**Trabajo Práctico Nº2**

**Validación y Verificación**

Maria Camila Ezama

Joaquin Fioriti

Mariano Garcia

Ramiro Pruis

Fecha de entrega: 17/9/21

Taller de Programación I

**Inspecciones de código**

Con el uso de la checklist, o lista de comprobación de errores, se procedió al análisis del código para poder detectar fácilmente percances en el mismo. El grupo se conformo por un lector y por tres inspectores. La lectura del código fue en forma secuencial, comenzando por la clase Empresa, y luego de terminar la lectura de cada clase, recurrimos a las otras que habían sido involucradas en los distintos métodos de la primera. Este modo de operar fue realizado con todo el código en su extensión.

Durante el proceso de inspección se encontraron bastantes puntos que no aplicaban al programa como uso de archivos, ciclos for, entre otros. También, se observaron algunos errores que podrían traer problemas futuros en la ejecución:

* En la clase Domicilio, el atributo contrato de tipo IContratacion no tiene declarado ningún indicador de acceso. Esto podría ser un problema, ya que podría modificarse el contrato erróneamente en tiempo de ejecución.
* En las clases *TitularFisico* y *TitularJuridico* en el metodo getCostoFinal() hay un cálculo que se presta a la ambigüedad. respuesta -= respuesta \* 0.10;

Si bien se entiende que primero se resuelve lo de la derecha, y esto se le resta al atributo de la izquierda, quizás un paréntesis en el cálculo podría esclarecer un poco la sentencia.

* En la clase TitularJuridico dice “Clonear” en lugar de clonar como mensaje de excepción. Si bien se entiende que se refiere a clonar, una buena práctica es utilizar los conceptos de forma adecuada.

**Recorridas (tipo prueba de escritorio)**

Comenzamos a hacer los recorridos mentales:

Se instanciaria mediante el método getInstance() la única instancia de “Empresa”. A partir de ahí se puede comenzar a cargarle los titulares mediante su método addTitular. A continuación se deben cargar los domicilios de los titulares con el uso de addDomicilio, que utiliza la clase DomicilioFactory para instanciar uno. Demtro de este método se instancia un domicilio a partir de una dirección y un contrato, que se obtiene a partir de los parámetros del método, pudiendo utilizar o no el patrón decorator implementado. Además, se verifica que el domicilio que se quiera ingresar en ese momento no haya sido ingresado previamente.

Luego, en la clase Domicilio notamos que hay un atributo que no tiene un indicador de acceso declarado, por lo que este se podría modificar accediendo a los domicilios desde el método getDomicilios del Titular, pudiendo generar errores en la funcionalidad.

Por último, se prueba el método imprimirFactura del Titular, el cuál hace llamadas a distintos métodos de la misma clase, pudiendo así comprobar los cálculos obtenidos mediante getCosto y getCostoFinal.

**Recorridas**

La clase empresa es la de mayor nivel jerárquico ya que es la que contiene, indirectamente, a todas las demás clases. Un posible hilo de ejecución es instanciar esta clase, para luego agregar los titulares. Aquí encontramos una posible falla:

Los titulares, a diferencia de los domicilios, no se agregan al arraylist llamando al factory, que es quien tiene como precondición que algunos atributos no pueden ser null, el documento válido, etc. Entonces, el método “addTitular(Titular titular)” recibe como parámetro un Titular, que no necesariamente es obtenido del Factory, pudiendo causar posibles inconvenientes en un futuro, como un atributo nulo.

Un posible caso a tener en cuenta es a la hora de la clonación de un titular físico. En el que se hace una clonación general (super) y luego como tenemos un arraylist, se hace la clonación de ese arraylist. El problema es que no se clona uno por uno los elementos dentro de la estructura. Generando así que el titular clonado y el original posean la misma referencia a los domicilios. Aquí depende el uso que se le de a los clones a lo largo del programa, pero podría generar un problema, ya que la clase domicilio posee un atributo público que no es primitivo, que es el tipo de contratación. Entonces es posible cambiarla en el clon y que este también cambie en el original.

**Conclusión**

En fin, se podría decir que debido al alto nivel de documentación, especificando las condiciones de uso los métodos, el valor que deben tomar los parámetros de los mismos, se logró el desarrollo de un proyecto en el que los errores y las fallas son escasos. Las técnicas de validación y verificación utilizadas permitieron determinar las inexactitudes del código, para luego ser corregidas por el equipo encargado del desarrollo del mismo.