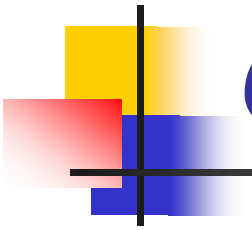
UNIVERSIDADE CATÓLICA DE PERNAMBUCO

Conceitos Básicos da Programação OO

Universidade Católica de Pernambuco
Ciência da Computação

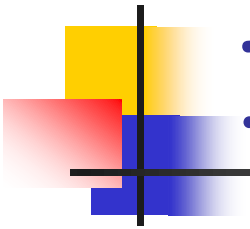
Prof. Márcio Bueno
poonoite@marciobueno.com

Fonte: Material da Profª Karina Oliveira



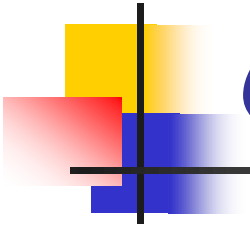
Objetivos

- Aprender os principais conceitos relacionados a Programação Orientada a Objetos (POO).
- Aplicar os conceitos aprendidos na linguagem Java, mostrando vários exemplos de código.



Introdução

- Programação Imperativa x Programação OO
  - Programação Imperativa ⇒ modularização dos programas baseada nas ***funções*** que um sistema vai oferecer ao usuário.
  - Programação Orientada a Objetos ⇒ modularização dos programas baseada na definição dos *objetos* (***dados***) a serem manipulados pelo sistema e, em seguida, das ***atividades*** que os objetos poderão realizar.



Conceitos Básicos

- **Classe**
  - Tipo de dado que agrupa um conjunto de variáveis (atributos) e funções (métodos) que podem realizar ações sobre as variáveis.
- **Atributos**
  - São as propriedades ou características de uma classe.
  - É a informação contida nos atributos que diferencia os objetos.

Conceitos Básicos

- **Métodos**
  - São as operações (funções) que um objeto pode realizar.
  - Os métodos podem ser usados para:
    - Apresentar ou alterar o valor de um atributo do objeto;
    - Expor as funcionalidades que um objeto pode oferecer.
- **Objetos**
  - São representações reais de uma classe.
  - Também são conhecidos como instâncias da classe

Conceitos Básicos

- Em Java, de uma forma geral, classes e atributos são especificados da seguinte forma:

```
/* classe */
public class NomeDaClasse{
   // Corpo da classe
}

/* atributo */
<tipo> nomeAtributo [= valorInicial];
```

Conceitos Básicos

- Exemplo:

```
public class Pessoa {
      int idade = 0;
      String nome;
      boolean casado;
}
```

- Note também que podemos especificar que um atributo será inicializado com um valor específico. Neste exemplo, o atributo idade é inicializado com 0 toda vez que for criado um objeto da classe Pessoa.

Conceitos Básicos

- De uma forma geral, métodos são definidos da seguinte forma:

```
<TipoRetorno> nomeMetodo (<Parâmetros>) {
    // Corpo do método
}
```

- Onde Parâmetros:

```
<tipoParam> <nomeParam>, ...
```

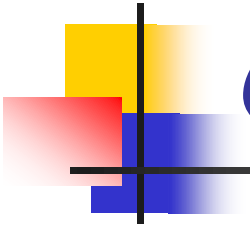
Conceitos Básicos

- Exemplo:

```
public class Circulo {
   double raio;

   double comprimento( ) {
      return ( 2 * 3.14 * raio);
   }
   double area( ) {
      return (3.14 * raio * raio);
   }
}
```

- Para descobrir que métodos devemos implementar, podemos fazer a seguinte questão:
  - Quais as operações típicas realizadas sobre um círculo?



Construtor

- Além de métodos e atributos, a definição de uma classe também inclui a implementação de **construtores**.
- **Construtores** são métodos especiais utilizados para inicializar objetos, podendo inicializar os valores dos seus atributos.
- Não apresentam um tipo de retorno explícito e possuem o mesmo nome da classe.

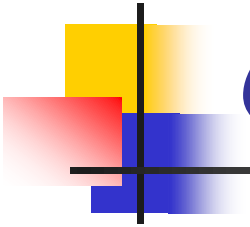
- Cada classe pode ter um ou mais construtores sobrecarregados (**overloading - sobrecarga**), variando-se na quantidade e no tipo dos parâmetros fornecidos.

Construtor

- Exemplos:

```
/* Construtor default */
Circulo ( ) {
     raio = 2;
}
/* Construtor com argumento */

Circulo(double r) {
     raio = r;
}
```



Construtor

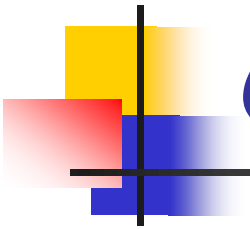
- Criando um objeto
  1. Escrever uma aplicação que declara uma variável do tipo de dado (classe) que se deseja criar um objeto.
  2. Inicializar a variável com uma chamada ao construtor da classe usando a palavra-chave **new**.

