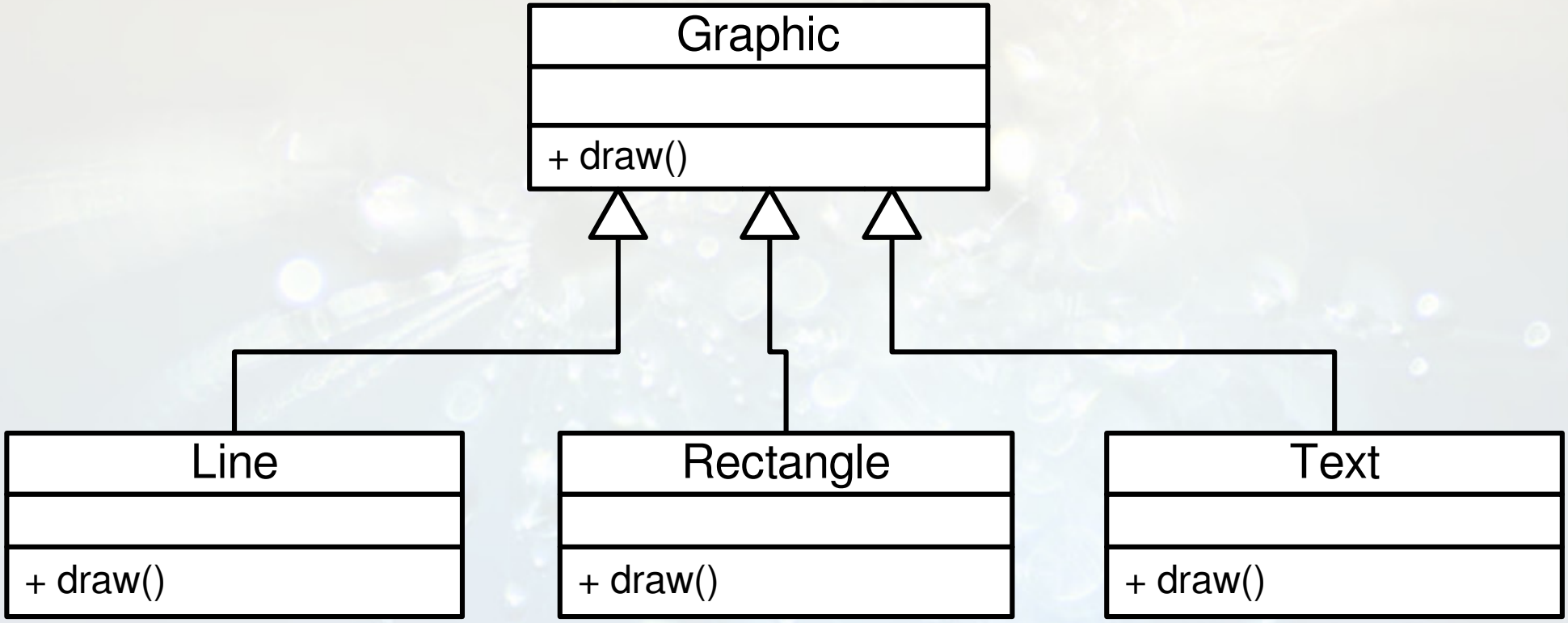
# Programação Orientada a Objetos

# Interface

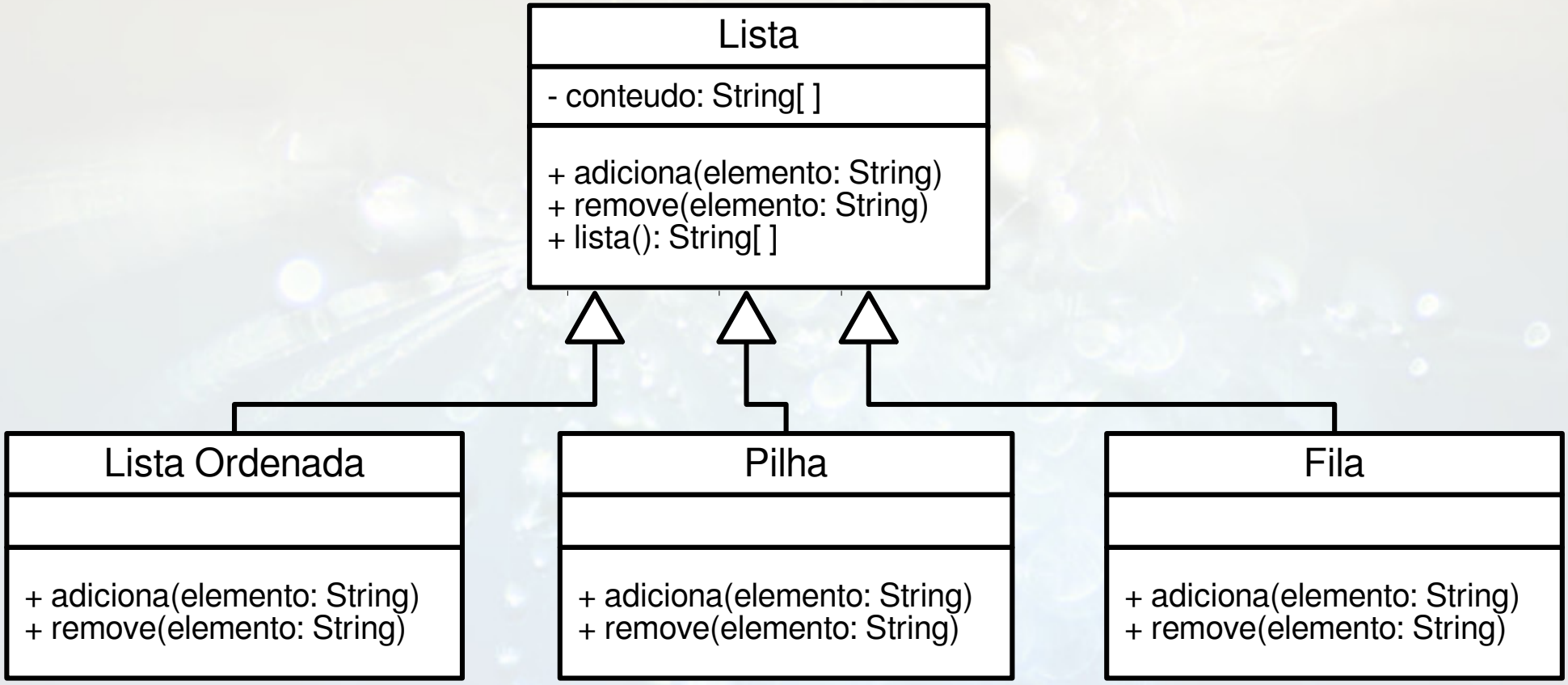
André Santanchè
Laboratory of Information Systems – LIS
Instituto de Computação – UNICAMP
Maio 2020

