Interfaces

Interfaces

- Interfaces no modelo de OO
- Interfaces em UML
- Uma interface implementada por diversas classes
- Uma classe implementando diversas interfaces
- Herança de interfaces
- Hierarquia de interfaces
- Exemplos de interfaces em Java

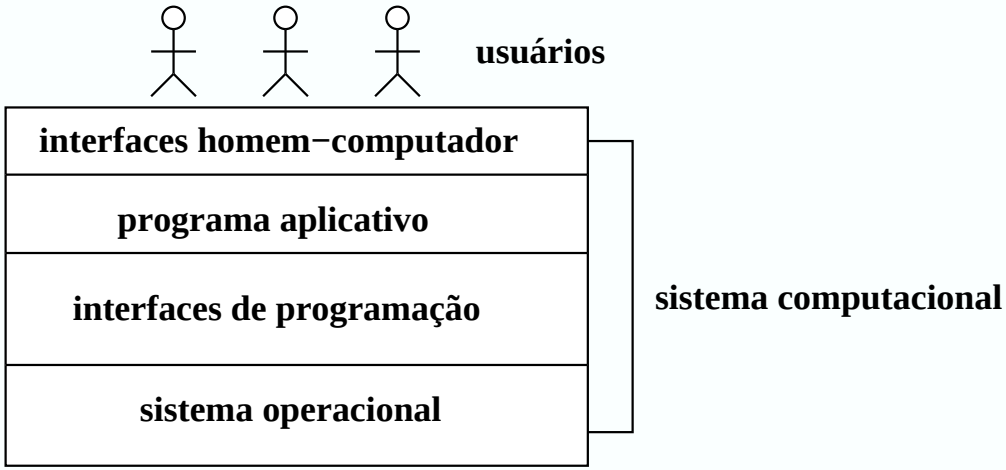
Definição de Interface

- Em geral, o conceito de **interface** define uma superfície que separa duas fases de um mesmo sistema
- Uma fase é definida como sendo uma parte homogênea pertencente a um sistema heterogêneo

Exemplos de Interfaces em Sistemas Computacionais (I)

- Interface humano-computador: camada de um sistema, responsável pela interação entre o usuário e os processos da aplicação
- Interface de programação ou API ("Application Programming Interface"): camada composta por um conjunto de funções que podem ser chamadas pelos programas aplicativos para usar os serviços fornecidos pelo sistema operacional

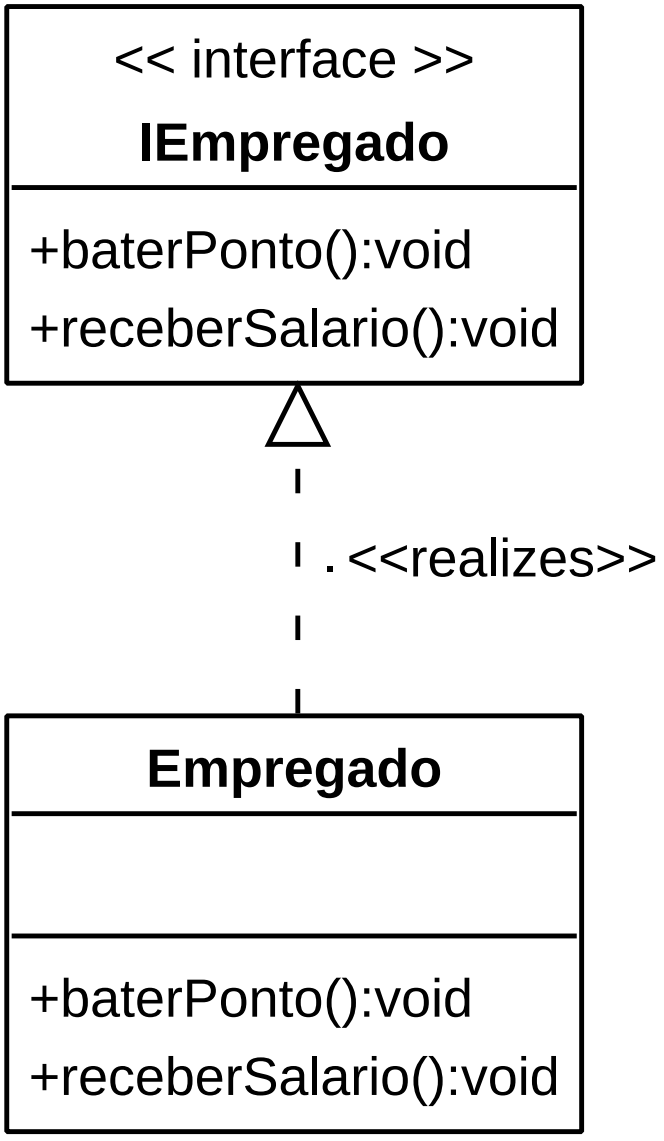
Exemplos de Interfaces em Sistemas Computacionais (II)



Interfaces no Modelo de Objetos

- Em OO, uma interface é usada para definir um tipo que descreve o comportamento externamente visível de uma classe, objeto ou uma outra entidade
- **Definição de Interface**: uma interface Java especifica um tipo abstrato de dados (TAD) através da definição de um conjunto de operações e suas respectivas assinaturas de métodos
- A implementação de um TAD especificado por uma interface Java é feita através de uma classe
- Uma classe pode implementar várias interfaces
- Uma interface pode ser implementada por diversas classes

