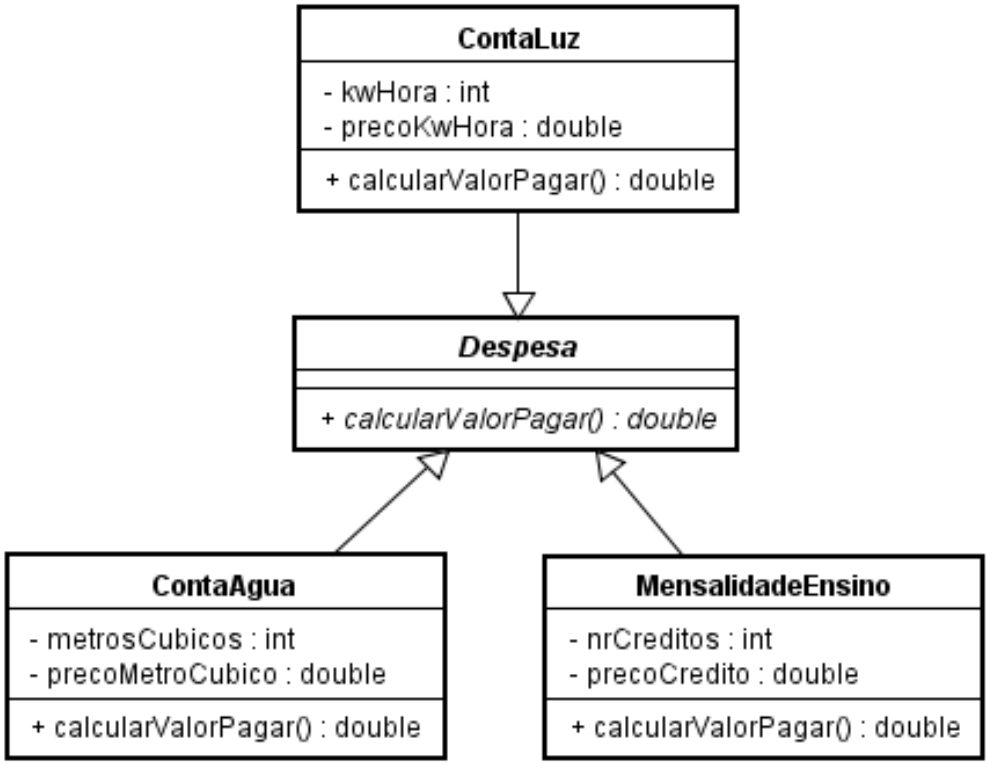
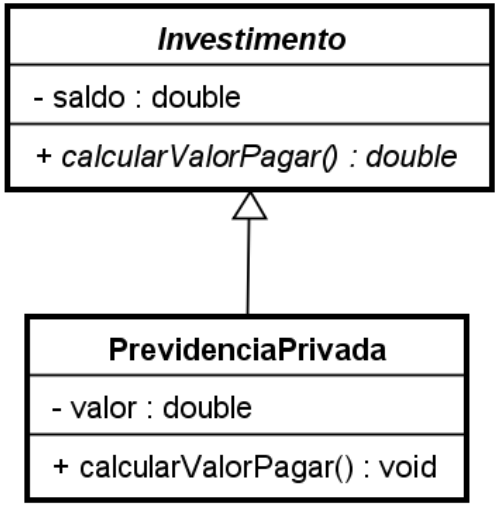
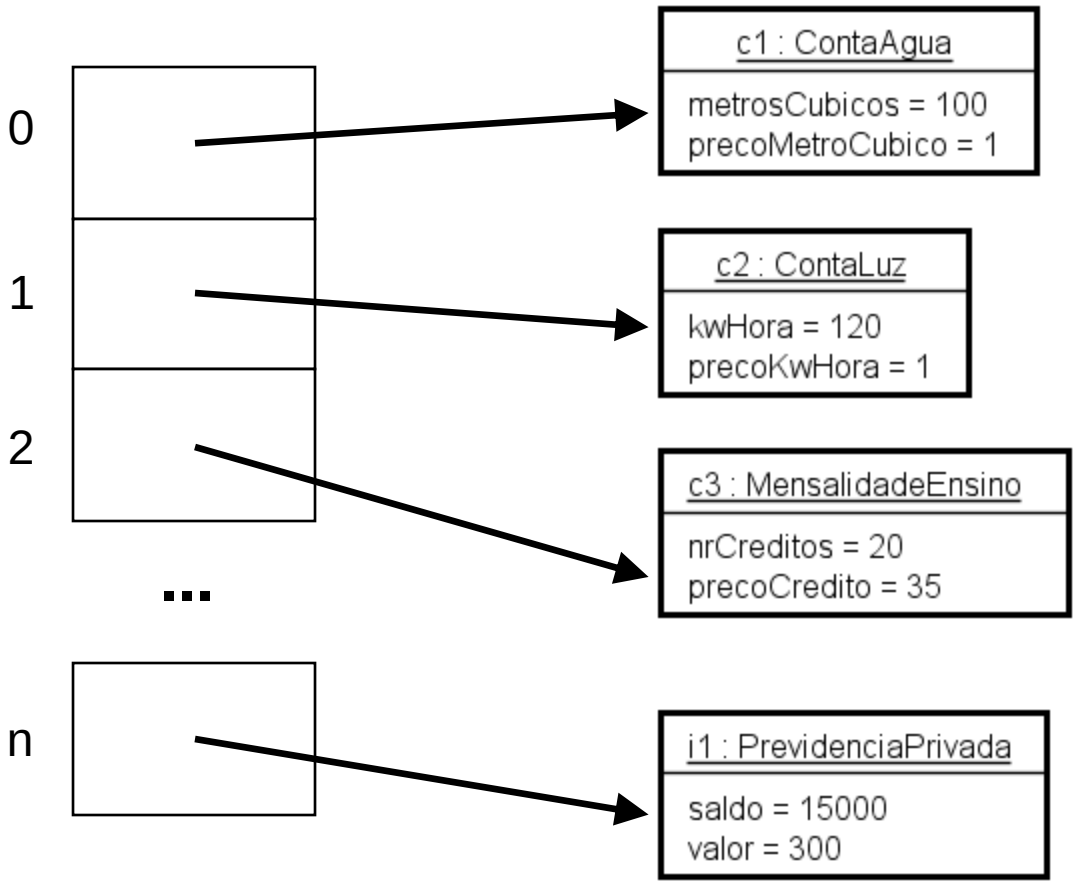
Interface

