

Classes Abstratas

Orientação a Objetos - DCC025

Prof. Edmar Welington Oliveira
oliveira.edmar@ufjf.edu.br

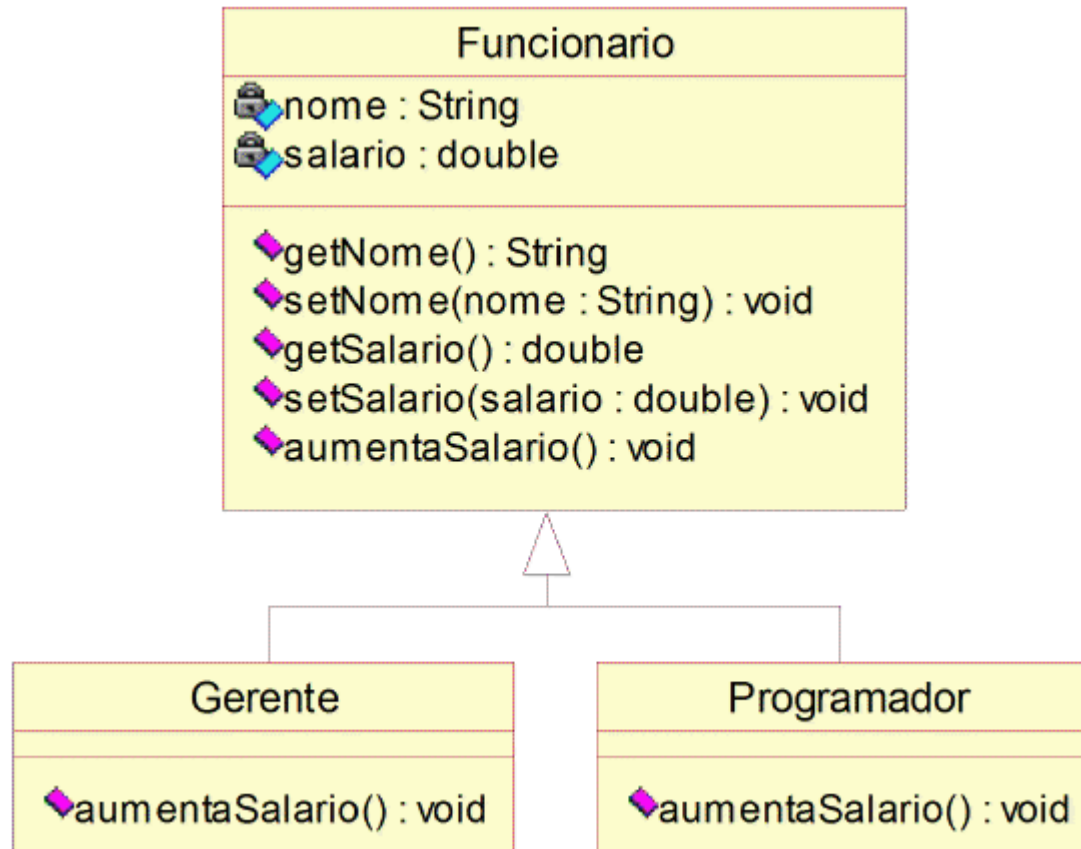
Universidade Federal de Juiz de Fora - UFJF
Departamento de Ciência da Computação - DCC

Classes Abstratas

- Fato

- Em alguns casos, uma superclasse torna-se tão geral que acaba sendo vista como um modelo para outras classes e não como uma classe com instâncias específicas que são usadas.

Classes Abstratas



Classes Abstratas

■ Definição

- Uma classe abstrata é uma classe que **não pode ser instanciada** e possui nenhum, um ou vários **métodos abstratos**. Um método abstrato é um método sem corpo cuja assinatura é precedida da palavra `abstract` e que não possui implementação (corpo vazio)
- Somente são utilizadas como superclasses em hierarquias de herança.
 - Logo, são também chamadas de superclasses abstratas.
- Métodos declarados como abstratos em uma superclasse devem, obrigatoriamente, ser implementados na(s) subclasses(s). Erro de compilação, caso contrário.

