Trabajo Especial Programación III

Grupo 10

Sistema de Contratación de Servicios
2° parte

Laserna, Lautaro Diego
Viejo, Julián Darío

Descripción general:

La aplicación consiste en un sistema de administración de una empresa proveedora de servicios de internet, celular, tv-cable y teléfono fijo. el programa permite mediante una interfaz gráfica de usuario agregar y eliminar titulares del sistema, así como también ver la información de cada titular, agregar y dar de baja contrataciones y poder visualizar las facturas de cada uno (tanto pagas como adeudadas) y poder pagar dichas facturas.
El programa contiene un emulador de paso de tiempo (EPT) con el cual podemos avanzar de mes en mes para que se generen nuevas facturas.
También posee un simulador de AFIP, el cual irrumpe cada cierto tiempo en el sistema para realizar clonación de las facturas de la empresa, imposibilitando que se agreguen nuevos titulares mientras se realizan las clonaciones, asimismo la AFIP no puede irrumpir si está en curso la creación de un nuevo titular, de ser el caso deberá esperar a que la acción finalice.

Descripcion del codigo fuente:

La aplicación consiste en una clase principal llamada Empresa, la cual es un Singleton. Además de 2 clases externas a la empresa que son el EPT y la AFIP, ambas Singleton también. La Empresa contiene un ArrayList de titulares. La clase Titular es una clase abstracta que es extendida por 2 clases; Titular Jurídico, la cual no acepta clonación de sus facturas; y Titular Físico, la cual si acepta clonación de sus facturas y además implementa el patrón State, ya que este puede tener estado de “Sin Contratación”, “Con Contratación” o “Moroso”. Ambos tipos de titulares son creados por la clase TitularFactory.
Cada titular tendrá un ArrayList de facturas y un ArrayList de contrataciones. Las contrataciones son de tipo IContratacion (interfaz) e implementa el patrón Decorator, además, son creadas con la clase ContratacionFactory. Las facturas por su parte, son de tipo Factura, y se generan automáticamente cada vez que se pasa al siguiente mes en la clase EPT, ya que la clase Titular es Observer del EPT.

La clase AFIP implementa la interfaz Runnable, su método Run se encarga de realizar visitas de forma periódica (cada 1 minuto) a la empresa, con el fin de clonar las facturas de los titulares de la empresa. Esta clase además se extiende de Observable, y su observador es la clase Controlador (patrón MVC), cada vez que la clase AFIP genere las clonaciones y el reporte notifica a su observador para que este se encargue de desplegar una ventana que muestre el reporte generado. Como último detalle, para poder realizar las clonaciones y el reporte, la clase AFIP debe obtener el bloqueo de un recurso compartido en la clase Empresa, en caso contrario deberá esperar a que el recurso se libere para realizar su cometido, en el caso de tener que esperar, la Empresa notificará al Controlador para que despliegue un panel de aviso. ya que Empresa también se extiende de Observable y cuyo observador es también el Controlador.

La clase VentanaPrincipal es la encargada de la representación visual de la mayoría del sistema. Consiste de 2 paneles principales, en uno se encontrará una lista con todos los titulares de la empresa, y en el otro, la información del titular seleccionado, junto con sus contrataciones y facturas. Este segundo panel, se despliega una vez seleccionado un titular. A su vez, esta ventana permite funcionalidades como: Agregar Titular, Eliminar Titular, Pasar un mes, Agregar una Contratación y Eliminar una contratación; los botones de habilitan y deshabilitan según sean necesarios o no (La propia ventana es KeyListener y MouseListener de ella misma). Las acciones de eliminación siempre preguntan si se está seguro antes de realizarse. Además, las acciones no permitidas (como eliminar una contratación de un titular físico en estado moroso) generan carteles que notifican al usuario. Tiene como ActionListener al Controlador, el cual se encarga de realizar los cambios en el Modelo con los datos obtenidos de la ventana.

