Examen 111Mil –

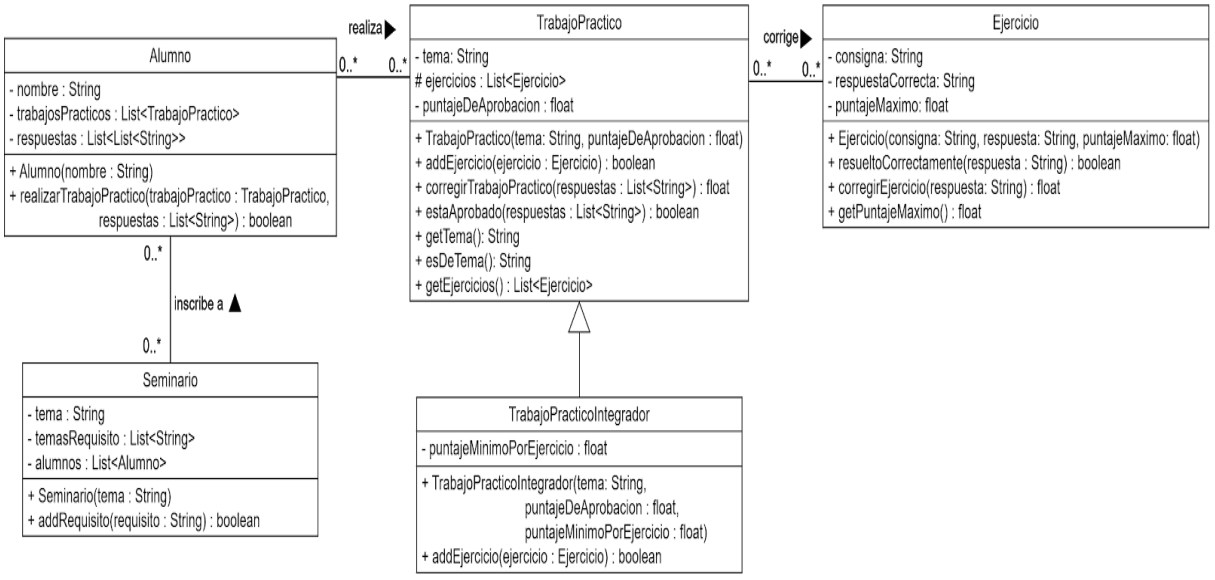
Además de TrabajosPracticos... TrabajosPracticosIntegradores

La Facultad de Ciencias Humanas brinda seminarios para sus estudiantes. Para aprobar los seminarios a los que se inscribieron, los estudiantes deben aprobar diferentes trabajos prácticos. Dada la gran cantidad de alumnos inscriptos en la Facultad, se necesita un sistema que organice los seminarios y sus inscripciones, los alumnos y los trabajos prácticos que realizaron, y finalmente, los trabajos prácticos disponibles.

Como saben que cursamos el programa 111Mil los profesores de la Facultad se pusieron en contacto con nosotros para que los ayudemos a construir el sistema.

Ejercicio 1. Implementar desde el diagrama de clases

Los profesores de la Facultad realizaron un diagrama de clases preliminar del sistema y nos pidieron que implementemos la clase **`TrabajoPracticoIntegrador`** de acuerdo con el diagrama. El resto de las clases serán implementadas por los profesores.



Tenga en cuenta que el método **`addEjercicio(ejercicio : Ejercicio) : boolean`** de la clase TrabajoPractico analiza ciertas restricciones y retorna **`true`** si el ejercicio puede ser agregado a la lista de ejercicios y **`false`** en el caso contrario. El método **`addEjercicio`** de la clase **`TrabajoPracticoIntegrador`**, además de chequear las mismas restricciones de la clase

TrabajoPractico, sólo debe permitir agregar aquellos ejercicios cuyo **`puntajeMaximo`** sea mayor al **`puntajeMinimoPorEjercicio`**.

