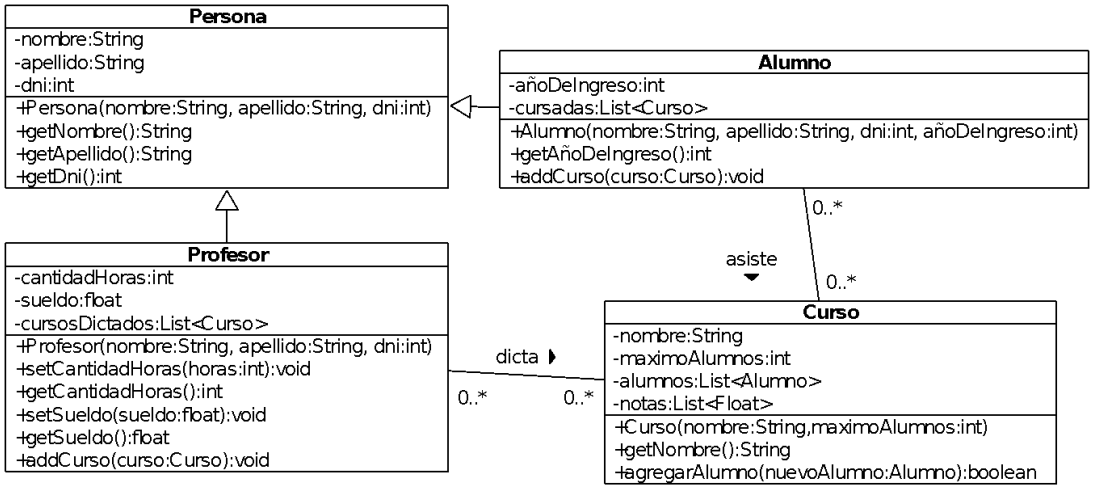
Examen 111Mil – Cursos

La Universidad Nacional de Westeros (UNW) decidió comenzar a desarrollar un sistema para administrar los cursos que se dictan cada año. La aplicación permitirá registrar los cursos que se ofrecen, junto a los alumnos inscriptos en los mismos y los profesores que los dictan. Como saben que cursamos el programa 111Mil nos invitaron a ser parte del desarrollo.

Ejercicio 1. Implementar desde el diagrama de clases

Docentes de la UNW realizaron un diagrama UML preliminar del sistema y nos pidieron que implementemos la clase *Alumno* según el diagrama (el resto de las clases serán implementadas por la UNW). Inicialice los atributos que crea necesarios. Los docentes nos indicaron que, una vez que hayamos implementado la clase *Alumno*, ejecutarán el siguiente código para comprobar que no existen errores:

```
Curso curso111= new Curso("111", 40);
Alumno alumno=new Alumno("Joffrey", " Baratheon", 25346123, 2018);
alumno.addCurso(curso111);
```



Posible Solución

```
public class Alumno extends Persona{
   private int añoDeIngreso;
   private List<Curso> cursadas;

   public Alumno(String nombre, String apellido, int dni, int añoDeIngreso) {
      super(nombre, apellido, dni);
      this.añoDeIngreso=añoDeIngreso;
      this.cursadas=new ArrayList<>();
   }

   public int getAñoDeIngreso() {
      return this.añoDeIngreso;
   }

   public void addCurso(Curso curso){
      this.cursadas.add(curso);
   }
}
```

Ejercicio 2. Implementar un método a partir de un enunciado

Programar en Java la funcionalidad para obtener todos los cursos que dicta un determinado profesor que tengan al menos un alumno que haya ingresado con anterioridad a un determinado año pasado por parámetro.

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

Posible Solución

En la clase Profesor

```
public List<Curso> obtenerCursosConIngresanteAnterioresA(int añoDeIngreso){
    Iterator<Curso> it=this.cursosDictados.iterator();
    List<Curso> cursosConIngresantesAnteriores=new ArrayList<>();
    while (it.hasNext()) {
        Curso curso = it.next();
        if(curso.tieneIngresanteAnteriorA(añoDeIngreso)){
            cursosConIngresantesAnteriores.add(curso);
        }
    }
    return cursosConIngresantesAnteriores;
}
```

En la clase Curso

```
public boolean tieneIngresanteAnteriorA(int añoDeIngreso) {
    Iterator<Alumno> it= this.alumnos.iterator();
    while (it.hasNext()) {
        Alumno alumno = it.next();
        if(alumno.getAñoDeIngreso()<añoDeIngreso)
            return true;
    }
    return false;
}
```

Ejercicio 3. Implementar y documentar

Dada la gran aceptación que tienen los cursos de la UNW, cada curso tiene un número máximo de alumnos que pueden inscribirse. Este número varia según el curso. Dado que los docentes de la UNW están muy ocupados preparando exámenes para sus alumnos, no pudieron terminar la implementación de esta funcionalidad en la clase *Curso*. Por ello, nos solicitaron que implementemos el método *agregarAlumno* según se indica en el diagrama de clases. El mismo recibe por parámetro el alumno a inscribir. El método solo debe agregar al alumno a la lista *alumnos* si aun no se supero el número máximo del curso. En caso de cumplirse esta condición, el método además deberá agregar el curso en la lista *cursos* del alumno inscripto y retornar true. En caso contrario retornará false.

