Interfaces

DC – UFRPE

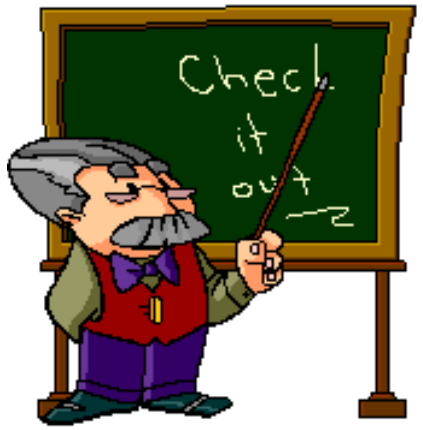
Programação II

Prof. Gustavo Callou

gustavo.callou@ufrpe.br

Roteiro

- Interfaces em Java



Interface

- Conjunto de métodos para um tipo
- Padroniza o comportamento (métodos) para as diferentes implementações de um tipo
  - Implementações podem ter estruturas diferentes (atributos)
- Exemplo: Interface da Classe Carro independe do carro
  - acelerar()
  - freiar()
  - virarDireita()
  - virarEsquerda()

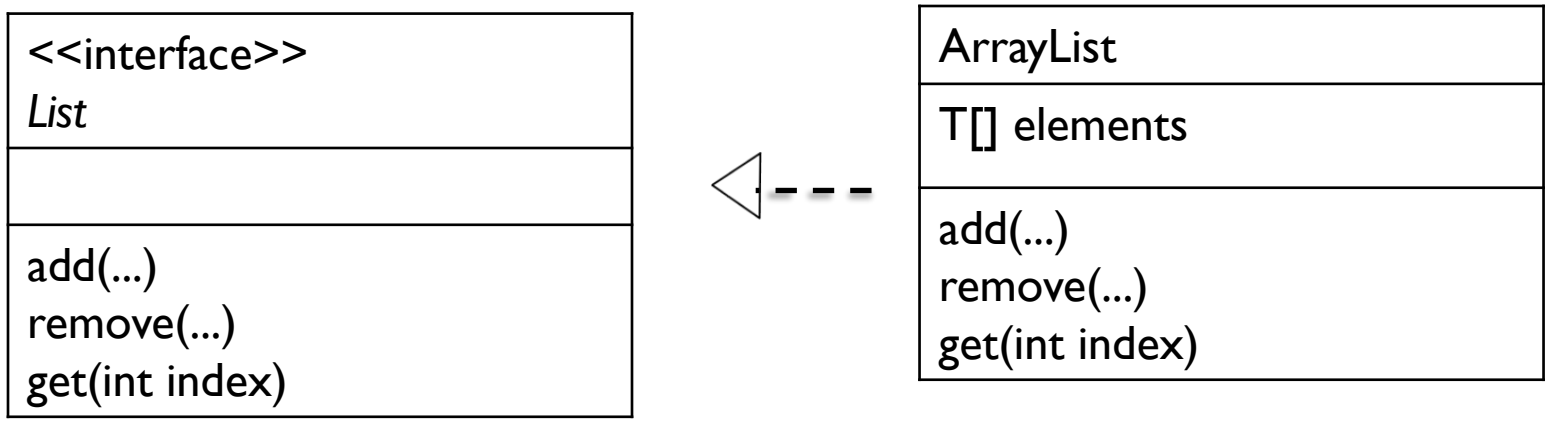
Interface

<<interface>> *ControleRemoto*
static int volume maximo = 10; static int volume minimo = 0;
aumentarVolume() diminuirVolume()

<<interface>> *TelefoneCelular*
ligar() desligar() discar()

