UFRPE

Aula 4 a 8

Richarlyson A. D'Emery
site: https://sites.google.com/site/profricodemery/mpoo
grupo: http://groups.google.com/group/**mpoo_uast**
email grupo: **mpoo_uast**@googlegroups.com
contato: rico_demery@yahoo.com.br

UFRPE

Parte 1 - Teoria

Sumário

UFRPE

- A Linguagem Java
- Conceitos
- Objetos
- Classes
- Mensagens e Métodos
- Encapsulamento
- This
- Polimorfismo
- Herança
- Termos Relacionados
- Propriedades

Linguagem Java

UFRPE

- É uma linguagem de programação de propósito geral, orientada a objetos e adequada para a Internet.
  - *Utilizada para o desenvolvimento de qualquer tipo de software, com sintaxe similar ao C.*
  - *Modelo OO baseado em Smalltalk e Simula67*
  - *Applets JAVA*
- A linguagem JAVA foi projetada e implementada por um pequeno grupo pessoas, coordenado por James Gosling, na Sun Microsystems em Mountain View, Califórnia, em 1991.

Conceitos

UFRPE

- Definição:
  - O termo orientação a objetos significa organizar o mundo real como uma coleção de objetos que incorporam *estrutura de dados* e *um conjunto de operações* que manipulam estes dados.
- Todas as linguagens orientadas a objetos possuem três características básicas:
  - *Objetos*
  - *Polimorfismo*
  - *Herança*

Objetos

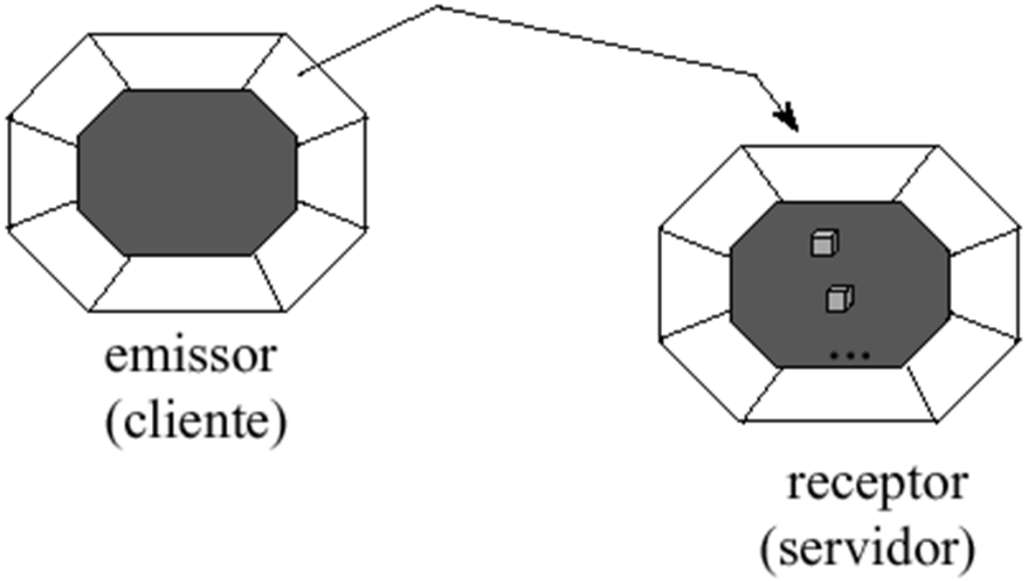
UFRPE

- É uma entidade lógica que contém dados e código para manipular esses dados.
- Coisas do mundo real
- Pode ser concreto ou um conceito(carro, cadeira, gerente)
- Pode ser simples ou complexo
- Possui atributos (caractéristicas) e operações (comportamento).

