



**INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA**
RIO GRANDE DO SUL
Câmpus Feliz

Sobrescrita de métodos

Prof. Moser Fagundes

Programação II

Sumário

- Sobrescrita de construtores
- Sobrescrita de métodos
- Exercícios

Sobrescrita de métodos

Sobrescrita ou **overriding** é uma técnica utilizada para **modificar** o comportamento (*método*) de uma class-pai na classe-filha.

Consiste em sobrescrever (redefinir) um método já existente na superclasse.

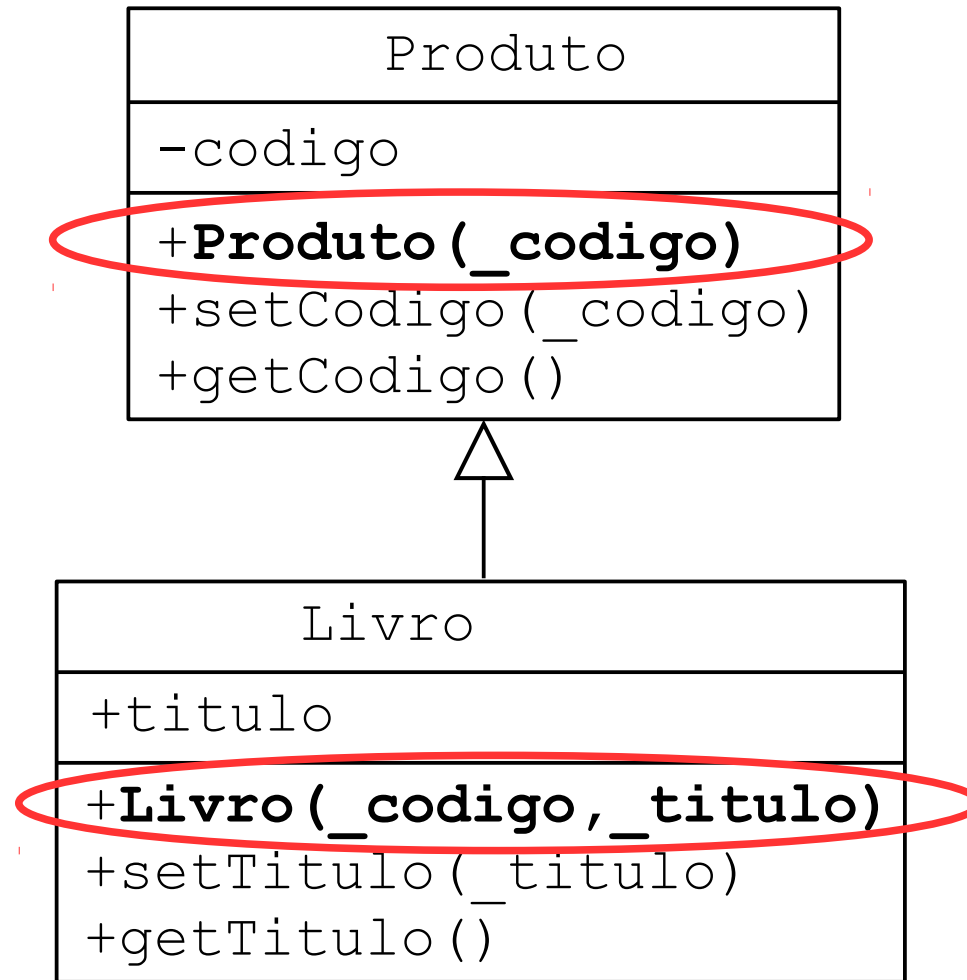
Podemos sobrescrever qualquer método, incluindo os construtores de classe.

Sobrescrita de construtores

- Na **sobrescrita** de construtores, não usamos o mesmo nome do método, uma vez que um construtor sempre tem o mesmo nome da classe a qual ele pertence.
- Porém, podemos chamar o construtor da classe pai usando a palavra-chave **super**.

Exemplo

Neste exemplo,
sobrescrevemos o
construtor da classe
Produto na
subclasse **Livro**



Exemplo


Crie a classe **Produto**, contendo o atributo privado **código** e métodos **get** / **set**.

Logo após, crie uma subclasse **Livro** que contém o atributo privado **título** e métodos **get** / **set**.

Na classe **Produto** adicione um construtor, e logo após, sobrescreva o construtor na subclasse **Livro**.

Exemplo de Produto

```
package produto;  
  
public class Produto {  
  
    private int codigo;  
  
    public Produto(int _codigo) {  
        codigo = _codigo;  
        System.out.println("Codigo foi inicializado!");  
    }  
}
```



Este construtor inicializa `codigo` e imprime uma mensagem avisando o usuário.


Livro subclasse de Produto

```
package produto;

public class Livro extends Produto {

    private String titulo;


    public Livro(int _codigo, String _titulo) {
        super(_codigo);
        titulo = _titulo;
        System.out.println("Titulo foi inicializado!");
    }
}
```



Este construtor inicializa `titulo` e imprime uma mensagem avisando o usuário, mas antes disso ele chama o construtor da superclasse para inicializar `codigo`.

