**Tema 1 (30 puntos)**

1. **(5 puntos) ¿Cuándo usaría un constructor privado?**
   1. Cuando te aburras del public
   2. Si desea evitar que se instancie esa clase fuera de ella misma
   3. Si desea proteger a sus miembros (atributos y métodos) de modificaciones externas
   4. Nunca, No está permitido
2. **(5 puntos)** ¿Cuándo debe ser declarada una clase abstracta?
   1. Cuando no tiene sentido lógico instanciarla
   2. Para que no pueda ser usada como interface
   3. Para que ninguna clase herede de ella
   4. Porque no tiene métodos abstractos
3. **(5 puntos) ¿Qué pasaría si trata de correr y compilar el siguiente código Java?**

public class Bground extends Thread{

public static void main(String argv[]){

Bground b = new Bground();

b.run();

}

public void start(){

for (int i = 0; i <10; i++){

System.out.println("Value of i = " + i);

}

}

}

* 1. No compila, indicando que no hay un método run definido para la clase Thread.
  2. Compila, pero se produce un error en tiempo de ejecución indicando que no hay un método run definido para la clase Thread.
  3. Compila y se ejecuta imprimiendo los valores del 0 al 9.
  4. Compila; pero al ejecutarse no produce ninguna salida en pantalla.

1. **(5 puntos)** Suponga que la clase Undergraduate hereda de la clase Student y esta última hereda de la clase Person. Dada la siguiente declaración de variables:

Person p = new Person();

Student s = new Student();

Undergraduate ug = new Undergraduate();

¿Cuál de las siguientes asignaciones son legales?

i. p = ug;

ii. p = new Undergraduate();

iii. ug = new Student();

iv. ug = p;

v. s = new Person();

1. **Para la pregunta i y ii utilice el siguiente código:**

public class MtermEx

{

private int x;

public MtermEx(int newValue)

{

x = newValue;

}

}

* 1. **¿Cuál de lo siguiente acerca de la clase MtermEx es verdadero?**
     1. No tiene clase padre
     2. Su clase padre es Object
     3. Su clase padre es Java
     4. No se puede heredar de ella
     5. Tiene una clase hija por defecto llamada Object
  2. Si q1 y q2 son objetos de MtermEx, entonces q1.equals(q2)
     1. Ocurre un error de sintaxis debido a que equals no ha sido definida en MtermEx
     2. Es verdadero si q1 y q2 ambos almacencan el mismo valor de x
     3. Es verdadero si q1 y q2 referencian al mismo Objeto de MtermEx
     4. Nunca es verdadero
     5. 66
     6. Lanza una excepción de tipo NullPointerException

1. **Dado el siguiente codigo seleccione la respuesta correcta y explique:**

1. import java.util.\*;

2. public class Garage {

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

4. Map<String, String> hm = new HashMap<String, String>();

5. String[] k = {null, "2", "3", null, "5"};

6. String[] v = {"a", "b", "c", "d", "e"};

7.

8. for(int i=0; i<5; i++) {

9. hm.put(k[i], v[i]);

10. System.out.print(hm.get(k[i]) + " ");

11. }

12. System.out.print(hm.size() + " " + hm.values() + "\n");

13. }

14. }

**Cual es el resultado más aproximado?**

A. a b c a e 4 [c, b, a, e]

B. a b c d e 4 [c, b, a, e]

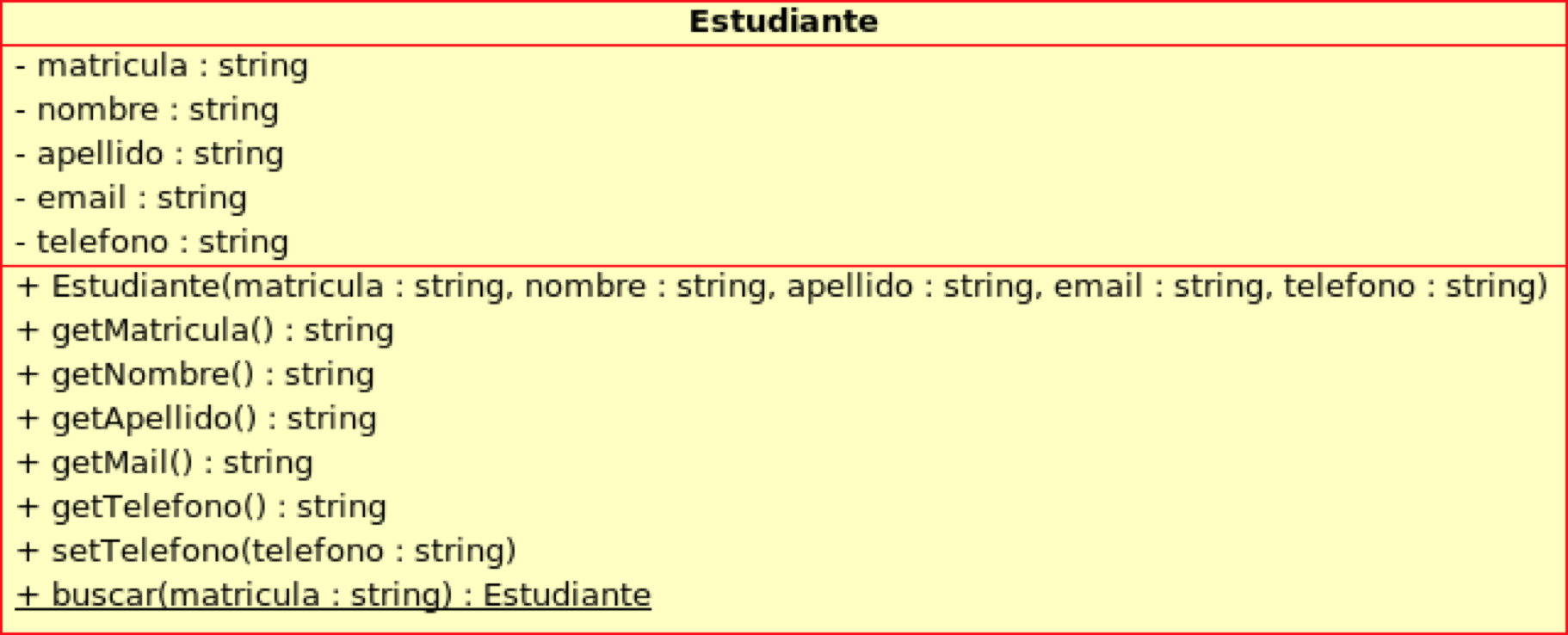
C. a b c d e 4 [c, d, b, e]