Objetos

UFRPE

- Objetos interagem e comunicam-se através de *mensagens...*
- *...as mensagens identificam os métodos a serem executados no objeto receptor*



Classes

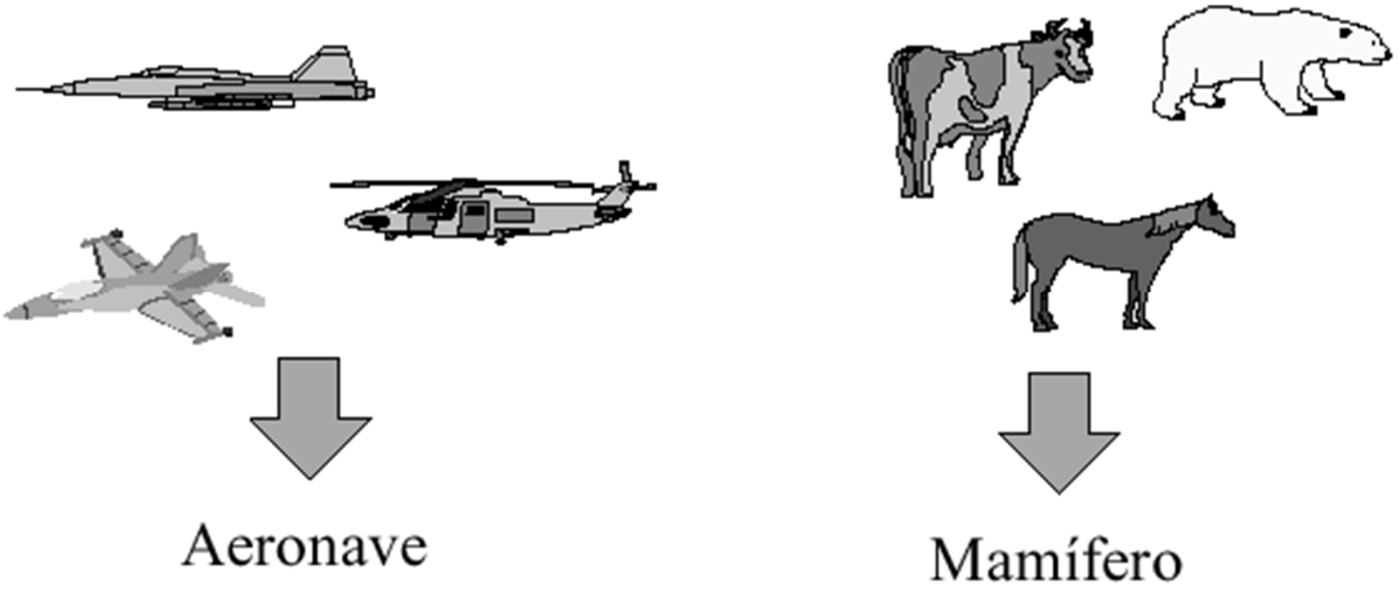
UFRPE

- Classes:
  - É um tipo de dado, como os já conhecidos, para declarar variáveis.
  - Uma variável de uma classe é chamada de Objeto.
  - Definir uma classe não cria um objeto, assim como um tipo de variável NÃO é uma variável.
- Variáveis de Classe:
  - São os dados declarados em uma classe
- Instância:
  - Temos uma instância de uma classe quando declaramos um objeto a partir de uma classe. É semelhante à declaração de uma variável.

Abstração

UFRPE

- Focalizar o essencial, ignorar propriedades acidentais



- A abstração deve ser sempre feita com algum objetivo, porque este determina o que é e o que não é importante.

Classes

UFRPE

- Objetos com similaridades são agrupados em classes
- Compostas por atributos e operações
- Classes não são objetos
- Classes servem como modelos para objetos
- Objetos são instâncias de classe

Representação Gráfica de Classes

UFRPE

Nome da Classe
atributos
métodos

Funcionario
+ nome: String - salario: float
- alteraSalario(): void

Classes

UFRPE

- Exemplo de declaração de classe em Java:

```
public class Funcionario {
  String nome;     //variável(atributo)
  private float salario;  //variável(atributo)
  void alteraSalario() {} //método(operação)
}
```

Mensagens e Métodos

UFRPE

- Para invocar um método de um objeto, deve-se enviar uma mensagem para este objeto.
- Para enviar uma mensagem deve-se:
  - identificar o objeto que receberá a mensagem
  - identificar o método que o objeto deve executar
  - passar os argumentos requeridos pelo método
- Um objeto possui:
  - um estado (definido pelo conjunto de valores dos seus atributos em determinado instante)
  - um comportamento (definido pelo conjunto de métodos definido na sua interface)
  - uma identidade única

Encapsulamento

UFRPE

- Não podemos acessar os dados de um objeto diretamente. (Fenômeno da caixa preta)
- Para acessar suas variáveis de instância, devemos fazer através de métodos.
- Não precisamos saber como as variáveis de instância são armazenadas para poder utilizá-las.
- Considere o seguinte trecho de código C:

```
double d = 0;
d += 2.5;
```

Encapsulamento

UFRPE

- Como um *double* é representado internamente?
- O que acontece quando você usa o operador +=?
- Você alguma vez se importou com isto?
- Você deveria se preocupar?

