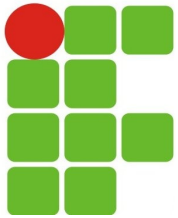
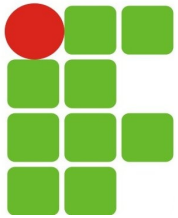


Encapsulamento e Métodos (Construtores e Estáticos)
Sobrecarga de Métodos

João Paulo Q. dos Santos

joao.queiroz@ifrn.edu.br



Roteiro

- Conceitos sobre Encapsulamento;
- Variável **this**;
- Métodos Construtores;
- Sobrecarga de Métodos;
- Métodos Acessadores e Modificadores;
- Métodos e atributos estáticos.

Encapsulamento

- É o processo de proteger os membros de uma classe (atributos e métodos), permitindo somente que os membros necessários (públicos) sejam acessados. Vantagens:
  - **Facilidade de manutenção:** A manutenção de classes pode ser efetuada de maneira isolada, ou seja, se um atributo tiver seu formato alterado, os usuários desta classe poderão continuar a usá-la sem se preocupar com a alteração.
  - **Redução de acoplamento:** Devemos evitar criar objetos usando o operador **new**. O mais recomendado é que a própria classe retorne um objeto de seu tipo, ou seja, devemos executar um método publico da classe (estático), que deverá retornar um objeto para uso. Quanto menor o acoplamento entre as classes, maior pode ser a reutilização dessa classe
  - **Segurança de informações:** Toda a comunicação com o objeto ocorre por meio dos métodos modificadores e e de acesso (set e get), oferecendo segurança na atribuição de novos valores.

Encapsulamento

- Existem quatro modificadores de acesso que devem ser explicitamente informados na criação de uma classe, método ou atributo:

Visibilidade	Public	Protected	Default ou Friendly	Private
Da mesma Classe	sim	sim	sim	sim
De alguma classe no mesmo pacote	sim	sim	sim	não
De uma classe fora do pacote	sim	não	não	não
De uma subclasse no mesmo pacote	sim	sim	sim	não
De uma subclasse fora do pacote	sim	sim	não	não

Métodos

- Os métodos são mecanismos utilizados na Orientação a Objetos para implementar as operações que os objetos serão passiveis de realizar.
- Os métodos podem ser:
  - Métodos Construtores;
  - Métodos Acessadores e Modificadores;
  - Métodos Estáticos;
  - Métodos Abstratos.

Construtores

- Sempre que uma classe é instanciada, o método construtor é invocado sobre a nova instancia;
- O compilador Java fornece um construtor padrão caso não seja definido nenhum outro construtor;
- O construtor é um método público, sem tipo de retorno, com o mesmo nome da classe;
- Caso o construtor receba argumentos, eles são especificados nos parênteses após o nome da classe no comando **new**;
- Podemos sobrecarregar construtores.

```
Circulo c = new Circulo ();
```

Sobrecarga de métodos

“É a capacidade de definir mais de um método com o mesmo nome, mas com assinaturas diferentes em uma única classe. O compilador na hora da ativação do método determina, por meio da assinatura, qual o método será ativado”

Sobrecarga de construtores

Sobrecarga

```
package br.edu.ifrn.exemplos;

public class Circulo {

    private String cor;
    private float raio;

    public Circulo (){
        this.cor = " ";
        this.raio = 0;
    }
    public Circulo (String cor){
        this.cor = cor;
        this.raio = 0;
    }
    public Circulo (float raio){
        this.cor = " ";
        this.raio = raio;
    }
    public Circulo (String cor, float raio){
        this.cor = cor;
        this.raio = raio;
    }
}
```

// Os Métodos Get e Sets foram omitidos

Variável this

- A variável **this** é uma referencia para o próprio objeto e deve ser usada nas seguintes situações:
  - Estando dentro de um construtor, podemos executar outro construtor que tenha uma assinatura diferente. Para isto a execução do comando de chamada para outro construtor deve ser o primeiro comando executado no método;
  - Resolver ambiguidade de nome entre um atributo e um parâmetro ou variável local de algum método;
  - Retornar a própria referência da instância em algum método.

