**Pseudocódigo**

//Preguntas

class Pregunta{

    /\*

        Optamos por no hacer un Factory de preguntas en este momento porque

        el comportamiento a la hora de calcular la nota es igual en

        cada una de estas.

    \*/

*String* enunciado

*String* respuestaCorrecta

*String* respuestaAlumno

*double* peso

*boolean* esCorrecta(){

        return respuestaCorrecta == respuestaAlumno

    }

}

//Examenes

class Examen{

*List*<*Pregunta*> preguntas

*CriterioCorreccion* criterioDeCorreccion

*double* notaMinima

*String* alumno

    /\*

        Optamos por no modelar a un alumno como objeto ya que por ahora el

        único comportamiento que tendría sería saber si aprobó un examen.

        Se utilizará un String dentro del examen y se validará por nombre de

        alumno.

    \*/

*double* notaParcial(){

        return preguntas.filter(pregunta *->* pregunta.esCorrecta()).sum(pregunta *->* pregunta.peso())

    }

*double* notaFinal(){

        return criterioDeCorreccion.calcularPuntaje(this)

    }

*boolean* aprobo(){

        return this.notaFinal() >= notaMinimaAprobacion

    }

}

//Criterios de corrección

interface CriterioCorreccion{

*double* calcularPuntaje(*Examen* *examen*)

    /\*

        Este método contará con un algoritmo para calcular la nota final a

        partir de la nota parcial (que es la sumatoria de pesos de las

        preguntas correctas) y el total de preguntas que tiene el examen. Cada

        criterio lo realizará a su forma

    \*/

}

class RestaDePuntos implements *CriterioCorreccion*{

*double* puntosQueSeRestan

}

class ReglaDeTresSimple implements *CriterioCorreccion*{

}

class TablaDeConversionDePuntos implements *CriterioCorreccion*{

}

interface ConjuntoCriterios implements *CriterioCorreccion*{

    /\*

        Esta interface nos permite trabajar de diferentes formas

        con un conjunto de criterios y evitar la repetición de

        lógica en ciertos casos

    \*/

}

class MayorNotaDelConjunto extends *ConjuntoCriterios*{

}

class NotaPromedioDelConjunto extends *ConjuntoCriterios*{

}

**1) Caso de prueba de aprobación del examen de un alumno**

examenJuan.aprobo()

**2) Diagrama de clases**

<http://www.plantuml.com/plantuml/png/bPDTRkem48NVimgB9xWIExX2NC7jKwdImWOcpf0QEJFHsAweKFVUnFaXK2HeNXATu_Fbp4HPEW_YGsaJRS4vbGeMWJoS4bLVMo-62eKKI1l8UImAkYfW7L-n26eFDyr_DfJKvd8EcKLLeUCa2XcpHI25haTCPybNL-J_4KgajiQpSVvltsoXgkx9HNCbngCORXdQC2dTIMiyY-CAn1usXam9FrGurkfu4cW7k3wOWcW3TZgxDfuC3N8_85J26SVX37cKFMYyqViqPccmEbYGj9dqWDDs7mhZxRocjsR9QsnEiLEch2pMldVty5KCl0RSOaJGY5ZOchWJTDl8kqSTaZl8ckIAwGF5rTwZlIxtmdHeKlr1z_3xD_39yiBDFcnFkGXVaqVP9fOAbvWR_bKywJxCtEBUpnUJFvFX7neiCSLxxTredooHyb3QRm00>

**3a)**

class Examen{

*int* preguntasMinimasDeAprobacion

*List*<*Pregunta*> preguntasCorrectas(){

        return preguntas.filter(pregunta *->* pregunta.esCorrecta())

    }

*boolean* aprobo(){

        return (this.notaFinal() >= notaMinimaAprobacion) && (preguntasCorrectas.size() >= preguntasMinimasDeAprobacion)

    }

}

**3b)**

//Preguntas

class Pregunta{

*ManejoError* manejoDeError

*double* peso(){

        return peso - manejoDeError.puntosQueResta(peso)

    }

}

//Criterios de Manejo de Error

interface ManejoError{

*double* puntosQueResta(*double* *peso*)

}

class SinRestaDePuntos implements *ManejoError*{

*double* puntosQueResta(*double* *peso*){

        return 0

    }

}

class RestaDeLaMitad implements *ManejoError*{

*double* puntosQueResta(*double* *peso*){

        return peso / 2

    }

}

class RestaDe1Punto implements *ManejoError*{

*double* puntosQueResta(*double* *peso*){

        return 1

    }

}

Otra opción es directamente utilizar un atributo en la pregunta, pero de aparecer otras opciones en el futuro esta solución permite implementarlas de forma mas sencilla.