UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

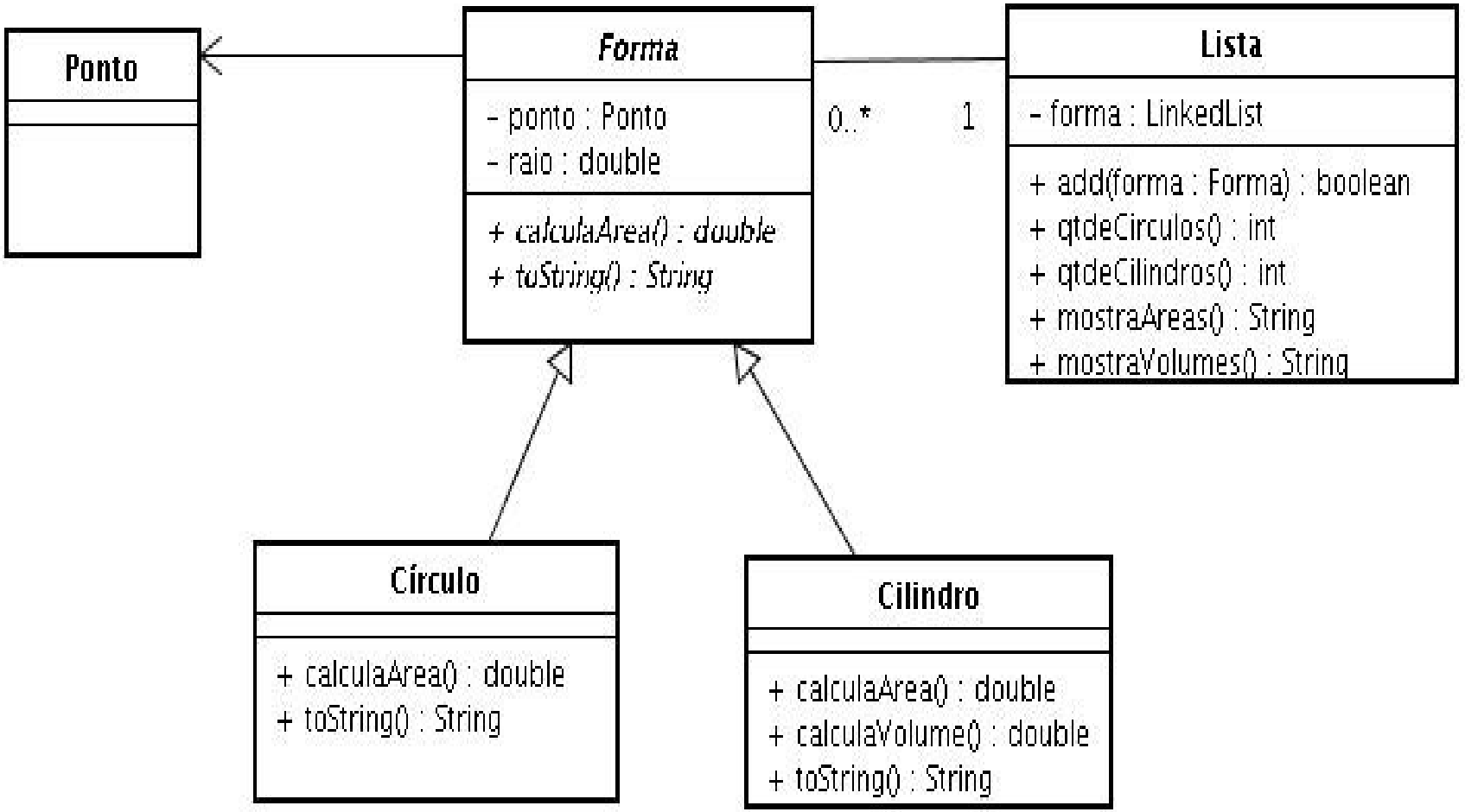
Programação Orientada a Objetos

Ciências da Computação

Herança

Polimorfismo

Estudo de Caso



Polimorfismo

Estudo de Caso

```
public int qtdeCirculos() {
      int num = 0;

      Iterator i = forma.iterator();
      Forma f;

      while (i.hasNext()) {
            f = (Forma) i.next();
            if (f instanceof Circulo) num++;
      }
      return num;
}
```

Polimorfismo

Estudo de Caso

```
public String mostraAreas() {
      String areas = null;

      Iterator i = lista.iterator();
      Forma f;

      while (i.hasNext()) {
            f = (Forma) i.next();
            areas += “\n” + f.calculaArea();
            }
      }
      return areas;
}
```