# Retomando a questão da Generalização

# Generalizando uma Família Gráfica



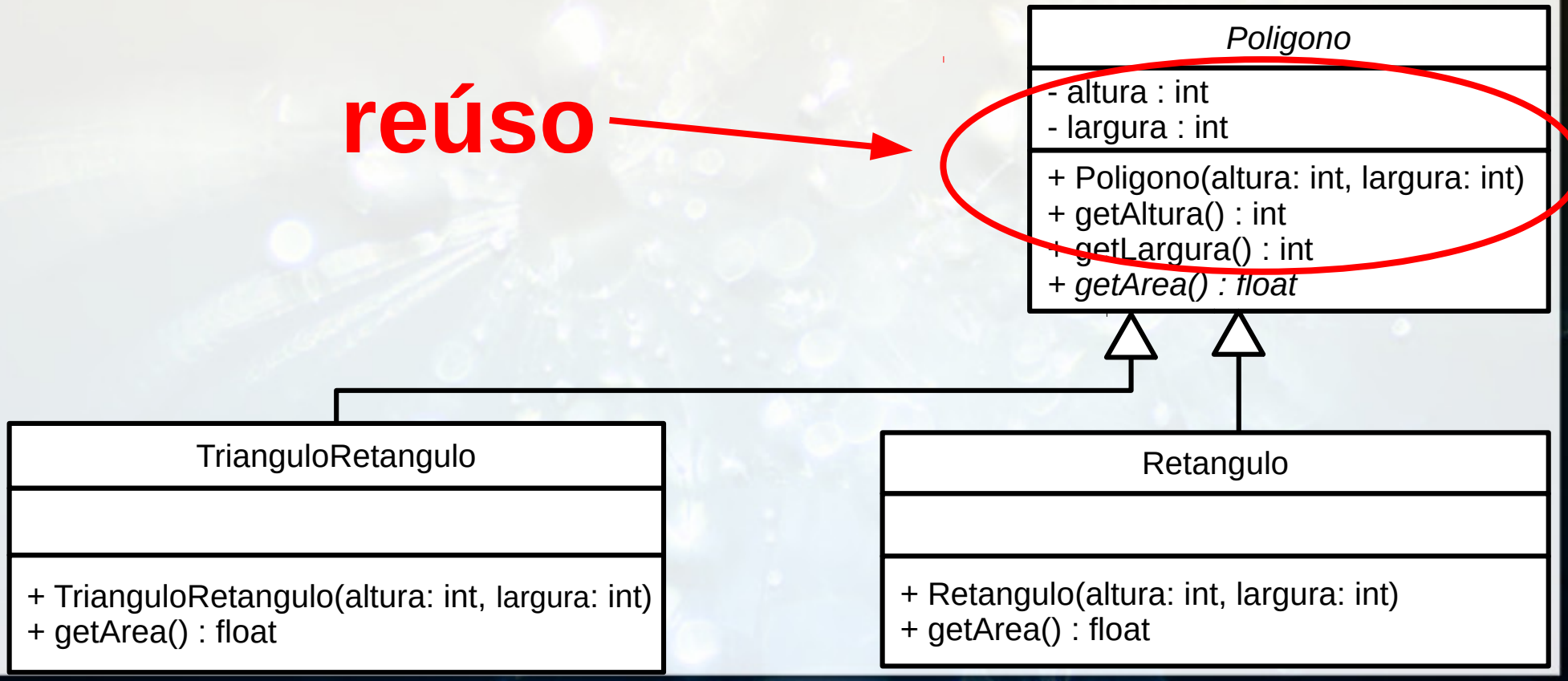
# Generalizando uma Família de Listas



# Classe Abstrata
# Dois Papéis Interligados

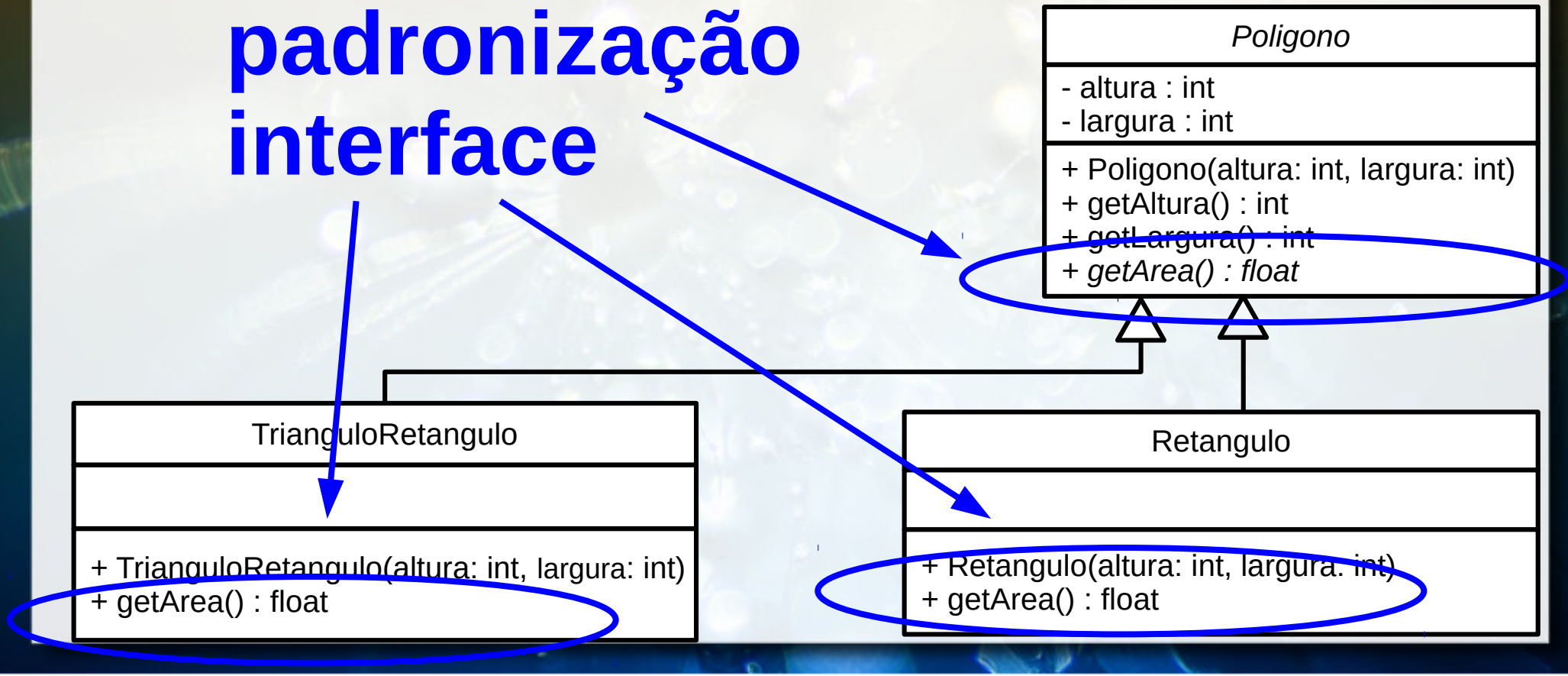
- Permite padrozinar uma interface de acesso sem implementação associada
- A herança permite reúso de código

