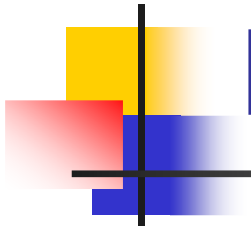
Java Standard Edition (JSE)

Capítulo 07. Classes abstratas

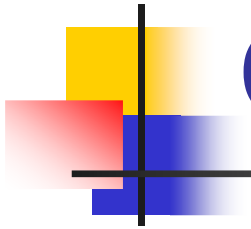
Esp. Márcio Palheta

MSN: marcio.palheta@hotmail.com



Novos recursos a aprender

- Conceito de classes abstratas;
- Declaração e uso de classes abstratas;
- Identificação de oportunidades de uso;



Cenário 01 - Bonificação

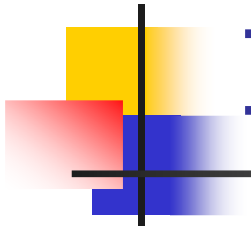
- A empresa Empretech tem o hábito de pagar gratificação a seus funcionários, conforme visto no capítulo anterior;
- Um colaborador pode ser Gerente ou um funcionário comum;
- O pagamento das gratificações deve ser armazenado
- A seguir, veremos as classes citadas;

Cenário 01: Classe Funcionario

```
public class Funcionario {
    private String nome;
    private String cpf;
    private double salario;

    public double getBonificacao(){
        return this.salario * 1.2;
    }

    public String getNome() {..
    public void setNome(String nome) {..
    public String getCpf() {..
    public void setCpf(String cpf) {..
    public double getSalario() {..
    public void setSalario(double salario) {..
}
```

Cenário 01: ControleBonificacao

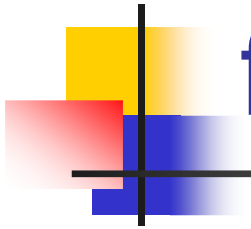
```
public class ControleBonificacoes {
    private double totalDeBonificacoes = 0;

    public void registra(Funcionario funcionario) {
        System.out.println("Adicionando bonificacao " +
                "do funcionario: " + funcionario);
        totalDeBonificacoes += funcionario.getBonificacao();
    }

    public double getTotalDeBonificacoes() {
        return this.totalDeBonificacoes;
    }
}
```



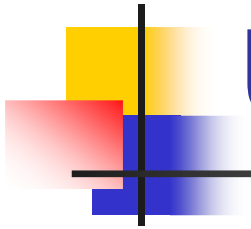
Itens importantes:

- O método registra() recebe qualquer referência a Funcionario ou um subtipo;
- A classe Funcionario está sendo usada para o polimorfismo;
- Com isso, podemos criar métodos genéricos que podem ser utilizados por seus objetos;
- OPS! E se fosse criada uma classe genérica? Abstrata?

Cenário: Outros tipos de funcionário

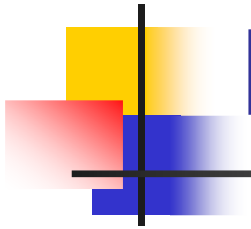


- Após a definição do plano de cargos, ficou definido que todo funcionário deve ser: Diretor, Gerente ou Operário;
- A classe Funcionario poderia conter os métodos e atributos comuns, mas deixa de fazer sentido o comando:
  - Funcionario func = new Funcionario();
- E o que fazer para evitá-lo?



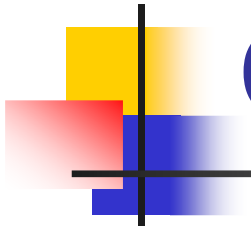
Uma classe modelo

- O que passa a ser nossa classe Funcionario, uma vez que a empresa só tem Diretor, Gerente ou Operário?
- A classe Funcionario oferece um modelo com as funcionalidades comuns;
- O modelo não pode ser instanciado;
- A implementação desse modelo fica a cargo de suas subclasses concretas;