Motivação


ContaLuz
- kwHora : int
- precoKwHora : double
+ calcularValorPagar() : double
Despesa
+ calcularValorPagar() : double
ContaAgua
- metrosCubicos : int
- precoMetroCubico : double
+ calcularValorPagar() : double
MensalidadeEnsino
- nrCreditos : int
- precoCredito : double
+ calcularValorPagar() : double


Investimento
- saldo : double
+ calcularValorPagar() : double
PrevidenciaPrivada
- valor : double
+ calcularValorPagar() : void

Motivação


0
1
2
...
n
c1 : ContaAgua
metrosCubicos = 100
precoMetroCubico = 1
c2 : ContaLuz
kwHora = 120
precoKwHora = 1
c3 : MensalidadeEnsino
nrCreditos = 20
precoCredito = 35
i1 : PrevidenciaPrivada
saldo = 15000
valor = 300

Motivação

```
Object[] pagamentos = new Object[4];

pagamentos[0] = new ContaAgua();
pagamentos[1] = new ContaLuz();
pagamentos[2] = new PrevidenciaPrivada();
pagamentos[3] = new MensalidadeEstudo();
...
double totalPagamentos = 0;

for (Object o : pagamentos) {
    if (o instanceof Despesa) {
        totalPagamentos += ((Despesa) o).calcularValorPagar();
    } else if (o instanceof Investimento) {
        totalPagamentos += ((Investimento) o).calcularValorPagar();
    }
}

System.out.println("Total de pagamentos = " + totalPagamentos);
```

Interface

- Interfaces permitem especificar um conjunto de comportamentos desejáveis em objetos. Pode ser aplicado à objetos de classes distintas (não relacionadas)

- A interface contém um conjunto de definições de métodos e constantes.
  - A *Interface* não contém implementação de métodos
  - Também não contém variáveis