# Retângulo Java



# Retângulo Java

**padronização**
**interface**



# Padronizando Interface por Herança

## Limites

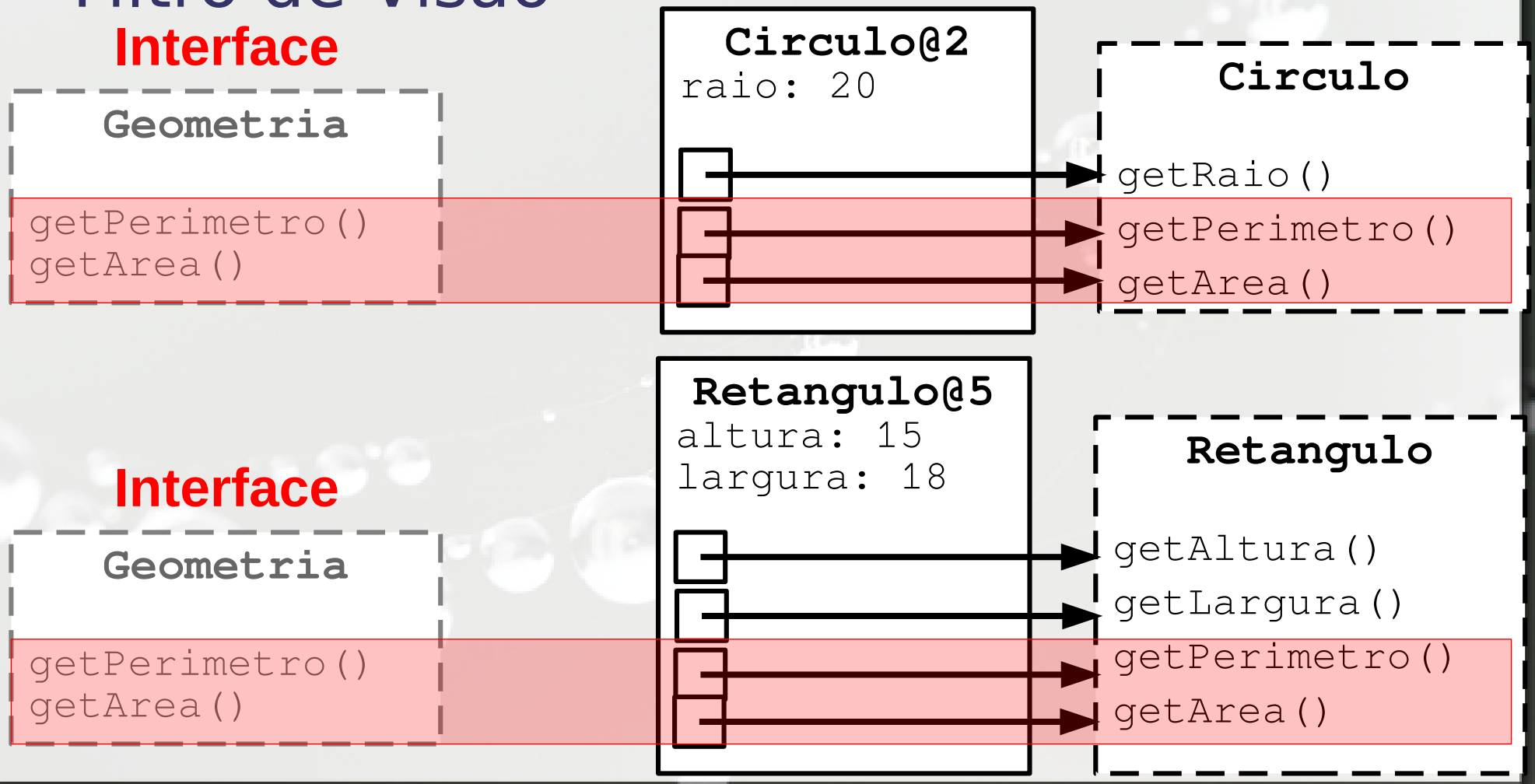
- Muitas vezes o único objetivo é a padronização da Interface
- Herança limita o reuso do código apenas da classe que padroniza a interface
- Não é possível padronizar duas interfaces na mesma classe

# Interface

# Interface

- Mecanismo de generalização – padronização de interface
- Não tem relação com herança
  - não é um mecanismo de reúso de código
- Funciona como um “filtro de visão” em classes

# “Filtro de Visão”

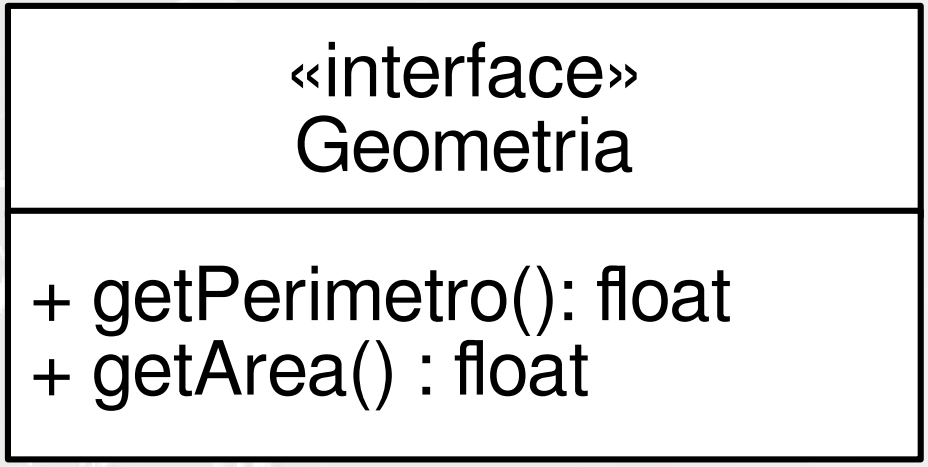


# Interface

- Declara um conjunto de métodos
- Métodos deverão ser implementados por todas as classes que implementam a interface

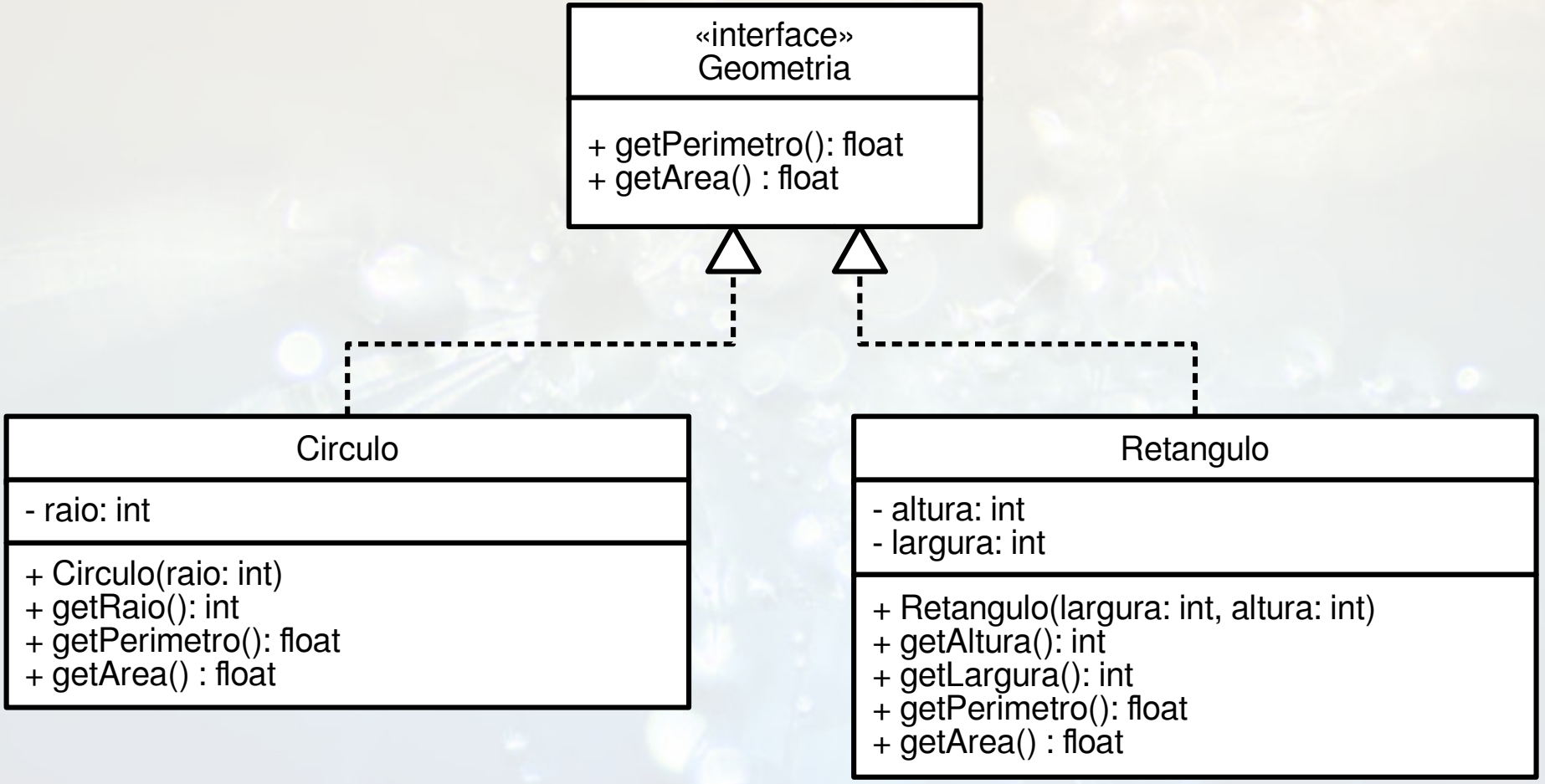
# Interface Geometria

```
public interface Geometria {
    public float getPerimetro();
    public float getArea();
}
```