Encapsulamento

UFRPE

- Todo o acesso aos dados do objeto é feito através da chamada a uma operação (método) da sua interface
- Mudanças na implementação de um objeto, que preservem a sua interface externa, não afetam o resto do sistema.
- A interface (pública) de um objeto declara todas as operações permitidas (métodos)
- Implementado em Java através dos modificadores de acesso
  - Public
  - Private
  - Protected

Encapsulamento

UFRPE

Modificador	Mesma Classe	Mesmo Pacote	Subclasse	Todos
public	SIM	SIM	SIM	SIM
protected	SIM	SIM	SIM	
default	SIM	SIM		
private	SIM			

A Referência this

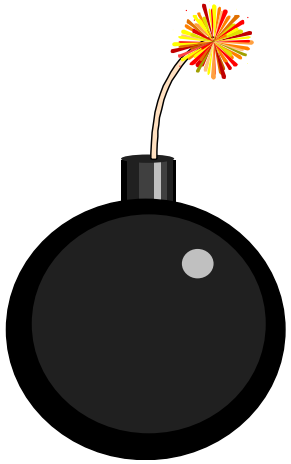
UFRPE

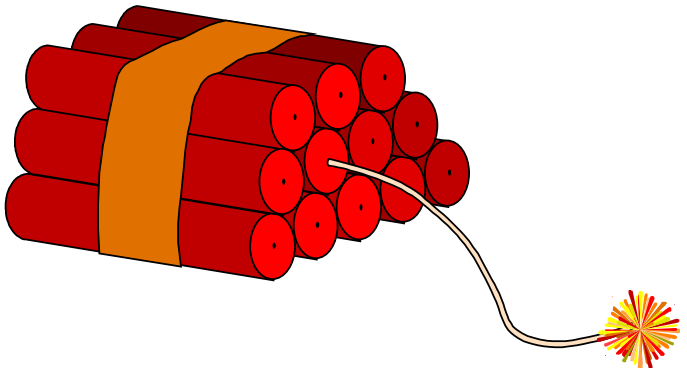
- Métodos de instância recebem um argumento chamado this que é uma referência ao objeto corrente.
- O uso explícito do this é necessário quando:
  - Quando o nome de um parâmetro do método for o mesmo de uma variável de instância
  - Quando for necessário passar uma referência do objeto corrente como parâmetro para outro método

Polimorfismo

UFRPE

- O Polimorfismo ocorre quando uma mesma mensagem chegando a objetos diferentes provoca respostas diferentes.





Polimorfismo

UFRPE

- O Polimorfismo ocorre quando uma mesma mensagem chegando a objetos diferentes provoca respostas diferentes.
- Ex.1:

```
int  soma (int a, int b){
  int c=a+b;
  return c;
}
double  soma (double a, double c){
  double d=a+b;
  return d;
}
```