Las clases VentanaAFIP, VentanaFactura y VentanaCreacionTitular, son ventanas secundarias que realizan una acción específica. VentanaAFIP, como ya mencionamos, aparecerá cuando llegue la AFIP a pedir el duplicado de todas las facturas. La VentanaFactura se encarga de mostrar el detalle de una factura seleccionada. Además nos da la opción de pagarla, si todavia no esta paga. Por último, la VentanaCreacionTitular se abre al querer agregar un titular, tomando el control del bloqueo para que no se pueda abrir la VentanaAFIP. Permite crear un titular y verifica por medio del Controlador si el DNI del titular ya existe en la empresa antes de crearlo, lanzando una ventana de advertencia si es el caso.

La clase PersistenciaBIN, que implementa IPersistencia, es la encargada de persistir el sistema. La lectura se realiza únicamente al inicio del programa, pero la escritura se realiza varias veces, una vez al terminar el main, y una vez cada vez que se realiza una acción en la ventana principal, con el fin de no perder ningún dato al realizar un cambio.

Diagrama de clases del MODELO:

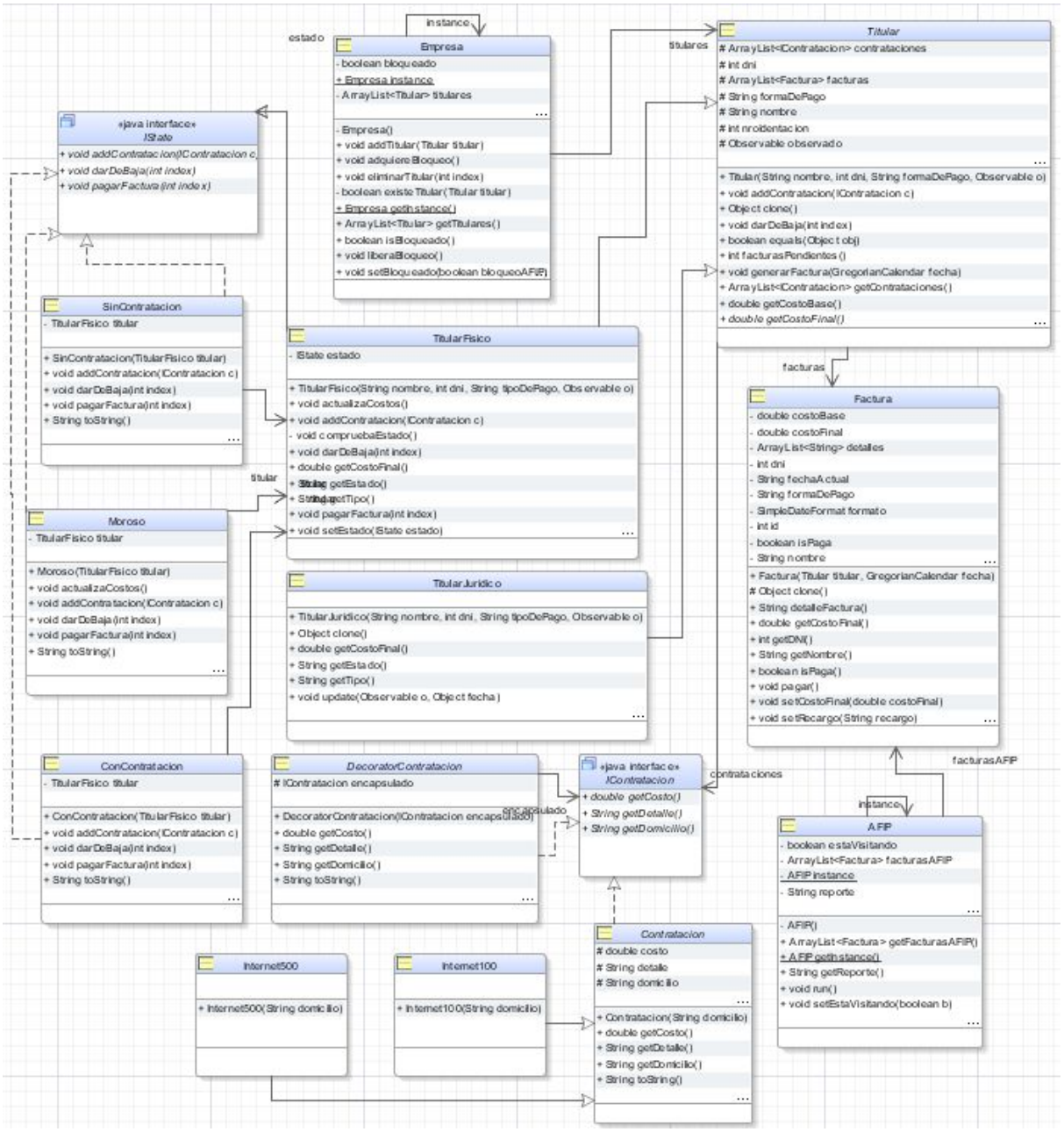
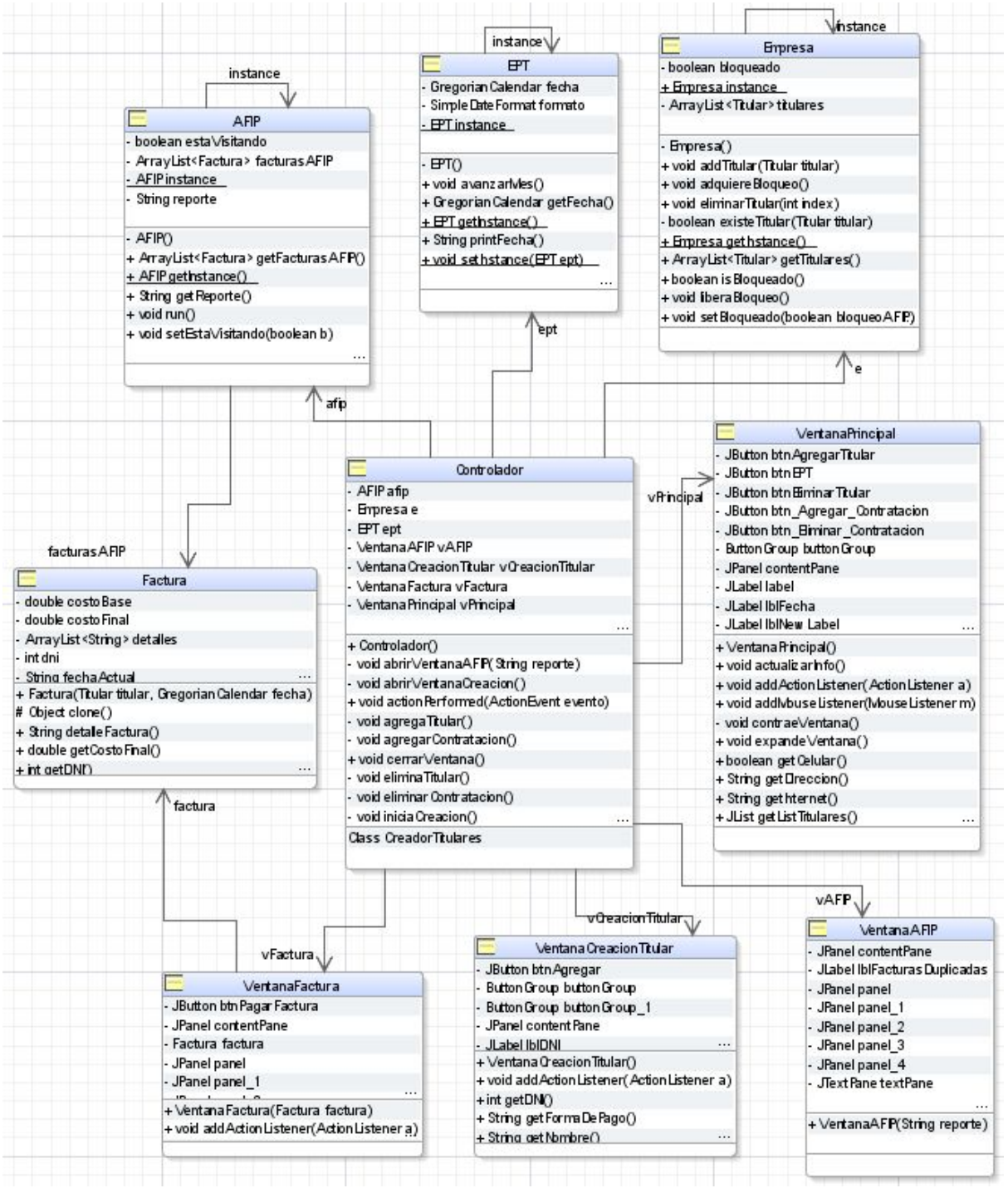
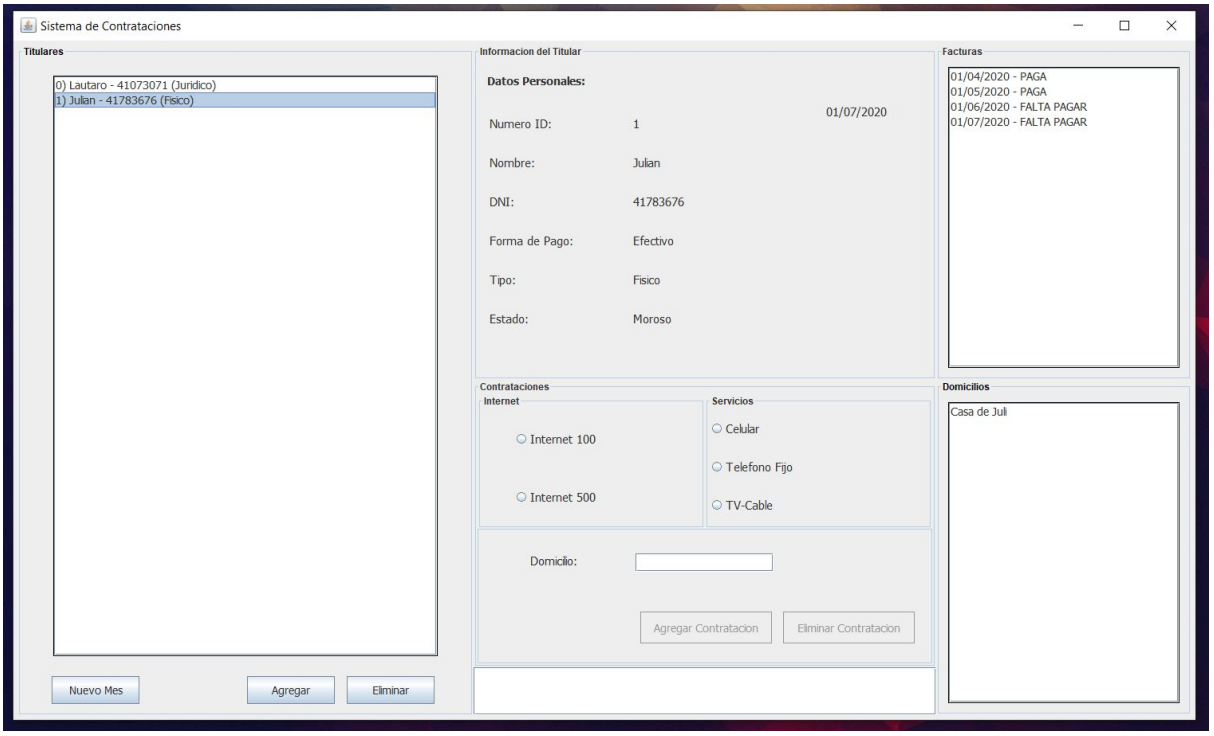
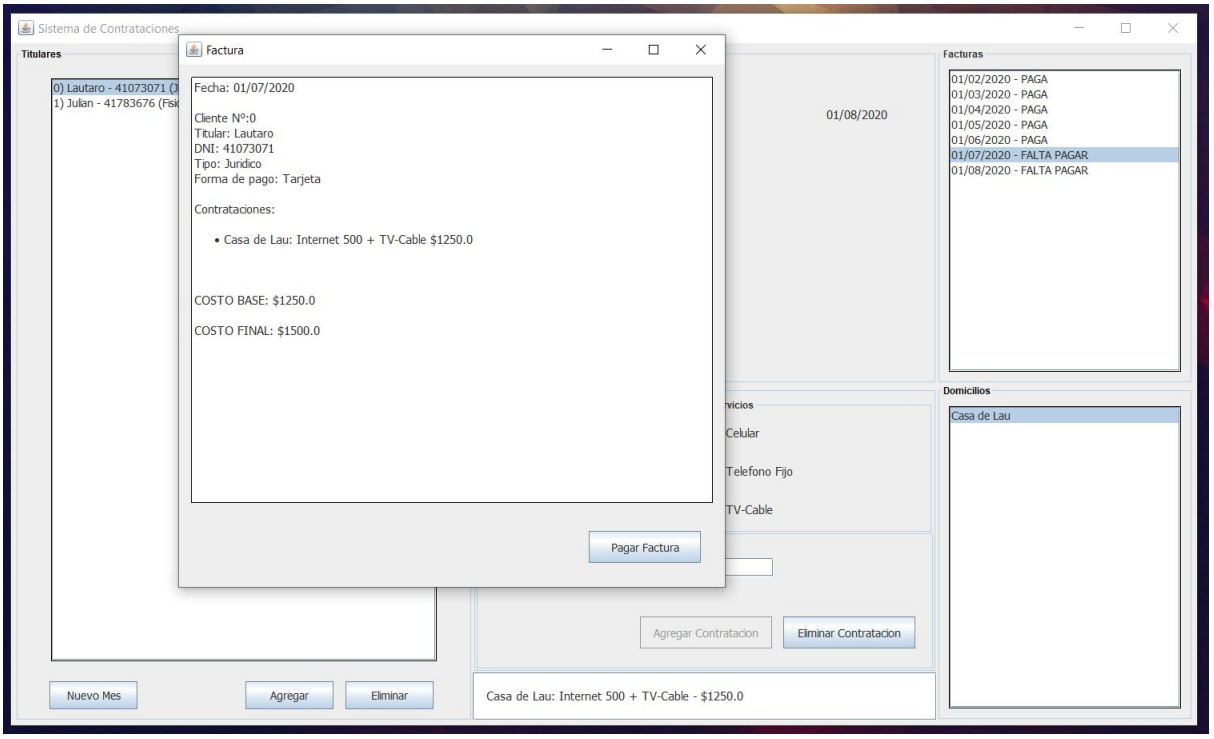


