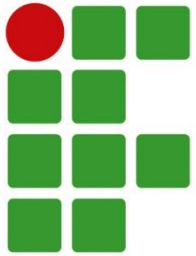


INSTITUTO FEDERAL
Goiás

Instituto Federal de Goiás
Câmpus Goiânia
Bacharelado em Sistemas de Informação
Disciplina: Programação Orientada a Objetos I

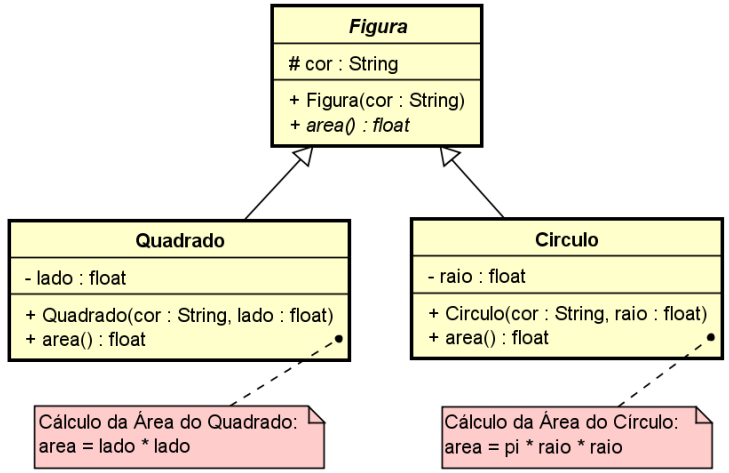
Classes Abstratas e Interfaces

Prof. Ms. Renan Rodrigues de Oliveira
Goiânia - GO

Classes Abstratas

Às vezes é útil declarar classes - chamadas classes abstratas - para as quais você nunca pretende criar objetos.

- Essas classes não podem ser usadas para instanciar objetos porque são classes incompletas.
- O propósito de uma classe abstrata é fornecer uma superclasse apropriada a partir da qual outras classes podem herdar e assim podem compartilhar um design comum.
- Na figura ao lado, as subclasses herdam a noção do que seria uma forma.
- As subclasses devem declaras as “partes ausentes” para tornaram-se classes “concretas”.
- As classes que podem ser utilizadas para instanciar objetos são chamadas de classes concretas.

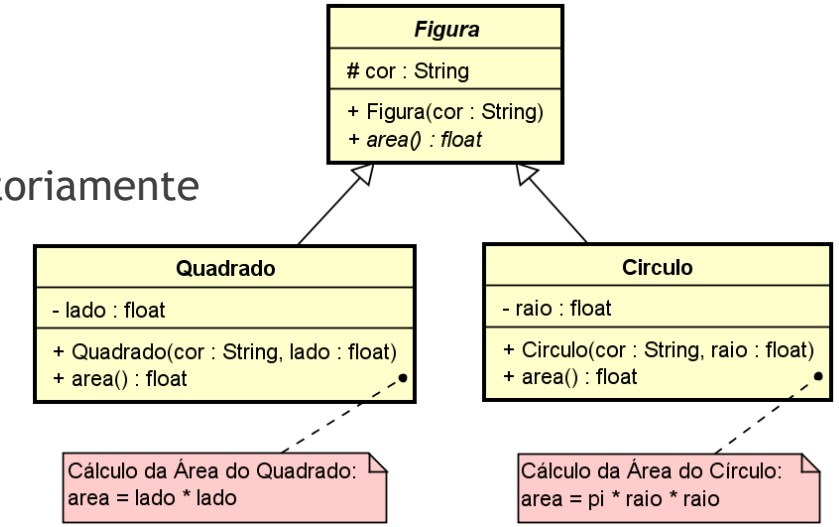


Classes Abstratas

Uma classe abstrata declara atributos e comportamentos comuns (ambos abstratos e concretos) das várias classes em um hierarquia de classes.