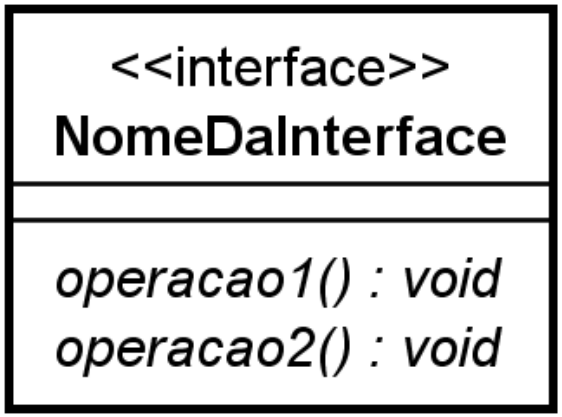
Interface

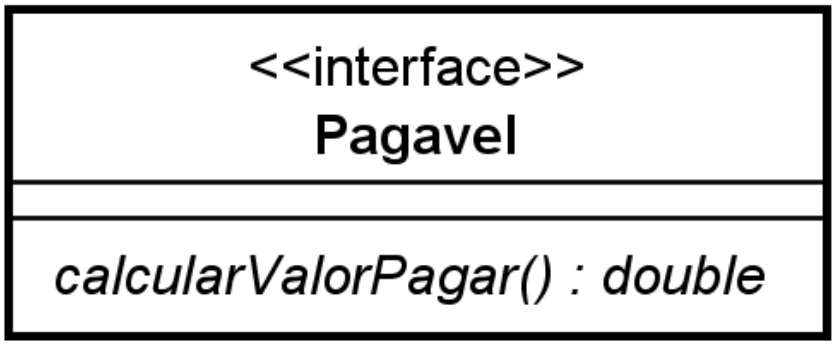
- Classes podem implementar uma ou mais interfaces

- A classe que implementa uma interface concorda em implementar todos os métodos definidos pela interface, portanto, concorda em desempenhar determinado(s) comportamento(s).

Em UML

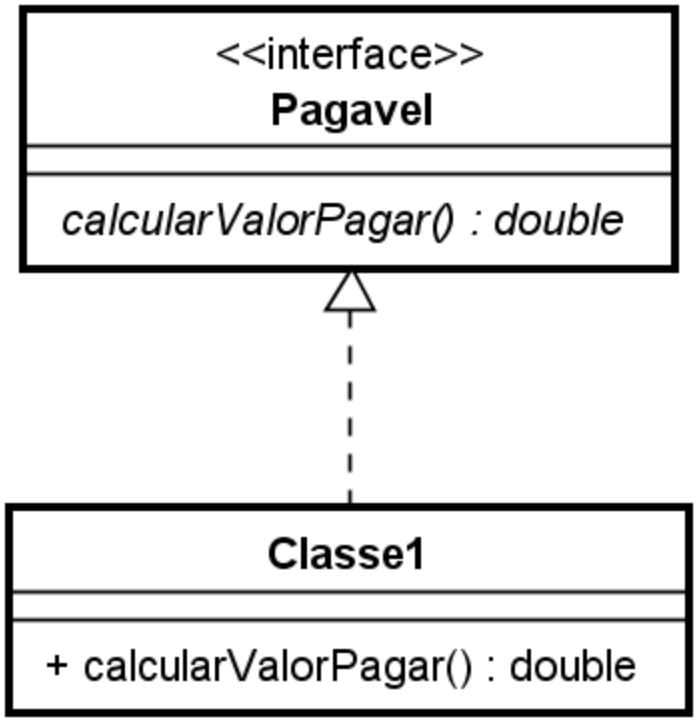
- Uma interface pode ser representada como uma classe estereotipada.



