Resumen Test Doubles

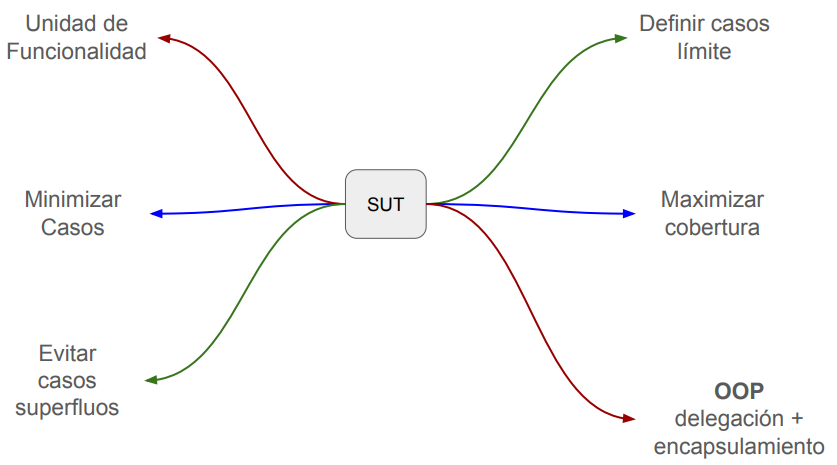
Es un lenguaje de patrones, que utiliza varios patrones para resolver un problema y cada uno de estos patrones ataca una de las variaciones del problema.

Test de unidad: testeo de la minima unidad de ejecucion/funcionalidad. Su objetivo es aislar cada parte de un programa y mostrar que funciona correctamente. Cada test confirma que un metodo produce el output esperado ante un input conocido.

Paradoja del testing: Escribir caos de testing es desealbe, pero a la vez es costoso y aburrido. Por ello testear todos los metodos NO es practico. El objetivo debe ser minimizar los casos y maximizar su cobertura.

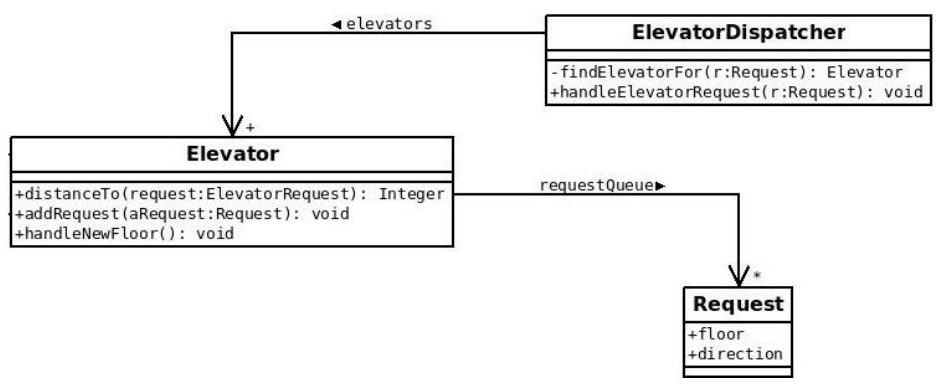
.Testear setters y getters solo tiene sentido si programamos un compilador/parser. Esto entiendo seria porque estariamos desarrollando el set y no testeando que funcione el set, osea testeamos que FUNCIONE.

Fuerzas del problema:

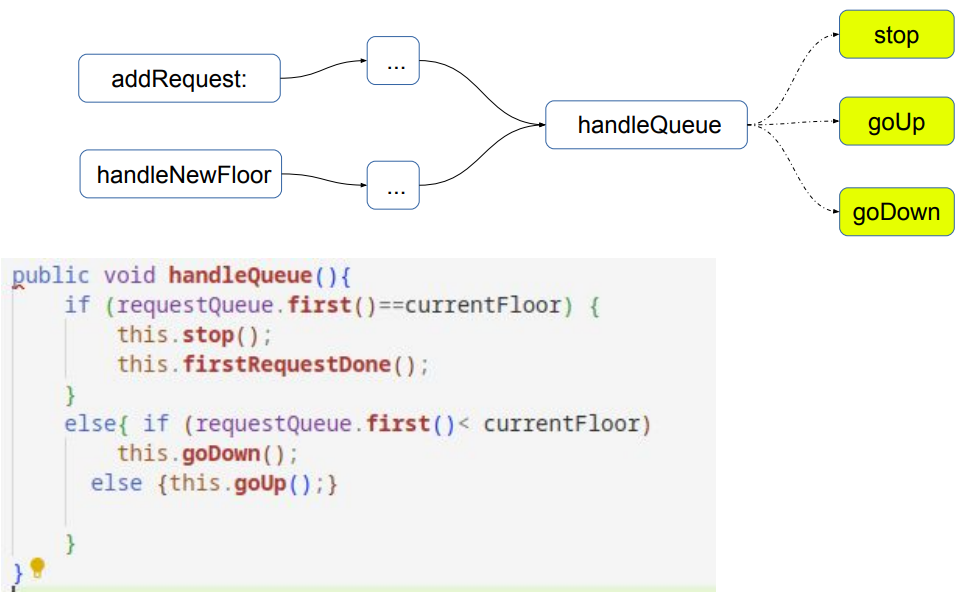


La linea roja es la mas interesante, debido a que al desarrollar orientado a objetos, encapsular funcionalidades puede ser complejo debido al envio de mensajes, la comunicación entre clases, etc. Muchas veces no se puede dejar a un objeto solo pq por si solo no se vale para completar una funcionalidad.

