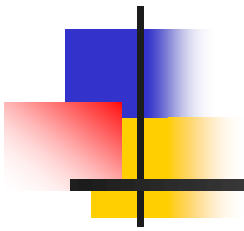
UNIVERSIDADE CATÓLICA DE PERNAMBUCO

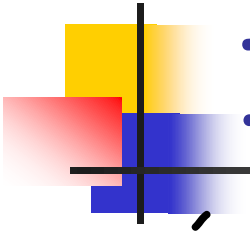


Interfaces

Universidade Católica de Pernambuco
Ciência da Computação

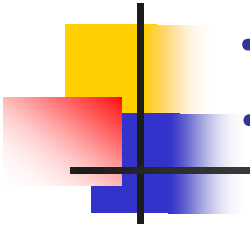
Prof. Márcio Bueno
poonoite@marciobueno.com

Fonte: Material da Profª Karina Oliveira



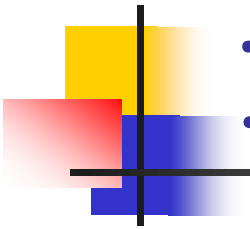
Interfaces

- É utilizada para agrupar conceitos em termos de projeto das classes.
- É um conjunto de declarações de métodos (nome, tipo de retorno e parâmetros) desprovidos de implementação.
- Cabe ao programador que deseja implementar uma determinada interface providenciar uma implementação para os métodos definidos na mesma.



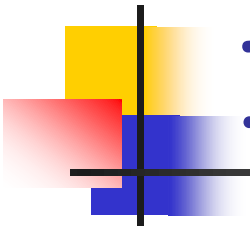
Interfaces

- Toda interface pode ser vista como uma classe abstrata, mas a recíproca não é verdadeira.
- Quando dizemos que uma classe implementa uma interface obrigatoriamente essa classe tem que implementar todos os métodos declarados na interface.
- Neste sentido, uma interface pode ser vista como sendo um contrato estabelecido entre a classe que implementa a interface (provedora do serviço) e a classe que vai utilizar os métodos definidos na interface (cliente do serviço).



Interfaces

- Uma interface pode ser vista como projeto puro enquanto que uma classe é um misto de projeto e implementação.
- Uma interface não contém atributos, apenas assinaturas de métodos.
- Constantes podem ser declaradas na interface para serem compartilhadas entre as classes do sistema.
- Todos os métodos da interface são públicos.
- Uma interface em Java só pode ser pública ou default (*friendly*).



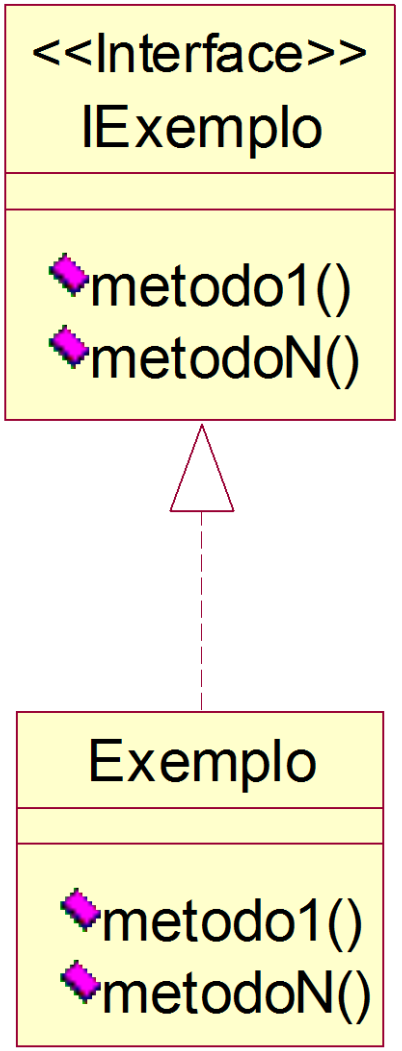
Interfaces

- Pode-se ter atributos, variáveis ou parâmetros de métodos cujo tipo é uma interface.
- Não se pode instanciar objetos de interfaces, mas sim de classes que implementam as interfaces.
- Serve como base para o desenvolvimento em camadas (dados, negócio, interface).