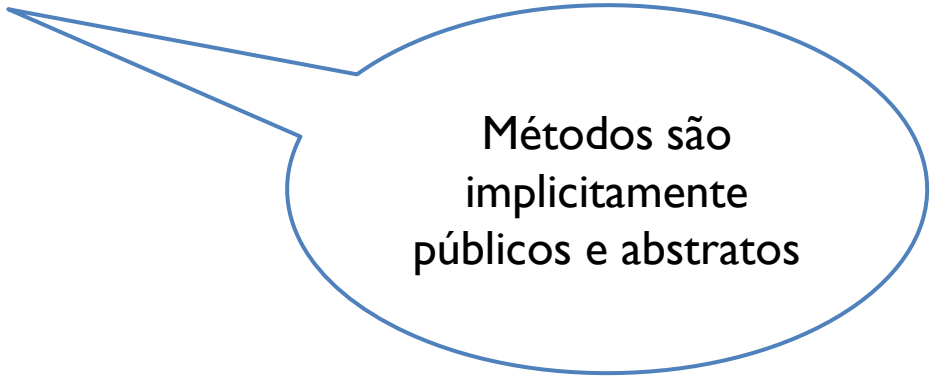
Interface em Java

- Classe 100% abstrata (todos os métodos são abstratos)
  - Não tem atributos
  - Pode ter constantes
- Uma interface é implementada quando outro tipo define corpo para seus métodos
- Exemplo: ArrayList implementa a interface List



Interface em Java

```
public interface ControleRemoto {
    static final int maximo = 10;
    static final int minimo = 0;
   void aumentar();
   void diminuir();
}
```



Implementando interface

```
public class ControleLimitaVolume implements ControleRemoto {
    private int volume = 0;
    void aumentar() {
        if(volume < ControleRemoto.maximo) { volume++; }
    }
    void diminuir() {
        if(volume > ControleRemoto.minimo) { volume--; }
    }
}
```

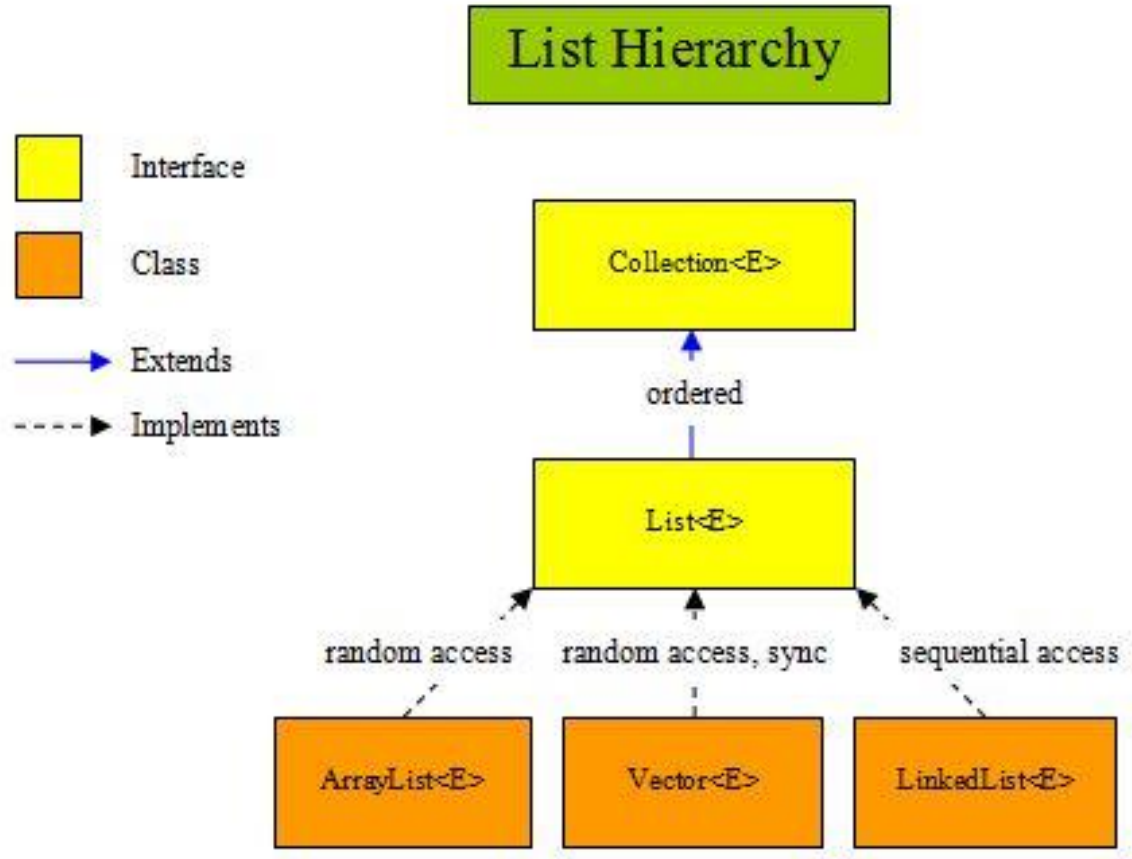
Implementando interface

- Uma classe pode implementar qualquer quantidade de interfaces
  - Simula herança múltipla

