

Observer

Inicio - Observer

Objetivos

Se ejercitarán problemas donde se aplicará el patrón de diseño observer en diferentes alternativas.

Actividad

Ejercicio 0 – Diferentes alternativas de Observer

En el archivo del material adicional de este trabajo practico se encuentran tres formas diferentes de implementar el patrón Observer en Java. Las implementaciones están simplificadas, pero muestran diferentes niveles de realizar las notificaciones y de ser notificados. En este ejercicio debe analizar las alternativas e indicar ventajas y desventajas de cada una. Si cree conveniente plantear alternativas intermedias entre solución y solución describalas.

Discuta con el ayudante las diferentes alternativas encontradas, las ventajas y desventajas.

¿Cómo extendería las soluciones para los casos en que la misma clase observadora se encuentre observando simultáneamente a más de un sensor?

Ejercicio 1 – Publicaciones

Los miembros de un laboratorio están sumamente interesados en recibir las últimas actualizaciones de los artículos relacionados con sus temas de investigación. Es por ello que cuentan con un sistema de referencias y publicaciones bibliográficas. En este sistema es posible cargar artículos científicos indicando el título, los autores, las filiaciones de cada uno (Universidad, Laboratorio de Investigación, etc), el tipo de artículo, el lugar donde fue publicado y las palabras claves. Por otro lado, además de poder cargar artículos, los investigadores quieren poder suscribirse para recibir artículos de acuerdo a los criterios precedentes. Es decir, si un investigador desea recibir notificaciones para cada nueva publicación de la conferencia “Smalltalks” se registra al sistema y cada vez que se ingresa una nueva publicación de esa conferencia recibe un mensaje de aviso.

Las suscripciones pueden combinar cualquiera de los campos que se incluyen en la carga.

1. Modele el problema utilizando diagrama de clases UML.
2. Indique cual o cuales patrones de diseño utilizó indicando los roles.
3. Implemente los test necesarios relacionados con la funcionalidad de agregar un artículos y verificar las notificaciones.
4. Implemente la funcionalidad en Java.

Ejercicio 2 – Encuentros Deportivos

Se desea desarrollar una aplicación que concentre resultados de distintos encuentros deportivos (partido), y que

distribuya los mismos a distintos servidores interesados en determinada información en particular.

Cada vez que se lleva a cabo un partido, se ingresa en el servidor:

- el resultado (asumamos que simplemente se trata de un texto)
- los contrincantes (lista de nombres de contrincantes)
- el deporte

Es importante tener en cuenta que los servidores que reciben la información, pueden estar suscriptos a un conjunto de deportes, por lo que podría pasar que haya servidores que solo quieran recibir información de Tenis, otros de Fútbol y Basquet y otros de Ping Pong. Esto quiere decir, que debe tener en cuenta a qué información están suscriptos para luego enviar las notificaciones.

Además, existen aplicaciones móviles que le permiten a una persona recibir los resultados de aquellos deportes o contrincantes que les interesa. Para ello, se suscriben a la información relacionada a un deporte y/o un contrincante particular.

En las alternativas, es imprescindible que se minimice la interacción entre el servidor y las aplicaciones móviles. Esto significa que solamente se les puede enviar un mensaje a las aplicaciones móviles cuando se produzca una actualización de alguna de sus suscripciones.

1. Realice un diagrama de clases UML que modele una solución.
2. Implemente en Java utilizando TDD.

Ejercicio 3 – Concursos de preguntas y respuestas

En esta oportunidad necesitamos el desarrollo de un sistema para organizar un concurso de preguntas y respuestas. En el juego se debe determinar, de un número de participantes, el primero que llegue a responder correctamente las preguntas listadas en un cuestionario.

En el juego participan solamente cinco participantes. La mecánica es la siguiente, cada participante debe solicitar sumarse a la siguiente partida. Luego de solicitar la participación, el jugador espera que el servidor del juego le notifique que esta listo para jugar. Con la notificación del servidor también recibe el listado con las cinco preguntas que conforman la partida. Una vez que el jugador fue notificado y conoce el listado de las preguntas, puede enviarle al servidor de a una por vez las respuestas a las preguntas que recibió. Esta acción, puede hacerla solamente después que el servidor le haya enviado la notificación de aceptación y hasta que el servidor decida que terminó el juego.

Cada jugador puede enviar una única respuesta por vez y debe indicar a que pregunta corresponde. Por cada respuesta que recibe el servidor realiza las siguientes acciones:

- Verifica si la pregunta es correcta o no.
- En caso de ser correcta, contabiliza que el jugador que realizó la pregunta la respondió correctamente.
- Notifica al jugador que respondió correctamente.
- Notifica a todos los jugadores de la ronda, el nombre del jugador que respondió correctamente y el enunciado de la pregunta (sin incluir la respuesta).
- Si el jugador respondió correctamente a la última pregunta sin responder, finaliza la partida y notifica a todos los participantes el nombre del ganador.

En caso de no ser correcta la respuesta, el juego solamente notifica al jugador que su respuesta no es correcta.

El juego, está encargado de poder esperar jugadores para iniciar, una vez iniciado recibir las respuestas y notificar de los sucesos como fue indicado, además debe poder finalizarse. Todo lo anterior incluyendo las notificaciones pertinentes.

Por su parte, el jugador debe poder desenvolverse en el desarrollo del juego sin poder realizar acciones no permitidas. Cuando las realiza debe notificarse en consola que esa acción no es permitida. Nota: no se tiene en cuenta el caso en que algún jugador quede fuera de una partida.

Para este ejercicio debe realizar lo siguiente:

1. Modelar una solución orientada a objetos utilizando un diagrama de clases UML.
2. Indicar, si es que es posible, que patrones de diseño utilizo en su solución y los roles de las clases involucradas.
3. Realice un test de unidad que permita verificar que cuando el servidor recibe una respuesta incorrecta, notifica correctamente.
4. Implementar la funcionalidad en Java.

