**Diseño y programación orientada a objetos**

**Proyecto 3: Diseño**

**Daniel Esteban Aguilera Figueroa** 202010592

**José Vicente Rincón Celis** 202013321

En primer lugar, para definir el documento de diseño y sus partes lo haremos de tal forma que habrá un sistema de niveles en el cual se irán mirando y procesando los factores vistos en cada uno de los procesamientos de forma progresiva. Teniendo esto cuenta podemos empezar con el nivel 1. En el .zip estará archivado el UML que usamos para darle forma a nuestro modelo de diseño.

Primero debemos identificar los requerimientos del supermercado, en este caso los funcionales y no funcionales:

**Nivel 1: Contexto del problema:**

El problema de este proyecto radica en que hay que implementar nuevos requerimientos a dos programas con interfaces que ya son funcionales, es decir que al momento de hacer esta actualización debemos tener cuidado en que el programa funcione correctamente, hacer las modificaciones necesarias en cuanto a las relaciones de todos los objetos del programa y una actualizacion en las interfaces graficas para que se puedan usar los nuevos requerimientos. Cabe resaltar que el proceso en cuanto al uso de los programas sigue siendo el mismo (se adjuntan las siguientes imágenes tomadas del documento de diseño del proyecto 2).

Diagrama

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

En ambos procesos podemos observar cómo es el usuario siempre es el que inicia el proceso, en el primer proceso hablamos de un proceso cerrado por lo cual la información que retorne no será comunicada por el cliente, mientras que en el segundo proceso toda la información retornada tiene que ser comunicada al cliente por medio del usuario.

Por otro lado, tenemos la relacion de los dos programas:

Gráfico, Gráfico de burbujas

Descripción generada automáticamente

Como se puede ver supermercado abarca todos los procesos, y los dos programas que nos piden que son sistema P.O.S e inventario, estos dos se relacionan y en esa relación se encuentra el proceso de venta de un artículo es decir , el pasar un artículo por la caja registradora conlleva a la creación de un recibo si no existe, y si existe se agrega a la factura, después hay que decirle al inventario que las unidades disponibles se redujeron y que si no queda más producto en las estanterías llenarlos de nuevo y esto conlleva a una constante actualización del inventario.

Para empezar con el proceso de diseño se definen las funcionalidades de alto nivel que son requeridas para la actualizacion de nuestro programa:

* Es necesario agregar el proceso en el cual el cliente pueda hacer uso de la cantidad de puntos que desee en su factura (siempre y cuando dicha cantidad no supere los puntos disponibles del cliente).
* En la generación de la factura tiene que haber una sección con toda la información relacionada a los puntos, es decir , el total de puntos acumulados antes de esa compra, el total de puntos redimidos en esa compra, el total de puntos acumulados por esa compra y el total de puntos disponibles al final de la compra.
* Actualización de la interfaz del sistema P.O.S de tal manera que se pueda mostrar el desempeño que ha tenido un producto en cierto intervalo de tiempo elegido por el trabajador.
* El supermercado debe soportar los 4 tipo de promociones (descuento,regalos,combos,puntos multiplicados) y deben quedar registradas en la factura si alguna es valida para aplicarse.

Restricciones:

* Los puntos siempre tienen que ser valores enteros.
* Por cada mil pesos comprados se le otorgara al cliente un punto.
* Cada punto redimido equivale a 15 pesos.
* La cantidad de puntos redimidos no puede superar a la cantidad disponible de puntos antes de la compra.
* Todas las promociones validas según la fecha y cargadas en el archivo P.O.S siempre seran tenidas en cuenta.
* Si se crea un archivo nuevo para promociones mientras el sistema P.O.S esta corriendo, el programa debera reiniciarse para tener en cuenta las promociones.

**Nivel 2: Cambios interfaz**

Teniendo en cuenta los anteriores requerimientos podemos hacer las modificaciones necesarias en la interfaz que ya tenemos se adapte a los que necesitamos:

A picture containing table

Description automatically generated

Se agrego un botón extra a la interfaz de inventario el cual nos va a permitir mostrarle al usuario la grafica del comportamiento del desempeño de un producto en lapso dado. El cual saldrá en una ventana emergente.

