Orientação a Objetos 1

Métodos Abstratos

Prof. Dr. Vinícius Camargo Andrade

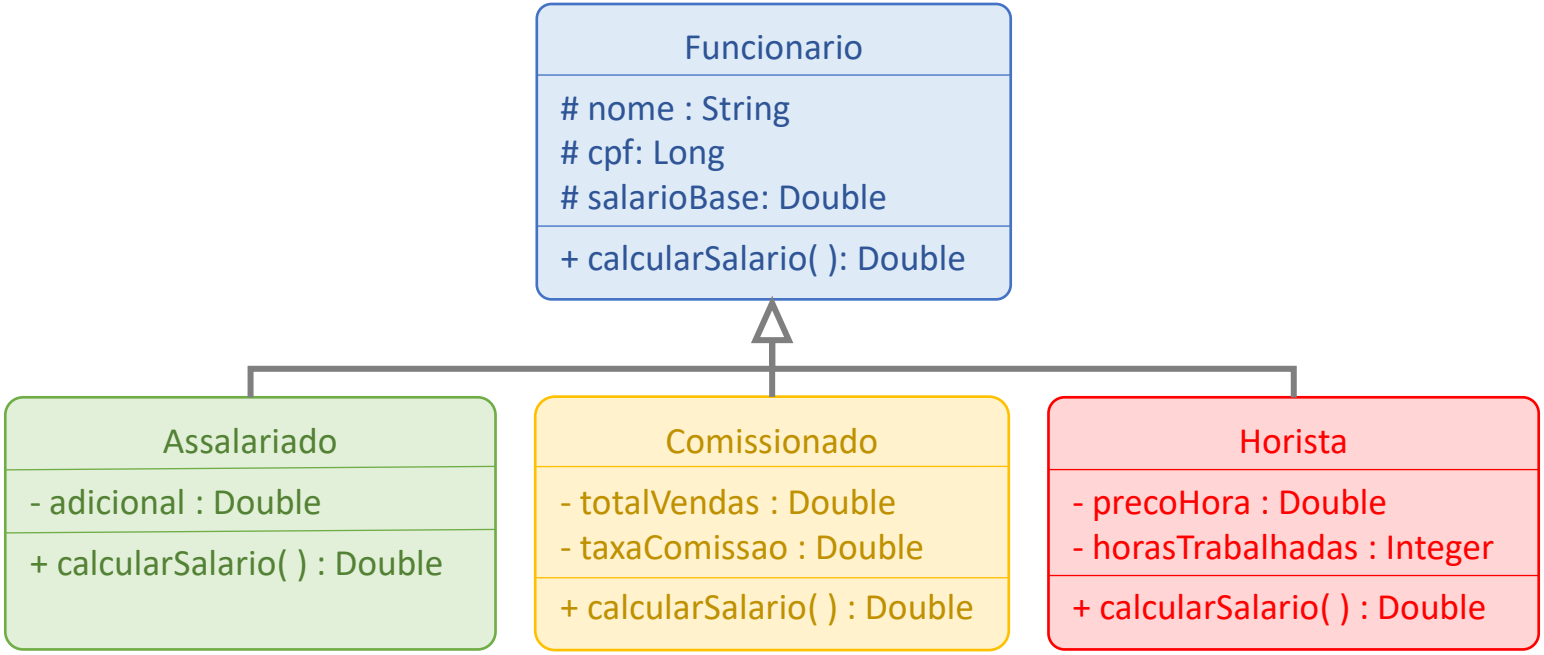
vcandrade@utfpr.edu.br

Departamento Acadêmico de Informática
Universidade Tecnológica Federal do Paraná

Cenário

Considere o mesmo exemplo do cálculo de salário para diferentes funcionários.

