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 2014 – MODELO A – 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: Un patrón de diseño …

a. proporciona una descripción de un problema común sin dar ningún detalles de implentación.
b. no se aplica en resolución de problemas de orientación a objetos.
c. proporciona una descripción de un pequeño conjunto de clases que ayuda a resolver un problema.
d. no puede implementarse en Java

Pregunta 2: Respecto a las clases internas …

a. Las instancias de la clase interna no están necesariamente asociadas a instancias de la clase circundante.
b. No se consideran una parte de la clase circundante.
c. No pueden acceder a los métodos privados de la clase circundante.
d. Presentan un acoplamiento muy estrecho con la clase circundante.

Pregunta 3: Un método cohesionado …

a. Será responsable de al menos una tarea bien definida, pero puede serlo de más.
b. Será responsable de una y sólo una tarea bien definida.
c. Es aquel método abstracto que se ha instanciado en una clase determinada.
d. Es aquel que se crea en una clase interna para ser invocado desde la clase circundante.

Pregunta 4: 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             if (n!=0)
6             {
7                 n = n + 1;
8                 System.out.println("El número es " + n);
9             }
10        }
11    }
```

¿Cuál es la línea que provoca que el código produzca un error de compilación?

a. Línea 5.
b. Línea 3.
c. Línea 2.
d. Línea 1.

Pregunta 5: Sea el siguiente fragmento de código de código:

```
1     Random randomGenerator;
2     randomGenerator = new Random();
3     int index = randomGenerator.nextInt();
4     System.out.println(index);
```

¿En qué línea del código anterior se produce un error de compilación?

a. No se produce error de compilación
b. En la línea 1
c. En la línea 2
d. En la línea 3

Pregunta 6: Sea el siguiente fragmento de código modificado de la clase `MailItem` mostrada en el libro de texto:

```
1     public class MailItem {
2         static int num1 = 10;
3         public static void main (String args []) {
4            int num2 = 5;
5            new MailItem ();
6         }
7         public MailItem () {
8            int aux = this.num2;
9            if (aux > 1) {
10                System.out.println(aux);
11           }
12        }
13    }
```

¿Cuál es el resultado que produce?

a. Se produce un error de compilación.
b. Se produce un error de ejecución.
c. No produce ningún error pero no muestra nada por pantalla.
d. No se produce ningún error y muestra por pantalla el valor 5.

Pregunta 7: ¿Cuál de las siguientes sentencias se ejecuta de manera correcta?

a. `String matriz [] = {"Coche", "Avión", "Tren"};`
b. `String matriz = {"Coche", "Avión", "Tren"};`
c. `String matriz [] = new String {"Coche" "Avión" "Tren"};`
d. `String matriz [] = { "Coche" "Avión" "Tren"};`

Pregunta 8: Dado que un elemento `Button` puede propiciar el lanzamiento de un `ActionEvent`, ¿qué tipo de listener habría que implementar en la clase que quiera gestionar este evento?

a. `WindowListener`
b. `ActionListener`
c. `ComponentListener`
d. `PushListener`

Pregunta 9: ¿En qué condiciones puede volverse a invocar un constructor de una clase para un objeto después de que ese objeto haya sido creado?

a. Cuando queremos resetear todos los campos del objeto a sus valores iniciales.
b. Cuando se ha creado un objeto abstracto y se le quiere dar valores iniciales a sus atributos.
c. Cuando se implementa una interfaz para el objeto en cuestión.
d. Nunca.

Pregunta 10: 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    }
```

¿Cuál es el resultado de compilar y ejecutar este código?

a. Se produce un error de ejecución al definir un `ArrayList` de 5 elementos y querer insertar 6 elementos.
b. No se produce ningún tipo de error y proporciona el resultado por pantalla "Funciona".
c. La línea 4 provoca un warning pero se ejecuta sin problemas proporcionando el resultado por pantalla "Funciona".
d. La línea 7 provoca un warning pero se ejecuta sin problemas proporcionando el resultado por pantalla "Funciona".

Pregunta 11: ¿Cómo se llama el entorno de pruebas que soporta la prueba estructurada de unidades y las pruebas de regresión en Java?

a. JDK
b. JBoss
c. Javadoc
d. JUnit

Pregunta 12: Respecto al constructor de la subclase …

a. Debe siempre invocar al constructor de su superclase como primera instrucción. Si no incluye esta llamada, Java intentará insertar una llamada automáticamente.
b. No debe invocar nunca al constructor de su superclase como primera instrucción. Si la incluye esta llamada, Java ignorará esta llamada automáticamente.
c. Debe siempre invocar al constructor de su superclase como última instrucción. Si no incluye esta llamada, Java intentará insertar una llamada automáticamente.
d. Debe siempre invocar al constructor de su superclase como última instrucción. Si no incluye esta llamada, Java generará un error de compilación.