A continuación, mostraremos un diagrama de flujo del funcionamiento de las compras con el sistema POS actualizado. Este nos permitirá preguntarle al usuario si quiere redimir puntos o acumular puntos.

Diagrama

Descripción generada automáticamente

Adicionalmente, deberemos tener en cuenta el sistema de promociones al final de una compra, el cual estará ahí presente y se aplicara sobre los productos que se tienen en el carrito.

Por último, dentro de este nivel tendremos el inicio de planeación de posibles errores. Teniendo en cuenta que tenemos muchos JTextFields y demás componentes en las interfaces, estos tienden a entregar strings vacíos cuando no se escribe nada sobre ellos o cuando no hay selecciones validas, esto podría causar algún error, por ende, es un factor para tener en cuenta. Por otro lado, es posible que buscando algún lote en especifico no nos permita continuar ya que este podría ser null indicándonos que no existe.

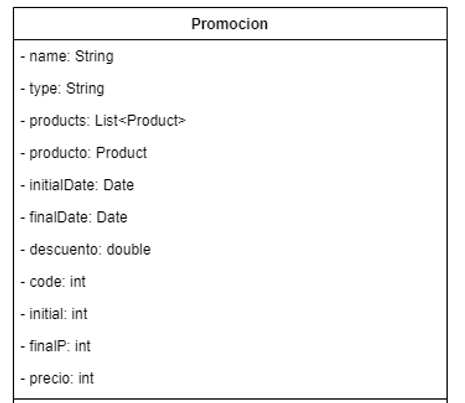
**Nivel 3: Aclaraciones**

Explicacion del recuadro rojo:

* En la primera imagen en donde esta el recuadro rojo habria un boton con la finalidad de poder ver el desempeño del producto después de haber buscado por el código de barras, además tendrá que seleccionar la fecha de inicio y final para poder realizar la gráfica.
* En cuanto planeación queremos tener en cuenta que tendremos JOptionPane para poder realizar este requerimiento, el cual saldrá cuando se quiera preguntar por la fecha de inicio y la fecha de fin.

**Diagrama del sistema P.O.S**

El diagrama de flujo simplificaod del sistema P.O.S nos da una vision general del proceso y esto es importante debido a que en este proyecto solo se afectara el proceso de factura donde tendremos que implementar los descuentos y volver valido el sistema de puntos para que sea redimido por dinero al momento de pagar. Adicionalmente, las promociones serán una clase aparte la cual contendrá los 4 tipos de promociones y siempre estarán aplicadas al final de una compra. El ejemplo de la clase será algo del estilo:



Donde tendremos diferentes atributos los cuales pueden pertenecer a cualquier de las 4 promociones. Tenemos entonces que, las promociones cuentan con nombre, tipo, producto, productos, fecha inicial, fecha final, descuento, código, valor inicial, valor final y el precio.

En cuanto a la gráfica, esta aparecerá luego de digitar el código y elegir las fechas. Tendremos un sistema de persistencia aparte para poder generar las gráficas.

En adición al control de errores, se va a tener en cuenta la creación de un método el cual nos permitirá manejar los errores dentro de la interfaz ya que esta se encarga del manejo de estos de forma directa. Usaremos el método BULK para indicar al usuario el problema y que este se encargue de solucionar el problema, ya que generalmente los errores con este programa no hacen parte del código base si no de selecciones invalidas por parte del usuario.

**Nivel 4: Reflexiones**

UML en la carpeta Data.

**Reflexiones:**

* Pensando en las ventajas que presenta nuestro diseño podemos destacar que es fácil de implementar ya que están todas las clases necesitadas presentes y pudimos llegar a la última fase del diseño por iteración, por ende, fue posible realizar un UML el cual nos ayudara más adelante. Adicionalmente, todo esta explicado de forma bastante organizada y nos permite comprender qué papel juega cada objeto en la ventana desplegada para uso del usuario.

