



Instituto Politécnico Nacional
Unidad Profesional
Interdisciplinaria De Ingeniería
Campus Zacatecas



Ingeniería en Sistemas Computacionales

Práctica 6- Herencia

Alumna: Vanessa Melenciano Llamas

Boleta: 2020670081

Profesora: Monreal Mendoza Sandra Mireya

Materia: Programación Orientada a Objetos

Grupo: 2CM2

Fecha: 25 de noviembre de 2020

Índice

Introducción	3
Objetivos.....	3
Desarrollo	4
Diseño y funcionamiento de la solución.....	4
Funcionamiento	12
Errores detectados	12
Posibles mejoras	12
Conclusión	12
Referencia.....	13

Introducción

La herencia es específica de la programación orientada a objetos, donde una clase nueva se crea a partir de una clase existente. La herencia (a la que habitualmente se denomina subclase) proviene del hecho de que la subclase (la nueva clase creada) contiene los atributos y métodos de la clase primaria. La herencia permite que las propiedades de la superclase se propaguen a las subclases en una jerarquía de clases, por lo tanto, es un mecanismo que permite a una clase de objetos compartir la representación y los métodos de otra clase de objetos. Es una forma de reutilización de software, en la cual para crear una nueva clase se absorben los datos y comportamientos de una clase existente y se mejoran con capacidades nuevas. Existen dos tipos de herencia: simple y múltiple.

La herencia simple se da cuando una clase se deriva de una sola clase base. La herencia múltiple se da cuando una clase derivada hereda de varias clases base (posiblemente no relacionadas).

La principal ventaja de la herencia es la capacidad para definir atributos y métodos nuevos para la subclase, que luego se aplican a los atributos y métodos heredados. Esta particularidad permite crear una estructura jerárquica de clases cada vez más especializada. La gran ventaja es que uno ya no debe comenzar desde cero cuando desea especializar una clase existente. Como resultado, se pueden adquirir bibliotecas de clases que ofrecen una base que puede especializarse a voluntad. Gracias a esto, lograremos un código mucho más limpio, estructurado y con menos líneas de código, lo que lo hace más legible.

En Java tenemos que tener claro cómo llamar a la clase principal de la que heredamos y aquella que hereda de ella, así, 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.

En Java, cada clase solo puede derivarse de otra clase. Esa clase se llama superclase, o clase padre. La clase derivada se llama subclase o clase secundaria.

Utiliza la palabra clave `extends` para identificar la clase que extiende su subclase. Si no declara una superclase, su clase amplía implícitamente la clase `Object`. El objeto es la raíz de todas las jerarquías de herencia; Es la única clase en Java que no se extiende de otra clase.

Objetivos

Desarrollar un proyecto donde se emplee la herencia simple.

Desarrollo

Diseño y funcionamiento de la solución

```
package figurasgeometricas;

/**
 * Nombre: Vanessa Melenciano Llamas
 * Tema del programa: Herencia
 * Descripción: Relación entre objetos
 * Fecha: 25/11/20
 */
public abstract class Figura {
    public String nombre;
    public int grosorBorde;
    public int color;

    public Figura() {
    }
    public Figura(int col) {
        color = col;
    }
    public void setNombre(String nom) {
        nombre=nom;
    }
    public String getNombre() {
        return nombre;
    }
    public void setGrosorBorde(int gb) {
        grosorBorde = gb;
    }

    public void setGrosorBorde(int gb) {
        grosorBorde = gb;
    }
    public int getGrosorBorde() {
        return grosorBorde;
    }
    public void setColor(int col) {
        color=col;
    }
    public int getColor() {
        return color;
    }

    //METODO ABSTRACTO
    public abstract void dibujar();
}
```

La clase de Figuras tiene los métodos y atributos que usarán las siguientes clases, como son, el nombre, grosor de Borde y el color, así como el método de dibujar.

```
package figurasgeometricas;

] /**
  * Nobre: Vanessa Melenciano Llamas
  * Tema del programa: Herencia
  * Descripcion: Relacion entre objetos
  * Fecha: 25/11/20
- */
public abstract class Figura2D extends Figura{
]     public Figura2D(){
-     }
]     public Figura2D(int color){
-     |     super( col: color);
-     }
    public abstract float calcularArea();
    public abstract float calcularPerimetro();
}
```