Exemplos

- A palavra reservada *implements* é utilizada para expressar este conceito.

- Exemplo 1:



Exemplos

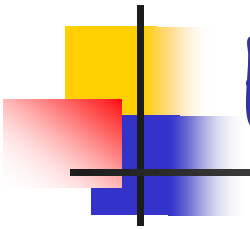
- Exemplo 1:

```
public interface IExemplo {
        public void metodo1();
        public void metodoN();
}
```

```
public class Exemplo implements IExemplo
{

 }
```

Erro de compilação porque a classe Exemplo não implementa todos os métodos definidos na interface IExemplo.

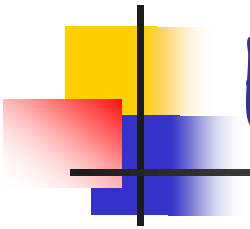


Exemplos

- Exemplo 1: Corrigindo...

```
public class Exemplo implements IExemplo {
        public void metodo1() {
                System.out.println("Executando metodo1( )...");
        }

        public void metodoN() {
                System.out.println("Executando metodoN( )...");
        }
}
```



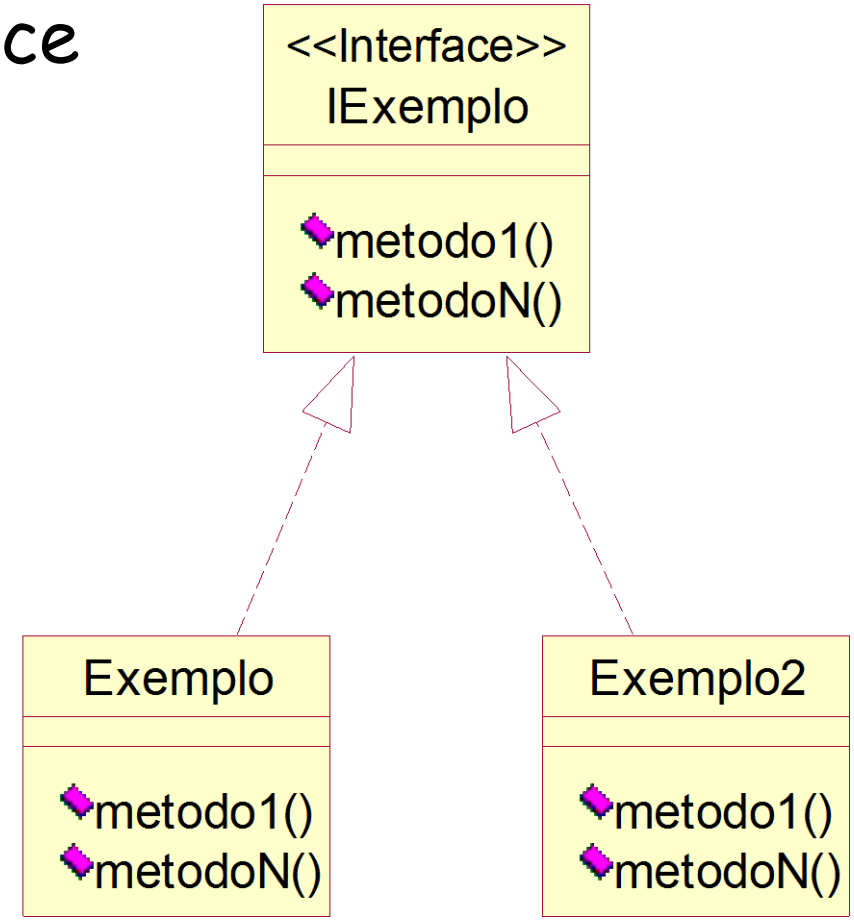
Exemplos

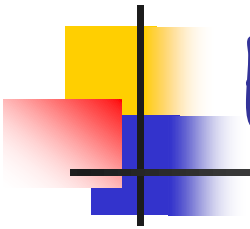
- Exemplo 1: Aplicação

```
public class AplicacaoIExemplo {
        public static void main(String[ ] args) {
                IExemplo ex = new Exemplo( );
                ex.metodo1( );
                ex.metodoN( );
        }
}
```