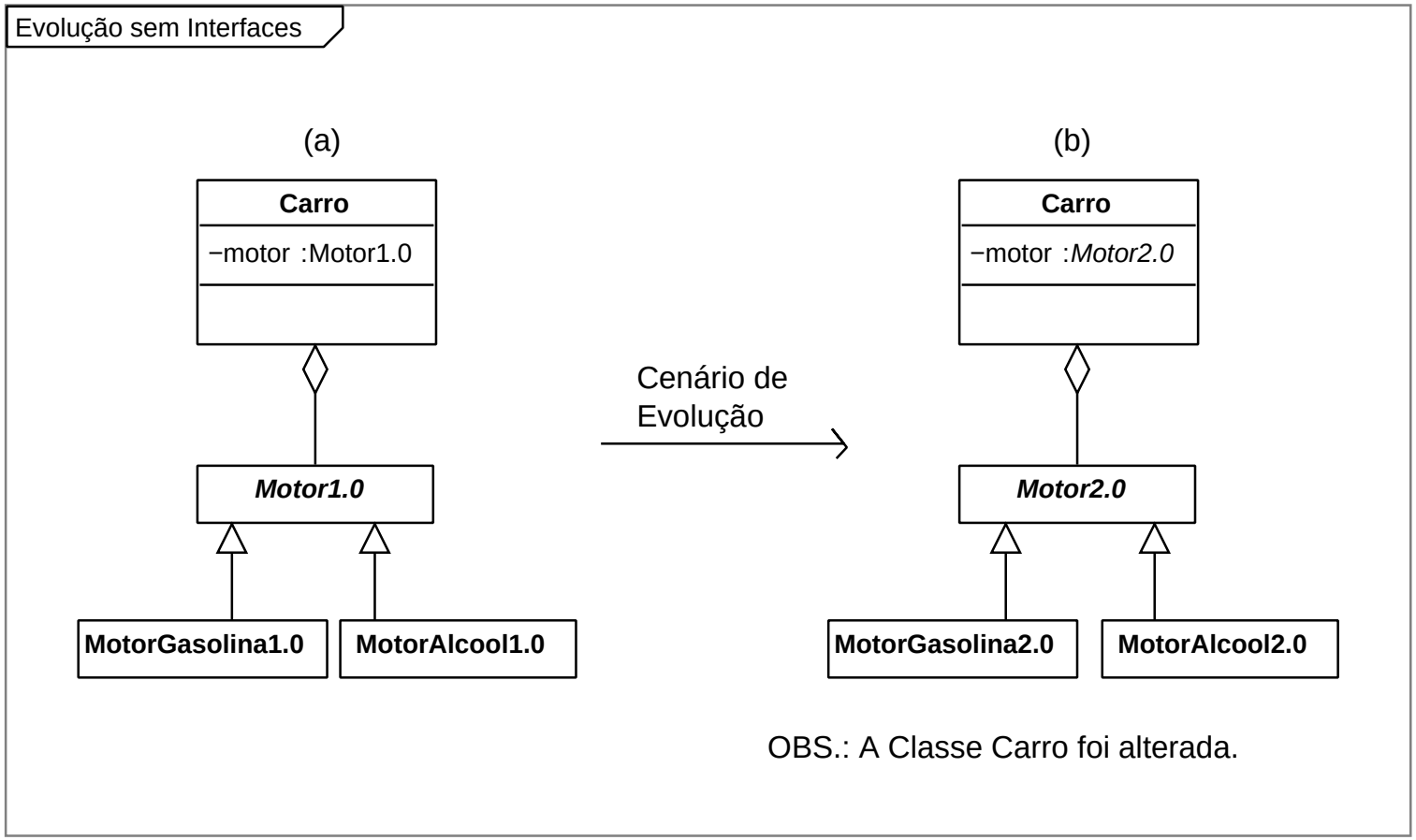
Exemplo de Interface


<< interface >>
IEmpregado
+baterPonto():void
+receberSalario():void
. <<realizes>>
Empregado
+baterPonto():void
+receberSalario():void

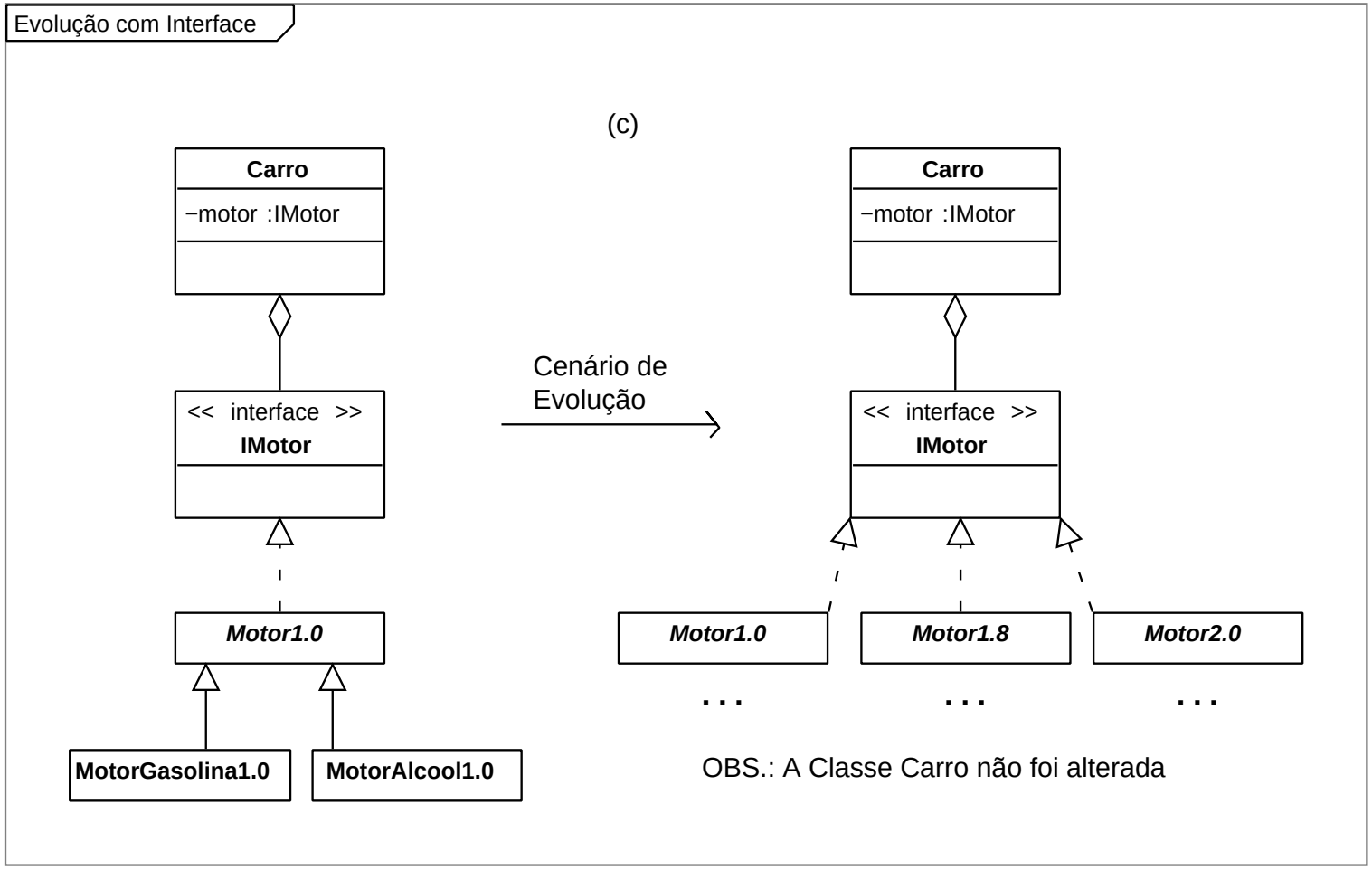
Interfaces em UML (I)