O principal uso é servir de base para concepção de classes.

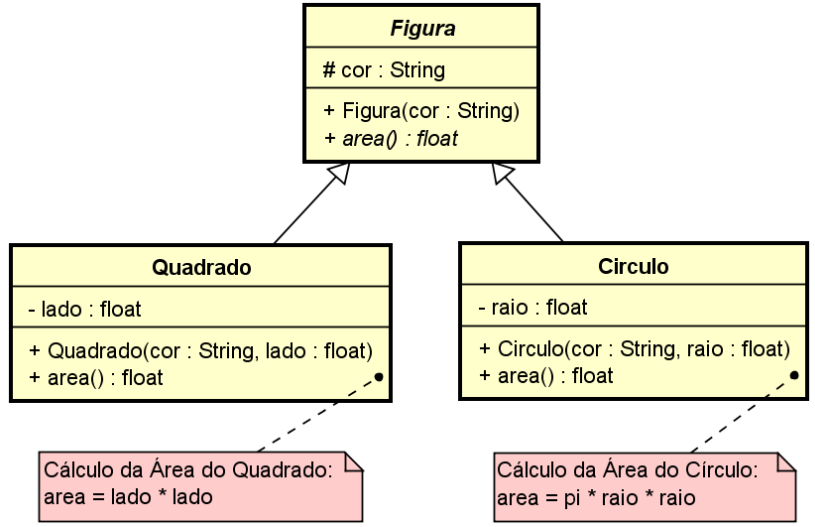
- Contém um ou mais métodos abstratos.
  - Métodos que não possuem implementação;
  - Tem por objetivo indicar que devem ser obrigatoriamente sobrescrito em um subclasse concreta.
- Também pode conter métodos concretos.
  - Métodos que possuem implementação;
  - Não necessitam ser sobrescritos nas subclasses.
- Não é possível instanciar objetos de uma classe abstrata.
  - Uma subclasse que não prover implementações para os métodos abstratos herdados, deve obrigatoriamente ser abstrata.



Classes Abstratas

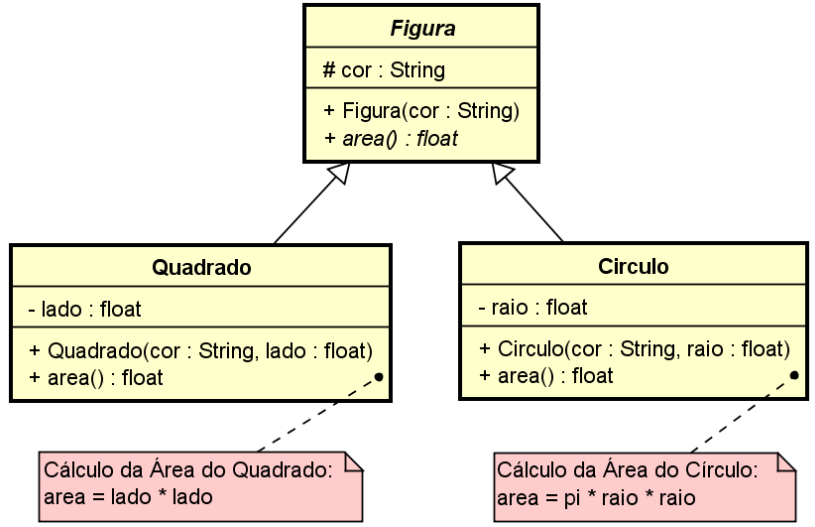
Implementando a Classe Figura

```java
package br.com.renanrodrigues.classeAbstrata;

public abstract class Figura {

    protected String cor;

    public Figura(String cor) {
        this.cor = cor;
    }

    abstract public float area();

    public String getCor() {
        return cor;
    }

    public void setCor(String cor) {
        this.cor = cor;
    }
}
```