D. a b c, seguido por excepcion.

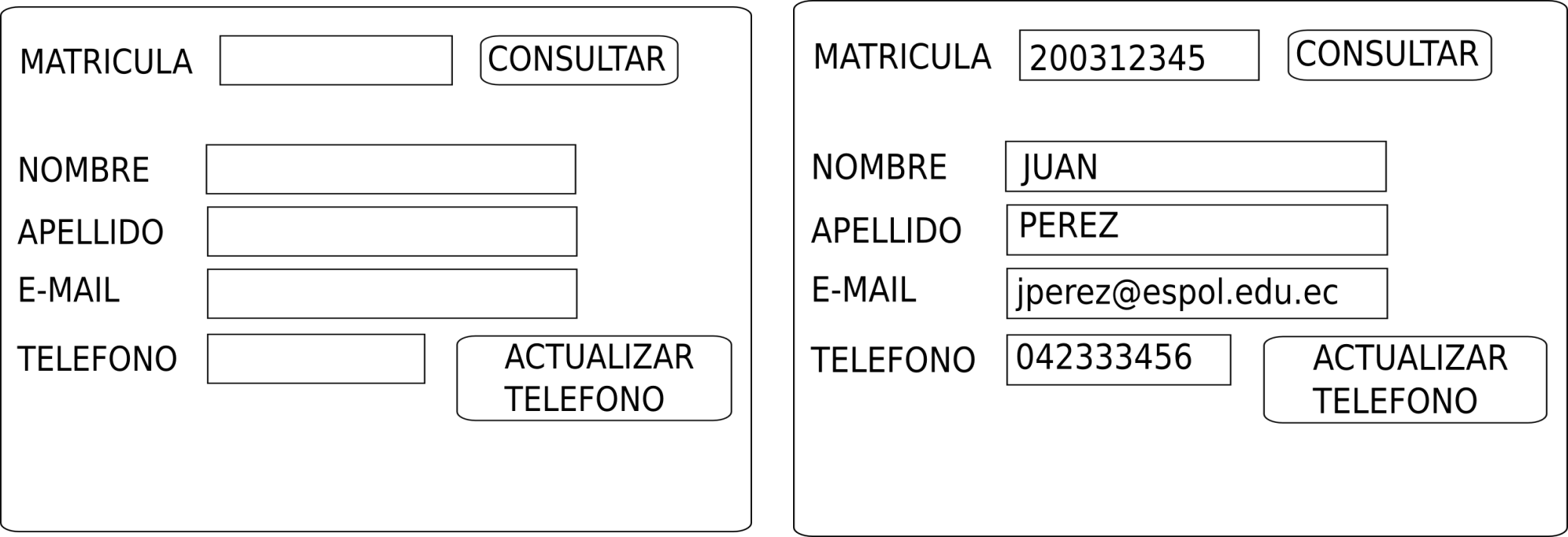
E. Compilacion falla

**Tema 2 (35 puntos)**

Se necesita implementar una aplicación que permite tanto realizar una búsqueda de un estudiante a través de la matrícula como actualizar su número telefónico. Para esto se dispondrá de un repositorio de datos con los datos de todos los estudiantes existentes y el siguiente diagrama UML:



1. **(16 puntos)**Implemente los métodos:
   1. **(8 puntos)** **setTelefono**: actualiza el teléfono del estudiante actual en el repositorio de datos que contiene los datos de todos los estudiantes
   2. **(8 puntos)** buscar: busca en el repositorio al estudiante con la matrícula indicada.
2. **(19 puntos)** Construya la clase **ConsultaEstudiante** que represente la siguiente ventana. Esta permitirá al usuario ingresar un numero de matricula. Al dar click en el botón Consultar se mostrará la información del estudiante. El usuario podrá únicamente modificar el campo del número telefónico y este se actualizará cuando de click en el botón de Actualizar Teléfono.