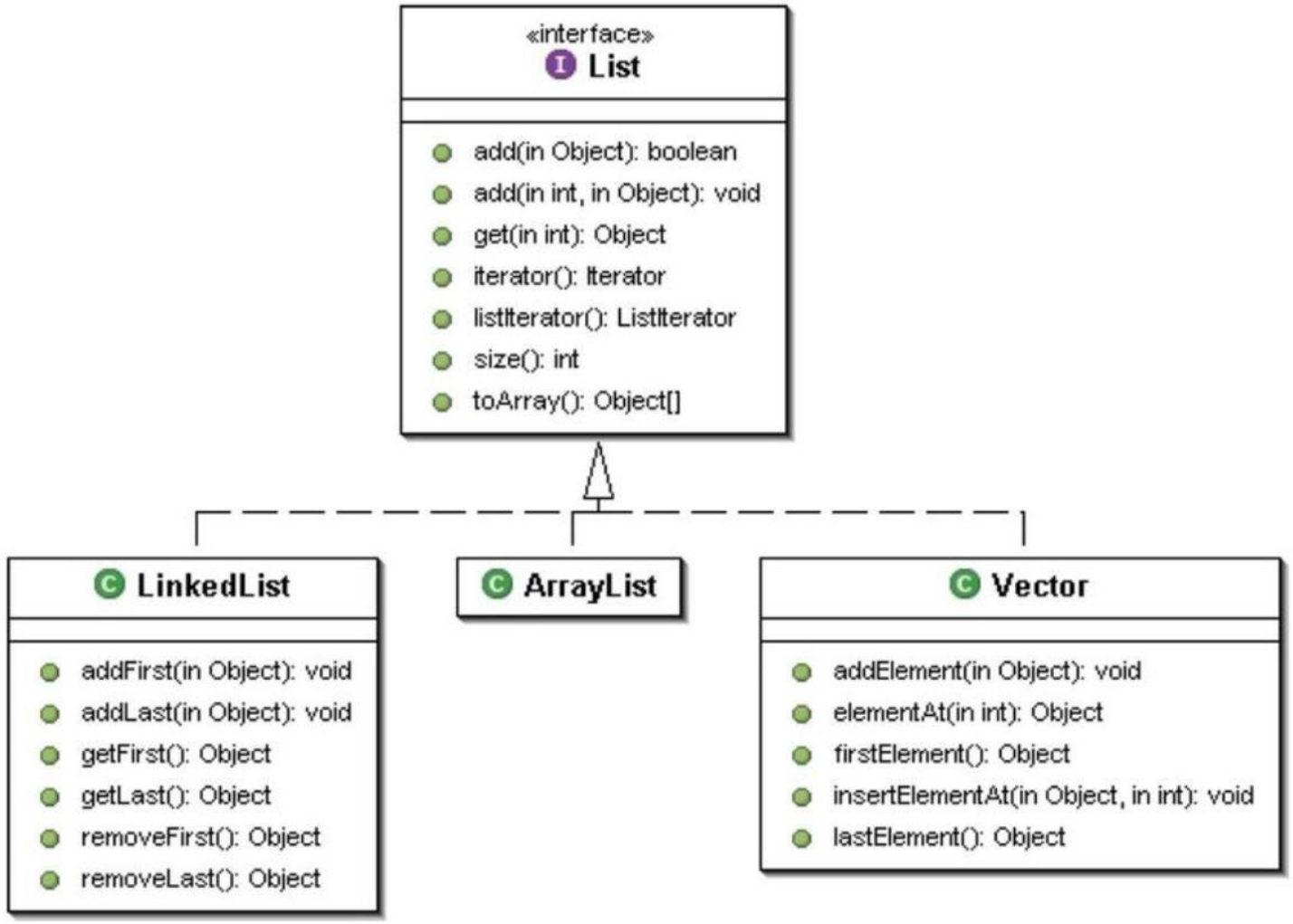
```
public class X extends Y implements A, ..., Z {
                    ...
}
```

