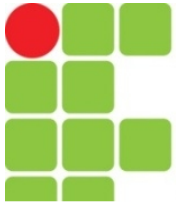

Instituto Federal de Santa Catarina – IFSC
Campus São José

Programação Orientada a Objetos

Sobrecarga de Métodos e Palavras Reservadas: *this*, *static* e *final*

Prof. Francisco de Assis S. Santos, Dr.

São José, 2014.

Sobrecarga de Métodos

- Métodos sobrecarregados devem possuir assinaturas diferentes
  - Neste caso, a assinatura de método e representada pelo número de parâmetros e pelo tipo dos parâmetros

```
public class Data{
   private int dia, mes, ano;

   public void alterarData(int d){
      this.dia = d;
   }
   public void alterarData(int d, int m){
      this.dia = d; this.mes = m;
   }
   public void alterarData(int d, int m, int a){
      this.dia = d; this.mes = m; this.ano = a;
   }
}
```

```
Data d = new Data();
d.alterarData(31);
d.alterarData(31,12);
d.alterarData(31,12,1969);
```

Palavra Reservada *this*

```
public class Complexo{
   private int real;
   private int imaginario;

   public Complexo(int a, int b){
      this.real = a; this.imaginario = b;
   }
   public void soma(Complexo c){
      this.real = this.real + c.real;
      this.imaginario = this.imaginario + c.imaginario;
   }
   public void imprimir(){
      System.out.println(this.real + "," + this.imaginario);
   }
}
```

```
Complexo a = new Complexo(1,2);
Complexo b = new Complexo(3,4);
a.soma(b);
a.imprimir();
b.imprimir();
```

Palavra Reservada *this*

```
public class Bicicleta{
   private double valor;

   public void imprimirValor(){
      System.out.println(this.valor
           );
   }
}
```

```
public class Principal{
   private double valor;

   public void imprimirValor(){
      System.out.println(this.valor);
   }
   public void teste(){
      Bicicleta b = new Bicicleta();
      b.imprimirValor();
   }
   public static void main(String[]
        args){
       Principal p = new Principal();
       p.imprimirValor();
       p.teste();
   }
}
```

- Na instrução acima identificada o *this* é referência para o objeto de qual classe? Principal ou Bicicleta?
- Em relação a instrução: p.teste, pode ser substituída por this.teste?

Membros de classe estáticos: *static*

Atributos **não** estáticos - cada instância da classe terá uma copia distinta deste atributo.

```
public class Celular{
   private int total;
   public Celular(){
      this.total = this.total + 1;
   }
   public void incrementar(){
      this.total = this.total + 1;
   }
   public int getTotal(){
      return this.total;
   }
}
```

```
Celular a = new Celular();
Celular b = new Celular();
a.incrementar();  b.incrementar();

System.out.println(a.getTotal()); // o que sera' impresso?
System.out.println(b.getTotal()); // o que sera' impresso?
```

Membros de classe estáticos: *static*

•Atributos estáticos **ficam comuns para todos os objetos** que foram instanciados para esta classe, sendo assim chamados de “atributos da Classe”

- Não se pode usar o *this* para acessar um membro estático. Deve-se usar o nome da Classe

• Classes podem possuir métodos estáticos e estes podem ser invocados sem que necessite criar uma instância da classe

- Métodos estáticos geralmente são usados para acessar atributos estáticos

Membros de classe estáticos: *static*

```
public class Celular{
    private static int total = 0;
    private int serial;

    public Celular(int s){
       this.serial = s;
       Celular.total = Celular.total + 1;
    }
    public static int getTotal(){
       return Celular.total;
    }
    public int getSerial(){
       return this.serial;
    }
}
```

```
System.out.println(Celular.getTotal()); // o que sera' impresso?
Celular c = new Celular(123);
Celular d = new Celular(456);
System.out.println(Celular.getTotal()); // o que sera' impresso?
System.out.println(d.getSerial()); // o que sera' impresso?
```

Modificador *final*

O modificador final pode ser usado em atributos ou métodos de uma classe, bem como em variáveis locais

- Uma vez que atribuiu valores para variáveis ou atributos, estes não poderão ser alterados
- Por convenção, constantes deverão ser escritas em letras maiúsculas
- Métodos não poderão ser sobrescritos (conceito de herança)

Modificador *final*

```
public class Celular{
   private final int FREQUENCIA = "1800";
   private final int SERIAL;

   public Celular(int s){
    this.SERIAL = s;
   }

   public final void iniciarChamada(){
   // ....
   }
}
```

Resumo

- **Sobrecarga de métodos**
  - Uma classe pode ter mais de um método com o mesmo nome, porém com assinaturas diferentes
- **A palavra *this* e uma referência para o objeto atual**
  - Apesar de não obrigatório na maioria dos casos, seu uso é desejado para facilitar a leitura do código
- **Atributos estáticos ficam comum para todos objetos instanciados da classe**
  - Imagine que e uma variável compartilhada entre todos os objetos da classe
- **Modificador final e usado para definir constantes**
  - Atributos final não poderão ter seu valor alterado

Exercício 1

Analise a classe utilitária Math (java.lang.Math) e as afirmativas abaixo:

- http://docs.oracle.com/javase/7/docs/api/java/lang/Math.html
  - Para obter a raiz quadrada do numero 4, basta: double d = Math.sqrt(4);
  - Math.PI e uma constante que contem o valor aproximado de PI. Para imprimir este valor, basta: System.out.println(Math.PI).

Responda:
Dos conceitos apresentados nesta aula, quais deles a classe java.lang.Math faz uso? Justifique sua resposta

Exercício 2

Crie uma classe utilitária para trabalhar com datas. A classe deverá prover as seguintes funcionalidades:

- Receber uma data como parâmetro e retornar uma String com a data por extenso
  - Se receber somente um inteiro, então retornar o dia por extenso
  - Se receber dois inteiros, então retornar dia e mês por extenso
  - Se receber três inteiros, então retornar dia e mês por extenso e ano (não precisa ser por extenso)
- Receber um inteiro como parâmetro e retornar o nome do respectivo mês. Ex: ao passar o numero 2, a classe deve retornar “fevereiro”
- Receber duas datas como parâmetro no formato (dia, mês, ano) e retornar a diferença em dias entre estas (primeira - segunda)
- Indicar se o ano recebido como parâmetro e ou não um ano bissexto.

Em Java, invoque os métodos da classe.

Referências

Notas de aula do Prof. Emerson Ribeiro de Mello