CLASE MVC

El MVC es un patrón de arquitectura de las aplicaciones de software, es decir, reutilizamos el diseño. Observar el diagrama (en la siguiente página) y el ejemplo Ejemplo-MVC (y las transparencias).

Estas son las ideas más importantes:

- El modelo es independiente de la vista y del controlador. La vista NO debe ver las clases del modelo. Esto quiere decir que dado un modelo yo podría poner una vista u otra sin tocar el modelo, y que yo podría poner un controlador un otro sin tocar el modelo.

  La vista se puede crear posteriormente sin necesidad de modificar el modelo. En vuestra práctica el modelo es la clase TrafficSimulator.java

- El modelo ofrece una interfaz (en la teoría la hemos llamado ObservadorModelo) para que las vistas implementen los métodos con los que el modelo va a notificar los cambios.

  En vuestra práctica la interfaz es Observable.java

- Ciclo:
  - ✓ Las vistas ven el controlador y le avisan de la acción realizada por el usuario.
  - ✓ El controlador recibe las peticiones de la vista y responde a ellas modificando el modelo
  - ✓ El modelo, después de hacer los cambios, se los notifica a las vistas.
  - ✓ Se visualizan los nuevos datos por la vistas (son observadoras del modelo)
- Para ello, las vistas se tienen que registrar al modelo. El modelo va a tener un arrayList que contiene los observadores que van a recibir las notificaciones. Cuando una vista se registra se lo pide al controlador, el controlador se lo pide al modelo, y éste añade la vista al arrayList de observadores.
- Para que una vista pueda recibir las notificaciones además de registrarse al modelo debe implementar la interfaz ObservadorModelo y dar cuerpo a todos sus métodos. Estos métodos se ocupan se visualizar los nuevos datos por la vista.
- La vista se registra en la última línea de su constructora antes de: **this**.setVisible(**true**);
- En el main, haremos:

  Modelo modelo = new Modelo();

  Controlador control = new Controlador (modelo);

  Vista vista = new Vista (control);

- Observar el ejemplo Ejemplo-MVC y ver que cuando se hace el new de la vista ya queda añadida al arrayList de observadores del modelo. A partir de ahí el usuario interactúa con la vista y comienza el ciclo descrito anteriormente.

- Pensad, dada una práctica realizada con este patrón, lo que supone tener que cambiar la vista porque ahora la necesitamos de otra manera. Pues supone nada más que crear la otra vista, registrarla en su constructora, y añadirla al arrayList de observadores del modelo. Y ya está!!!! Reutilizable, no? Sin tener que tocar el resto de la práctica.

- Con esta idea general, estáis listos para entender las transparencias correspondientes a este tema.

Diagrama:

Nota: la línea que expresa que la vista debería implementar la interfaz debería ser discontinua, problemas con el editor…