Trabajo: Implementación de una LPS Orientada Aspectos

Realizado por: Vivian Mendoza

Descripción: Durante este trabajo de realizó un análisis de una LPS implementando técnicas de variabilidad basada en Aspectos para el caso de estudio de un cajero automático. Para lo cual fue necesario definir los puntos de corte pertinentes, para intervenir los métodos que fueran necesarios, los puntos de intervención para la manipulación de argumentos, excepciones u otros métodos de una clase y diseñar cuidadosamente el código fuente que específica los cambios sobre el código original para lograr la característica variable del producto, de acuerdo a un modelo de características obtenido del enunciado del proyecto. A continuación, se encuentran los modelos diseñados y demás información sobre la técnica de variabilidad implementada

Modelo de Características:

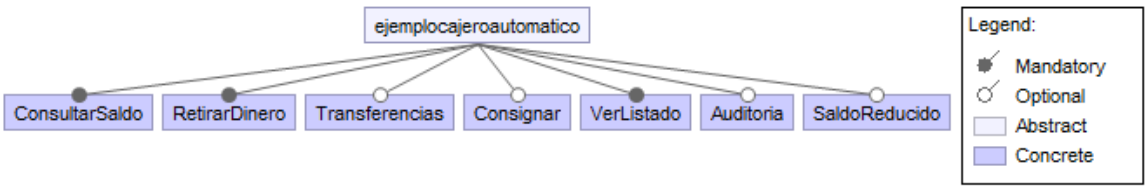
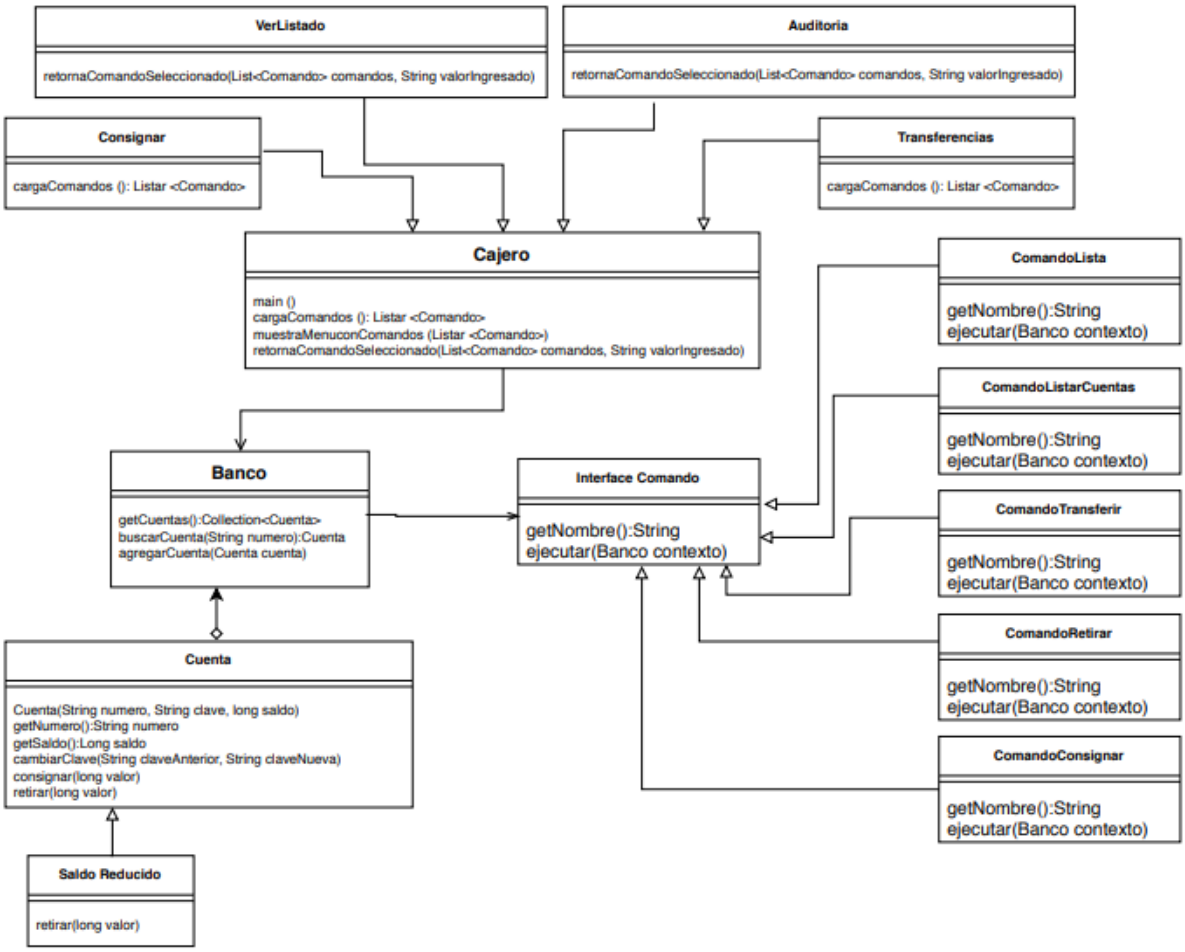


