



chrome://welcome/Compartilhar o seu link com: luciorocha @ professores.utfpr.edu.br

Matheus H. A. Pereira: Cópia de Aula 14 - POCO4A - 27/09/2022 - Exercícios propostos

Leonardo G. Fagote Cópia de Aula 14 - POCO4A - 27/09/2022 - Exercícios propostos

Alexandre Calisto: Aula 14

Guilherme Conceição Ramalho:

Cópia de Aula 14 - POCO4A - 27/09/2022 - Exercícios propostos

Gabriel Candelária: Cópia de Aula 14 - POCO4A - 27/09/2022 - Exercícios propostos

Camila Costa: Cópia de Aula 14 - POCO4A - 27/09/2022 - Exercícios propostos

Andrei Fernandes Zani: Cópia de Aula 14 - POCO4A - 27/09/2022 - Exercícios propostos

Felipe Antonio Magro: Cópia de Aula 14 - POCO4A - 27/09/2022 - Exercícios propostos

Rafael Zaupa Watanabe: Cópia de Aula 14 - POCO4A - 27/09/2022 - Exercícios propostos

Arnald: Aula 14 - 27/09

Michael Pariz Pereira: Cópia de Aula 14 - POCO4A - 27/09/2022 - Exercícios propostos

Ruan Mateus Trizotti: Aula 14

MARIA EDUARDA PEDROSO :

Mariana Pedroso Naves: Aula 14 - POO

Gustavo Nunes : Cópia de Aula 14 - POCO4A - 27/09/2022 - Exercícios propostos

- 1) (Online) Acesse o link e realize as atividades propostas:

<https://codeboard.io/projects/346997>

```
/**
 * TODO 1: Classe Principal: Defina um objeto do tipo MembroUniversitario.
 *      Instancie esse objeto com um tipo Estudante.
 * TODO 2: Classe Principal: Utilize o objeto instanciado para
 *      invocar o metodo 'getHistorico()'.
 * TODO 3: Classe Principal: Defina um objeto do tipo Egresso.
 *      Instancie esse objeto com um tipo Estudante.
 * TODO 4: Classe Principal: Utilize o objeto instanciado para
 *      invocar o metodo 'getDiploma()'.
 */

public class Principal {

    public abstract class MembroUniversitario {
        protected String nome;
        protected float nota;
        public abstract void getHistorico();
        MembroUniversitario(String nome, float nota){
```

```

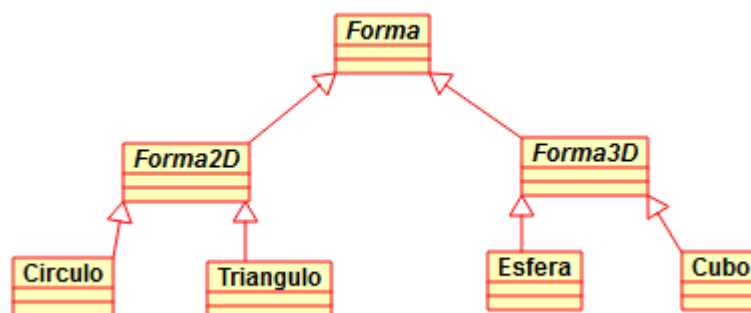
        this.nome = nome;
        this.nota = nota;
    }
}
public interface Egresso {
    public abstract void getDiploma();
}
public class Estudante extends MembroUniversitario implements Egresso {
    public Estudante(String nome, float nota){
        super(nome,nota);
    }
    public void getHistorico(){
        System.out.println("Historico: " + nome + " Nota: "+nota);
    }
    public void getDiploma(){
        System.out.println("Diploma: " + nome + " Nota: "+nota);
    }
}

public void iniciar(){
    //TODO1
    MembroUniversitario aluno = new Estudante("jose",20f);
    //TODO2
    aluno.getHistorico();
    //TODO3
    Egresso egresso = new Estudante("carlos",20f);
    //TODO4
    egresso.getDiploma();
}

public static void main(String[] args) {
    Principal principal = new Principal();
    principal.iniciar();
}
}

```

2) Observe a Figura a seguir:



- a) Implemente a hierarquia de Classes mostrada na Figura. Apenas as Classes folha são Classes concretas, as demais são classes abstratas.
- b) A Classe Forma2D deve conter o método getArea.
- c) A Classe Forma3D deve conter os métodos getArea e getVolume.
- d) Crie uma Classe Principal que tenha um vetor de Formas com objetos de cada Classe concreta.
- e) O programa deve imprimir o tipo de cada objeto instanciado.
- f) Modifique o exercício para incluir todas as classes em uma classe interna.

```
public abstract class Forma {  
    protected float area;  
  
    public Forma ( float area ){  
        this.area = area;  
    }  
  
    public abstract float getArea();  
}
```

```
public abstract class Forma2D extends Forma {  
  
    public Forma2D( float area ){  
        super( area );  
    }  
}
```

```
public abstract class Forma3D extends Forma {  
  
    public abstract float getVolume();  
}
```

```
public class Circulo extends Forma2D {  
  
    public Circulo( float area ){  
        super( area );  
    }  
}
```

```
public class Triangulo extends Forma2D {
}
```

```
public class Esfera extends Forma3D {
}
```

```
public class Cubo extends Forma3D {
}
```

```
public class Principal {

    public Principal(){
        Forma [ ] lista = new Forma[10];

        Circulo c1 = new Circulo(1.23f);
        lista[0] = c1;      //Forma (superclasse) ←Circulo (subclasse)

    }

    public static void main(String [ ] args){
        Principal principal = new Principal();
    }

}
```

```
public abstract class Forma {
    protected float area;

    public Forma ( float area ){
        this.area = area;
    }

    public abstract float getArea();
}

public abstract class Forma2D extends Forma {

    public Forma2D( float area ){
        super( area );
    }
    public abstract float getArea();

    public String toString(){
        return this.getClass().getSimpleName() + " Area: " + getArea();
    }

}
```

```
public abstract class Forma3D extends Forma {

    public abstract float getVolume();

    public String toString(){
        return this.getClass().getSimpleName() +
            " Area: " + getArea() +
            " Volume: " + getVolume();
    }
}

public class Circulo extends Forma2D {
    private float raio;
    public Circulo( float raio ){
        this.raio = raio;
    }
    public float getArea(){
        return (float) ( Math.PI*Math.pow(raio,2) );
    }
}

public class Triangulo extends Forma2D {
    private float base, altura;

    public Triangulo( float base, float altura ){
        this.base = base;
        this.altura = altura;
    }
    public float getArea(){
        return (float) ( base * altura / 2 );
    }
}

public class Esfera extends Forma3D {
    private float raio;
    public Esfera(float raio){
        this.raio = raio;
    }
    public float getArea(){
        return 4*Math.PI*pow( raio, 2 );
    }
    public float getVolume(){
        return (float) (4/3*Math.PI*Math.pow( raio, 3 ) );
    }
}

public class Cubo extends Forma3D {
    private float base;
```

```

    public Cubo(float base){
        this.base = base;
    }
    public float getArea(){
        return Math.pow( base, 2 ) * 6;
    }
    public float getVolume(){
        return Math.pow( base, 3 );
    }
}

public class Principal {

    public Principal(){
        Forma [ ] lista = new Forma[4];

        Circulo c1 = new Circulo(1.23f);
        lista[0] = c1;      //Forma (superclasse) ← Circulo (subclasse)

        Triangulo t1 = new Triangulo(4.56f, 2f);
        lista[1] = t1;      //Forma (superclasse) ← Triangulo (subclasse)

        Cubo c2 = new Cubo( 7.89f );
        lista[2] = c2;      //Forma (superclasse) ← Cubo (subclasse)

        Esfera e1 = new Esfera( 5.67f );
        lista[3] = e1;      //Forma (superclasse) ← Esfera (subclasse)

        for( Forma forma : lista )
            System.out.println( forma );

    }

    public static void main(String [ ] args){
        Principal principal = new Principal();
    }

}

```

3) Faça a implementação Orientada a Objetos do problema anunciado a seguir:

a) Crie 3 (três) classes não relacionadas por herança: Construção, Carro e Bicicleta.

b) Dê a cada Classe atributos e comportamentos únicos que não estão presentes em outras classes.

- c) Crie a Interface EmissaoCarbono com um método getEmissaoCarbono.
 - d) Cada Classe deve implementar a Interface EmissaoCarbono.
 - e) Invoque o método getEmissaoCarbono de cada objeto.
-