Cenário


Funcionario
nome : String
cpf: Long
salarioBase: Double
+ calcularSalario( ): Double
Assalariado
- adicional : Double
+ calcularSalario( ) : Double
Comissionado
- totalVendas : Double
- taxaComissao : Double
+ calcularSalario( ) : Double
Horista
- precoHora : Double
- horasTrabalhadas : Integer
+ calcularSalario( ) : Double

```java
public class Funcionario {

    protected String nome;
    protected Long cpf;
    protected Double salarioBase;

    public Funcionario(String nome, Long cpf, Double salarioBase) {

        this.nome = nome;
        this.cpf = cpf;
        this.salarioBase = salarioBase;
    }

    public Double calcularSalario() {

        return null;
    }
```

```java
public class Assalariado extends Funcionario {

    private Double adicional;

    public Assalariado(String nome, Long cpf, Double salarioBase, Double adicional) {

        super(nome, cpf, salarioBase);
        this.adicional = adicional;
    }

    @Override
    public Double calcularSalario() {

        return super.getSalarioBase() + this.getAdicional();
    }
```

```java
public class Comissionado extends Funcionario {

    private Double totalVendas;
    private Double taxaComissao;

    public Comissionado(String nome, Long cpf, Double salarioBase, Double totalVendas, Double taxaComissao) {

        super(nome, cpf, salarioBase);
        this.totalVendas = totalVendas;
        this.taxaComissao = taxaComissao;
    }

    @Override
    public Double calcularSalario() {

        return super.getSalarioBase() + (this.getTotalVendas() * this.getTaxaComissao());
    }
```

```java
public class Horista extends Funcionario {

    private Double precoHora;
    private Double horasTrabalhadas;

    public Horista(String nome, Long cpf, Double salarioBase, Double precoHora, Double horasTrabalhadas) {

        super(nome, cpf, salarioBase);
        this.precoHora = precoHora;
        this.horasTrabalhadas = horasTrabalhadas;
    }

    @Override
    public Double calcularSalario() {

        return super.getSalarioBase() + (this.getHorasTrabalhadas() * this.getPrecoHora());
    }
```

Problema

Problema

Nota-se que não há como o método calcularSalario( ) da classe Funcionário possuir uma implementação de maneira genérica, isto é, uma fórmula que possa ser útil aos funcionários Assalariado, Comissionado e Horista.

Problema

```java
public class Funcionario {

    protected String nome;
    protected Long cpf;
    protected Double salarioBase;

    public Funcionario(String nome, Long cpf, Double salarioBase) {

        this.nome = nome;
        this.cpf = cpf;
        this.salarioBase = salarioBase;
    }

    public Double calcularSalario() {

        return null;
    }
```

Problema

```java
public class Funcionario {

    protected String nome;
    protected Long cpf;
    protected Double salarioBase;

    public Funcionario(String nome, Long cpf, Double salarioBase) {

        this.nome = nome;
        this.cpf = cpf;
        this.salarioBase = salarioBase;
    }

    public Double calcularSalario() {

        return null;
    }
```

Solução

Solução

Para casos como este, utiliza-se métodos abstratos.

Métodos Abstratos

Métodos Abstratos

Métodos abstratos são úteis quando não há como implementar uma funcionalidade de maneira genérica.

Métodos Abstratos

Métodos abstratos são métodos que não possuem implementação.