- A tarefa de uma interface Java é separar de forma explícita, diferentes grupos de classes de uma aplicação; isto é, estabelecer fronteiras explícitas entre partes homogêneas de um sistema computacional heterogêneo

Interfaces em UML (II)



Interfaces em UML (III)



Interfaces em UML (IV)

- No caso (a), a classe Carro contém uma referência direta para um objeto do tipo Motor1.0. Uma dependência entre Carro e Motor1.0 é estabelecida.
- No caso (b), a classe Motor1.0 foi substituída pela classe Motor2.0. Devido à dependência direta entre as classes, a classe Carro também precisa ser atualizada.

Interfaces em UML (V)

- No caso (c), as classes Carro e Motor1.0 se acoplam através de uma interface IMotor. Carro contém uma referência para um objeto de tipo IMotor, e as classes Motor1.0, Motor1.8 e Motor2.0 implementam essa mesma interface IMotor.
- A dependência entre Carro e Motor1.0 é reduzida. Carro é capaz de operar com objetos do tipo IMotor que podem ser instanciados a partir das classes MotorGasolina1.0, MotorAlcool1.8 ou MotorGasolina2.0, por exemplo.

Interfaces e Realizações

- Uma interface Java contém apenas assinaturas de métodos e definições de constantes
- Uma característica essencial é a ausência de implementação, tanto de métodos, quanto de estrutura de dados. Portanto, uma interface é abstrata.
- Todas as operações de uma interface são sempre **implicitamente públicas** e **abstratas**
- Uma classe(concreta ou abstrata), em contraste com uma interface, pode ter definições de atributos, de métodos concretos e de métodos abstratos

Interface X Classe Abstrata

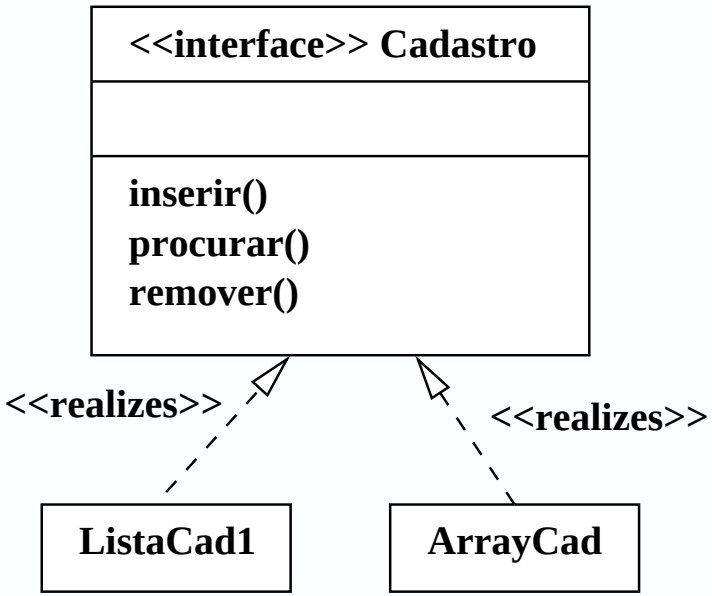
Classe Abstrata:

- não pode ser instanciada
- pode ter atributos e constantes
- pode ter métodos abstratos (ou não)
- pode ter métodos concretos (ou não)

Interface:

- não pode ser instanciada
- só possui constantes
- só possui métodos abstratos

Interface Impl. por Diversas Classes (I)

