| | |
|---|---|
| | **Carátula para entrega de prácticas** |
| Facultad de Ingeniería | Laboratorio de docencia |

# Laboratorios de computación salas A y B

| | |
|---|---|
| *Profesor:* | Ing. José Antonio Ayala Barbosa |
| *Asignatura:* | Programación Orientada a Objetos |
| *Grupo:* | 01 |
| *No de Práctica(s):* | 08 |
| *Integrante(s):* | Romero Ramírez Alejandro |
| *No. de Equipo de cómputo empleado:* | |
| *No. de Lista o Brigada:* | 35 |
| *Semestre:* | 2021-1 |
| *Fecha de entrega:* | Jueves 3 de diciembre, 2020 |
| *Observaciones:* | |

CALIFICACIÓN:

___

# PRÁCTICA 8: CLASES ABSTRACTAS E INTERFACES

**DESARROLLO**

Se han trabajado con las siguientes clases heredadas de la clase Polígono: Cuadrilátero y Triángulo

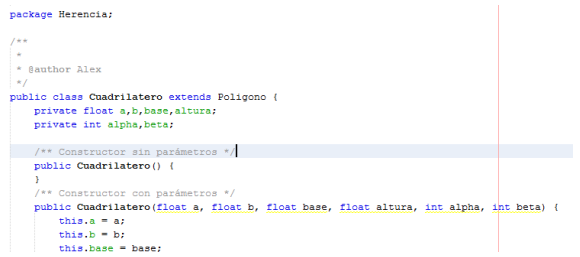
```
 */
package Herencia;

/**
 *
 * @author Alex
 */
public class Poligono {
    /** Constructor sin parámetros */
    public Poligono(){

    }
    /** @return Área del polígono */
    public float area(){
        return 0;
    }
    /** @return Perímetro del polígono */
    public float perimetro(){
        return 0;
```

```
/** Constructor sin parámetros */
public Poligono(){

}
/** @return Área del polígono */
public float area(){
    return 0;
}
/** @return Perímetro del polígono */
public float perimetro(){
    return 0;
}

@Override
public String toString() {
    return "Poligono{" + '}';
```

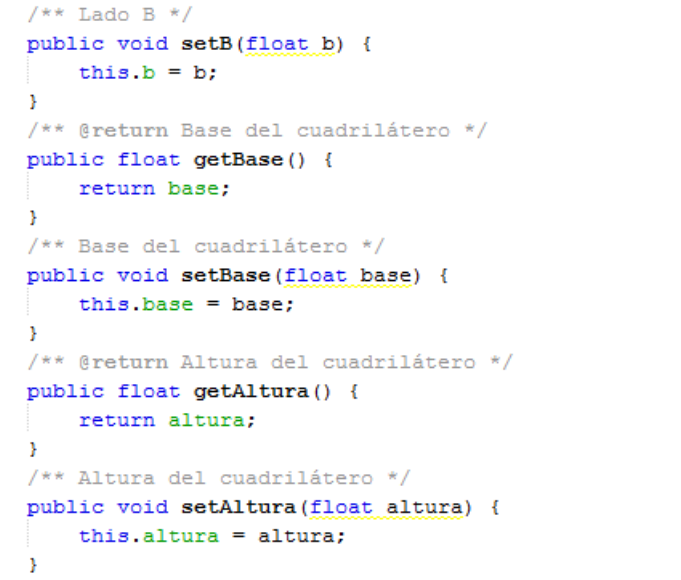
Clase Triángulo:

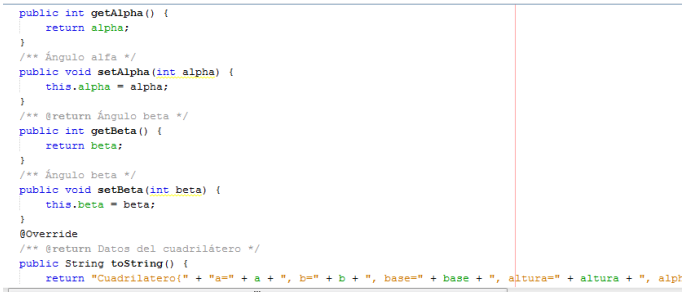
```
        this.base = base;
        this.altura = altura;
        this.alpha = alpha;
        this.beta = beta;
        this.gamma = gamma;
    }
    /** @return Lado A */
    public float getA() {
        return a;
    }
    /** Lado A */
    public void setA(float a) {
        this.a = a;
    }
    /** @return Lado B */
    public float getB() {
        return b;
    }

}
/** @return Lado C */
public float getC() {
    return c;
}
/** Lado C */
public void setC(float c) {
    this.c = c;
}
/** @return Base del triángulo */
public float getBase() {
    return base;
}
/** Base del triángulo */
public void setBase(float base) {
    this.base = base;
}
/** @return Altura del triángulo */
public float getAltura() {

public float getAltura() {
    return altura;
}
/** Altura del triángulo */
public void setAltura(float altura) {
    this.altura = altura;
}
/** @return Ángulo alfa */
public int getAlpha() {
    return alpha;
}
/** Ángulo alfa */
public void setAlpha(int alpha) {
    this.alpha = alpha;
}
/** @return Ángulo beta */
public int getBeta() {
    return beta;
```

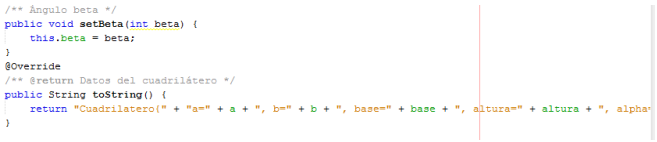
Clase Cuadrilátero:



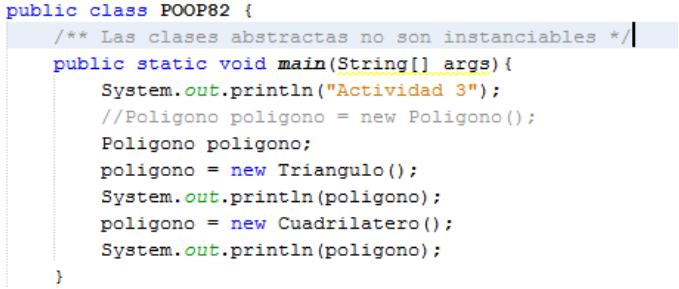
```
        this.b = b;
        this.base = base;
        this.altura = altura;
        this.alpha = alpha;
        this.beta = beta;
    }
    /** @return Lado A */
    public float getA() {
        return a;
    }
    /** Lado A */
    public void setA(float a) {
        this.a = a;
    }
    /** @return Lado B */
    public float getB() {
        return b;
    }
    /** Lado B */
    public void setB(float b) {
```







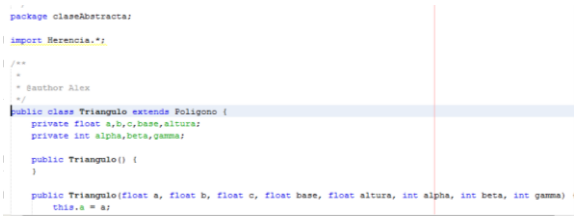
Se ha trabajado de igual manera, pero con clases abstractas (no instanciables):



Clase Polígono:

```java
*/
package claseAbstracta;

/**
 *
 * @author Alex
 */
public abstract class Poligono {
    public abstract float area();
    public abstract float perimetro();

    @Override
    public String toString() {
        return "Poligono{" + '}';
    }
}
```