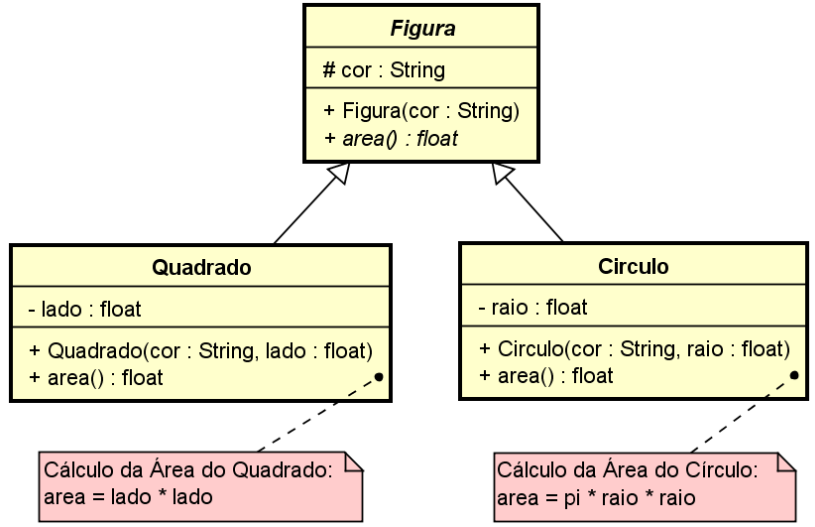
Classes Abstratas

Implementando a Classe Quadrado




```java
package br.com.renanrodrigues.classeAbstrata;

public class Quadrado extends Figura {
    private float lado;

    public Quadrado(String cor, float lado) {
        super(cor);
        this.lado = lado;
    }

    @Override
    public float area() {
        return lado * lado;
    }

    public float getLado() {
        return lado;
    }

    public void setLado(float lado) {
        this.lado = lado;
    }

    @Override
    public String toString() {
        return "Quadrado [lado=" + lado + ", cor=" + cor + "]";
    }
}
```

Classes Abstratas
Implementando a Classe Circulo

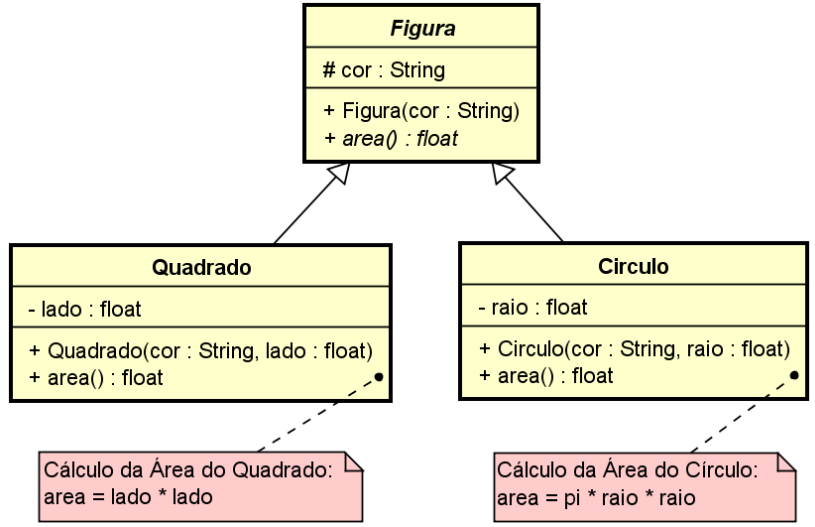



```java
package br.com.renanrodrigues.classeAbstrata;

public class Circulo extends Figura {
    private float raio;

    public Circulo(String cor, float raio) {
        super(cor);
        this.raio = raio;
    }

    @Override
    public float area() {
        return (float) Math.PI * raio * raio;
    }

    public float getRaio() {
        return raio;
    }

    public void setRaio(float raio) {
        this.raio = raio;
    }

    @Override
    public String toString() {
        return "Circulo [raio=" + raio + ", cor=" + cor + "]";
    }
}
```

