 Diseño de Sistemas 2014

K3001

Jueves TM

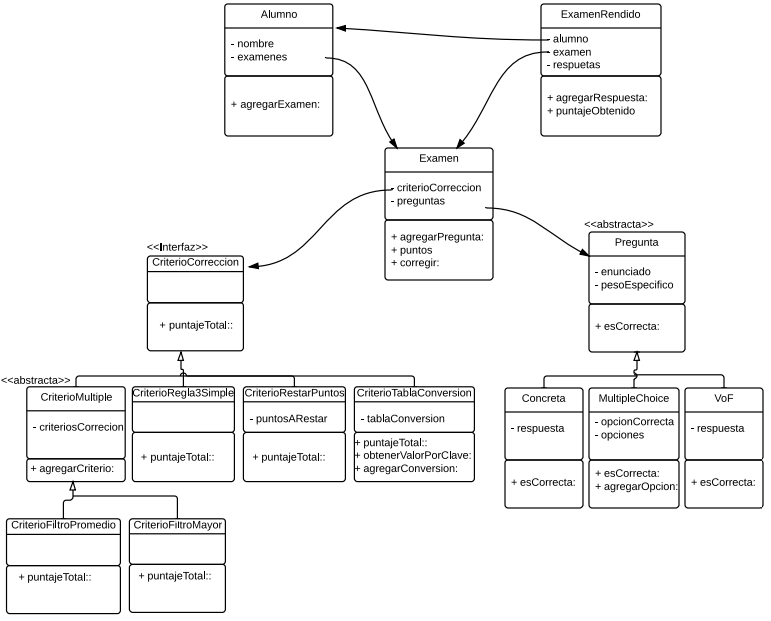
**Corrector de Examenes**

**Grupo 4**

|  |  |
| --- | --- |
| Nombre y Apellido | Legajo |
| Julian Fuks | 147.291-4 |
| Axel Suvalski | 147.290-2 |
| Nicolas Orchow | 146.700-1 |
| Joel Melamed | 146.804-2 |
| Eric Lifszyc | 146.655-0 |



**Diagrama de clases**

****

De las decisiones importantes de diseño que tomamos, podemos destacar:

- Hacer una clase Pregunta, de la cual hereden las subclases PreguntaConcreta, PreguntaVoF, y PreguntaMultipleChoice. De esta manera, no solo nos ahorramos código pudiendo reutilizar funciones, sino que también permitimos redefinir los comportamientos distintos que tengan las subclases en funciones como ver si la respuesta es correcta.

- Delegar responsabilidades en el cálculo del puntaje de un examen rendido, creando una interface CriterioCorrección, la cual es implementada por las subclases CriterioTablaConversion, CriterioRegla3Simple, CriterioRestarPuntos, CriterioFiltroMayor y CriterioFiltroPromedio. El hecho de crear interfaces lo utilizamos para definir al atributo en la clase Examen de tipo CriterioCorreccion, logrando una mayor declaratividad y abstracción del modelo real.   
 Al igual que en el TP anterior, la restricción de que Java sea un lenguaje tipado nos obliga a crear una interfaz cada vez que necesitamos que un parámetro pueda ser de varios tipos distintos (con algún comportamiento en común)

Con esto logramos simplicidad, flexibilidad (ya que por ejemplo, si se agregara una nueva forma de corrección al examen con diferente comportamiento, o bien un nuevo tipo de pregunta, sería muy fácil agregarla y adaptarla a la solución) y menor grado de acoplamiento

- Uno de los puntos más importante de este diseño es que agregamos testing a la solución. Hicimos un unit test para la mayoría de las clases, lo cual nos otorga un poco más de seguridad a la hora de definir la calidad de la solución. Nos permitió identificar algunos errores que por ahí pudimos haber tenido, lo que nos garantiza una mayor robustez en la solución.

La parte central de la solución es la corrección de un examen, la cual la implementamos de la siguiente manera:

(Clase Examen)

**public** **float** corregir(ArrayList<Object> respuestas)

{

**float** puntaje = 0f;

**for**(**int** i=0;i <= \_preguntas.size()-1; i++)

{

**if**(\_preguntas.get(i).esCorrecta(respuestas.get(i)))

{

puntaje = puntaje + \_preguntas.get(i).getPesoEspecifico();

}

};

**return** \_criterioCorreccion.puntajeTotal(**this**.puntos(), puntaje);

}

En el mismo, se puede ver como logramos una buena expresividad y declaratividad, obteniendo primero en base a las preguntas y respuestas, la cantidad de preguntas respondidas correctamente, y, dado esto, le delegamos la responsabilidad de calcular el puntaje total a la clase criterioCorreccion que hayamos instanciado, enviándole ese valor como parámetro, junto a la cantidad de preguntas.