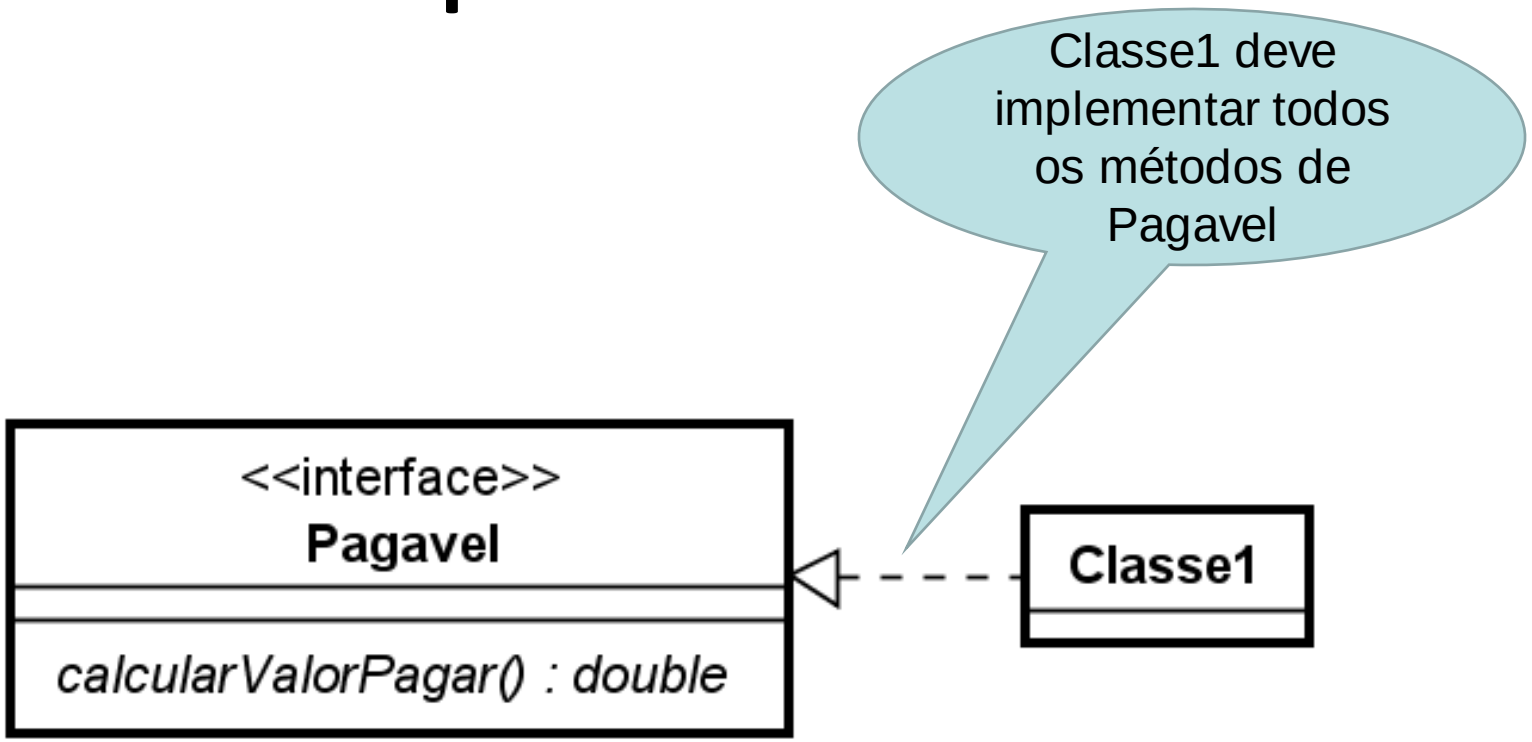


Exemplo

Para indicar que uma classe desempenha o(s) comportamento(s) de uma interface, utilizamos o relacionamento de “realização”.



