**METODOLOGÍA DE LA PROGRAMACIÓN**

**DIAGRAMA DE CLASES**

**INGENIERÍA DEL SOFTWARE**

RODRIGO MONTILLA

FERNANDO PRIETO

SERGIO OCTAVIO MANCEBO

En esta práctica nos piden que realicemos un diagrama de clases correspondiente al sistema descrito. En este diagrama reflejamos todos los patrones que necesitaría el sistema, además de una breve explicación de las decisiones tomadas a cerca de los patrones de diseño para este programa.

Para la creación de los personajes se ha diseñado un patrón de diseño Factory Method, encargado de la creación de un personaje en particular. Por ejemplo, en la creación de un vampiro.

El patrón Singleton es el encargado de asegurarse que solo existirá una única instancia de una clase. Este patrón nos servirá para leer, escribir o guardar en fichero todos los datos del usuario, personaje, partida, etc.

Voy respondiendo:

- Los patrones factory no están bien. Necesitas fábricas concretas para producir el producto concreto. La fábrica que tenéis no se sabe si produce Vampiros, Licántropos o Cazadores, pero tiene que producir algo concreto (no abstracto, como es la clase Personaje). Podéis seguir el ejemplo que hice en clase el viernes del concesionario, los utilitarios y los tanques. Si estais un poco perdidos, preguntadme el jueves en clase y lo vemos, que será mejor explicarlo en directo. La otra factoría tendría el mismo problema pero además ahí encaja más un patrón estructural y no uno de creación.

- Creo que va a ser mejor que haga algún ejemplo del patrón observer el jueves, con diagrama y algo de código en pizarra. Seguramente le venga bien a más compañeros y quedará más claro. En resumen es que se necesita una interfaz o superclase que haga el trabajo de publicar, que hereden o la implementen los que van a tener que enviar notificaciones. También tiene que haber interfaz, superclase o alguien que implemente los métodos de los subscriptores. El o los publicadores tienen que tener un array de objetos subscriptores mientras que los subscriptores tienen que tener un método "actualizar" que haga el trabajo que se tenga que hacer cuando el publicador notifica. El publicador invoca el método "actualizar" de todos los subscritores que tiene en el array en el momento en el que tiene que notificar algo. Es complicado explicarlo así, lo vemos el jueves en la pizarra o concertamos una tutoría antes.

- Mínimo deberíais poder usar un patrón creacional, otro de comportamiento y otro estructural. Aunque el enunciado admite el uso de varios y no hay una única manera de obtener una solución correcta.

Un saludo.