Programación Modular y Orientación a Objetos - Laboratorio 2

Requisitos previos

El alumnado debe conocer la perspectiva Java del entorno de desarrollo Eclipse, que fue presentada en el laboratorio anterior. Asimismo, se supone que es capaz de crear clases con atributos y métodos sencillos.

Objetivos

Este laboratorio se centrará en realizar pruebas unitarias mediante el *framework* JUnit con objeto de evaluar el funcionamiento de los métodos implementados en cada una de las clases. Además, se introducirá la notación UML como forma de representar las clases.

Al finalizar este laboratorio, el alumnado deberá ser capaz de:

- Identificar y comprender los diagramas de clases UML.
- Utilizar JUnit para crear pruebas unitarias que permitan comprobar la corrección de las aplicaciones desarrolladas.

Motivación

A lo largo del curso se van a utilizar los diagramas de clases UML. Más adelante, en otras asignaturas como Ingeniería del Software, se va a seguir utilizando la notación UML para representar diversos aspectos del desarrollo de aplicaciones, como los diagramas de casos de uso o de secuencia. Es por este motivo que conviene familiarizarse con la notación UML desde el principio.

Por otra parte, es importante desenvolverse en el uso de las librerías básicas del entorno JUnit, que se usarán a lo largo de la asignatura para efectuar las pruebas unitarias.

Tarea 1: las pruebas unitarias con JUnit

La siguiente figura presenta el diagrama UML de la clase Persona desarrollada en el laboratorio anterior.

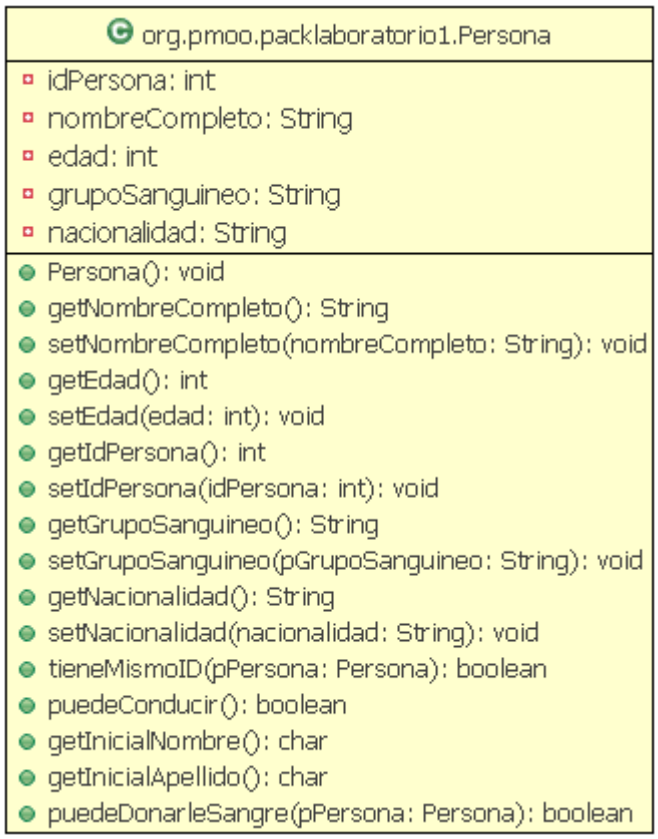

Figura 1 - Diagrama UML de la clase Persona (generado por el *plug-in* AmaterasUML de Eclipse).

En esta actividad se pide escribir los casos de prueba necesarios para comprobar la correcta implementación de la clase Persona. Para ello, habrá que probar la corrección de la operación constructora *Persona*() y los métodos *setNombreCompleto*(), *setEdad*(), *setIdPersona()*, *setGrupoSanguineo()*, *setNacionalidad()*, *tieneMismoID()*, *puedeConducir()*, *getInicialNombre()*, *getInicialApellido()* y *puedeDonarleSangre()*. Nótese que no se pide comprobar la corrección de los métodos *get*, porque si las pruebas se implementan adecuadamente no hará falta, ya que al definir casos de prueba para otros métodos (por ejemplo, los *set*) se estará evaluando a la vez el funcionamiento de los *get*.

MUY IMPORTANTE: Hay que pensar en TODOS los casos de prueba de cada método. Así, cada test de un método será el equivalente a un programa de prueba de los que se utilizaron en la asignatura "Programación Básica", en el que se efectúa una aserción (*assert*) por cada caso de prueba identificado.

