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 D – 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: ¿Cuál es el resultado de compilar y ejecutar el siguiente código?

```
import java.util.*;
public class Test {
    public static void main (String[ ] Args) {
        ArrayList<String> lista=new ArrayList<String>();
        lista.add("uno");
        lista.add("dos");
        lista.add("tres");
        for(String valor:lista){
            System.out.print("Elimino "+valor+" - ");
            lista.remove(valor);
        }
    }
}
```

a. Muestra en consola: "Elimino uno - "
b. Muestra en consola: "Elimino uno - " y una excepción del tipo ConcurrentModificationException
c. Muestra en consola: "Elimino uno – Elimino dos – Elimino tres - "
d. Muestra en consola una excepción del tipo ConcurrentModificationException

Pregunta 2: ¿Cuál es el resultado de compilar y ejecutar el siguiente código?

```
import java.util.*;
public class Test {
    public static void main (String[ ] Args) {
        ArrayList<String> lista=new ArrayList<String>();
        lista.add("uno");
        lista.add("dos");
        lista.add("tres");
        Iterator<String> it=lista.iterator();
        while(it.hasNext()){
            String valor=it.next();
            System.out.print("Elimino "+valor+" - ");
            it.remove();
        }
    }
}
```

a. Muestra en consola: "Elimino uno - "
b. Muestra en consola: "Elimino uno - " y una excepción del tipo ConcurrentModificationException
c. Muestra en consola: "Elimino uno – Elimino dos – Elimino tres - "
d. Muestra en consola una excepción del tipo ConcurrentModificationException

Pregunta 3: ¿Cuál de las siguientes afirmaciones es cierta?

a. El término acoplamiento describe lo bien que una unidad de código se corresponde con una tarea lógica o con una entidad.
b. El término cohesión describe la interconexión de las clases.
c. Se debe tender a un acoplamiento débil.
d. Se debe tender a un acoplamiento fuerte.

Pregunta 4: ¿Cuál es el resultado de compilar y ejecutar el siguiente código?

```
public class Padre
{
    private int x;
    public Padre(int valor) { x = valor;}
}
public class Hijo extends Padre
{
    public Hijo(){}
}
public class Test {
    public static void main (String[ ] Args) {
        Hijo h=new Hijo();
        System.out.println("Clase instanciada");
    }
}
```

a. Muestra en consola: "Clase instanciada "
b. Error de compilación. Hay que poner super() en el constructor de la clase Hijo para acceder al constructor por defecto de la clase Padre.
c. Error de compilación. Hay que escribir explícitamente el constructor Padre() en la clase Padre.
d. Error de compilación. Hay que poner super() en el constructor de la clase Hijo para acceder al constructor por defecto de la clase Padre y hay que escribir explícitamente el constructor Padre() en la clase Padre.

Pregunta 5: ¿Cuál es el resultado de compilar y ejecutar el siguiente código?

```
public class Padre
{
    void metodoA(){
        System.out.println("Ejecuto el método A de la clase Padre");
    }
}
public class Hijo extends Padre
{
    void metodoA(){
        System.out.println("Ejecuto el método A de la clase Hijo");
    }
}
public class Test {
    public static void main (String[ ] Args) {
        Padre h=new Hijo();
        h.metodoA();
    }
}
```

a. Muestra en consola: "Ejecuto el método A de la clase Hijo "
b. Muestra en consola: "Ejecuto el método A de la clase Padre "
c. Error de compilación. Tipos incompatibles.
d. Ninguna de las anteriores.

Pregunta 6: ¿Cuál es el resultado de compilar y ejecutar el siguiente código?

```
public class Padre
{
    void metodoA(){
        System.out.println("Ejecuto el método A de la clase Padre");
    }
}
public class Hijo extends Padre
{
    void metodoA(){
        System.out.println("Ejecuto el método A de la clase Hijo");
    }
}
public class Test {
    public static void main(String[] args) {
        Hijo h=new Padre();
        if(h instanceof Hijo)
            System.out.println("Instancia de la clase Hijo");
        else
            System.out.println("Instancia de la clase padre");
    }

}
```

a. Muestra en consola: "Instancia de la clase Hijo "
b. Muestra en consola: "Instancia de la clase Padre"
c. Error de compilación. Tipos incompatibles.
d. Ninguna de las anteriores.

Pregunta 7: Indica cual de las siguientes afirmaciones es correcta.

a. Declarar un campo o un método protegido (protected) permite acceder directamente a él desde las subclases directas o indirectas.
b. Declarar un campo o un método protegido (protected) permite acceder directamente a él únicamente desde las subclases directas.
c. Los miembros definidos como private en una subclase son accesibles para los objetos de otras clases.
d. Los miembros definidos como private en una superclase son accesibles para los objetos de sus subclases.

Pregunta 8: ¿Cuál es el resultado de compilar y ejecutar el siguiente código?

```
public class Padre
{
    void metodoA(){
        System.out.println("Ejecuto el método A de la clase Padre");
    }
}
public class Hijo extends Padre
{
    void metodoA(){
        System.out.println("Ejecuto el método A de la clase Hijo");
    }
}
public class Test {
    public static void main(String[] args) {
        Hijo h=(Hijo)new Padre();
        if(h instanceof Hijo)
```

```
            System.out.println("Instancia de la clase Hijo");
        else
            System.out.println("Instancia de la clase padre");
    }

}
```

a. Muestra en consola: "Instancia de la clase Hijo "
b. Muestra en consola: "Instancia de la clase Padre"
c. Error de compilación. Tipos incompatibles.
d. Error en tiempo de ejecución. Lanza una excepción ClassCastException.

