UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA – ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

71901072 – PROGRAMACIÓN ORIENTADA A OBJETOS (GRADO EN INGENIERÍA INFORMÁTICA / TECNOLOGÍAS DE LA INFORMACIÓN)

JUNIO/SEPTIEMBRE 2015 – MODELO C – NO ESTÁ PERMITIDO EL USO DE MATERIAL ADICIONAL

PARTE TEÓRICA - TEST [2,5 PUNTOS]:

Solo una de las respuestas es válida. Las respuestas correctas se puntuarán con +1.0, mientras que las respondidas de manera incorrecta se puntuarán con -0.25. Las no contestadas no tendrán influencia ni positiva ni negativa en la nota.

Pregunta 1: Según el texto de la bibliografía básica de la asignatura, los constructores …

a. Almacenan datos de manera persistente dentro de un objeto.
b. Implementan el comportamiento de un objeto.
c. Son responsables de garantizar que un objeto se configure apropiadamente en el momento de usarlo por primera vez, siempre y cuando haya sido creado previamente.
d. Ninguna de las anteriores.

Pregunta 2: En BlueJ, ¿qué nos permite experimentar con expresiones Java?

a. El Pad Code
b. El Patch Code
c. El Pan Code
d. Ninguna de las anteriores.

Pregunta 3: Por modularización entendemos …

a. El proceso de dividir un todo en partes laxamente definidas que puedan construirse y examinarse en conjunto y que interactúen de formas bien definidas.
b. El proceso de dividir un todo en partes bien definidas que puedan construirse y examinarse en conjunto y que interactúen de formas bien definidas.
c. El proceso de dividir un todo en partes bien definidas que puedan construirse y examinarse por separado y que interactúen de formas bien definidas.
d. Ninguna de las anteriores.

Pregunta 4: Queremos compilar el siguiente código que se puede encontrar en el texto base de la asignatura y que ha sido modificado convenientemente. ¿Cuál es el resultado que obtenemos al compilar?

```
1     public class MailItem
2     {
3         private static String from;
4         private String to;
5         private String message;
6
7         public static MailItem (String from, String to, String m)
8         {
9             this.from = from;
10            this.to = to;
11            this.message = m;
12        }
13    }
```

a. El código compila sin errores
b. Produce un error de compilación en la línea 3.
c. Produce un error de compilación en la línea 11.
d. Ninguna de las anteriores.

Pregunta 5: Queremos compilar el siguiente código que se puede encontrar en el texto base de la asignatura y que ha sido modificado convenientemente. Al compilar, BlueJ nos da error de compilación. ¿Qué deberemos cambiar para que el programa compile?

```
1     import java.util.ArrayList;
2
3     public class MusicOrganizer
4     {
5         private ArrayList <String> files;
6         private MusicPlayer player;
7
8         public MusicOrganizer ()
9         {
10            files = new ArrayList <String> ();
11            player = new MusicPlayer();
12        }
13
14        public void startPlayingFile (int index)
15        {
16            String filename = files.get(index);
17            player.startPlaying (filename);
18        }
19
20        public void stopPlaying ()
21        {
22            player.stop();
23        }
24    }
```

a. Definir la clase `MusicPlayer` convenientemente, con al menos los métodos `stopPlaying()` y `startPlayingFile (int index)`, e importarla (si fuese necesario) mediante la instrucción `import MusicPlayer;`
b. Modificar la línea 10 para que quede así: `files = new ArrayList <String> (0);`
c. Modificar la línea 6 para que quede así: `public MusicPlayer player;`
d. Hay que aplicar los cambios indicados en a, b y c, puesto que si alguno no se aplicase, el código no compilaría.