Por último, en este nivel tenemos ya los métodos para el manejo de errores, los cuales son:

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Estos se encuentran en ambas interfaces y permiten una indicación pertinente al usuario. Es de esta forma que tendremos en cuenta los errores en nuestro programa.

Aclaraciones a tener en cuenta. También se encuentran aclaraciones del proyecto anterior en caso de que necesiten ayuda para guiarse con el programa.

**Aclaraciones:**

* El programa está dividido en dos interfaces las cuales se encuentran en dos paquetes distintos para delimitar su uso y mostrar que se usan de forma separada. Estas interfaces son las clases “InventoryWindow.java” en el paquete de graphicInterfaceInventory y la otra es “CajeroWindow.java” en el paquete graphicInterface. Por lo tanto, tenemos 2 main en nuestro proyecto. Esto se preguntó en una sesión de cupitaller y dijeron que mientras se informara en el documento como se usaba cada interfaz no habría problema.
* Para la carga no es necesario escribir el nombre completo del archivo junto a su extensión. Con solamente escribir el nombre es posible cargar.
* Los archivos disponibles para el uso de carga son:
  1. datafile
  2. datafile11
* Se corrigió un error presente en el proyecto 1 el cual no permitía continuar con el sistema de inventario luego los lotes vencidos de un producto.
* Se implemento el sistema de consistencia de datos lo cual permite que sea usado el proyecto de forma constante sin ningún problema y perder algún tipo de dato. Todo fue implementado usando el formato .csv. Este sistema es presente para el inventario y el sistema de caja guardando datos de productos, lotes e información del cliente.
* El requerimiento de las imágenes de cada producto se muestra al dar al botón consultar, ya que es mejor tener una factura limpia donde solamente se muestre la cantidad, el nombre del producto y el precio. Por otro lado, al consultar los productos que se encuentran en la compra, es mucho mejor tener una idea visual.
* Las imágenes de los productos deben ser 225x225 ya que de esta forma no se provoca un resize de los componentes de la interfaz.
* Se implemento el sistema de persistencia de datos de forma definitiva teniendo el archivo datafileff.csv como base de datos. Se entregará vacío a la hora de la entrega del proyecto para demostrar que este se va llenando a medida que se van cargando archivos en el inventario. Adicionalmente, se guardan todos los datos en este archivo al cerrar la ventana de inventario. Cuando se cierra el sistema de caja este se actualiza teniendo en cuenta las ventas realizadas.
* La base de datos para los clientes es clientData.csv, allí se guardan todos los datos de estos y se actualizan cada vez que inician.
* Se realiza un resize debido al layout cuando se digita un archivo y no se encuentra, para arreglar el resize es necesario escribir el nombre del archivo de forma correcta.
* A lo largo del procedimiento se hacen uso de DefaultListModel para realizar los Joptionpane y así poder trabajar de forma cómoda.
* Se dejaron los métodos usados en las interfaces de consola, la entrega se realiza de un proyecto conteniendo el proyecto 1 y 2. Ambos métodos se encuentran. Para poder usarlo es simplemente correr InventoryWindow.java o cajeroWindow.java, lo demás se puede ver. El procesamiento contiene todos los procedimientos, los paquetes graphicInterface y graphicInterfaceInventory la interfaz gráfica.
* La interfaz de caja no funciona si no se carga algo anteriormente en la interfaz de vocabulario.
* Las imágenes de los productos deben ser png y deben estar guardadas en la carpeta. ./data/productImg.
* El botón lista productos en la interfaz de caja permite visualizar las imágenes que tiene cada producto.
* Repito, para usar la caja se debe cargar el inventario.
* Los descuentos se aplican automáticamente, solo basta con que revisen los documentos para así tener una mejor idea y saber que productos tienen descuento y así poder ver su funcionamiento.
* Para graficar el desempeño simplemente basta con poner el código, elegir las fechas las cuales salen en JOptionPane y así graficar. No todos los productos se pueden graficar. Gracias.