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 D – NO ESTÁ PERMITIDO EL USO DE MATERIAL ADICIONAL

PARTE TEÓRICA - TEST [2,5 PUNTOS]:

Sólo 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 método de acceso o selector:

a. Habitualmente devuelve void.
b. Devuelve siempre información sobre el estado de un objeto.
c. Devuelve siempre un objeto de la clase Object.
d. Permite acceder al constructor de la clase que lo define.

Pregunta 2: Un método de modificación o mutador:

a. Habitualmente devuelve void.
b. Devuelve siempre información sobre el estado de un objeto.
c. Permite modificar el estado únicamente de los campos públicos de la clase.
d. Permite acceder al constructor de la clase que lo define.

Pregunta 3: Dada la siguiente definición de clase:

```
public class MusicOrganizer
{
    private int valor;

    public MusicOrganizer (int n)
    {
        valor=n;
    }

    public int calcular()
    {
        int resultado=1;
        int numero=valor;

        if (valor>=1)
        {
            while(numero>=1){
                resultado*=numero;
                numero--;
            }
            return resultado;
        }
        else{
            return 1;
        }

    }

    public static void main (String [] args){

        MusicOrganizer t=new MusicOrganizer(4);
        int v=t.calcular();
```

```
            System.out.println(v);

        }
    }
```

El resultado de su ejecución sería:

a. 24
b. 6
c. 120
d. 0

Pregunta 4: Cual de las siguientes afirmaciones es falsa:

a. La documentación de la librería de clases de Java muestra detalles acerca de todas las clases de la librería.
b. La interfaz de una clase describe lo que una clase hace y cómo se puede utilizar sin mostrar su implementación.
c. Los modificadores de acceso definen la visibilidad sólo de los campos.
d. Las clases pueden tener campos. Estos se conocen con el nombre de variables de clase o variables estáticas.

Pregunta 5: Indique cual de las siguientes afirmaciones no es correcta respecto al uso de la herencia en JAVA:

a. Evita el tener que declarar constructores
b. Evita la duplicación de código
c. Facilita la reutilización del código
d. Facilita la ampliabilidad y mantenimiento del código

Pregunta 6: Se ha visto en la asignatura 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[]) {
        MessagePost p = new MessagePost();
        Post post=p;
        post.enviarMensaje();
    }
}
```

Cual sería el resultado de ejecutar el método main:

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

Pregunta 7: Las pruebas de regresión se definen como:

a. La ejecución de las pruebas pasadas previamente para asegurarse de que la nueva versión aún las pasa.
b. La ejecución de pruebas automatizadas aleatorias sobre los distintos valores que puede recibir la clase evaluada.
c. La aplicación sistemática del conjunto de casos de prueba base que se definieron justo al comenzar con el desarrollo de la aplicación y que no varían nunca a lo largo de éste.
d. El conjunto de pruebas negativas necesarias para demostrar que la clase evaluada falla.

Pregunta 8: En una simulación de los zorros y los conejos se puede definir una clase abstracta Animal. En una versión modificada de la simulación el código podría ser:

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

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

class Zorro extends Animal {
      String nombre = "Zorro";

      public String nombreAnimal(){
            return(nombre);
      }

      public void metodo(List<Animal> newAnimals){

          System.out.println("Animal");
      }
}

public class ZorrosConejos {
      public static void main(String[] args) {
            Animal 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 en tiempo de ejecución

Pregunta 9: Dado el código de la clase MusicOrganizer. ¿Cuál sería el resultado de la ejecución del método main?

```
import java.util.ArrayList;


public class MusicOrganizer
{

    private ArrayList<String> files;

    public MusicOrganizer()
    {files = new ArrayList<String>();}
```

```
    public void addFile(String filename)
    {files.add(filename);}

    public int getNumberOfFiles()
    {return files.size();}

    public void listFile(int index)
    {
        if(index >= 0 && index < files.size()) {
            String filename = files.get(index);
            System.out.println(filename);
        }
    }

    public void removeFile(int index)
    {
        if(index >= 0 && index < files.size()) {
            files.remove(index);
        }
    }
}
public class Test
{
    public static void main(String[] args) {
        MusicOrganizer mo=new MusicOrganizer();

        mo.addFile("Disco 1");
        mo.addFile("Disco 2");
        mo.addFile("Disco 3");

        mo.listFile(1);
        mo.removeFile(1);
        mo.listFile(1);
    }
}
```

a. Disco 2 null
b. Error en tiempo de ejecución
c. Disco 1 Disco 2
d. Disco 2 Disco 3

Pregunta 10: Dado el siguiente fragmento de código, podemos afirmar que la salida del programa:

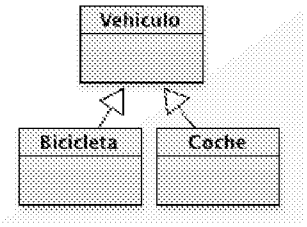
```
import java.util.Random;