Polimorfismo
Princípio da Substituição

- Lembre-se de que é possível atribuir um objeto de subclasse a uma referência de superclasse, graças ao princípio da substituição ou *principle of substitutability*
-
- *"se temos duas classes, A e B, tal que a classe B é uma subclasse de A, é possível substituir instâncias da classe B por instâncias da classe A em qualquer situação com nenhum efeito observável".*

Polimorfismo

Princípio da Substituição

- ....
- Forma forma;
- Circulo circulo = new Circulo();
- forma = circulo; // princípio da substituição: 'círculo' substituído por 'forma'
- forma.calculaArea() // invoca o método calculaArea() de Círculo através de Forma.
-

- forma = new Cilindro(); //princípio da substituição: objeto de Cilindro substituído // por 'forma'
- forma.calculaArea(); // invoca o método calculaArea() de Cilindro através de Forma.
- .....

Polimorfismo

- Polimorfismo
  - Programas extensíveis
  - Processamento de objetos de superclasse de forma genérica
  - Facilita a adição de classes à hierarquia
    - Pouca ou nenhuma modificação requerida
    - Apenas partes do programa que necessitam conhecimento direto da nova classe devem ser alteradas

Polimorfismo

Vinculação Dinâmica de Métodos

- Vinculação dinâmica de métodos
  - Em tempo de execução, a chamada ao método é direcionada à versão apropriada
  - Método da classe apropriado é chamado

Polimorfismo
Vinculação Dinâmica de Métodos

- Exemplo
  - **`Quadrado`**, **`Círculo`**, e **`Cilindro`** são subclasses de **`Forma`**
    - Cada uma delas pode ter um método **`desenha()`**
  - A chamada a **`desenha()`** usa referências de superclasse
    - Em tempo de execução, o programa determina para qual classe a referência está realmente apontando
    - Chama o método **`desenha()`** apropriado

Novas Classes e Alocação Dinâmica

- Alocação Dinâmica
  - Acomoda novas classes
  - Tipo do objeto não é requerido em tempo de compilação
  - Em tempo de execução, a invocação ao método é combinada com o objeto

Polimorfismo

Métodos e Classes `final`

- Declarando variáveis **`final`**
  - Elas não podem ser modificadas após a declaração
  - Devem ser inicializadas na declaração
- Declarando métodos **`final`**
  - Não pode ser sobreposto em uma subclasse
  - métodos **`static`** e **`private`** são implicitamente **`final`**
- Declarando classes **`final`**
  - Não pode ser uma superclasse (novas classes não podem ser derivadas a partir dela)
  - Todos os métodos na classe são implicitamente **`final`**

Polimorfismo

Superclasses Abstratas e Concretas

- Classes Abstratas (superclasses abstratas)
  - Único propósito é ser uma superclasse
    - Outras classes herdam dela
  - Não é possível instanciar objetos de uma classe abstrata
    - Variáveis de instância e construtores continuam a existir
  - Classe declarada com a palavra-chave **`abstract`**
- Classe concreta
  - É possível instanciar objetos
  - Específica
- Hierarquias de classe
  - Classes gerais são usualmente **`abstract`**
    - **`Forma`** - muito genérica para ser concreta

Polimorfismo

- Com o polimorfismo
  - Novas classes podem ser adicionadas facilmente
  - A invocação a um método pode resultar em diferentes ações, dependendo do objeto que recebe a invocação
- Referências
  - Referências podem ser criadas para classes abstratas
    - Não é permitida a instanciação de objetos de classes abstratas
-

