Amazon

Laboratorio 3

Axel Llobet
Luciano Amarilla Joaquin

Desarrollo

Para comenzar establecimos la creación de las calases que harían de base principal para el funcionamiento de del sistema. Con esto en mente decidimos que seria conveniente crear dos tipos de usuarios para poder acceder a distintas funciones del sistema, por esto creamos las clases Business y Client. Ya que teníamos variables en común, creamos una superclase abstracta que contuviera estas, también creamos la clase Product para completar la base del programa. Mientras avanzábamos nos dimos cuenta que sería útil tener administradores que realicen ciertas funciones dentro del programa, ya que este seria una cuenta y que a diferencia de las otras no tenía importancia ni sentido que tuviera todas las variables que las anteriores compartían, decidimos hacer una superclase que constara de las variables que Admin requería y que las otras cuentas también compartían. Ya que en un futuro se podrían agregar distintos tipos de administradores con nuevas variables o cumplan distintas funciones, por lo cual lo establecimos de forma tal que se puedan generar nuevas clases de ser necesario.

Resumen

La aplicación permite crear dos tipos de cuentas, una personal que permite realizar compras y otra empresarial que permite publicar productos para su comercialización. También existen cuentas administrador, pero son creadas únicamente por administradores para que el acceso sea restringido a los empleados. Los clientes pueden realizar compras, ver un registro detallado de estas. Las empresas podrán administrar la información de sus productos, publicar nuevos, ver registros de ventas, aumentar los precios de todos sus productos de manera porcentual y aumentar la cantidad de existencias de los productos. Los administradores podrán banear cuentas, ver la lista de las cuentas y productos registrados, persistir el sistema y realizar una limpieza del mismo.

Informe Técnico

El sistema se maneja a través de la clase Menu que nos da acceso a las distintas funciones del sistema, recibe una instancia de la clase Amazon que es la que contiene los registros de las cuentas en sus diferentes instancias y los productos.

Los registros de las cuentas se almacenan en un HashSet ya que necesitamos una colección que no permita duplicados y no nos interesa el orden en el que se almacenan, a diferencia de los productos que se almacenan en un LinkedHashSet ya que no queremos permitir que existan duplicados.

En el caso de las cuentas definimos en el equals la restricción de que para ser iguales deben tener el mismo Username, para evitar la creación de cuentas con el mismo nombre, ya que este es atributo que utilizamos para el igreso al sistema y de existir un duplicado generaría un conflicto al momento de ingresar.

En el caso de los productos definimos en el equals la restricción de que para ser iguales dos objetos deben tener el mismo idBusiness y el userName, esto lo definimos así porque pueden existir 2 productos iguales pero que sean vendidos por cuentas diferentes, lo que puede y probablemente pase. Pero con estas restricciones permitimos que esto ocurra y al mismo tiempo limitamos a las cuentas Business la publicación de 2 productos iguales.

Dentro de las cuentas tenemos tres instancias diferentes:

Los Client tienen un Arraylist de ShoppingRecord, estos son una clase que guarda un ArrayList de productos y dos variables de tipo Instant ya que esta clase almacena las fechas en el uso horario cero y ya que trabajamos con varios países, esto nos permite convertir las fechas almacenadas al uso horario del lugar donde se esté ejecutando el programa. El registro de compras debe poder repetirse y tener un orden, por lo cual se almacenan las nuevas entradas siempre en el primer lugar de la colección.

Los Business tienes un HashMap con un Instant en la Key y un Product en el Value, para almacenar las ventas que se realicen, no importa el orden, ya que al guardarse con fechas se puede utilizar esto como filtro para sacar la información útil como total mensual, anual, cantidad de X productos vendidos en X mes/meses, por desgracia no alcanzamos a aplicar esta función.

Los Admin no cuentan con ninguna variable ni método, ya que sirve para diferenciarse del resto de las cuentas y permite el acceso a varios métodos de la clase Amazon. Esto se pensó de esta manera teniendo en cuenta la escalabilidad a futuro del sistema, en el cual en un futuro podríamos tener que crear nuevas variantes de la clase Admin que tengan distintos niveles de acceso por ejemplo y esta manera nos facilitaría el trabajo futuro.

Información Útil

Se cargaron en el sistema varias cuentas para poder observar el funcionamiento de algunas funciones de la aplicación, cuya información es la siguiente:

Tipo de Cuenta	Nombre	Contraseña
Administrador	axel	123
Empresa	fede	123
Empresa	juan	123
Empresa	maria	123
Cliente	carla	123
Cliente	roberto	123

Conclusiones

Para concluir cabe aclarar que debido a problemas personales, que escapaban a nuestro control, nos vimos con poco tiempo para generar un código más limpio y corregir errores de diseño que surgieron en el afán de completar la tarea y que a pesar de reconocerlos nos implicaba un tiempo de trabajo del que ya no disponíamos, como por ejemplo la verificación del nombre de usuario, la

cual se analiza a través de un método en la clase Amazon y que podría simplemente realizarse utilizando la lógica de la colección HashSet. Nos vimos obligados a dejar algunas ideas como la implementación del tiempo de envió que variaba entre países a medio terminar ya que no realiza un análisis más complejo como se planteó originalmente y se terminó generalizando. También tenemos pleno conocimiento del gran riesgo que conlleva utilizar las posiciones dentro de los arreglos y que es poco recomendable pero debido a que nos limitaba el tiempo no pudimos implementar una interfaz gráfica donde podríamos haberlo remplazado por ejemplo con botones para generar otras correlaciones que implican menos riesgos de errores en tiempo de ejecución. Y nos vimos obligados a generar una base de datos de prueba muy simple e insuficiente para demostrar todas las funciones que implementamos.