Clase Triángulo:



```java
package claseAbstracta;

import Herencia.*;

/**
 *
 * @author Alex
 */
public class Triangulo extends Poligono {
    private float a,b,c,base,altura;
    private int alpha,beta,gamma;

    public Triangulo() {
    }

    public Triangulo(float a, float b, float c, float base, float altura, int alpha, int beta, int gamma) {
        this.a = a;
        this.b = b;
        this.c = c;
        this.base = base;
        this.altura = altura;
        this.alpha = alpha;
        this.beta = beta;
        this.gamma = gamma;
    }

    public float getA() {
        return a;
    }

    public void setA(float a) {
        this.a = a;
    }

    public float getB() {
        return b;
    }

    public void setB(float b) {
        this.b = b;
    }

    public float getC() {
        return c;
    }

    public void setC(float c) {
        this.c = c;
    }

    public float getBase() {
        return base;
    }

    public void setBase(float base) {
        this.base = base;
    }

    public float getAltura() {
        return altura;
    }

    public void setAltura(float altura) {
        this.altura = altura;
    }

    public int getAlpha() {
        return alpha;
    }

    public void setAlpha(int alpha) {
```

```
    public void setAlpha(int alpha) {
        this.alpha = alpha;
    }

    public int getBeta() {
        return beta;
    }

    public void setBeta(int beta) {
        this.beta = beta;
    }

    public int getGamma() {
        return gamma;
    }

    public void setGamma(int gamma) {
        this.gamma = gamma;
    }
```

```
@Override
public String toString() {
    return "Triangulo(" + "a=" + a + ", b=" + b + ", c=" + c + ", base=" + base + ", altura=" + altura
}

@Override
public float area() {
    throw new UnsupportedOperationException("Not supported yet."); //To change body of generated method
}

@Override
public float perimetro() {
    throw new UnsupportedOperationException("Not supported yet."); //To change body of generated method
}
```

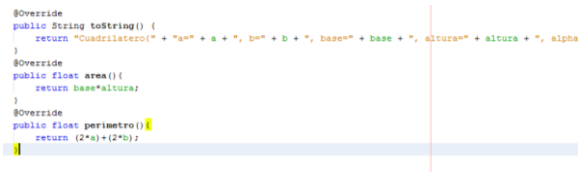
Clase Cuadrilátero

```
 */
package claseAbstracta;

import Herencia.*;

/**
 *
 * @author Alex
 */
public class Cuadrilatero extends Poligono {
    private float a,b,c,d,base,altura;
    private int alpha,beta;

    public Cuadrilatero() {
    }

    public Cuadrilatero(float a, float b, float base, float altura, int alpha, int beta) {
        this.a = a;
```

```
        this.a = a;
        this.b = b;
        this.base = base;
        this.altura = altura;
        this.alpha = alpha;
        this.beta = beta;
    }

    public float getA() {
        return a;
    }

    public void setA(float a) {
        this.a = a;
    }

    public float getB() {
        return b;
    }
```

```
public void setB(float b) {
    this.b = b;
}

public float getBase() {
    return base;
}

public void setBase(float base) {
    this.base = base;
}

public float getAltura() {
    return altura;
}

public void setAltura(float altura) {
```

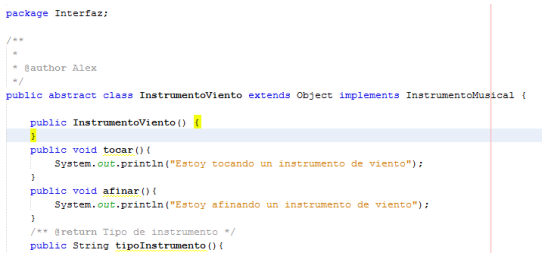
```
public void setAltura(float altura) {
    this.altura = altura;
}

public int getAlpha() {
    return alpha;
}

public void setAlpha(int alpha) {
    this.alpha = alpha;
}

public int getBeta() {
    return beta;
}

public void setBeta(int beta) {
    this.beta = beta;
}
```



Las interfaces son contratos que especifican qué es lo que debe hacer una clase, no cómo lo hace. Aquí se muestra un ejemplo de ello:

```java
package Interfaz;

/**
 *
 * @author Alex
 */
public interface InstrumentoMusical {
    //Por defecto todos nuestros metodos son public y abstract
    void tocar();
    void afinar();
    String tipoInstrumento();
}
```