Implementando interface

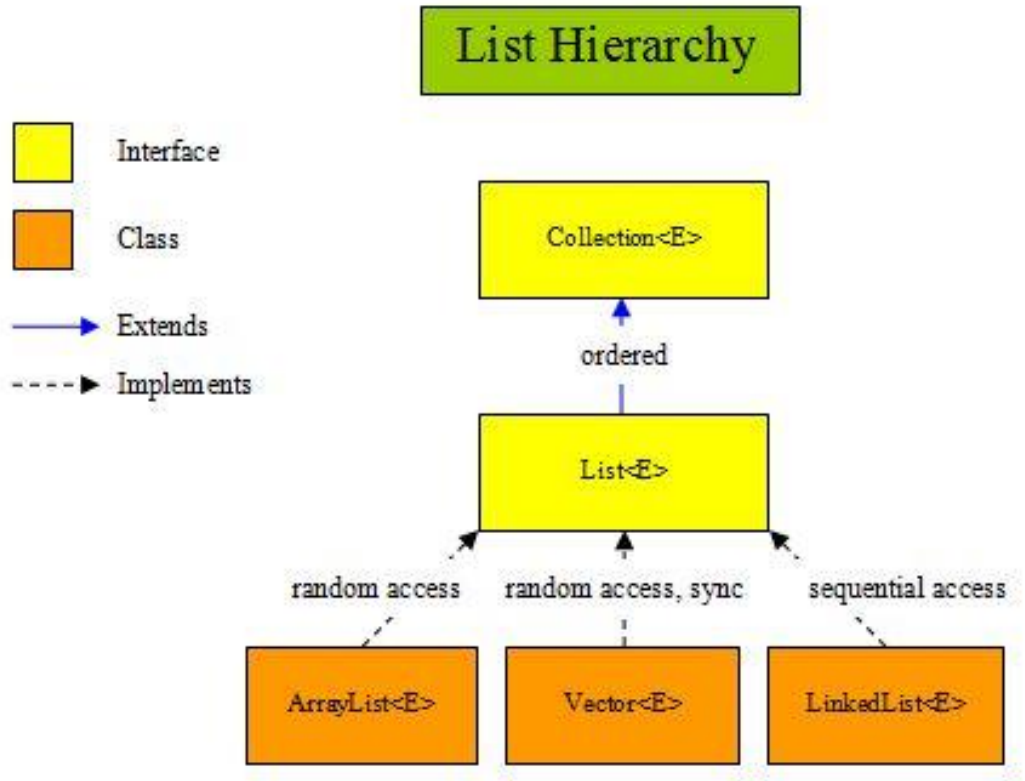
- Exemplo:

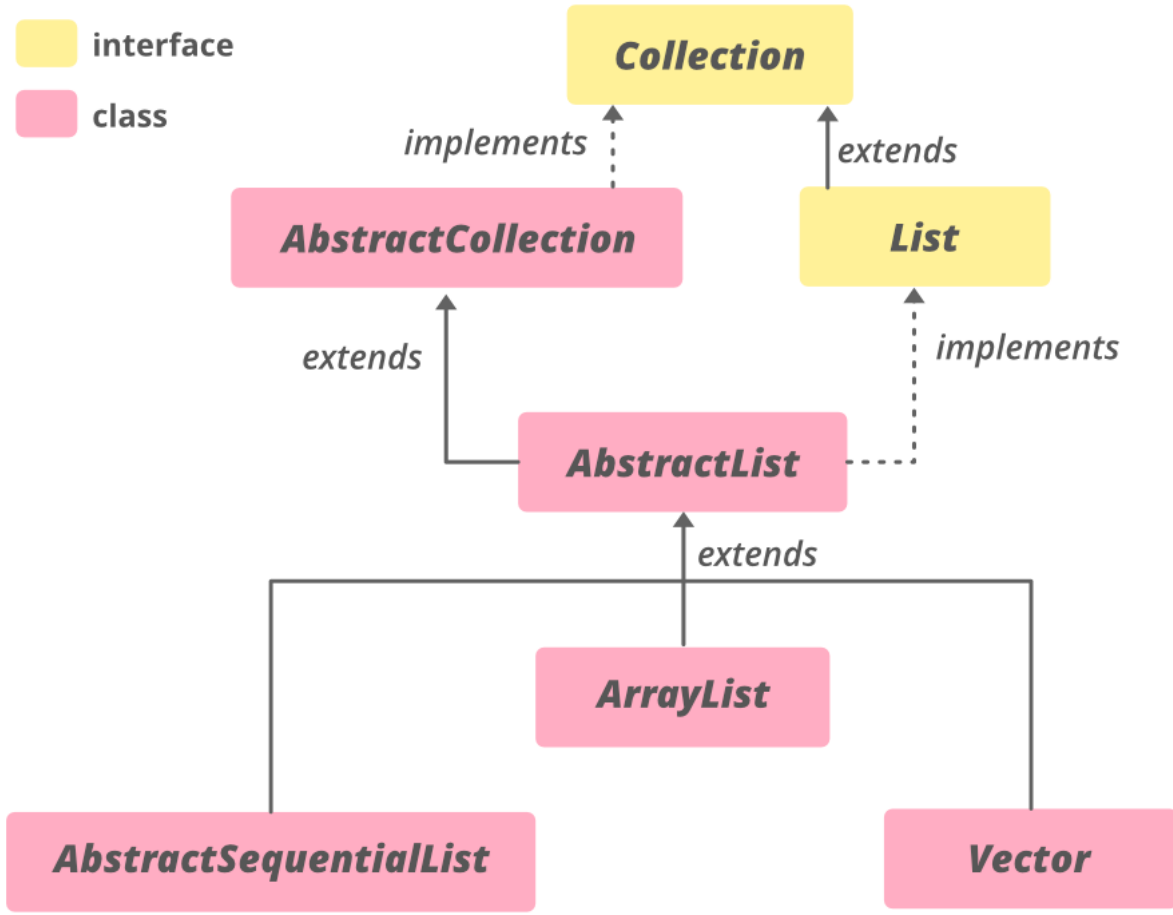


Implementando interface



Implementando interface





Implementando interface

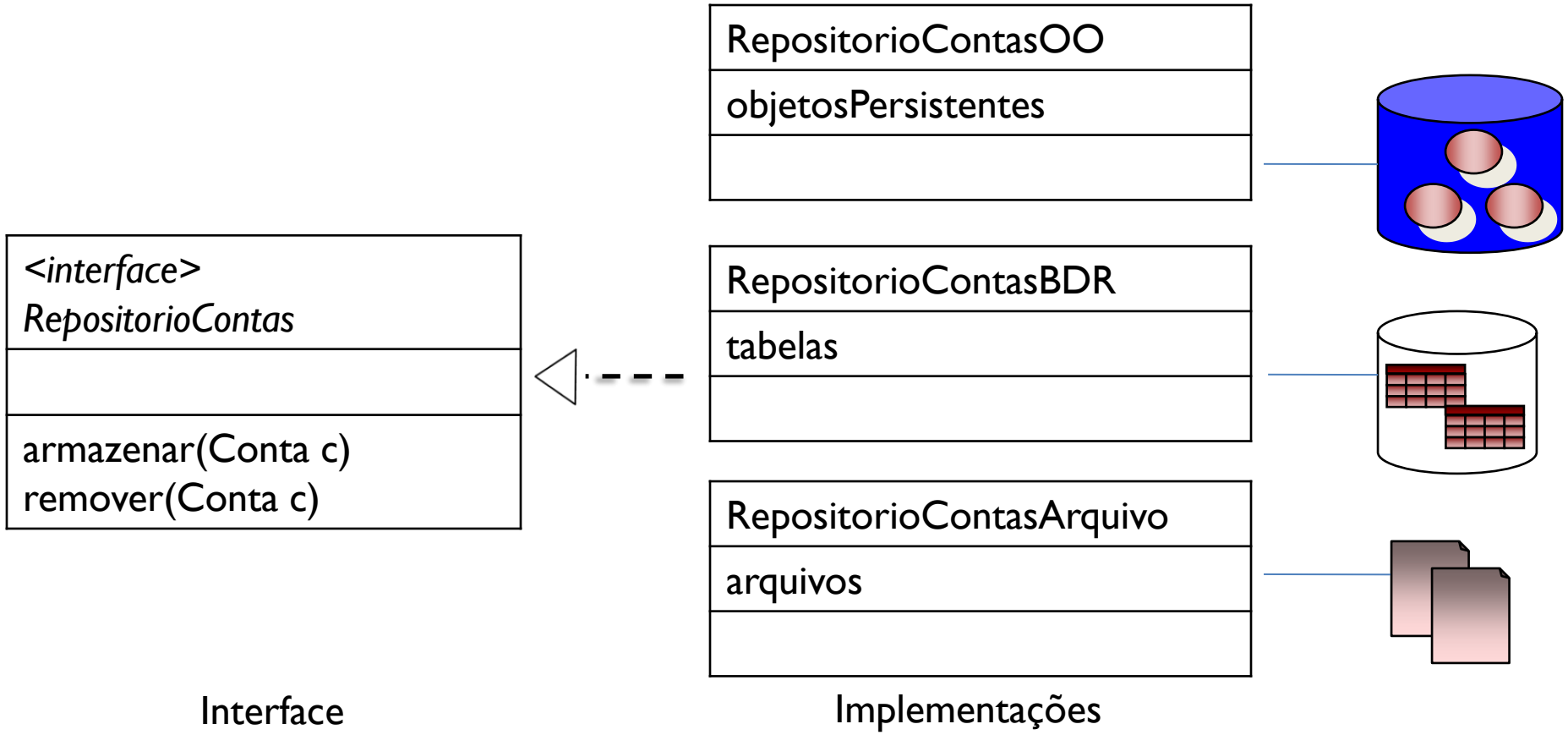
- Uma classe que implementa uma interface pode ser concreta ou abstrata
  - Concreta se implementa todos os métodos da interface (ex: ControlePadrão)
  - Abstrata se implementa parcialmente os métodos da interface (ex: AbstractList)

