Trabajo Práctico Final

Programación III

Parte II

Grupo 01

Cardelli, Frasca Ponce, Herrera, Martinez Franco

## Introducción

El objetivo de este trabajo consiste en programar un sistema para gestionar contrataciones de servicio de monitoreo y seguridad, así como también la gestión de abonados y facturación.

La implementación del mismo, ha requerido la puesta en práctica de los conocimientos adquiridos en relación a la programación orientada a objetos brindados por la cátedra. Entre ellos se destacan: la herencia, el polimorfismo, el principio de Liskov, el uso de patrones (Singleton, Factory, Decorator, Double Dispatching, MVC, Observer-Observable, State), el uso de excepciones y asertos, la persistencia (serialización binaria, DAO y DTOS), concurrencia, etc.

## Desarrollo del programa

En primera instancia, nos focalizamos en corregir los errores de la parte I. Luego, dialogamos y esquematizamos las posibles soluciones para desarrollar lo pedido. Además, definimos el contrato de nuestro programa, donde establecimos el conjunto de precondiciones, postcondiciones e invariantes de cada método, las excepciones que podrían lanzarse, los asserts en el proceso de desarrollo, etc.

Una vez con el contrato definido, se procedió a realizar la implementación del programa secuencialmente, siguiendo cada una de las etapas del proceso de desarrollo.

## Aspectos del programa

Entre los aspectos más fundamentales de esta nueva entrega podemos resaltar el uso del patrón MVC, para proveer una interfaz gráfica para el usuario. Este está formado por tres componentes, una vista, un controlador y un modelo. En nuestro caso, usamos como modelo el sistema preexistente (con sus modificaciones correspondientes). Cabe destacar, que se implementó, en la ventana, un área de texto que reemplaza a la consola. Esta se utiliza para informar todo lo que ocurre en el sistema (cambios de estado de las personas físicas, informa las excepciones, etc). Opinamos también, que al incorporar esto se facilita la tarea de probar distintas

variables y testear el código, ya que se evita tener que escribir los casos particulares como parte del código.

A su vez, una implementación nueva fue el uso de la persistencia. De esta manera, se permite no perder los datos del sistema cada vez que se cierra el programa. Para ello, utilizamos la serialización binaria y el patrón DAO/DTO. Este último fue requerido ya que la clase Sistema, cuya persistencia es necesaria, no podría ser serializable ya que es un Singleton (ya que de cierta forma estaríamos creando una nueva instancia).

También se utilizó la concurrencia en relación a la solicitud de técnicos, por el hecho de que en la vida real un técnico puede atender a una persona por vez. Así, muchos abonados pueden solicitar simultáneamente el servicio, pero si no hay ningún técnico disponible, estos deben esperar.

El otro patrón utilizado para destacar, fue el patrón State. Este patrón permite cambiar el comportamiento de las personas físicas según si poseen alguna contratación, si no poseen ninguna, o si adeudan.

## Dificultades encontradas

Como todo desarrollo de programa, se han encontrado dificultades a medida que se realizaba su implementación.

Nuestra mayor dificultad fue la interacción correcta del controlador con el modelo. Luego de testear varios casos, llegamos a la conclusión de que la vista recibe correctamente del controlador los mensajes que debe mostrar por su área de texto. A su vez, envía correctamente los eventos a este a medida que se detectan (click del mouse, presionar botones, cerrar la ventana, etc). También se realiza correctamente la validación de los datos ingresados por el usuario, ya que si estos son incorrectos los botones que los precisan permanecen inhabilitados. Si bien varias de las funciones del sistema funcionan como es esperado y mandan los mensajes correspondientes al controlador y este hace que se muestren en la vista, el botón de “Contratar servicio” no parece funcionar, ya que no se muestra ningún mensaje. Sin embargo, fuera de esta interacción, realizamos pruebas de escritorio de los métodos y funcionalidades, que parecerían estar implementadas correctamente.

## Conclusión

Como cierre de esta entrega creemos haber desarrollado un sistema que utiliza la persistencia de los datos, recursos compartidos entre objetos, interfaces gráficas, patrones de comportamiento, etc.

Si bien sabemos que la implementación tiene cosas por mejorar, creemos haber logrado un mejor entendimiento y aprendizaje de los distintos patrones, particularmente MVC, Observer-Observable, State y el uso de la concurrencia.