Exemplos

- Exemplo 1: Adicionando novas implementações da interface



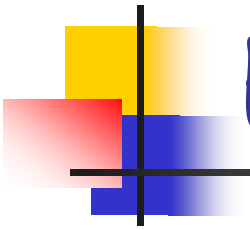


Exemplos

- Exemplo 1: Nova implementação da interface.

```
public class Exemplo2 implements IExemplo {
    public void metodo1() {
        System.out.println("metodo1( ) de Exemplo2...");
    }

    public void metodoN() {
        System.out.println("metodoN( ) de Exemplo2...");
    }
}
```

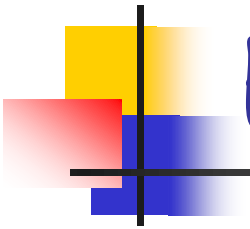


Exemplos

- Exemplo 1: Alterando a aplicação para usar a nova implementação da interface:

```
public class AplicacaoIExemplo {
        public static void main(String[ ] args) {
                System.out.println("Uso de Interface:");
                IExemplo ex = new Exemplo2( );
                ex.metodo1( );
                ex.metodoN( );
        }
}
```

Conclusões: Polimorfismo;
Vários objetos tratados da mesma forma.



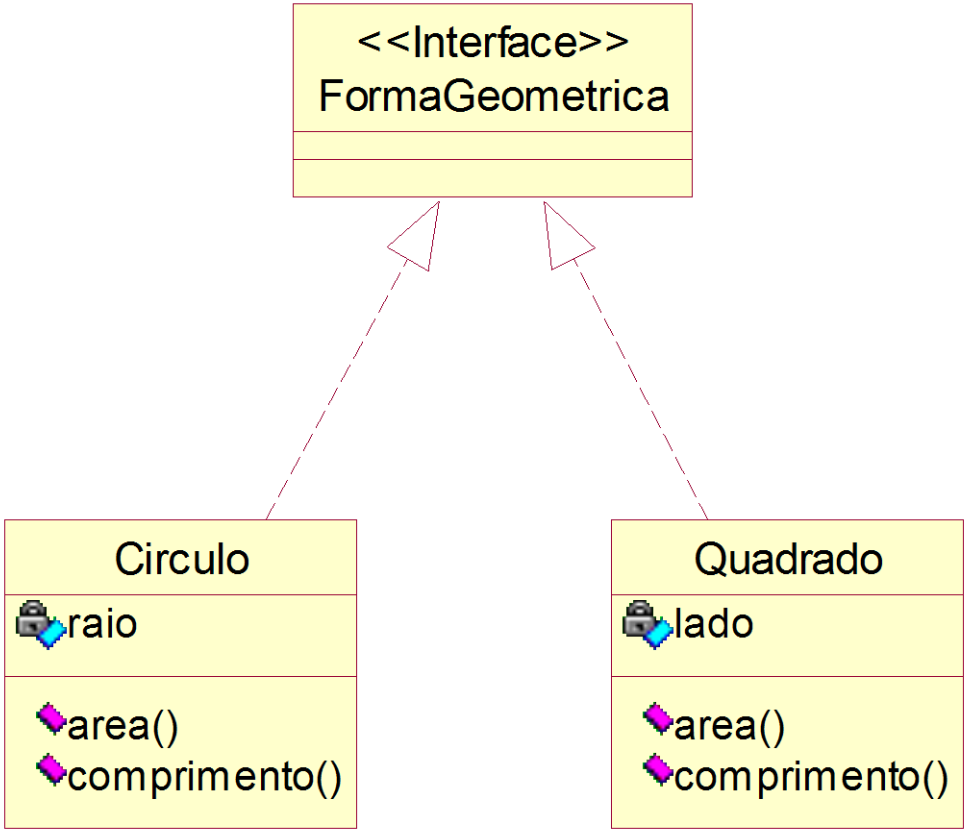
Exemplos

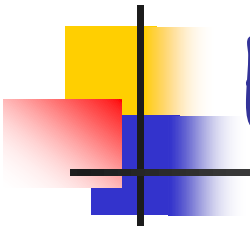
- Exemplo 2: Definição de interface com constante.

```
public interface IExemplo2 {
        public void metodoA();
        public void metodoB();
        public static final int VALOR = 10;
}
```