- Ex.2:

```
System.out.println();
```

Herança

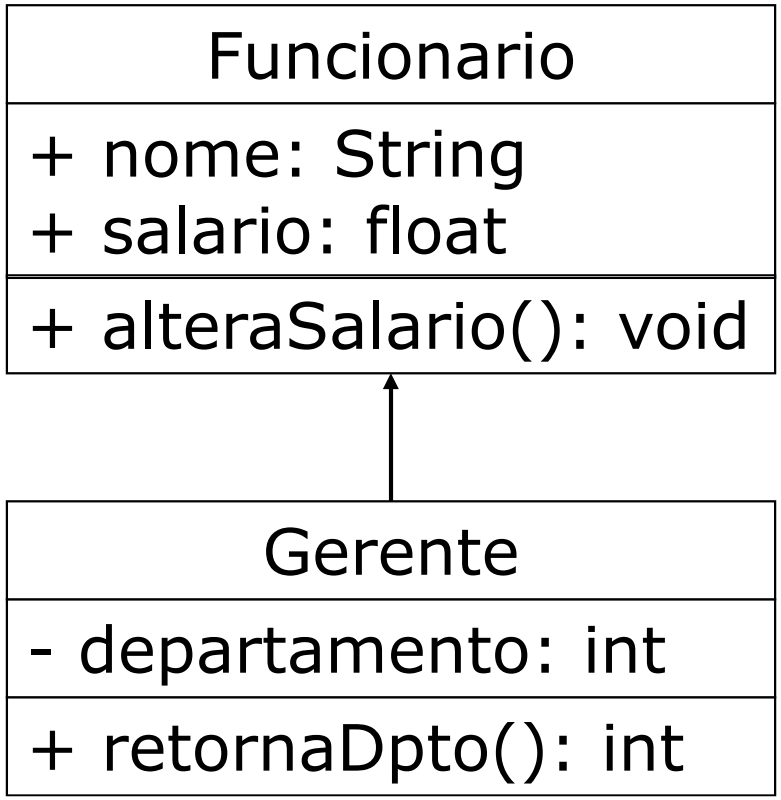
UFRPE

- É um mecanismo para definir novas classes a partir de classes existentes
- Prove reusbilidade
- Permite agrupar classes relacionadas de modo que elas possam ser gerenciadas de forma coletiva
- Java implementa herança com o uso de extends
  - Exemplo:

```
public class Gerente extends Funcionario{ }
```

Herança

UFRPE



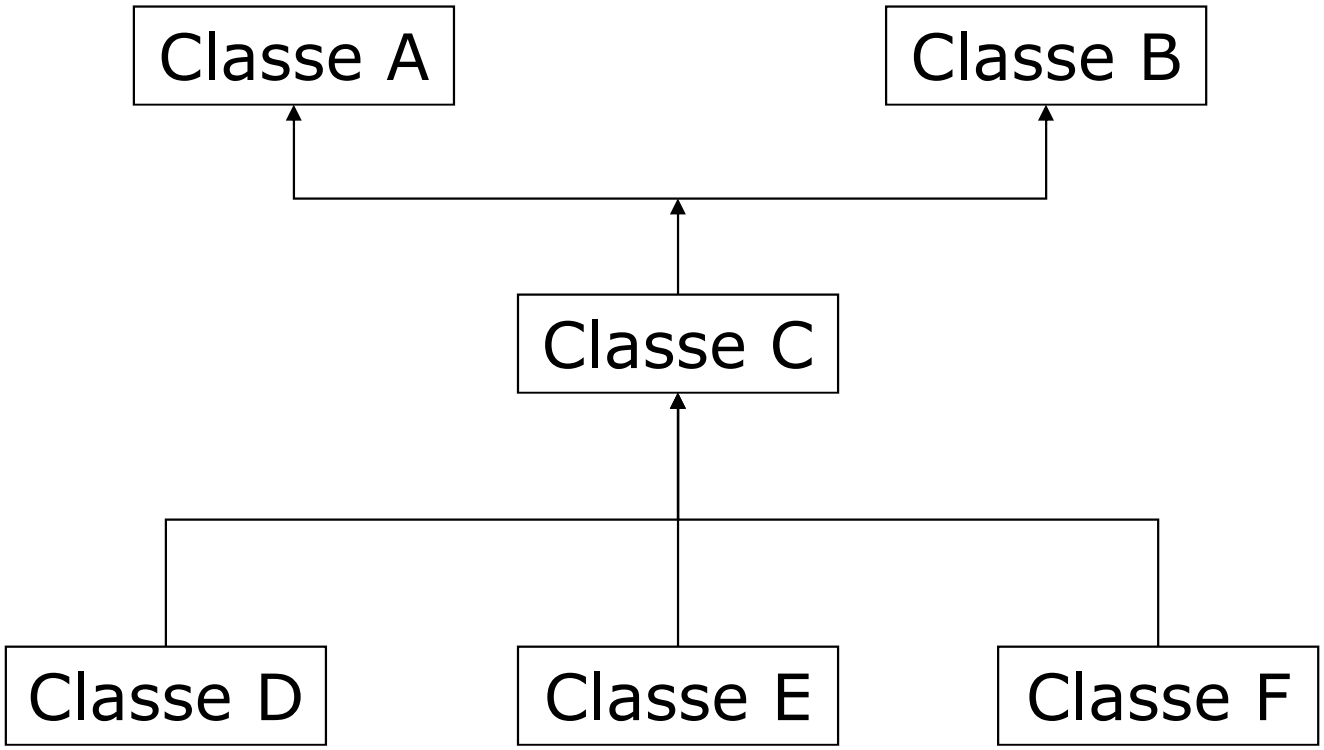
Herança Simples

UFRPE

- Uma classe herda diretamente de uma única classe.
- Java só permite herança simples.
- Melhora a legibilidade.
- Restrição contornada com o uso de interfaces.