Classes Abstratas

- Definição

- **Classes abstratas** são classes que não podem ter instâncias diretas. Contudo, suas subclasses podem ser instâncias. Por outro lado, uma classe **concreta** é uma classe instanciável - pode ter instâncias diretas
- Ex: Se X é uma classe abstrata, não se pode executar o código a seguir:
`X objeto1 = new X();`

Classes Abstratas

Classe1
atributo1
metodo1()



Classe Concreta



Obj1: Classe1
atributo1
metodo1()

Classe2
atributo2
metodo2()

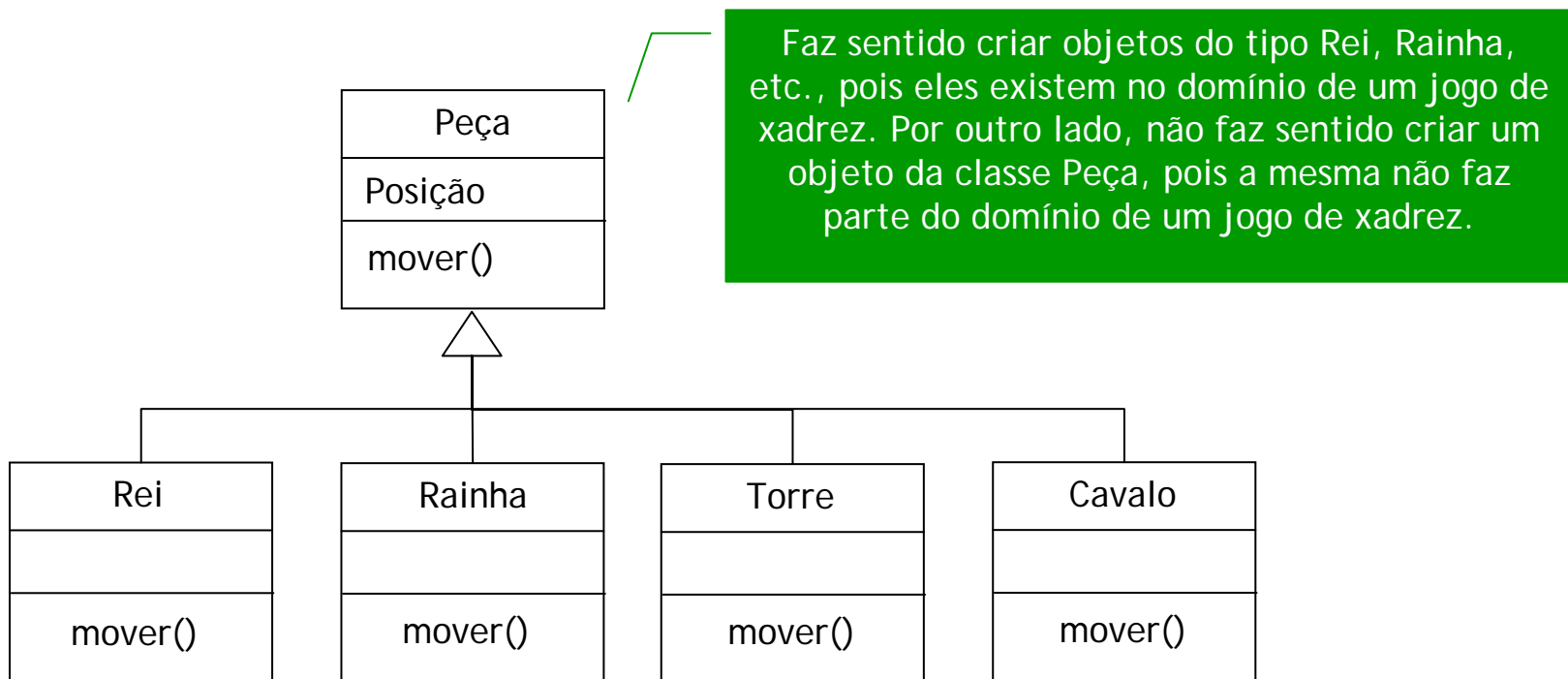


Classe Abstrata



Obj2: Classe2
atributo2
metodo2()

Classes Abstratas



OBS: não há proibição em se criar um objeto da classe Peça. Contudo, como tal criação não faz sentido (ela só serve de molde para as subclasses), insere-se uma "proibição" declarando-a como sendo **abstrata**

Classes Abstratas

■ Entendendo Classes Abstratas

- Suponha que no banco todas as contas possuem um tipo específico, por exemplo, conta poupança, conta corrente ou conta salário e que elas são modeladas dentro do nosso sistema pelas seguintes classes.

```
class Conta {  
    private double saldo;  
  
    // MAIS ATRIBUTOS E MÉTODOS  
}
```

```
class ContaCorrente extends Conta {  
    private double limite;  
  
    // MAIS ATRIBUTOS E MÉTODOS  
}
```

```
class ContaPoupanca extends Conta {  
    private int diaDoAniversario;  
  
    // MAIS ATRIBUTOS E MÉTODOS  
}
```