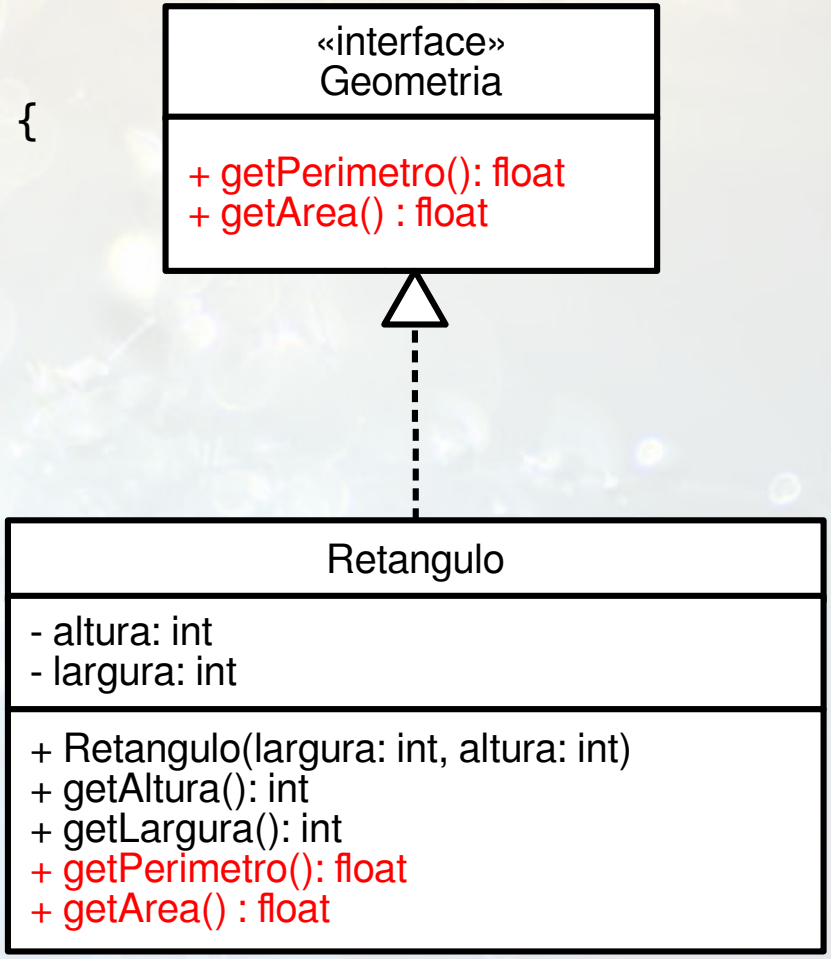

- Indica que todas as classes que a implementarem precisarão implementar getPerimetro() e getArea() com as assinaturas indicadas.

# Interface Geometria



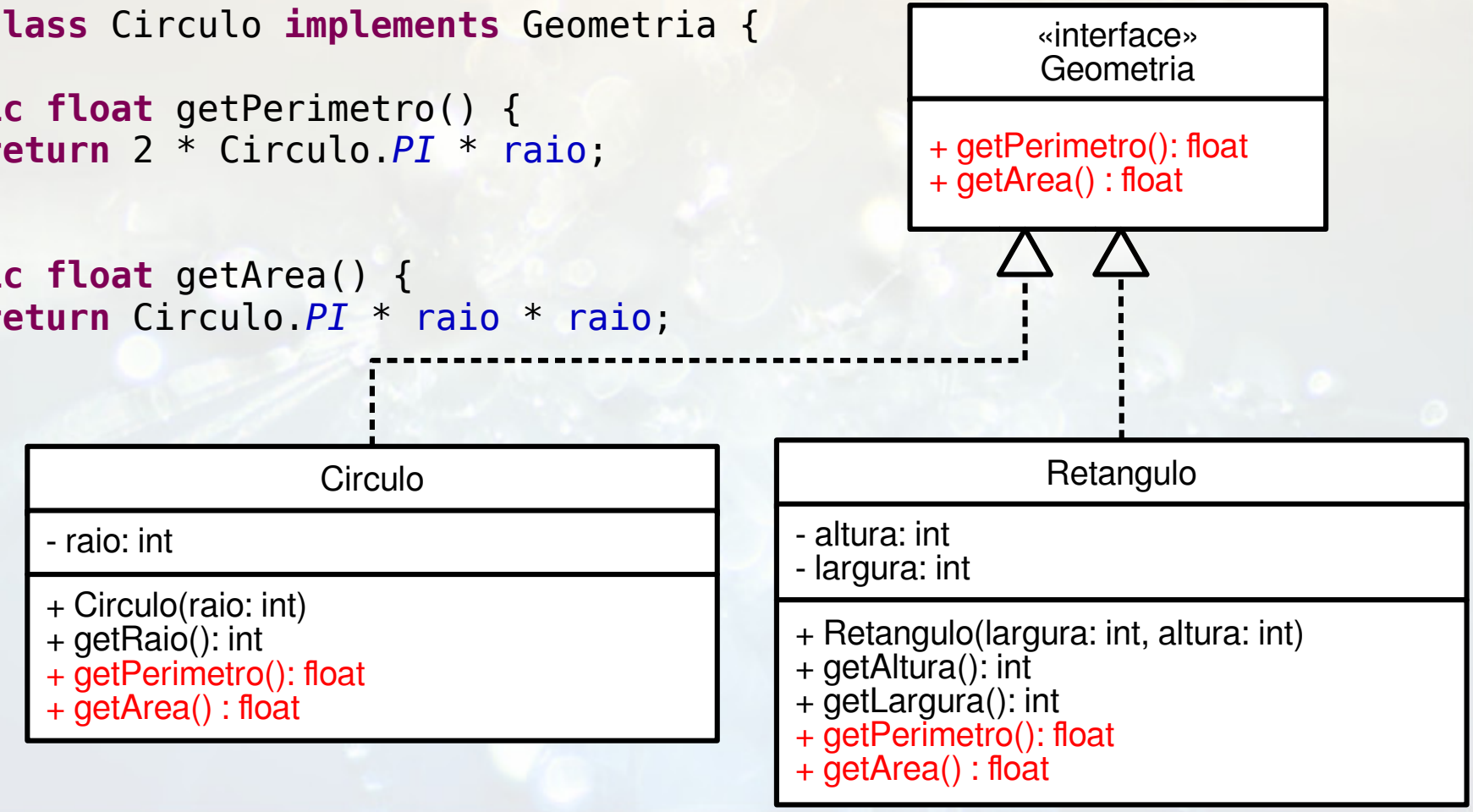
# Retangulo implementa Geometria

```
public class Retangulo implements Geometria {
    ...
    public float getPerimetro() {
        return 2 * (altura + largura);
    }

    public float getArea() {
        return altura * largura;
    }
}
```