Posible Solución

Ejercicio 1. Implementar desde el diagrama de clases

```
public class TrabajoPracticoIntegrador extends TrabajoPractico{

	private float puntajeMinimoPorEjercicio;

	public TrabajoPracticoIntegrador(String tema, float puntajeDeAprobacion, float
puntajeMinimoPorEjercicio) {
		super(tema, puntajeDeAprobacion);
		this.puntajeMinimoPorEjercicio = puntajeMinimoPorEjercicio;
	}

	@Override
	public boolean addEjercicio(Ejercicio e){
		if(e.getPuntajeMaximo() < puntajeMinimoPorEjercicio)
			return false;
		return super.addEjercicio(e);
	}
}
```

Ejercicio 2. Implementar un método a partir de un enunciado

Se requiere implementar un método para determinar si un alumno aprobó al menos un trabajo practico de un tema dado. En caso positivo el método deberá retornar true y false en caso contrario.

Para ayudarnos en nuestra tarea, los profesores implementaron el siguiente método en la clase **`TrabajoPractico`**:

```
public boolean esDeTema(String tema) {
	return this.tema.equals(tema);
}
```

Además, tenga en cuenta que la variable respuestas almacena las respuestas para cada uno de los trabajos prácticos que rindió el alumno. Por ejemplo, las respuestas para el trabajo práctico que se encuentra en la posición 1 de la lista trabajosPracticos, se encuentran almacenadas en una lista en la posición 1 de la lista respuestas.

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

Posible Solución

Ejercicio 2. Implementar un método a partir de un enunciado

En la clase Alumno:

```
public boolean aproboTema(String tema){

        for(int i=0;i<trabajosPracticos.size();i++){
        TrabajoPractico tp = trabajosPracticos.get(i);
        if(tp.esDeTema(tema))
                if(tp.estaAprobado(respuestas.get(i)))
                return true;
        }
        return false;
}
```

Ejercicio 3. Implementar y Documentar

En una primera etapa del sistema, tanto los trabajos prácticos como los trabajos prácticos especiales se aprobaban (es decir, estaAprobado retornaba true) si el puntaje obtenido era mayor o igual que puntajeDeAprobacion. Ahora, los profesores de la Facultad decidieron cambiar las condiciones de aprobación de un TrabajoPracticoIntegrador, sobreescribiendo el comportamiento de estaAprobado. Para ello, agregaron el atributo **`minimoEjerciciosCorrectos`** en la clase **`TrabajoPracticoIntegrador`**. Ahora, el **`TrabajoPracticoIntegrador`** va a estar aprobado si el alumno obtuvo un puntaje mayor al mínimo de aprobación Y resolvió de forma correcta una cantidad de ejercicios mayor a **`minimoEjerciciosCorrectos`**.

Tenga en cuenta el siguiente método de la clase **`Ejercicio`**:

```
public boolean resueltoCorrectamente(String rta){
        return corregirEjercicio(rta) == puntajeMaximo;
}
```

Implemente el método **`estaAprobado`** y elabore la documentación técnica utilizando Javadoc. Incluya tanto la descripción del método así como también las anotaciones que correspondan.

Posible Solución

Ejercicio 3. Implementar y Documentar

```
        /**
         * Dada una lista de respuestas, determina si el trabajo práctico
         * fue aprobado o no.
         * @param List de respuestas al trabajo práctico para verificar
         * si fue aprobado
         * @return true si el trabajo práctico fue aprobado y false
         * en el caso contrario
        */
```

```
  public boolean estaAprobado(List<String> respuestas) {

        int respuestasCorrectas = 0;

        for(int i=0;i<ejercicios.size();i++)
            if(ejercicios.get(i).resueltoCorrectamente(respuestas.get(i)))
                respuestasCorrectas++;

        return ( super.estaAprobado(respuestas) && respuestasCorrectas >
minimoEjerciciosCorrectos);
  }
```

Ejercicio 4. Seguimiento de Código

Teniendo en cuenta los siguientes métodos:

- Clase **TrabajoPractico**, método **toString() : String**

```
@Override
public String toString() {
      String resultado = "";
      Iterator<Ejercicio> it = ejercicios.iterator();
      while(it.hasNext()){
            Ejercicio e = it.next();
            resultado += e.toString() +" ";
      }
      return resultado;
}
```