Classes Abstratas

Implementando a Classe Circulo

```java
package br.com.renanrodrigues.classeAbstrata;

public class TesteFigura {

    public static void main(String[] args) {

        //Figura f = new Figura("Amarelo");

        Quadrado q = new Quadrado("Verde", 5f);
        System.out.println(q.area());

        Circulo c = new Circulo("Azul", 3f);
        System.out.println(c.area());

    }
}
```



Saída do Programa

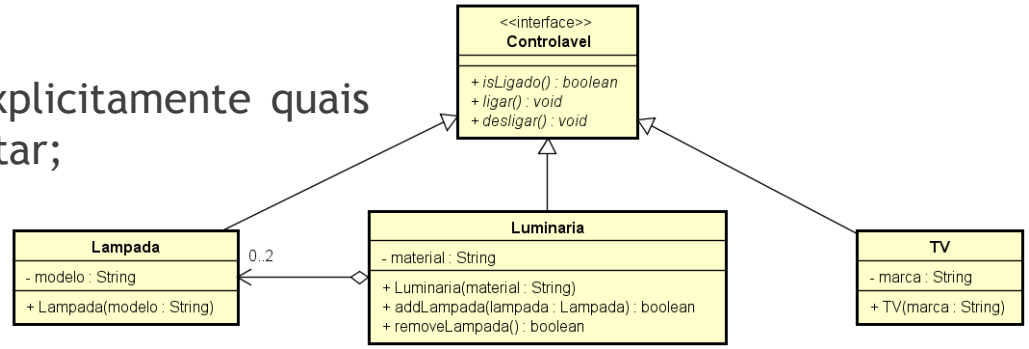
```
25.0
28.274334
```

Interfaces

Uma interface costuma ser utilizada no lugar de uma classes abstrata quando não há nenhuma implementação padrão a herdar
(não há nenhum atributo e nenhuma implementação padrão de método).

Uma classe abstrata, além de definir métodos abstratos, pode implementar alguns métodos. Uma interface define apenas um conjunto de métodos abstratos.

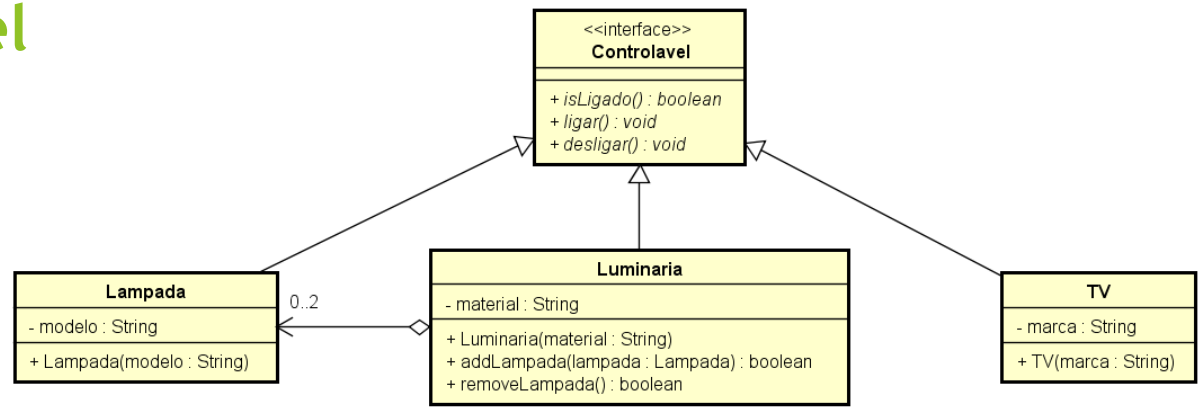
- Trazem a especificação do conjunto de operações públicas sem código de implementação;
- Ao contrário das classes, define um novo tipo sem fornecer a implementação;
- A interface age como um contrato, o qual define explicitamente quais métodos uma classe deve obrigatoriamente implementar;
- Uma interface deve ser implementada por uma classe;
- Uma interface não pode ser instanciada;
- Uma interface pode estender, via herança, outra interface.