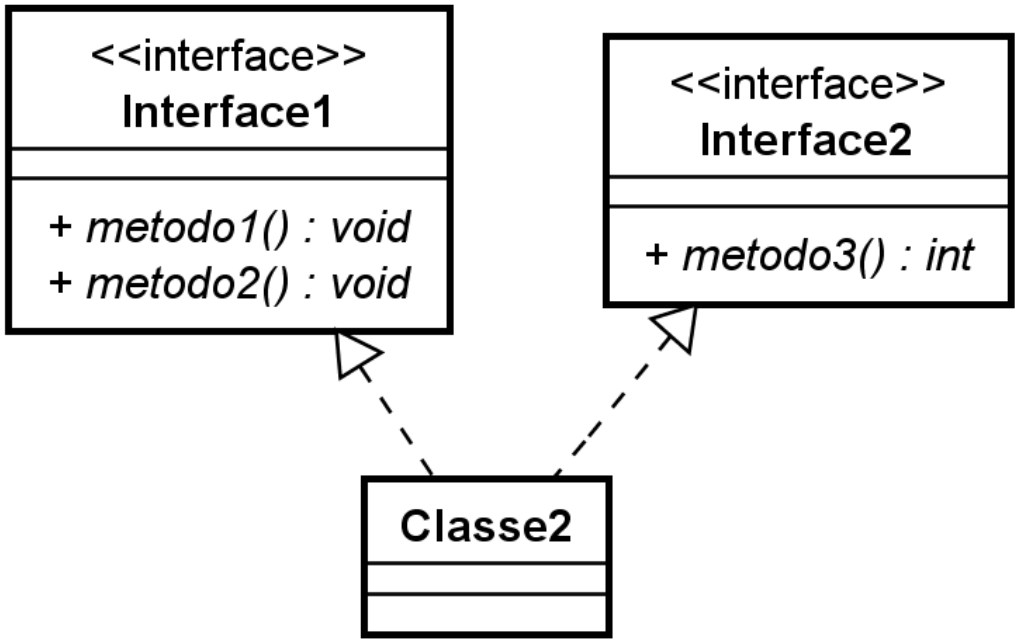
Exemplo



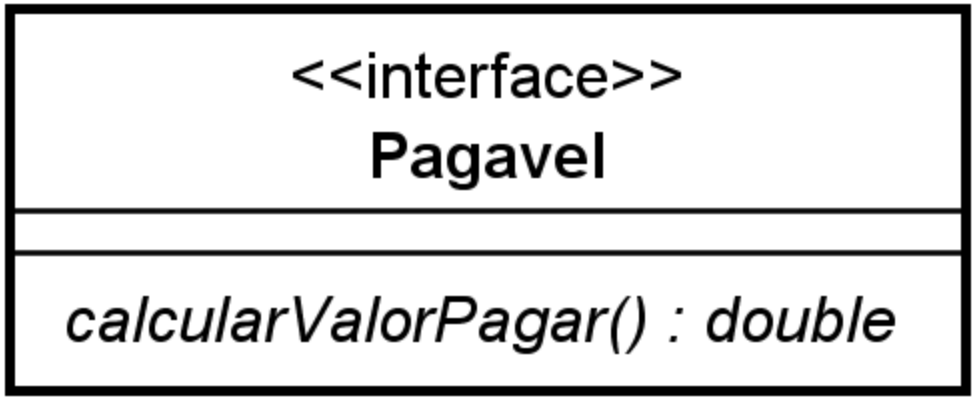
Afirmamos que:
Classe1 implementa a interface Pagavel ou
Classe1 realiza a interface Pagavel

Múltiplas Interfaces

- Em Java, uma classe pode implementar múltiplas interfaces



Tradução para a linguagem Java




```
public interface Pagavel {

    double calcularValorPagar();

}
```

Tradução para a linguagem Java

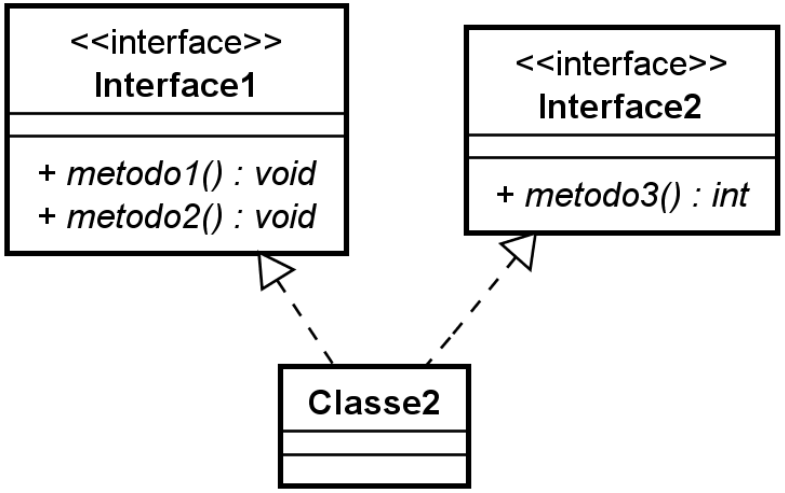
- Utilizamos a palavra reservada **implements** para anunciar as interfaces que a classe realiza.

```
public class Classe2 implements Pagavel {

    @Override
    public double calcularValorPagar() {
        ...
    }

}
```

Tradução para a linguagem Java




```
public class Classe2 implements Interface1, Interface2 {

    ...
    public void metodo1() {...}
    public void metodo2() {...}
    public int metodo3() {...}

}
```

Polimorfismo X Interface

- O conceito de polimorfismo também pode ser aplicado quando tratamos de objetos de classes que implementam interfaces.

