ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN

PROGRAMACIÓN ORIENTADA A OBJETOS

TERCERA EVALUACIÓN

I TÉRMINO - 2017

NOMBRE:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ PARALELO:\_\_\_\_\_\_\_\_\_\_\_\_\_

**TEMA 1 (10 puntos)**

**Use la siguiente información para responder las siguientes preguntas:**

Polígono es una clase que define polígonos regulares (figuras como triángulos equiláteros, cuadrados y pentágonos regulares -polígonos donde todos los lados tienen la mismo largo). La clase tiene, entre otros métodos públicos, un método denominado area(), que no toma parámetros, y devuelve como tipo double el área del polígono. Las clases Cuadrado, EquiTriangulo y Pentagono se derivan de Poligono. Cuadrado y EquiTriangulo cada uno tiene, entre otros métodos públicos, un método con nombre area(), que no toma parámetros y devuelve como tipo double el área de un cuadrado y EqiTriangle, respectivamente. El Pentágono no define un método denominado area().

**1.1 ¿Qué atributos y métodos la clase Cuadrado hereda de Poligono? (1 Punto)**

a) Solo aquellos que tienen acceso public o default

b) Todos excepto los que son private

c) Solo los públicos

d) Todos excepto aquellos que tienen acceso protected

e) Todos

**1.2 Dado que P es un objeto de la clase Poligono, C es un objeto de tipo Cuadrado, T es un objeto de tipo EquiTriangulo y Pt es un objeto de tipo Pentagono. Responda la siguiente pregunta**

**Para cada una de las siguientes sentencias indique si el código es válido o no. En caso de ser válido indique en qué clase se encuentra la definición del método area al que se está llamando (3 puntos) (Justifique su respuesta)**

|  |  |
| --- | --- |
|  | **Respuesta** |
| T.area() |  |
| P = C  P.area() |  |
| Pt.area() |  |
| T = P  T.area() |  |

**1.3 Suponga que desea llamar al método area() de Poligono en la definición del método area() de EquiTriangulo. Ambos métodos tienen la misma firma. Cómo se hace? (1 Punto)**

a) (polygon)area();

b) super.area();

c) this.area()

d) area();

e) parent.area()

**1.4 Suponga que la función area() de la clase Poligono se vuelve abstracta. Cuál de los siguientes enunciados es verdadero. (Escoja todos los que aplique) (1 Punto)**

a) Poligono debe ser una clase abstracta

b) Objetos de tipo Poligono no pueden ser instanciados

c) Cualquier clase derivada de Poligono debe sobrescribir area()

d) La función area() en polígono no va a cambiar su implementación

**1.5 Por cada uno de los siguientes enunciados, explique en qué escenario se esperaría usar. Para obtener puntaje completo debe indicar las ventajas o desventajas de su uso. (1 PUNTO c/u)**

a) Usar un Treeset en vez de usar un Hashset

b) Clases que son marcadas como final

c) Atributos en una clase que son marcados como private

d) Declarar una interface en vez de una clase

**TEMA 2**

**2.1 Dado el siguiente código (5 Puntos)**

**Justificación:**

**1: public class Dog {**

**2: public String name;**

**3: public void parseName() {**

**4: System.out.print("1");**

**5: try {**

**6: System.out.print("2");**

**7: int x = Integer.parseInt(name);**

**8: System.out.print("3");**

**9: } catch (NumberFormatException e) {**

**10: System.out.print("4");**

**11: }**

**12: }**

**14: public static void main(String[] args) {**

**15: Dog leroy = new Dog();**

**16: leroy.name = "Leroy";**

**17: leroy.parseName();**

**18: System.out.print("5);**

**19: }**

**20: }**

**Seleccione la respuesta correcta (Justifique)**

**a.** 12

**b.** 12345

**c.** 124

**d.** 1245

**2.2 Dado el siguiente código (5 Puntos)**

1: import java.util.\*;

2: public class MyPancake {

3: public static void main(String[] args) {

4: List<String> x = new ArrayList<String>();

5: x.add("3");

6: x.add("7");

7: x.add("5");

8: List<String> y = MyPancake.doStuff(x);

9: y.add("7");

10: System.out.println(x);

11: }

12: List<String> doStuff(List<String> z) {

13: z.remove("7");

14: return z;

15: }

16: }

17:

**El código anterior no compila, indique qué haría para que compile. Una vez corregido el error indique cuál sería la salida en pantalla (Justifique su respuesta)**

**2.3 Dado el siguiente código (5 Puntos)**

1: public class TestThread implements Runnable{

2: public static void main(String[] args) {

3: Thread t1 = new Thread(new TestThread(),"Thread-1");

4: Thread t2 = new Thread(new TestThread(),"Thread-2");

5: t1.start();

6: t2.start();

7: }

8: public static void run() {

9: for(int i = 0; i < 2; i++)

