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 B – 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: Si la primera parte de una clase TicketMachine tuviera la siguiente estructura:

```
public class TicketMachine {
      private String nombre = "ACME";
      private String registro = "0000";

      public TicketMachine(String registro) {
            this.registro = registro;
      }

      public TicketMachine(String nombre) {
            this.nombre = nombre;
      }

      public static void main (String [] args){
            String nombre = "maquina1";
            TicketMachine tm = new TicketMachine(nombre);
      }
}
```

¿Qué pasaría al ejecutar el método main?

a. tm.nombre tendría el valor de "maquina1".
b. tm.registro tendría el valor de "maquina1".
c. El programa daría un error de ejecución.
d. El programa daría un error de compilación.

Pregunta 2: Se dice que un objeto es inmutable si:

a. Su contenido o estado cambia después de su creación.
b. Su contenido o estado no puede cambiarse después de su creación.
c. Existirá más que una copia de su contenido o estado después de su creación.
d. Su contenido o estado es visible fuera de la clase en la que esta definido.

Pregunta 3: Hay una clase MessagePost que hereda de otra Post. Si ambas clases tuvieran la siguiente estructura:

```
class Post {
      public String mensaje = "En Post";

      public void enviarMensaje(){
            System.out.println(mensaje);
      }
}

public class MessagePost extends Post {
      public String mensaje = "En MessagePost";

      public void enviarMensaje(){
            System.out.println(mensaje);
      }
```

```
        public static void main(String args[]) {
          Post p = new MessagePost();

          System.out.print(p.mensaje + " ");

          p.enviarMensaje();
      }
  }
```

¿Cual sería el resultado de ejecutar el método main?:

a. En Post En Post
b. En MessagePost En Post
c. En Post En MessagePost
d. En MessagePost En MessagePost

Pregunta 4: ¿En BlueJ, cómo se pueden ver los métodos que tiene una librería del sistema como java.lang.String?:

a. En el menú 'Edit(Editar)' hay una entrada para manejar las clases en la librería.
b. En el menú 'View(Vista)' hay una entrada para manejar las clases en la librería.
c. En el menú 'Tools(Herramientas)' hay una entrada para manejar las clases en la librería.
d. No se puede hacer en BlueJ.

Pregunta 5: En una simulación de los zorros y los conejos se puede definir una clase abstracta Animal. En una versión de la simulación, el código podría ser:

```
import java.util.List;
abstract class Animal {
      String nombre = "Animal";

      abstract public void act(List<Animal> newAnimals);
}

class Zorro extends Animal {
      String nombre = "Zorro";
      public String nombreAnimal(){
            return(nombre);
      }
}

public class ZorrosConejos {
      public static void main(String[] args) {
            Zorro z = new Zorro();
            System.out.println(z.nombreAnimal());
      }
}
```

¿Cual sería el resultado de ejecutar el método main?:

a. Animal
b. Zorro
c. Un error de compilación
d. Un error de ejecución

Pregunta 6: ¿Cuál sería la signatura de un método público suma que tenga un parámetro que es un array de int y que devuelva un int?:

a. public int suma(int numeros)
b. public int suma(int[] números[])
c. public int[] suma(int numeros)
d. public int suma(int[] numeros)

Pregunta 7: Para anticipar las excepciones existe la instrucción try. Dado el siguiente código para extraer la extensión de un archivo:

```
public class Archivo {
      public void tipoExtension(String nombre) {
            try {
                  String ext = nombre.substring(nombre.indexOf('.'),
                              nombre.length());
                  System.out.println(ext);
            } catch (StringIndexOutOfBoundsException ex) {
                  System.out.print("Archivo no tiene extensión");
            } catch (ArithmeticException ex) {
                  System.out.print("Error aritmético");
            } catch (NullPointerException ex) {
                  System.out.print("Error del puntero");
            } finally {
                  System.out.print(". En clausula finally");
            }
            System.out.print(". Después del try");
      }

      public static void main(String[] args) {
            Archivo ae = new Archivo();
            ae.tipoExtension("foo");
      }
}
```

¿Cual sería el resultado de ejecutar el método main?:

a. Archivo no tiene extensión. En clausula finally. Después del try
b. Error aritmético. En clausula finally. Después del try
c. Error del puntero. En clausula finally. Después del try
d. En clausula finally. Después del try

Pregunta 8: En un reloj digital la clase ClockDisplay gestiona las horas y los minutos. Se podría añadir un método alarma para comprobar para activar un despertador:

```
private final String alarma = "11:00";
public void alarma(String hora){
      System.out.println("Hora == Alarma is:" + hora == alarma);
}

public static void main(String[] args) {
      Archivo ae = new Archivo();
      ae.alarma("11:00");
}
```

¿Cual sería el resultado de ejecutar el método main()?:

a. Hora == Alarma is:
b. Hora == Alarma is: 11:00
c. Hora == Alarma is: 11:00 false
d. false

