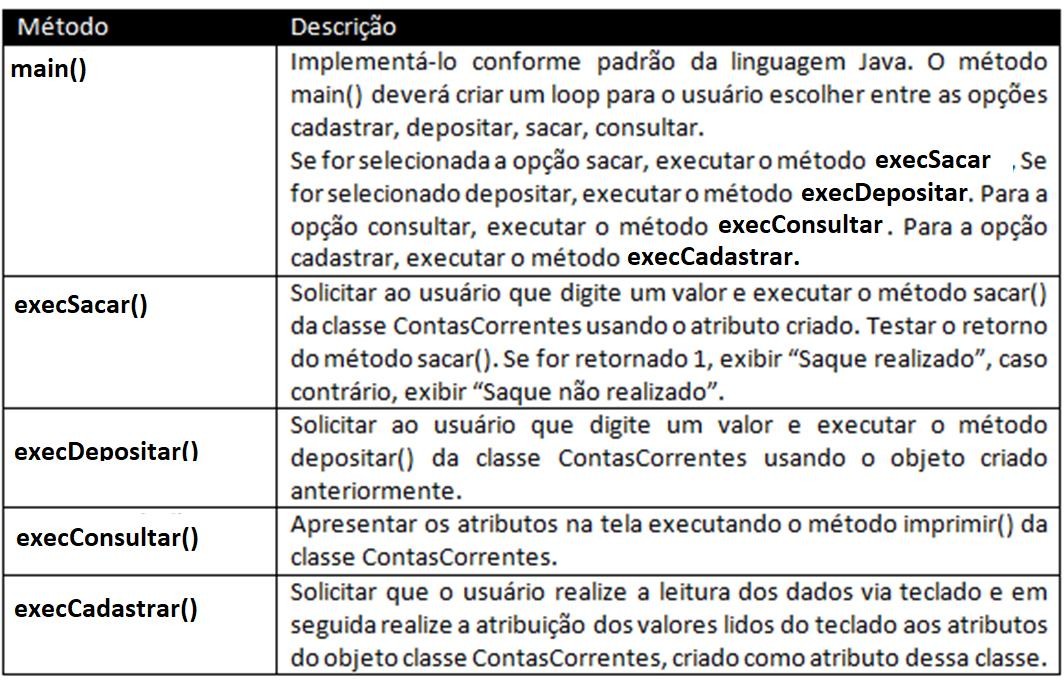
## Parte b) Na segunda classe java chamada **ContaTeste**

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.



### 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://java.sun.com/)

[http://www.guj.com.br](http://www.guj.com.br/)

[http://www.portaljava.com](http://www.portaljava.com/)

Maromo

### FIM