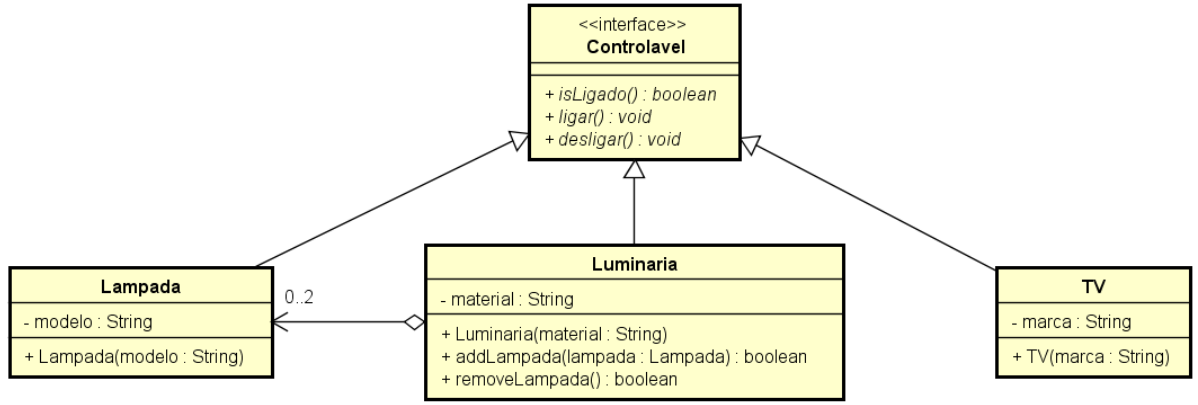
Interfaces

Implementando a Interface Controlavel

```java
package br.com.renanrodrigues.interfaces;

public interface Controlavel {

    public boolean isLigado();
    public void ligar();
    public void desligar();

}
```

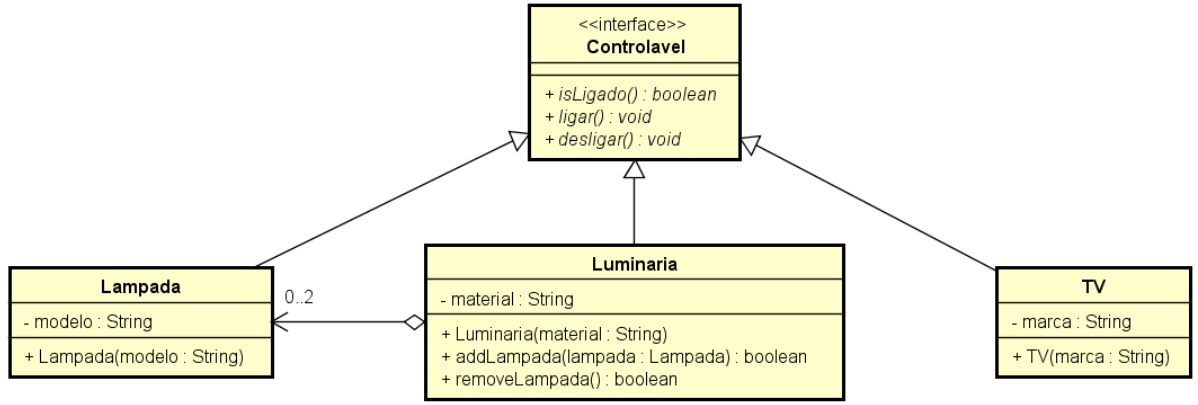


Interfaces
Implementando a Classe TV




```java
package br.com.renanrodrigues.interfaces;

public class TV implements Controlavel {

    private boolean estado;
    private String marca;

    public TV (String marca) {
        this.marca = marca;
    }

    public boolean isLigado() {
        return estado;
    }

    public void ligar() {
        estado = true;
    }

    public void desligar() {
        estado = false;
    }

    public String getMarca() {
        return marca;
    }

    public void setMarca(String marca) {
        this.marca = marca;
    }

    @Override
    public String toString() {
        return "TV [estado=" + estado + ", marca=" + marca + "]";
    }
}
```