- Clase **Ejercicio**, método **toString() : String**

```
@Override
public String toString() {
      return consigna;
}
```

¿Qué se imprimirá al ejecutar el siguiente código?

```
Ejercicio e1 = new Ejercicio("ejercicio1", "respuesta1", 4);
Ejercicio e2 = new Ejercicio("ejercicio2", "respuesta2", 3);
Ejercicio e3 = new Ejercicio("ejercicio3", "respuesta3", 2);
Ejercicio e4 = new Ejercicio("ejercicio4", "respuesta4", 1);
Ejercicio e5 = new Ejercicio("ejercicio5", "respuesta5", 4);

TrabajoPractico tp1 = new TrabajoPractico("geografía económica",4);
```

```
TrabajoPractico tp2 = new TrabajoPractico("geografía política",6);

tp1.addEjercicio(e1);
tp1.addEjercicio(e2);
tp1.addEjercicio(e3);

tp2.addEjercicio(e3);
tp2.addEjercicio(e4);
tp2.addEjercicio(e5);

tp1.addEjercicio(e4);

System.out.println(tp1);
System.out.println(tp2);
```

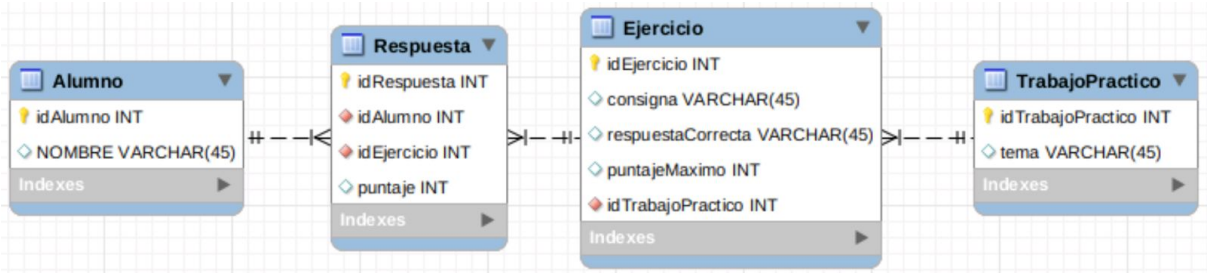
Posible Solución

Ejercicio 4. Seguimiento de Código

ejercicio1 ejercicio2 ejercicio3 ejercicio4
ejercicio3 ejercicio4 ejercicio5

Ejercicio 5. Consultas SQL

Dado el diagrama de entidad-relación parcial correspondiente al sistema que estamos desarrollando, escribir una consulta que liste para cada id de alumno, la suma de puntajes obtenidos en trabajos prácticos de tema "geografía política". Los id de alumno deben estar ordenados de forma descendente de acuerdo al puntaje obtenido.



Indique el resultado de aplicar la consulta a los siguientes datos:

Alumno	
idAlumno	nombre
1	Harry Potter

2	Hermione Granger
3	Draco Malfoy

Ejercicio				
idEjercicio	consigna	respuestaCorrecta	puntajeMaximo	idTrabajoPractico
1	ejercicio1	respuesta1	7	1
2	ejercicio2	respuesta2	7	1
3	ejercicio3	respuesta3	7	2

TrabajoPractico	
idEjercicio	Tema
1	geografía política
2	geografía económica
3	estadística

Respuesta			
idRespuesta	idAlumno	idEjercicio	puntaje
1	2	2	2
2	1	2	4
3	2	3	5
4	3	1	6
5	3	3	1
6	2	1	7

Posible Solución

Ejercicio 5. Consultas SQL

2 9

3 6
1 4

```
SELECT idAlumno, SUM(puntaje) FROM respuesta
    INNER JOIN ejercicio ON respuesta.idEjercicio = ejercicio.idEjercicio
        WHERE ejercicio.idTrabajoPractico IN (SELECT idTrabajoPractico FROM trabajopractico
        WHERE tema = "geografía política")
        GROUP BY idAlumno
        ORDER BY SUM(puntaje) DESC
```