# Circulo implementa Geometria

```
public class Circulo implements Geometria {
    ...
    public float getPerimetro() {
        return 2 * Circulo.PI * raio;
    }

    public float getArea() {
        return Circulo.PI * raio * raio;
    }
}
```

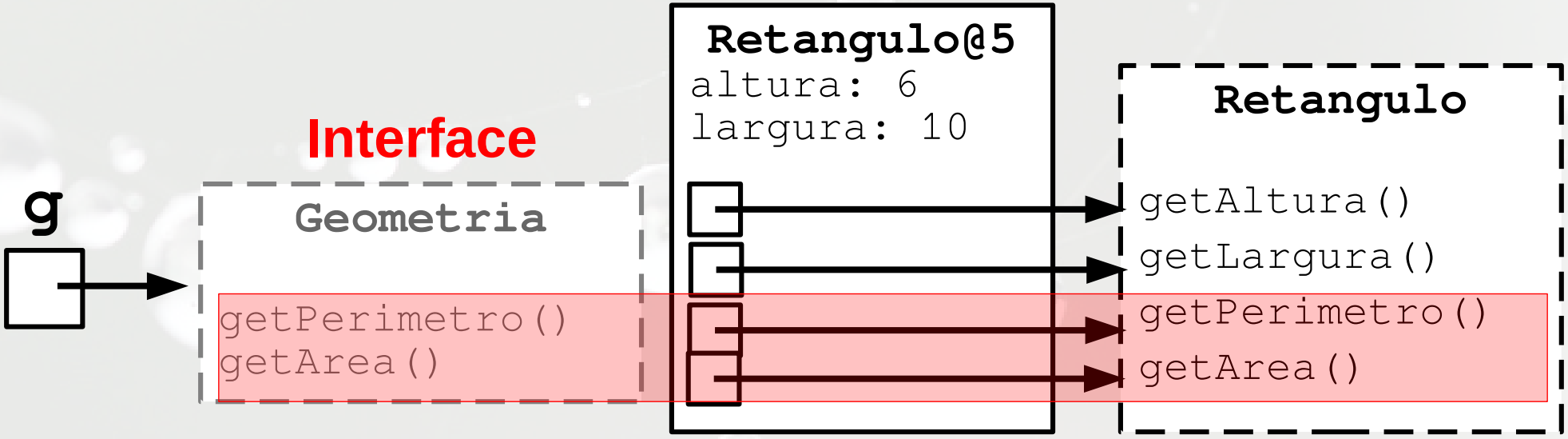


# Usando Interfaces

- Uma variável declarada em uma interface A referência de qualquer classe B que implementa a interface
- Através dessa variável, só estarão “visíveis”:
  - Métodos declarados na Interface A

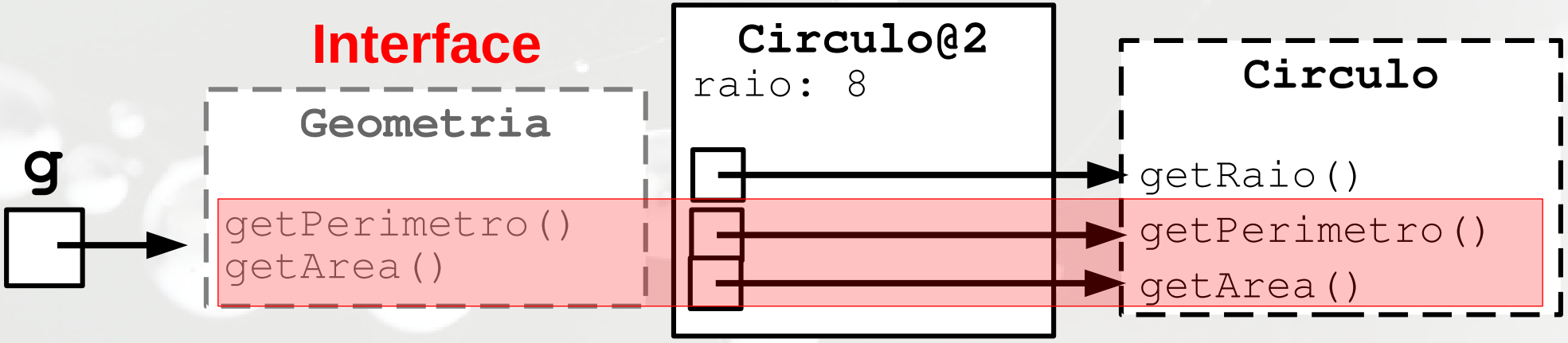
# “Filtro de Visão”

```
Geometria g = new Retangulo(6, 10);

System.out.println("Perímetro do retângulo: " + g.getPerimetro());
System.out.println("Área do retângulo: " + g.getArea());
```



# “Filtro de Visão”

```
g = new Circulo(8);

System.out.println("Perímetro do círculo: " + g.getPerimetro());
System.out.println("Área do círculo: " + g.getArea());
```



