Classes Abstratas

DC – UFRPE

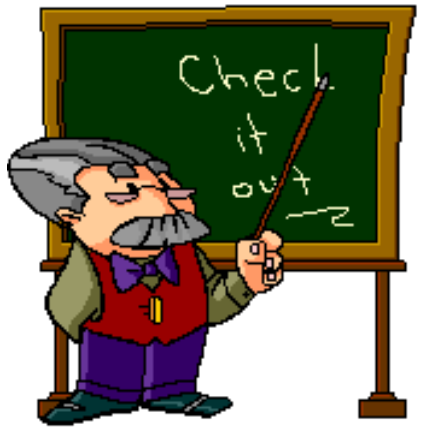
Programação II

Prof. Gustavo Callou

gustavo.callou@ufrpe.br

Roteiro

- Motivar o uso e aprender a definir classes abstratas em Java



Refinando os tipos de empregado

- EmpregadoAssalariado: pagamento é o salário
- EmpregadoAssalariadoComissionado: pagamento é o saláro + a comissão
- EmpregadoHora (novo tipo): recebe por hora trabalhada

Como definir um tipo de empregado independente da forma de pagamento?

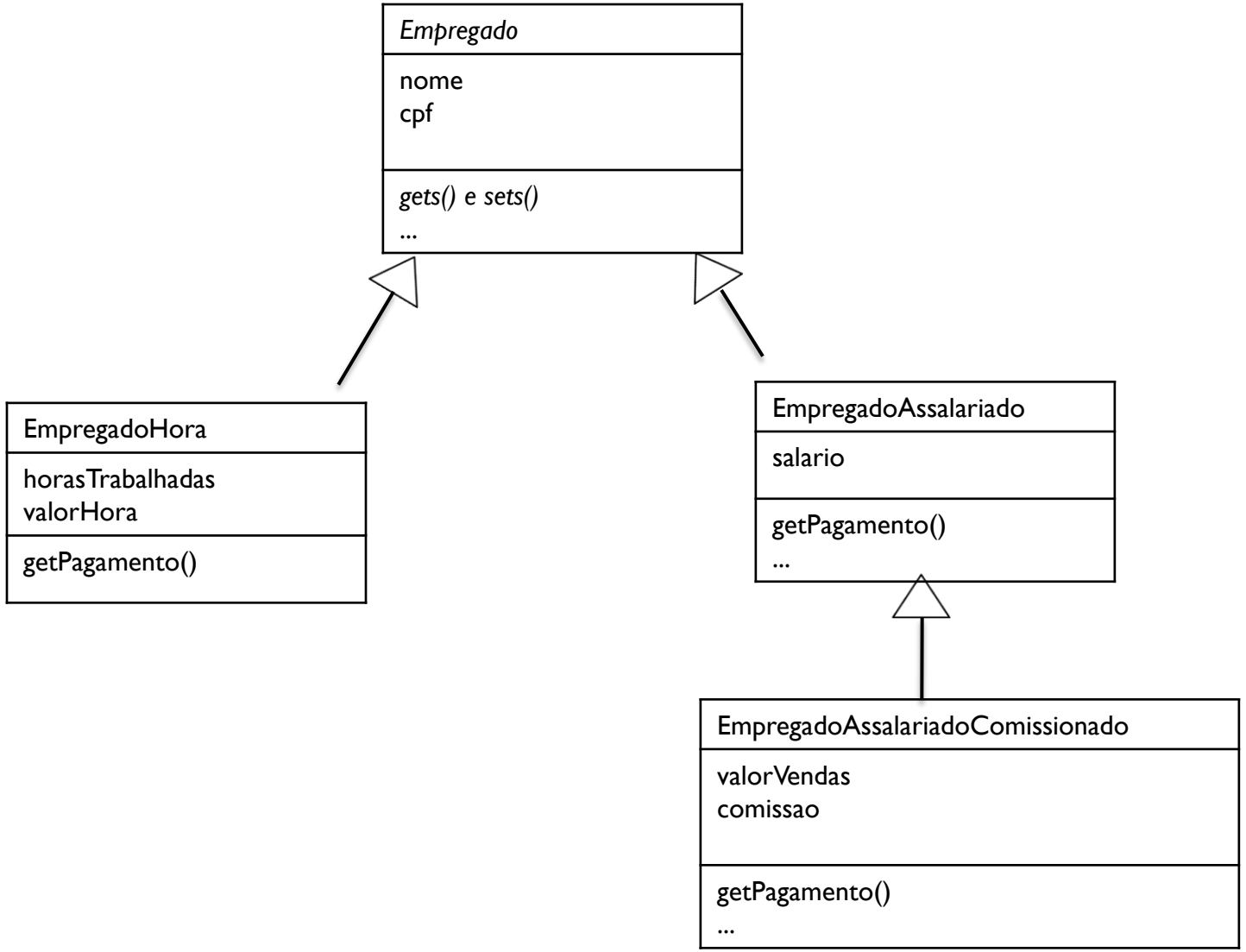
Refinando os tipos de empregado

- EmpregadoAssalariado: pagamento é o salário
- EmpregadoAssalariadoComissionado: pagamento é o salário + a comissão
- EmpregadoHora (novo tipo): recebe por hora trabalhada

