Universidad
Europea de Madrid

LAUREATE INTERNATIONAL UNIVERSITIES

DISEÑO Y CODIFICACIÓN

Programación orientada a objetos

Descripción breve

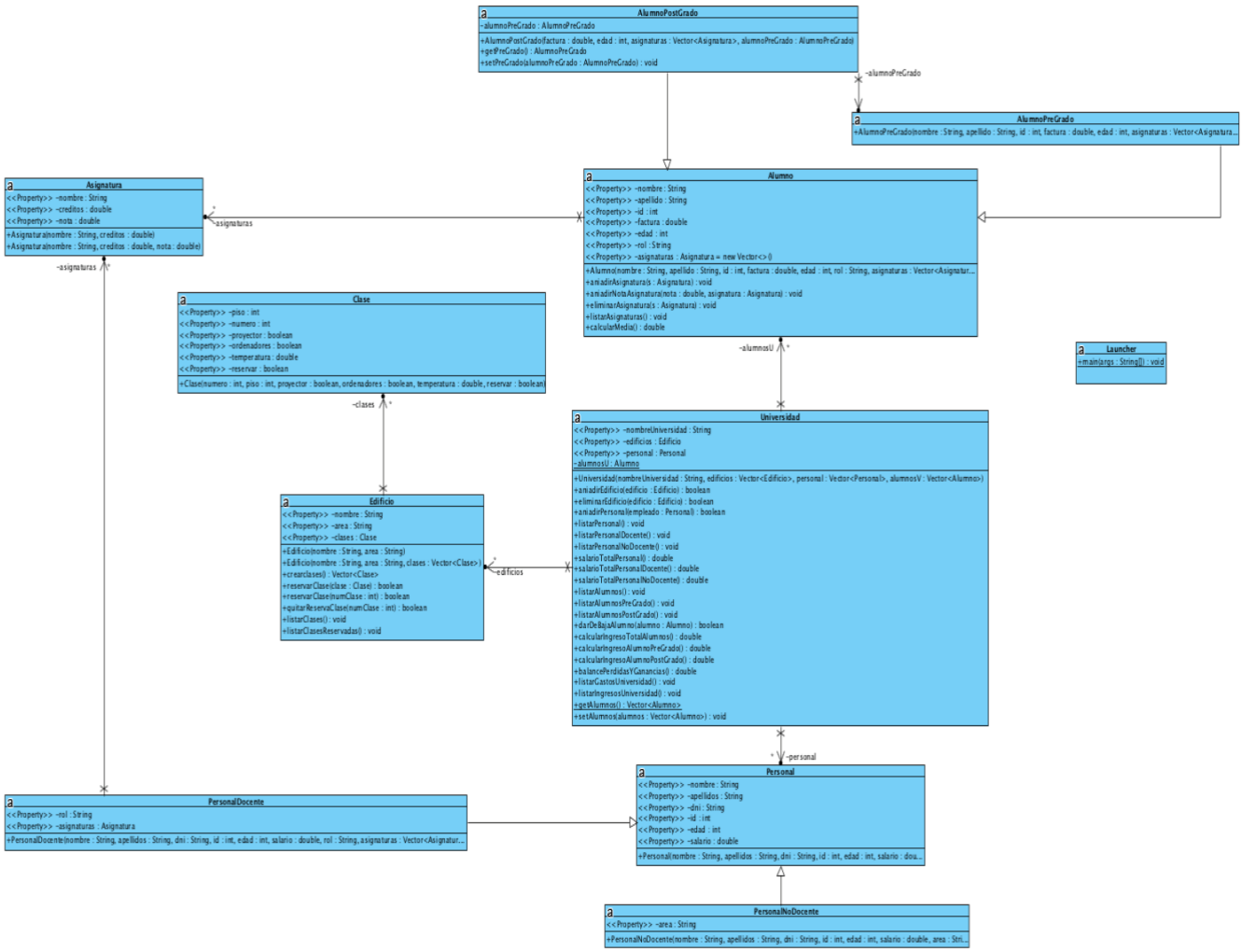
Diseño y codificiación de la estructura de la Universidad Europea de Madrid

José Aceituno García y Diego Fernández Herrero

1- Introducción:

Tal y como nos dice el enunciado, el ejercicio consiste en crear un diagrama de clases con la estructura de la Universidad Europea, y posteriormente codificar el mismo. Para ello, en primer lugar, nos pusimos a esbozar diagramas en papel para entender la estructura del problema, y una vez entendida la estructura y habiendo definido las pautas a seguir nos pusimos a codificar el diagrama. Evidentemente, a medida que íbamos codificando, nos encontrábamos con algún leve problema en el diseño, ya que nos faltó definir más funciones dentro diagrama provisional. Esto no nos supuso ningún contratiempo, todo lo contrario, ya que gracias a esto pudimos definir muchas más funciones de las esperadas.

2- Diagrama UML



El resultado de este proyecto, nos llevó a codificar 10 clases, más una de pruebas. UML realizado con “Visual Paradigm”.

3- Toma de decisiones

Aunque en el diseño de algo tan cerrado, como es el diseño de una universidad, no deberían variar muchas cosas, nosotros hemos tenido unas ideas que quizá rompan un poco con lo esperado. En primer lugar y dado que estamos en la asignatura de Programación Orientada a Objetos, hemos aplicado los conocimientos adquiridos acerca del polimorfismo y sus propiedades. Ya que java acepta herencia simple, hemos sacado partido de ello. Hemos creado dos clases padre (Personal y Alumno) de las cuales sus hijos (Personal docente, Personal no Docente; Alumno Pregrado y Alumno Posgrado) heredaban de ellas. No solo heredaban, sino que, aprovechando esta propiedad, pudimos guardar los objetos de tipo Personal en un mismo vector sin necesidad de distinguir entre los dos tipos de personal. Lo mismo pasó con los objetos de tipo Alumno. Una vez guardados, sacarlos del vector distinguiendo cuál de ellos queríamos sacar fue fácil, gracias al cast y al instanceof.

Respecto a la comodidad para el usurario, hemos generado funciones que por ejemplo te rellenen el vector de clases sin tener que meterlas tu a mano. Otra característica de este programa, es la matriculación de los alumnos de postgrado, ya que hemos considerado que, para poder ser un alumno de postgrado, antes ese alumno, ha tenido que cursar un pregrado. Es por esto, que el constructor de la clase AlumnoPostgrado recibe como argumentos una factura (precio que tiene que pagar anualmente), una edad (ya que este alumno habrá crecido), un vector de asignaturas (que engloben las asignaturas del master) y un alumno de pregrado. Así hemos entendido que debe ser una universidad y así lo hemos realizado. Al momento de matricular a un alumno en un postgrado, inmediatamente ese alumno se desmatricula del pregrado realizado.

4- Conclusiones

Con este proyecto, los alumnos pueden establecer su primer contacto con la Ingeniería del Software, ya que este proyecto además de programar, consistía en realizar una elicitación de requisitos, para poder estructurar el proyecto y llevarlo a cabo a través de una manera más organizada y cuadriculada, y no tan “a lo loco” como en otras ocasiones. En conclusión, nos ha parecido un gran ejercicio para practicar.