

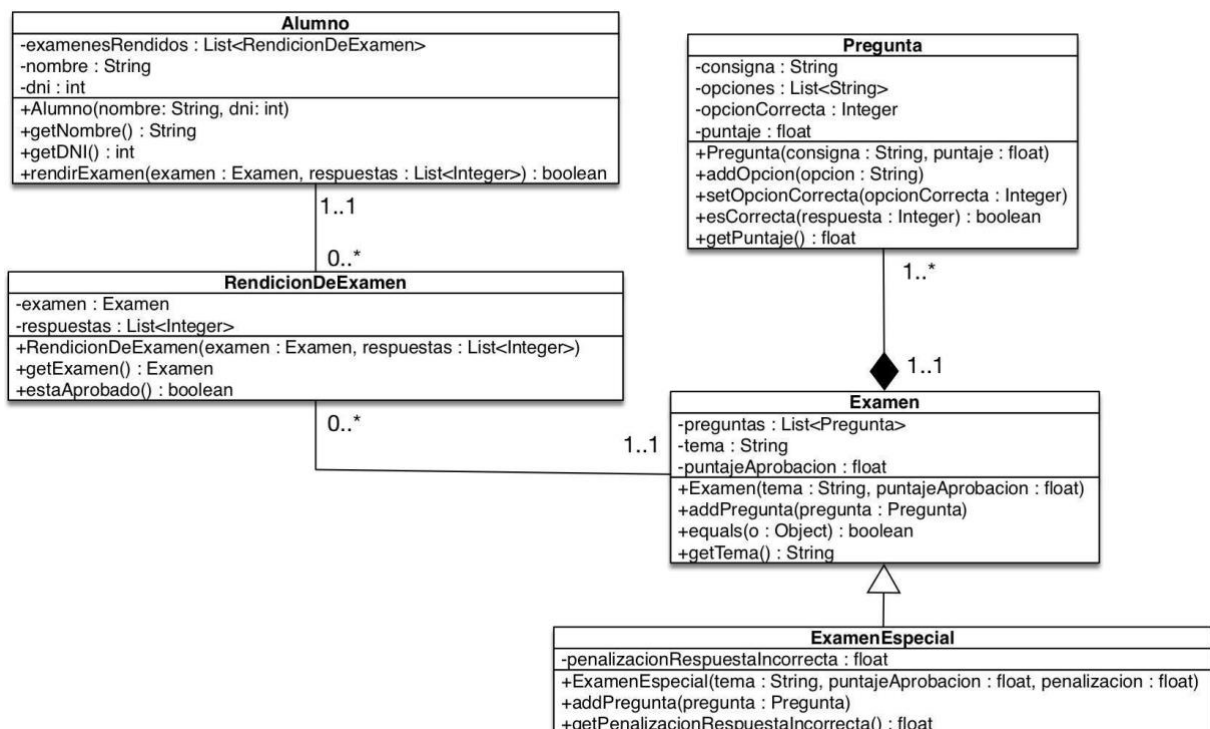
Examen 111Mil – Exámenes Online

Una empresa de cursos online se enteró que cursamos el programa 111Mil y nos contrató para que los ayudemos con su sistema. Específicamente, la empresa está desarrollando un sistema de cursos online donde cada curso se evalúa mediante un examen múltiple choice. Cada examen está compuesto por un conjunto de preguntas donde cada pregunta tiene un puntaje específico. Actualmente, la empresa cuenta con 2 tipos de exámenes uno donde se penalizan las respuestas incorrectas y otro donde no se penalizan.

Ejercicio 1. Implementar desde el diagrama de clases

Personal de la empresa realizó un diagrama UML preliminar del sistema y nos pidieron que implementemos la clase *ExamenEspecial* según el diagrama. Tenga en cuenta que el método *addPregunta* debe agregar una pregunta solamente cuando el puntaje de la misma sea mayor a la penalización por respuesta incorrecta del examen especial. Además, tenga en cuenta que la clase *Examen* implementa el método *addPregunta* de la siguiente manera:

```
public void addPregunta(Pregunta pregunta){
    this.preguntas.add(pregunta);
}
```



Ejercicio 2. Implementar un método a partir de un enunciado

Programar en Java la funcionalidad para que dada una lista de respuestas (`List<Integer> respuestas`) calcular automáticamente la nota de un examen "standard" o especial.

Tenga en cuenta que la nota de los exámenes "standard" es la suma de los puntajes de las respuestas correctas. En el caso de los exámenes especiales, la nota es la suma de los puntajes de las respuestas correctas menos la penalización por cada respuesta incorrecta.

Además, tenga en cuenta que la variable *opcionCorrecta* de la clase *Pregunta* indica el índice de la lista *opciones* donde se encuentra la respuesta correcta. Por ejemplo, si para la pregunta "¿Cuál es la capital de Santa Fe?" la lista *opciones* contiene ["Santa Fe", "Rosario", "Paraná"], *opcionCorrecta* será 0. En este sentido, el método *esCorrecta* de la clase *Pregunta* devuelve dado un índice si la opción es la respuesta correcta o no. En el ejemplo anterior, *esCorrecta* devolverá true solo cuando el valor pasado por parámetro sea 0.