Classes Abstratas

```
class Conta {  
    private double saldo;  
  
    // MAIS ATRIBUTOS E MÉTODOS  
}
```

A classe Conta não define uma conta de fato - é apenas um molde

```
class ContaPoupanca extends Conta {  
    private int diaDoAniversario;  
  
    // MAIS ATRIBUTOS E MÉTODOS  
}
```

Para cada conta do domínio do banco devemos criar um objeto da classe correspondente ao tipo da conta.

```
class ContaCorrente extends Conta {  
    private double limite;  
  
    // MAIS ATRIBUTOS E MÉTODOS  
}
```

Por exemplo, se existe uma conta poupança no domínio do banco devemos criar um objeto da classe ContaPoupança

Faz sentido criar uma classe ContaPoupança pois existem contas poupanças no domínio do banco.

Classes Abstratas

```
class Conta {  
    private double saldo;  
  
    // MAIS ATRIBUTOS E MÉTODOS  
}
```

Suponha que o banco ofereça extrato detalhado das contas e para cada tipo de conta as informações e o formato desse extrato detalhado são diferentes. Além disso, a qualquer momento o banco pode mudar os dados e o formato do extrato detalhado de um dos tipos de conta.

Não vale a pena ter uma implementação de método na classe Conta para gerar extratos detalhados. Porque?

Classes Abstratas

```
class Conta {  
    private double saldo;  
  
    // MAIS ATRIBUTOS E MÉTODOS  
}
```

Poderíamos, simplesmente, não definir nenhum método para gerar extratos detalhados na classe conta.

Quais seriam os problemas disso?

Classes Abstratas

```
class Conta {  
    private double saldo;  
  
    // MAIS ATRIBUTOS E MÉTODOS  
}
```

■ Métodos Abstratos

- Para garantir que toda classe concreta que deriva direta ou indiretamente da classe CONTA tenha uma implementação de método para gerar extratos detalhados e além disso que uma mesma assinatura de método seja utilizada, devemos utilizar o conceito de métodos abstratos.

Métodos Abstratos

```
class Conta {  
    private double saldo;  
  
    // MAIS ATRIBUTOS E MÉTODOS  
}
```



```
abstract class Conta {  
    private double saldo;  
  
    // GETTERS AND SETTERS  
  
    public abstract void imprimeExtratoDetalhado();  
}
```

Ao definir um método abstrato em uma classe, devemos, obrigatoriamente, declará-la como abstrata

Classes Abstratas

```
class ContaPoupanca extends Conta {  
    private int diaDoAniversario;  
  
    public void imprimeExtratoDetalhado() {  
        System.out.println("EXTRATO DETALHADO DE CONTA POUPANÇA");  
  
        SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy HH:mm:ss");  
        Date agora = new Date();  
  
        System.out.println("DATA: " + sdf.format(agora));  
        System.out.println("SALDO: " + this.getSaldo());  
        System.out.println("ANIVERSÁRIO: " + this.diaDoAniversario);  
    }  
}
```

```
// ESSA CLASSE NÃO COMPILA  
class ContaPoupanca extends Conta {  
    private int diaDoAniversario;  
  
}
```

Toda classe concreta que deriva direta ou indiretamente da classe CONTA será obrigada a ter uma implementação desse método. Caso contrário a classe não compila.



Classes Concretas

```
1 public class Classe1 {  
2  
3     private int atributo1;  
4  
5     public void metodo1() {  
6         }  
7 }
```

Classe Concreta
Classe1

```
1 public class Teste {  
2  
3     public static void main(String args[]) {  
4  
5         Classe1 Obj1 = new Classe1();  
6     }  
}
```

Instanciação da Classe1

Classes Abstratas

```
1 public class Teste {  
2  
3     public static void main(String args[]){  
4  
5         Classe1 Obj1 = new Classe1();  
6  
7     }  
8 }
```

✖ Cannot instantiate the type Classe1

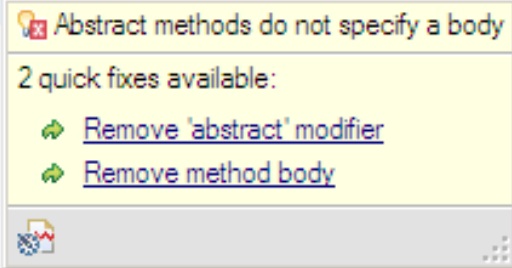
```
1 abstract public class Classe1 {  
2  
3     private int atributo1;  
4  
5     abstract public void metodo1();  
6 }
```