Conceito de OO: encapsulamento

- Separar a interface de um componente de sua implementação
  - Interface é imutável e define a forma de interagir com o componente
  - Implementação forma particular de comportamento para um método
- Programar orientado a interfaces aumenta modularização
- Exemplo:
  - Sabendo usar java.util.ArrayList sabemos utilizar qualquer outro tipo que implementa java.util.List

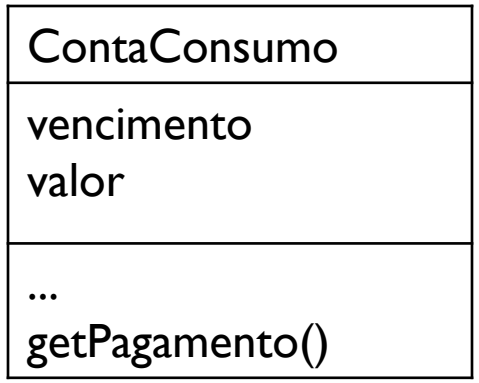
Exemplo: interface para repositório

- Uma interface de repositório pode possuir diferentes implementações



Exemplo: Classe ContaConsumo

- Representa uma conta de luz, água, etc

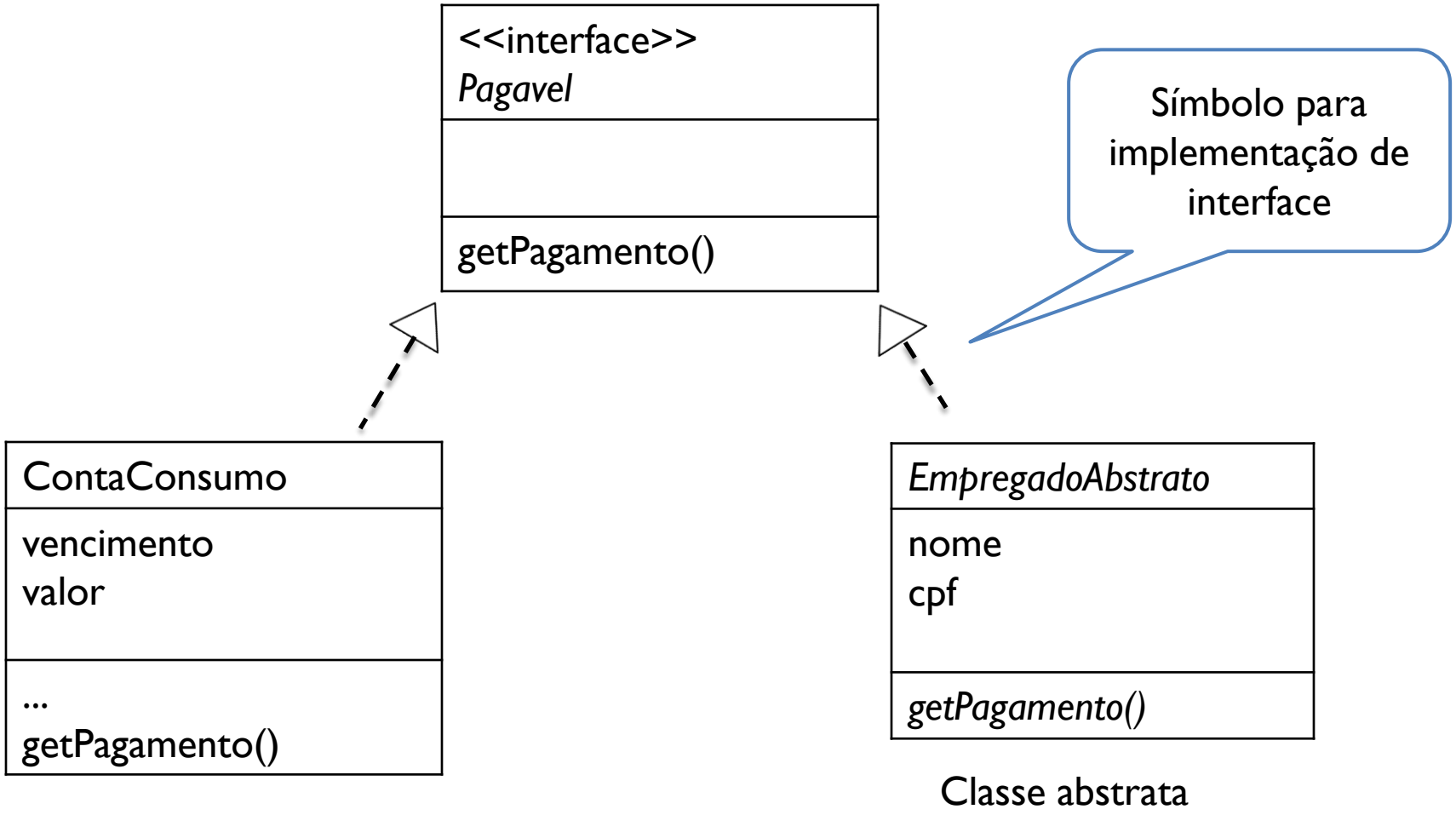


- Pergunta: como fazer um código único para calcular o valor de pagamento de contas de consumo como de empregados já que eles tem uma interface em comum
  - Método getPagamento()

Interfaces

- Interfaces podem ser usadas para criar tipos bastante gerais
- Exemplo: Interface Pagavel
  - Coisas que são pagáveis implementam o método **double** getPagamento()
  - Portanto, tanto uma contas e empregados são pagáveis

Interfaces


<<interface>>
Pagavel
getPagamento()
Símbolo para implementação de interface
ContaConsumo
vencimento
valor
...
getPagamento()
EmpregadoAbstrato
nome
cpf
getPagamento()
Classe abstrata
Classe concreta

Definindo interface Pagavel

```
public interface Pagavel {
    double getPagamento();
}
```

