TP Validación y Verificación

Inspección:

-**Definición de roles**

Moderador: Agüero Sebastián, Mujica Juan Manuel.

Secretario: Navarro Pablo.

Lector: Vucetic Ivo.

Inspector: Todos.

Autor: Cátedra.

-Etapa1: Planeamiento de la inspección:

1) El autor entrega el objeto a ser inspeccionado

Al grupo 6 le corresponde el Codigo B entregado por la cátegra.

2) Distribuir material a los participantes

Participantes: Agüero Sebastián, Mujica Juan Manuel, Navarro Pablo, Vucetic Ivo
Cada participante descargo el código en su respectiva computadora.

3) Asignar responsabilidades

Se realiza la subdivisión de los items de la checklist a verificar.

I) Desviación de los objetos
    I.1
II) Omisión de Objetos
    II.1
III) Defectos en los Objetos
    III.1,2,3, ... ,14
IV) Inconsistencia en los Objetivos
    IV.1
V) Ambigüedad en los Objetivos
    V.1,2
VI) Redundancia en los Objetivos
    VI.1,2,3
VII) Efectos Colaterales de los Objetivos
    VII.1,2

Agüero Sebastián:
    I.1
    II.1
    III.1
    III.2
    III.3
    III.4

Mujica Juan Manuel:
- III.5
- III.6
- III.7
- III.8
- III.9
- III.10

Navarro Pablo:
- III.11
- III.12
- III.13
- III.14
- IV.1
- V.1

Vucetic Ivo:
- V.2
- VI.1
- VI.2
- VI.3
- VII.1
- VII.2

-Etapa 2: Presentación de la inspección:

4) Presentar un vistazo general del producto de software a inspeccionar.

Decorator:
- DecoratorCelular.java
- DecoratorContratacion.java
- DecoratorTelFijo.java
- DecoratorTv.java

Excepciones:
- DomicilioRepetidoException.java

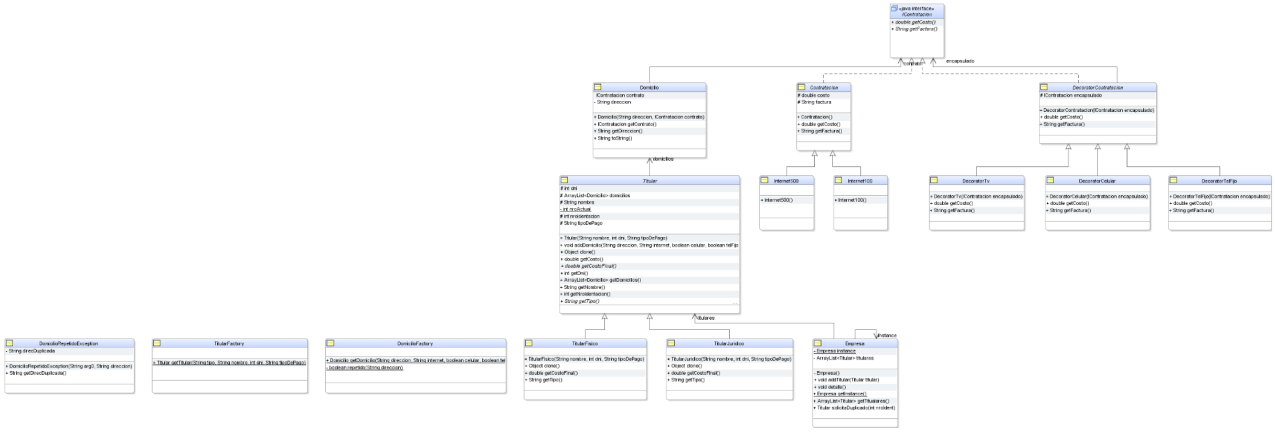
Factory:
- DomicilioFactory.java
- TitularFactory.java

Interfaces:
- IContratacion.java

Modelo:
- Contratación.java
- Domicilio.java
- Empresa.java
- Internet100.java
- Internet500.java
- Titular.java
- TitularFisico.java
- TitularJuridico.java

Se presenta el modelo del problema con el UML siguiente:

-



Etapa3: Preparación de la Reunión:

5-7) Definir el orden en que se realizará la lectura del código, pudiendo ser en forma secuencial, por el flujo de datos, por el flujo de control, etc.