Métodos Acessadores

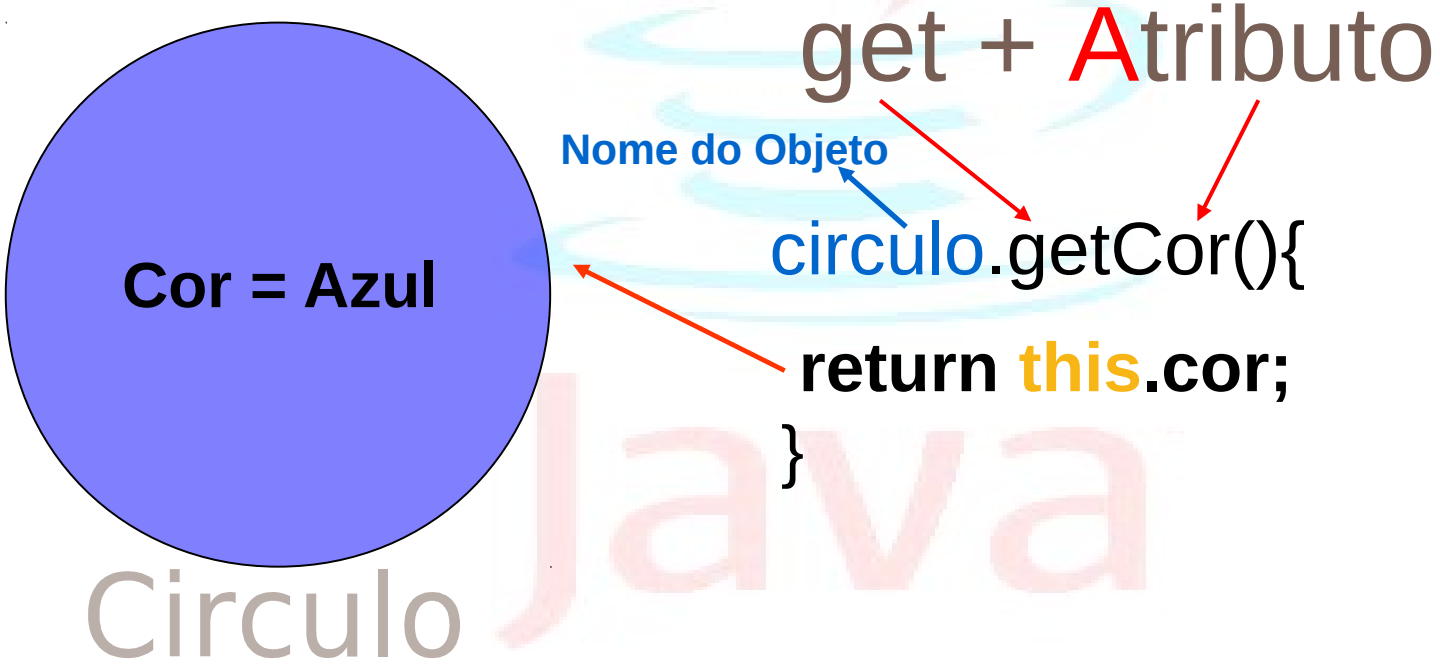
```
public class Circulo {

      private String cor;

       public String getCor (){
             return this.cor;
       }


       public void setCor (String cor){
             this.cor = cor;
       }

}
```

O nome dos métodos devem conter prefixo get seguido do nome do atributo que será realizada a leitura. Após o prefixo get o nome do atributo deve ter o primeiro caractere em maiúsculo.

O nome dos métodos devem conter prefixo set seguido do nome do atributo que será realizada a escrita. Após o prefixo set o nome do atributo deve ter o primeiro caractere em maiúsculo.

