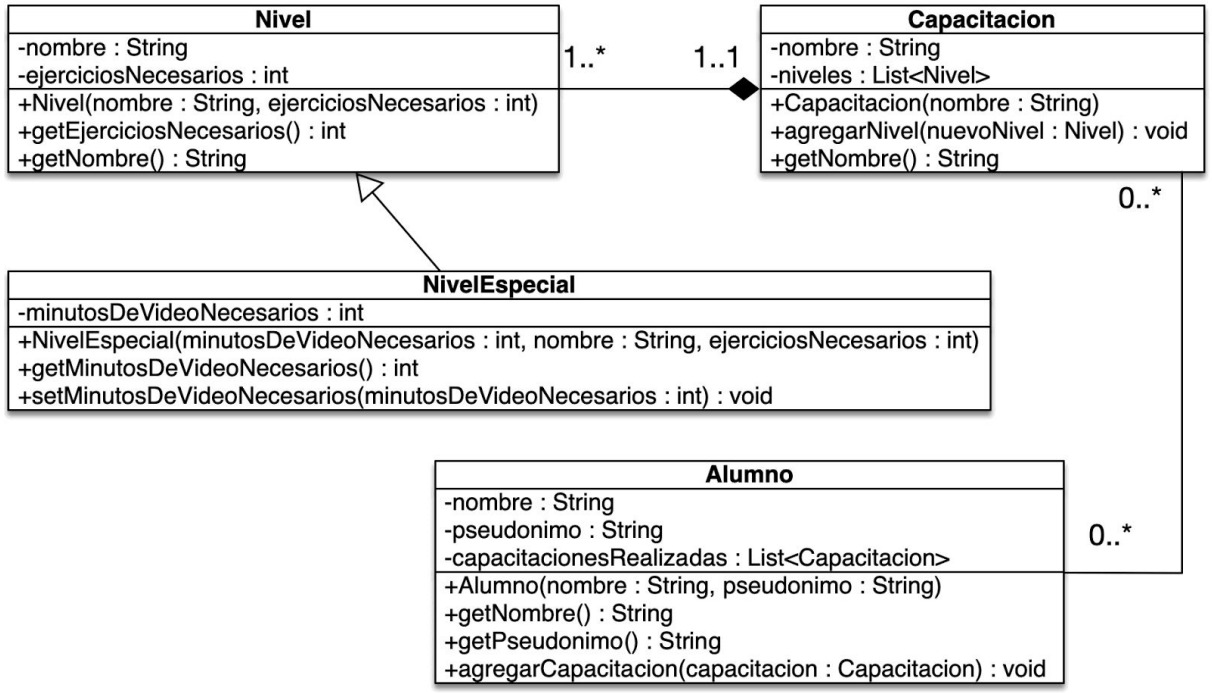
Examen 111Mil – Capacitaciones de programación

Marcela quiere comenzar una empresa que ofrezca capacitaciones de programación online. Con el objetivo de maximizar la cantidad de capacitaciones a ofrecer sin invertir demasiado capital, todas las capacitaciones tendrán un esquema similar. Específicamente, una capacitación estará compuesta por un conjunto de niveles. Para superar un nivel, el alumno deberá realizar un determinado número de ejercicios. Además, habrá niveles especiales donde también se deberá mirar un número determinado de videos para poder superar el nivel. Como Marcela se enteró que cursamos el programa 111Mil, nos contacto para que la ayudemos con el desarrollo.

Ejercicio 1. Implementar desde el diagrama de clases

Marcela realizó un diagrama UML preliminar del sistema y nos pidió que implementemos la clase *NivelEspecial* según el diagrama.



Posible solución

```
public class NivelEspecial extends Nivel{
   private int minutosDeVideoNecesarios;

   public NivelEspecial(int minutosDeVideoNecesarios, String nombre, int ejerciciosNecesarios) {
      super(nombre, ejerciciosNecesarios);
      this.minutosDeVideoNecesarios = minutosDeVideoNecesarios;
   }

   public int getMinutosDeVideoNecesarios() {
      return minutosDeVideoNecesarios;
   }

   public void setMinutosDeVideoNecesarios(int minutosDeVideoNecesarios) {
      this.minutosDeVideoNecesarios = minutosDeVideoNecesarios;
   }
}
```

Ejercicio 2. Implementar un método a partir de un enunciado

Marcela comenzó a trabajar en el mecanismo que permite saber cuándo un alumno termina un determinado nivel. En el caso de los niveles "comunes", el nivel se dará por terminado cuando el número de ejercicios realizado por el alumno sea mayor o igual a los ejercicios necesarios del nivel. Para ello Marcela implementó el siguiente método en la clase Nivel:

```
public boolean terminaNivel(List<Integer> estadoAlumno){
      return estadoAlumno.get(0)>=this.getEjerciciosNecesarios();
}
```

Es decir, cuando se trate de un nivel "común" este método se llamará con una lista que tenga sólo un elemento. Por ejemplo, la lista [13] indicará que el alumno realizo 13 ejercicios.