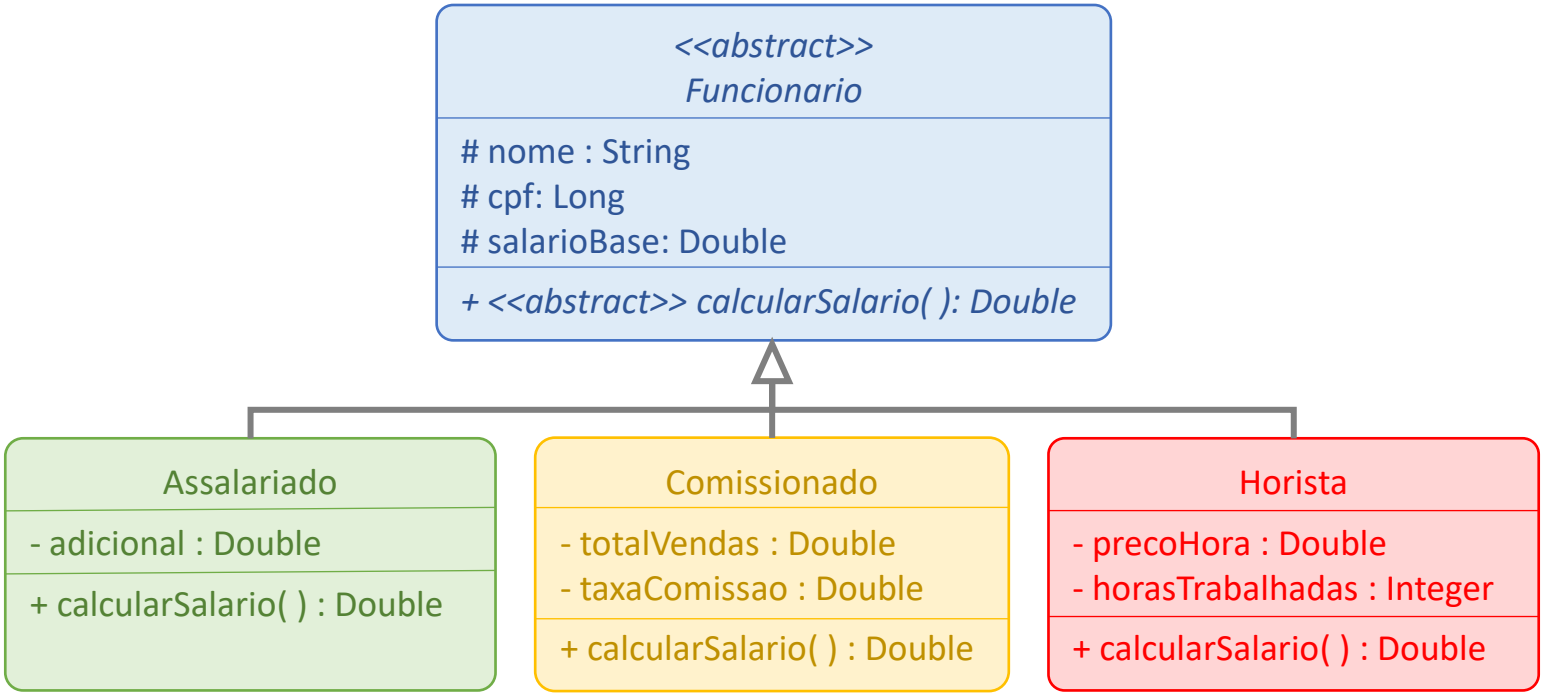
Métodos Abstratos

Se uma classe possuir pelo menos um método abstrato, obrigatoriamente esta classe deve ser abstrata.