Pessoa Física ou Jurídica

- Imagine a classe Pessoas e duas classes filhas: PessoaFisica e PessoaJuridica;
- Em um cadastro, por exemplo, o cliente sempre será do tipo PessoaFisica ou PessoaJuridica;
- Faz sentido que o cliente seja simplesmente do tipo Pessoa?
- A classe pessoa serve apenas de modelo, permitindo o polimorfismo;



Classes abstratas

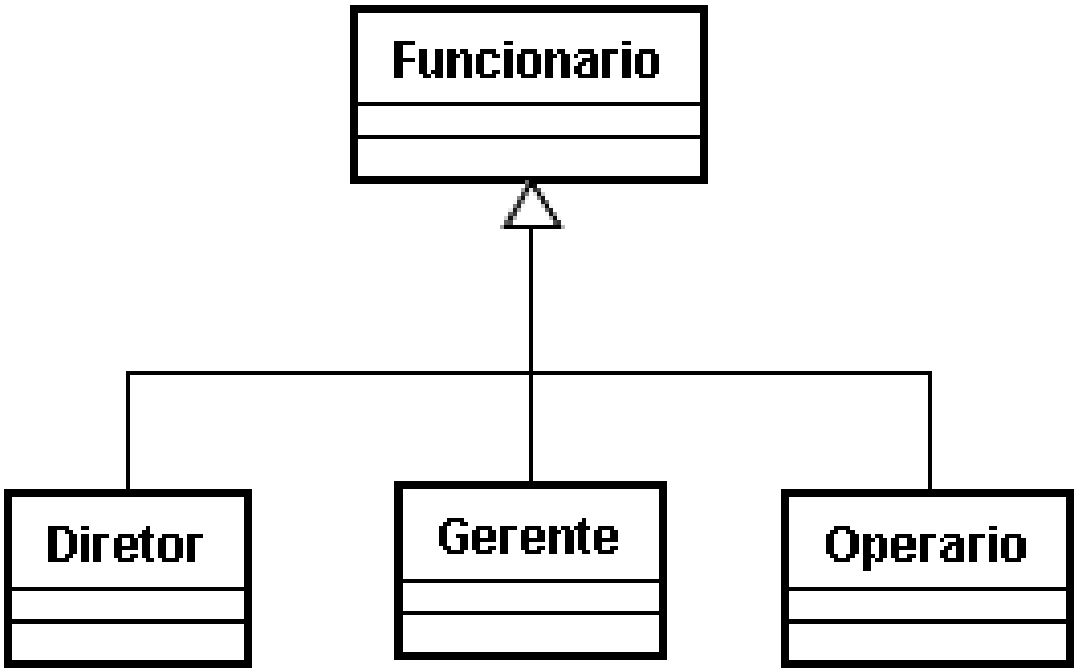
- São classes que servem de modelo para outras classes;
- Não podem ser instanciadas;
- A implementação deve ocorrer em suas subclasses concretas;
- Utilizadas para mantermos o polimorfismo;

Classe abstrata Funcionario: abstract class

```
1  package br.fucapi.sje.model.bean;
2
3  abstract class Funcionario {
4      private String nome;
5      private String cpf;
6      private double salario;
7
8      public double getBonificacao(){
9          return this.salario * 1.2;
10     }
11
12     public String getNome() {
15     public void setNome(String nome) {
18     public String getCpf() {
21     public void setCpf(String cpf) {
24     public double getSalario() {
27     public void setSalario(double salario) {
30 }
```

Representação Gráfica UML

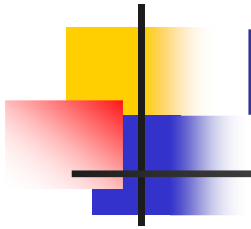
- Diagrama de classes



A classe concreta Gerente

- A classe Gerente é filha da classe Funcionario e, aqui, sobrescreve o método getBonificacao();

```
1 package br.fucapi.jse.model.bean;
2
3 public class Gerente extends Funcionario{
4     public double getBonificacao(){
5         return this.getSalario() * 1.4;
6     }
7 }
```