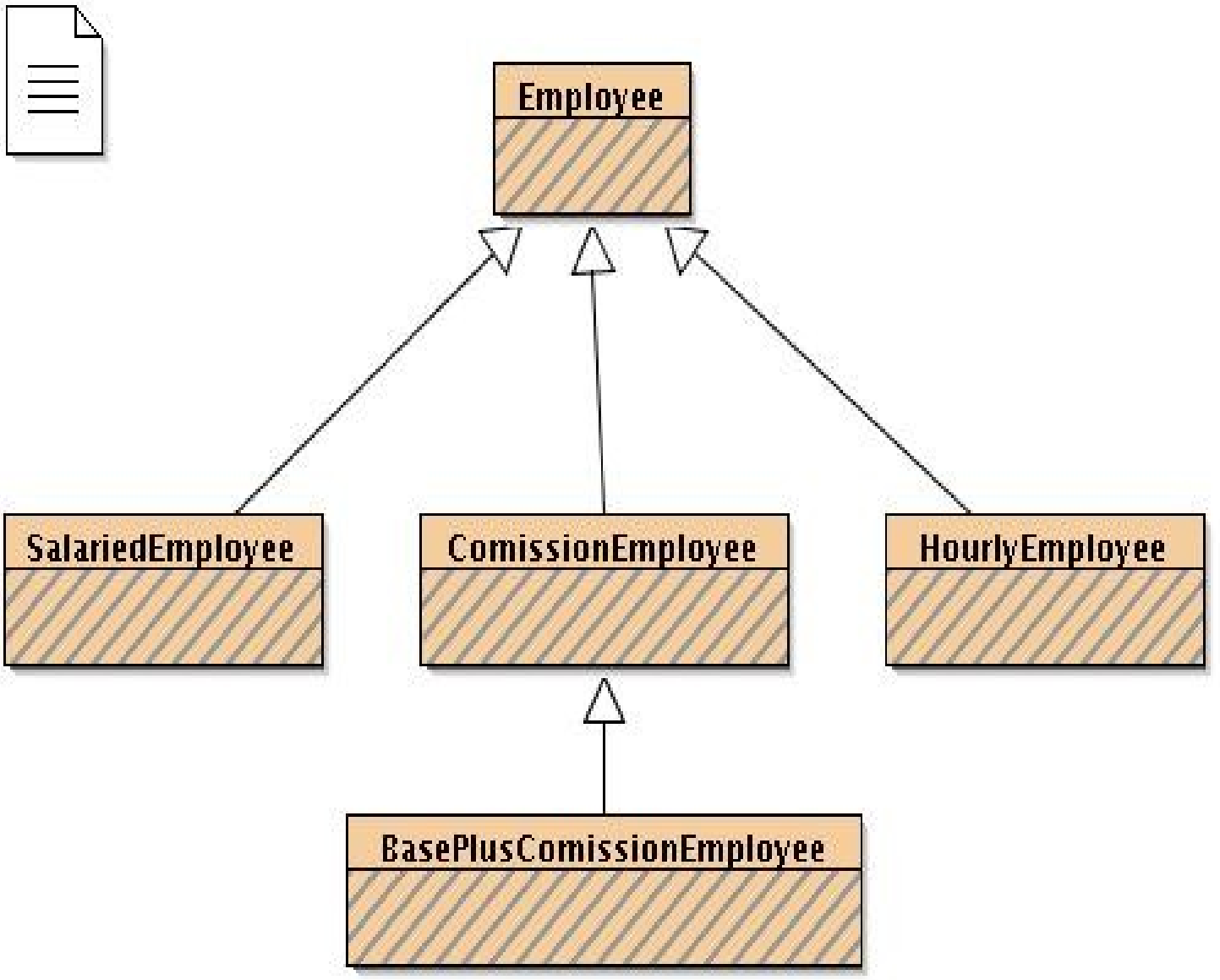
Polimorfismo

Métodos abstratos

- métodos abstratos
  - palavra-chave **`abstract`**
    - Uma classe com método **`abstract`** deve ser **`abstract`**
  - métodos **`abstract`** devem ser sobrepostos na subclasse
    - Se isso não ocorre, a subclasse deve ser **`abstract`**

