UNIVERSIDAD NACIONAL de MAR DEL PLATA

FACULTAD DE INGENIERÍA

Informe TP 2do Parcial

Grupo 8

Integrantes:

COLIHUINCA IBARRA, JULIAN
OLLER, MARÍA JOSEFINA
LUPO SEVILLANO, MATÍAS EZEQUIEL

INTRODUCCIÓN

Este documento redacta nuestra experiencia al desarrollar la aplicación “Contratación de Servicios” para el Trabajo Práctico grupal del segundo parcial. Aquí relatamos cómo seguimos con el proyecto, nuestras decisiones y técnicas usadas durante el desarrollo y los problemas que encontramos y solucionamos en el testeo de la aplicación.

DESARROLLO

Cambios a clases originales

Factura

Las Facturas fueron un gran problema. Eliminamos las referencias a Persona para evitar el conflicto de la clonación infinita, también eliminamos la referencia a la lista de contrataciones, agregando detalles.

Persona

Agregamos las funciones que se encarga de la lista de Contrataciones,tales como agregar,eliminar o modificar alguna contratación. También se agregó la función de pagar la factura para respetar el diseño original de Persona y el Patrón State. Agregamos la función de precioOriginal que calcula el total de los servicios contratados y sus agregados sin el descuento ni el recargo. De esa manera, si la Factura no está pagada, se mostrará el precio total mencionado anteriormente. En el momento de pagar la Factura, se actualiza el precio total a pagar aplicando descuento y/o recargo por el tipo de pago y/o por el estado Moroso.

Sistema y Prueba

Eliminamos dichas clases ya que, no la necesitamos y actualmente nos manejamos con una interfaz gráfica aplicando el patrón MVC.

Excepciones

No modificamos nada, pero originalmente cuando se capturaba alguna excepción, se imprimía el mensaje en la consola, entonces realizamos cambios necesarios para que dichas excepciones fueran capturadas por el controlador y así mostrar un cartel de advertencia.

Patrón State

Con Contratación,Sin Contratación, Moroso

Desarrollamos dichas clases para el patrón State para la persona física, la cual en el momento de crear un nuevo Socio de tipo Física, su estado arranca en Sin Contratación, luego en el momento de contratar un nuevo servicio, automáticamente el estado se pasa al estado de Con contratación. Desde ahí, cuando se quiera pagar se analiza si tiene deudas, se pasa al estado moroso. Sucede lo mismo con el eliminar Contratación, también se analiza luego de haber eliminado, si todavía hay contratación/es , si no hay nada, se pasa al estado de Sin Contratación.

Debido a las restricciones estipuladas en el enunciado del trabajo, creamos tres excepciones (NoPuedePagarException,NoPuedeContratarException y NoPuedeDarDeBajaException) para que fueran capturadas por el controlador y así lanzar algún cartel de advertencia.

Al principio, pensamos realizar una clase aparte que observara al EPT junto al Gestor de Facturación y se encargue de revisar la cantidad de facturas sin pagar de cada abonado al pasar el mes y pase a estado moroso a aquellos que cumplan la condición, pero esto tuvo un problema que no logramos identificar por lo que decidimos pasar la responsabilidad de este control al Gestor de Facturación.

Patrón Observer/Observable

Gestor de Facturación

Desarrollamos dicha clase que permite gestionar todo el sistema relacionado con la facturación. Esta clase observa al Emulador de Paso de Tiempo y se encarga de generar una nueva factura para cada abonado al pasar el mes, además de revisar la cantidad de facturas sin pagar de los mismos y pasar a morosos a los que tengan dos o más facturas sin pagar al iniciar el nuevo mes.

Emulador de Paso de Tiempo

Es la clase encargada de avanzar el tiempo un mes, además de avisar al Gestor de Facturación para que realice sus acciones correspondientes.

Patrón MVC

Definimos un conjuntos de vista, un modelo y un controlador, para poder separar la lógica de la aplicación de la lógica de la vista.

Controlador MVC

Su clase “Controlador” es una clase encargada de manejar los datos dados por las vistas correspondientes y les pasas dichos datos al modelo para las operaciones que se requieran efectuar.

ModeloMVC

Su clase “SistemaContrataciones” se encarga de hacer prácticamente todo, la cual algunas de las funciones que realiza son las siguientes:

- Agregar un nuevo socio.
- Agregar un nuevo servicio.
- Modificar algún servicio.
- Eliminar un servicio.
- Clonar las facturas (Para AFIP).
- Pagar factura.

VistaMVC

Establecemos un conjunto de vistas para que estas puedan interactuar con el usuario para luego solicitar actualización al controlador.

Las clases de vistas definidas son las siguientes:

- ventanaAltaSocio
- ventanaAgregarServicio
- ventanaModificarServicio
- ventanaQuitarServicio
- ventanaPagar
- ventanaAFIP
- ventanaPrincipal

Interfaces

Creamos las interfaces tales como I_Factura y I_ColeccionDeFacturas para crear la colección de Facturas, para eso también creamos la clase de colección genérica y a su vez ListaFacturas. Al principio no encontrábamos su beneficio, después de haber implementado y usado sus funciones, encontramos su beneficio. Su beneficio es que no hace falta crear las funciones de AMB (agregar,modificar o eliminar), ya vienen incorporadas.