Criando objetos Livro

```
package produto;  
  
public class Exemplo {  
  
    public static void main(String[] args) {  
  
        Livro l1 = new Livro(123, "Java Basico");  
        Livro l2 = new Livro(456, "O Mundo de Sofia");  
  
    }  
}
```



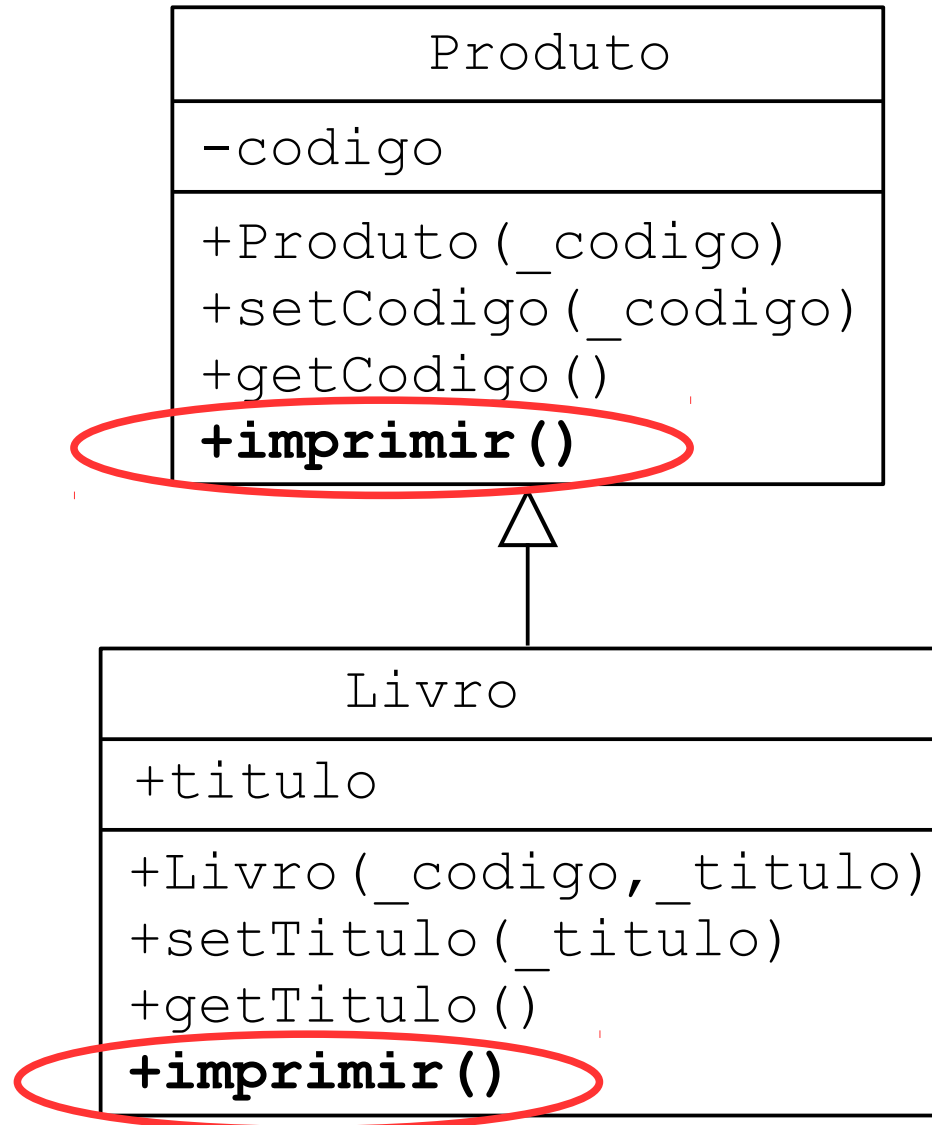
Quando criamos estes dois objetos Livro, as mensagens que colocamos nos construtores mostram o fluxo de execução.

Exemplo

No exemplo a seguir, vamos criar o método `imprimir()` na classe `Produto`, e logo após, vamos sobrescrever o método `imprimir()` na subclasse `Livro`.

Exemplo

Neste exemplo,
sobrescrevemos o
construtor da classe
Produto na
subclasse **Livro**



imprimir() em Produto


```
package produto;

public class Produto {

    private int codigo;

    public Produto(int _codigo) {
        codigo = _codigo;
    }

    public void imprimir() {
        System.out.println("Codigo: " + codigo);
    }
}
```



Nesta versão, retiramos as mensagens do construtor e criamos um método imprimir() que escreve o código do produto.

imprimir() em Livro


```
package produto;
```

```
public class Livro extends Produto {
```

```
    private String titulo;
```

```
    public Livro(int _codigo, String _titulo) {  
        super(_codigo);  
        titulo = _titulo;  
    }
```

```
    public void imprimir() {  
        super.imprimir();  
        System.out.println("Titulo: " + titulo);  
    }  
}
```



Na subclasse, sobrescrevemos imprimir() para imprimir o título do livro, mas antes chamamos o imprimir da superclasse.

Executando método imprimir()

```
package produto;  
  
public class Exemplo {  
  
    public static void main(String[] args) {  
  
        Livro l1 = new Livro(123, "Java Basico");  
        l1.imprimir();  
  
        Livro l2 = new Livro(456, "O Mundo de Sofia");  
        l2.imprimir();  
    }  
}
```



Após criarmos estes dois objetos Livro, os seus dados são impressos pelos seus respectivos métodos imprimir()

Exercícios

Lista no Moodle!