Sin embargo, en el caso de los niveles especiales, además de realizar ejercicios, el alumno también debe mirar videos que tienen alguna de las siguientes duraciones: 1, 2, o 3 minutos. Por lo que un nivel especial se dará por terminado cuando el número de ejercicios realizado por el alumno sea mayor o igual a los ejercicios necesarios del nivel y además la suma de las duraciones de los videos vistos sea mayor o igual a los minutos de video necesarios para el nivel. Marcela nos pidió que sobre-escribamos el método *terminaNivel* en la clase *NivelEspecial* para realizar este cálculo.

Tenga en cuenta que la lista estadoAlumno siempre tendrá en la primera posición la cantidad de ejercicios realizados por el alumno y en las posiciones subsiguientes la duración individual de cada uno de los videos vistos. Por ejemplo, para la lista [12, 2, 3, 1, 2] el alumno realizó 12 ejercicios y vio 4 videos con duración de 2, 3, 1, y 2 minutos respectivamente.

Posible solución

```
  @Override
  public boolean terminaNivel(List<Integer> estadoAlumno) {
     int minutosDeVideoAlumno=0;
     for(int i=1; i< estadoAlumno.size();i++){
        minutosDeVideoAlumno += estadoAlumno.get(i);
     }

     return super.terminaNivel(estadoAlumno) && minutosDeVideoAlumno >=
minutosDeVideoNecesarios;
  }
```

Ejercicio 3. Implementar un método a partir de un enunciado

Programar en Java la funcionalidad para obtener sólo aquellos niveles de una capacitación en que los ejercicios necesarios para terminarla sean mayor o igual a un valor mínimo pasado por parámetro. Por ejemplo, suponga que la capacitación "Bases de datos" tiene 3 niveles 1, 2 y 3 cuyos ejercicios necesarios son 10, 20 y 30 respectivamente. Si el valor mínimo pasado por parámetro es 15, entonces la funcionalidad sólo retornará los niveles 2 y 3.

Implemente los métodos que considere necesarios indicando para cada uno de ellos a qué clase corresponden.

Posible solución

Clase Capacitacion

```
  public List<Nivel> obtenerNivelesConPuntajeMinimo(int cantEjerciciosMinima){
```

```
        List<Nivel> nivelesConMinimoEjercicios=new ArrayList<>();

        for (Nivel nivel : this.niveles) {
            if(nivel.getEjerciciosNecesarios()>=cantEjerciciosMinima)
                nivelesConMinimoEjercicios.add(nivel);
        }

        return nivelesConMinimoEjercicios;
    }
```

Ejercicio 4. Seguimiento de código

Marcela programó los siguientes métodos en la clase Capacitacion pero no fue muy clara en el nombrado de métodos y variables.

Clase Capacitacion

```
    public void metodoMisterioso(int variableMisteriosa, String nombreNuevoNivel){
        int variableMisteriosa2=otroMetodoMisterioso().getEjerciciosNecesarios()+variableMisteriosa;
        System.out.println(variableMisteriosa2);
        this.niveles.add(new Nivel(nombreNuevoNivel, variableMisteriosa2));
    }

    private Nivel otroMetodoMisterioso() {
        Nivel variableMisteriosa=null;
        for (Nivel nivel : this.niveles) {
            if(variableMisteriosa==null ||
variableMisteriosa.getEjerciciosNecesarios()<nivel.getEjerciciosNecesarios())
                variableMisteriosa=nivel;
        }
        return variableMisteriosa;
    }
```