Diagrama de clases de la VISTA-CONTROLADOR:



GUI:


Sistema de Contrataciones
Titulares
0) Lautaro - 41073071 (Juridico)
1) Julian - 41783676 (Fisico)
Informacion del Titular
Datos Personales:
01/07/2020
Numero ID: 1
Nombre: Julian
DNI: 41783676
Forma de Pago: Efectivo
Tipo: Fisico
Estado: Moroso
Facturas
01/04/2020 - PAGA
01/05/2020 - PAGA
01/06/2020 - FALTA PAGAR
01/07/2020 - FALTA PAGAR
Contrataciones
Internet
Internet 100
Internet 500
Servicios
Celular
Telefono Fijo
TV-Cable
Domicilios
Casa de Juli
Domicilio:
Agregar Contratacion
Eliminar Contratacion
Nuevo Mes
Agregar
Eliminar


Sistema de Contrataciones
Factura
Fecha: 01/07/2020
Cliente Nº:0
Titular: Lautaro
DNI: 41073071
Tipo: Juridico
Forma de pago: Tarjeta
Contrataciones:
• Casa de Lau: Internet 500 + TV-Cable $1250.0
COSTO BASE: $1250.0
COSTO FINAL: $1500.0
Pagar Factura
Titulares
01/08/2020
Facturas
01/02/2020 - PAGA
01/03/2020 - PAGA
01/04/2020 - PAGA
01/05/2020 - PAGA
01/06/2020 - PAGA
01/07/2020 - FALTA PAGAR
01/08/2020 - FALTA PAGAR
Domicilios
Casa de Lau
Celular
Telefono Fijo
TV-Cable
Agregar Contratacion
Eliminar Contratacion
Nuevo Mes
Agregar
Eliminar
Casa de Lau: Internet 500 + TV-Cable - $1250.0

Inconvenientes:

1) El primer inconveniente apareció al plantear los cambios a realizar con respecto a la primera parte del proyecto, se consideró que el diseño logrado en la primera etapa necesitaba cambios para poder acoplarse a los requerimientos de la segunda etapa. En la primera parte el diseño consistia básicamente de una clase Empresa la cual contenia una colección de titulares, y a su vez los titulares tenian una coleccion de domicilios, cada domicilio con una contratacion. En este primer diseño la clase Factura no existía y se consideró un método que “generaba una factura” pero consistia en una recopilación de los datos del titular y mostrar los mismos por consola. Cabe destacar que en esta descripción no se detalla el resto del diseño porque no aporta cambios respecto a la segunda etapa (decorator, factory, etc..).
   Para el desarrollo de la segunda parte del proyecto se consideró que había que cambiar esta parte del diseño, y fue implementado en su lugar una clase Empresa que contiene una colección de titulares al igual que antes, pero los titulares poseen ahora dos colecciones diferentes, una de contrataciones y una de facturas. La colección de contrataciones contiene las contrataciones del cliente y por cada una hay un domicilio asociado, y en cuanto a la colección de facturas, es donde se almacenan las facturas generadas cada mes para el titular en base a las contrataciones que posee al momento de generar dicha factura. Una vez realizado este cambio se prosiguió con el desarrollo de la segunda parte del trabajo.

2) Un inconveniente que se presentó durante el desarrollo de la persistencia fue al intentar aplicarla a clases que implementan el patrón singleton (Empresa y EPT) con las cuales no se conseguía obtener una lectura/escritura correcta. Al momento de realizar la lectura daba la sensación de que se estaba produciendo un doble instanciamiento de las clases singleton. Por ejemplo, se declaraba una variable Empresa e = null; al comienzo del método main y luego se realizaba la lectura sobre esta variable, pero si luego se utilizaba el método estático Empresa.getInstance(); la instancia obtenida no se correspondía con la lectura (se perdían los titulares contenidos por la empresa), de forma que para solucionarlo se pensó en usar el método getInstance(); solamente si el archivo para lectura no existe, y en caso contrario se desarrolló un método llamado setInstance(); el cual acopla la lectura con la empresa sin que haya inconvenientes al utilizar el getInstance().

3) Otro problema se presentó en la implementación del patrón Observer, el cual vincula al EPT (observado) y a los titulares (observers) para que, cuando se produzca el cambio de fecha y se avance al siguiente mes los titulares se encargen de la generación de una nueva factura en base a las contrataciones que poseen en el momento. El inconveniente surge de forma similar a lo sucedido con el patrón singleton y la persistencia, básicamente las referencias de observado y observadores se rompían entre escrituras y lecturas, la solución encontrada fue la de implementar un método llamado setObservers(); (método de la clase Empresa) el cual se encarga de reconstruir las referencias luego de realizar las lecturas.

4) La concurrencia nos generó un problema también. Tuvimos dudas de cómo implementarla, y cuando nos decidimos por una implementación, esta tenia muchas fallas. La falla principal que nos ocurría era que al llegar la AFIP, si no se cerraba la ventana podía volver a aparecer, haciendo que 2 AFIPs estén al mismo tiempo. Pudimos resolver este problema con una variable boolean para que no se llame de nuevo si ya estaba activa. Además, al ser una implementación forzada, tuvimos que crear nuevos métodos para “abrir” la ventana de creación de titulares, y poner en marcha un thread cada vez que eso ocurría.

Conclusión:

Como conclusión del proyecto especial, nos gustó en general la propuesta dada, y fue muy satisfactorio terminar el proyecto y poder probar cada detalle para verificar si andaba en la interfaz gráfica. Trabajar de a dos fue muy llevadero, pero nos consumió mucho más tiempo del esperado, pensamos que si hubiésemos sido tres personas el proyecto se hubiese terminado antes, pero aun así nos gustó trabajar en pareja porque la pasamos muy bien programando, además de que pudimos plasmar lo que ambos pensábamos en el proyecto.
Queremos agradecer a la cátedra por darnos la oportunidad de cursar desde nuestras casas, y estar tan atentos a nuestras dudas e inquietudes.