**ANÁLISIS Y DISEÑO DE SOFTWARE: PRÁCTICA 5**

**EXPLICACIÓN DE DISEÑO**

Nuestro diseño final cuenta con cuatro interfaces:

* ObservableProperty: define una propiedad observable, y permite añadir y quitar observers
* AdjustableTime: define un tiempo de tarea incrementable (extiende de ObservableProperty)
* PropertyObserver: define un observador, e implementa un método propertyChanged() que permite notificar al observador cuando se produce un cambio en una propiedad observable
* Function: define una función ejecutable (para los apartados de la consola)

Comenzamos por implementar el diseño Observable. Para ello contamos con una clase abstracta, DefaultObservableProperty la cual, empleando genericidad, define una propiedad observable genérica. Esta propiedad va a contener un value y un array de observers suscritos. Por otro lado, la clase TiempoTareas, que extiende de la anterior, define los tiempos dedicados y estimados de las tareas. Se trata de una propiedad observable y que, a su vez, observa a los tiempos de sus subtareas asociadas.

En cuanto a la implementación de las jerarquías de tareas, empleamos dos clases: Task y Tasks. Task da forma a la tarea individual, que va a estar formada por un taskName, un TiempoTareas estimado y un dedicado, así como un TreeSet de tareas hijas y una tarea padre. Por otro lado, la clase Tasks va a estar formada por un TreeSet de tareas, y se va a implementar con diseño Singleton.

Finalmente contamos con las clases FullConsole y TextConsole, las cuales dan forma a la consola que emplearemos para trabajar con nuestras tareas. TextConsole cuenta con un HashMap donde almacena las funciones ejecutables (Function) asociadas a un key String, que sería el nombre de la función (start, stop…). Además, TextConsole cuenta con un método para añadir funciones al HashMap. FullConsole, por su parte, extiende de TextConsole y será la que gestione tanto la tarea actual como los comandos que se ejcutarán en nuestra consola, mostrando los cambios por pantalla.