OBS: A aplicação de *new* pode ser comparada a chamada de funções de alocação de memória em C, tal como *malloc*.

Construtor

- Criando um objeto - Exemplo 1:

```
public class Exemplo {
  public static void main(String[] args){
    System.out.println("Criando objetos...");
    Circulo c1, c2, c3;
    c1 = new Circulo();
    c2 = new Circulo(5);
    c3 = new Circulo(2);
    System.out.println("Objetos criados!");
  }
}
```



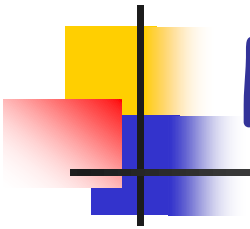
Construtor

- Destruindo um Objeto
  - Ao contrário de linguagens como C que precisam liberar memória explicitamente através de comandos como o free( ), em Java não é necessário preocupar-se com isso.
  - A liberação de memória é feita de forma automática pelo coletor de lixo.
  - Objetos que não estão mais sendo utilizados ou referenciados por variáveis do programa têm seu espaço de memória liberado.

Construtor

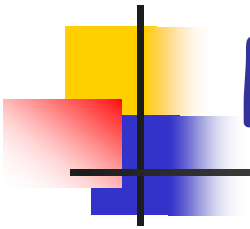
- Criando um objeto - Exemplo 2:

```
public class Exemplo {
   public static void main(String[] args){
      System.out.println("Criando objetos...");
      Circulo c1, c2;
      c1 = new Circulo();
      c1 = new Circulo(5);
      c2 = c1;
      c2 = new Circulo(10);
      System.out.println("Objetos criados!");
   }
}
```



Palavra-Chave *this*

- **this** refere-se a um objeto.
- Pode ser usado para indicar que os atributos ou métodos que estão sendo acessados pertencem ao próprio objeto.
- É muito útil para diferir os atributos da classe em relação aos parâmetros e variáveis locais de um método.

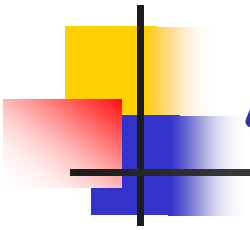


Palavra-Chave *this*

- Exemplos:

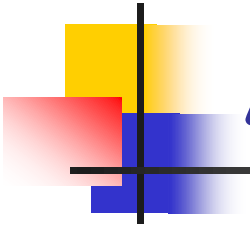
```
/* Construtor */
Circulo(double raio) {
      this.raio = raio;
}
```

```
/* Método */
double area( ) {
      double valorArea = 0;
      valorArea = 3.14 * this.raio * this.raio;
      return valorArea;
}
```



Acessando Atributos

- Para acessar atributos de um objeto, utiliza-se o nome do objeto (**e não da classe**), seguido do caractere ponto (.), mais o nome do atributo que deseja-se acessar.
- Sintaxe: `objeto.nomeAtributo`



Acessando Métodos

- A sintaxe utilizada para acionar métodos do objeto é idêntica a sintaxe utilizada para acessar os atributos.
- Sintaxe: `objeto.nomeMetodo( )`

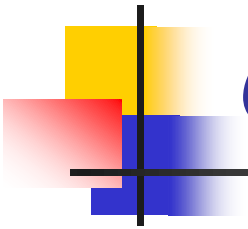
Acessando Atributos e Métodos

- Exemplo:

```
public class Exemplo {
 public static void main(String[] args) {
  Circulo c1;
  c1 = new Circulo(5);
  double r = c1.raio;
  double a = c1.area();
  System.out.println("Raio de c1 = " + r);
  System.out.println("Área de c1 = " + a);
  c1.raio = 10;
  System.out.println("Raio de c1 = " + c1.raio);
  System.out.println("Área de c1 = " + c1.area( ));
 }
}
```

Conceitos Básicos

- Exercício 1:
  - Implementar a classe **Quadrado** com as seguintes definições:
    - O atributo lado;
    - Dois construtores, sendo um deles default.
    - O método double area( );
    - O método double comprimento( );
    - O método void desenha( );
    - Implementar a aplicação **AplicacaoQuadrado** que cria um objeto do tipo Quadrado, a partir do lado informado pelo usuário, e imprime o desenho do quadrado, o valor da sua área, comprimento e lado.



Conceitos Básicos

- Exercício 2:
  - Implementar a classe **TrianguloRetangulo** com as seguintes definições:
    - Os atributos base, altura e hipotenusa;
    - Dois construtores, sendo um deles default.
    - O método double area( );
    - O método double comprimento( );
    - Implementar a aplicação **AplicacaoTriangulo** que cria um objeto do tipo TrianguloRetangulo, a partir de valores informados pelo usuário, e imprime o valor da sua área, comprimento e atributos.