La clase de Figuras2D, como su nombre lo dice, es la que tiene los métodos en común de las figuras de dos dimensiones, solo calcular el área y el perímetro. Hereda de la clase Figura pero no usa ninguna de las clases de esta última.

```

public class Circulo extends Figura2D{
    private float radio;
    public Circulo () {
        radio = 7;
    }
    public Circulo (float r){
        radio=r;
    }
    public void setRadio(float r){
        radio = r;
    }
    public float getRadio(){
        return radio;
    }
    @Override
    public float calcularArea(){
        float f=0;
        f=(float) (Math.PI*Math.pow( a: radio, b: 2));
        return f;
    }
    @Override
    public float calcularPerimetro(){
        float p=0;
        p=(float) (2*Math.PI*radio);
        return p;
    }

    @Override
    public float calcularArea(){
        float f=0;
        f=(float) (Math.PI*Math.pow( a: radio, b: 2));
        return f;
    }
    @Override
    public float calcularPerimetro(){
        float p=0;
        p=(float) (2*Math.PI*radio);
        return p;
    }
    @Override
    public void dibujar(){
        System.out.println( a: " ");
    }
    public float cambiarTamanio(){
        radio *=2;
        return radio;
    }
}

```

```

public class Cuadrado extends Figura2D {
    private float lado;
    public Cuadrado() {
        lado = 5;
    }
    public Cuadrado(float l) {
        lado=l;
    }
    public void setLado(float lad) {
        lado = lad;
    }
    public float getLado() {
        return lado;
    }

    @Override
    public float calcularArea() {
        float f=0;
        f=(float) (lado*lado);
        return f;
    }
    @Override
    public float calcularPerimetro() {
        float p=0;
        p=(float) (lado*4);
        return p;
    }
    @Override
    public void dibujar() {
        System.out.println(" ");
    }

    public float cambiarTamanio() {
        lado *=3;
        return lado;
    }
}

```



```

public class Triangulo extends Figura2D{
    private float base;
    private float altura;
    public Triangulo(){
        base = 9;
        altura = 6;
    }
    public Triangulo(float b, float a){
        base = b;
        altura = a;
    }
    public void setBase(float b){
        base = b;
    }
    public float getBase(){
        return base;
    }
    public void setAltura(float a){
        altura = a;
    }
    public float getAltura(){
        return altura;
    }

    @Override
    public float calcularArea(){
        float f=0;
        f=(float) (base*altura*(1/2));
        return f;
    }
    @Override
    public float calcularPerimetro(){
        float p=0;
        p=(float) (base *3);
        return p;
    }
    @Override
    public void dibujar(){
        System.out.println(" // ");
    }
    public float cambiarTamano(){
        base *=2;
        return base;
    }
}

```


Las clases de triángulo, círculo y cuadrado, heredan directamente de la clase Figura2D, ya que cada una de ellas calcula el área y perímetro de las correspondientes figuras, pero al mismo tiempo, tienen un método llamado dibujar, el cual es de Figura.

```

package figurasgeometricas;

/**
 * Nobre: Vanessa Melenciano Llamas
 * Tema del programa: Herencia
 * Descripcion: Relacion entre objetos
 * Fecha: 25/11/20
 */
public abstract class Figura3D extends Figura{
    public Figura3D() {
    }
    public Figura3D(int color){
        super( col:color);
    }
    public abstract float calcularVolumen();
}

```

La clase de Figuras3D tiene la misma función que Figuras2D, tiene el método que tienen en común las clases de las figuras de tres dimensiones, en este caso solo el método de celular volumen

```

public class Cubo extends Figura3D{
    private Cuadrado cuad ;
    public Cubo() {
        cuad = new Cuadrado();
    }
    public Cubo(float l){
        cuad = new Cuadrado(l);
    }
    public void setLado(Cuadrado l){
        cuad = l;
    }
    public Cuadrado getLado() {
        return cuad;
    }
    public float calcularVolumen() {
        float f=0;
        f=(float) (cuad.calcularArea()*cuad.getLado());
        // f = Math.pow(cuad.getLado(), 3);
        return f;
    }
    public float calcularAreaSuperficial() {
        return cuad.calcularArea()*6;
    }
    public void dibujar() {
        System.out.println( "[]");
    }
}