Herança Múltipla

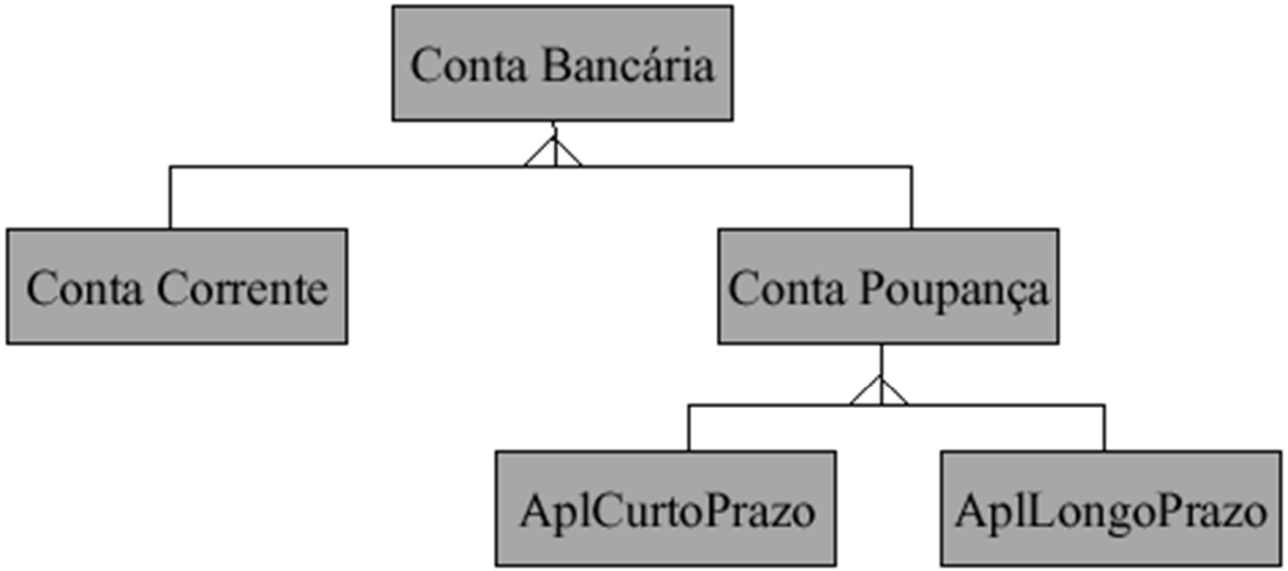
UFRPE



Herança

UFRPE

Exemplo:



Hierarquia de Classes de Contas Bancárias

Termos Técnicos Relacionados

UFRPE

- ***Ligação Dinâmica:***
  - É o processo de ligar (link) um programa com suas bibliotecas durante sua execução.
  - Nas linguagens convencionais, a ligação é estática, isto é, todas as conexões ocorrem antes da execução do programa.
  - Na ligação dinâmica, as bibliotecas são carregadas e descarregadas da memória conforme solicitadas durante a execução do programa.
  - No ambiente Windows, essas bibliotecas possuem a extensão “DLL” (Dynamic Link Library).

Termos Técnicos Relacionados

UFRPE

- **Programação Visual**
  - É a maneira de se programar desenhando os objetos no lugar de descrever seu código.
  - A Programação Visual está sempre associada à orientação a objetos.
- **Métodos Construtores e Destrutores**
  - Métodos Construtores: alocam dinamicamente na memória uma instância de uma classe.
  - Métodos Destrutores: retiram a instância da memória, liberando-a.
  - A alocação das instâncias é dinâmica.

Termos Técnicos Relacionados

UFRPE

- **Persistência**:
  - É o tempo que um objeto permanece na memória. Algumas instâncias são criadas no início da execução do programa e só são destruídas no final do programa.
  - Outras instâncias são necessárias apenas por algum tempo. Quando um objeto não é mais necessário, ele é destruído e seu espaço na memória liberado. A recuperação do espaço é denominada ***Coleta de Lixo*** (Garbage Collection).