Exercícios

- Exercício 1: Forma Geométrica



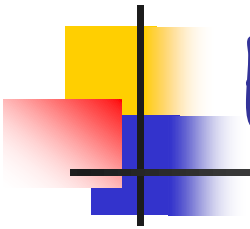


Exercícios

- Exercício 1: Interface Forma Geométrica

```
interface FormaGeometrica {
    public double area( );

    public double comprimento( );
}
```



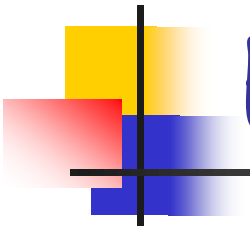
Exercícios

- Exercício 1: Classe Círculo

```
public class Circulo implements FormaGeometrica {
        private double raio;
        private static final double PI = 3.14;

        // Implementar construtor e métodos get e set

        public double area( ) {
                return (PI * raio * raio);
        }
        public double comprimento( ) {
                return (2 * PI * raio);
        }
}
```

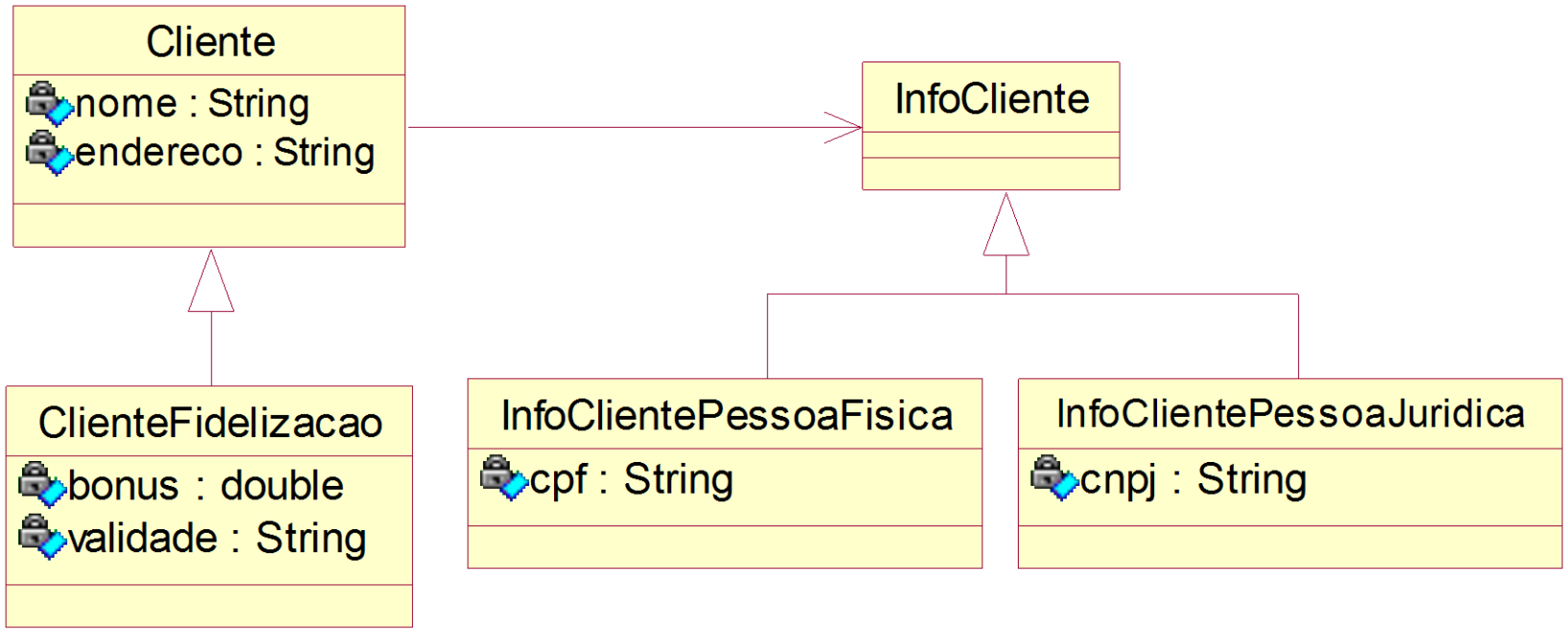


Exercícios

- Exercício 1:
  - Construir a classe Quadrado que implementa a interface FormaGeometrica;
  - Implementar programa de testes que declara duas variáveis do tipo FormaGeometrica e instancia dois objetos um do tipo Circulo e outro do tipo Quadrado. Exibir a área e o comprimento dos objetos instanciados.

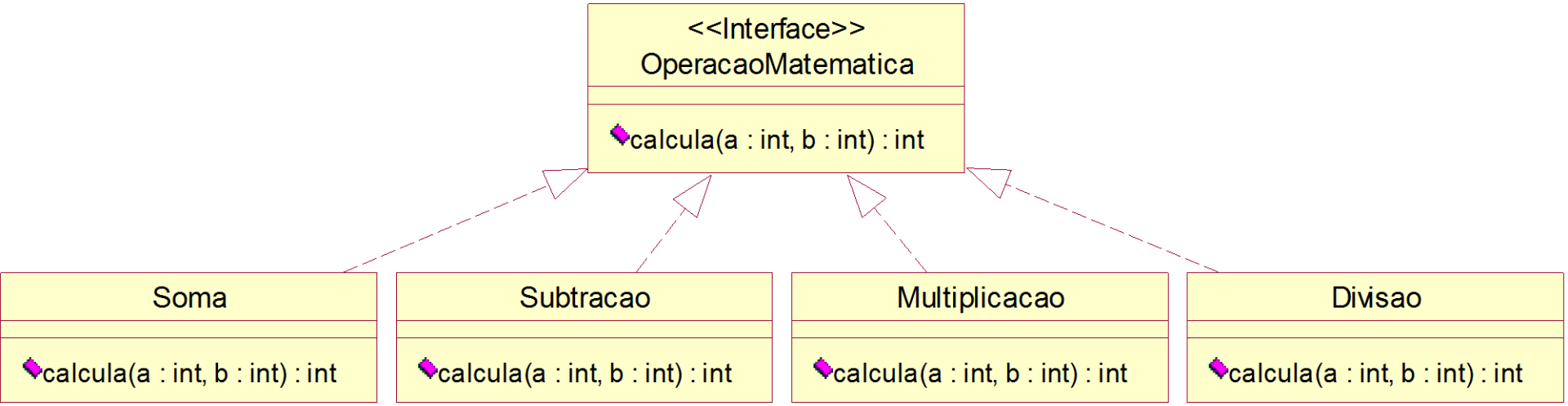
Exercícios

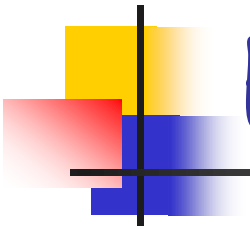
- Exercício 2:
  - A classe InfoCliente do modelo de clientes abaixo poderia se tornar uma interface?



