INSTITUTO SUPERIOR POLITECNICO DE TECNOLOGIAS E CIENCIAS

1º Semestre

2025/26

- Classes abstractas
- Interfaces
- Classes abstractas vs Interfaces

Classes abstractas

- **Classe** em que pelo menos um dos métodos de instancia não é implementado.
  - Exemplo

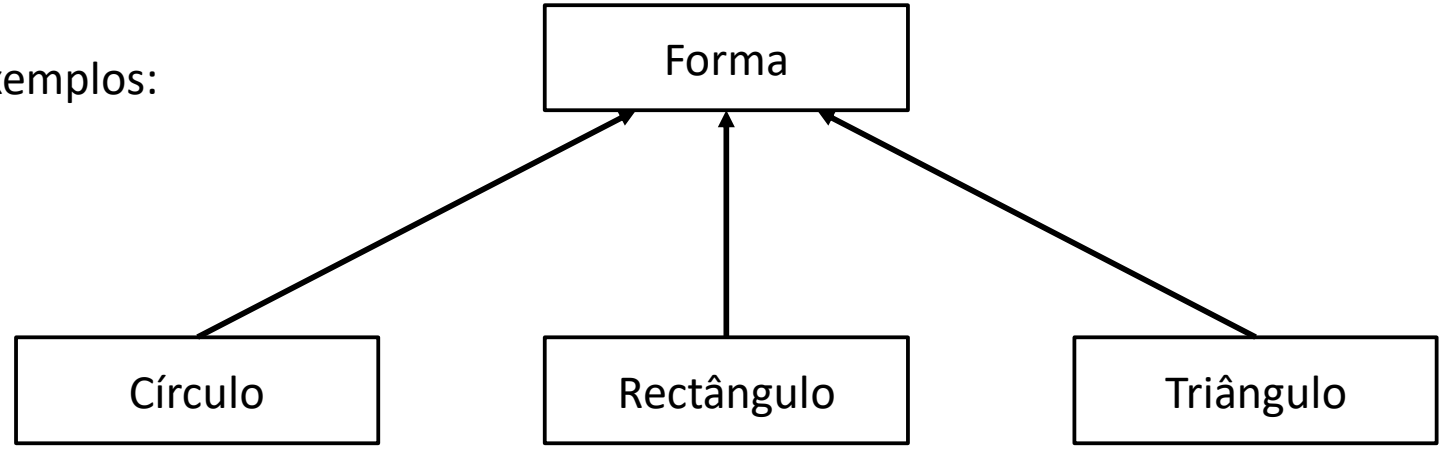
```
public abstract class Forma {

        public  abstract double area();

        public  abstract double perimetro();
}
```

- Não é possível criar instâncias de uma classe abstracta.
- Mecanismo de herança mantém-se.
- Uma subclasse de uma classe abstracta herda os seus métodos, podendo implementá-los ou não.
  - Se implementar todos os métodos passará a ser uma classe concreta.
  - Se deixar algum por implementar passará a ser também uma classe abstracta.

Classes abstractas
Para que servem?

- Definir **uma linguagem comum** a um conjunto de classes que herdam a classe abstracta.

- Exemplos:



- A classe Círculo, Rectângulo e Triângulo como são subclasses de uma classe **abstracta** para serem **concretas** terão de implementar obrigatoriamente todos os métodos da superclasse ou seja as três subclasses vão ter uma **linguagem comum**(métodos com a mesma assinatura mas com implementações diferentes).

- No futuro se surgir uma nova figura geométrica e herdar da classe **Forma** automaticamente falará a mesma língua que as restantes classes.

- Desta forma também se têm um controlo maior das subclasses criadas, uma vez que estas terão de respeitar regras impostas pela superclasse.

```
13  public class Rectangulo extends Forma {
14
15      private double comprimento;
16      private double largura;
17
18      public Rectangulo(double comprimento, double largura) {
19
20          this.comprimento = comprimento;
21          this.largura = largura;
22      }
23
24      @Override
25      public double area() {
26          return comprimento * largura;
27      }
28
29      @Override
30      public double perimetro() {
31          return 2 * (comprimento + largura);
32      }
33  }
```

- O rectângulo é que sabe como calcular a sua área e perímetro por isso faz todo sentido ser ele mesmo a implementar a sua logica para esses cálculos.