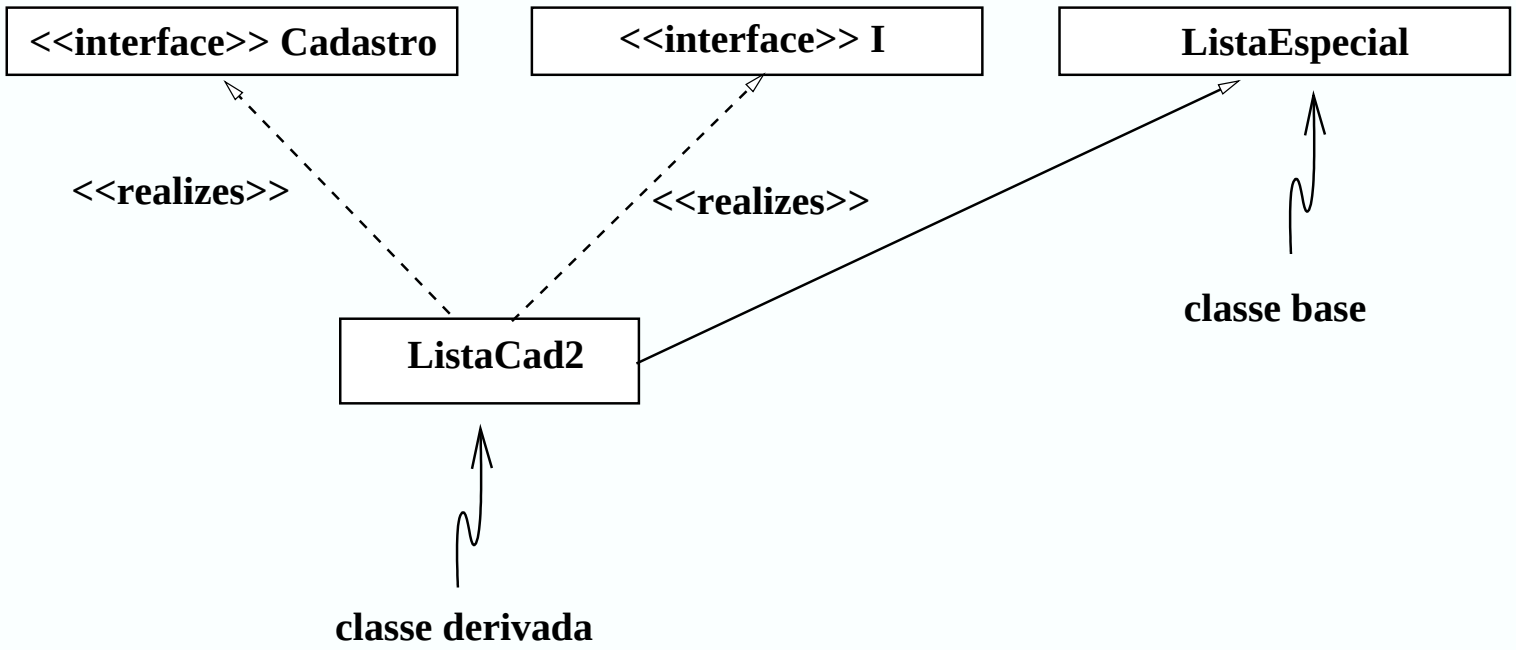


Interface Impl. por Diversas Classes (II)

```
interface Cadastro{...}
class ListaCad1 implements Cadastro{...}
class ArrayCad implements Cadastro{...}
```

- Caso uma classe não implemente alguma operação de uma de suas interfaces, o método torna-se implicitamente abstrato na classe, tornando a classe também implicitamente abstrata

Uma Classe Impl. Diversas Interfaces (I)

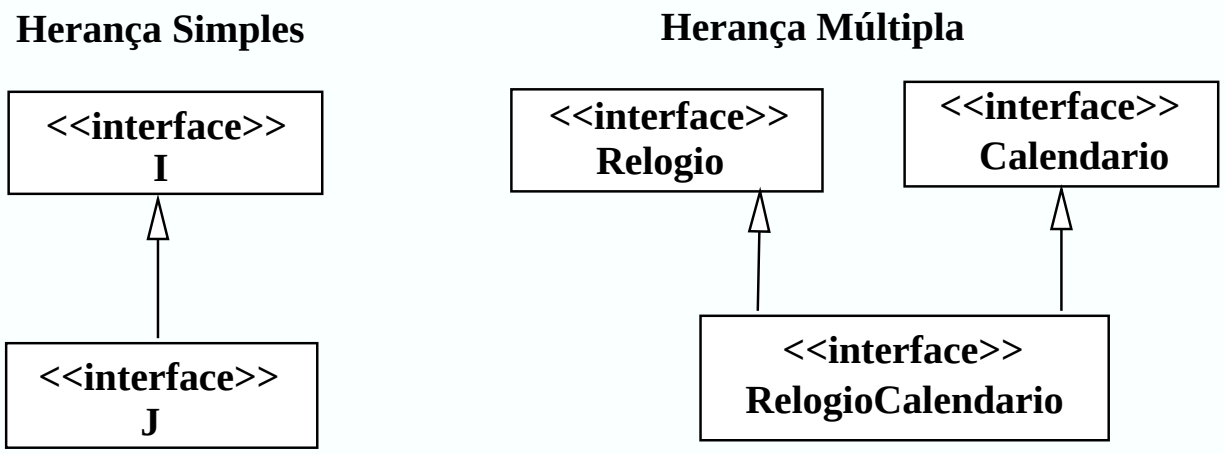


Uma Classe Impl. Diversas Interfaces (II)

```
interface Cadastro{...}
interface I{...}
class ListaCad2 implements Cadastro, I extends
ListaEspecial{...}
```

- Em Java, uma classe pode herdar **apenas de uma classe base** (herança simples), mas ela pode implementar várias interfaces.