Exercícios

- Exercício 3:



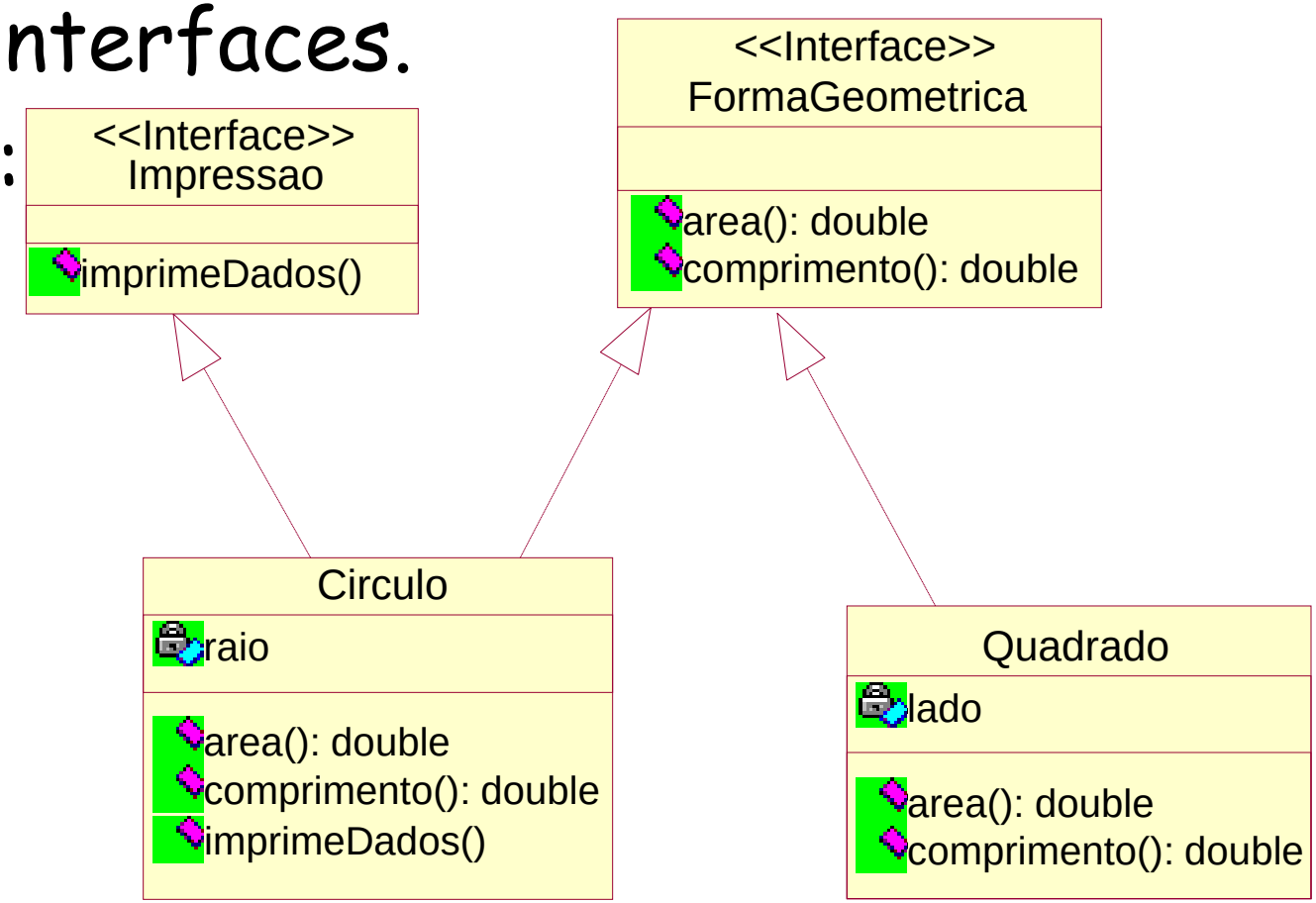


Exercícios

- Exercício 3:
  - Implementar uma aplicação que declara uma variável polimórfica do tipo OperacaoMatematica.
  - A partir de dados fornecidos pelo usuário, a aplicação deve realizar uma operação matemática e imprimir o seu resultado.
  - Oferecer para o usuário um menu para a escolha entre as operações matemáticas disponíveis.
  - OBS 1: Não defina a e b como atributos.
  - OBS 2: Implemente um construtor padrão para cada uma das classes.

