**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)**
**SEPTIEMBRE 2014 – 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:** Indica cual de las siguientes declaraciones es válida para el método main:

```
a. public static void main(String args[]);
b. static public void main(String);
c. public static void main(String );
d. public static int main(String args[]);
```

**Pregunta 2:** Indica cual de las siguientes afirmaciones es correcta:

a. Los métodos de modificación no cambian el estado de un objeto.
b. Las sentencias de asignación almacenan el valor representado por el lado derecho de la sentencia en una variable nombrada a la izquierda.
c. El alcance de una variable define la sección de un método en la que la variable puede ser accedida.
d. Los métodos de acceso devuelven información sobre el estado de una instancia.

**Pregunta 3:** Indica cual de las siguientes afirmaciones es correcta:

a. Un depurador es una herramienta de software que ayuda a examinar cómo compila una aplicación.
b. Una llamada a método interno consiste en que los métodos pueden llamar a otros métodos de la misma clase como parte de su implementación.
c. Una llamada a método externo consiste en que los métodos pueden llamar a métodos de otras clases abstractas usando la notación de punto.
d. Los objetos pueden crear otros objetos usando el operador "instanceof".

**Pregunta 4:** Supongamos que queremos implementar una Agenda, ¿cuál sería la salida del siguiente código?

```
public class Agenda {

          public static void main(String argv[]){
            Agenda agenda = new Agenda();
          }

          protected Agenda(){
            for(int i=0; i<10; i++){
                System.out.println(i);
            }
          }
}
```

a. Error de Compilación ya que los constructores no pueden ser declarados como "protected".
b. Error en tiempo de ejecución ya que los constructores no pueden ser declarados como "protected".
c. Compilación correcta y salida de los dígitos de 0 a 10.
d. Compilación correcta y salida de los dígitos de 0 a 9.

**Pregunta 5:** Indica cual de las siguientes afirmaciones es correcta:

a. El lenguaje Java tiene tres tipos de ciclo: while, while-do y for.
b. Un ciclo while es similar en su estructura y propósito que el ciclo for-each.
c. El tipo de la variable de ciclo no tiene porqué ser el mismo que el tipo del elemento declarado para la colección que estamos recorriendo con un ciclo.
d. Un índice es un objeto que proporciona funcionalidad para recorrer todos los elementos de una colección.

**Pregunta 6:** Dado el siguiente fragmento de código, indique cuál de las siguientes afirmaciones es el resultado de su ejecución:

```
if(" Problema ".trim().toLowerCase() == "problema")
        System.out.println("Igual");
else
        System.out.println("No Igual");
```

a. El código provocará un error de compilación.
b. El código provocará un error en tiempo de ejecución.
c. El código compilará e imprimirá "Igual".
d. El código compilará e imprimirá "No Igual".

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

a. La prueba es la actividad de descubrir si una pieza de código produce el comportamiento pretendido.
b. Una aserción es una expresión que establece una condición que esperamos que resulte verdadera.
c. Un seguimiento es la actividad de trabajar a través de un segmento de código línea por línea, mientras se observan cambios de estado y otros comportamientos de la aplicación.
d. Todas las respuestas anteriores son correctas.

**Pregunta 8:** Indica cual de las siguientes afirmaciones es correcta:

a. El acoplamiento describe la conectividad de los propios objetos de una clase
b. Un sistema débilmente acoplado se caracteriza por la imposibilidad de modificar una de sus clases sin tener que realizar cambios en ninguna otra
c. Un encapsulamiento apropiado en las clases reduce el acoplamiento
d. El término acoplamiento describe cuánto se ajusta una unidad de código a una tarea lógica o a una entidad

**Pregunta 9:** Basado en el ejemplo de la Base de Datos de CDs y DVDs visto en la asignatura en el capítulo 8, ¿cuál sería la salida del siguiente código?

```
public class BaseDeDatos {

            public final void metodoAgregarElemento(){
                    System.out.println("Agregar Elemento");
            }
}

public class BaseDeDatosDeMusica {

      public static void main(String argv[]){
            BaseDeDatos db = new BaseDeDatos();
```

```
        db.metodoAgregarElemento();
    }
}
```

a. Error en tiempo de compilación indicando que una clase con métodos finales deben ser declarada también como final.
b. Error en tiempo de compilación indicando que no se puede heredar de una clase con métodos finales.
c. Error en tiempo de ejecución indicando que BaseDeDatos no ha sido definida como final.
d. Éxito en la compilación y salida " Agregar Elemento".