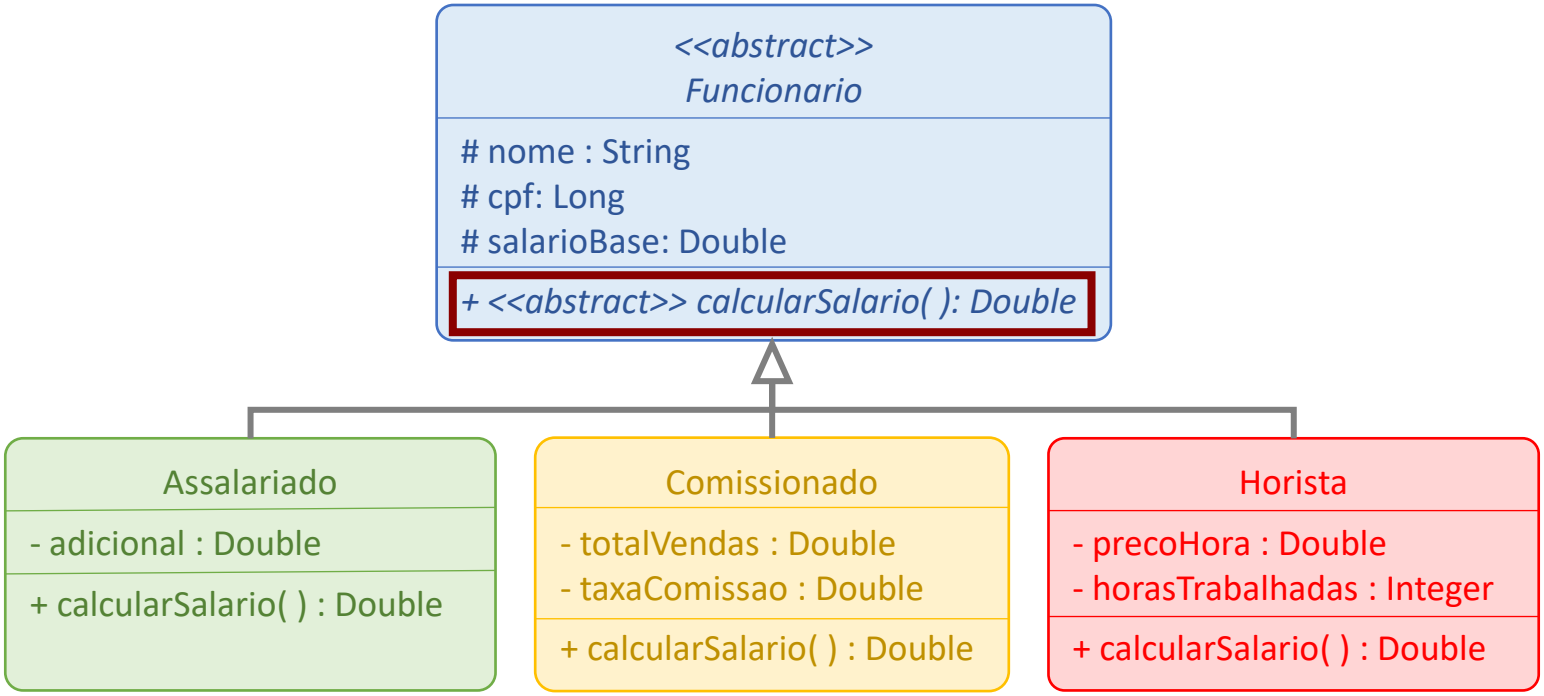
Modelo

Métodos Abstratos

Modelo


<<abstract>>
Funcionario
nome : String
cpf: Long
salarioBase: Double
+ <<abstract>> calcularSalario( ): Double
Assalariado
- adicional : Double
+ calcularSalario( ) : Double
Comissionado
- totalVendas : Double
- taxaComissao : Double
+ calcularSalario( ) : Double
Horista
- precoHora : Double
- horasTrabalhadas : Integer
+ calcularSalario( ) : Double

Modelo


<<abstract>>
Funcionario
nome : String
cpf: Long
salarioBase: Double
+ <<abstract>> calcularSalario( ): Double
Assalariado
- adicional : Double
+ calcularSalario( ) : Double
Comissionado
- totalVendas : Double
- taxaComissao : Double
+ calcularSalario( ) : Double
Horista
- precoHora : Double
- horasTrabalhadas : Integer
+ calcularSalario( ) : Double

Implementação

Métodos Abstratos

Implementação

Para se definir um método abstrato, usa-se a palavra reservada abstract na assinatura do método.

Implementação

```java
public abstract class Funcionario {

    protected String nome;
    protected Long cpf;
    protected Double salarioBase;

    public Funcionario(String nome, Long cpf, Double salarioBase) {

        this.nome = nome;
        this.cpf = cpf;
        this.salarioBase = salarioBase;
    }

    public abstract Double calcularSalario();

    public String getNome() {
        return nome;
    }
```