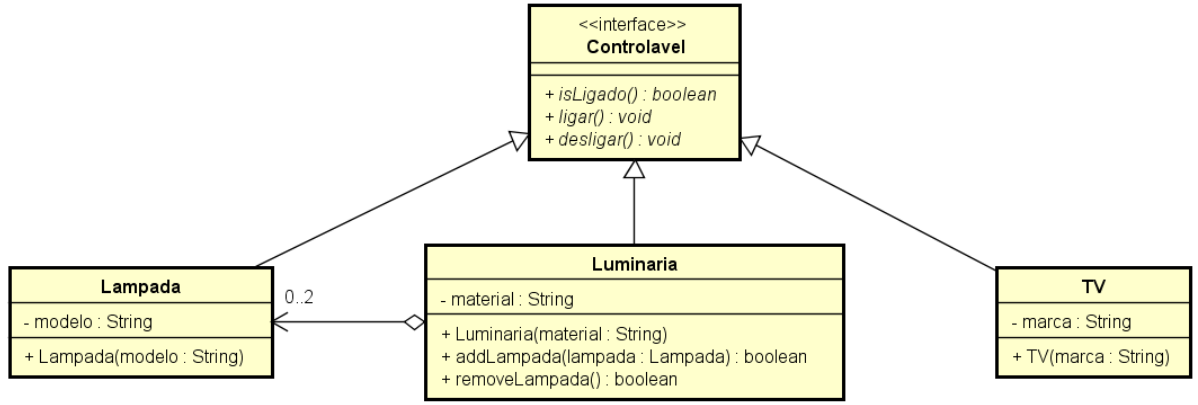
Interfaces
Implementando a Classe TV




```java
package br.com.renanrodrigues.interfaces;

public class TV implements Controlavel {

    private boolean estado;
    private String marca;

    public TV (String marca) {
        this.marca = marca;
    }

    public boolean isLigado() {
        return estado;
    }

    public void ligar() {
        estado = true;
    }

    public void desligar() {
        estado = false;
    }

    public String getMarca() {
        return marca;
    }

    public void setMarca(String marca) {
        this.marca = marca;
    }

    @Override
    public String toString() {
        return "TV [estado=" + estado + ", marca=" + marca + "]";
    }
}
```