Erro ao tentar instanciar
objeto da classe 1

Classes abstratas não
podem ser instanciadas

Classes Abstratas

```
1 abstract public class Classe1 {  
2  
3     private int atributo1;  
4  
5     abstract public void metodo1() {  
6  
7     }  
8 }  
9  
10
```



Métodos abstratos não podem ter
"corpo" - não importa se o
mesmo estiver vazio

Métodos Abstratos

- Métodos Abstratos

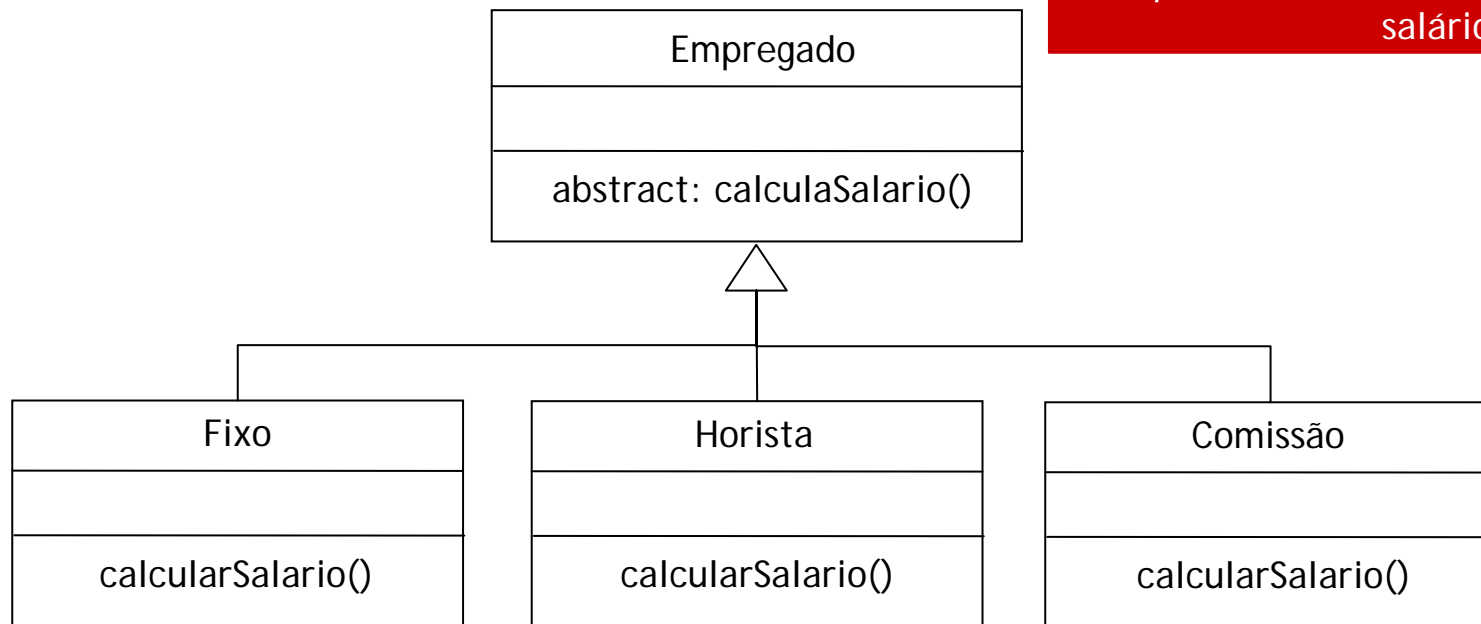
- São apenas declarados (com seu nome, modificadores, tipo de retorno e lista de argumentos), não tendo um corpo que contenha os comandos da linguagem que o método deve executar.
- OBS: se uma classe contém um método declarado como abstrato, as classes que herdarem desta deverão, obrigatoriamente, implementar o método abstrato com o nome, modificador, tipo de retorno e argumentos declarados na classe ancestral.

Métodos Abstratos

- Métodos Abstratos

- OBS: se uma classe tiver algum método abstrato, a classe também deverá, obrigatoriamente, ser declarada com o modificador “abstract”
 - Contudo, uma classe pode ser declarada como abstrata mesmo que nenhum método tenha sido, explicitamente, declarado como abstrato. Ou seja, uma classe pode ter só métodos concretos (exemplos são os métodos get e set).