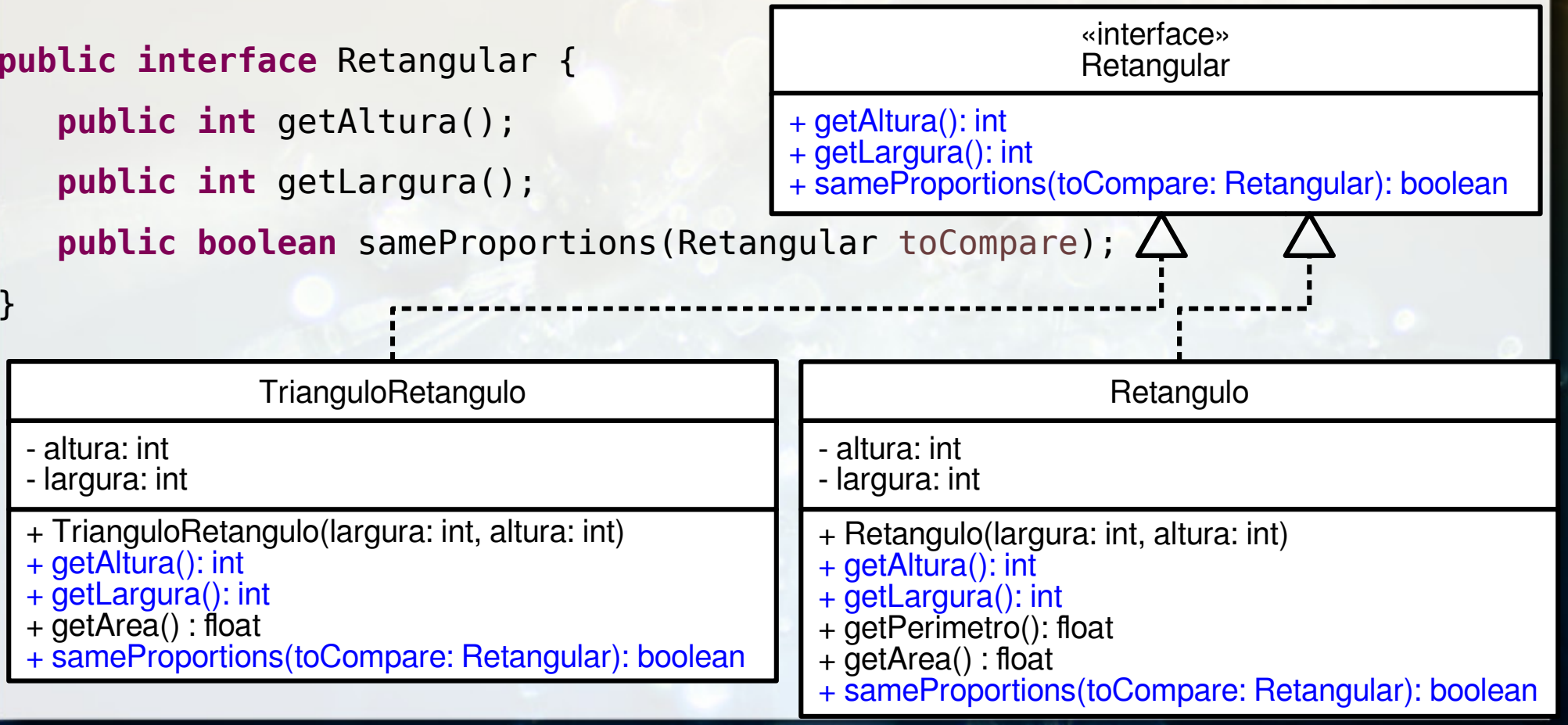
# Programação Orientada a Objetos

# Tópicos Avançados sobre Interfaces

André Santanchè
Laboratory of Information Systems – LIS
Instituto de Computação – UNICAMP
Maio 2020

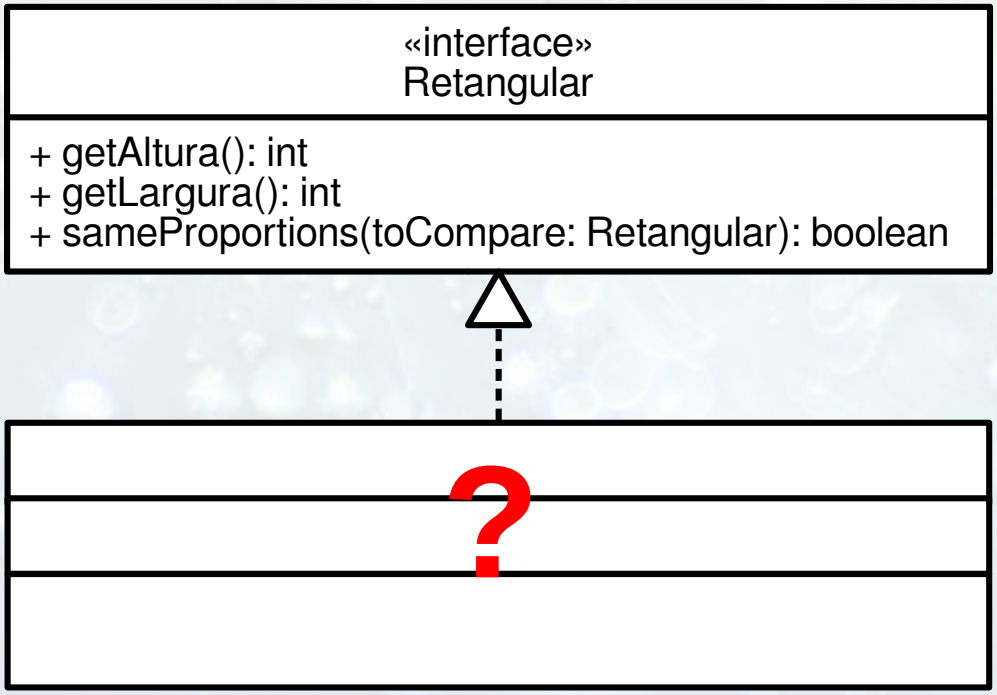
# Recebendo Interface como Parâmetro

```
public interface Retangular {
    public int getAltura();
    public int getLargura();
    public boolean sameProportions(Retangular toCompare);
}
```

