Ejercicios Tema 1

Ejercicios Adaptados de Apuntes y Exámenes de EDA
Germán Moltó
gmolto@dsic.upv.es
Estructuras de Datos y Algoritmos
Escuela Técnica Superior de Ingeniería Informática
Universidad Politécnica de Valencia

Ejercicio de Clases: Persona

- Se desea almacenar cierta información sobre personas. En concreto, interesa saber:
  - NIF: Número de Identificación Fiscal
  - Altura: Expresada en centímetros.
  - Edad: Expresada en años.
- Por defecto, una persona para la que se no especifique dicha información se asume que tendrá 25 años, NIF: 11111111A y 175 cm de altura.
- Se pide:
  - Diseña la clase **Persona** con sus atributos, constructores, consultores y modificadores.
  - Diseña la clase **TestPersona** que cree dos personas, una con la información por defecto y otra con la que tú mismo inventes, y que muestre la información por pantalla.

Ejercicio Herencia: Yogures

- En primer lugar, diseña la clase **Yogur** sabiendo que un yogur siempre tiene 120.5 calorías. Se requiere implementar un método consultor para obtener sus calorías.
- A continuación, diseña la clase **YogurDesnatado** sabiendo que siempre tiene la mitad de calorías que un Yogur normal.
- Finalmente, construye una clase **TestYogures** con un método principal que cree un objeto Yogur y un YogurDesnatado y muestre sus calorías.

Ejercicio Herencia: Felino

- Dada la clase Felino, diseñar la clase Gato de manera que cuando hable, emita un maullido.

```
public class Felino {
   protected String loQueHablo;
   public Felino(){
         loQueHablo = "Soy un Felino";
   }
   public String habla(){ return loQueHablo;}
}
```

- La solución deberá incorporar algún mecanismo de herencia para reutilizar el código ya desarrollado (clase Felino).

Ejercicio de Herencia: Personas

- Diseña una jerarquía de clases apropiada para diferenciar entre una **Persona**, un **Ingeniero** y un **IngenieroInformatico**, sabiendo que:
- Una Persona puede *hablar* y *comer.*
- Un ingeniero, además, puede *razonar y trabajarEnGrupo*
- Un ingeniero informático, además, puede *crearPrograma.*
- Se pide:
  - Construye las clases necesarios junto con los correspondientes métodos cuya implementación se deja libremente al alumno.

Ejercicio: Diseño de Constructores

- Dadas las clases Base y Derivada:

```
public class Base{
    public int bPublico; int bProtegido;  private int bPrivado;
}
public class Derivada extends Base{
   public int dPublico; private int dPrivado;
}
```

- Diseñar un constructor de Base con 3 parámetros y un constructor de Derivada con 5 parámetros.

Ejercicio Interfaces: Arrancar

- Se desea diseñar una aplicación que modele un conjunto de vehículos. Todo vehículo incorporado a la aplicación debe **obligatoriamente** implementar el método **arrancar** que no devuelve nada.
- Considerar únicamente la definición de un Coche y una Moto.

1. Plantear la solución empleando el mecanismo de interfaces.
2. ¿Se podría plantear la solución empleando otros mecanismos?
3. Diseñar la clase **Arrancador** que tiene un único método **arrancaVehiculos** que, dado un array de Vehículos, los arranca.

Ejercicio: Clase Cilindro (I)

- Dadas las siguientes clases Java:

```
public class Circulo {
      private String tipo;
              double radio;

      public Circulo(double r) {
          this.radio = r;
          this.tipo = "Circulo";
      }
      public Circulo(double r, String t) {this.radio = r; this.tipo = t;}

      public double area(){ return Math.PI * radio * radio;}
      public double perimetro(){ return 2 * Math.PI * radio;}
      public String toString(){ return "Circulo de radio "+radio+"\n";}
}
```