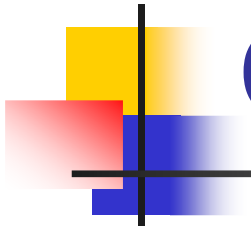
Métodos abstratos

- Se o método getBonificacao não fosse sobrescrito, seria herdado da classe mãe, retornando 20% do salário;
- Uma vez que cada tipo de funcionário tem um percentual diferente para o cálculo, faz sentido implementá-lo na classe Funcionario?
- Precisamos que cada classe sobrescreva o método getBonificacao();

Método abstrato: getBonificacao()

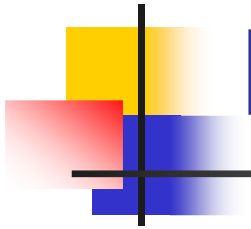
- Para garantir que cada subclasse implemente um determinado método, devemos declará-lo como abstrato:

```
3  abstract class Funcionario {
4      private String nome;
5      private String cpf;
6      private double salario;
7
8      public abstract double getBonificacao();
9
```



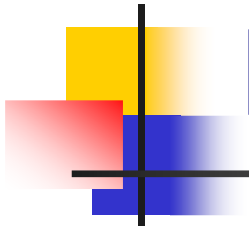
Considerações finais

- Qualquer classe filha da classe Funcionario, é obrigada a implementar o método getBonificacao();
- Caso o método não seja implementado, ocorrerá erro de compilação;



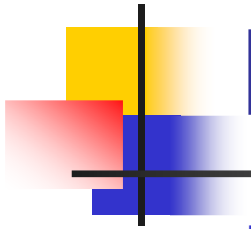
Exercício 01

- Crie um novo Java Project: Capitulo07
- Utilize o pacote: br.fucapi.treinamento.jse
- Crie a classe abstrata Funcionario, com atributos, métodos de get e set, além do método abstrato getBonificacao();
- Crie as classes Gerente e Operario, filhas de Funcionario;
- O que acontece quando criamos as classes filhas;



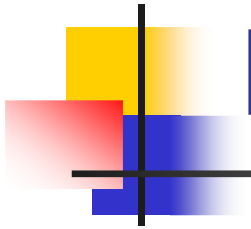
Exercício 02

- Na classe Funcionário, inclua o campo int matricula, apenas com método de get;
- Altere o código de Funcionario para que a matrícula seja gerada automaticamente
- Crie uma classe teste, com as seguintes opções:
  - Criar e armazenar N funcionários (ArrayList);
  - Imprimir total de bonificações por grupo: Gerente, Operário ou ambos;



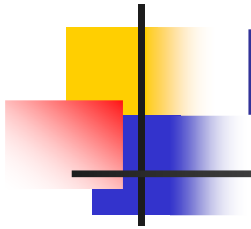
Exercício 03

- Desenvolva uma classe abstrata que contenha as características básicas de polígonos genéricos no plano (triângulos, retângulos etc).
- Características: nome, número de lados e o tamanho de cada lado;
- Métodos comuns: impressão e alteração de dados, cálculo da área e perímetro;
- Defina subclasses triângulos, retângulos e quadrado.
- Desenvolva um programa de teste que permita:
  - Criar e armazenar N polígonos (Array dinâmico);
  - Listar os dados de polígonos armazenados;
  - Excluir um polígono, de acordo com o nome;
  - Consultar os dados de um polígono, passando o nome;



Bibliografia

- Java - Como programar, de Harvey M. Deitel
- Use a cabeça! - Java, de Bert Bates e Kathy Sierra
- (Avançado) Effective Java Programming Language Guide, de Josh Bloch



Referências WEB

- SUN: www.java.sun.com

Fóruns e listas:

- Javaranch: www.javaranch.com
- GUJ: www.guj.com.br

Apostilas:

- Argonavis: www.argonavis.com.br
- Caelum: www.caelum.com.br

Java Standard Edition (JSE)

Capítulo 07. Classes abstratas

Esp. Márcio Palheta

MSN: marcio.palheta@hotmail.com