POO x Técnicas Tradicionais

UFRPE

• **Métodos**	• **Procedimentos e funções**
• **Variáveis de Instância**	• **Dados**
• **Mensagens**	• **Chamadas de procedimento e funções**
• **Classes**	• **Tipos de Dados**
• **Herança**	• **Não existe**
• **Chamadas sob controle do sistema**	• **Chamada sob controle do programador**

Propriedades

UFRPE

- As propriedades NÃO SÃO *variáveis de instância*.
- As propriedades chamam *métodos* que alteram as variáveis de instância.
- Quando atribuímos um dado a uma propriedade, é chamado um método SET.
- Quando pegamos um dado, é chamado um método GET.
- Exemplos:

objeto.propriedade = valor

- ***Isto equivale a:***

```
objeto.SET_prop(valor)
```

dado = objeto.propriedade

- ***Isto equivale a:***

```
objeto.GET_prop(dado)
```

UFRPE

Parte 2 – Implementação

Sumário

UFRPE

- JDK - Java Development Kit
- Compilando e Executando
- Comentários
- Palavras-chave
- Identificadores
- Declarações
- O método main ()
- Convenções
- Exercícios

JDK - Java Development Kit

UFRPE

- Pacotes:
  - Entrada e Saída
  - Interface Gráfica
  - Comunicação de Rede
  - Thrads
- Ferramentas:
  - Compilador
  - AppletViewer
  - Interpretador
  - Gerador de Documentos
  - Gerador de Arquivos jar

Compilando

UFRPE

- Erros comuns de compilação:
  - `javac: Command not found`
  - `HelloWorld.java: 10: Method printl(java.lang.String) not found in class java.io.PrintStream System.out.printl ^ (“Hello World!)”;`
  - `HelloWorld: 4: Public class HelloWorld must be defined in a file called “HelloWorld.java”`

C:\>javac HelloWorld.java //Compilador

Executando

UFRPE

- Erros comuns de execução:
  - `Can't find class HelloWorld`
  - `Exception in thread "main"`
    `java.lang.NoSuchMethodError: main`

C:\>java HelloWorld //Interpretador ou JIT

Comentários

UFRPE

- Em Java é possivel comentar código de três formas diferentes

```
// Comentário de uma linha

/* Comentário
   de várias
   linhas */

/** Comentário para documentação */
```

Palavras-chave em Java

UFRPE

abstract	**do**	**implements**	**private**	**this**
boolean	**double**	**import**	**protected**	**throw**
break	**else**	**instanceof**	**public**	**throws**
byte	**extends**	**int**	**return**	**transient**
case	**false**	**interface**	**short**	**true**
catch	**final**	**long**	**static**	**try**
char	**finally**	**native**	**strictfp**	**void**
class	**float**	**new**	**super**	**volatile**
continue	**for**	**null**	**switch**	**while**
default	**if**	**package**	**synchronized**	

Identificadores

UFRPE

- São nomes dados às variáveis, classes ou métodos
- Podem iniciar com uma letra Unicode, um cifrão($) ou um underscore(_)
- São sensíveis ao caso e não tem tamanho máximo
- Exemplo:
  - meuNome
  - nome_usuario
  - _valor
  - $contador
  - HelloWorld
  - @mail //ilegal

Declaração de Classes

UFRPE

- Sintaxe de uma classe Java

```
<modificador> class <nome> {
        <declaração de atributos>
        <declaração de construtores>
        <declaração de métodos>
}
```

- Exemplo

```
public class Retangulo {
        int base, altura;
        public int calculaArea(){
                return base*altura;
        }
}
```

Declaração de Atributos

UFRPE

- Sintaxe de um atributo Java

```
<modificador> <tipo> <nome> = <valor inicial>;
```

- Onde:
  - **<tipo>** = `byte | short | int |long | char | float | double | boolean |<nome de uma classe>`
- Exemplo:

```
public int x;
static double calc = 45.76;
String nome = "Linguagem de Programação II";
```

Declaração de Métodos

UFRPE

- Sintaxe de um método em Java

```
<modificadores> <tipo de retorno> <nome> (<parametros>)  {
        <expressão>
}
```

- Exemplo:

```
public int calculaArea (int b, int a) {
        int base = b, altura = a;
        return base*altura;
}
```