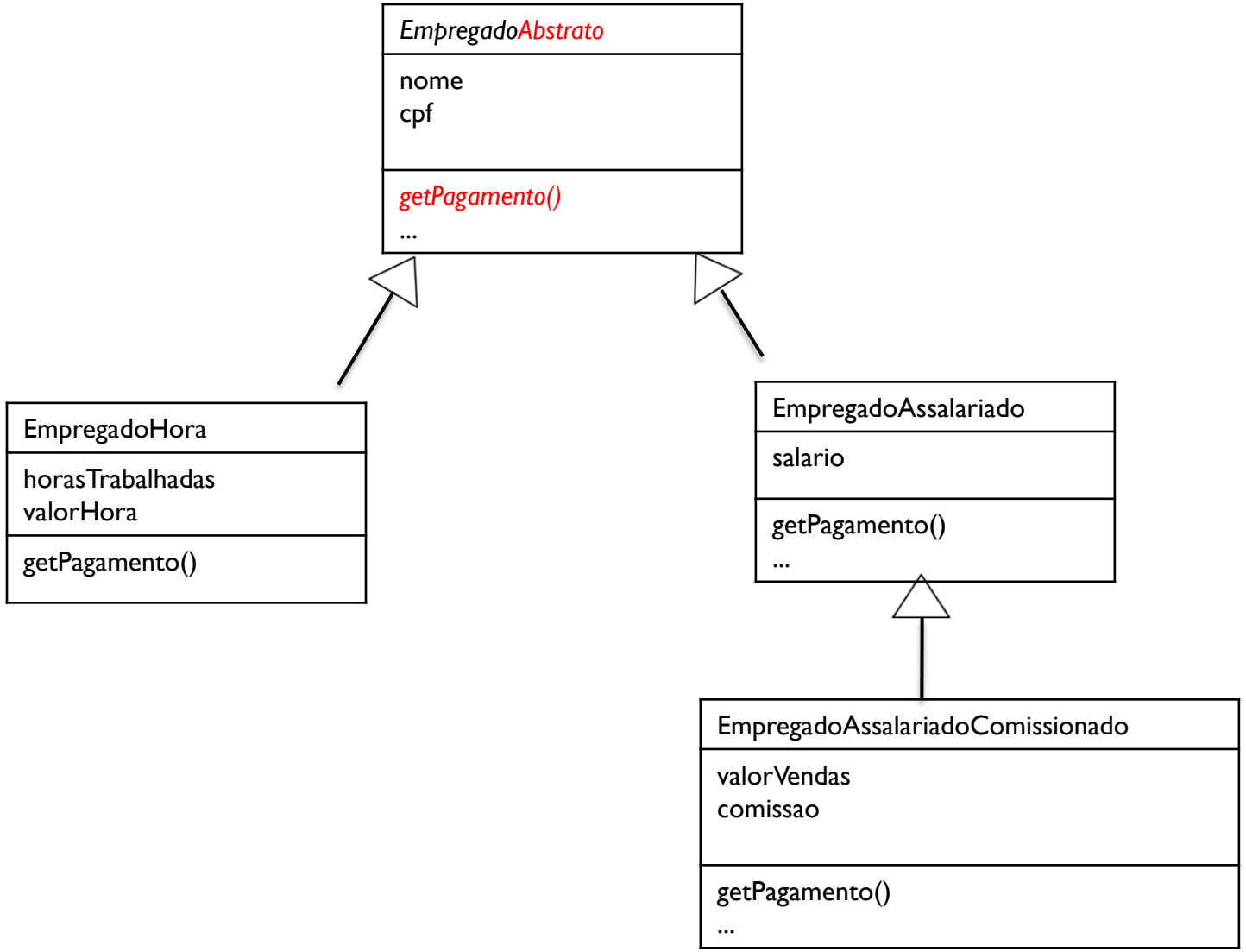
Como definir um tipo de empregado que obrigue a todos empregados a terem uma forma específica de pagamento?

Definindo uma classe empregado abstrata!

