PROGRAMAÇÃO ORIENTADA A OBJETOS

Prof. Milton Palmeira Santana

Classes Abstratas

- Você sabe que o principal motivo pelo qual se modela uma classe é a criação de objetos por meio da sua instanciação. Pelo menos é assim que temos feito. No entanto, por alguns motivos plenamente justificáveis poderemos criar classes a partir das quais não se pode gerar instâncias.

- A classe abstrata é uma classe que não permite a geração de instâncias a partir dela, isto é, não permite que sejam criados objetos.

Classes Abstratas

- Imagine que estamos modelando a classe Pessoa, com todos aqueles atributos que já conhecemos: nome, data de nascimento, endereço e estado civil, entre vários outros. Se considerarmos a necessidade da divisão do modelo “pessoa” em outros dois (pessoa física e pessoa jurídica, por exemplo), chegaríamos à conclusão de que o ente “pessoa” é capaz de agrupar alguns atributos comuns aos dois subtipos de pessoa e, por isso, pode servir de base para a geração de pessoa física e pessoa jurídica.

- Assim, na condição de classe que servirá apenas como plataforma para a criação de outras, a classe Pessoa pode ser declarada como abstrata.

Classes Abstratas

» Código:

```
public abstract class Pessoa {
// corpo da classe.
}
```

» Já sabemos que o uso de classe abstrata auxilia quando atributos (preferencialmente vários) podem ser definidos para outros objetos de um dado tipo. Há, contudo, a possibilidade de que sejam criados também métodos abstratos, que compartilham comportamentos como outros objetos. Assim, cada método não implementado na classe abstrata também é indicado como abstract, embora esse expediente possa ser efetivado por meio das interfaces.

Classes Abstratas

- Exemplo Classe Funcionário:
- O que exatamente vem a ser a nossa classe Funcionario?

- Nossa empresa tem apenas Diretores, Gerentes, Secretárias, etc. Ela é uma classe que apenas idealiza um tipo, define somente um rascunho.

- Utilizamos a palavra-chave abstract para impedir que ela possa ser instanciada.

- Então, herdemos essa classe reescrevendo o método getBonificacao:

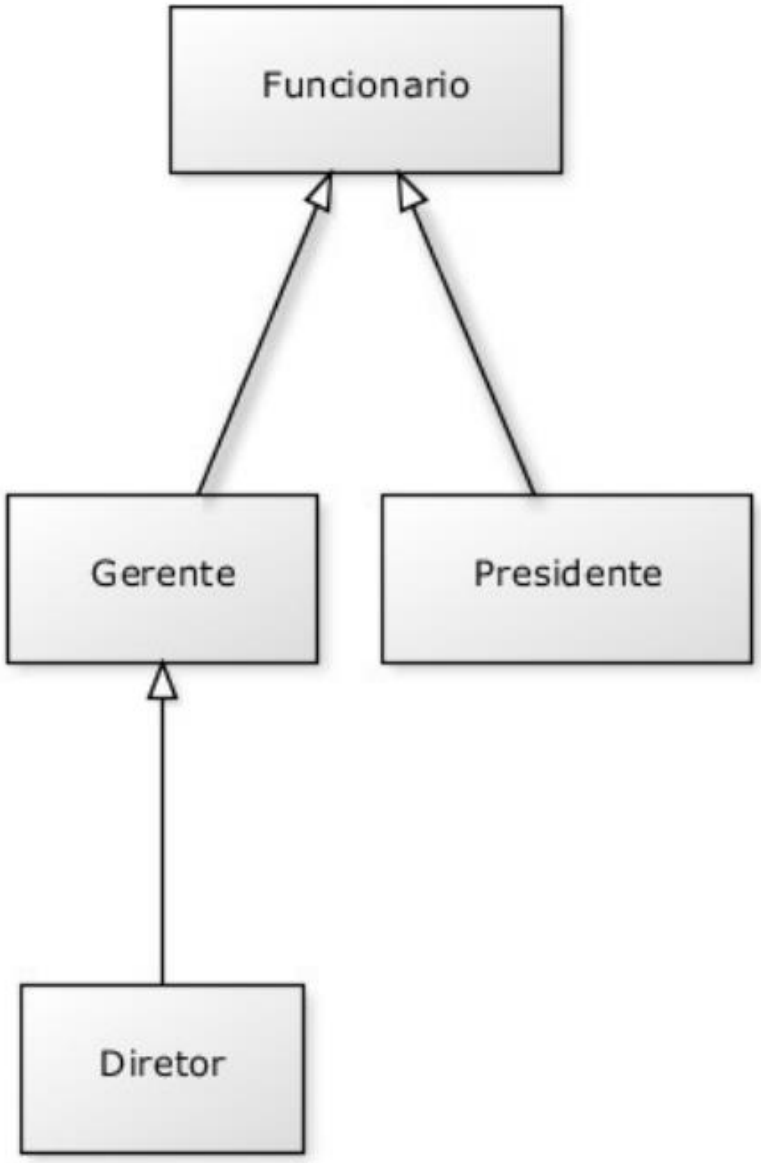
Anhanguera

Classes Abstratas

» Exemplo Classe Funcionário (tente criar um objeto):

```
public abstract class Funcionario {

    protected double salario;

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

    // outros atributos e métodos comuns a todos Funcionarios

}
```

Classes Abstratas



Classes Abstratas

- Mas qual é a real vantagem de uma classe abstrata? Poderíamos ter feito isso com uma herança comum. Por enquanto, a única diferença é que não podemos instanciar um objeto do tipo Funcionario, que já é de grande valia, dando mais consistência ao sistema.

Métodos Abstratos

- Se o método getBonificacao não fosse reescrito, ele seria herdado da classe mãe, fazendo com que devolvesse o salário mais 20%.

- Levando em consideração que cada funcionário em nosso sistema tem uma regra totalmente diferente a fim de ser bonificado, faz algum sentido ter esse método na classe Funcionario? Será que existe uma bonificação padrão para todo tipo de Funcionario? Parece que não, cada classe filha terá um método diferente de bonificação, pois, de acordo com nosso sistema, não existe uma regra gera.

Anhanguera

Métodos Abstratos

- Poderíamos, então, jogar fora esse método da classe Funcionario? O problema é que, se ele não existisse, não poderíamos chamar o método apenas com uma referência a um Funcionario, pois ninguém garante que essa referência aponta para um objeto o qual tem esse método.

- Em Java, existe um recurso no qual, em uma classe abstrata, podemos escrever que determinado método será sempre escrito pelas classes filhas. Isto é, um método abstrato.

Anhanguera

Métodos Abstratos

```
public abstract class Funcionario {

    public abstract double getBonificacao();

    // outros atributos e métodos

}
```


Anhanguera