Pregunta 9: ¿Cuál es el resultado de compilar y ejecutar el siguiente código?

```
public class Padre
{
    void metodoA(){
        System.out.print("Ejecuto el método A de la clase Padre. ");
    }
}
public class Hijo extends Padre
{
    void metodoA(){
        System.out.print("Ejecuto el método A de la clase Hijo. ");
        super.metodoA();
    }
}
public class Test {
    public static void main(String[] args) {
        Padre p=new Hijo();
        p.metodoA();
    }

}
```

a. Muestra en consola: "Ejecuto el método A de la clase Padre. "
b. Muestra en consola: "Ejecuto el método A de la clase Hijo."
c. Muestra en consola: "Ejecuto el método A de la clase Padre. Ejecuto el método A de la clase Hijo.
d. Ninguna de las anteriores.

Pregunta 10: ¿Cuál es el resultado de compilar y ejecutar el siguiente código?

```
public abstract class Padre
{
    abstract void metodoA();
}
public class Hijo extends Padre
{
    void metodoA(){
        System.out.print("Método A de Hijo. ");
    }
}
public class Hija extends Padre
{
   void metodoA(){
        System.out.print("Método A de Hija. ");
    }
}
import java.util.*;
public class Test {
    public static void main(String[] args) {
```

```
        List<Padre> lista=new ArrayList<Padre>();
        lista.add(new Hijo());
        lista.add(new Hijo());
        lista.add(new Hija());
        for(Iterator<Padre> it=lista.iterator();it.hasNext();){
            Padre p=it.next();
            p.metodoA();
        }
    }
}
```

a. Muestra en consola: "Método A de Hijo. Método A de Hijo. Método A de Hija."
b. Error de compilación. No se puede incluir un iterador dentro de un bucle for.
c. Error de compilación. Las clases Hijo e Hija deben declararse como abstractas.
d. Error en tiempo de ejecución. Es obligatorio hacer un cast en cada elemento extraído del ArrayList para invocar al método correcto

Pregunta 11: ¿Cuál es el resultado de compilar y ejecutar el siguiente código?

```
public abstract class Padre
{
    void metodoA(){
        this.metodoB();
    }
    abstract void metodoB();
}
public class Hijo extends Padre
{
    void metodoB(){
        System.out.print("Método B de Hijo. ");
    }
}
import java.util.*;
public class Test {
    public static void main(String[] args) {
        Hijo h=new Hijo();
        h.metodoA();
    }
}
```

a. Muestra en consola: "Método B de Hijo. "
b. Error de compilación. No se puede invocar un método abstracto desde la clase abstracta Padre
c. Error de compilación. La clase abstracta Padre debe declarar todos sus métodos como abstractos.
d. Ninguna de las anteriores.

Pregunta 12: ¿Cuál de las siguientes afirmaciones es falsa?

a. Una interfaz en Java es una especificación de un tipo que no define implementación para alguno de sus métodos.
b. Para que una subclase de una clase abstracta se transforme en concreta, debe proporcionar implementaciones para todos los métodos abstractos heredados.
c. El objetivo de una clase abstracta es servir como una superclase de otras clases.
d. Las llamadas a métodos de instancia no privados desde dentro de una superclase siempre se evalúan en el contexto más amplio del tipo dinámico del objeto.

Pregunta 13: ¿Cuál de las siguientes afirmaciones es falsa?

a. Las clases internas anónimas son una estructura muy útil a la hora de implementar escuchas de sucesos.

b. Una interfaz GUI se construye disponiendo componentes en pantalla. Los componentes se representan mediante objetos.
c. Para definir la colocación de los componentes de una GUI se utilizan gestores de diseño gráfico.
d. Un objeto puede escuchar los sucesos de los componentes extendiendo una clase abstracta de escucha de sucesos.

Pregunta 14: ¿Cuál de las siguientes afirmaciones es falsa?

a. La serialización permite leer y escribir en una única operación objetos completos, pero no jerarquías de objetos.
b. Una excepción no comprobada es un tipo de excepción cuyo uso no requiere ninguna comprobación por parte del compilador.
c. Una excepción comprobada es un tipo de excepción cuyo uso requiere comprobaciones adicionales por parte del compilador.
d. Una aserción es un enunciado de un hecho que debe ser cierto durante la ejecución del programa.

Pregunta 15: ¿Cuál de las siguientes afirmaciones es falsa?

a. Las pruebas son la actividad consistente en averiguar si un fragmento de código presenta el comportamiento deseado.
b. Un recorrido manual es la actividad consistente en analizar un segmento de código línea a línea mientras que se observan los cambios de estado y otros comportamientos de la aplicación.
c. Una prueba negativa es una prueba de un caso que se espera que funcione correctamente y que finalmente no funciona.
d. Si la condición definida en una aserción es falsa, decimos que la aserción ha fallado.

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]** Modifique aquellas clases que considere oportunas para incluir la funcionalidad de tener clientes VIP. A estos clientes se les aplicará un descuento del 10% en todos los productos.

c) **[2,0 puntos]** Modifique los métodos de venta y generación de facturas para aplicar el descuento en el precio final de cada producto en el caso en el que el cliente sea VIP.

d) **[2,5 puntos]** Proporcione un método (o métodos) que permitan mostrar por pantalla un listado, en **modo gráfico**, de los clientes VIPs ordenados de mayor a menor en función del gasto realizado.