Validador de Doble Máster

Programación II

Ingeniería de Sistemas de Información + Administración y Dirección de Empresas

CEU San Pablo

Escuela Politécnica Superior

Cristina Abdul Massih

Javier Linares Castrillón

Madrid, Mayo de 2019.

Índice

Introducción.....3

Recursos utilizados 3

Diagrama de GANTT 4

Diseño 5

Análisis de complejidad 6

Referencias 7

Introducción

La finalidad de esta guía es recopilar todas las tareas realizadas para desarrollar la práctica propuesta *"Validador de doble Máster"*. Además, se hará un estudio de la complejidad temporal de los métodos públicos implementados.

El programa en cuestión toma un *doble Máster* y devuelve si es o no válido. Para que sea válido, éste ha de cubrir todas las asignaturas de los *Másteres* simples que lo forman y ha de ser preciso, es decir, que todas las asignaturas del *doble Máster* se encuentren en alguno de los *Másteres.*

Recursos utilizados

Este proyecto ha sido realizado en Linux. Se ha hecho uso de los ordenadores proporcionados por la Universidad y de ordenadores personales, utilizando el software de virtualización *VirtualBox*.

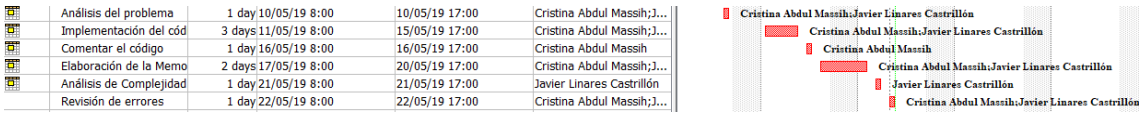
-Se ha hecho uso de herramientas como *Maven,* para la gestión y construcción del proyecto, y *Git,* para gestionar cambios y almacenar elementos de forma remota.

-Las herramientas de documentación utilizadas han sido *ProjectLibre,* para realizar el diagrama de GANTT, y *Umbrello,* para ilustrar el diseño del programa.

-El coste de toda herramienta ha sido gratuito.

Diagrama de GANTT

Las tareas realizadas para llevar a cabo la práctica se exponen en un diagrama GANTT mostrado en la Figura 1.1 y en la Figura 1.2. La duración total del proyecto ha sido de 10 días.



	Análisis del problema	1 day	10/05/19 8:00	10/05/19 17:00	Cristina Abdul Massih;J...
	Implementación del cód	3 days	11/05/19 8:00	15/05/19 17:00	Cristina Abdul Massih;J...
	Comentar el código	1 day	16/05/19 8:00	16/05/19 17:00	Cristina Abdul Massih
	Elaboración de la Memo	2 days	17/05/19 8:00	20/05/19 17:00	Cristina Abdul Massih;J...
	Análisis de Complejidad	1 day	21/05/19 8:00	21/05/19 17:00	Javier Linares Castrillón
	Revisión de errores	1 day	22/05/19 8:00	22/05/19 17:00	Cristina Abdul Massih;J...

Figura 1.1 – diagrama GANTT.

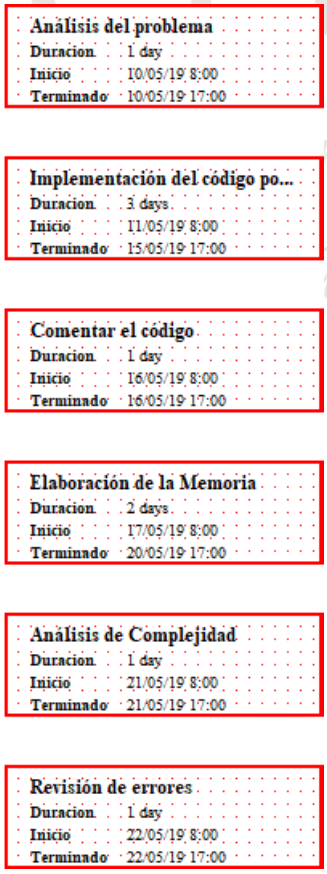


Figura 1.2 – tareas realizadas.

