Prueba 1 de Evaluación Continua
Análisis y Diseño de Software (2016/2017)

Contesta el ejercicio 1 y 2 en esta hoja, y el 3 en el espacio al dorso

Apellidos: **Nombre:**

Ejercicio 1: Ciclo de vida software y Análisis de requisitos (2,25 puntos)

1. Enumera tres tipos de requisitos No Funcionales (0,75 puntos)

Rendimiento, Seguridad, Recursos, etc.

2. Por regla general, cuál de los siguientes tipos de mantenimiento requiere un mayor esfuerzo: (0,7 puntos)
   a. Mantenimiento Correctivo
   b. Mantenimiento Perfectivo

   Justifica brevemente la respuesta:

En el caso del mantenimiento perfectivo, los cambios añaden nuevos requisitos, lo cual a su vez implica cambios en el diseño, codificación, realizar pruebas, etc. En el mantenimiento correctivo dependiendo del fallo puede haber o no cambios de diseño, y habitualmente únicamente implica cambios de código.

3. De los siguientes requisitos indica si son Funcionales (**F**) o No Funcionales (**NF**) (0,8 puntos)
   a. La aplicación permitirá crear nuevos tipos de productos y editar los existentes. **(F)**
   b. La aplicación se podrá ejecutar en Windows, Linux, y Mac OS. **(NF)**
   c. Todos los datos que se almacenen en disco estarán cifrados mediante AES-256 **(NF)**
   d. El listado de usuarios se mostrará ordenado, en primer lugar por apellidos y en segundo lugar por nombre. **(F)**

Ejercicio 2: Herencia y polimorfismo (3 puntos)

Dadas las clases Java de la derecha, indica cuál de las siguientes líneas de código numeradas contiene algún error e indica la razón del error (debajo de la línea), e indica qué valor se imprime en caso de ser correcta:

1. Pieza p4 = new Peon(4);

Correcta

2. Peon p5 = new Peon(5);

Correcta

3. Pieza p = new Pieza();

Incorrecta. Pieza es abstracta

4. System.out.println(p5.mejorQue(p4));

Salida: true

5. System.out.println(p4.mejorQue(p5));

Incorrecta. Pieza no tiene el método mejorQue

6. System.out.println(p4.getValor());

Salida: 4

```
abstract class Pieza {
   public int getValor() { return -1; }
}
class Peon extends Pieza {
   private int valor;
   public Peon(int v) { valor = v; }
   public int getValor() { return valor;}
   public boolean mejorQue(Pieza p) {
       return valor > p.getValor();
   }
}
```

Prueba 1 de Evaluación Continua
Análisis y Diseño de Software (2016/2017)

Contesta el ejercicio 1 y 2 en esta hoja, y el 3 en el espacio al dorso

Ejercicio 3: Diagramas de clase (4,75 puntos)

Como parte del diseño de una aplicación para una empresa de transporte de mercancías se debe incluir que los envíos tienen un identificador numérico y están compuestos de varios bultos que pueden ser voluminosos, pesados o frágiles.
Los voluminosos se caracterizan por sus tres dimensiones (ancho, largo y alto). Los pesados se caracterizan por su peso en kilogramos. Los frágiles tienen una descripción del contenido y el valor por el cual se debe asegurar su envío. Todos los bultos tienen una tarifa de envío que se calcula de manera distinta para cada tipo de bulto y el precio del envío es la suma de las tarifas de envío de sus bultos más un extra que depende del número total de bultos. Cada envío debe asignarse a un transportista que tenga capacidad para gestionarlo. Los transportistas pueden encargarse de gestionar varios envíos pero ninguno de ellos puede estar formado por bultos que de manera conjunta superen los límites máximos de volumen, pesado o valor asegurado.

Se pide:

a. Realiza el diagrama de clases que describe la parte del diseño descrita arriba. No es necesario incluir constructores, getters ni setters. (4 puntos)
b. Añade los métodos necesarios para: (0,75 puntos)
    1. calcular el precio de un envío y las tarifas de envío de los bultos.
    2. conocer si un transportista puede gestionar determinado envío
    3. añadir un bulto a un envío