Interfaces

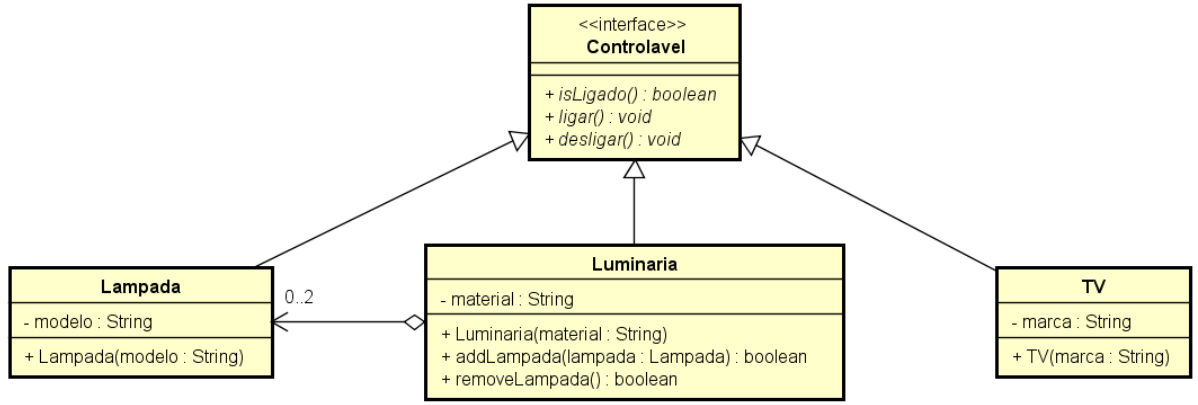
Implementando a Classe Lampada




```java
package br.com.renanrodrigues.interfaces;

public class Lampada implements Controlavel {

    private boolean estado;
    private String modelo;

    public Lampada (String modelo) {
        this.modelo = modelo;
    }

    public boolean isLigado() {
        return estado;
    }

    public void ligar() {
        estado = true;
    }

    public void desligar() {
        estado = false;
    }

    public String getModelo() {
        return modelo;
    }

    public void setModelo(String modelo) {
        this.modelo = modelo;
    }

    @Override
    public String toString() {
        return "Lampada [estado=" + estado + ", modelo=" + modelo + "]";
    }
}
```

Interfaces

Implementando a Classe Luminaria




```java
package br.com.renanrodrigues.interfaces;

public class Luminaria implements Controlavel {

    private String material;

    private Lampada lmpd1;
    private Lampada lmpd2;

    public Luminaria (String material) {
        this.material = material;
    }

    public boolean addLampada(Lampada lampada, int ordem) {
        boolean sucesso = false;

        if ( (ordem == 1) && (lmpd1 == null) && (lampada != lmpd2) ) {
            lmpd1 = lampada;
            sucesso = true;
        } else if ( (ordem == 2) && (lmpd2 == null) && (lampada != lmpd1) ) {
            lmpd2 = lampada;
            sucesso = true;
        }

        return sucesso;

    }
```

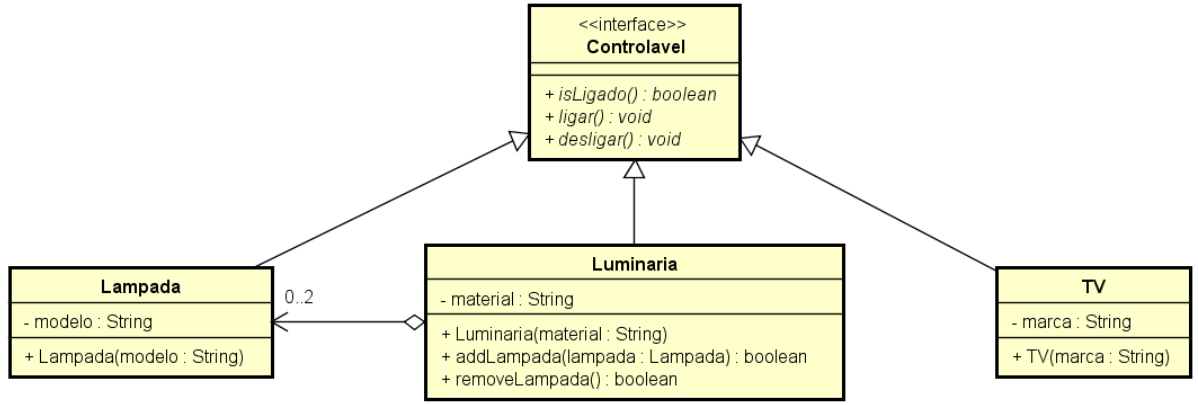
CONTINUA >>

Interfaces

Implementando a Classe Luminaria

```
33      public void ligar() {
34
35          if (lmpd1 != null) {
36              lmpd1.ligar();
37          }
38
39          if (lmpd2 != null) {
40              lmpd2.ligar();
41          }
42      }
43
44      public void desligar() {
45          if (lmpd1 != null) {
46              lmpd1.desligar();
47          }
48
49          if (lmpd2 != null) {
50              lmpd2.desligar();
51          }
52      }
```