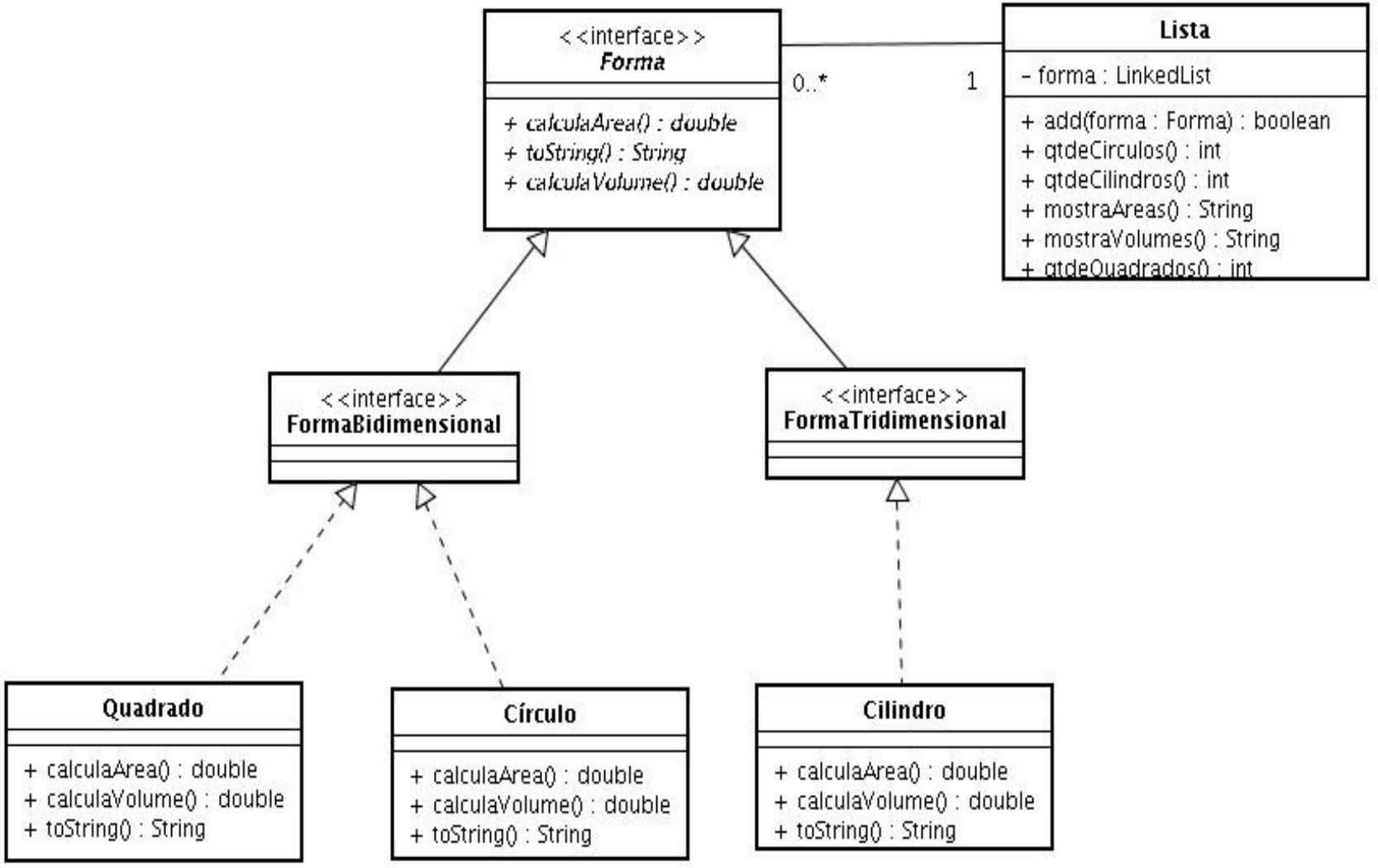
Polimorfismo

Estudo de Caso: Folha de Pagamentos



Polimorfismo

Estudo de Caso: Criando e Utilizando Interfaces



Polimorfismo

Estudo de Caso: Criando e Utilizando Interfaces

```
public interface Forma {

    public abstract double calculaArea();
    public abstract double calculaVolume();
    public abstract String toString();
}
```

```
public interface FormaBidimensional extends Forma {

}
```

Polimorfismo

Estudo de Caso: Criando e Utilizando Interfaces

```
public class Quadrado implements FormaBidimensional {
                ...
                ...
}
```

Polimorfismo

Utilizando Interfaces

- Interface
  - Palavra-chave **`interface`**
  - Possui um conjunto de métodos **`public abstract`**
  - Pode conter dados **`public final static`**
- Usando interfaces
  - Classe especifica que usa interfaces com a palavra-chave **`implements`**
  - Classe deve definir todos os métodos abstratos existentes na interface
    - Deve usar o mesmo número de argumentos, mesmo tipo de retorno

Polimorfismo

Utilizando Interfaces

- Usando interfaces
  - Usar interfaces é como assinar um contrato
    - “Eu definirei todos os métodos especificados na interface"
  - O mesmo que o relacionamento “é um" da herança
  - Interfaces são usadas no lugar de classes abstratas
    - Usadas quando não há nenhuma implementação padrão
  - Tipicamente tipos de dados públicos
    - Interface definida em seu próprio arquivo **`.java`**
    - Nome da interface é o mesmo nome do arquivo
  -

