	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: 1

No de Práctica(s): Práctica 8 Polimorfismo

Integrante(s): José Luis Arroyo Chavarría
Francisco Moisés Barrera Guardia

No. de Equipo de cómputo empleado:

No. de Lista o Brigada:

Semestre: 3

Fecha de entrega: 03/12/2020

Observaciones:

CALIFICACIÓN: _____

I. Previo

- Clases: Polígono, Triangulo, Cuadrilátero.

➢ ***Clase Triangulo:***

```
Main class Triangulo;
        float lado_1, lado_2, lado_3;
        Int angulo_1, angulo_2, angulo_3;
        System.out.scan(lado_1, lado_2);
        Área=(lado_1*lado_2)/2;
        System.out.println("el área es" Area);
```

➢ ***Clase Polígono:***

```
Main class polígono;
        Int nlados;
        Int lado_1, lado_2, lado_3, lado_4, lado_5;
        System.out.scan(nlados);
          System.out.scan(lado_1, lado_2, lado_3, lado_4, lado_5);
        Área=(n*l*a)/2;
        System.out.println("el área es" Area);
```

➢ ***Clase cuadrilátero:***

```
Main class cuadrilátero;
        float lado_1, lado_2, lado_3, lado_4;
        Int angulo_1, angulo_2, angulo_3, angulo_4;
        System.out.scan(lado_1, lado_2, lado_3, lado_4);
        Área=(lado_1*lado_2);
        System.out.println("el área es" Area);
```

II. Objetivo

Implementar el concepto de polimorfismo en un lenguaje de programación orientado a objetos.

III. Introducción

En esta práctica, se verán los diferentes tipos de polimorfismo que existen, además de las formas en que se puede utilizar el mismo, cabe mencionar, que para el desarrollo de esta práctica, se elaboraron 3 clases como lo fueron triangulo, polígono y cuadrilátero, para que a partir de estas clases se fuera más fácil el desarrollo de las actividades.

El polimorfismo se refiere a la propiedad por la que es posible enviar mensajes sintácticamente iguales a objetos de tipos distintos. El único requisito que deben cumplir los objetos que se utilizan de manera polimórfica es saber responder al mensaje que se les envía.

La apariencia del código puede ser muy diferente dependiendo del lenguaje que se utilice, más allá de las obvias diferencias sintácticas

IV. Desarrollo

- *Actividades en Lab:*

1. Clases base se comportan como subclases
2. Métodos con instanceof
3. Clase Abstracta
4. Interfaz

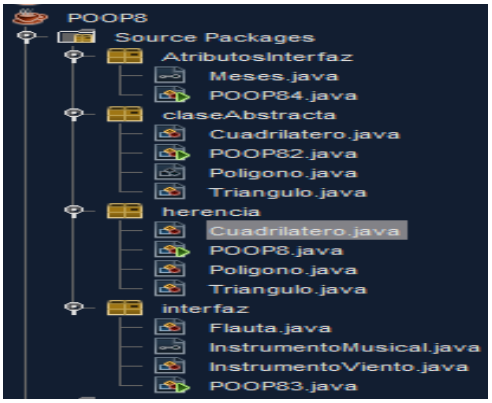
5. Atributos de interfaces

- *Actividades:*

A partir de una jerarquía de clases, implementar referencias que se comporten como diferentes objetos.