Diseño

El programa consta de dos paquetes, *domain* y *test.* El paquete *domain (*Figura 2.1) incluye las clases *Asignatura.java, Master.java* y *DobleMaster.* El paquete *test* (Figura 2.2) incluye la clase *DobleMasterTest.java*.

Asignatura.java implementa la interfaz *Comparable*, para poder comparar objetos de esta clase. *DobleMaster.java* hereda de *Master.java*.

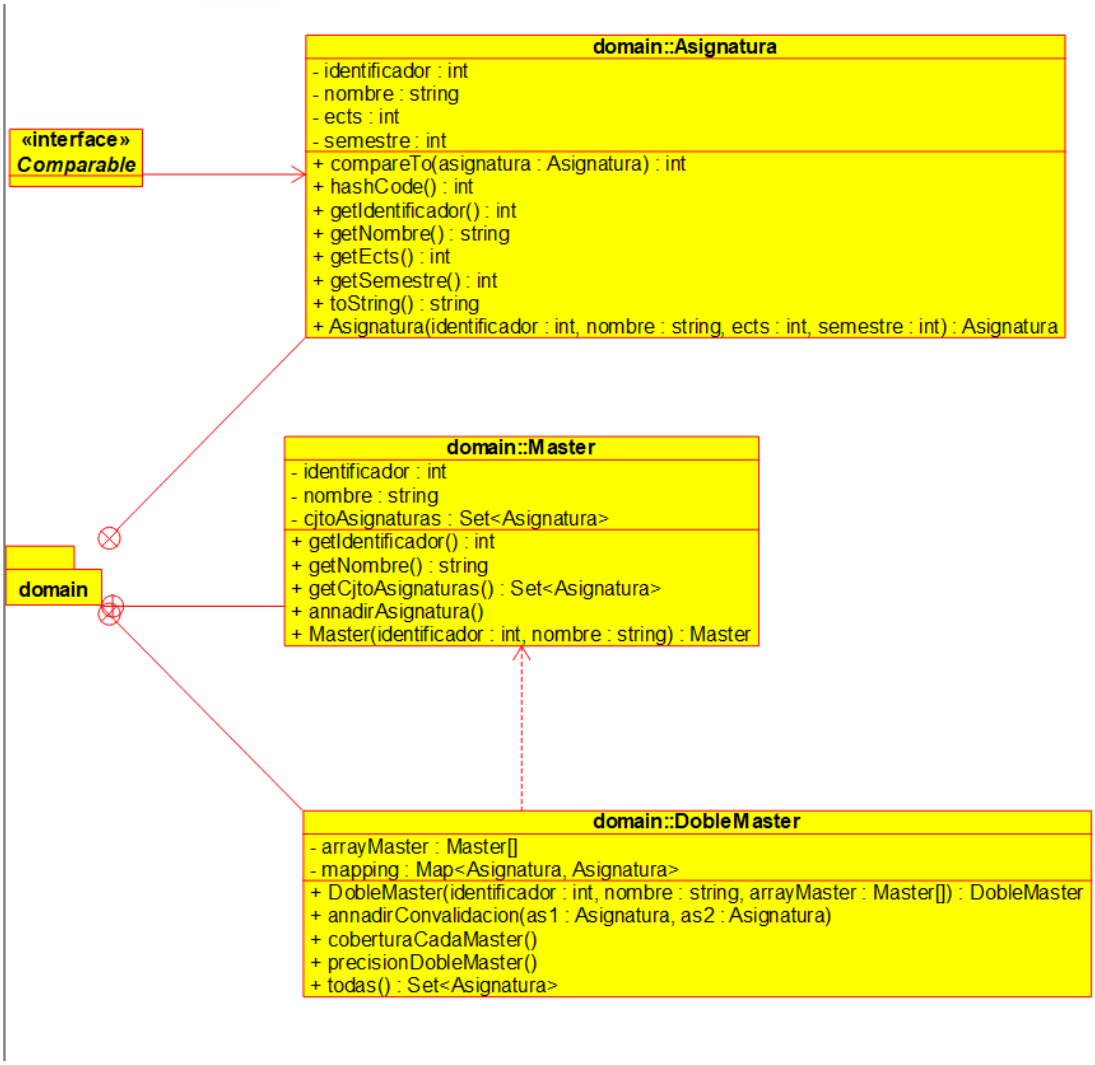


Figura 2.1 - *Paquete domain con sus respectivas clases.*

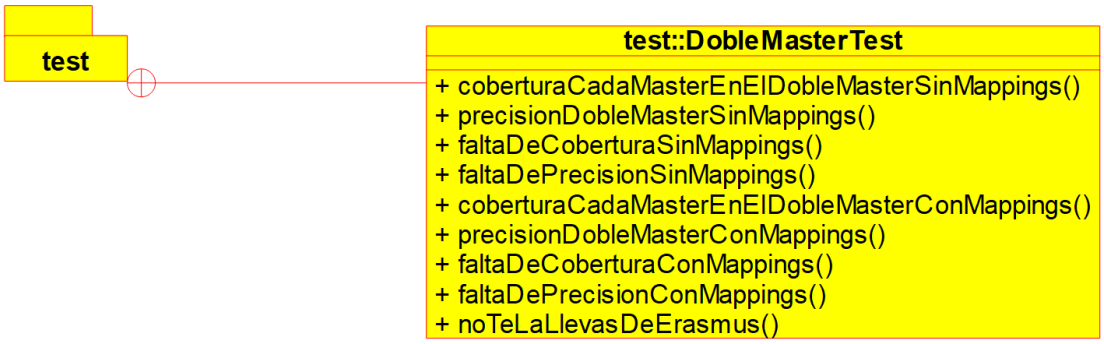


Figura 2.2 - *Paquete test con su respectiva clase.*

Análisis de complejidad

Se estudia la complejidad temporal de los métodos públicos de la clase *DobleMaster*.

- ***annadirConvalidacion():*** *la complejidad temporal de añadir en un HashMap es siempre constante.* ***T ∈ O(1)****.*

- ***coberturaCadaMaster():*** *este método se encarga de comprobar que, para toda asignatura de cada máster, ésta esté en el dobleMaster. La búsqueda de un elemento en un* ***HashSet*** *es constante. La complejidad para ver si un elemento está cubierto es* ***T ∈ O(2)****. Como el conjunto de asignaturas tiene* ***n*** *elementos, comprobar que todos estén cubiertos tiene una complejidad temporal lineal,* ***T ∈ O(n)****.*

- ***precisionDobleMaster():*** *puesto que depende del número de asignaturas, tiene complejidad temporal lineal* ***T ∈ O(n)****.*

- ***todas():*** *obtener un elemento de un* ***HashMap*** *tiene complejidad temporal constante. Convalidadas o no, se añaden todas las asignaturas del conjunto. Ha de repetir esta operación* ***n*** *veces donde* ***n*** *es el número de asignaturas pertenecientes al doble máster. Por lo tanto, este método tiene complejidad temporal lineal* $T \in O(n)$.

En la clase *Master* se encuentra un método cuya complejidad temporal merece la pena ser estudiada.

- ***annadirAsignatura():*** *la operación de añadido en un* ***HashSet*** *es constante, luego este método tiene complejidad temporal constante* $T \in O(1)$.

Referencias

- *Java API – Oracle Docs*

- *Data Structures and Algorithms Analysis in Java. Mark Allen Weiss. Florida International University.*

- *Core Java, Volume I – Fundamentals. Cay S.Horstmann. Prentice hall.*