Herança Simples e Múltipla de Interfaces (I)

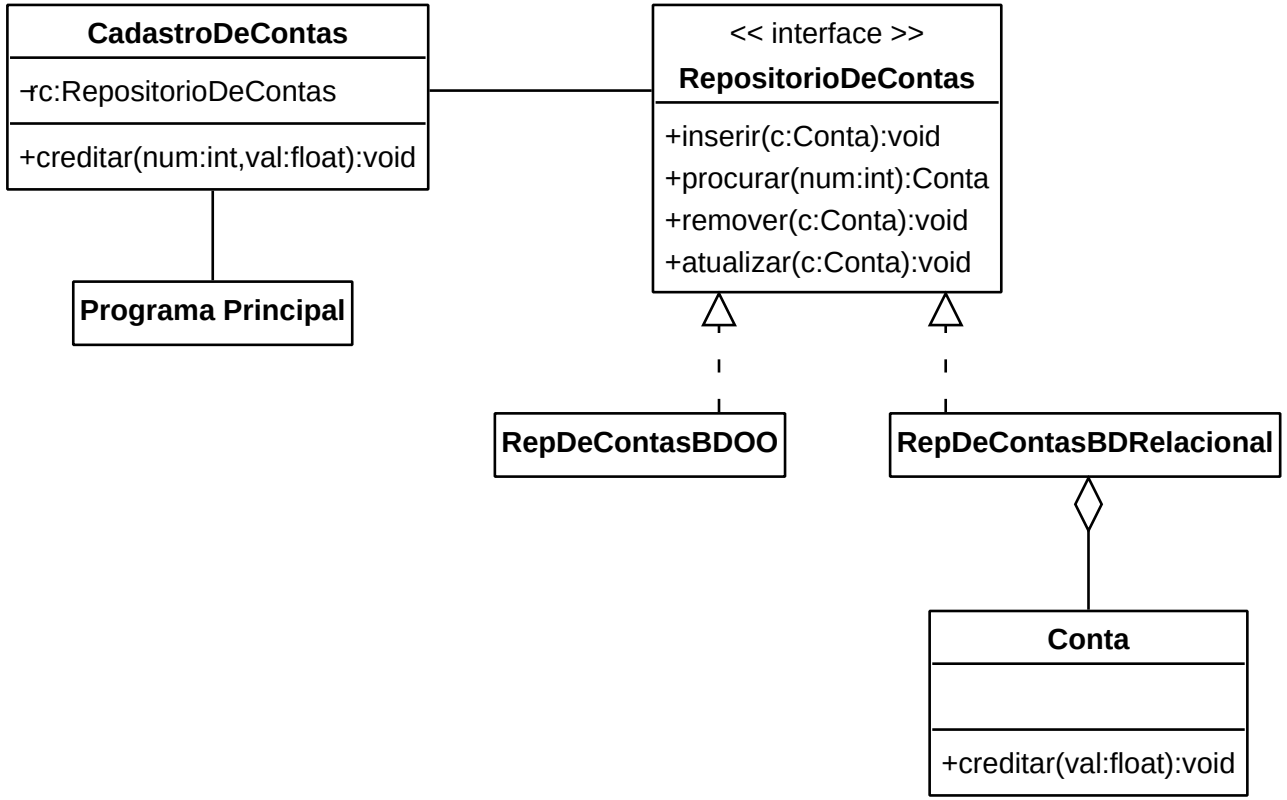


Her. Simples e Múltipla de Interfaces (II)

```
interface I{...}
interface J extends I{...}  // herança simples
                            // de interfaces
interface Relogio{...}
interface Calendario{...}
interface RelogioCalendario
                  extends Relogio, Calendario{...}
                // herança múltipla de interfaces
```

Exemplo: Integração Objeto-Relacional

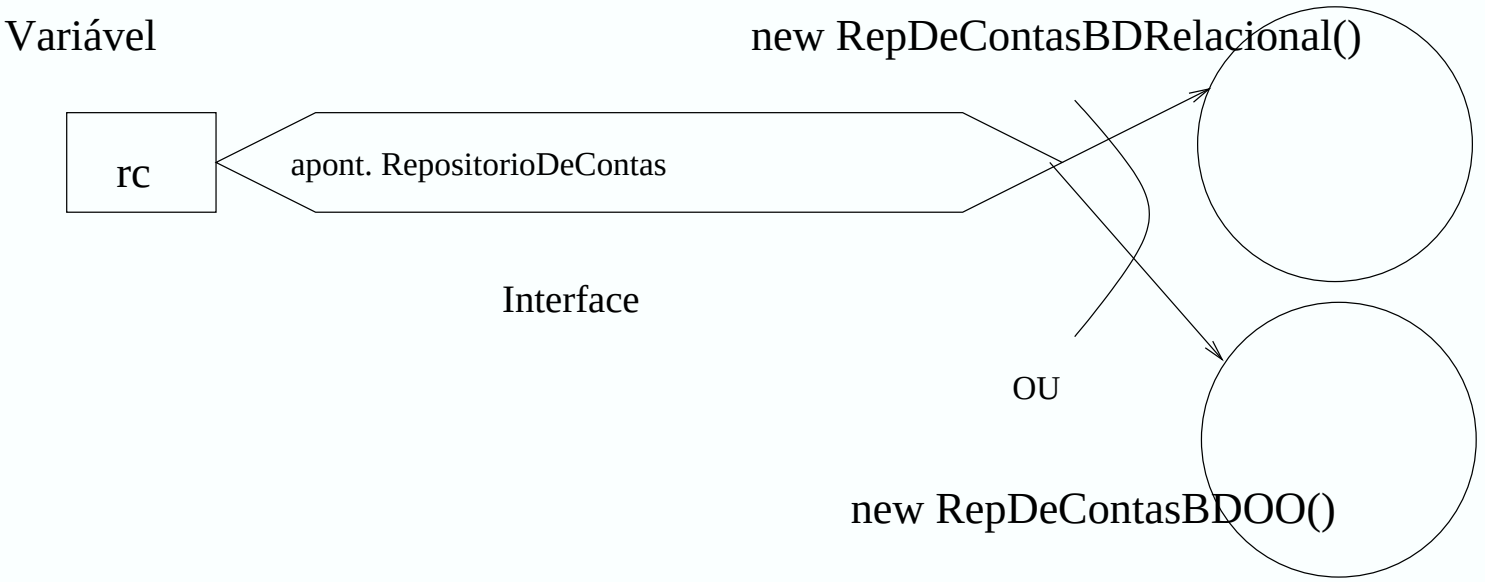
Integração Objeto-Relacional: separação explícita das classes de negócios das classes que manipulam e armazenam dados em um BD relacional.