- Mas apesar disso a superclasse nunca perde o controlo sobre ela

- Variáveis não são abstractas
- Construtores não são abstractos
- Métodos privados não são abstractos
- Métodos de classe não são abstractos

- Classes abstractas
- Interfaces
- Classes abstractas vs Interfaces

Interfaces

- Em programação orientada aos objectos, uma **interface** é um tipo abstracto que é usado para especificar aquilo que as classes têm de obrigatoriamente implementar.

- As interfaces podem definir **apenas as assinaturas dos métodos** e **constantes** (definidas como **static** e **final**) e nunca a sua definição,

- Permite definir um comportamento comum a duas ou mais classes que não possuam qualquer relação hierárquica entre elas.

- Através de interfaces, consegue-se concretizar polimorfismo no programa.

```
public abstract interface IOrdem {

        public abstract boolean igual ();
        public abstract boolean maior ();
        public abstract boolean menor();
}
```

- Uma **interface** é (implícita) e obrigatoriamente abstracta (abstract)
- As **constantes** declaradas numa interface são implícita e obrigatoriamente:
  - **public static final**
- Os métodos declarados numa **interface** são (implícita) e obrigatoriamente públicos e abstractos
  - **public abstract**
- Uma classe que **implemente** uma dada interface têm obrigatoriamente que implementar todos os métodos declarados na interface

```
1  public interface IOrdem {
2
3      boolean igual();
4
5      boolean maior();
6
7      boolean menor();
8  }
9
10 public class Ordem implements IOrdem {
11
12     @Override
13     public boolean igual() {...}
14
15     @Override
16     public boolean maior() {...}
17
18     Override
19     public boolean menor() {...}
20 }
```

- Todas as classes que implementem a interface **IOrdem** têm em comum o definido em **IOrdem**

- As interfaces têm a sua própria hierquia.

- Exemplo:

```
1  public interface IAmovivel {
2
3      void movimento (double x, double y);
4  }
5
6
7  public interface IComMotor extends IAmovivel{
8
9      public static final int limiteVel = 120;
10     public String motor();
11
12 }
```

- Uma classe que implemente a interface IComMotor terá obrigatoriamente que implementar:
  - Todos os métodos da interface IComMotor
  - Todos os métodos de todas as superInterfaces

```java
public interface IAmovivel {

    void movimento (double x, double y);
}


public interface IComMotor extends IAmovivel{

    public static final int limiteVel = 120;
    public String motor();

}
```

```java
public class Veiculo implements IComMotor {

    ...
    public String motor() {...}

    public void movimento (double x, double y) {...}


}
```

- Uma interface pode ser sub-interface de várias interfaces
  - Permitindo o mecanismo de **herança múltipla**.

```java
public interface IAssociadoISPTEC {

    public int getIdentificador();
}

public class Aluno implements IAssociadoISPTEC{

    int numeroAluno = 2;

    @Override
    public int getIdentificador(){
        return this.numeroAluno;
    }
}

public class Professor implements IAssociadoISPTEC{

    int numeroDocente = 1;

    @Override
    public int getIdentificador(){
        return this.numeroDocente;
    }
}

public class TestaInterface {

    public static void main (String [] args){

        IAssociadoISPTEC [] associados = new IAssociadoISPTEC[2];
        associados [0] = new Aluno();
        associados [1] = new Professor();

        for(IAssociadoISPTEC a: associados)
            System.out.println(a.getIdentificador());

    }
}
```

- Classes abstractas
- Interfaces
- **Classes abstractas vs Interfaces**

Classes abstractas vs Interfaces

- Uma **classe abstracta** pode não ser 100% abstracta.
- Uma **interface** é sempre 100% abstracta.
- Uma **classe abstracta** não impõe às suas subclasses a implementação obrigatória dos seus métodos abstractos.
- Uma **interface** impõe a qualquer classe que declare implementá-la a implementação de todos os seus métodos abstractos.
- As **classes abstractas** pertencem à hierarquia de classes
- As **interfaces** têm a sua própria hierarquia.

Dúvidas