V. Código fuente

- ***Actividades en Lab:***



1. Clases base se comportan como subclases y Métodos con instanceof

```
package herencia;
public class POOP8 {
   public static void main(String[] args)
{
System.out.println("1****************
****************************** ");
      /* Las clases Base pueden
comportarse como sus subclases */
      Poligono poligono = new
Poligono();
      System.out.println("Poligono " +
poligono);
      Object objeto = new Object();
      System.out.println("Object " +
objeto);
      objeto = poligono;
      System.out.println("Object como
Poligono " + objeto);
      Object objeto2 = poligono;
      System.out.println("Object2 " +
objeto2);
      Object objeto3 = new Poligono();
      System.out.println("Object3 " +
objeto3);
System.out.println("2****************
****************************** ");
      poligono = new Triangulo();
      System.out.println(poligono);
      selectorPoligonos(poligono);
      poligono = new Cuadrilatero();
      System.out.println(poligono);
      selectorPoligonos(poligono);
      poligono = new Poligono();
      System.out.println(poligono);
      selectorPoligonos(poligono);
   }
   public static void
selectorPoligonos(Poligono poligono){
      if(poligono instanceof Triangulo){
         System.out.println("El objeto
es un triangulo");
      }else if(poligono instanceof
Cuadrilatero){
         System.out.println("El objeto
es un cuadrilatero");
      }else if(poligono instanceof
Poligono){
         System.out.println("El objeto
es un Poligono");
      }else{
         System.out.println("El objeto
es otra figura");
      }
   }
}
```

```
package herencia;
public class Poligono {
   public Poligono() {
   }
   public float area(){
      return 0;
   }
```

```
    public float perimetro(){
        return 0;
    }
    @Override
    public String toString() {
        return "Poligono{" + '}';
    }
}
```

```
package herencia;
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) {
        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 + ", alpha=" + alpha + ", beta=" + beta + ", gamma=" + gamma + '}';
    }
}
```

```
package herencia;
public class Cuadrilatero extends Poligono{
    private int alpha, beta;
    private float a, b, base, altura;
    public Cuadrilatero() {
    }
```

```
    public Cuadrilatero(int alpha, int beta, float a, float b, float base, float altura) {
        this.alpha = alpha;
        this.beta = beta;
        this.a = a;
        this.b = b;
        this.base = base;
        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;
    }
    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) {
        this.altura = altura;
    }
    @Override
    public String toString() {
        return "Cuadrilatero{" + "alpha=" + alpha + ", beta=" + beta + ", a=" + a + ", b=" + b + ", base=" + base + ", altura=" + altura + '}';
    }
}
```

```
run:
1**********************************************
Poligono Poligono{}
Object java.lang.Object@15db9742
Object como Poligono Poligono{}
Object2 Poligono{}
Object3 Poligono{}
2**********************************************
Triangulo{a=0.0, b=0.0, c=0.0, base=0.0, altura=0.0, alpha=0, beta=0, gamma=0}
El objeto es un triangulo
Cuadrilatero{alpha=0, beta=0, a=0.0, b=0.0, base=0.0, altura=0.0}
El objeto es un cuadrilatero
Poligono{}
El objeto es un Poligono
BUILD SUCCESSFUL (total time: 0 seconds)
```

2. Clase Abstracta

```
package claseAbstracta;
public class POOP82 {
    public static void main(String[] args)
{
System.out.println("3**********
********** ");
        //Poligono poligono = new Poligono();
        Poligono poligono;
        poligono = new Triangulo();
        System.out.println(poligono);
        poligono = new Cuadrilatero();
        System.out.println(poligono);
    }
}
package claseAbstracta;
public abstract class Poligono {
    public abstract float area();
    public abstract float perimetro();
    @Override
    public String toString() {
        return "Poligono{" + '}';
    }
}

package claseAbstracta;
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) {
        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 + ", alpha=" +
alpha + ", beta=" + beta + ", gamma="
+ gamma + '}';
    }
    @Override
    public float area(){
        return base*altura/2;
    }
    @Override
    public float perimetro(){
        return a+b+c;
    }
}

package claseAbstracta;
public class Cuadrilatero extends
Poligono{
    private int alpha, beta;
    private float a, b, base, altura;
    public Cuadrilatero() {
    }
    public Cuadrilatero(int alpha, int
beta, float a, float b, float base, float
altura) {
        this.alpha = alpha;
        this.beta = beta;
        this.a = a;
```

```
        this.b = b;
        this.base = base;
        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;
    }
    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) {
        this.altura = altura;
    }
    @Override
    public String toString() {
        return "Cuadrilatero{" + "alpha="
+ alpha + ", beta=" + beta + ", a=" + a
+ ", b=" + b + ", base=" + base + ",
altura=" + altura + '}';
    }
    @Override
    public float area(){
        return base*altura;
    }
    @Override
    public float perimetro(){
        return 2*a+2*b;
    }
}
```

```
run:
3*********************************************
Triangulo{a=0.0, b=0.0, c=0.0, base=0.0, altura=0.0, alpha=0, beta=0, gamma=0}
Cuadrilatero{alpha=0, beta=0, a=0.0, b=0.0, base=0.0, altura=0.0}
BUILD SUCCESSFUL (total time: 0 seconds)
```

3. Interfaz

```
package interfaz;
public class POOP83 {
    public static void main(String[] args)
{
System.out.println("4******************
****************************** ");
        //InstrumentoMusical instrumento
= new instrumentoMusical();
        InstrumentoMusical instrumento;

        instrumento = new Flauta();
        instrumento.tocar();
        instrumento.afinar();
System.out.println(instrumento.tipoIns
trumento());
        System.out.println(instrumento);
    }
}
```

```
package interfaz;
public interface InstrumentoMusical {
    //Por defecto todos los metodos son
public y abstract
    void tocar();
    void afinar();
    String tipoInstrumento();
}
```

```
package interfaz;
```

```
public class InstrumentoViento extends Object implements InstrumentoMusical {
    public InstrumentoViento() {
    }
    @Override
    public void tocar(){
        System.out.println("Estoy tocando un instrumento de viento");
    }
    @Override
    public void afinar(){
        System.out.println("Estoy afinando un instrumento de viento");
    }
    @Override
    public String tipoInstrumento(){
        return "Instrumento de Viento";
    }
    @Override
    public String toString() {
        return "InstrumentoViento{" + '}';
    }
}
```

```
package interfaz;
public class Flauta extends InstrumentoViento{
    public Flauta() {
    }
    @Override
    public String tipoInstrumento(){
        return "Flauta";
    }
    @Override
    public String toString() {
        return "Flauta{" + '}';
    }
}
```

```
run:
4**************************************************
Estoy tocando un instrumento de viento
Estoy afinando un instrumento de viento
Flauta
Flauta{}
BUILD SUCCESSFUL (total time: 0 seconds)
```