- Podemos criar uma variável polimórfica definindo que seu tipo é uma *interface.*

Polimorfismo X Interface

- Exemplo:

```
Pagavel[] pagamentos = new Pagavel[4];

pagamentos[0] = new ContaAgua();
pagamentos[1] = new ContaLuz();
pagamentos[2] = new PrevidenciaPrivada();
pagamentos[3] = new MensalidadeEstudo();

double totalPagamentos = 0;

for (Pagavel p : pagamentos) {
    totalPagamentos += p.calcularValorPagar();
}

System.out.println("Total de pagamentos = " + totalPagamentos);
```

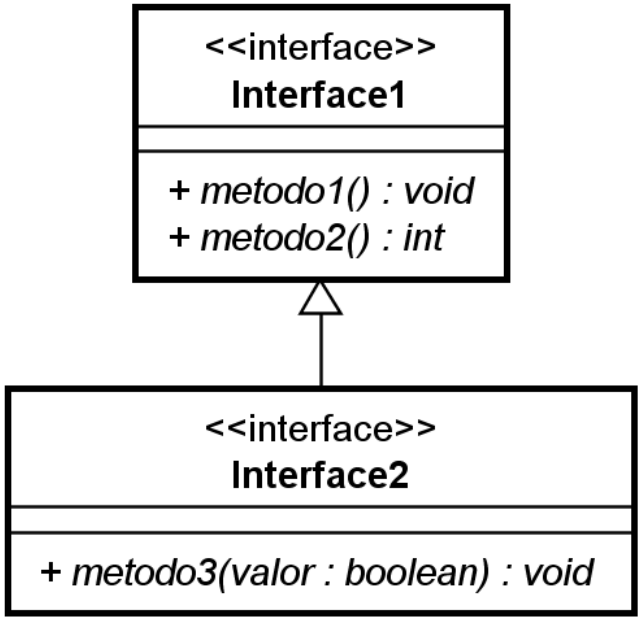
Observações

- Os métodos declarados numa interface não precisam utilizar modificador de acesso.
  - O único que é admissível é *public*.

- Ao indicar que uma classe abstrata implementa uma interface, a classe abstrata não precisa implementar os métodos da interface, porém suas subclasses concretas devem implementar (subclasses diretas abstratas também não precisam).

Herança de interface

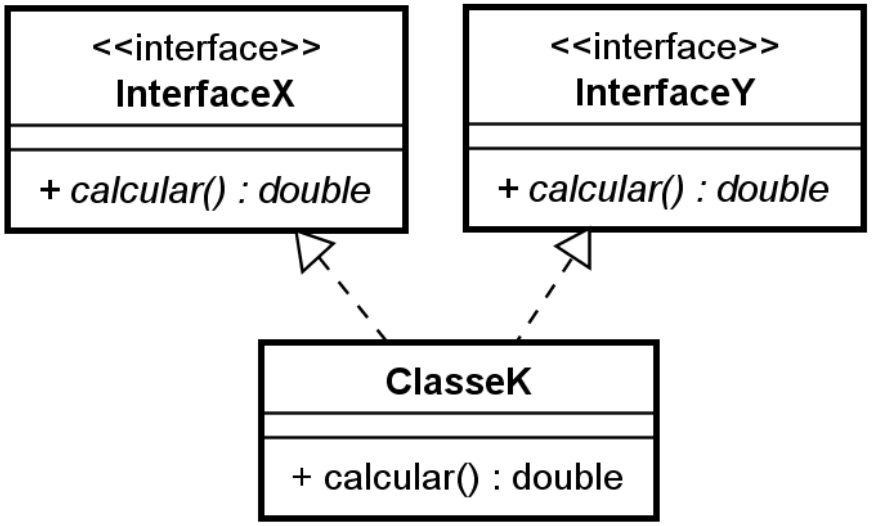
- Uma interface pode herdar a assinatura de métodos de outra interface



A classe que quiser implementar a interface *Interface2* deve possuir algoritmos para os métodos:

- *metodo1()*
- *metodo2()*
- *metodo3()*

Herança de interface




```
public class ClasseK implements InterfaceX, InterfaceY {

    public double calcular() {
        return 0;
    }

}
```

Interface no Java8

FURB
UNIVERSIDADE DE BLUMENAU

Interface no Java 8

- Incluir novos métodos em interfaces “antigas” requer implementar estes métodos novos;
- Com Java 8 é possível definir uma “implementação padrão” para um método.

```
public interface Interface1 {

    void metodo1(String str);

    default void log(String str) {
        System.out.println("Log:"+str);
    }

}
```

- Como interface não definem variáveis, métodos com implementação limitam-se a utilizar dados originados de parâmetros