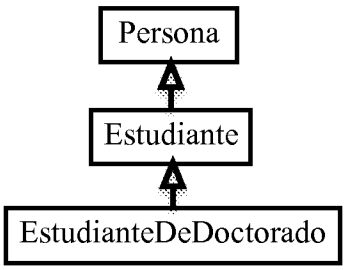
Pregunta 9: En la práctica de este año hay que crear un tablero para el juego Pacman. Para ello se puede dibujar una matriz de cuadros negros, dejando otros en blanco. ¿Cuál de los siguientes métodos nos permite dibujar un cuadrado negro entero en la pantalla?:

a. fillRect(int x, int y, int anchura, int altura);
b. fillRect(int anchura, int altura);
c. fillCuadro(int x, int y, int anchura, int altura);
d. fillCuadro(int anchura, int altura);

Pregunta 10: ¿Qué es un banco de pruebas? :

a. Uno o más objetos que se emplean en más de una prueba.
b. Uno o más objetos que se emplean en una sola prueba.
c. Uno o más objetos que se emplean para encontrar errores sintácticos en el código.
d. Ninguno de los anteriores.

Pregunta 11: Dada la siguiente jerarquía de clases:



¿Cuáles de las siguientes asignaciones serían legales?:

1. Persona p1 = new Estudiante();
2. Persona p2 = new EstudianteDeDoctorado();
3. EstudianteDeDoctorado edd = new Estudiante();
4. Estudiante e1 = new EstudianteDeDoctorado();

a. 1,2,3,4
b. 1,2,3
c. 2,3,4
d. 1,2,4

Pregunta 12: ¿Existen varios modelos para la construcción de software. ¿Cuáles son dos de los más conocidos?:

a. Modelo en cascada y modelo de análisis
b. Modelo en cascada y desarrollo iterativo
c. Modelo en cascada y prueba incremental
d. Desarrollo iterativo y prueba incremental

Pregunta 13: En el organizador de música podemos usar un ArrayList para guardar los nombres de las canciones:

```
import java.util.ArrayList;
public class OrganizadorMusica {

	ArrayList<String> canciones = new ArrayList<String>();
	public static void main(String[] args) {
		OrganizadorMusica mo = new OrganizadorMusica();
		mo.canciones.add("Al amanecer");
		System.out.println("¿Existe la canción?: " + mo.existe("Al
amanecer"));
	}

	public boolean existe(String cancion) {
			XXX
			if(titulo.equals(cancion))
				return(true);
		}
		return(false);
	}
```

```
}
```

¿Qué habrá que poner en vez de los XXX para que al ejecutar main el programa produzca el resultado "Existe la canción?: true"?

a. while (String titulo: canciones) {
b. for (String titulo in canciones) {
c. for (String titulo: canciones) {
d. for (int i=0; i < canciones.length; i++) { String titulo = canciones.get(i);

Pregunta 14: En las diferentes versiones de un proyecto del juego zuul se pueden plantear diferentes versiones del método getExitString; por ejemplo:

`public String getExitString() {`
`  String returnString = "Exits:";`
`  if (northExit != null)`
`  returnString += "north ";`
`  if (eastExit != null)`
`  returnString += "east ";`
`  if (westExit != null)`
`  returnString += "west ";`
`  if (southExit != null)`
`  returnString += "south ";`
`  return returnString;`
`}`
	`public String getExitString() {`
	`  String returnString = "Exits:";`
	`  Set<String> keys = exits.keySet();`
	`  for(String exit : keys) {`
	`  returnString += " " + exit;`
	`  }`
	`  return returnString;`
	`}`

Versión A	Versión B

¿Cuál de las dos versiones muestra más acoplamiento?

a. A
b. B
c. Son iguales
d. No muestran acoplamiento ninguno

Pregunta 15: En un visor de imágenes se usa botones para que el usuario pueda cambiar el tamaño de la imagen; por ejemplo:

```
smallerButton = new JButton("Smaller");
smallerButton.addActionListener(XXX {
            public void actionPerformed(ActionEvent e) { makeSmaller(); }
      });
toolbar.add(smallerButton);
```

¿Qué tipo de Listener habrá que usar (donde están los XXX en el código anterior) para detectar que el usuario ha hecho clic en el botón?

a. new EventListener()
b. new ButtonListener()
c. new ActionListener()
d. new ActionPerformedListener()

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 `FantasmaPinky`. Especifica sus atributos y métodos y justifica las decisiones de implementación que creas importantes. Recuerda que este

fantasma 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) **[1,5 puntos]** Implementa un método que gestione las dos tipos de colisiones que puede haber 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, y las colisiones entre los fantasmas, que supondrán un cambio de dirección en los fantasmas involucrados.

d) **[1,5 puntos]** Indique los cambios que serían necesarios en el diseño y la implementación para permitir que haya diferentes niveles y que en cada uno de éstos se añada un nuevo tipo de fantasma (además de los que ya existen).