Diagrama de Clases:



Descripción de los Aspectos que implementan las caracteísticas:

1. **Característica Opcional Consignar:**

Pointcut: `call ( * ejemplo.cajero.Cajero.cargaComandos(..));`

thisJoinPoint utilizados: Ninguno, se intervino el punto de corte luego del resultado arrojado por el método original

Descripción del código: Se utiliza un advice luego de que el método en el punto de corte guarde y retorne las opciones a mostrar en el menú, habilitando o escondiendo la opción de comando para consignaciones en el cajero, por medio del archivo de configuración

2. **Caracteristica Opcional Transferencia**

Pointcut: `call ( * ejemplo.cajero.Cajero.cargaComandos(..));`

thisJoinPoint utilizados: Ninguno, se intervino el punto de corte luego del resultado arrojado por el método original

Descripción del código: Se utiliza un advice luego de que el método en el punto de corte guarde y retorne las opciones a mostrar en el menú, habilitando o escondiendo la opción de comando para transferencias en el cajero, por medio del archivo de configuración

3. Caracteristica Opcional SaldoReducido

Pointcut:

```
call(*ejemplo.cajero.modelo.Cuenta.retirar(..)throws *..*Exception*);
```

thisJoinPoint utilizados: Se manipularon el método getSaldo y los argumentos del método retirar, controlando también las excepciones

Descripción del código: Se modifica el método para el retiro de acuerdo al saldo actual en la cuenta, de manera que si la característica de Saldo Reducido se encuentra activa no se permite un retiro de una cuenta mientras su saldo sea menor a 200000, realizando un manejo de la excepción del método. Sin embargo, al estar inactiva esta característica opcional, los retiros de cualquier monto seguirán siendo posibles para las cuentas mientras el valor a retirar sea positivo y mayor al saldo de actual de la cuenta.

4. Caracteristica Obligatoria Ver Listado

Pointcut:

```
call(*ejemplo.cajero.Cajero.retornaComandoSeleccionado(..));
```

thisJoinPoint utilizados: Manipulación de los atributos del método que retorna la opción de operación escogida por los usuarios.

Descripción del código: Se utiliza el método existente para retornar el comando seleccionado y se guarda cada una de las operaciones ejecutadas por un usuario en un archivo, este se lee y escribe (actualiza) luego de cada selección de operación de un usuario; el listado de las operaciones escogidas por los usuarios en el transcurso del día se puede ver únicamente luego de las 23 horas.

5. Caracteristica Obligatoria Auditoría

Pointcut:

```
call(*ejemplo.cajero.Cajero.retornaComandoSeleccionado(..));
```

thisJoinPoint utilizados: Manipulación de los atributos del método que retorna la opción de operación escogida por los usuarios.

Descripción del código: Se utiliza el método existente para retornar el comando seleccionado y se guarda cada una de las operaciones ejecutadas por un usuario en un archivo, este se lee y escribe (actualiza) luego de cada selección de operación de un usuario; el listado de las operaciones escogidas por los usuarios es almacenada en un archivo o log de auditoría con información de fecha de la operación y tipo de operación realizada pro los diferentes usuarios.s

Video demostración de la Línea de productos:
https://github.com/vivimendoza16/Cajero-Automatico

Acceso al repositorio con el Código Fuente:
https://github.com/vivimendoza16/Cajero-Automatico

Conclusiones y Lecciones Aprendidas:

Para un caso de estudio real se logró identificar los puntos de corte necesarios para intervenir el código utilizando implementación de la variabilidad con técnicas basadas en Aspectos. También se comprendió la manera de manipular los puntos de intervención para obtener y manipular no solo los métodos de una clase sino también sus argumentos e incluso el manejo de excepciones.

Se encontró que con AspectJS no se requieren archivos de configuración adicionales, sino que era suficiente con realizar los cambios sobre la configuración del modelo de características, además los cambios se implementan de una manera mucho más eficiente y modular.

A diferencia de la implementación de variabilidad basada en programación orientada a objetos, no se requieren interfaces o clases adicionales.

En el caso de AspectJs el lenguaje de dominio y su sintaxis es sencillo de manipular, a pesar de que no se cuenten con mucha experiencia en desarrollo.