Implemente el método solicitado y elabore la documentación técnica utilizando Javadoc. Incluya tanto la descripción del método como los tags que correspondan.

Posible Solución

```
 /**El metodo agrega un nuevo alumno en el curso solo si aun no se supero el
limite maximo de inscriptos.
     *
     * @param nuevoAlumno alumno a agregar al curso
     * @return devuelve true en caso de poder agregar al alumno y false en caso
contrario
```

```
 */
public boolean agregarAlumno(Alumno nuevoAlumno){
   if(this.alumnos.size()<this.maximoAlumnos){
      this.alumnos.add(nuevoAlumno);
      nuevoAlumno.addCurso(this);
      return true;
   }
   return false;
}
```

Ejercicio 4. Seguimiento de código

Los docentes de la UNW sobrescribieron el método toString en las clases Persona y Alumno de la siguiente manera:

Clase Persona

```
@Override
public String toString() {
   return this.nombre+" "+ this.apellido+" "+ this.dni;
}
```

Clase Alumno

```
@Override
public String toString() {
   return super.toString()+" "+ this.añoDeIngreso+" "+ this.cursadas.size();
}
```

Considerando estos cambios, ¿Qué imprimirá el programa al ejecutar el siguiente código? Recuerde que el método *agregarAlumno* inscribe a un alumno en un Curso siempre y cuando no se haya alcanzado el número máximo de inscriptos.

```
Curso c1=new Curso("Programacion", 4);
Curso c2=new Curso("Bases de Datos", 2);
Curso c3=new Curso("Tecnicas", 3);

Alumno a1=new Alumno("Daenerys", "Targaryen", 123456, 2018);
Alumno a2=new Alumno("Jon", "Snow", 123457, 2017);
Alumno a3=new Alumno("Cersei", "Lannister", 123458, 2018);

c1.agregarAlumno(a1);
c1.agregarAlumno(a2);
c1.agregarAlumno(a3);

c2.agregarAlumno(a1);
c2.agregarAlumno(a2);
c2.agregarAlumno(a3);

c3.agregarAlumno(a2);

System.out.println(a1);
System.out.println(a2);
System.out.println(a3);
```

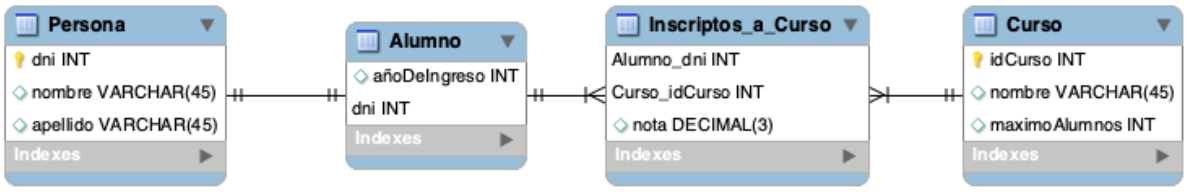
Posible Solución

Daenerys Targaryen 123456 2018 2

Jon Snow 123457 2017 3
Cersei Lannister 123458 2018 1

Ejercicio 5. Consulta SQL

Dado el diagrama de entidad relación, escriba la consulta SQL que liste la cantidad de alumnos ingresantes por año de ingreso (del año mas reciente al mas antiguo). La consulta solo debe tener en cuenta los alumnos que están inscriptos en al menos un curso. Además, dadas las siguientes tuplas de ejemplo, determinar el resultado de la consulta.



Persona		
123456	Daenerys	Targaryen
123457	Jon	Snow
123458	Cersei	Lannister
Alumno		
2018	123456	
2017	123457	
2018	123458	

Inscriptos_a_curso		
123456	1	8
123457	1	5
123458	1	7
123456	2	7
123457	2	4
123458	2	5
123456	3	10
123457	3	6
123458	3	9
Curso		
1	Programacion	30
2	Bases de datos	30
3	Tecnicas	40

Posible Solución

select a.añoDeIngreso, count(a.añoDeIngreso) from Alumno a where a.dni IN (Select i.Alumno_dni from Inscriptos_a_Curso i) group by a.añoDeIngreso order by a.añoDeIngreso DESC

2018 2
2017 1

Ejercicio 6. Hibernate

Dado el diagrama de entidad-relación presentado en el ejercicio anterior y el diagrama UML presentado en el ejercicio 1, escriba la línea del archivo de mapeo de Hibernate (en formato XML o anotación) correspondiente al mapeo del atributo “apellido”.

Posible Solución

<property name=”apellido” type=”java.lang.String”/>