Implementação

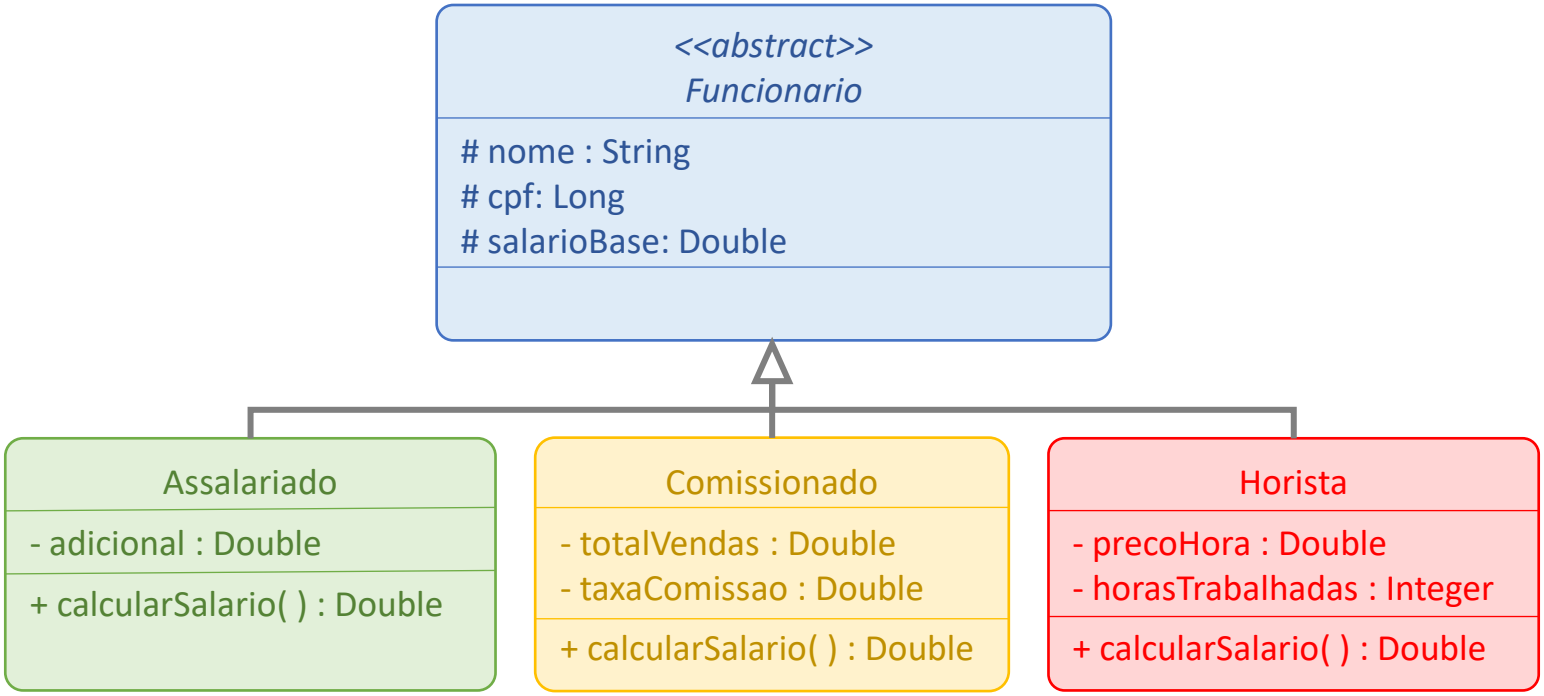
```java
public abstract class Funcionario {

    protected String nome;
    protected Long cpf;
    protected Double salarioBase;

    public Funcionario(String nome, Long cpf, Double salarioBase) {

        this.nome = nome;
        this.cpf = cpf;
        this.salarioBase = salarioBase;
    }

    public abstract Double calcularSalario();

    public String getNome() {
        return nome;
    }
```

Por que utilizar?

Por que utilizar?

Por que declarar um método na superclasse que não pode ser implementado de maneira genérica? Não seria fácil apenas declarar o métodos nas subclasses Assalariado, Comissionado e Horista?

Por que utilizar?



```java
public abstract class Funcionario {

    protected String nome;
    protected Long cpf;
    protected Double salarioBase;

    public Funcionario(String nome, Long cpf, Double salarioBase) {

        this.nome = nome;
        this.cpf = cpf;
        this.salarioBase = salarioBase;
    }

//    public abstract Double calcularSalario();
```

```java
public class Assalariado extends Funcionario {

    private Double adicional;

    public Assalariado(String nome, Long cpf, Double salarioBase, Double adicional) {

        super(nome, cpf, salarioBase);
        this.adicional = adicional;
    }

    public Double calcularSalario() {

        return super.getSalarioBase() + this.getAdicional();
    }
```

```java
public class Comissionado extends Funcionario {

    private Double totalVendas;
    private Double taxaComissao;

    public Comissionado(String nome, Long cpf, Double salarioBase, Double totalVendas, Double taxaComissao) {

        super(nome, cpf, salarioBase);
        this.totalVendas = totalVendas;
        this.taxaComissao = taxaComissao;
    }

    public Double calcularSalario() {

        return super.getSalarioBase() + (this.getTotalVendas() * this.getTaxaComissao());
    }
```

```java
public class Horista extends Funcionario {

    private Double precoHora;
    private Double horasTrabalhadas;

    public Horista(String nome, Long cpf, Double salarioBase, Double precoHora, Double horasTrabalhadas) {

        super(nome, cpf, salarioBase);
        this.precoHora = precoHora;
        this.horasTrabalhadas = horasTrabalhadas;
    }

    public Double calcularSalario() {

        return super.getSalarioBase() + (this.getHorasTrabalhadas() * this.getPrecoHora());
    }
```

```java
public class FuncionarioTeste {

    public static void main(String[] args) {

        String opcao = "Assalariado";

        Funcionario funcionario = null;

        switch (opcao) {
            case "Assalariado":
                funcionario = new Assalariado("Maria de Oliveira", 98765432199L, 5000.00, 2000.00);
                break;

            case "Comissionado":
                funcionario = new Comissionado("Carlos Santos", 15975398755L, 5000.00, 70000.00, 0.10);
                break;

            case "Horista":
                funcionario = new Horista("Bruna Rodrigues", 75375
                break;

            default:
                System.out.println("Opção inválida.");
        }
        System.out.println("Funcionário: R$" + funcionario.calcularSalario());
    }
}
```