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

Ejercicio 3. Implementar y documentar

El personal de la empresa esta muy ocupado ultimando detalles de la aplicación y están atrasados con algunos requerimientos. Dado que no pudieron terminar la clase *Alumno* nos solicitaron que implementemos el método *rendirExamen* según indica el diagrama de clases. El método debe agregar a la lista *exámenesRendidos* una nueva rendición de examen solamente cuando el alumno no haya aprobado anteriormente el examen que se busca rendir. En caso de agregar una nueva rendición el método retornará true (false en caso contrario).

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

Ejercicio 4. Seguimiento de código

El personal de la empresa sobrescribió el método `toString` en las clases *Alumno* y *RendicionDeExamen* de la siguiente manera:

Clase *Alumno*

```
@Override
public String toString() {
    String historiaAcademica = nombre + ": ";
    Iterator<RendicionDeExamen> it = examenesRendidos.iterator();
    while (it.hasNext()) {
        RendicionDeExamen rendicion = it.next();
        historiaAcademica += rendicion.toString() + " *** ";
    }
    return historiaAcademica;
}
```

Clase *RendicionDeExamen*

```
@Override
public String toString() {
    String aprobado = "Desaprobo";
}
```

```

    if(examen.estaAprobado(respuestas))
        aprobado="Aprobo";
    return examen.getTema()+" (" +aprobado+)";
}

```

Considerando estos cambios, ¿Qué imprimirá el programa al ejecutar el siguiente código? Recuerde que el método *rendirExamen* agrega a la lista *exámenesRendidos* una nueva rendición de examen solamente cuando el alumno no haya aprobado anteriormente el examen que se busca rendir.

```

Examen e1=new Examen("Capitales de provincia", 2);

Pregunta p1=new Pregunta("¿Cual es la capital de Santa Fe?", 2);
p1.addOpcion("San Fe");
p1.addOpcion("Rosario");
p1.addOpcion("Parana");
p1.setOpcionCorrecta(0);

Pregunta p2=new Pregunta("¿Cual es la capital de Corrientes?", 1);
p2.addOpcion("Parana");
p2.addOpcion("Corrientes");
p2.addOpcion("Goya");
p2.setOpcionCorrecta(1);

Pregunta p3=new Pregunta("¿Cual es la capital de Misiones?", 1);
p3.addOpcion("Obera");
p3.addOpcion("Puerto Iguazu");
p3.addOpcion("Posadas");
p3.setOpcionCorrecta(2);

e1.addPregunta(p1);
e1.addPregunta(p2);
e1.addPregunta(p3);

Alumno a1=new Alumno("Juan", 123876456);
List<Integer> respuestasE1=new ArrayList<>();
respuestasE1.add(0);
respuestasE1.add(2);
respuestasE1.add(2);
a1.rendirExamen(e1, respuestasE1);
System.out.println(a1);

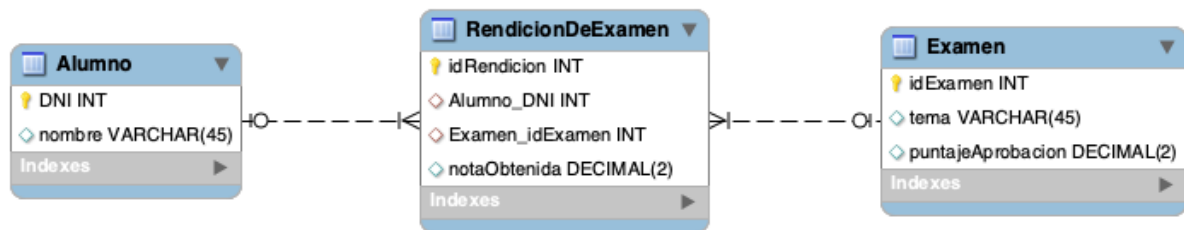
Alumno a2=new Alumno("Alberto", 193762086);
List<Integer> respuestasE2=new ArrayList<>();
respuestasE2.add(1);
respuestasE2.add(0);
respuestasE2.add(1);
a2.rendirExamen(e1, respuestasE2);
System.out.println(a2);

respuestasE1=new ArrayList<>();
respuestasE1.add(1);
respuestasE1.add(1);
respuestasE1.add(2);
a1.rendirExamen(e1, respuestasE1);
System.out.println(a1);

```

Ejercicio 5. Consulta SQL

Dado el diagrama de entidad-relación parcial, escriba la consulta SQL que liste para todos los exámenes rendidos al menos 1 vez por un alumno, la cantidad de veces que el alumno rindió ese examen. Junto a la cantidad de veces que rindió, liste el nombre del alumno y el tema del examen.



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

Alumno	
12312312	Han Solo
23423423	Luke Skywalker
34534534	Darth Vader

Examen		
1	Pilotaje de Halcon Milenario	4
2	Telepatia proyectiva	7
3	Pelea con sable de luz	7

RendicionDeExamen			
1	12312312	1	1
2	12312312	1	2
3	12312312	1	4
4	23423423	2	8
5	23423423	3	5
6	23423423	3	8
7	34534534	2	8
8	34534534	3	9