

Trabajo Practico grupal

Programación III

Segunda Entrega

Integrantes:

Barriga, Nahuel

Bailon, Guido

Hernandez, Julieta

Figueras, Gregorio

Trabajo Practico grupal – Programación III

Segunda entrega

Puesto que es la segunda entrega, se procede a explicar las nuevas funcionalidades de nuestra aplicación de sistema de contracciones de servicios de seguridad, dando por entendido que se tiene en cuenta lo realizado en la primera parte del trabajo practico.

Uso de recursos compartidos y threads

Los usuarios pueden solicitar la visita de un técnico. Siendo que el técnico puede estar ocupado al momento de ser solicitado, el usuario entra a una cola de espera hasta adquirir el permiso para recibir la consulta técnica.

Para resolver esta situación, en la cual dos o mas hilos de ejecución no pueden utilizar el recurso compartido en simultaneo, se utilizaron métodos synchronized. Este proceso se realiza en la clase 'TecnicoFactory', donde se aplicaron los métodos synchronized, uno llamado 'ConsultaTecnica', el cual recibe un abonado y, mientras no hayan técnicos disponibles, lo pone en espera hasta que alguno se libere y pueda realizar la consulta. Además, con el uso del patrón **observer-observable**, se mantiene una referencia entre el controlador y el técnico, permitiendo, a través del "notifyObservers" que se lanza, modificar la lista de técnicos disponibles en la ventana principal. El otro método llamado 'TerminaConsulta' efectúa la finalización de la consulta permitiendo al resto de los hilos competir por el recurso (pero solo uno podrá acceder a este).

Para lograr la simulación, la clase Abonado se extiende de Thread e implementa el método 'run()'. Este método invoca a los métodos synchronized del técnico. El llamado entre ellos lleva un tiempo para simular el tiempo que requiere una consulta técnica.

Uso del Patrón State

Debido a que el abonado físico puede registrar diferentes acciones dependiendo del estado en que se encuentre, se utilizo el Patrón State para encapsular su comportamiento.

En la interfaz 'IState' se definen los métodos de las acciones que el abonado puede realizar, estos son: 'PagarFactura', 'ContratarNuevoServicio' y 'BajaDeUnServicio'.

Dentro del paquete 'State', se encuentran las clases de cada estado, estos son: 'SinContratacionesState', 'ConContratacionesState' y 'MorosoState'. Funcionan de modo tal que retornan un string para avisar el cambio de estado o la imposibilidad del mismo, ya que no es trivial el cambio de estos sino que sigue una correspondencia, la cual resulta ser:

Sin contrataciones: Al reclutar un nuevo abonado, comienza con este estado. La única acción que se puede realizar es contratar un nuevo servicio y pasar al estado ConContratacionesState.

Con Contrataciones: Con este estado, el abonado puede realizar las tres acciones posibles, es decir, pagar una factura, contratar un nuevo servicio y dar de baja un servicio. En el caso que el abonado tenga un solo servicio contratado y lo quiera de dar de baja, este pasa al estado SinContratacionesState.

Moroso: Se presenta cuando el abonado no ha pagado dos facturas consecutivas. Lo único que puede realizar es el pago de una factura pero con un recargo del 30%. Cuando haya pagado todas las facturas debidas, pasa al estado 'ConContratacionesState'.

Persistencia

La persistencia se da por codificación XML, siendo esta mas útil debido a que tiene un formato humanamente mas legible en comparación de la persistencia binaria.

Se opto por implementar serializable en todas las clases que necesitábamos serializar, estas son Abonado, Técnico y Sistema. Si bien esta ultima presenta el patrón Singleton, a modo de facilitar el código, no se utilizo un patrón DTO.

La persistencia se presenta al finalizar y al comenzar la jornada, para ello, se debe apretar los botones en la ventana principal 'IniciarJornada' y 'FinalizarJornada'.

Interfaz grafica (patrón MVC)

En esta sección, nos adentraremos en los detalles mas significativos, ya que la clase ventana contiene una extensa cantidad de líneas de código. Dividimos las clases en dos paquetes: el paquete del controlador y el paquete de la vista. En este ultimo se encuentras las clases de la VentanaPrincipal, VentanaAgregaServicio, VentanaAgregarTecnico y VentanaFacturas, las cuatro con referencia al controlador, mientras que el controlador posee referencia hacia el modelo y a las ventana.

En las clases del paquete vista se utilizaron, entre otros, los métodos: 'keyReleased()', para la obtención de los valores de los campos de texto y para la habilitación o des habilitación de componentes, el método 'mouseReleased()' para realizar una serie de acciones cuando se suelta el botón del mouse después de haberlo presionado y el método 'setActionListener()' para la asociación del ActionListener a los botones de la ventana.

El controlador, por su parte, implementa ActionListener, y allí desembocan todos los eventos que genera presionar los botones de la ventana. El controlador busca los datos, y los vuelca en la ventana.

Breve explicación de cómo funciona el programa:

Cuando ingresamos, lo primero que se vera será la vista de la ventanaPrincipal. Para comenzar, se debe precionar el boton 'IniciarJornada' y podremos crear un Abonado (es necesario ingresar tanto el nombre como el DNI para que el sistema permita definir el tipo de abonado y la forma de pago) y crear tecnicos. Para este ultimo, se abra una ventana para incruir los datos del mismo.

Una vez creado el abonado, se puede contratar un servicio. Para ello, se abrirá la vista de VentanaAgregaServicio. Aquí se especifica el tipo de servicio, qué incluirá y, en el caso que se desee, qué promociones tendra. Una vez agregado el servicio, se mostrara un mensaje indicando que el servicio fue agregado correctamente.