**Tema 6 (35 puntos)**

Este es un juego de 2 jugadores en el que cada jugador se desplaza a lo largo de un laberinto lanzando un dado. El laberinto está compuesto por casillas.

Existen 3 tipos de dados en el juego:

**normal**, donde el resultado de tirar el dado es un número del 1 al 6

**lento**, donde el resultado de tirar el dado es un número del 1 al 3

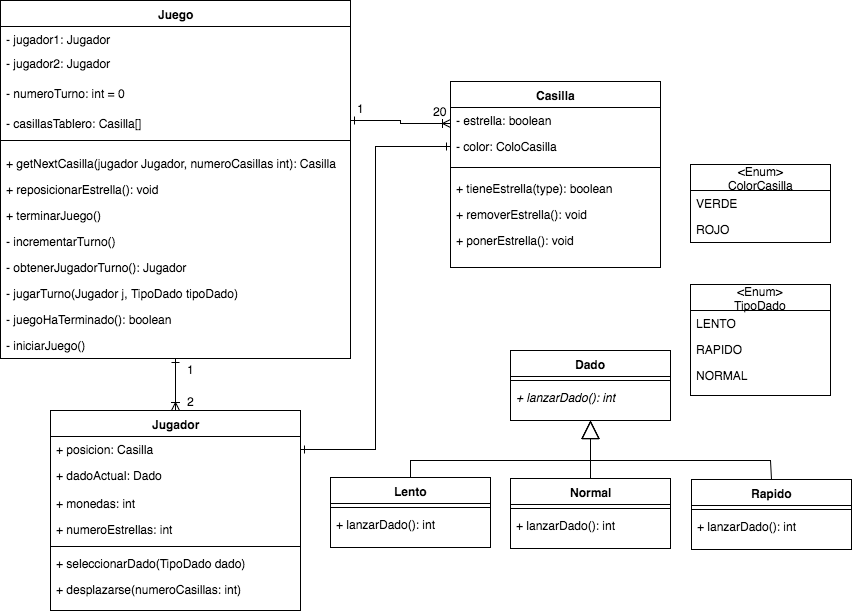
**rápido**, donde el resultado de tirar el dado es un número del 4 al 6.

En su turno el jugador selecciona un tipo de dado y lo lanza. El jugador se desplaza a lo largo del laberinto el número de casillas que indica el dado. Al inicio del juego se coloca una estrella en una de las casillas del laberinto.

Cuando un jugador avanza hacia una casilla debe evaluarse el tipo. Si es una casilla *verde*, el jugador **gana** monedas. Si es una casilla *roja*, el jugador **pierde** monedas. Si es una casilla con *estrella*, el jugador tiene la oportunidad de obtener una y estar más cerca de ganar el juego. (ver método **jugarTurno())**

El juego se finaliza al completar 10 turnos. Gana el juego el jugador que más estrellas haya acumulado hasta ese momento.

Considere el siguiente modelo UML del juego:



El método **getNextCasilla**(Jugador jugador, int numeroCasillas):**Casilla** recibe un jugador y el número de casillas que el jugador debe avanzar y retorna la nueva casilla del jugador.

**Implementar los siguientes métodos**:

**En la clases de tipo Dado:**

* **lanzarDado(): int (5 puntos)**

Este método retorna un número entero aleatorio. Si el dado es normal un número del 1 al 6, si el dado es rápido un número del 4 al 6 y si el dado es lento un número del 1 al 3.

**En la clase *Jugador*:**

* **seleccionarDado(TipoDado dado) (2.5 puntos)**

Este método cambia el tipo de dado que el jugador quiere utilizar en su turno.

* **desplazarse(int numeroCasillas) (2.5 puntos)**

Este método recibe el número de movimientos y asigna al jugador a una nueva casilla.

**En la clase *Juego*:**

* **reposicionarEstrella() (10 puntos)**

Este método recorre todo el tablero hasta encontrar la posición actual de la estrella, remueve la estrella de esa posiciñon y luego posiciona la estrella en una nueva casilla al azar. (Note que el tablero es un array de Casillas)

* **jugarTurno(Jugador j, TipoDado tipoDado) (15 puntos)**
* Actualiza el el dado del jugador,
* Lanza el dado que tenga seleccionado y desplazar al jugador a una nueva casilla de acuerdo el número de casillas indicadas por el dado.
* Por último de acuerdo a la casilla en la que cayó el jugador se deben realizar una de las siguientes acciones
* Si la casilla tiene una **estrella y el jugador tiene 20 monedas o más,** aumentar una estrella al jugador, disminuir 20 monedas al jugador y reposicionar la estrella.
* Si la casilla tiene una **estrella y el jugador no tiene 20 monedas,** no sucede nada
* Si la casilla es **verde,** aumentar 3 monedas al jugador
* Si la casilla es **roja,** disminuir 3 monedas al jugador.