

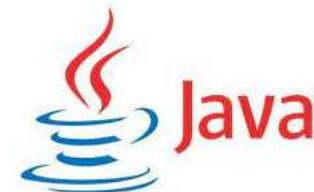


Residência em Tecnologia da Informação e Comunicação

Funções e Métodos em Java

Professor:

Alvaro Degas Coelho



INSTITUIÇÃO EXECUTORA



COORDENADORA



APOIO





Residência
em Software

Contexto

- Precisamos de alguns conceitos serem revisados
- Já vimos em C++



Classe

- Abstração: “Uma classe abstrai um conjunto de objetos com características similares.”
- Objeto: “Uma classe define as características e métodos de um conjunto de objetos.”
- Código-fonte do Objeto: “Uma classe é um arquivo texto que define as características e métodos de um conjunto de objetos.”

```
public class Retangulo {  
    public float Base;  
    public float Altura;  
    public float Area(){  
        return this.Base * this.Altura;  
    }  
}  
  
public class Quadrado {  
    public float Lado;  
    public float Area(){  
        return this.Lado * this.Lado;  
    }  
}  
  
public class Triangulo {  
    public float Base;  
    public float Altura;  
    public float Area(){  
        return this.Base * this.Altura / 2;  
    }  
}
```



Residência
em Software

- Objeto é uma entidade que pode ser física, conceitual ou de software. Ou seja, uma representação genérica.
- Instância é usada com o sentido de exemplo. É a concretização da classe. Ou seja, são os objetos de fato criados e ocupando espaço na memória.

Objetos e Instâncias

Classe Cachorro:



1 Objeto
Cachorro



3
objetos
Cachorr
o
Instâncias
da classe
Cachorro



Totó
“Epaminônda
s”

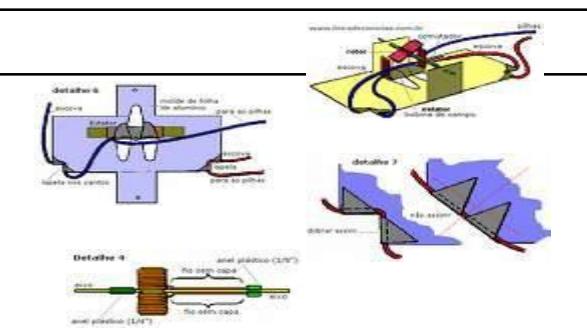


Residência
em Software

Encapsulamento

- É o mecanismo que permite separar detalhes de funcionamento “Características (variáveis) e funções (métodos)” de sua interface.
- Exemplo: Para utilizarmos um liquidificador, não precisamos saber detalhes de seu funcionamento. A única interface que conhecemos são seus botões.

Classe →
Liquidificador“Li
quidificador.txt”



Instância de um
objeto de
liquidificador: Muito
mais fácil de usar.



Encapsulamento - Atributos

- São características (variáveis) da classe e podem ser divididos em dois tipos básicos:
 - Atributos de Instância: “determinam o estado de cada objeto.”
 - Atributos de Classe: “possue um estado que é compartilhados por todos os objetos de uma classe.”

- Em contraste com a estática dos dados, os *métodos* definem as ações a serem tomadas em diversos momentos da execução de um programa. Como em outras linguagens, como C, C++, Pascal, Fortran, etc, os métodos correspondem aos conceitos comuns de funções, procedimentos ou subrotinas.
- Estes são apenas conjuntos ordenados de declarações de dados, comandos e expressões. Em termos simples, são os métodos que realizam todas as tarefas para as quais o programa foi escrito, por exemplo, realizar cálculos, resumir informações de um arquivo, produzir um relatório, criar um gráfico, gerar um filme de animação, etc.

Métodos

Residência
em Software

Classe -> Fábrica Bicicletas

Atributos

- Cor
 - Tipo material
 - Tamanho roda
- Métodos*
- Andar para frente
 - Travar

Objecto -> Bicicleta nº 1

Atributos

- Cor: verde
 - Tipo material: alumínio
 - Tamanho roda: 26
- Métodos*
- Andar para frente
 - Travar

Objecto -> Bicicleta nº 2

Atributos

- Cor: amarela
 - Tipo material: ferro
 - Tamanho roda: 20
- Métodos*
- Andar para frente
 - Travar



Residência
em Software

Declaração de Métodos

- A declaração mais simples que podemos fazer de um método (lembrando que isso deve ser feito dentro de uma classe) é a seguinte:

```
void [nome do método] () {  
    [corpo do método]  
}
```

- Onde o **[nome do método]** é um identificador que define o nome pelo qual o método é conhecido, e **[corpo do método]** consiste de uma lista ordenada de declaração de variáveis, de expressões e de comandos.