10 System.out.print(Thread.currentThread().getName() + " ");

11: }

12: }

**Seleccione la o las respuestas correctas (Justifique)**

A. Compilation fails.

B. No output is produced.

C. The output could be Thread-1 Thread-2 Thread-1 Thread-2

E. The output could be Thread-1 Thread-1 Thread-2 Thread-2

**Tema 3 (10 puntos)**

**Dado el siguiente código (puede tener errores):**

1: interface Guerrero{

2: abstract int getFuerza() throws Exception;

3: }

4: interface Saiyajin implements Guerrero{

5: int nivel();

5: }

7: public class Jugador implements Saiyajin,Serializable{

8: public static final boolean estaVivo = true;

9: public int fuerza;

10: @Override

11: public int getFuerza() throws Exception {

12: return fuerza;

13: }

14: public boolean equals(Saiyajin obj) {

15: if(obj.nivel()== 3){

16: return true;

17: }

18: return false;

19: }

20: }

**3.1 Indique los errores en el código anterior y cómo los resolvería (4 puntos):**

**3.2 La clase Jugador ¿requiere sobreescribir obligatoriamente el método equals(Saiyajin obj)? (Justifique) (1 punto)**

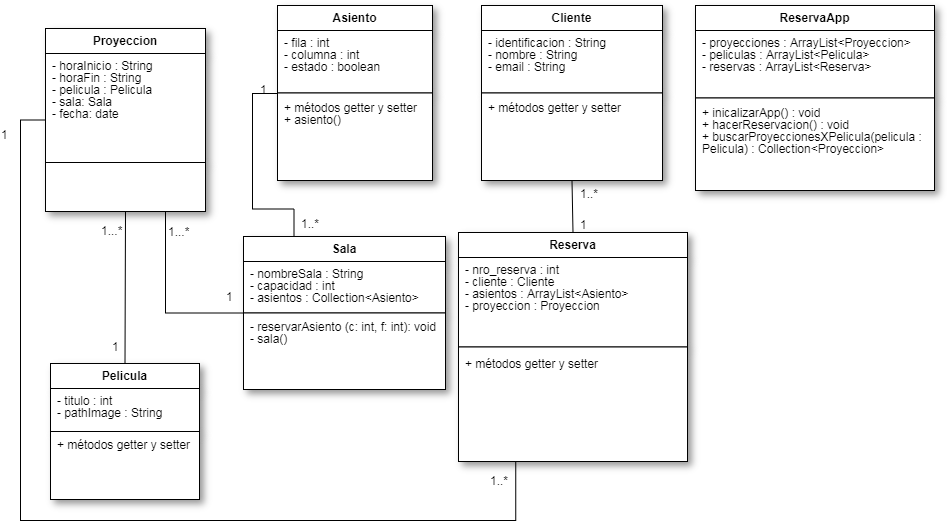
**3.3 ¿Es legal realizar la siguiente asignación? Asuma que se encuentra en otra clase - (Justifique) (1 punto)**

Jugador.estaVivo = false;

**3.4 Realice los cambios necesarios para que la clase Jugador implemente la clase Comparable para que el ordenamiento natural de los objetos de la clase Jugador sea en orden ascendente por la variable de instancia fuerza. (4 puntos)**

**Tema 4 (Desarrollo)**

Un cine desea crear una aplicación de escritorio para gestionar la reserva y venta de entradas de sus proyecciones. El cine dispone de varias salas y de una programación de películas para cada sala, en un rango de horarios desde las 4 de la tarde a las 10 de la noche (con un intervalo de separación entre sesiones de dos horas o dos horas y media).Todas las salas de cine tienen la misma capacidad, 108 asientos, distribuidos en 12 filas y 15 columnas.



asiento

1. Implemente la clase Sala con los atributos y métodos establecidos en el diagrama de clases (10 puntos).

- el constructor Sala(nombreSala: String), fija el nombre de la sala y crea los asientos de la misma. Recuerde que todas las salas tienen el mismo número de asientos distribuidos en 12 filas y 15 columnas. La variable de instancia ¨estado¨ de la clase Asiento se inicializa en ¨true¨ que indica que el asiento está disponible.

- el método reservarAsiento(c: int, f: fila ), recibe la fila y columna de un asiento y fija el estado del asiento en esa fila y columna a False.

b. Implemente el método equals en la clase Película (3 puntos).

c. Cree una excepción llamada BookingException (5 puntos).

**Clase ReservaApp**

d. En la clase **ReservaApp** implemente el método **buscarProyeccionesPorPelicula(Pelicula p)** que recibe una Pelicula y retorna una colección de las Proyecciones de esa Película (5 puntos).

e. Implemente el método **hacerReservación(String nombreCliente, String cedula, Proyeccion proyeccion, Collection<Asiento> asientos)** de la clase **ReservaApp** que recibe los datos del cliente, la proyección y la colección de asientos que el cliente quiere reservar.