```

```

public class Esfera extends Figura3D{
    private float radio;
1   public Esfera(){
    }
1   public Esfera(float r){
        radio=r;
    }
1   public void setRadio(float r){
        radio = r;
    }
1   public float getRadio(){
        return radio;
    }
    @Override
1   public float calcularVolumen(){
        float f=0;
        f=(float) ((4/3 ) *Math.PI*Math.pow( a: radio, b: 3));
        return f;
    }
    @Override
1   public void dibujar(){
        System.out.println( x: "O");
    }
}

```

```

public class Piramide extends Figura3D{
    private float altura;
    private Triangulo tri;
    public Piramide(){
        tri = new Triangulo();
    }
    public Piramide(float a, float b, float a2){
        tri = new Triangulo( b: a, a: b);
        altura = a2;
    }
    public Piramide(float a, Triangulo t){
        tri = t;
    }
    public float calcularVolumen(){
        float f=0;
        f=(float) ((1/3) * (tri.calcularArea() * altura));
        return f;
    }
    public float calcularAreaSuperficial(){
        return tri.calcularArea() * 3;
    }
    public void dibujar(){
        System.out.println( x: "OO");
    }
}

```

Las clases de pirámide, cubo y esfera, se encargan de calcular el volumen de cada respectiva figura, usando el método heredado de Figuras3D, así como la función dibujar, que viene directamente de la clase Figuras.

```
public class TestFiguras {
    public static void main(String args[]){
        Esfera esf = new Esfera( r:5);
        System.out.println("El volumen de la esfera es: " + esf.calcularVolumen());

        Cubo c = new Cubo( l:7);
        System.out.println("El volumen del cubo es: " + c.calcularVolumen());

        Piramide p = new Piramide( a:2, b:5, a2:3);
        System.out.println("El volumen de la piramide es: " + p.calcularVolumen());

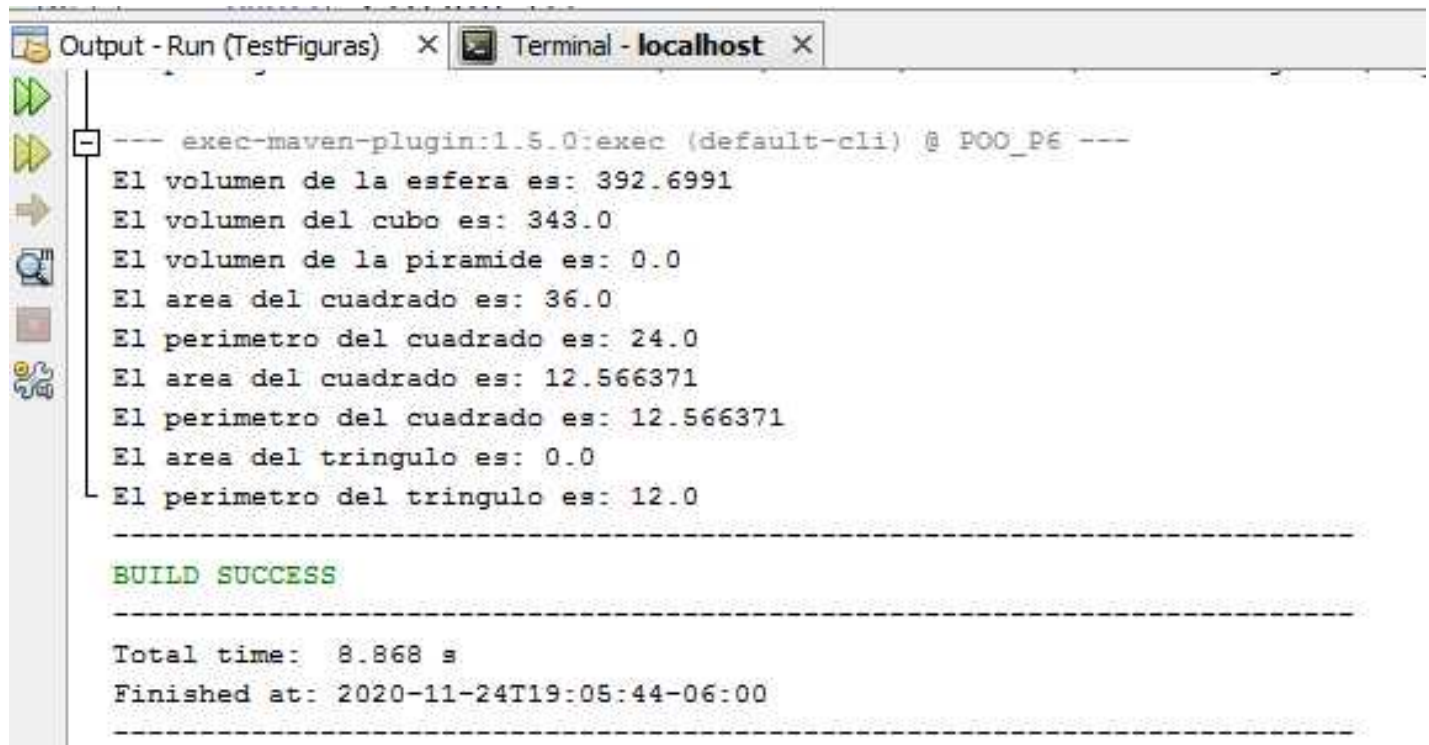
        Cuadrado cua = new Cuadrado( l:6);
        System.out.println("El area del cuadrado es: " + cua.calcularArea());
        System.out.println("El perimetro del cuadrado es: " + cua.calcularPerimetro());

        Circulo cir = new Circulo( r:2);
        System.out.println("El area del cuadrado es: " + cir.calcularArea());
        System.out.println("El perimetro del cuadrado es: " + cir.calcularPerimetro());

        Triangulo tri = new Triangulo( b:4, a:7);
        System.out.println("El area del tringulo es: " + tri.calcularArea());
        System.out.println("El perimetro del tringulo es: " + tri.calcularPerimetro());
    }
}
```