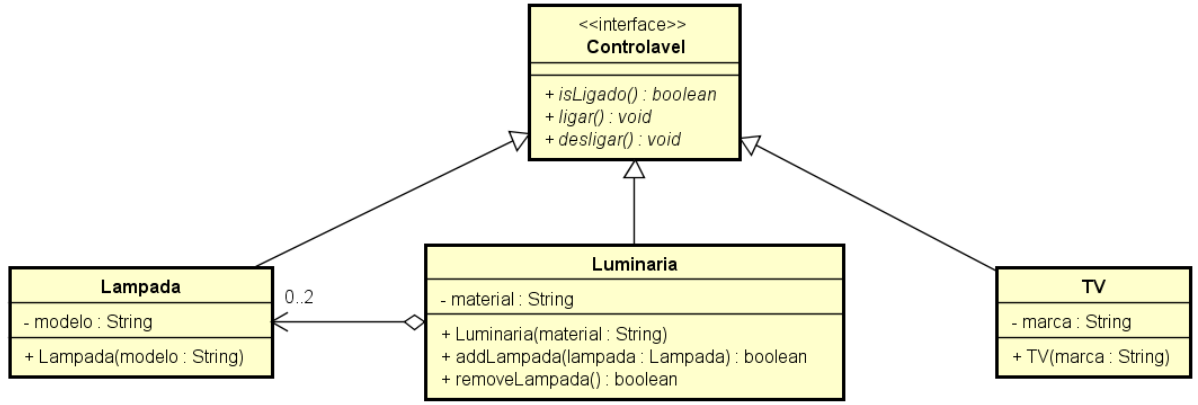
CONTINUA >>



Interfaces

Implementando a Classe Luminaria

```
54      public String getMaterial() {
55          return material;
56      }
57
58      public Lampada getLampada(int ordem) {
59          Lampada retorno;
60
61          switch (ordem) {
62              case 1: retorno = lmpd1; break;
63              case 2: retorno = lmpd2; break;
64              default: retorno = null;
65          }
66
67          return retorno;
68      }
69
70      @Override
71      public String toString() {
72          return "Luminaria [material=" + material + ", lmpd1=" + lmpd1
73                  + ", lmpd2=" + lmpd2 + "]";
74      }
75  }
```



Interfaces
Programa Principal

```java
package br.com.renanrodrigues.interfaces;

public class TesteControlavel {

    public static void main(String[] args) {

        TV tv = new TV("LG");
        System.out.println(tv);
        System.out.println(tv.isLigado());
        tv.ligar();
        System.out.println(tv);
        System.out.println(tv.isLigado());

        Lampada lmpd1 = new Lampada("Fluorescente");
        Lampada lmpd2 = new Lampada("LED");
        Luminaria luminaria = new Luminaria("Ferro");
        System.out.println(luminaria);
        System.out.println(luminaria.isLigado());
        luminaria.addLampada(lmpd1, 1);
        luminaria.addLampada(lmpd2, 2);
        System.out.println(luminaria);
        System.out.println(luminaria.isLigado());

        luminaria.ligar();
        System.out.println(luminaria);
        System.out.println(luminaria.isLigado());
    }
}
```

Saída do Programa

```
TV [estado=false, marca=LG]
false
TV [estado=true, marca=LG]
true
Luminaria [material=Ferro, lmpd1=null, lmpd2=null]
false
Luminaria [material=Ferro, lmpd1=Lampada [estado=false, modelo=Fluorescente], lmpd2=Lampada [estado=false, modelo=LED]]
false
Luminaria [material=Ferro, lmpd1=Lampada [estado=true, modelo=Fluorescente], lmpd2=Lampada [estado=true, modelo=LED]]
true
```