Pregunta 6: Queremos compilar el siguiente código que se puede encontrar en el texto base de la asignatura. Al compilar, BlueJ podría darnos un error en tiempo de compilación y/o un error en tiempo de ejecución. ¿Cómo deberemos proceder para que el código compile y se ejecute correctamente?

```
1     public class Ejemplo
2     {
3         public static Vehicle v;
4         public static Car c;
5         public static Bicycle b;
6
7         public static void main ()
8         {
9             c = new Car();
10            v = c;
```

```
11              b = (Bicycle) c;
12              b = (Bicycle) v;
13               System.out.println("Funciona");
14          }
15      }
```

a. Con independencia de cómo definamos las clases `Vehicle`, `Car` y `Bicycle`, siempre nos dará un error en tiempo de ejecución en la línea 12.
b. Con independencia de cómo definamos las clases `Vehicle`, `Car` y `Bicycle`, siempre nos dará un error en tiempo de compilación en la línea 11.
c. Si definimos que `Car extends Vehicle` y que `Bicycle extends Car`, conseguimos eliminar todos los errores del código y muestra el mensaje por pantalla "Funciona".
d. Si definimos que `Vehicle extends Bicycle` y que `Car extends Vehicle`, conseguimos eliminar todos los errores del código y muestra el mensaje por pantalla "Funciona".

Pregunta 7: Queremos compilar el siguiente código que se puede encontrar en el texto base de la asignatura. ¿Qué ocurre al compilarlo con BlueJ?

```
1     import java.util.List;
2     interface Actor
3     {
4         void act (List <Actor> newActors);
5         boolean isActive();
6     }
```

a. Compila, no proporcionando ningún error en tiempo de compilación.
b. Compila, no proporcionando ningún error en tiempo de compilación, del mismo modo que también compilaría si prescindiéramos de la línea 1.
c. No compila. Hay que modificar la línea 2 quedando de la siguiente manera: `public interface Actor`
d. No compila. Hay que modificar la línea 4 quedando de la siguiente manera: `public void act (List <Actor> newActors);`

Pregunta 8: Según el texto de la bibliografía básica de la asignatura …

a. El tipo estático de una variable v y el tipo dinámico de una variable v coinciden siempre.
b. El tipo estático de una variable v se modifica automáticamente al modificar el tipo dinámico de la variable.
c. El tipo estático siempre coincide con el tipo declarado en la instrucción de declaración de la variable.
d. Ninguna de las anteriores.

Pregunta 9: Según el texto de la bibliografía básica de la asignatura, con respecto a la igualdad de referencias podemos afirmar …

a. La igualdad de referencia tiene en cuenta el contenido de los objetos.
b. No es posible usar la igualdad de referencias para realizar comparaciones de cadenas de caracteres.
c. El método `equals` heredado de la clase `Object` no permite comprobar que existe una igualdad de referencias.
d. Ninguna de las anteriores.

Pregunta 10: Queremos compilar el siguiente código que se puede encontrar en el texto base de la asignatura y que hemos modificado. ¿Qué ocurre al compilarlo con BlueJ?

```
1     import java.util.List;
2
3     public abstract class Animal
4     {
5         public void act (List<Animal> newAnimals, char animals) {}
6         public static void act (List<Animal> newAnimals, int animals);
7         abstract public static void act (List<Animal> newAnimals, String animals);
8     }
```

a. Las líneas 5 y 6 provocan errores de compilación.
b. Las líneas 5 y 7 provocan errores de compilación.
c. Las líneas 6 y 7 provocan errores de compilación.
d. Las líneas 5, 6 y 7 provocan errores de compilación.

Pregunta 11: Queremos compilar el siguiente código que se puede encontrar en el texto base de la asignatura y que hemos modificado. ¿Qué ocurre al compilarlo con BlueJ?

```
1     public class Ejemplo
2     {
3         public static void main (String []args)
4         {
5             int[] numeros = new int[] {1,2,3};
6
7             System.out.print(numeros);
8             System.out.print(numeros.length);
9             System.out.print(numeros.last);
10            System.out.print(numeros.first);
11        }
12    }
```

a. Se produce un error de compilación en las líneas 8, 9 y 10
b. Se produce un error de compilación en las líneas 9 y 10.
c. Se produce un error de compilación en línea 10.
d. Ninguna de las anteriores.

Pregunta 12: ¿Cuál de las siguientes opciones permite modificar una cadena declarada como `String input;`?

a. `input.toUpperCase();`
b. `input.trim();`
c. `input.startsWith("hola");`
d. Ninguna de las anteriores.

Pregunta 13: En relación a los conceptos de acoplamiento y cohesión, podemos afirmar …

a. Un alto grado de acoplamiento implica necesariamente un alto grado de cohesión.
b. Un bajo grado de acoplamiento no implica necesariamente un alto grado de cohesión.
c. En un diseño de clases perseguimos un bajo grado de cohesión y un bajo acoplamiento
d. Ninguna de las anteriores.