Exemplo: Integração Objeto-Relacional(II)

RepositorioDeContas rc = new RepDeContasBDRelacional();
RepositorioDeContas rc = new RepDeContasBDOO();

Diferentes Tipos de Objetos



Classes Concretas que implementam a interface

Exemplo: Objeto-Relacional - Código (I)

```
// arquivo CadastroDeContas.java
class CadastroDeContas{
   private RepositorioDeContas rc; // definição de
           // atributo do tipo RepositorioDeContas
   CadastroDeContas(RepositorioDeContas r){
      rc = r; // recebe referência de um objeto cuja
// classe implementa a interface RepositorioDeContas
   }
   void creditar(int num, float val){
      Conta c = rc.procurar(num);
      c.creditar(val);
      rc.atualizar(c);
   }
} // fim da classe CadastroDeContas
```

Exemplo: Objeto-Relacional - Código (II)

```
// arquivo Principal.java
public class Principal{
   public static void main(String args[]){
      CadastroDeContas cadContas =
             new CadastroDeContas(
                new RepDeContasBDRelacional());
      cadContas.creditar(1, 100);
         ...
   }
} // fim da classe Principal
```

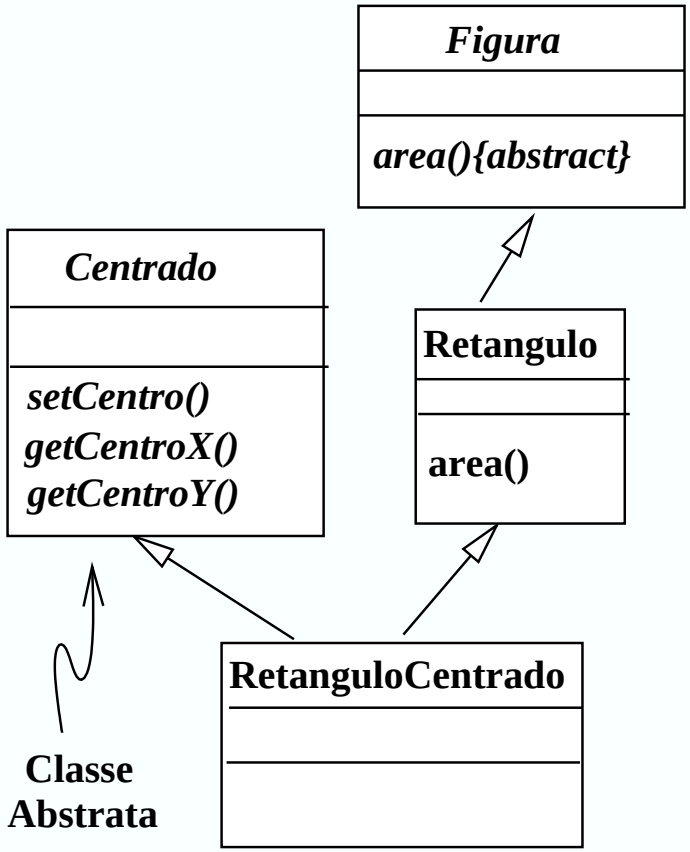
Hierarquia de Interfaces X Hierarquia de Classes (I)

- Uma interface especifica de forma explícita um TAD e uma classe fornece a implementação da especificação de um TAD
- Interfaces representam um nível superior de abstração ao das classes
- Isto é, podemos projetar todas as interfaces de uma aplicação explicitamente, antes de decidirmos a sua forma de implementação

Hierarquia de Interfaces X Hierarquia de Classes (II)

- Essa separação entre o projeto de interface e o projeto de classes é análogo à separação do projeto arquitetônico e o projeto de engenharia de uma construção civil

Herança Múltipla de Classes



Uma Alternativa à Herança Múltipla de Classes

- Interfaces podem ser usadas no lugar de herança múltipla, em linguagens como Java, que não dão suporte a esse recurso.
- O uso de interfaces, nesses casos, tem a vantagem de sempre produzir herança por comportamento, já que todas as classes concretas que implementam uma interface devem oferecer implementações para todos os métodos definidos por ela.
- qual seria a desvantagem?

Exemplo 2 - Java (I)

```
public interface Centrado{
   public void setCenter(double x, double y);
   public double getCentroX();
   public double getCentroY();
} // fim da interface Centrado
```