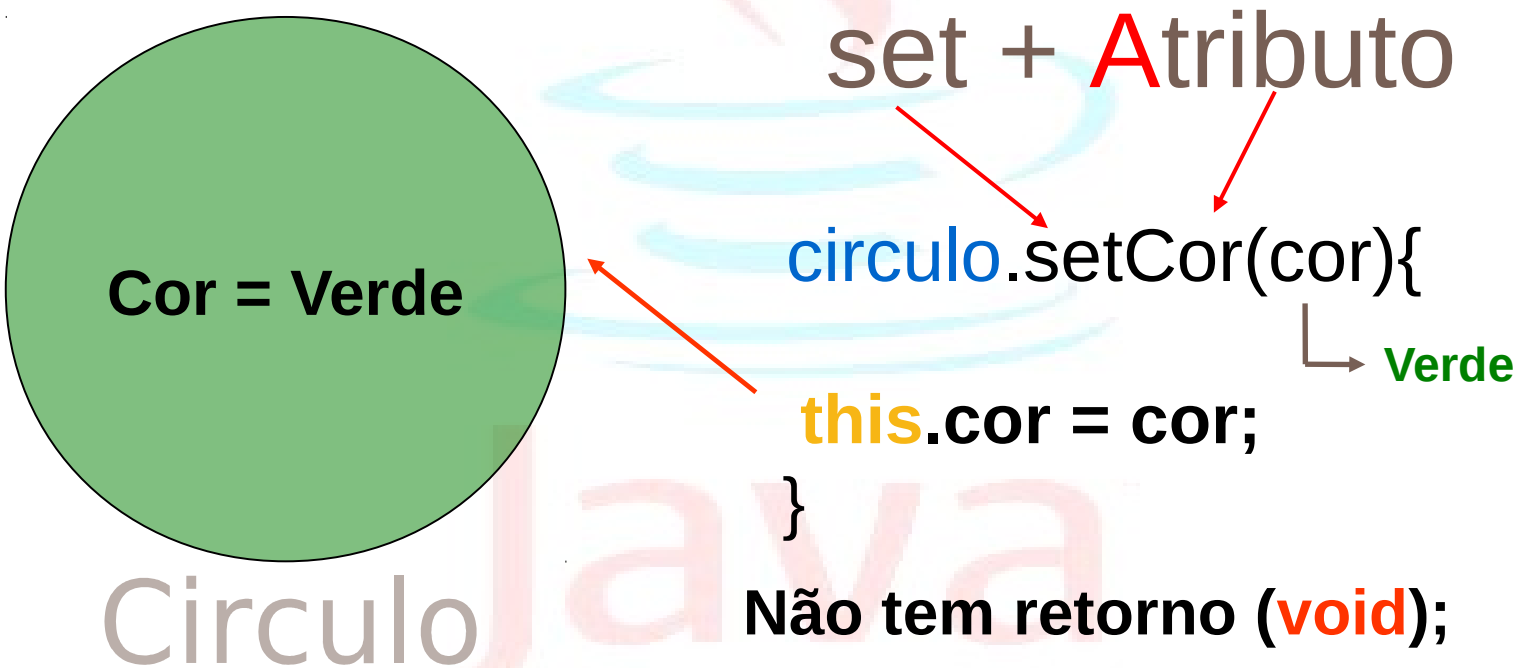
Metodos acessadores - GET

- Os métodos **gets** (**acessadores**) são geralmente utilizados para realizar uma leitura sobre um dado atributo de um objeto.



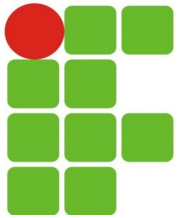
Metodos acessadores - SET

- Os métodos **sets** (**modificadores**) são utilizados quando se deseja alterar o estado de dado objeto.



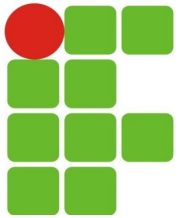
Método e atributos estáticos

- São métodos que pode ser utilizados sem precisar instanciar a classe;
- O mesmo ocorre para atributos do tipo estático:
  - Mas para que um atributo estático possa ser acessível é preciso que o mesmo seja público, ele é único independente da quantidade de objetos.
- O método ou variável estático faz com que ele pertençam à própria classe e não ao objeto.



Exemplo

```
package br.edu.ifrn.exemplos;

public class ExemploMetodoEstatico {

   /*atributo estático. Visível para
   todos os outros objetos*/
   private static int objetos = 0;

   /*o método ExemploMetodoEstatico
   incrementa o atributo estático objetos*/
   public ExemploMetodoEstatico() {
      objetos++;
   }

   // método estático
   public static int getObjetos() {
      return objetos;
   }
}
```



Exemplo

```
package br.edu.ifrn.exemplos;

public class ExemploUsandoMetodoEstatico {

    public static void main(String args[]) {

        // fazendo acesso ao método getObjetos de forma correta
        System.out.println("Quantidade objetos = "+
ExemploMetodoEstatico.getObjetos());

        System.out.println("Criando um objeto");

        ExemploMetodoEstatico d = new ExemploMetodoEstatico();

        //ocorre warning (aviso) pois estamos acessando um
metodo estatico
        d.getObjetos();

        System.out.println("Quantidade de objetos criados = " +
ExemploMetodoEstatico.getObjetos());
    }
}
```

Dúvidas