Refinando os tipos de empregado



Refinando os tipos de empregado

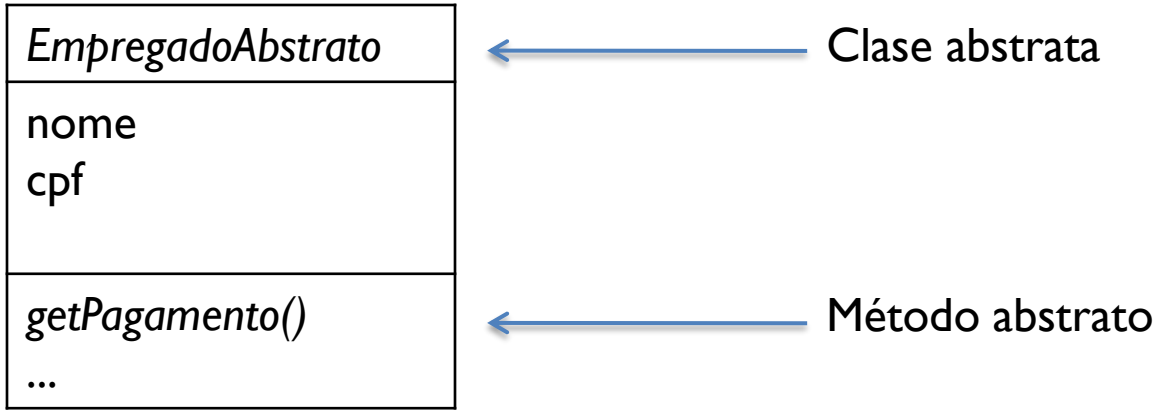


Classe Abstrata

- EmpregadoAbstrato provê o método getPagamento() além dos gets e sets.
- Não faz sentido termos uma implementação de getPagamento() para a classe Empregado, visto que o pagamento depende exclusivamente do tipo do empregado.
- Então cada subclasse que herdar de EmpregadoAbstrato vai ter a implementação de getPagamento() específica para ele.
- Então a classe EmpregadoAbstrato vai servir como uma "interface" para todos os empregados que herdarem dela. Ela obriga que as subclasses implementem getPagamento ou também serão abstratas.

Método e Classe Abstrata

- Método é abstrato (**abstract**) quando não possui corpo
- Classe com método abstrato é abstrata e não pode ser instanciada
- Exemplo:

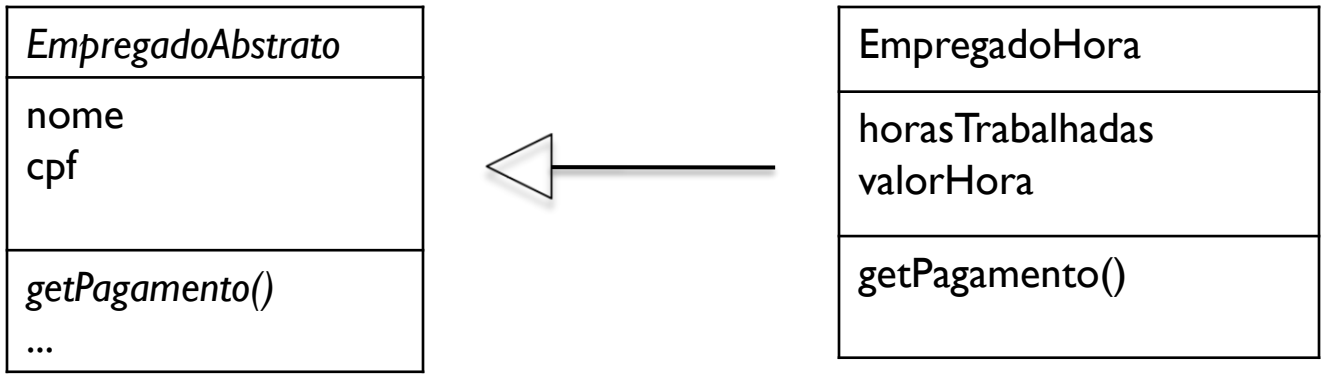


EmpregadoAbstrato

```
public abstract class EmpregadoAbstrato {
    protected String nome, cpf;
    public Empregado(String nome, String cpf) {
                                this.nome = nome;
                                this.cpf = cpf;
    }
    public abstract double getPagamento();
     //métodos gets, sets, equals e toString
}
```

De abstrato para concreto

- Método abstrato:
  - sobrescrito: vira concreto
  - não sobrescrito: continua abstrato
- Subclasse que implementa os métodos abstratos torna-se concreta
- Exemplo:



EmpregadoHora – Classe concreta

```
public class EmpregadoHora extends EmpregadoAbstrato {
  protected double valorHora;
  protected int horasTrabalhadas;
  //código do construtor
  @Override
  public double getPagamento() {
    return valorHora * horasTrabalhadas;
  }
   // métodos get, set e toString
}
```

EmpregadoAssalariado – Classe concreta

```
public class EmpregadoAssalariado extends EmpregadoAbstrato {
  protected double salario;
  // código do construtor
  @Override
  public double getPagamento() {
    return salario;
  }
  //métodos get, set, toString
}
```

EmpregadoAssalariado – Classe concreta

```
public class EmpregadoAssalariadoComissionado extends EmpregadoAssalariado {
  protected double salario;
  // código do construtor
  @Override
  public double getPagamento() {
    return super.salario + valorComissao();
  }
  //métodos get, set, toString
}
```

Exercício 1

- Observe a implementação da classe abstrata Empregado e seus subtipos
- Crie uma classe chamada TesteClasseAbstrata com um método main. No método main
  - Crie objetos para todos os tipos de empregados
  - Armazene estes objetos num array
  - Percorra o array chamando o método toString utilizando polimorfismo