4. Atributos de interfaces

```
package AtributosInterfaz;
public class POOP84 {
    //PSVM
    public static void main(String[] args)
{
System.out.println("5**********
****************************** ");
        System.out.println("El mes
"+Meses.CUATRO+" corresponde a:
");
System.out.println(Meses.NOMBRE_
MESES[Meses.CUATRO]);
    }
}
```

```
package AtributosInterfaz;
public interface Meses {
    //Atributos son:
        //public static final
    int UNO = 1, DOS = 2, TRES = 3,
CUATRO = 4, CINCO = 5, SEIS = 6;
    int SIETE = 7, OCHO = 8, NUEVE =
9, DIEZ = 10, ONCE = 11, DOCE = 12;
    String[] NOMBRE_MESES =
{"","enero","febrero","marzo","abril","m
ayo","junio",
"julio","agosto","septiembre","octubre"
,"noviembre","diciembre"};
}
```

```
run:
5**************************************************
El mes 4 corresponde a:
abril
BUILD SUCCESSFUL (total time: 0 seconds)
```

- ***Actividades:***

A partir de una jerarquía de clases, implementar referencias que se comporten como diferentes objetos

- *Clases base se comportan como subclases:*

La herencia es un pilar importante de OOP (Programación Orientada a Objetos). Es el mecanismo en Java por el cual una clase permite heredar las características (atributos y métodos) de otra clase. Aprenda más a continuación.

En el lenguaje de Java, una clase que se hereda se denomina superclase. La clase que hereda se llama subclase. Por lo tanto, una subclase es una versión especializada de una superclase. Hereda todas las variables y métodos definidos por la superclase y agrega sus propios elementos únicos.

- *Métodos con instanceof:*

Se utiliza para probar si el objeto es una instancia del tipo especificado (clase, subclase o interfaz).

El instanceof en java también se conoce como operador de comparación de tipos porque compara la instancia con el tipo. Devuelve verdadero o falso. Si aplicamos el operador instanceof con cualquier variable que tenga un valor nulo, devuelve falso.

- *Clase Abstracta:*

Es posible definir métodos abstractos, los cuales se caracterizan por el hecho de que no pueden ser implementados en la clase base. De ellos, solo se escribe su signatura en la superclase, y su funcionalidad –polimórfica– tiene que indicarse en las subclases.

- *Interfaz:*

Es una colección de métodos abstractos y propiedades constantes.

En las interfaces se especifica qué se debe hacer pero no su implementación. Serán las clases que implementen estas interfaces las que describen la lógica del comportamiento de los métodos.

- *Atributos de interfaces:*

Son la herramienta canónica que ofrece el lenguaje para añadir metainformación a una clase. En el caso más simple, el atributo únicamente decora una clase marcándola, pero tienen la ventaja de que el propio atributo puede contener información adicional, por lo que podemos incluir más metainformación.

VI. Conclusión

- José Luis Arroyo Chavarría:

En esta práctica pude visualizar las diferentes formas de polimorfismo que conjunto con lo aprendido anteriormente nos ayudara para realizar diferentes trabajos con conjunto a otros que necesitan propiedades o ayuda pero no solo eso sino vi la importancia o utilidad de la interfaz para realizar estos proyectos. Espero ver más de estos ejercicios en el futuro.

- Francisco Moisés Barrera Guardia:

En esta práctica aprendí que es el polimorfismo, y como se ocupa, aparte de reforzar los temas anteriores, con ejercicios que contienen uno o varios temas juntos, para así meter el tema de polimorfismo, y esta práctica fue muy rápida por que el profesor explico a detalle todos y cada uno delos ejercicios vistos.

VII. Referencias

- https://es.wikipedia.org/wiki/Interfaz_(Java)
- https://blog.koalite.com/2015/03/interfaces-marcadoras-atributos-y-convenciones/
- https://www.javatpoint.com/downcasting-with-instanceof-operator
- https://www.abrirllave.com/java/clases-abstractas.php
- https://javadesdecero.es/poo/herencia-java-tipos-ejemplos/#:~:text=La%20herencia%20es%20un%20pilar,y%20m%C3%A9todos)%20de%20otra%20clase.&text=Hereda%20todas%20las%20variables%20y,agrega%20sus%20propios%20elementos%20%C3%BAnicos.
- https://es.wikipedia.org/wiki/Polimorfismo_(inform%C3%A1tica)