Polimorfismo
Utilizando Interfaces

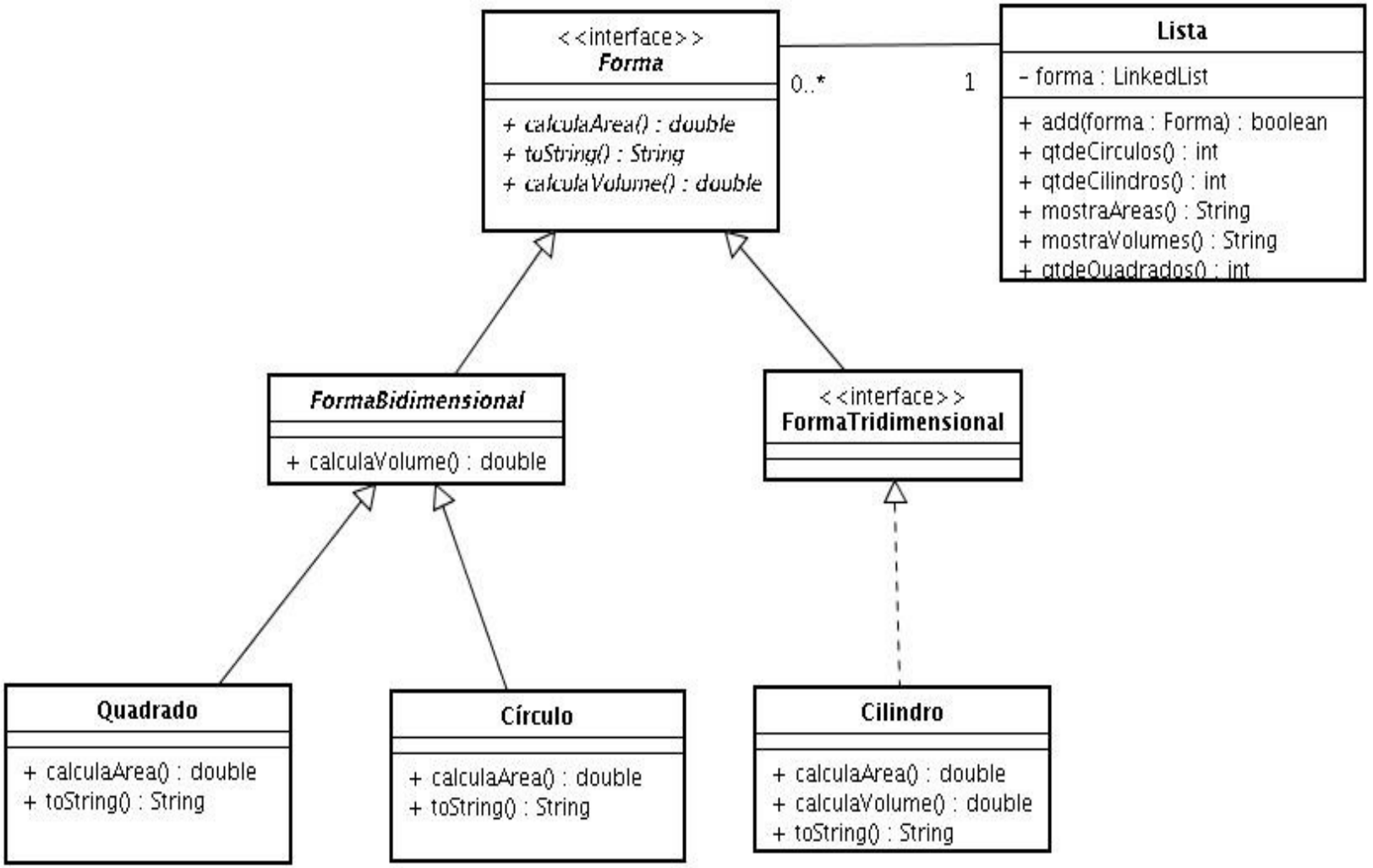
- Outro uso para interfaces
  - Definir um conjunto de constantes usadas por muitas classes
  -
```
public interface Constantes {
  public static final int UM = 1;
  public static final int DOIS = 2;
  public static final int TRES = 3;
}
```

Polimorfismo
Estudo de Caso

- **Retornando ao exemplo ...**
-
- Como ficaria a implementação se a interface FormaBidimensional fosse uma classe abstrata? Quais as implicações dessa mudança?

Polimorfismo

Estudo de Caso



Polimorfismo
Estudo de Caso

```
public double calculaVolume()  {
        return 0;
}
```

- `calculaVolume()` seria implementado apenas na classe abstrata FormaBidimensional e herdado por Quadrado e Círculo.

Créditos

Prof. Sérgio T. Carvalho

sergio@inf.ufg.br