**Pregunta 10:** Indica cual de las siguientes afirmaciones es correcta:

a. La declaración de un campo o de un método como "protected" permite el acceso directo al mismo desde las subclases (solo directas).
b. Las llamadas a métodos en Java permite que la misma llamada a un método en diferentes momentos pueda invocar diferentes métodos, dependiendo del tipo dinámico del parámetro de retorno a la hora de hacer la invocación.
c. La llamada a "super" en un determinado método (que no sea un constructor) tiene que ocurrir en su primera sentencia dentro de dicho método.
d. Ninguna de las anteriores.

**Pregunta 11:** Indica cual de las siguientes afirmaciones es correcta:

a. Todos los métodos de una interfaz son abstractos.
b. Las interfaces no contienen ningún constructor.
c. En una interfaz sólo se permiten los campos constantes.
d. Todas las respuestas anteriores son correctas.

**Pregunta 12:** Dado un visor de imágenes, ¿Cuál sería la salida del siguiente código?

```
import java.awt.*;
public class Pulsador extends Frame{
    public static void main(String argv[]){
      Pulsador MiPulsador=new Pulsador();
    }

    Pulsador(){
      Button BotonHola=new Button("Hola");
      Button BotosAdios=new Button("Adios");
      add(BotonHola);
      add(BotosAdios);
      setSize(300,300);
      setVisible(true);
    }
}
```

a. Dos botones uno al lado del otro ocupando todo el marco, "Hola" en la izquierda y "Adios" en la derecha.
b. Dos botones uno encima del otro diciendo, "Hola" arriba y "Adios" abajo.
c. Un solo botón ocupando el marco entero diciendo "Hola".
d. Un solo botón ocupando el marco entero diciendo "Adios".

**Pregunta 13:** Teniendo en cuenta el modelo en cascada presente en la construcción del software, indica cual de las siguientes fases NO pertenece al desarrollo de software:

a. Análisis del problema.
b. Prueba Unitaria.
c. Prueba Secuencial.
d. Entrega del sistema al cliente.

**Pregunta 14:** ¿Cuál sería la salida del siguiente código?

```
int i=1;
 switch (i) {
 case 0:
 System.out.print("cero ");
 break;
 case 1:
 System.out.print("uno ");
 case 2:
 System.out.print("dos ");
 break;
 default:
 System.out.print("otro ");
 }
```

a. uno
b. uno otro
c. uno dos
d. uno dos otro

**Pregunta 15:** Indica cual de las siguientes afirmaciones es correcta:

a. El proceso de autoboxing se lleva a cabo automáticamente cuando se usa un valor de un tipo no primitivo en un contexto que requiere un tipo objeto.
b. Los objetos subtipo pueden usarse cada vez que se espera un supertipo. Esto se conoce como supertipación.
c. Las clases que están vinculadas mediante una relación de herencia forman una jerarquía de herencia.
d. Todas las respuestas anteriores son falsas.

## 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 cada una en una esquina del laberinto.
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 (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 la clase `FantasmaMinky`. Este fantasma buscará colisionar con Pac-Man. Para acercarse a Pac-Man rodeará los obstáculos al contrario de las manecillas del reloj y aumentará su velocidad después de que un cierto número de puntos sean comidos (por ejemplo cada 25 puntos aumentará su velocidad). Especifica sus atributos y métodos y justifica las decisiones de implementación que creas importantes así como los cambios en las clases que creas necesario.
c) **[1,5 puntos]** Implementa un método que permita la existencia de cuatros puntos más grandes de lo normal presentes en las esquinas del laberinto y que proporcionarían a PacMan la habilidad temporal (10 segundos) de comerse a los fantasmas. Los fantasmas se volverían azules durante este periodo de tiempo y después, en caso de haber sido comidos aparecerían de nuevo cada uno por una esquina diferente.
d) **[1,5 puntos]** Indique los cambios que serían necesarios en el diseño y programa para permitir que hubiera un nuevo tipo de Fantasma denominado “InvInky” que tuviera

velocidad constante, tuviera un movimiento errático sin la intención de colisionar con PacMan y que además tuviera dos tipos de colisiones:

a. En caso de colisionar con los otros Fantasmas, tanto “InvInky” como el otro fantasma involucrado, cambiarían de sentido alejándose uno del otro.
b. En el caso de PacMan, no habría colisión sino que para PacMan sería como un objeto invisible que no le causa ningún tipo de daño ni cambio en su comportamiento.