Pregunta 14: Queremos compilar el siguiente código que se puede encontrar en el texto base de la asignatura y que hemos modificado. El código compila sin causar ningún error de compilación, pero no muestra nada por pantalla. ¿Qué falta por añadir para que el código muestre algo por pantalla al crear un objeto de la clase `ImageViewer` dentro del entorno de BlueJ?

```
1     import java.awt.*;
2     import java.awt.event.*;
3     import javax.swing.*;
4
5     public class ImageViewer extends JFrame
6     {
7         public ImageViewer()
8         {
9             super("ImageViewer");
10            makeFrame();
11
12        }
13
14        private void makeFrame()
15        {
16            Container contentPane = getContentPane();
17            JLabel label = new JLabel("I am a label.");
18            contentPane.add(label);
19
20        }
21    }
```

a. Añadir la instrucción this.pack() en la línea 11.
b. Añadir la instrucción pack() en la línea 19.
c. No hay que añadir nada. Se puede ver el texto "I am a label" en pantalla.
d. Ninguna de las anteriores

Pregunta 15: Queremos compilar el siguiente código que se puede encontrar en el texto base de la asignatura y que ha sido modificado convenientemente. Se produce un error de compilación. ¿Qué línea es la que contiene un error, tal que si la modificamos convenientemente, el código compila y no provoca ningún error de compilación en BlueJ?

```
1     import java.io.*;
2
3     public class Ejemplo
4     {
5         public static void main () throws IOException
6         {
7             String filename = new String ("EJEMPLO");
8             try {
9                 throw IOException ();
10            }
11            catch (Exception e) {
12                System.out.println("Unable to save to "+ filename);
13            }
14        }
15    }
```

a. El error está en la línea 5
b. El error está en la línea 7
c. El error está en la línea 9
d. El error está en la línea 11

PARTE PRÁCTICA [6,5 PUNTOS]:

La práctica del presente curso ha sido una terminal punto de venta (por sus siglas, TPV) que ha servido para estudiar y practicar los mecanismos de la Programación Orientada a Objetos.

Definición de TPV y Características

Según la Wikipedia (www.wikipedia.org), un terminal punto de venta (cuyo acrónimo es TPV hace referencia al dispositivo y tecnologías que ayudan en la tarea de gestión de un establecimiento comercial de venta al público que puede contar con sistemas informáticos especializados mediante una interfaz accesible para los vendedores.

Los TPV permiten la creación e impresión del tique de venta mediante las referencias de productos, realizan diversas operaciones durante todo el proceso de venta, así como cambios en el inventario. También generan diversos reportes que ayudan en la gestión del negocio. Los TPV se componen de una parte hardware (dispositivos físicos) y otra software (sistema operativo y programa de gestión).

En nuestro caso concreto, el hadware será un ordenador tipo PC o similar y nuestro software será una aplicación desarrollada en Java que se ejecutará sobre dicho equipo.

Funcionalidades

Los TPV permiten la implementación desde labores simples de gestión de una venta, hasta operaciones más complejas como es la gestión de almacén o inventario, gestión de facturación o gestión de clientes. En esta práctica, se propondrá diferentes funcionalidades para el sistema de gestión del TPV:

- Llevar un control de diferentes elementos que existen en nuestro establecimiento. Así, los productos habrán de estar identificados en el sistema por, al menos, los siguientes datos: código descriptivo (por ejemplo, el código de barras), descripción, precio unitario sin IVA, IVA aplicable, precio unitario con IVA, cantidad disponible en stock. □
- El sistema debe permitir dar de alta nuevos productos, dar de baja productos existentes así como modificar los datos del mismo. □
- Realizar la importación y/o exportación de los productos a/desde ficheros (u otro método similar que el alumno considere en su lugar). □
- Llevar un control de las diferentes ventas que se producen. Así, el sistema deberá llevar un control de tickets generados, de modo que cada ticket se considerará una venta. Cada ticket tiene que tener un código de identificador único. Una forma de generar un código único podría ser de la forma AAAAMMDDHHMM, donde AAAA es el año en curso, MM el mes en que se genera la venta, DD el día de la venta, HHMM las horas y minutos en las que se inicia la venta. Asumiremos que sólo hay un TPV, por lo que no procede que haya dos ventas simultáneas. □
- La venta consistirá en la inclusión de varios productos en una lista, generándose una línea por cada producto vendido. Cada línea mostrará, al menos, el código del producto, la descripción del producto, la cantidad de unidades vendidas, el precio unitario con IVA, el IVA que se le aplica y el importe total de la venta de ese producto según el número de unidades vendidas. □
- El proceso de venta implicará automáticamente un proceso de actualización del inventario. De este modo, si se introduce un□código que no pertenece a ningún producto, o si se introduce un producto que no existe en stock (o más unidades de las existentes), el programa deberá mostrar los errores correspondientes.
- El sistema deberá permitir también introducir un producto a vender en el ticket haciendo una