cannot find symbol
  symbol: method calcularSalario()
  location: variable funcionario of type Funcionario

Dereferencing possible null pointer

(Alt-Enter shows hints)

Resposta

Ao declarar o método abstrato na superclasse, pode-se utilizar o conceito de polimorfismo, ou seja, a partir da declaração de um objeto do tipo genérico, pode-se invocar o também método genérico, porém em tempo de execução o interpretador executará o método do objeto instanciado.

```java
public abstract class Funcionario {

    protected String nome;
    protected Long cpf;
    protected Double salarioBase;

    public Funcionario(String nome, Long cpf, Double salarioBase) {

        this.nome = nome;
        this.cpf = cpf;
        this.salarioBase = salarioBase;
    }

    public abstract Double calcularSalario();

    public String getNome() {
```

```java
public class Assalariado extends Funcionario {

    private Double adicional;

    public Assalariado(String nome, Long cpf, Double salarioBase, Double adicional) {

        super(nome, cpf, salarioBase);
        this.adicional = adicional;
    }

    @Override
    public Double calcularSalario() {

        return super.getSalarioBase() + this.getAdicional();
    }
```

```java
public class Comissionado extends Funcionario {

    private Double totalVendas;
    private Double taxaComissao;

    public Comissionado(String nome, Long cpf, Double salarioBase, Double totalVendas, Double taxaComissao) {

        super(nome, cpf, salarioBase);
        this.totalVendas = totalVendas;
        this.taxaComissao = taxaComissao;
    }

    @Override
    public Double calcularSalario() {

        return super.getSalarioBase() + (this.getTotalVendas() * this.getTaxaComissao());
    }
```

```java
public class Horista extends Funcionario {

    private Double precoHora;
    private Double horasTrabalhadas;

    public Horista(String nome, Long cpf, Double salarioBase, Double precoHora, Double horasTrabalhadas) {

        super(nome, cpf, salarioBase);
        this.precoHora = precoHora;
        this.horasTrabalhadas = horasTrabalhadas;
    }

    @Override
    public Double calcularSalario() {

        return super.getSalarioBase() + (this.getHorasTrabalhadas() * this.getPrecoHora());
    }
```

```java
public class FuncionarioTeste {

    public static void main(String[] args) {

        String opcao = "Assalariado";

        Funcionario funcionario = null;

        switch (opcao) {
            case "Assalariado":
                funcionario = new Assalariado("Maria de Oliveira", 98765432199L, 5000.00, 2000.00);
                break;

            case "Comissionado":
                funcionario = new Comissionado("Carlos Santos", 15975398755L, 5000.00, 70000.00, 0.10);
                break;

            case "Horista":
                funcionario = new Horista("Bruna Rodrigues", 75375375398L, 5000.00, 25.00, 40.00);
                break;

            default:
                System.out.println("Opção inválida.");
        }
        System.out.println("Funcionário: R$" + funcionario.calcularSalario());
    }
}
```

```
run:
Funcionário: R$7000.0
BUILD SUCCESSFUL (total time: 0 seconds)
```

Exercício

Exercício

Desenvolva um sistema que possibilite realizar o cálculo da área e perímetro das seguintes formas geométricas: Círculo, Quadrado e Retângulo. Sabe-se que o círculo, quadrado e retângulo possuem uma cor. Além disso, o círculo possui um raio. O quadrado, um lado. E o retângulo, base e altura.