Para que una reservación sea exitosa todos los asientos de la reserva deben estar disponibles. Si la reserva es exitosa se crea un objeto de tipo Reserva que se almacena en el ArrayList reservas y cambia el estado de los asientos pasados como parámetros a reservado. Si la reserva no puede ser realizada el método debe lanzar la excepción BookingException (10 puntos).

**Clase ReservaPane**

f. Construya la siguiente pantalla para el ingreso de la reserva:

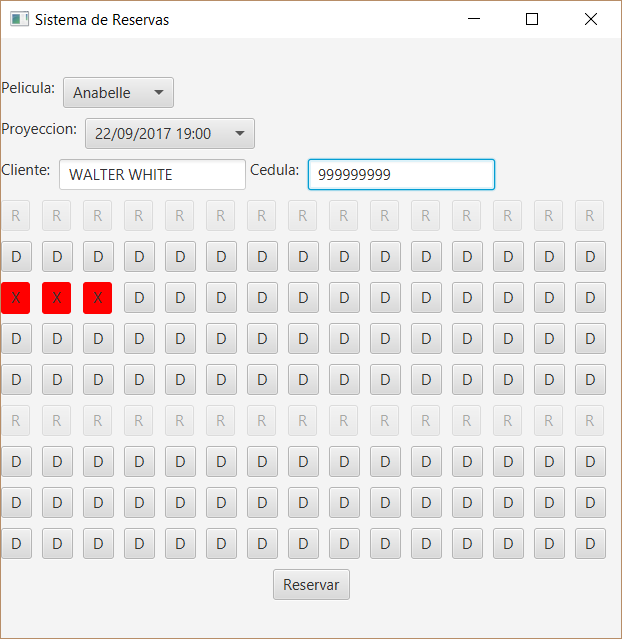
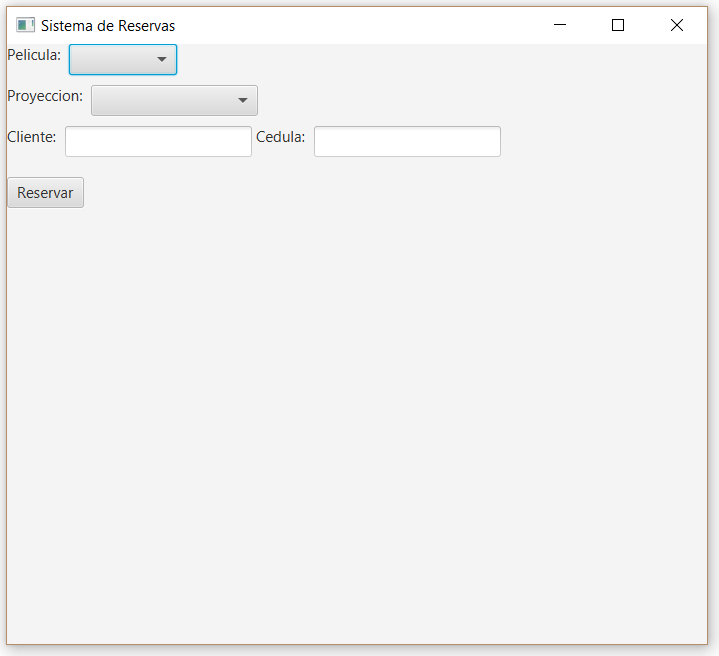


Figura 1

Figura 2

f1. Al iniciarse la aplicación se debe mostrar la pantalla en Figura1. El combo de películas se debe cargar con la información de las películas en ReservaApp y el combo de proyecciones se inicializa vacío. (7 puntos).

f.2 Cuando el usuario selecciona una película se debe cargar en el combo de proyecciones las proyecciones correspondientes a esa película (5 puntos).

f.3 Cuando el usuario selecciona una proyección se debe mostrar en pantalla los asientos de la sala que corresponden a la proyección seleccionada. Cada asiento es un botón. Los botones de los asientos que están reservados aparecen deshabilitados y con la letra R, los botones de los asientos que están disponibles aparecen habilitados y con la letra D (Vea Figura 2) (10 puntos).

f.4 Cuando se da clic en el botón de uno de los asientos disponibles este debe cambiar de color a rojo. Si se da click sobre un asiento seleccionado anteriormente el asiento se desmarca (5 puntos).

f.5 Al dar click en reserva se debe llamar al método hacerReservacion definido en ReservaApp. El método debe validar que todos los campos estén llenos y al menos un asiento esté seleccionado. Si la reserva es exitosa se debe mostrar un mensaje de reserva exitosa y volver al estado original de la aplicación. En caso de que la reserva no haya sido exitosa se debe mostrar un mensaje de error adecuado (10 puntos).