Residência
em Software

Declaração de Métodos

- A primeira palavra-chave, **void**, define o valor retornado pelo método, neste caso, nenhum. Podemos usar qualquer tipo de dado válido como valor de retorno de um método. Nesse caso, ao terminar, o método seria obrigado a devolver um dado do tipo especificado. Por exemplo,

```
class Numero {  
    double x = 1;  
    void print() {  
        System.out.println("O valor e " + x);  
    }  
}
```

Declaração de Métodos

- Define uma classe chamada **Numero**, a qual contém uma variável **x**, inicializada com 1, e um método sem valor de retorno, **print**, que apenas escreve um texto e o valor de **x**, através da chamada do método **System.out.println**.
- O par de parênteses adiante do nome do método introduz uma lista (vazia, neste caso) de argumentos. A chamada de um método pode ser acrescida de parâmetros, os quais são associados aos seus respectivos argumentos

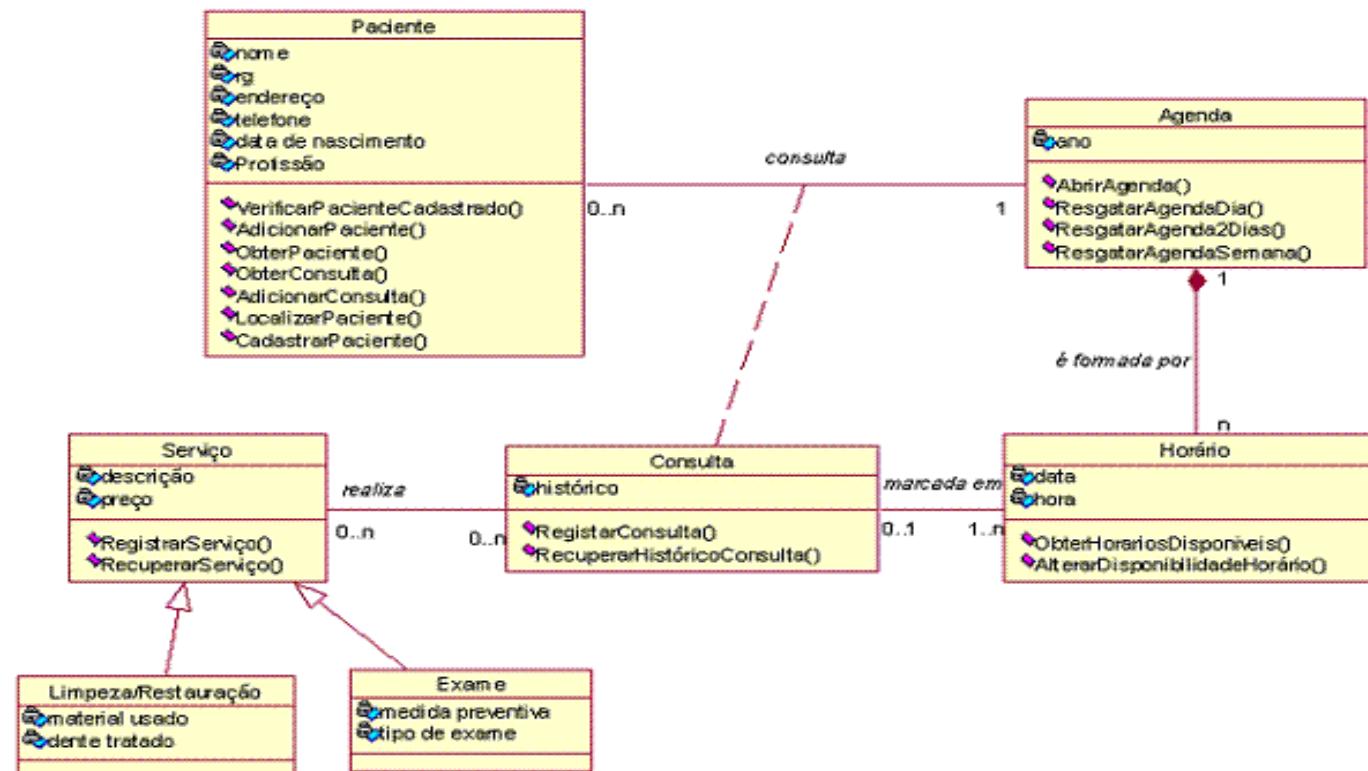


Notação UML

- A Unified Modeling Language (UML) é uma linguagem de modelagem não proprietária de terceira geração.
- A UML não é uma metodologia de desenvolvimento, o que significa que ela não diz para você o que fazer primeiro e em seguida ou como projetar seu sistema, mas ela lhe auxilia a visualizar seu desenho e a comunicação entre objetos.
- Aonde Baixar: <http://jude.change-vision.com/jude-web/index.html>



Residência
em Software



Funções: importante saber

- Em Java TUDO é orientado a objeto
 - Portanto, toda função é um método
- Declaração
 - tipoDeRetorno nomeDoMetodo(parâmetros)
- Já vimos exemplos de funções nas aulas anteriores

Funções void

- Funções que não precisam retornar nenhum dado
 - Retoram um valor “vago” – void
- Tipicamente: funções que executam operações de entrada ou saída
- Exemplo
 - Crie uma classe Estudante com os atributos Nome, DataNascimento e CPF
 - Note que DataNascimento vai ser do tipo Date
 - Date x;
 - x=new Date();
 - No momento da criação do objeto Date, o sistema armazena no mesmo a Data/Hora do relógio do computador.
 - Date x=new Date();
 - System.out.println(x);
 - java.text.SimpleDateFormat()
 - Crie um método para mostrar a Data de Nascimento na tela
 - Tipo void

Mostrando uma data na tela

- Como colocar uma data digitada pelo usuário em uma variável Date?
 - Conversores.
 - Suponha que o usuário tenha digitado uma data numa String chamada strData;
 - Readln() lembra?
 - `java.text.SimpleDateFormat();`
