Taller 5 - Patrones

1. Proyecto

<https://github.com/freeplane/freeplane.git>

Freeplane es un proyecto abierto que permite la creación de mapas conceptuales. Este proyecto busca ser “el segundo cerebro del estudiante”, permitiéndole organizar sus ideas en diagramas útiles. Está programado en java usando OSGi además de Java Swing.

1. Explicación conceptual del patrón y del contexto

El patrón Chain of Responsability es un patrón de comportamiento de objetos que permite manejar que objeto maneja diferentes tareas, pasándoselas entre ellos jerárquicamente. Este funciona dándole la oportunidad a varios objetos de aceptar tareas en el programa.

Los elementos principales de este patrón son:

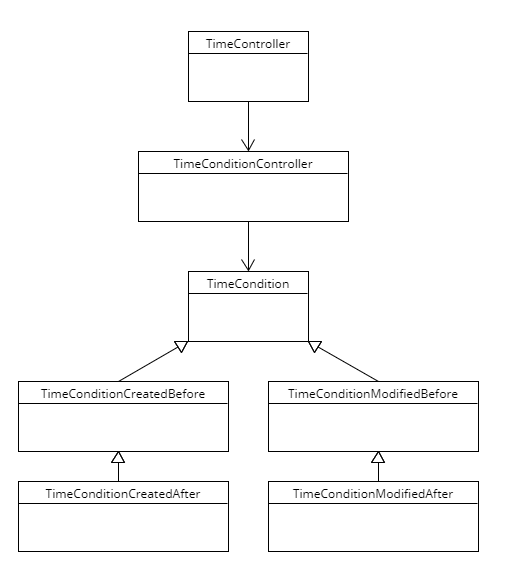
1. Handler: define como se ejecutan las tareas
2. ConcreteHandler: ejecuta las tareas de las cuales es responsable y puede acceder a su sucesor en caso de no poder, le pasa la tarea.

El propósito de utilizar este patrón es permitir que cuando un objeto inferior haga un llamado, este pueda ser manejado por objetos organizados en fila, para que si alguno no puede responderlo, se pase al siguiente.

1. Aplicación del patrón en el proyecto

En este proyecto se utiliza el patrón en /freeplane/src/main/java/org/freeplane/features/time, que representa una de las utilidades que trae consigo la aplicación, que es poder mostrar la hora. Para el desarrollo de esto se crean 4 tipo de TimeCondition que heredan de este mismo objeto. Acá un objeto TimeCondition, o alguna de sus herencias, puede lanzar eventos dependiendo de como se haya creado el objeto de tiempo. En este caso el evento llega a TimeConditionController donde se intenta manejar, si esto no se puede se pasa a TimeController.

1. Aparición del patrón



Acá mostramos un diagrama de clases básico modelando esta parte de la aplicación. En este caso puntual el Handler es el TimeController y el ConcreteHandler es TimeConditionController. El objeto TimeCondition en este caso no se usa como un Handler ya que no maneja eventos, sino funciona como pilar del modelo de herencia.

1. Ventajas

Este patrón permite a un objeto como TimeCondition mandar eventos sin necesidad de preocuparse por quien los va a manejar y controlar, permitiendo que se cree un programa más desacoplado y ordenado.

En este punto del proyecto es positivo usar este patrón ya que los cuatro tipos de TimeCondition pueden lanzar eventos distintos que requieran ser controlados de maneras distintas.

También permite que el manejo de tareas se distribuya entre varios objetos, disminuyendo el acoplamiento.

1. Desventajas

La desventaja principal de usar este patrón en este punto del proyecto es que no se garantiza que todas las tareas tengan respuesta. Si un objeto inferior manda un evento que no se puede responder por ninguno de los objetos Handler, este será ignorado. Para corregir esto se debe analizar todos los eventos posibles que puedan salir y responder adecuadamente a los que lo necesiten.