Classes Abstratas






Ao declarar o método como abstrato, o programador força que todas as subclasses implementem uma forma de calcular o salário

Classes Abstratas

```
1 abstract public class Funcionario {  
2  
3     abstract public int calcularSalario();  
4  
5 }
```



```
1 public class Fixo extends Funcionario{  
2  
3  
4  
5 }  
6
```

 The type Fixo must implement the inherited abstract method Funcionario.calcularSalario()
2 quick fixes available:
 [Add unimplemented methods](#)
 [Make type 'Fixo' abstract](#)

```
1 public class Fixo extends Funcionario{  
2  
3     public int calcularSalario() {  
4         return 0;  
5     }
```

Corpo da classe Fixo não contém o método calcularSalario - Logo, é exibido uma mensagem de erro. A classe Fixo é obrigada a implementar tal método (seu corpo não pode ser vazio)



Classes Abstratas

```
1 abstract public class Funcionario {  
2  
3     abstract public int calcularSalario();  
4  
5 }
```

```
1 public class Fixo extends Funcionario{  
2  
3     public int calcularSalario(){  
4         return 0;  
5     }  
6 }
```

```
1 public class Horista extends Funcionario{  
2  
3     public int calcularSalario(){  
4         return 0;  
5     }  
6 }
```

O método calcularSalario() é processado
polimorficamente

Classes Abstratas

```
1 abstract public class Funcionario {  
2  
3     abstract public int calcularSalario();  
4  
5 }
```

Porque declarar calcularSalario como abstract?

Não faz sentido fornecer uma implementação desse método na classe Funcionario. Não podemos calcular os vencimentos para um Funcionário geral - é preciso conhecer o tipo de empregado para, então, determinar a cálculo apropriado dos vencimentos

Classes Abstratas

- Objetivo
 - O objetivo de criarmos classes abstratas é para encapsular outras classes com comportamento comum. Elas podem surgir naturalmente na modelagem ou serem criadas para promover o reuso.

Exercícios

Exercício 01

- Implemente o que se pede
- Seu objetivo é, considerando os conceitos de OO, projetar um programa para o cálculo de imposto de renda de contribuintes do tipo Pessoa Física (PF), Pessoa Jurídica (PJ) e Pessoa Especial (PE). Toda PF possui CPF e renda bruta. Toda PJ possui CNPJ e renda bruta. Toda PE possui NS (Número Social) e renda bruta. O cálculo do imposto, realizado através do método `calcImposto()`, deve ser feito da seguinte forma: (i) PJ: o imposto corresponde a 10% da renda bruta da empresa, (ii) PF: conforme tabela apresentada em sala e (iii) PE: o imposto corresponde a 10% de renda bruta. Considere uma classe `ReceitaFederal`, que recebe um conjunto de contribuintes e, para cada um, imprime um relatório informando a identificação do mesmo, a renda bruta e a renda descontada dos imposto. Crie os construtores das classes. Crie uma classe **Principal** que teste seu programa (deve ser criado uma pessoa de cada tipo).

Exercício 02

Crie uma classe abstrata chamada CartaoWeb. Essa classe representa todos os tipos de cartões web e conterá apenas um atributo: destinatário (String). Nessa classe você deverá também declarar o método "public abstract void showMessage()". Crie classes filhas da classe CartaoWeb: DiaDosNamorados, Natal, Aniversario. Cada uma dessas classes, deve conter um método construtor que receba o nome do destinatário do cartão. Cada classe também deve implementar o método showMessage(), mostrando uma mensagem ao usuário com seu nome e que seja específica para a data de comemorativa do cartão. Por exemplo, essa poderia ser uma mensagem de um cartão de dia dos namorados:

"Querida Maria,

Feliz Dia dos Namorados!

Espero que esse tenha sido o único cartão do dia dos namorados que tenha ganhado nessa data! ;-) "

*De todo meu coração,
João"*

No método main de uma classe qualquer, crie um array de CartaoWeb. Insira de forma alternada, instâncias dos 3 tipos de cartões nesta array. Após, use um laço for para exibir as mensagens deste cartão chamando o método showMessage(). Em que linha(s) acontece polimorfismo nesse código?

Classes Abstratas

Orientação a Objetos - DCC025

Prof. Edmar Welington Oliveira
oliveira.edmar@ufjf.edu.br

Universidade Federal de Juiz de Fora - UFJF
Departamento de Ciência da Computação - DCC