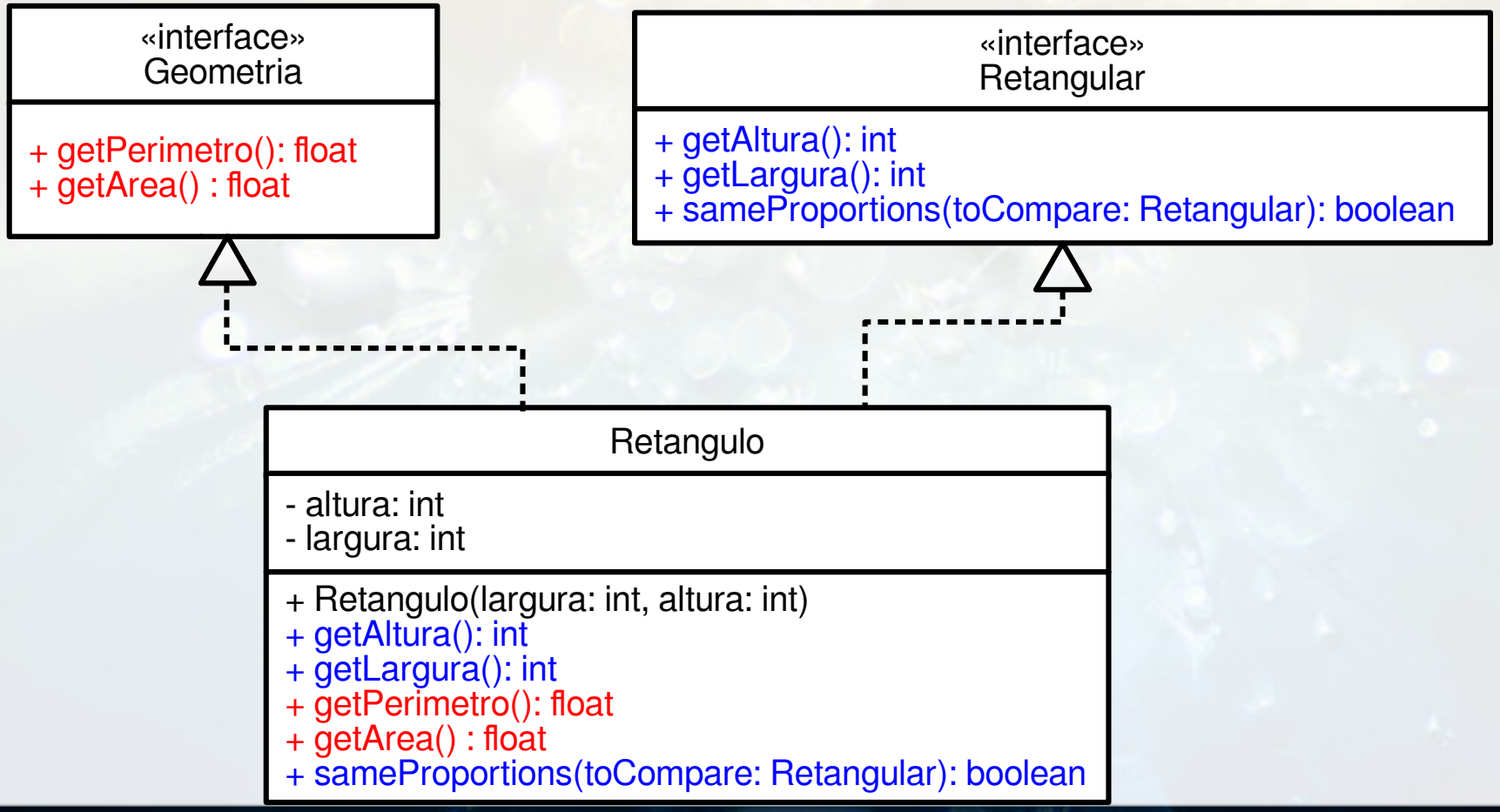


# Recebendo Interface como Parâmetro

```
public boolean sameProportions(Retangular toCompare) {
    return (largura/altura==toCompare.getLargura()/toCompare.getAltura());
}
```



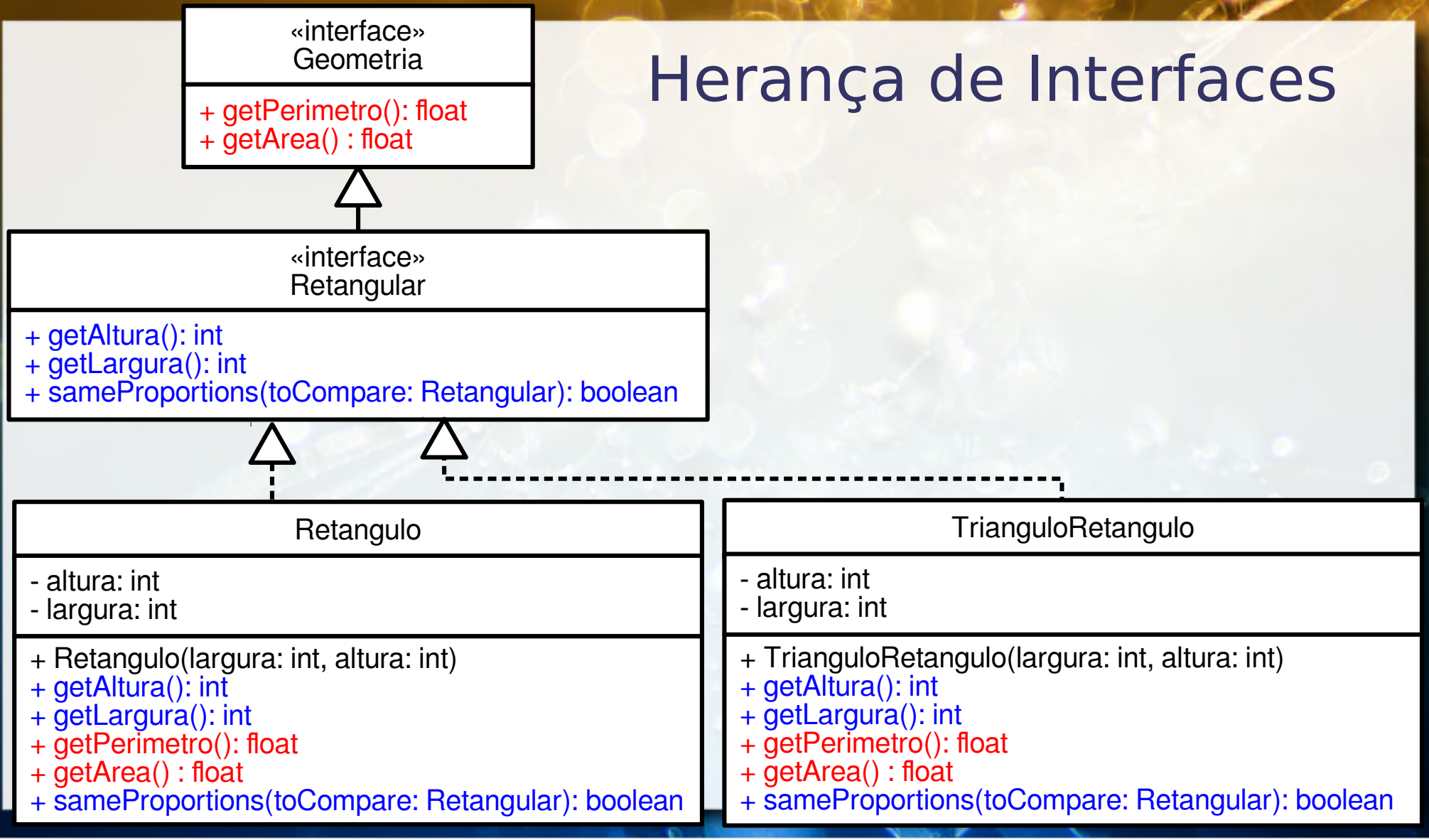
# Implementando Múltiplas Interfaces



# Reunindo Interfaces

Se quisermos reunir interfaces adotadas por `Retangulo` e `TrianguloRetangulo`?