public class Test
{
    public static void main(String[] args) {

        Random generadorAleatorios;
        generadorAleatorios=new Random();

        for(int n=0;n<=100;n++){
            System.out.println(generadorAleatorios.nextInt(n+1));
        }
    }
}
```

a. Generará valores aleatorios entre 0 y n en cada vuelta del bucle.
b. Generará valores aleatorios entre 0 y n+1 en cada vuelta del bucle.
c. Generará valores aleatorios entre 1 y n en cada vuelta del bucle.
d. Generará valores aleatorios entre 1 y n+1 en cada vuelta del bucle.

Pregunta 11: Indique cual de las siguientes afirmaciones es falsa en relación al desarrollo de la práctica obligatoria:

a. El método actionPerformed es el encargado de actualizar las posiciones y el estado del juego en cada intervalo de tiempo.
b. La vista del juego se implementa sobre un JPanel sobreescribiendo su método paint().
c. Para detectar las pulsaciones de teclado en el juego podemos hacer uso de la clase abstracta KeyAdapter.
d. El método actionPerformed hay que implementarlo en el modelo, nunca en el controlador.

Pregunta 12: Dada la siguiente jerarquía de clases:



Y la siguiente inicialización de objetos:

```
public static void main(String[] args) {
        Vehiculo v1=new Coche();
        Vehiculo v2=new Vehiculo();
        Bicicleta b=new Bicicleta();
        Coche c=new Coche();
    }
```

¿Cuáles de las siguientes asignaciones son correctas?

1. c=(Coche)v1;
2. c=(Coche)v2;
3. b=(Bicicleta) c;
4. b=v2;

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

Pregunta 13: Sea una nueva definición de las clases Post y MessagePost:

```
public class Post
{
   public String toString(){

       return "Mensaje 2 ";
    }
}

public class MessagePost extends Post
{
    public String toString(){

       return "Mensaje 1 "+super.toString();
    }
```

```
}

public class Test
{
    public static void main(String args[]) {
        Post p = new MessagePost();
        System.out.println(p);
    }
}
```

¿Qué se mostrará por pantalla al ejecutar el método main de la clase Test?

a. Error de Compilación
b. Mensaje 2 Mensaje 1
c. Mensaje 1 Mensaje2
d. Mensaje 2

Pregunta 14: En el código libreta de direcciones (address book) explicado en el libro de texto se hace uso de las aserciones.

```
public void removeDetails(String key)
{
   if(key == null){
      throw new IllegalArgumentException("Null key passed to
removeDetails. ");
   }
   if(keyInUse(key)) {
      ContactDetails details = book.get(key);
      book.remove(details.getName());
      book.remove(details.getPhone());
      numberOfEntries--;
    }
    assert !keyInUse(key);
    assert consistentSize() : "Inconsistent book size";
}
```

Indique cual de las siguientes afirmaciones es cierta:

a. Se muestra un error AssertionError en el caso en el que el método keyInUse devuelva como resultado falso.
b. Se muestra el mensaje "Inconsistent book size" en el caso en el que el método consistentSize devuelva como resultado falso.
c. Se muestra el mensaje "Inconsistent book size" en el caso en el que el método consistentSize devuelva como resultado verdadero.
d. Se lanza una excepción derivada de la clase Exception en el caso en el que el método keyInUse devuelva como resultado verdadero.

Pregunta 15: Indique cual de las siguientes afirmaciones relativas al uso de excepciones en Java es falsa:

a. Las excepciones comprobadas están pensadas para aquellos casos en los que el cliente debería esperar que una operación pueda fallar.
b. Las excepciones no comprobadas están pensadas para aquellos casos que nunca deberían fallar durante la operación normal.
c. Las excepciones no comprobadas heredan de la clase Error.
d. Las excepciones comprobadas heredan de la clase Exception.

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 `FantasmaClyde`. Especifica sus atributos y métodos y justifica las decisiones de implementación que creas importantes. Recuerda que este fantasma deambula sin una ruta específica por el laberinto.

c) **[1,5 puntos]** Implementa los métodos necesarios para gestionar el cambio de estado tanto en Pac-Man como en los fantasmas cuando éste pasa por encima de un punto grande. Es necesario indicar a qué clases pertenece cada uno de los métodos implementados.

d) **[1,5 puntos]** Indique los cambios que serían necesarios en el diseño y programa para permitir la existencia de "premios" (e.g. cerezas, fresas, naranjas, manzanas, etc) en cada nivel del juego que permitan a Pac-Man aumentar su puntuación al pasar por encima de ellos.