Siendo que un mismo abonado puede adquirir servicios para distintos domicilios, si se desea obtener la informacion del servicio de alguno de los domicilios o pagar la factura de estos, se debe precionar el abonado y en la lista de domicilios, el domicilio. Ademas, si se desea obtener una lista de las facturas de los servicios contratados por el abonado, se debe presionar el botón 'Mostrar facturas'.

Para solicitar la atencion de un tecnico, se debe precionar el abonado que requiere de la atencion y precionar el boton 'Solicitar'. Si no hay tecnicos disponibles, no se podra solicitarlos, en cambio, si los hubiera, se motrara un mensaje indicando que el abonado esta siendo atendido.

Cuando se preciona el boton 'IniciarJornada', los datos se despersistiran. Para finalizar la jornada, se debe precionar el boton 'FinalizarJornada' y los datos se persistiran en un archivo xml.

Por ultimo, para avanzar de mes, se debe precionar el boton 'SimularMes'. En consecuencia, se revisara el estado de cada abonado fisico para determinar si debe cambiar de estado.

Imágenes de las ventanas:

- Vista Principal

The screenshot shows the main application window with the following sections:

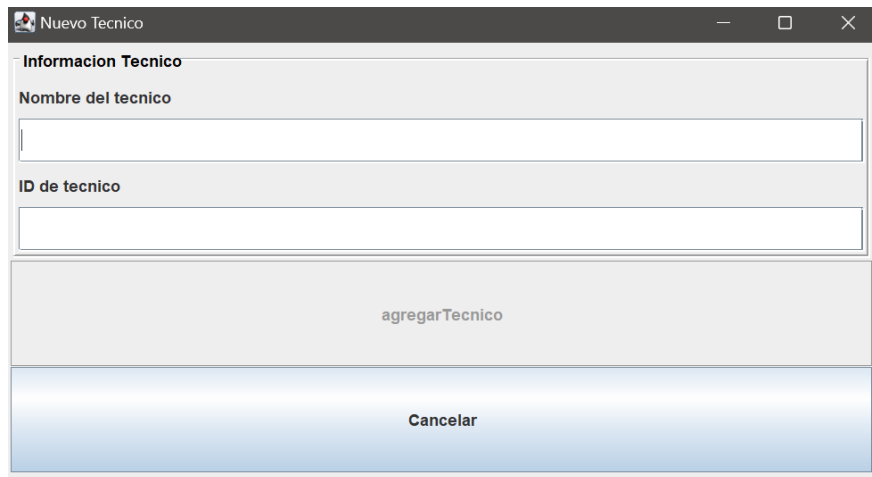
- Abonado:** A large empty text area for listing subscribers.
- Acciones de abonado:** A panel containing three buttons: "Paga servicio", "Contrata nuevo servicio", and "Dar de baja un servicio".
- Crear Abonado:** A form with input fields for "Nombre:" and "DNI:", radio buttons for "Juridico" and "Fisico", and radio buttons for "Cheque", "Efectivo", and "Credito". It also includes "Eliminar" and "Agregar" buttons.
- Acciones de tecnico:** A panel with "Solicitar", "Eliminar", and "Agregar" buttons.
- Domicilio:** A large empty text area for address information.
- Servicio:** A large empty text area for service details.
- Footer:** A row of buttons: "Mostrar facturas", "Inicia jornada", "Simular mes", and "Finaliza jornada".

En un ejemplo:

This screenshot shows the application window with sample data entered:

- Abonado:** A list containing two entries: "abonado1 - 43456789" and "abonado1 - 43456789".
- Acciones de abonado:** Same as the previous screenshot.
- Crear Abonado:** "Nombre:" is filled with "abonado1" and "DNI:" is filled with "43456789". "Juridico" is selected with a radio button. "Cheque" is selected under the payment method options.
- Acciones de tecnico:** Same as the previous screenshot.
- Domicilio:** Filled with "Maipu 4815".
- Servicio:** Filled with details: "Vivienda:", "Precio base: \$8500.0", "Cantidad de camaras:3", "Cantidad de botones antipanico:0", "Movil de acompanamiento:false", "Id del servicio=2", and "Promo:Plateada".
- Footer:** Includes a status bar with the text "Se inicio la jornada" and "Servicio agregado correctamenteServicio agregado correctamente", and the same row of action buttons as before.

- Vista agregar nuevo tecnico



The image shows a software window titled "Nuevo Tecnico". It contains two input fields: "Nombre del tecnico" and "ID de tecnico". Below these fields are two buttons: "agregarTecnico" and "Cancelar". The window has a standard Windows-style title bar with minimize, maximize, and close buttons.

Informacion Tecnico	
Nombre del tecnico	<input type="text"/>
ID de tecnico	<input type="text"/>
<input type="button" value="agregarTecnico"/>	
<input type="button" value="Cancelar"/>	

- Vista mostrar facturas

Conclusión

En conclusión, y como cierre de este trabajo práctico, la aplicación de los patrones vistos durante la segunda mitad del cuatrimestre, sumados a lo visto en la primera parte, nos permitieron entender a fondo como funciona un sistema que podríamos llegar a utilizar o desarrollar en la vida profesional.

Además, destacamos una vez más lo fructífero, en términos de aprendizaje, que es esta metodología de proyecto en relación a un examen donde haya que desarrollar un código en tiempo límite.