Tarea 2: las relaciones entre clases

En este apartado se pide crear una nueva clase, llamada Coche, y relacionarla con la clase Persona. La relación entre ambas clases se establece gracias al atributo *propietario* de la clase Coche, que hará referencia a la persona que es dueña del vehículo. Lo que se pide en esta tarea es:

- Crear la clase Coche dentro de la carpeta src del mismo paquete que contiene la clase Persona. Los atributos de la clase Coche serán los identificativos **matricula** y **marcaModelo** (ambos de tipo String), y **propietario** (de tipo Persona), que indicará qué persona es dueña del coche. Como puede observarse en la figura siguiente, que muestra el diagrama de clases con la relación entre Coche y Persona, la constructora *Coche()* recibe como parámetro los valores de los atributos matricula y marcaModelo, asignando por defecto el valor *null* al atributo de tipo Persona.
- Crear el caso de prueba CocheTest, que incluya verificaciones para la constructora *Coche()* y los métodos *setMatricula()*, *setMarcaModelo()* y *setPropietario()*.
- Crear una suite de prueba que englobe todas las pruebas unitarias desarrolladas para las clases Persona y Coche.

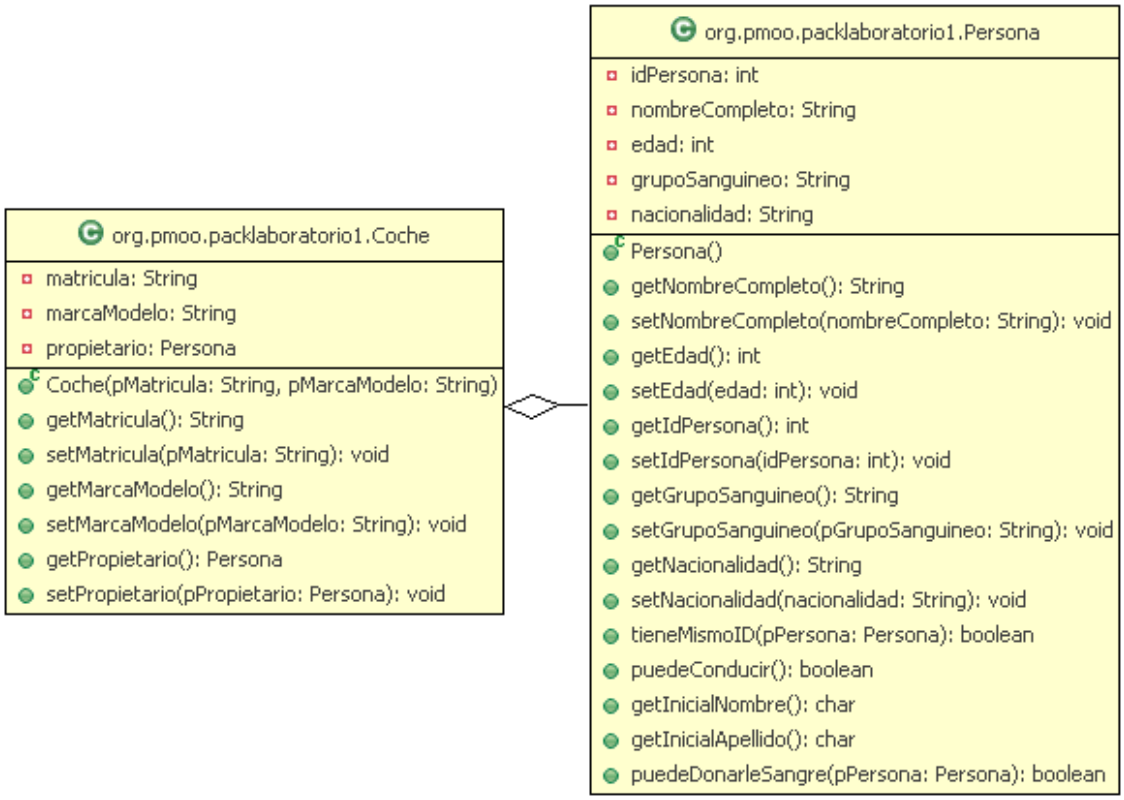

Figura 2 - Diagrama de clases UML, que incluye las clases Persona y Coche.