Por último, la clase TestFiguras, la cual solo contiene el main, donde se crea un objeto de cada figura para calcular el área y perímetro, en caso de las figuras de dos dimensiones, y el volumen de las figuras de tres dimensiones, esto llamando a la función encargada de realizar estas operaciones dentro de cada clase, para después mostrar el resultado en pantalla.

Funcionamiento



```
--- exec-maven-plugin:1.5.0:exec (default-cli) @ POO_P6 ---
El volumen de la esfera es: 392.6991
El volumen del cubo es: 343.0
El volumen de la piramide es: 0.0
El area del cuadrado es: 36.0
El perimetro del cuadrado es: 24.0
El area del cuadrado es: 12.566371
El perimetro del cuadrado es: 12.566371
El area del triangulo es: 0.0
El perimetro del triangulo es: 12.0

-----
BUILD SUCCESS
-----

Total time: 8.868 s
Finished at: 2020-11-24T19:05:44-06:00
-----
```

El programa imprime el resultado del volumen de las figuras 3D y el perímetro y área de las figuras 2D

Errores detectados

La mayor confusión en esta práctica, fue comprender la sintaxis de como una clase hereda a otras ciertas clases, creando nuevas basadas en otra ya existente, pero ya implementando esta metodología, tiene más sentido como funciona la herencia en POO.

Posibles mejoras

Lo que puede ayudar a una mejor comprensión de este tema, es practicar con más programas implementando la herencia, ya que los mayores problemas se pueden presentar teniendo una problemática que se debe atacar con esta metodología.

Conclusión

La herencia permite que se puedan definir nuevas clases basadas de unas ya existentes a fin de reutilizar el código, generando así una jerarquía de clases dentro de una aplicación. Si una clase deriva de otra, esta hereda sus atributos y métodos y puede añadir nuevos atributos, métodos o redefinir los heredados.

El objetivo de la práctica se cumplió, al hacer un correcto uso de la herencia simple para poder realizar correctamente lo que se solicitaba.

Referencia

Ottogalli Fernández, K. A., Martínez Morales, A. A., & León Guzmán, L. (2011). NASPOO: una notación algorítmica estándar para Programación Orientada a Objetos. *Universidad Privada Dr. Rafael Belloso Chacín*, pp. 81-102.