búsqueda por la descripción, además de con el código que lo identifica. □

- Realizar la importación y/o exportación de los diferentes tickets de ventas a/desde ficheros (u otro método similar que el alumno considere en su lugar). □
- Llevar un control de los diferentes clientes que trabajan con el establecimiento comercial. Así, los clientes habrán de estar identificados en el sistema por, al menos, los siguientes datos: código identificativo del cliente, NIF o CIF, nombre y apellidos / razón social, domicilio, fecha de alta en el sistema. □
- El sistema debe permitir dar de alta nuevos clientes, dar de baja clientes existentes así como modificar los datos de los mismos. □
- Realizar la importación y/o exportación de los clientes a/desde ficheros (u otro método similar que el alumno considere en su lugar). □
- Permitir generar facturas a partir de un conjunto de tickets. Puede generar facturas agrupando diferentes tickets siempre y cuando pertenezcan al mismo cliente y se han realizado dentro del mismo periodo fiscal (es decir, dentro del mismo año). La información que irá en cada factura deberá ser, al menos, la siguiente: número de la factura (identificador único), CIF del vendedor, razón social del vendedor, fecha de emisión de la factura, datos del cliente (los indicados con anterioridad, excepto la fecha de alta en el sistema), listado de los diferentes productos vendidos (especificando para cada producto, el ticket en el que se encuentra, su cantidad vendida e importe total) así como suma del total de la venta (valor total de la factura). □
- Realizar la importación y/o exportación de las facturas a/desde ficheros (u otro método similar que el alumno considere en su lugar). □
- Generación de listados: se deberá implementar, al menos, la emisión de tres listados, a saber: ventas realizadas en un intervalo de tiempo determinado agrupadas estas ventas por clientes, ventas realizadas en un intervalo de tiempo determinado a un cliente y ranking de productos más vendidos en un intervalo de tiempo determinado. □

a) **[1,0 puntos]** Diseñar utilizando un paradigma orientado a objetos, los elementos necesarios para la aplicación explicada de la práctica durante el curso. Es necesario identificar la estructura y las relaciones de herencia (mediante el uso de un diagrama de clases) y de uso de las clases necesarias para almacenar y gestionar esta información. Debe hacerse uso de los mecanismos de herencia siempre que sea posible. Se valorará un buen diseño que favorezca la reutilización de código y facilite su mantenimiento.

b) **[1,0 puntos]** Implementa un método (o métodos) que permitan la importación (cargar al programa) de los diferentes tickets de ventas desde fichero (u otro método similar que el alumno considere en su lugar). Justifíquese las opciones y decisiones que se tomen.

c) **[2,0 puntos]** Implementa un método (o métodos) que implementen el proceso de venta, junto con la actualización del inventario. De este modo, si se introduce un código que no pertenece a ningún producto, o si se introduce un producto que no existe en stock (o más unidades de las existentes), el programa deberá mostrar los errores correspondientes. Justifíquese las opciones y decisiones que se tomen.

d) **[2,5 puntos]** Proporcione un método (o métodos) que permita mostrar por pantalla un formulario básico en **modo gráfico** que permita recoger los parámetros necesarios para dar de alta un nuevo cliente en el sistema. El método deberá comprobar si el cliente existe, y si existe, mostrar el correspondiente mensaje por pantalla. Si no existe, procederá a dar de alta al cliente. Justifíquese las opciones y decisiones que se tomen.