 - `SimpleDateFormat sdf1= new SimpleDateFormat ("dd/MM/yyyy"); //há outras máscaras Date`
 - `dataUsuario=sdf1.parse(stx);`
- Crie um método para solicitar uma data de nascimento (String), criar uma variável do tipo Date() e devolver

Crie um método para ler e mostrar os dados de um cliente

- Use Getters e Setters
- Use seu método para mostrar a data de nascimento
- Use seu método para ler a data de nascimento
- Crie um método calculadade() que retorna um int com a idade de um cliente
 - Google
- Use seu método para mostrar a idade do cliente

Exercício

- Crie uma classe ListaNumeros
 - Dentro dela crie um arraylist com números (float)
 - Crie métodos novoNumero(float f) e listaNUmeros() (lista todos os números)
- Crie uma função media() que retorne a média dos números digitados
 - Use esta função e Escreva na tela
 - System.out.println(numeros.media());
- Crie uma função ordena() que ordene os números em ordem crescente
 - Pesquise o método sort() de ArrayList
- Crie uma função mediana() que retorne a mediana dos números digitados
 - Use esta função e Escreva na tela
- Crie uma função colocadoEm(int N) que receba um número e mostre o N-ésimoMAIOR número da lista
 - Note que la está em ordem crescente!

Exercício

- Crie uma classe CalculosEstatisticos
- Crie uma função factorial(int x) que liste o Fatorial de x
 - Teste este método usando sua função main
- Crie uma função arranjo(N, P) que mostre a quantidde de arranjos de N números tomados P a P
 - Teste este método usando sua função main
- Crie uma função combinacao(N, P) que mostre a quantidde de combinações de N números tomados P a P
 - Teste este método usando sua função main

$$A_{n,p} = \frac{n!}{(n-p)!}$$

$$C_{n,p} = \frac{n!}{p!(n-p)!}$$

Exercício

- Crie uma classe chamada GeradorPrimos
- Crie um método que receba um número inteiro N e liste todos os números divisores de N
 - Teste este método usando sua função main
- Crie um método PRIVADO que receba um número inteiro N e retorne a quantidade total de divisores dele
 - Sim, é semelhante ao anterior!
- Crie um método PÚBLICO que receba um número e use o método anterior para retornar true se o número é primo, ou false caso contrário
 - Teste este método usando sua função main (if e else)
- Crie um método publico que receba um número N e escreva na tela todos os números primos menores ou iguais a N
- Crie um método que receba um número N e escreva na tela os N primeiros números primos