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 2016 – MODELO C – NO ESTÁ PERMITIDO EL USO DE MATERIAL ADICIONAL

PARTE TEÓRICA - TEST [2,5 PUNTOS]:

El test consta de 14 preguntas y 2 preguntas adicionales de reserva. 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.

Las preguntas de reserva sólo tendrán utilidad en el caso de que alguna de las 14 preguntas iniciales del test sea anulada por cualquier circunstancia. Caso de ocurrir este hecho, si se produjera la anulación de alguna de las 14 preguntas iniciales, la primera pregunta de reserva sustituiría a la pregunta anulada. Caso de que una segunda pregunta de las 14 iniciales fuese anulada, entonces la segunda pregunta de reserva sustituiría a esta segunda pregunta anulada. En aquellos hipotéticos casos en los que se produjese la anulación de una tercera o sucesivas preguntas de las 14 iniciales, entonces sólo en ese caso, las preguntas tercera y sucesivas anuladas se considerarían como correctas (al no existir más preguntas de reserva que las sustituyan).

Pregunta 1: ¿De qué forma podemos invocar un método en el IDE BlueJ?

a. Haciendo clic con el botón derecho del ratón en un objeto, y seleccionando en el menú emergente el método correspondiente.
b. Haciendo clic con el botón derecho del ratón en una clase, y seleccionando en el menú emergente el método correspondiente.
c. Haciendo clic con el botón derecho del ratón en un objeto o clase indistintamente, y seleccionando en el menú emergente el método correspondiente.
d. Ninguna de las anteriores.

Pregunta 2: Según el texto de la bibliografía básica de la asignatura, ¿cuál de las siguientes afirmaciones **NO** es correcta?

a. Los campos almacenan datos de manera no persistente dentro de un objeto.
b. Los constructores son responsables de garantizar que un objeto se configure apropiadamente en el momento de crearlo por primera vez.
c. Los métodos implementan el comportamiento de un objeto; proporcionan su funcionalidad.
d. Ninguna de las anteriores.

Pregunta 3: 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 Prueba
2     {
3         public static void main (String args[])
4         {
5             String cadena1 = "ejemPLO";
6             String cadena2 = "EJEMPLO";
7             cadena1.toUpperCase();
8
9             if (cadena1.equals(cadena2))
10            {
11                System.out.println("Son iguales");
12            }
13            else
14            {
15                System.out.println("Son diferentes");
16            }
```

```
17        }
18    }
```

a. Se produce una excepción y la ejecución falla.
b. Se imprime por pantalla el mensaje: Son iguales.
c. Se imprime por pantalla el mensaje: Son diferentes.
d. Ninguna de las anteriores.

Pregunta 4: Según el texto de la bibliografía básica de la asignatura, ¿qué debilitaría la encapsulación?
a. Emplear el acceso protegido a los métodos de una clase.
b. Emplear el acceso protegido a los campos de una clase.
c. Emplear el acceso protegido a los constructores de una clase.
d. Ninguna de las anteriores.

Pregunta 5: Supongamos que reescribimos el ejemplo `BouncingBall` del libro de la forma en que se muestra a continuación:

```
1     public class BouncingBall {
2         int n;
3         public static void main (String args [])
4         {
5
6             if (n!=0)
7             {
8                 n = n + 1;
9                 System.out.println("El número es " + n);
10            }
11        }
12    }
```

El programa no compila. ¿Qué podemos cambiar para que funcione correctamente?
a. Sustituir la línea 2 por la siguiente: `int n = 0;`
b. Insertando en la línea 5 lo siguiente: `n = 0;`
c. Sustituir la línea 3 por lo siguiente: `public static void main ()`
d. Ninguna de las anteriores.

Pregunta 6: Sea el siguiente código modificado de la clase `MusicOrganizer` mostrada en el libro base:

```
1     import java.util.*;
2     public class MusicOrganizer {
3         public static void main (String args []) {
4             ArrayList <String> a = new ArrayList (5);
5             for (int i=0; i<=5; i++)
6             {
7                 a.add("Hola");
8             }
9             System.out.println(“Funciona”);
10        }
11    }
```

La compilación produce un warning. ¿Cómo podemos resolver ese problema?
a. Sustituyendo la línea 4 por: `ArrayList a = new ArrayList (5);`
b. Sustituyendo la línea 4 por: `ArrayList a = new ArrayList <String> (5);`
c. Sustituyendo la línea 4 por: `ArrayList <String> a = new ArrayList <String> (5);`
d. Tanto (a) como (c) resuelven el problema.

Pregunta 7: Queremos compilar el siguiente código que se puede encontrar en el texto base de la asignatura,

convenientemente modificado. ¿Qué ocurre al compilarlo con BlueJ?