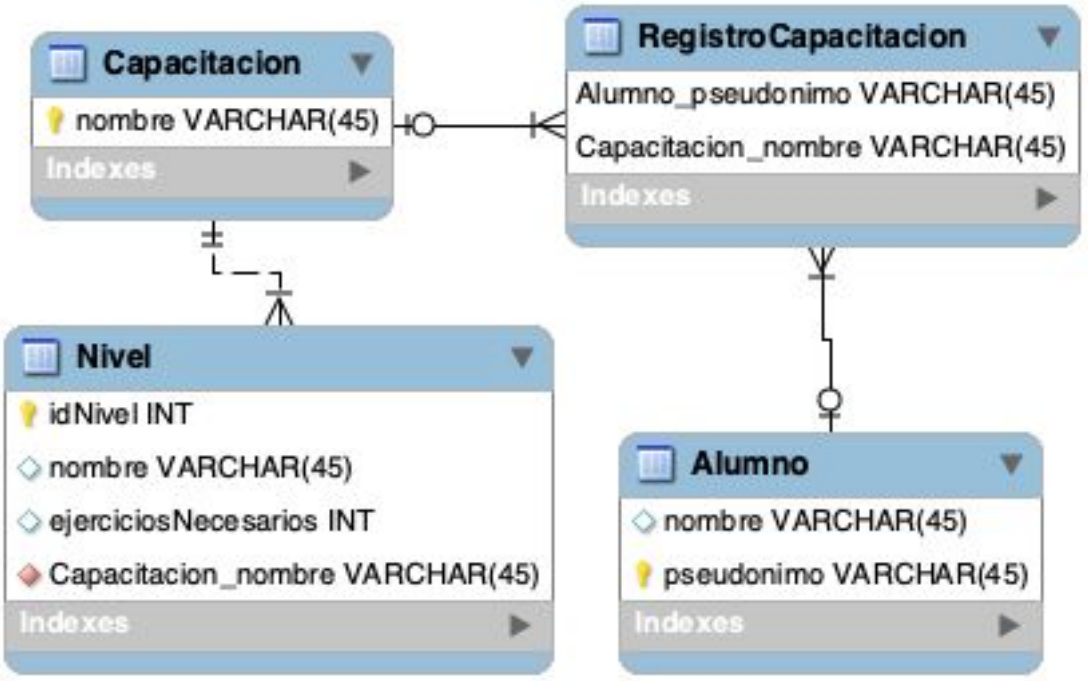
¿Qué imprimirá el programa al ejecutar el siguiente código?

```
        Nivel n1=new Nivel("1", 10);
        Nivel n2=new Nivel("2", 20);
        Nivel n3=new Nivel("3", 30);
        Nivel n4=new Nivel("4", 50);
        NivelEspecial n5=new NivelEspecial(10, "5", 40);

        Curso bd=new Curso("Bases de datos");
        bd.agregarNivel(n1);
        bd.agregarNivel(n2);
        bd.agregarNivel(n3);
        bd.agregarNivel(n4);
        bd.agregarNivel(n5);

        bd.metodoMisterioso(5, "6");
```

Solución

Ejercicio 5. Consulta SQL

Dado el diagrama de entidad-relación parcial, escriba la consulta SQL que liste para cada alumno la cantidad de niveles con ejerciciosNecesarios mayor a 30 que tienen las capacitaciones que realizó. Junto a este valor, liste el pseudónimo y el nombre del alumno.



Además, dadas las siguientes tuplas de ejemplo, determinar el resultado de la consulta.

Alumno	
Tatiana	Tati
Maria	Mari19
Anastasia	Nastya

Nivel			
1	Intro	10	Bases de Datos
2	Clase 1	20	Bases de Datos
3	Clase 2	35	Bases de Datos
4	L1	5	Programacion 1
5	L2	10	Programacion 1
6	Intro	20	Testing
7	Clase 1	30	Testing
8	Clase 2	40	Testing
9	Clase 3	50	Testing

RegistroCapacitacion	
Tati	Bases de Datos
Tati	Programacion 1
Mari19	Bases de Datos
Mari19	Testing

Nastya	Programacion 1

Capacitacion
Bases de Datos
Programacion 1
Testing

Posible solución

SELECT alu.nombre, alu.pseudonimo, count(n.nombre) FROM Alumno alu INNER JOIN RegistroCapacitacion r ON (alu.pseudonimo=r.Alumno_pseudonimo) INNER JOIN Capacitacion cap ON (r.Curso_nombre=cap.nombre) INNER JOIN Nivel n ON (cap.nombre=n.Capacitacion_nombre) WHERE n.ejerciciosNecesarios>30 GROUP BY alu.nombre, alu.pseudonimo

1 Tati Tatiana
3 Mari19 Maria