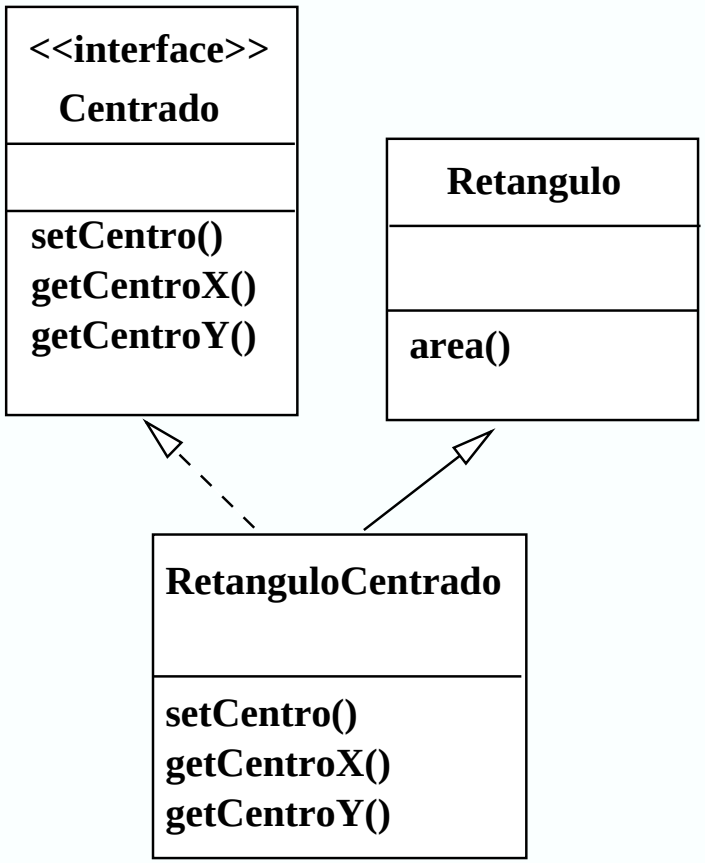
Exemplo 2 - Java (II)

```
public class RetanguloCentrado extends Retangulo
                                    implements Centrado{
   private double cx, cy; // novos atributos
   public RetanguloCentrado(double cx, double cy,
                                   double w, double h){
      super(w, h); // largura e altura do retângulo
      this.cx = cx;
      this.cy = cy;
   }
```

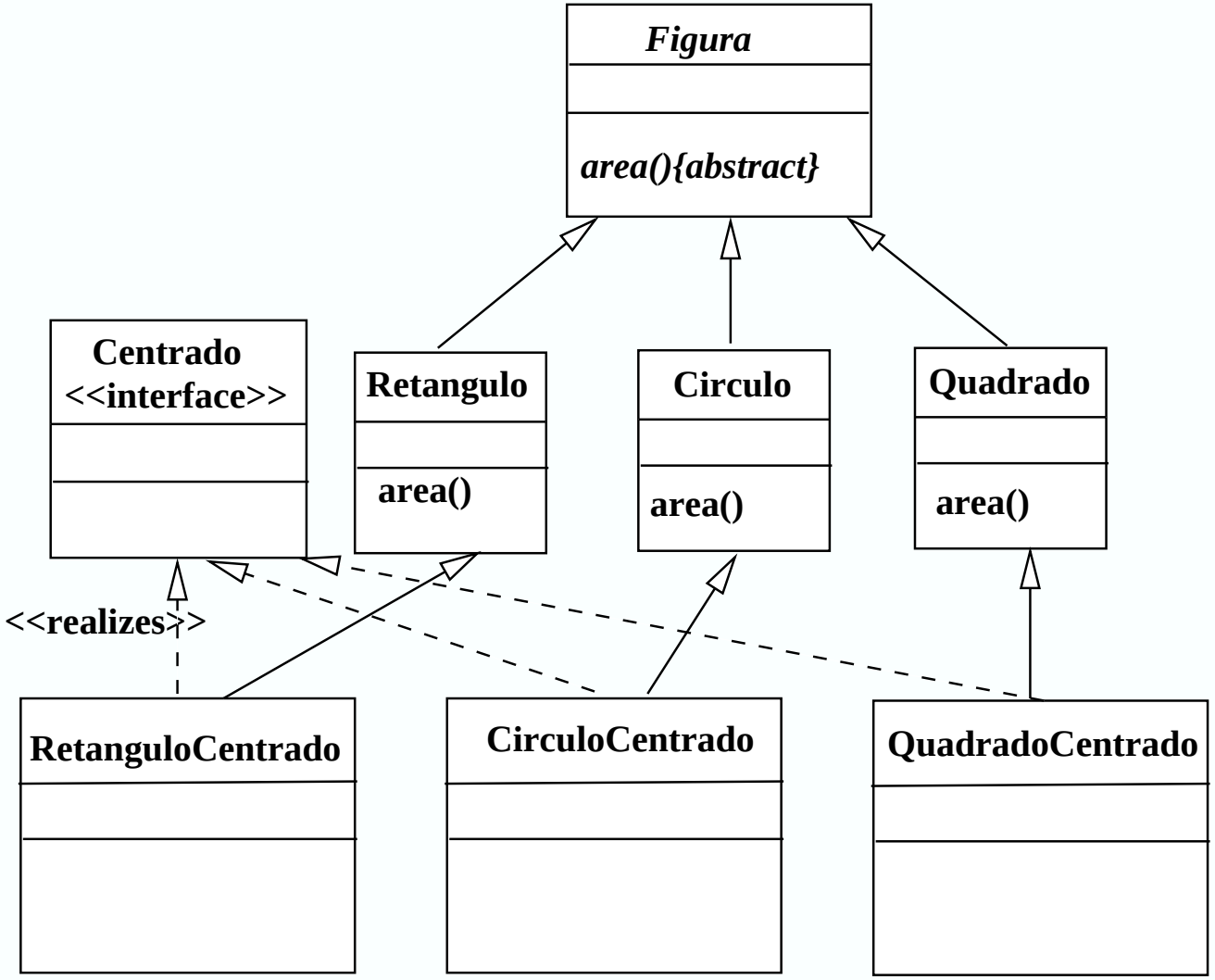
Exemplo 2 - Java (III)

```
    public void setCentro(double x, double y){
       cx = x; cy = y;
    }
    public getCentroX(){return cx;}
    public getCentroY(){return cy;}
} // fim da classe RetanguloCentrado
```

Exemplo 2 - Java (IV)



Exemplo 2 Completo (I)



Exemplo 2 Completo (II)

- A classe Figura é superclasse de Circulo, Quadrado e Retângulo. Cada classe derivada redefine a operação *area()* herdada do pai
- As classes Circulo, Quadrado e Retângulo não implementam a interface Centrado
- As instâncias dos tipos RetanguloCentrado, CirculoCentrado e QuadradoCentrado, que implementam a interface Centrado também podem ser tratadas como instâncias desse tipo.