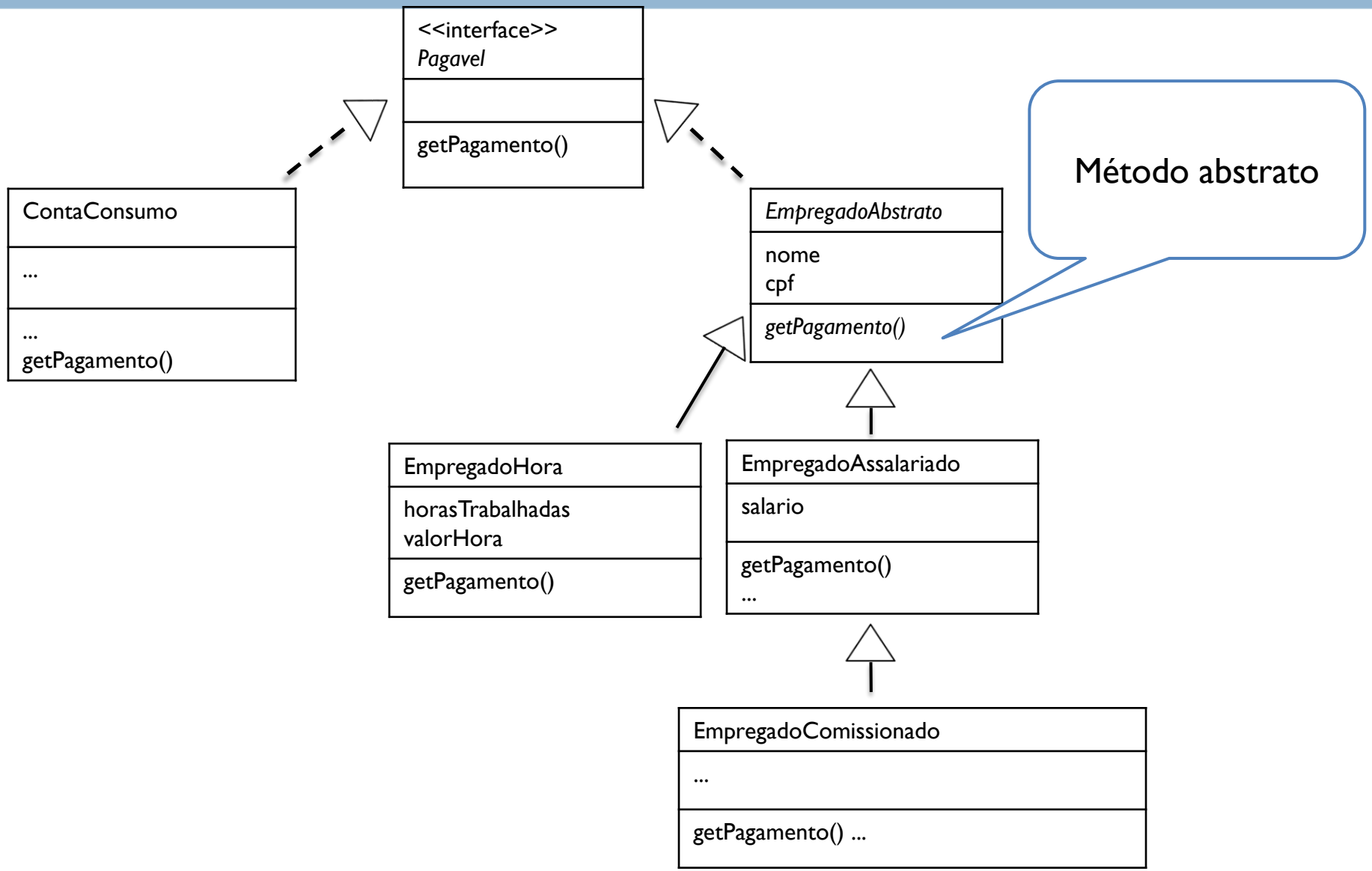
Implementando Interface Pagavel

```
import java.util.Date;
public class ContaConsumo implements Pagavel {
    private Date vencimento;
    private double valor;
     // código do construtor e outros métodos get/set
     public double getPagamento(){
                    return getValor();
          }
}
```

Implementando Interface Pagavel

```
public abstract class EmpregadoAbstrato implements Pagavel {
        protected String nome, cpf;
        public abstract double getPagamento();
        // método construtor e métodos get/set
}
```

Hierarquia para o Tipo Pagavel



Interfaces (exemplo de uso)

```
List<Pagavel> pagamentos = new ArrayList();

pagamentos.add(new EmpregadoHora(...));
pagamentos.add(new ContaConsumo(new Date(03,40,2012), 150.00));
double total = 0;
for(Pagavel pg : pagamentos) {
  total = total + pg.getPagamento();
}
```

Classes abstratas vs Interfaces

- Ambas não podem ser instanciadas
- Classes abstrata:
  - Atributos e métodos (ao menos um abstrato)
  - Subclasse **estende**
- Interface:
  - Todos os métodos são abstratos
  - Subclasse **implementa**

Algumas interfaces da API de Java

- java.lang.Comparable: método que retorna a ordem de um objeto (menor, igual ou maior) com relação a outro do mesmo tipo
  - Necessário implementar para utilizar algoritmos de ordenação da API de Java

- Java.lang.Runnable: métodos que inicia um fluxo de execução independente (thread) para o objeto
  - Necessário para criar Threads