Práctica 3 – Herencia (utilizando Java)

Objetivo:
Trabajar con el concepto de herencia y polimorfismo (utilizando Java).

NOTA: Trabajar sobre la carpeta "tema5" del proyecto

1 – A- Agregar la clase Triángulo a la jerarquía de clases vista en el tema 5 (paquete tema5 del proyecto). Triángulo debe heredar de Figura todo lo que es *común* y definir su constructor y sus atributos y métodos *propios*. Además debe redefinir el método *toString*.

B- Escriba un programa que instancie un triángulo y un cuadrado, con información leída desde teclado. Luego muestre en consola el área y perímetro de cada uno y su representación en String.

2- Queremos representar la información de empleados de un club: jugadores y entrenadores.

- Cualquier *empleado* se caracteriza por su nombre y sueldo básico.
- Los *jugadores son empleados* que se caracterizan por el número de partidos jugados y el número de goles anotados.
- Los *entrenadores son empleados* que se caracterizan por la cantidad de campeonatos ganados.

A- Implemente la jerarquía de clases, con los atributos de cada clase y métodos para obtener/modificar el valor de los mismos.

B- Implemente *constructores* para los jugadores y entrenadores, que reciban toda la información necesaria para inicializar el objeto en cuestión.

C- Cualquier empleado (jugador / entrenador) debe saber responder al mensaje c*alcularSueldoACobrar* (que calcula y devuelve el sueldo a cobrar) pero de manera diferente:

- Para los *jugadores*: el sueldo a cobrar es el sueldo básico y si el promedio de goles por partido es superior a 0,5 se adiciona un plus de otro sueldo básico.
- Para los *entrenadores*: el sueldo a cobrar es el sueldo básico al cual se le adiciona un plus por campeonatos ganados (5000$ si ha ganado entre 1 y 4 campeonatos; $30.000 si ha ganado entre 5 y 10 campeonatos; 50.000$ si ha ganado más de 10 campeonatos).

D- Cualquier empleado debe responder al mensaje *toString,* que devuelve un String que lo representa. La representación de cualquier empleado está compuesta por su nombre y sueldo a cobrar.

E- Escriba un programa principal que instancie un *jugador* y un *entrenador* con datos leídos desde teclado. Pruebe el correcto funcionamiento de cada método implementado.

NOTA: Tomar como base la clase Entrenador definida en la Actividad 3.

3- A- Modele e implemente las clases para el siguiente problema. Una garita de seguridad quiere identificar los distintos tipos de personas que entran a un barrio cerrado. Al barrio pueden entrar personas, que se caracterizan por su nombre, DNI y edad. Además pueden entrar trabajadores, estos son personas que se caracterizan además por la tarea que realizan en el predio.

Implemente constructores, getters y setters para las clases. Además tanto las personas como los trabajadores deben responder al mensaje toString(). A continuación se ejemplifica la representación a retornar por cada uno:

- Personas: "Mi nombre es **Mauro**, mi DNI es **11203737** y tengo **70** años"
- Trabajadores: "Mi nombre es **Mauro**, mi DNI es **11203737** y tengo **70** años. Soy **Corta césped.**"

B- Genere un programa que instancie una persona y un trabajador con datos leídos de teclado y muestre la representación de cada uno en consola.

NOTA: reutilice la clase Persona (tema 2).

4- Dada la siguiente jerarquía, indique qué imprime el programa.

```
public class ClaseA {

   public int dos(){
      return 2;
   }

   public int tres(){
      return this.dos() + this.siete();
   }

   public int siete(){
      return 9;
   }
}
```

```
public class ClaseB extends ClaseA{

   public int dos(){
      return 5;
   }

   public int cuatro(){
      return this.dos() + super.tres();
   }

    public int seis(){
      return this.dos();
   }
}
```

```
public class ClaseC extends ClaseB{

   public int uno(){
      return this.cuatro();
   }

   public int dos(){
      return 9;
   }

   public int cinco(){
      return super.seis();
   }
}
```

```
public class QueImprime {
   public static void main(String[] args) {
      ClaseC objC=new ClaseC();
      System.out.println(objC.cinco());
      System.out.println(objC.uno());
   }
 }
```

5- Un objeto *visor de figuras* se encarga de mostrar en consola cualquier figura que reciba y también mantiene cuántas figuras mostró. Analice y ejecute el siguiente programa y responda: ¿Qué imprime? ¿Por qué?

```
public class VisorFiguras {
   private int mostradas;

   public VisorFiguras(){
      mostradas=0;
   }

   public void mostrar(Figura f){
      System.out.println(f.toString());
      mostradas++;
   }

   public int getMostradas() {
      return mostradas;
   }
}
```

```
public class MainVisorFiguras {
   public static void main(String[] args) {
      VisorFiguras visor = new VisorFiguras();

      Cuadrado c1 = new Cuadrado(10,"Violeta","Rosa");
      Rectangulo r= new Rectangulo(20,10,"Azul","Celeste");
      Cuadrado c2= new Cuadrado(30,"Rojo","Naranja");

      visor.mostrar(c1);
      visor.mostrar(r);
      visor.mostrar(c2);

      System.out.println(visor.getMostradas());
   }
}
```

6- Modificar la clase Visor Figuras: ahora debe permitir guardar las figuras a mostrar (a lo sumo 5) y también mostrar todas las figuras guardadas en forma conjunta. Usar la siguiente estructura.

```
public class VisorFigurasModificado {
   private int guardadas;
   private Figura [] vector;

   public VisorFigurasModificado(){
      //completar
   }

   public void guardar(Figura f){
      //completar
   }

   public boolean quedaEspacio(){
      //completar
   }

   public void mostrar(){
      //completar
   }

   public int getGuardadas() {
      return guardadas;
   }

}
```

Luego realice un programa que instancie un visor, guarde dos cuadrados y un rectángulo en el visor y por último haga que el visor muestre sus figuras.

7- Agregar la clase Círculo (definida en la Act. 3) a la jerarquía de figuras del ejercicio 1.