Declaração de construtores

UFRPE

- Sintaxe de um construtir em Java

```
<modificadores> <nome da classe> (<parametros>)  {
        <expressão>
}
```

- Exemplo:

```
public Retangulo (int b, int a) {
        this.base = b;
        this.altura = a;
}
```

O método main()

UFRPE

- Uma aplicação java precisa ter o método main()
- Recebe um parâmetro: um array de String
- Tem sempre a mesma declaração:

```
public static void main (String [] args)
```

- O parâmetro args é passado pela linha de comando:

```
C:\> java Aplicacao nome1 valor1 "Uma String"
```

Sobrecarga de Métodos

UFRPE

- É possivel ter métodos com o mesmo nome numa mesma classe
- A lista de parametros deve ser diferente
- O tipo retornado pode ser diferente
- Exemplo:

```
public void println()
public void println(double d)
public void println(String s)
```

Convenções de Nomeação

UFRPE

- Classes
  - ContaCorrente
- Interfaces
  - Conta
- Métodos
  - calculaValor
- Variáveis
  - nomeGerente
- Constantes
  - MAXIMO_SALARIO
- Pacotes
  - package projeto.classes

Exercício 1

UFRPE

- Se primeiro Programa:
  - Digite e compile o programa:

```
public class ClasseJava {
   public static void main(String [] args) {
       System.out.println ("Primeiro Programa");
   }
}
```

Exercício 2

UFRPE

- Tipos de Dados e Operadores:
  - Digite e compile o programa abaixo:

```
public class TiposJava {
   public static void main(String [] args) {
     short s1, s2, s3;
     s1 = 17;
     s2 = 5;
     s3 = s1 + s2;
     f = 3.14;
   }
}
```

1) Comente as mensagens exibidas no console do compilador

2) Faça as devidas correções.

Exercício 3

UFRPE

- Digite e compile este programa:

```
public class TesteIncremento {
   public static void main(String [] args) {
     int x = 5;
     int y = x++;
     int z = ++x;
     int w = --x + y--;
     System.out.println("O valor de x: " +x);
     System.out.println("O valor de y: " +y);
     System.out.println("O valor de z: " +z);
     System.out.println("O valor de w: " +w);
   }
}
```

- Explique qual a diferença entre x++ e ++x

Exercício 4

UFRPE

- Geralmente as frutas contém casca e caroços. Escreva um programa que contenha uma classe chamada Frutas. Crie o método construtor e dois objetos para a classe Fruta, sendo os objetos: 1 fruta com casca e caroço e 1 fruta com casaca e sem caroço. Crie um método chamado retirarCaroco que retira o caroço da fruta, caso haja caroço.
- *Dica:* Você pode criar uma representação lógica, por exemplo, 0 caso a fruta não possua casca ou caroço e 1 caso contrário.

Exercício 5

UFRPE

- Crie uma classe chamada Conta que representa uma conta bancária. Determine os atributos e métodos desta classe
- Escreva uma aplicação que instancie quatro objetos da classe Conta

Exercício 6

UFRPE

- Crie uma classe chamada ContaPoupanca e outra chamada ContaCorrente, ambas subclasse de Conta. Determine os atributos e métodos específicos desta classe.
- Escreva uma aplicação que instancie um objeto de cada classe

Exercício 7

UFRPE

- Escreva um programa que contenha uma classe chamada Robot. Um robô geralmente tem nome, posição e direção. Crie o método construtor para a classe Robot. Utilize sobrecarga de método (polimorfismo) para outro construtor para inicializar um robô com um nome indicado, e supondo que esteja na posição (0,0) direcionado para o Norte.
- Crie dois métodos, sendo um para o robô andar 1 passo e outro para vários passos.
- Crie um método para mudar a posição do robô.
- Crie um método que retorna a configuração do robô.

Exercício 8

UFRPE

- Com o programa do exercício 6, crie um robô chamado Cria, e mova-o para:
  - 10 passos para o Norte
  - 10 passos para o Leste
- Acrescente um método chamado retornaPosZero() que leva o robô a sua posição inicial.

UFRPE

FIM

Prof. Richarlyson D'Emery

site: https://sites.google.com/site/profricodemery/mpoo

grupo: http://groups.google.com/group/**mpoo_uast**

email grupo: **mpoo_uast**@googlegroups.com

contato: rico_demery@yahoo.com.br