Múltiplas Interfaces

- Java Permite que uma classe implemente múltiplas interfaces.
- Exemplo 1:



Múltiplas Interface

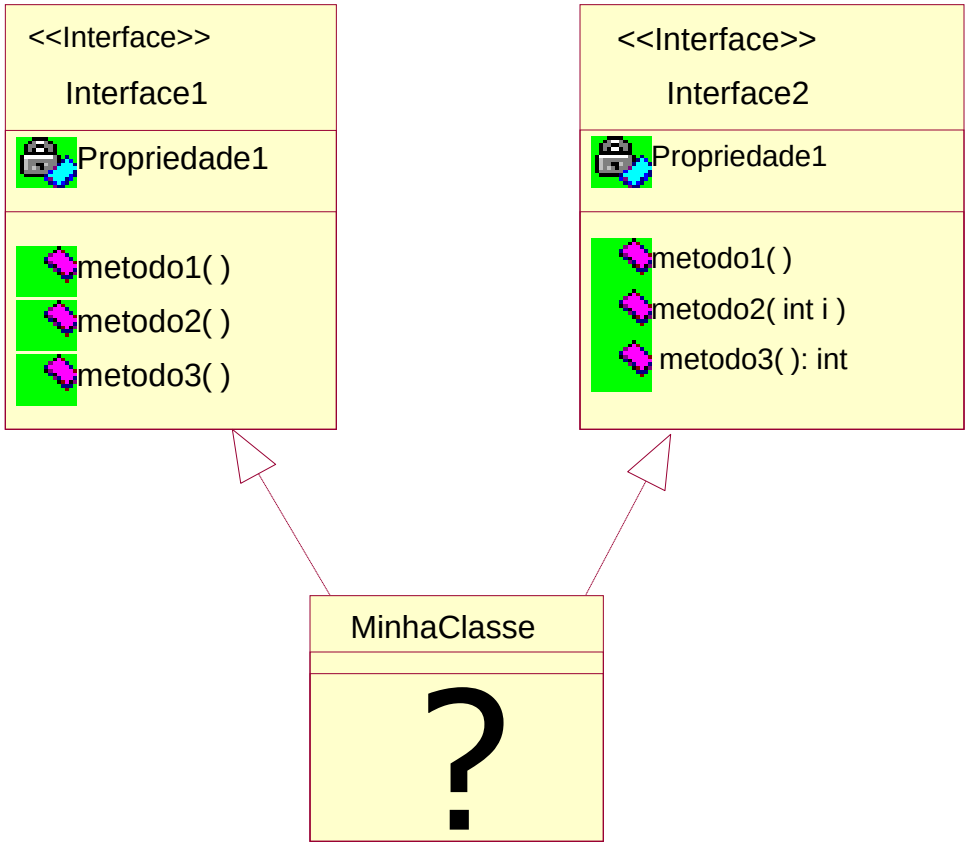
- Exemplo 1: Múltiplas interfaces

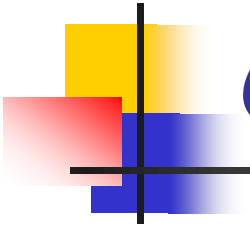
```
interface Impressao {
        public void imprimeDados( );
}

public class Circulo implements FormaGeometrica,
Impressao {
        ...
        public void imprimeDados( ) {
                System.out.println("Raio = " + raio);
        }
}
```

Conflito de Nomes

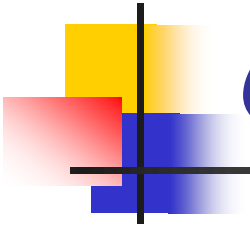
- Classe implementando múltiplas interfaces





Conflito de Nomes

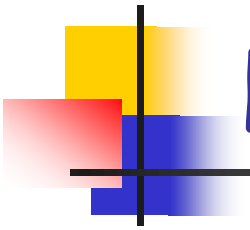
- O que acontece quando métodos com o mesmo nome aparecem nas duas interfaces?
  - Caso 1: Sobrecarga
  - Caso 2: A classe implementa um único método que atende as duas interfaces.
  - Caso 3: Métodos com a mesma assinatura, mas diferindo no tipo de retorno, a classe não poderá implementar as duas interfaces.