Dos clases abstractas que implementan esta interfaz son las siguientes:



```java
public void tocar(){
    System.out.println("Estoy tocando un instrumento de viento");
}
public void afinar(){
    System.out.println("Estoy afinando un instrumento de viento");
}
/** @return Tipo de instrumento */
public String tipoInstrumento(){
    return "Instrumento de viento";
}

@Override
public String toString() {
    return "InstrumentoViento{" + '}';
```

```java
 */
package Interfaz;

/**
 *
 * @author Alex
 */
public class Flauta extends InstrumentoViento{

    /** Constructor sin parámetros */
    public Flauta() {
    }
    /** @return Tipo del instrumento */
    public String tipoInstrumento(){
        return "Flauta";
    }

    @Override
    public String toString() {
```

```java
 */
public class Flauta extends InstrumentoViento{

    /** Constructor sin parámetros */
    public Flauta() {
    }
    /** @return Tipo del instrumento */
    public String tipoInstrumento(){
        return "Flauta";
    }

    @Override
    public String toString() {
        return "Flauta{" + '}';
    }
```

Las clases principales se encuentran en los siguientes archivos:

```java
package Herencia;

/**
 *
 * @author Alex
 */
public class POOP8 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args){
        System.out.println("Actividad 1:");
        Poligono p1 = new Poligono();
        /* Las clases base pueden comportarse como sus subclases */
        System.out.println("Poligono: "+p1);
        Object objeto = new Object();
        System.out.println("Object: "+objeto);
```

```java
        Object objeto = new Object();
        System.out.println("Object: "+objeto);
        objeto = p1;
        System.out.println("Object como poligono: "+p1);
        Object objeto2 = p1;
        System.out.println("Object2: "+objeto2);
        Object objeto3 = new Poligono();
        System.out.println("Object3: "+objeto3);
        System.out.println("Actividad 2:");
        p1 = new Triangulo();
        System.out.println(p1);
        selectorPoligonos(p1);
        p1 = new Cuadrilatero();
        System.out.println(p1);
        selectorPoligonos(p1);
        Poligono p2 = new Poligono();
        System.out.println(p2);
        selectorPoligonos(p2);
    }
    public static void selectorPoligonos(Poligono p1){
```

El Método Selector Polígonos permite especificar de qué método se trata gracias a un bloque condicional con la palabra reservada instanceof:

```
    Poligono p2 = new Poligono();
    System.out.println(p2);
    selectorPoligonos(p2);
}
public static void selectorPoligonos(Poligono p1){
    if(p1 instanceof Triangulo){
        System.out.println("El objeto es un triangulo");
    }
    else if (p1 instanceof Cuadrilatero){
        System.out.println("El objeto es un cuadrilatero");
    }
    else if (p1 instanceof Poligono){
        System.out.println("El objeto es un poligono");
    }
    else{
        System.out.println("El objeto es otra figura");
    }
```

```
*/
public class POOP82 {
    /** Las clases abstractas no son instanciables */
    public static void main(String[] args){
        System.out.println("Actividad 3");
        //Poligono poligono = new Poligono();
        Poligono poligono;
        poligono = new Triangulo();
        System.out.println(poligono);
        poligono = new Cuadrilatero();
        System.out.println(poligono);
    }
```

```
package Interfaz;

/**
 *
 * @author Alex
 */
public class POOP83 {
    public static void main(String[] args){
        //InstrumentoMusical instrumento = new InstrumentoM
        System.out.println("Actividad 4");
        InstrumentoMusical instrumento;
        instrumento = new Flauta();
        instrumento.tocar();
        instrumento.afinar();
        System.out.println(instrumento.tipoInstrumento());
        System.out.println(instrumento);
```

**CONCLUSIONES**

- Las interfaces son contratos que poseen siempre lo que dicen las clases.
- Las clases abstractas son simplemente patrones de diseño que no deben considerarse como objetos únicos
- Una interfaz o clase abstracta puede heredar de otras siempre y cuando se respeten las convenciones anteriores