Exemplo 2 Completo (III)

- Interfaces são tipos de Java, tal como as classes.
- Quando uma classe implementa uma interface, instâncias daquela classe podem ser atribuídas às variáveis do tipo interface.
- Entretanto, não é necessário fazer "casting" de irmãos de RetanguloCentrado para Centrado; antes de chamarmos os métodos *setCentro()* ou *getCentroX()* ou *getCentroY()*. Ex.: `Centrado c = new RetanguloCentrado();`
- RetanguloCentrado implementa *setCentro()*, *getCentroX()* e *getCentroY()* e herda o método *area()* da superclasse Retangulo.

Exemplo 2 Completo - Código (I)

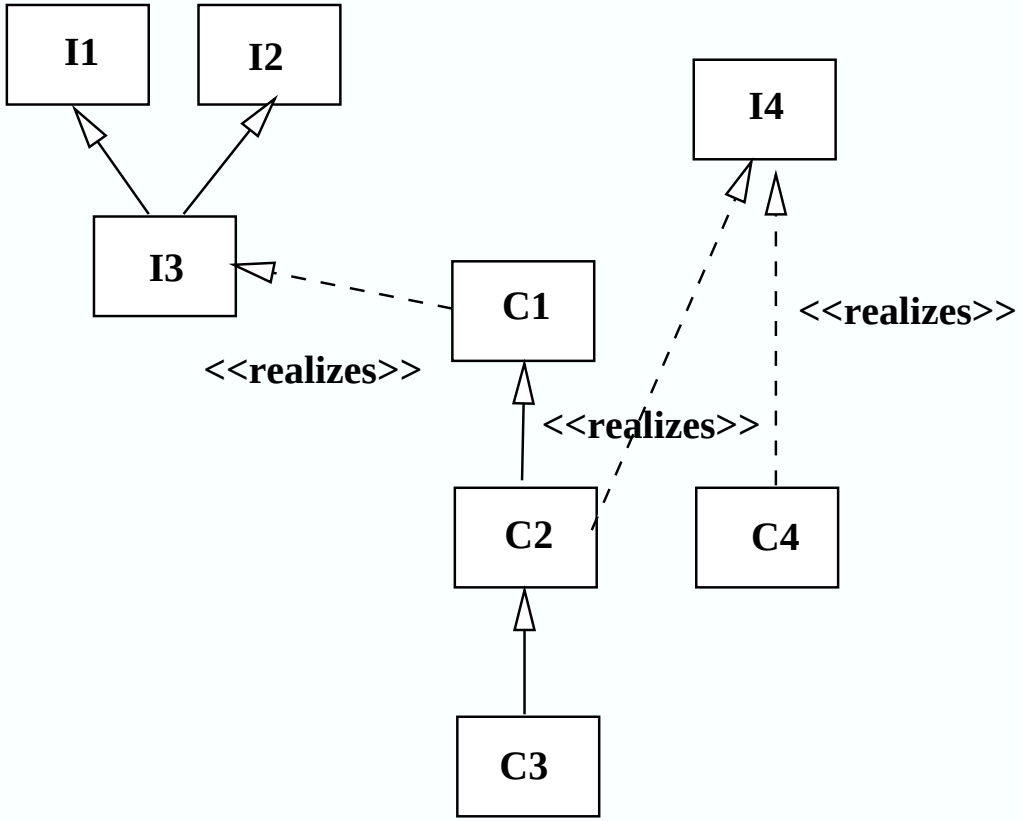
```
Figura[] shapes = new Figura[3]; // cria um array de
          // referências para objetos do tipo Figura
shapes[0] = new RetanguloCentrado(1.0, 1.0, 1.0);
shapes[1] = new CirculoCentrado(2.5, 2.0, 3.0);
shapes[2] = new QuadradoCentrado(2.3, 4.5, 3.4);
double totalArea = 0;
double totalDistance = 0;
for(int i=0; i < shapes.length; i++){
   totalArea += shapes[i].area();
   if (shapes[i] instanceof Centrado){ // é necessário
       // ‘‘casting’’ de tipoFigura para tipo Centrado
      Centrado c = (Centrado) shapes[i];
```

Exemplo 2 Completo - Código (II)

```
        double cx = c.getCentroX();
        double cy = c.getCentroY();
        double totalDistance+= Math.sqrt(cx*cx + cy*cy);
    } // fim do if
} // fim do for

System.out.println(``Área média:'' +
                   totalArea/shapes.length);
System.out.println(``Distância média:'' +
                   totalDistance/shapes.length);
```

Exemplo 3 (I)



Exemplo 3 (II)

- Dados os seguintes objetos:

```
C1 c1 = new C1();
C2 c2 = new C2();
C3 c3 = new C3();
C4 c4 = new C4();
```

- As seguintes atribuições são válidas: `I1 i1 = c1;`

```
//classe C1 implementa a interface I1
I1 i1 = c2; //classe C2 implementa a
interface I1
I1 i1 = c3; //classe C3 implementa a
interface I1
```

Exemplo 3 (III)

```
I2 i2 = c1; //classe C1 implementa a interface I2
I2 i2 = c2; //classe C2 implementa a interface I2
I2 i2 = c3; //classe C3 implementa a interface I2
I3 i3 = c1; //classe C1 implementa a interface I3
I3 i3 = c2; //classe C2 implementa a interface I3
I3 i3 = c3; //classe C3 implementa a interface I3
I4 i4 = c4; //classe C4 implementa a interface I4
I4 i4 = c2; //classe C2 implementa a interface I4
I4 i4 = c3; //classe C3 implementa a interface I4
```