Conflito de Nomes

- O que acontece quando constantes com o mesmo nome aparecem nas duas interfaces?
  - Usar os nomes qualificados para acessar essas constantes:

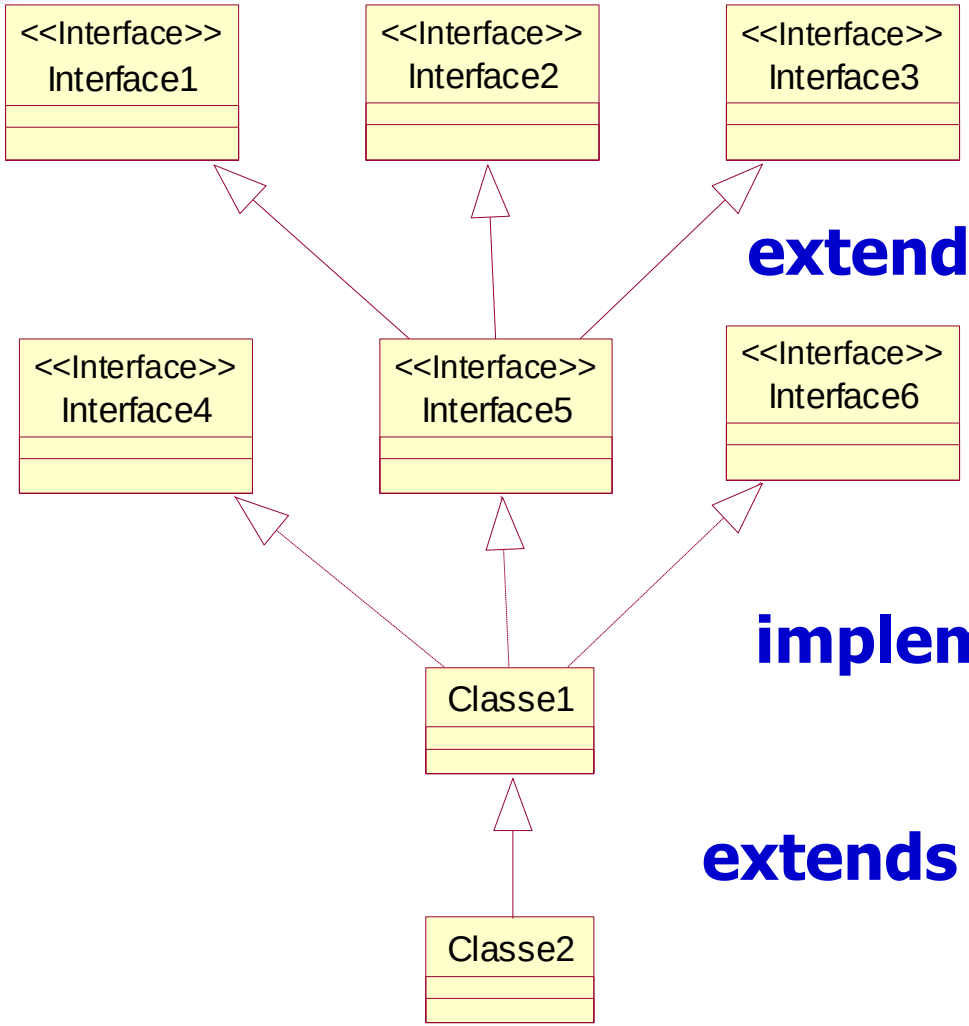
    NomeDaClasse.NOME_CONSTANTE



Herança Múltipla de Interface

- Java permite herança múltipla entre interfaces
  - O problema do conflito de nomes continua sendo resolvido da mesma forma como é resolvido entre classes e interfaces.

Hierarquia de Tipos



extends (herança e subtipo)

implements (subtipo)

extends (herança e subtipo)