Pregunta 13: Respecto a las variables polimórficas …

a. Toda variable de objeto en Java es potencialmente polimórfica.
b. Son aquellas que exclusivamente pertenecen a clases abstractas.
c. Son la instanciación de una clase abstracta, permitiendo sólo almacenar objetos de ese tipo.
d. Son aquellas que implementan una interfaz y que provienen de una clase abstracta.

Pregunta 14: 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 puedo crear instancias de A.
d. Que puedo crear instancias de A.

Pregunta 15: Se define como excepción comprobada …

a. Aquellas subclases de la clase estándar `RunnertimeException`

b. Aquellas subclases de la clase estándar `RunneableTimeException`
c. Aquellas subclases de la clase estándar `RunningtimeException`
d. Aquellas subclases de la clase estándar `RuntimeException`

PARTE PRÁCTICA [6,5 PUNTOS]:

La práctica del presente curso ha sido una versión del legendario arcade "Pac-Man". A continuación se muestra la propuesta del juego tal y como se solicitaba para la práctica del curso.

1. El juego constará de un solo nivel donde el jugador deberá comer todos los puntos de la pantalla.
2. El jugador controlará a Pac-Man y dispondrá de 1 vida.
3. Los fantasmas serán controlados por el ordenador teniendo en cuenta el comportamiento diferente de cada uno.
4. Pac-Man podrá moverse (Utilizando las flechas del teclado) arriba (Tecla Up), abajo (Tecla Down), izquierda (Tecla Left) y derecha (Tecla Right). Así mismo podrá pausar el juego pulsando la tecla "P".
5. El área de movimiento permitido para Pac-Man y los fantasmas será el mapa del único nivel disponible.
6. Será necesario comprobar que tanto Pac-Man como los fantasmas no superen los límites del mapa.
7. Los caminos del mapa solo permiten el paso de un individuo al mismo tiempo, por tanto habrá que tener en cuenta las colisiones.
8. Los fantasmas deben implementar comportamientos diferentes:
    a. Blinky, el fantasma rojo, buscará colisionar con Pac-Man. Para acercarse a Pac-Man calculará la distancia (por ejemplo medido en filas y columnas) e intentará primero acercarse verticalmente y luego horizontalmente.
    b. Pinky. Buscará colisionar con Pac-Man. Para acercarse a Pac-Man calculará la distancia (por ejemplo medido en filas y columnas) e intentará primero acercarse horizontalmente y luego verticalmente.
    c. Clyde. Él no persigue a Pac-Man, si no que deambula sin una ruta especifica.
9. Se deberán de detectar dos tipos de colisiones.
    a. Las colisiones entre Pac-Man y los fantasmas, lo que supondrá la pérdida de una vida o el final del juego en caso de no disponer de más vidas.
    b. Las colisiones entre los fantasmas, que supondrá un cambio de dirección en los fantasmas involucrados.
10. Habrá cuatro puntos más grandes de lo normal situados cerca de las esquinas del laberinto y proporcionarán a Pac-Man la habilidad temporal (5 segundos) de comerse a los fantasmas (todos ellos se vuelven azules mientras Pac-Man tiene esa habilidad). Después de haber sido tragados, los fantasmas se regeneran en "casa de fantasmas".
11. Será necesario implementar un contador con los puntos obtenidos en cada momento, teniendo en cuenta los objetos comidos. Un punto pequeño supone 10 puntos. Comer un fantasma 100 puntos.
12. Si el jugador finaliza el nivel del juego deberá aparecer un mensaje de felicitación y se volvería a mostrar la página inicial.

a) **[2 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 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 la clase `FantasmaSusto`. Este fantasma lo que hace es huir del Pac-Man, de tal manera que mide la distancia que hay entre Pac-Man y él, y se mueve arriba, izquierda, derecha o abajo en función de si la distancia que consigue con ese desplazamiento es el mayor alejamiento posible. Especifica sus atributos y métodos y justifica las decisiones de implementación que creas importantes. Como se ha dicho, para alejarse de Pac-Man calculará la distancia según el criterio que se considere oportuno (por ejemplo, medido en filas y columnas).

c) **[1,5 puntos]** Implementa un método que gestione el efecto "terremoto". El terremoto consiste en que el sistema coloca automáticamente todos los fantasmas en posiciones aleatorias del tablero, y a Pac-Man lo coloca en el centro del todos los fantasmas. Se debe especificar cómo se calcula la posición donde se coloca Pac-Man (a definir por el alumno).

d) **[1,5 puntos]** Indique los cambios que serían necesarios en el diseño y programa para permitir que conforme pasa el tiempo, los fantasmas vayan aumentando su velocidad de movimiento hasta que Pac-Man se come un punto grande, momento en el que todos los fantasmas vuelven a restaurarse a la velocidad de movimiento inicial.