# Herança de Interfaces



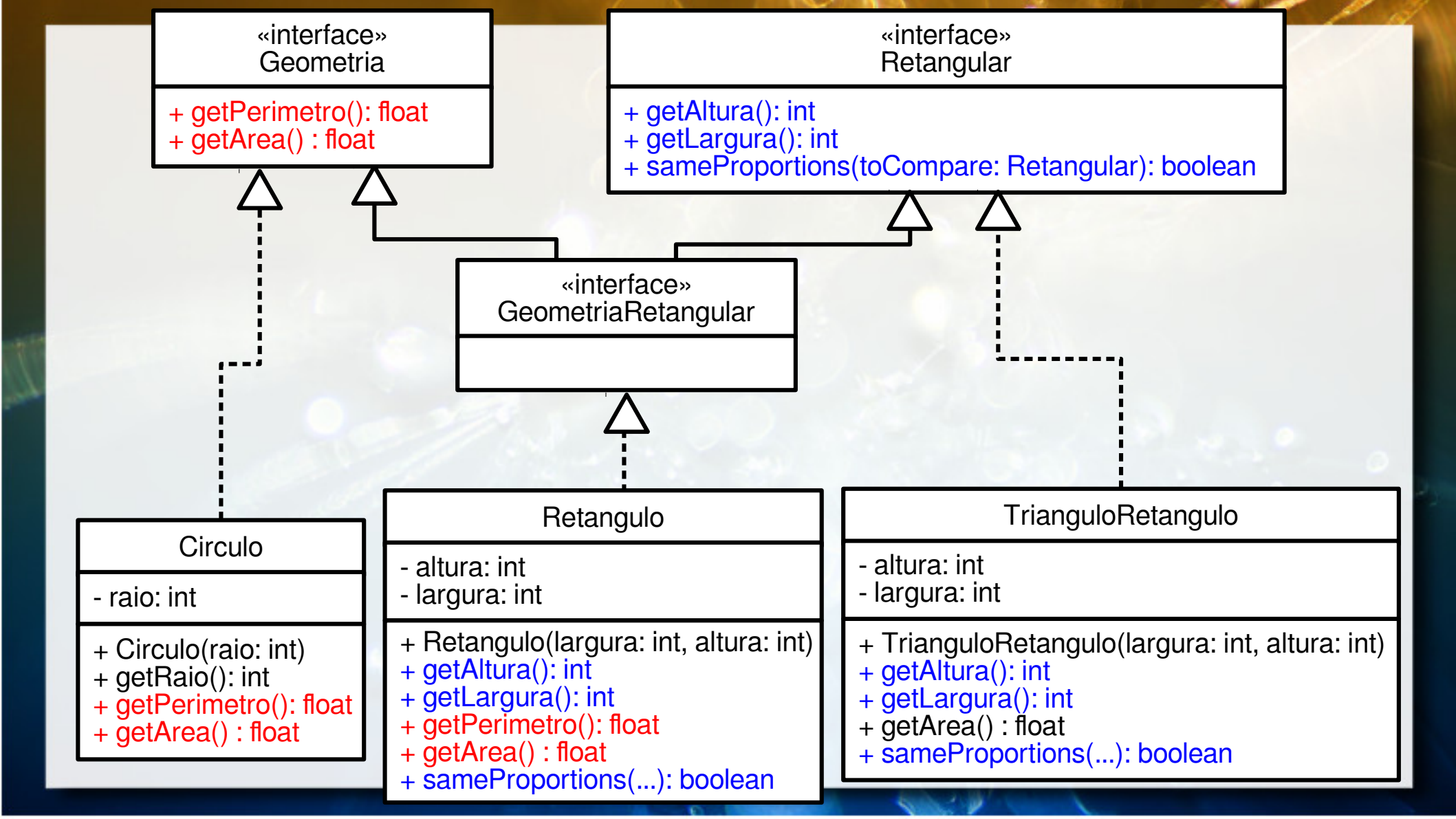
# Herança de Interfaces

- Mecanismo de extensão de interface a partir da existente
- Neste exemplo, não é possível dissociar `Retangular` de `Geometria`

# Herança Múltipla de Interfaces

# Herança de Múltiplas Interfaces

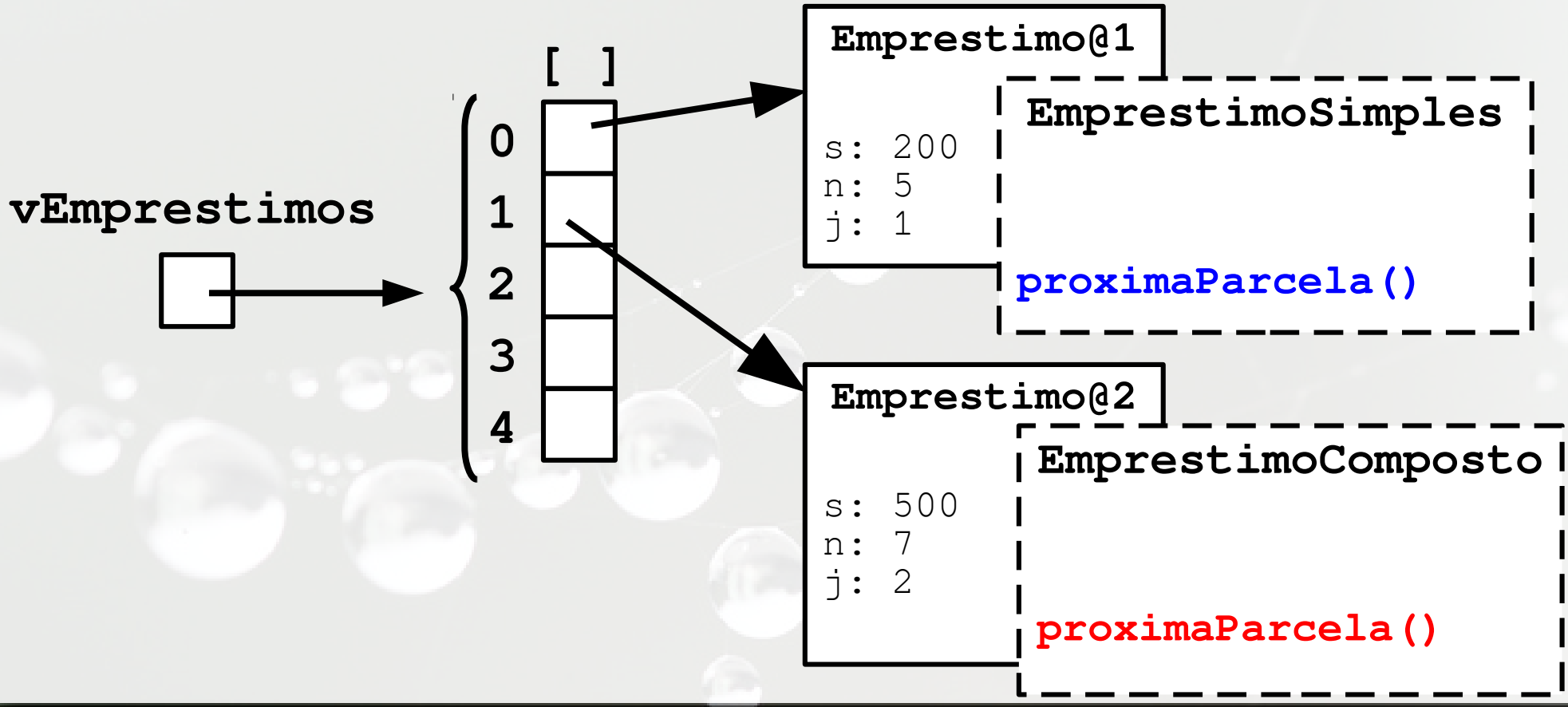
- Uma interface pode ser herdeira de mais de uma interface
- Será a combinação dos métodos de todas as interfaces que ela herda
  - mais métodos que possa acrescentar


«interface»
Geometria
+ getPerimetro(): float
+ getArea() : float
«interface»
Retangular
+ getAltura(): int
+ getLargura(): int
+ sameProportions(toCompare: Retangular): boolean
«interface»
GeometriaRetangular
Circulo
- raio: int
+ Circulo(raio: int)
+ getRaio(): int
+ getPerimetro(): float
+ getArea() : float
Retangulo
- altura: int
- largura: int
+ Retangulo(largura: int, altura: int)
+ getAltura(): int
+ getLargura(): int
+ getPerimetro(): float
+ getArea() : float
+ sameProportions(...): boolean
TrianguloRetangulo
- altura: int
- largura: int
+ TrianguloRetangulo(largura: int, altura: int)
+ getAltura(): int
+ getLargura(): int
+ getArea() : float
+ sameProportions(...): boolean

# Exercício

- Crie uma classe `ConjuntoEmprestimos` que os empréstimos simples e compostos

# Vetor de Interfaces



André Santanchè
http://www.ic.unicamp.br/~santanche

# Licença

- Estes slides são concedidos sob uma Licença Creative Commons. Sob as seguintes condições: Atribuição, Uso Não-Comercial e Compartilhamento pela mesma Licença.
- Mais detalhes sobre a referida licença Creative Commons veja no link: http://creativecommons.org/licenses/by-nc-sa/3.0/
- Agradecimento a Picture by Neal Fowler [ https://www.flickr.com/photos/31878512@N06/] por sua fotografia “Explosion” usada na capa e nos fundos, disponível em [https://flic.kr/p/oCNoe6]. Vide licença específica da fotografia.