```
1     import java.util.List;
2     public abstract class Animal
3     {
4         private boolean alive;
5         private String field;
6         private String location;
7
8         public abstract Animal(String field, String location)
9         {
10            alive = true;
11            this.field = field;
12            this.location = location;
13        }
14
15        abstract public void act(List<Animal> newAnimals);
16    }
```

a. Compila, no proporcionando ningún error en tiempo de compilación.
b. No compila. Se soluciona sustituyendo la línea 15 por la siguiente: `public void act(List<Animal> newAnimals);`
c. No compila. Se soluciona sustituyendo la línea 15 por la siguiente: `public abstract void act(List<Animal> newAnimals);`
d. Ninguna de las anteriores.

Pregunta 8: Según el texto de la bibliografía básica de la asignatura, las subclases de `Error` suelen estar reservadas para …
a. Los errores del sistema en tiempo de ejecución.
b. Los errores del sistema en tiempo de compilación.
c. Los errores de programación en tiempo de compilación.
d. Ninguna de las anteriores.

Pregunta 9: Según el texto de la bibliografía básica de la asignatura, ¿cuál es el método imprescindible y que ha de implementarse siempre de la interfaz `Serializable` cuando queremos implementar la serialización?
a. El método IOWrite.
b. El método InputOutputWrite.
c. El método WriteOutput.
d. Ninguno de los anteriores.

Pregunta 10: Queremos compilar el siguiente código que se puede encontrar en el texto base de la asignatura y que ha sido convenientemente modificado. El programa compila sin problemas pero no muestra por pantalla el texto "Ejemplo de texto". ¿Qué tendríamos que añadir / modificar para se mostrase?

```
1     import java.awt.*;
2     import java.awt.event.*;
3     import javax.swing.*;
4
5     public class ImageViewer
6     {
7         private JFrame frame;
8         public ImageViewer(){
9             makeFrame();
10        }
11
```

```
12        private void makeFrame(){
13            frame = new JFrame("ImageViewer");
14            Container contentPane = frame.getContentPane();
15            JLabel label = new JLabel("Ejemplo de texto");
16            contentPane.add(label);
17            frame.setVisible(true);
18        }
19    }
```

a. Añadir entre las líneas 9 y 10 lo siguiente: `frame.setVisible(true);` y eliminar la línea 17.
b. Añadir entre las líneas 15 y 16 lo siguiente: `frame.pack();`
c. Añadir entre las líneas 16 y 17 lo siguiente: `frame.pack();`
d. Sustituir la línea 17 por lo siguiente: `frame.pack();`

Pregunta 11: Según el texto de la bibliografía básica de la asignatura, ¿qué define el término acoplamiento?
a. La bondad de la correspondencia entre una unidad de código y una tarea lógica o entidad.
b. La interconexión existente entre clases, buscando un acoplamiento lo menor posible.
c. La capacidad de un objeto de comportarse como otra clase de la cual proviene.
d. Ninguna de las anteriores.

Pregunta 12: Sea el siguiente fragmento de código modificado de la clase `MailItem` mostrada en el libro de texto:

```
1     public class MailItem {
2         static String from;
3         static String to;
4         static String message;
5         int number;
6
7         public static void main (String args[]) {
8             MailItem m = new MailItem("Hola", "Adios", "Luego", 3);
9             System.out.println("Funciona");
10        }
11
12        public MailItem (String from, String to, String message, int number){
13            this.from = from;
14            this.to = to;
15            this.message = message;
16            this.number = number;
17        }
18    }
```

¿Cuál es el resultado de ejecutar el código?
a. Se produce un error de compilación
b. Se produce un error de ejecución en la línea 9.
c. Se produce un error de ejecución en la línea 16.
d. Ninguna de las anteriores.

Pregunta 13: Según el texto de la bibliografía básica de la asignatura, ¿qué podemos afirmar sobre las pruebas de regresión?
a. La modificación de software acarrea con mucha facilidad errores adicionales de software.
b. Las pruebas de regresión sobre un módulo determinado tras haberse hecho una modificación del código pueden obviarse si no se realizan cambios en ese módulo.
c. Los marcos de regresión permiten automatizar las pruebas de regresión.
d. Si no se automatizan, es más probable que las pruebas de regresión se lleven a cabo.

Pregunta 14: Según el texto de la bibliografía básica de la asignatura, ¿qué podemos afirmar sobre el concepto

de sustitución?
a. Pueden utilizarse objetos de un supertipo en cualquier lugar en el que se espera objetos de un subtipo.
b. Permite crear objetos de un clase que es abstracta.
c. Permite que una variable almacena objetos de diferentes tipos (en concreto, del tipo declarado o de cualquier supertipo del tipo declarado).
d. Ninguna de las anteriores

RESERVA 1: Si una clase `B` extiende una clase abstracta `A` que tiene un método abstracto `met`, ¿qué podemos afirmar?
a. Que necesariamente B es abstracta.
b. Que si B implementa el método met, entonces seguro que B no es abstracta.
c. Que no se pueden crear instancias de A.
d. Que puedo crear instancias de A.

RESERVA 2: Según el texto de la bibliografía básica de la asignatura, ¿qué puede usarse para generar la descripción de las interfaces de las clases a partir del código fuente?
a. JDK
b. JUnit
c. Code Pad
d. Ninguna de las anteriores

PARTE PRÁCTICA [6,5 PUNTOS]:

La Práctica del presente curso es diseño e implementación de un sistema integrado de gestión de una biblioteca (a partir de ahora, SIGB). En general, las funciones que tienen un SIGB son varias según el perfil de su usuario (que va desde el usuario de la biblioteca hasta su director) e incluyen las siguientes:
• Adquisiciones: la compra de materiales (libros en diferentes formatos, audiolibros, CDs de música, películas en DVD, etc.), gestión de compras, facturación, etc.
• Catalogar: la clasificación e indexación de los materiales de la biblioteca.
• Préstamos: prestar los materiales a los usuarios (tanto en papel como en otros formatos), reservas de materiales ya en préstamos, control de préstamos (emisión de avisos de materiales fuera de plazo), gestión de multas.
• Suscripciones: gestión de las suscripciones a revistas y periódicos.
• Catálogo en línea u OPAC (del inglés Online Public Access Catalog): interfaz pública a los servicios de la biblioteca (búsquedas, gestión de prestamos, etc.).
• Gestión de usuarios: altas, bajas, generación de tarjetas, historiales.

Funcionalidades

Los SIGB permiten la implementación desde labores simples de gestión de una alta de usuario, hasta operaciones más complejas como es la gestión de préstamos o inventario. En esta práctica, se propondrán diferentes funcionalidades para el sistema de gestión bibliotecaria:

- Añadir nuevos materiales a la colección de la biblioteca (rellenando los datos de un formulario). Cada tipo de material debería tener su propia colección (libros, revistas, periódicos, audio, video, etc.).
- Borrar materiales de la colección.
- Realizar búsquedas sencillas sobre los materiales.
- Gestionar suscripciones a revistas y periódicos.
- Gestión de usuarios: altas, bajas, generación de tarjetas, historiales de préstamo, control de acceso (diferenciar entre dos perfiles: usuarios y bibliotecarios).
- Realización básica de Préstamos: prestar un material si está disponible en la biblioteca, asignar fechas de devolución.

- Producir listados de préstamos según el tipo de material.
- Realizar búsquedas flexibles sobre los materiales en la biblioteca combinando varios campos de búsqueda.
- Control de préstamos: número máximo de ítems de préstamo (6 por usuario, independiente de tipo de material), emisión de avisos de materiales fuera de plazo, gestión de multas, etc.
- Producir listados de los materiales prestados.
- Realizar búsquedas flexibles sobre los materiales en varias bibliotecas a la vez combinando varios campos de búsqueda.
- Préstamos entre bibliotecas: poder solicitar materiales a otras bibliotecas y procesar las solicitudes de otras bibliotecas. El procesamiento de dichas solicitudes se lleva a cabo usando archivos de solicitud de la siguiente manera:
    - Preparar y exportar una lista de solicitudes de materiales que se quiere hacer a una biblioteca. Se prepara la lista usando un formulario para identificar el nombre de la biblioteca, el nombre del libro, el autor y el nombre de esta biblioteca. Una vez terminado, se guardar la lista en un archivo de texto. No es necesario en esta práctica preocuparse de cómo se enviaría el archivo a otras bibliotecas.
    - Importar y procesar un archivo de solicitudes para materiales proveniente de otra biblioteca. Se debe actualizar el estatus de cada material para marcarse como prestado, pero en vez del identificador del usuario debería aparecer el identificador de la biblioteca.
- Control de reservas: poder reservar un material si está ya prestado, gestión de avisos (al usuario con el material que convendría devolverlo porque hay alguien esperando y al usuario con la reserva cuando el material ya está devuelto).

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) **[1,5 puntos]** Implementa un método (o métodos) que permitan la importación (cargar al programa) de los diferentes usuarios que hay en el sistema de gestión de biblioteca. Justifíquese las opciones y decisiones que se tomen.

c) **[3,0 puntos]** Implementa un método (o métodos) que implementen el proceso devolución de un préstamo. Deberá tenerse en cuenta las implicaciones que un préstamo puede acarrear: sanciones (si se entrega fuera de plazo), actualización de ficheros que contienen los préstamos (indicar la solución que se plantea en este caso), actualización de las reservas que haya sobre ese libro, etc. Justifíquese las opciones y decisiones que se tomen.

d) **[1,0 puntos]** Proporcione un método (o métodos) que permita mostrar por pantalla un formulario básico en **modo gráfico** que permita generar las estadísticas de los préstamos que se encuentran almacenados en un fichero. La pantalla permitirá elegir entre dos listados: libros más prestados, usuarios más activos (con mayor número de préstamos). Pedirá un rango de fechas y aplicará ese criterio a la hora de buscar los contenidos en los ficheros. Los mostrará por pantalla de mayor a menor. Se pide expresamente la parte gráfica. El objetivo es ver el conocimiento y destreza en el uso de las librerías Swing y/o AWT. No desarrolle código asociado a la funcionalidad del préstamo. Justifíquese las opciones y decisiones que se tomen.