Persistencia

Al principio pensábamos hacerlo con la persistencia XML, pero nos dimos cuenta de que en una clase tiene un constructor privado ya que aplica el Patrón Singleton y aprovechamos dicho patrón para el Emulador de Paso del Tiempo. En algunas clases

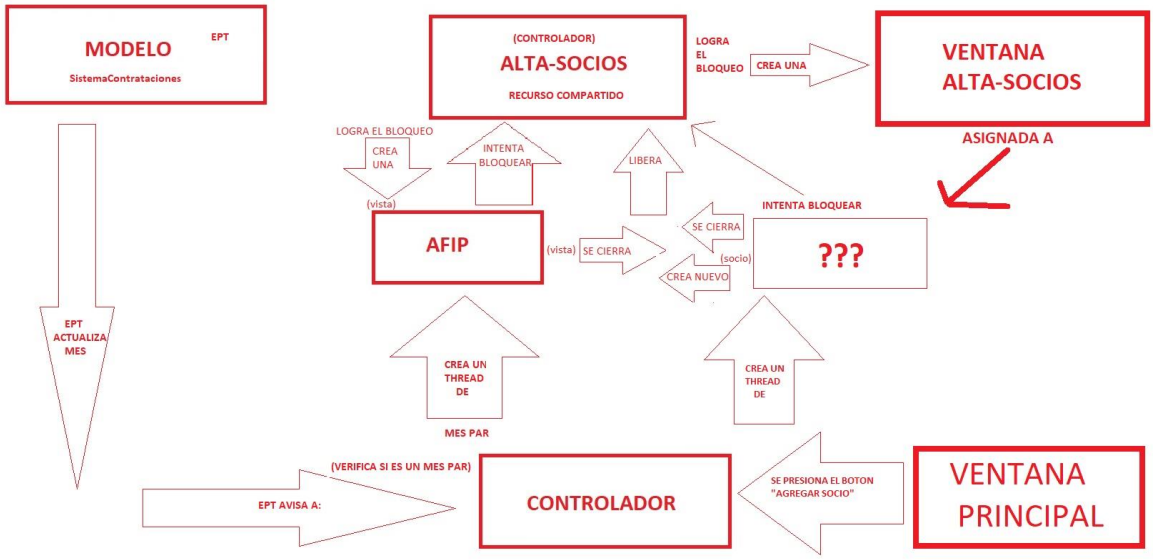
directamente no tienen sets y gets por lo que no se reúnen las condiciones necesarias para persistir con XML.

Entonces decidimos hacerlo con la persistencia binaria que no es tan estricta como la persistencia XML, y de igual manera nos dio un problema que no logramos resolver, al recuperar la persistencia, no se recupera una parte fundamental que es la colección de Facturas que a su vez es una colección genérica. Investigamos en el internet, probamos algunas formas de resolver y no tuvimos una respuesta favorable. Por lo que, al recuperar la persistencia, al realizar cualquier operación que tenga que ver con la colección de facturas saltará error por nullPointerException.

Finalmente pudimos corregir el problema pero no permite trabajar con los datos nuevos, no logramos identificar el problema.

Concurrencia

Esta parte nos fue difícil de pensar. No se nos ocurría qué parte sería el recurso compartido que bloquearía la posibilidad de agregar nuevos socios durante una visita de AFIP y viceversa. Este es un diagrama de una idea temprana que se nos ocurrió para resolver este problema.



Se nos ocurrió entonces crear dos controladores separados para las vistas de la AFIP y del alta de socios que se extendieran de Thread. Además, agregamos un método synchronized dentro del controlador principal que se activaría cuando alguno de estos controladores fuera instanciado, y dependiendo de qué tipo de controlador fuera (si de socios o de la AFIP) crearía la ventana correspondiente y mantendría al Thread “ocupado” hasta que esta se cierre. Decidimos que el AFIP aparezca en los meses PARES.

PRUEBA

Al hacer testeo, observamos que no nos andaban el patrón State, sobre todo en el estado Moroso, y el Gestor de Facturación. No logramos identificar por qué, pero el Gestor de Facturación y el Actualizador de estado (en ese momento, dos clases distintas que observaban al EPT) no compartían la lista de abonados que envía el EPT a sus observadores al avanzar un mes. Pudimos corregir los errores asignando la responsabilidad del actualizador de estado al Gestor de Facturación.

En el momento de clonar las facturas para el AFIP analizamos que no se clonaban las facturas del primer mes, sino que se clonaban recién del segundo mes. Pudimos solucionar este problema.

Al analizar el reporte de las facturas clonadas, observamos que el precio total pagado aplicando descuento y/o recargo era muy alto, se corrigió el error.

OBSERVACIONES

- La persistencia cumple con su función de guardar y recuperar información pero de igual manera, no logramos que trabaje con los datos nuevos.
- La concurrencia anda solo una parte, al momento en el que la AFIP hace la función, bloquea correctamente la ventana Alta Socio. Pero cuando ocurre de forma viceversa no ocurre lo mismo.
- Las vistas al momento de hacer click en las vista para habilitar componentes, no se desbloquean, hay que hacer doble click.