Ejercicio: Clase Cilindro (II)

```
public class Cilindro extends Circulo {
    private double altura;

    public Cilindro(double radioBase, double altura){...}

     public double area(){
        return 2*Math.PI*radio*radio + 2*Math.PI*radio*altura;
     }
     public double volumen(){
         return Math.PI*radio*radio*altura;
     }
}
```


Ejercicio: Clase Cilindro (III)

1. En la clase Circulo, ¿Qué modificador de visibilidad se le debe de asignar al atributo radio para que sea accesible desde la clase Cilindro y favorezca el principio de Ocultación de Información?
2. De las siguientes implementaciones del Constructor de Cilindro, una de ellas es incorrecta, ¿Cuál?, ¿Por qué?
   a) super(radioBase,"Cilindro"); this.altura=altura;
   b) super.radio = radioBase; super.tipo = "Cilindro"; this.altura=altura;
3. En la clase Cilindro, modificar las implementaciones de los métodos área y volumen para favorecer el principio de reutilización de software.
4. Redefínase la clase Cilindro para que en lugar de SER UN Circulo, tenga un Circulo como base. No se permite modificar la especificación del constructor de Cilindro.

Ejercicio: losAnimales (1/3)

- Dadas las siguientes clases:

```
public class Animal {
   public void sonido(){ System.out.println("Grunt");}
}
public class Muflon extends Animal {
   public void sonido(){ System.out.println("MOOOO!");}
   public void salto(){ System.out.println("hop!");}
}
public class Armadillo extends Animal {}
public class Guepardo extends Animal {
   public void sonido(){ System.out.println("Groar!");}
}
```

Ejercicio: losAnimales (2/3)

```
public class TestIAnimal {
   public static void main(String[] args){
         adoptaAnimal(new Armadillo());
          Object o = new Armadillo();
         Armadillo a1 = new Animal();
         Armadillo a2 = new Muflon();
   }
   private static void adoptaAnimal(Animal a){
         System.out.println("Ven, cachorrito!");
   }
}
```

- ¿Qué instrucciones provocan error de compilación?

Ejercicio: losAnimales (3/3)

- Dado el siguiente extracto de código:

```
public class Test2Animal {
   public static void main(String[] args){
        Animal a = new Armadillo(); a.sonido();
        a = new Muflon(); a.sonido();
        a = new Guepardo(); a.sonido();
   }
}
```

- ¿Cuál es el resultado de su ejecución?

Conversor de Divisas: Errores

- Busca los errores en las siguientes clases Java:

```
private interface ConversorDivisa
{
   double convierte(Double cantidad);
}
```

```
public class EuroDolarConversorDivisa extends ConversorDivisa
{
  private double conviert(float cantidad){
     return cantidad * 1.50;
   }

   public void toString(){
      return "Conversor de Euros a Dolares";
    }
}
```


La Lavadora

- Sean las siguientes clases e interfaces Java:

```
public class ElectroDomestico
{
   private int corriente;
   private double consumo;
    public ElectroDomestico(int corriente, double consumo){
      this.corriente = corriente;
      this.consumo = consumo;
       }
}
```

```
public interface TiposCorriente
{
    public static final int TIPO_220V = 1;
    public static final int TIPO_125V = 2;
}
```

- Diseña la clase Lavadora, sabiendo que siempre funciona a 220 v y siempre consume 4500 u.e.

Actores y Películas: Errores

- Corrige el código para que *mostrarReparto* funcione:

```
public abstract class Persona {
    private String nombre;
    public Persona(String nombre) {
       this.nombre = nombre;
}}
```

```
public class Actor extends Persona {
private String pelicula;
public Actor(String nombre, String pelicula) {
    this.nombre = nombre;
    this.pelicula = pelicula;
}}
```

```
public class Peliculas {
  public static void mostrarReparto(Actor lista[], String pelicula) {
   for (int i = 0; i <= lista.size; i++)
      if (lista[i].pelicula == pelicula) System.out.println(lista[i].toString());
   }
}
```


Ejercicio cortesía de Oscar Sapena