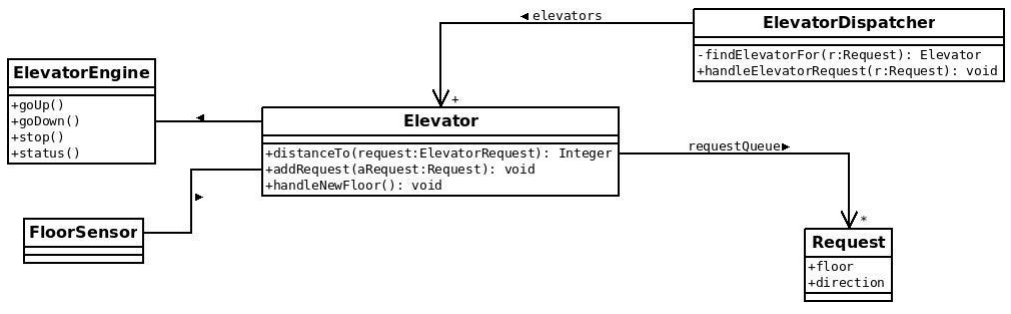
Ejemplo de desarrollo de Ascensores:



En este caso el orden de los pedidos que se hacen en un edificio, tiene en cuenta el piso del edificio y para donde va el ascensor (sube o baja) entonces dependiendo el piso y si el ascensor esta subiendo o bajando sera la entrega de pedidios. (Idea interesante para pensar como programador).



El HandleQueue estaria integrado en el ascensor y en el codigo se puede ver que según el requestQueue y el piso donde se encuentre, se para, sube o baja. Si el primer If es verdadero indica como completada la primera request de la queue.



Version extendida del codigo anterior, entiendo en cuenta el motor del ascensor y el sensor del piso.

En este caso el ElevatorEngine puede tener un bug, pero el resto ande bien entonces nosotros tenemos que tener la posibilidad de abstraernos de esto y poder testear sin necesidad de esta parte.

Aca es donde aparecen los Test Double.

Test Double:

El problema general, es necesario realizar pruebas de un “SUT” (System Under Testing) que depende de un modulo u objeto y el modulo u objeto requerido no se puede utilizar en el ambiente de pruebas (debido a un bug, porque todavia no esta desarrollado o etc).

Las pruebas que se pueden ejercitar: configuraciones validas del sistema, “salidas indirectas” del sistema, logica del sistema y protocolos.

Lenguajes de Patrones – Test Double:

.La idea es crear un objeto que es una maqueta (polimorfica) del objeto o modulo requerido (que no podemos usar, el que no tenemos). Esta maqueta recibira distintos mensajes y devolvera lo que nosotros queremos, puede ser una maqueta simple o compleja según lo que necesitemos y la complejidad / importancia del modulo que no tenemos.

Rangos de implementacion:

.Vamos a usar 4 patrones (Cascaron vacio -> simulacion)

.Test Stub: cascarón vacío. Sirve para que el SUT envíe los mensajes esperados. Se le envia mensajes sin esperar respuesta de este.

.Test Spy: Test Stub + registro de mensajes recibidos.

.Mock Object: test Stub + verification of mensajes recibidos (tambien los registra). Verifica lo que recibe.

.Fake Object: imitación. Se comporta como el módulo real (protocolos, tiempos de respuesta, etc). Este utiliza los mensajes que recibe para generar respuestas similares al objeto real. Se puede decir tambien que simula el comportamiento en tiempo y forma.

.El nivel de complejidad del problema y lo que necesitamos nos indicara cual de estos 4 patrones se utiliza.

.Con el Spy podemos verificar el envio de mensajes, verificar que se siga el envio debido y no se mande dos veces seguido el mismo mensaje por ejemplo.

.Con el Mock verificamos la validez de los comandos enviados con un pedido a alguna clase.

.El Fake, va a ser el mas complejo ya que hay que simular situaciones complejas. Simular el comportamiento.

Tema ascensores: para pensar, a la mañana se predefine que el ascensor vuelva al primer piso ya que es el horario de ingreso del personal y a la tarde que vaya a los pisos altos ya que es la hora de salida del personal.

.Hay que implementar clases según sea necesario. Luego se puede ir incrementado la dificultad y la complejidad del test.