5) Examinar el producto utilizando la lista de comprobación. Las anomalías que se detecten deben ser documentadas y enviadas al moderador.

6) Clasificar las anomalías detectadas para asegurar el mayor aprovechamiento de la reunión y enviarlas al autor.

Luego de que se examinara el codigo, se realizó una lista de las anomalías detectadas.

Lista de anomalías:

La clase TitularFisico.java en el método clone() debería clonar cada domicilio en lugar del arrayList.
En consecuencia la clase Domicilio.java debería implementar la interface cloneable.

13)Defectos en los objetos\Declaración de variables y constantes

En la clase Titular.java los atributos que tienen un indicador de acceso protected, pueden ser private.

15) Defectos en los objetos\Definición de métodos

Error ortográfico clase Empresa.java el método getTitualares()

17) Defectos en los objetos\Definición de métodos
En el constructor de la clase Titular.java los parámetros no son validados. Pero según el contrato no deben ser validados.

23) Defectos en los objetos\Referencia a los datos

En la clase DomicilioFactory en el método repetido(String direccion) puede ser null el ArrayList de domicilios del Titular, eso lanzaría una excepción en la linea 79.

En la clase Titular el método getCosto() puede ser null el ArrayList de domicilios del Titular, eso lanzaría una excepción en la linea 134.

En la clase Empresa en el método detalle() puede no tener ningún titular el ArrayList de titulares de la empresa, eso provocaría un error en la linea 86.

56) Defectos en los objetos\Entrada-Salida

No presenta asserts ni validaciones en la entrada de datos ni en la salida

64) Defectos en los objetos\Comentarios

El metodo imprimirFactura en la clase Titular no esta expresado en lenguaje plano

65) Defectos en los objetos\Comentarios

En las clases de decoratorCelular,TV y telFijo, en los geters debería decir que devuelve el valor que tiene el encapsulado, mas lo que le suma la clase en cuestión. No que le devuelve solo el valor concreto(200,250 y 300).

66) Defectos en los objetos\Comentarios

Clase Contratación\Comentario de la clase\Linea 10
1-Nos dice que tiene dos variables privadas, factura y costo, cuando en realidad son del tipo protected.

Clase Contratación\Comentario del método getFactura y getCosto\Linea 31 y 40
2-En ambos nos dice que son métodos abstractos, cuando en realidad no lo son, sino que ya le están dando un cuerpo.

Etapa 4: La Reunión

Luego de evaluar la lista de anomalías se decidió optar por el criterio de salida A) aprobar el producto pero que el autor realice las modificaciones necesarias para resolver las anomalías que se detectaron. Llegamos a ese criterio debido a la poca cantidad de anomalías detectadas en el proceso de inspección.

Recorridas (tipo prueba de escritorio):

Primera etapa: preparación de la reunión:
Durante esta etapa se realizan los mismos pasos que para el método de inspección de códigos

Segunda etapa: reunión:
Acá se realizan las pruebas de escritorio del conjunto de casos de testo presentados por el tester.

Conjunto de casos de testeo:

Caso 1:
Si se invoca al método Empresa.getInstance().detalle(); sin haberle agregado titulares a su ArrayList de titulares debería informar que la no existencia de titulares en el Array. Sin embargo nos puede arrojar un error relacionado a que no se haya cargado previamente titulares en el Array.

Caso 2:
Si se invoca al método getCosto() de una instancia de la clase Titular, a la que no se le haya agregado previamente algún domicilio. Al no tener como precondición la existencia de algún domicilio, puede ocasionar algún error.

Caso 3:
Si se invoca al método repetido() de la clase DomicilioFactory, siendo que existe algún titular el ArrayList de titulares de la empresa que no tenga domicilios. Al no tener como precondición la existencia de algún domicilio, puede ocasionar algún error.

Caso 4:
Si un titular físico se quiere clonar, se realiza la clonación del array de domicilio. Esto haría que si en el clon se modifica algún domicilio también se modificaría en el titular físico original. Ya que se está clonando un ArrayList que almacena las mismas referencias del original.
Se debería clonar en profundidad. Además la clase Domicilio no implementa la interface Cloneable.