
Orientação a Objetos
Encapsulamento
Herança
Composição
Polimorfismo
Princípio da abstração

Introdução

- O Polimorfismo é uma técnica para lidar com a complexidade dos *softwares*.
- Ele nos permite escrever programas que trata uma ampla variedade de classes relacionadas existentes e ainda que serão especificadas.
- Ou seja, é possível projetar e implementar sistemas mais flexíveis e extensíveis.
- Trata as “várias formas” de um objeto.
- Utiliza a herança para tal tarefa.

Introdução

- Quando utilizamos a herança é possível criar objetos da seguinte forma:

1. Referência a um objeto superclasse com outra superclasse.
   - EX: Funcionario f = new Funcionario();
2. Referência a um objeto subclasse com outra subclasse.
   - Ex: Professor p = new Professor();
3. Referência a um objeto superclasse com outra subclasse.
   - Ex: Funcionario p = new Professor();

Relação superclasse - subclasse

- É possível e seguro fazer uma referência a um objeto da superclasse com uma referência da subclasse.
- Isso é possível porque o objeto da subclasse "**é um**" objeto de sua superclasse.
- Quando criamos um objetos com essa relação, tratamos o objeto criado como um objeto da superclasse, porém, se o código de um método for sobrescrito, o código executado será o da subclasse.
- Funcionario p = new Professor();

Polimorfismo


```java
public class Funcionario {

    private String nome;
    private double salario;

    public double ganhoAnual(){
        double ganho = this.salario * 12;
        return ganho;
    }

    public void exibeDados(){
        System.out.println("Nome: " + nome
                + " Salário: " +salario);
    }
```

```java
public class Tecnico extends Funcionario{

    private double bonus = 100;

    public double ganhoAnual(){
        double ganho = (super.getSalario()+bonus)*12;
        return ganho;
    }
```

Polimorfismo


```
public class TesteFuncionario {

    public static void main(String[] args) {
        Funcionario f = new Tecnico();
        f.setNome("Nickerson");
        f.setSalario(1000);
        f.exibeDados();
        System.out.println(f.ganhoAnual());

        Funcionario f2 = new Funcionario();
        f2.setSalario(1000);
        System.out.println(f2.ganhoAnual());
    }
}
```

Classes Abstratas

- Podem existir casos em que a classe se comporta como um **tipo**, logo, supomos que os objetos desse **tipo** não serão instanciados.
- Nesses casos, chamamos a classe de **classe Abstrata.**
- O único objetivo dessa classe é servir de superclasse para outras classes.
- As classes que herdam de classes abstratas são conhecidas como **classes concretas.**
- Temos uma característica exclusiva desse tipo de classe, **os métodos abstratos.**

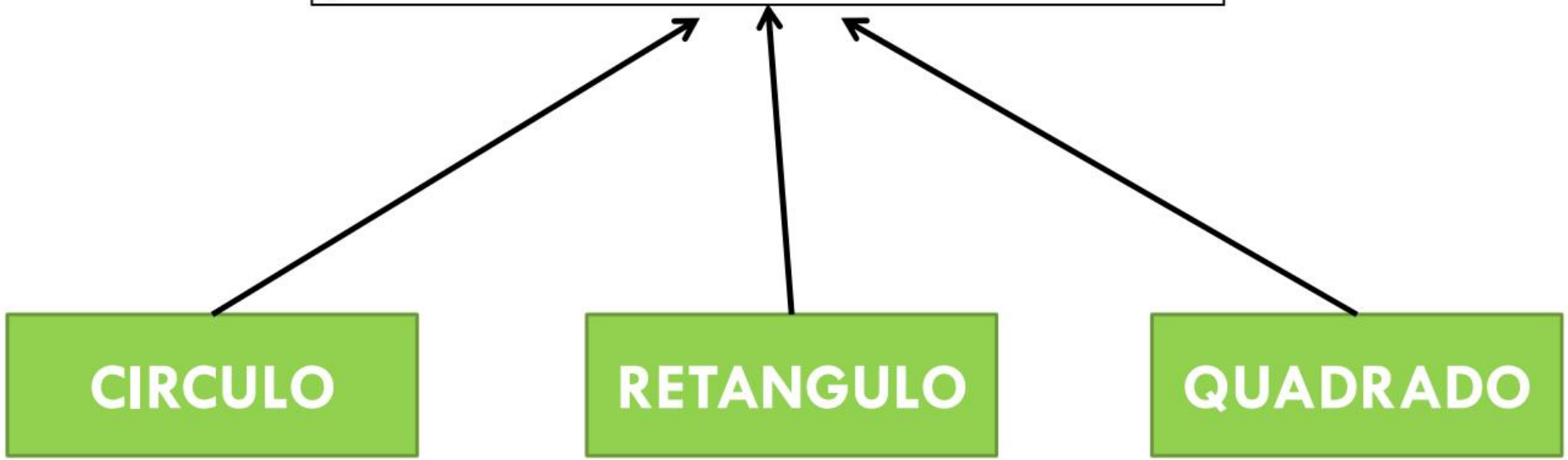
Classes Abstratas


```java
public abstract class Forma {

    public abstract void desenhar();

    public abstract void informacoes();

    public void teste(){
        System.out.println("Testando!!");
    }
}
```



Exercícios - POLIMORFISMO

- Criar uma superclasse chamada animal e as 3 seguintes subclasses: cachorro, cavalo e preguica. Segue as classes com seus respectivos atributos e métodos.
- Classe abstrata Animal possui um nome e uma idade e um método abstrato emitirSom
- Classe Cachorro herda de Animal e sobrescreve o método emitirSom.
- Classe Cavalo herda de Animal e sobrescreve o método emitirSom.
- Classe Preguica herda de Animal e sobrescreve o método emitirSom.
- Classe TesteAnimais que tem um vetor de Animal com 10 posições.
  - Coloque um Animal em cada posição
  - Depois percorra o vetor emitindo o som de cada Animal.

Referências

- **Apostila Caelum:** https://www.caelum.com.br/apostila-java-orientacao-objetos/orientacao-a-objetos-basica
- H.M. Deitel, P.J. Deitel, **Java Como programar.**