Facultad de Cs. de la Computación

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

UNIDAD ACADÉMICA

PROGRAMACIÓN II

ASIGNATURA

MANUAL TÉCNICO DEL PROYECTO FINAL

DOCUMENTO

INGENIERÍA EN CIENCIAS DE LA COMPUTACIÓN

PROGRAMA ACADÉMICO

OSWALDO GUERRERO RIVERA, MARTÍN CORONA JIMÉNEZ, DANTE CASTELÁN CARPINTEYRO

AUTORES

Manual de técnico

El presente manual técnico tiene como objetivo proporcionar la guía necesaria para que los desarrolladores y administradores del sistema sean capaces de entender, mantener y mejorar el software de forma eficiente. Este manual incluye detalles técnicos sobre la arquitectura del sistema, la estructura del código, las dependencias y las mejores prácticas para su uso y desarrollo. Para que el software descrito en este manual funcione correctamente, es necesario comprender las funciones, clases, métodos y estructura técnica del código.

Asimismo, el código en permanente actualización se puede descargar o clonar directamente desde el repositorio del proyecto: https://github.com/LTNGx39/Market, el cual está versionado con Git y alojado en GitHub.

Características principales

El presente producto será una valiosa herramienta para la administración de su negocio de ventas al mayoreo bajo el modelo de compra de membresía. Cuenta con características que le ofrecerán una ventaja competitiva frente a cadenas similares por su fluida ejecución gracias a un desarrollo optimizado.

Las características principales, son:

1. Gestión del inventario de productos: Implementación de clases y métodos para la creación, edición, eliminación y consulta de ítems. Uso de estructuras de datos eficientes para el manjeo del inventario.

2. Almacenamiento seguro y eficiente de los datos en archivos: Lectura y escritura de datos en archvios CSV utilizando BuffeferedReader y BufferedWriter. Manejo de rutas de archivos y directorios mediante java.nnio.file.Path.

3. Interfaz de usuario con un diseño amigable y minimalista.

4. Recomendaciones de compra de productos con baja existencia.

5. Validación de datos y manejo de errores personalizado, para dar soporte técnico eficiente en caso de errores.

Operación y código

Planteamiento del diagrama de clases

Para organizar óptimamente las clases, se planteó el siguiente diagrama de clases con las relaciones también indicadas:

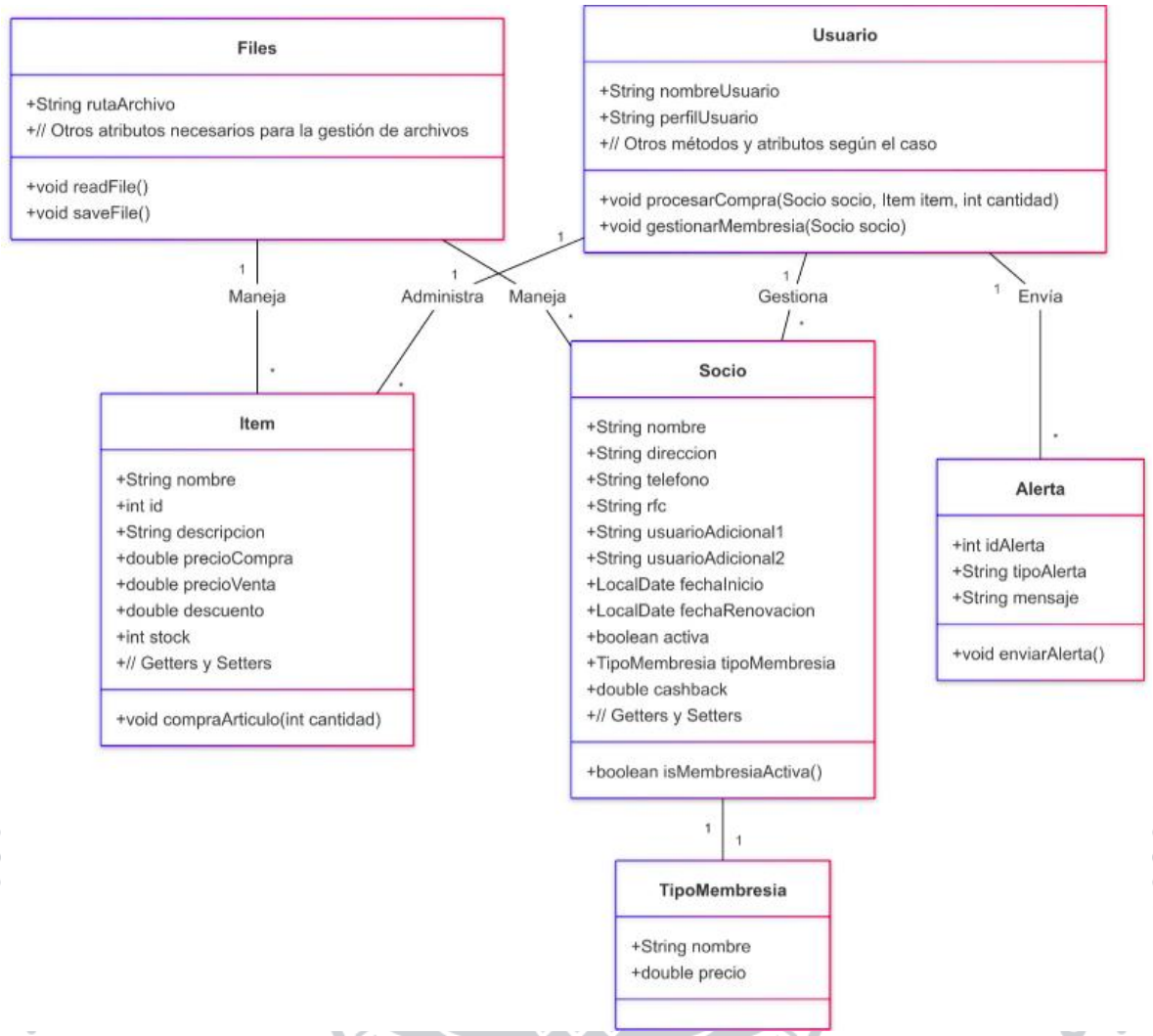

Figure 1: Vista del diagrama de clases empleado para la organizacióny planteamiento iniciales.

Interfaz y código

1. Para generar una interfaz sencilla y útil para el usuario, se presentan los detalles y descripción de componentes importantes.

Main

Informe Técnico del Archivo Java: Main.java

Clase Principal: Main

La clase Main actúa como el punto de entrada para la ejecución del programa. Contiene un único método principal, main, que es el encargado de inicializar la interfaz gráfica del usuario (GUI) a través de la creación de una instancia de MainFrame.

Métodos Principales

1. Método main
   - Descripción: Método estático que se ejecuta al iniciar el programa.
   - Parámetros:
     - String[] args: Argumentos de línea de comandos (sin uso en este caso).
   - Funcionalidad:
     1. Instancia la clase MainFrame: Se crea un objeto de la clase MainFrame, con dos argumentos que indican las dimensiones iniciales de la ventana (800 de ancho y 600 de alto).
     2. Hace visible la ventana: Llama al método setVisible(true) en el objeto MainFrame, lo que asegura que la ventana gráfica sea mostrada al usuario.

Variables Globales

Esta clase no define variables globales propias. La única instancia relevante (MainFrame) se maneja de forma local dentro del método main.

Dependencias Externas

- Paquete Importado: UI:
  - La clase depende de un paquete externo llamado UI. Este paquete contiene la clase MainFrame, que se utiliza para gestionar la interfaz gráfica.
  - No se proporciona más información sobre la implementación de MainFrame en este archivo.

Notas Finales

- Finalidad del Archivo: Este archivo tiene como objetivo iniciar la aplicación y lanzar la ventana principal de la interfaz gráfica.
- Bloques de Código Relevantes:
  - No existen bloques complejos, como bucles o estructuras try-catch, ya que el archivo se centra únicamente en la inicialización básica de la aplicación.

Este diseño sugiere una arquitectura modular, donde la responsabilidad de configurar y manejar la interfaz gráfica recae en la clase MainFrame, que debe estar definida en el paquete UI.

MainFrame

Informe Técnico del Archivo Java: MainFrame.java

Clase Principal: MainFrame

La clase MainFrame extiende de javax.swing.JFrame y se utiliza para crear una ventana principal que sirve como contenedor para diferentes componentes de la interfaz gráfica de la aplicación. Incorpora elementos como una barra de título personalizada (TitleBar), un sistema de cambio de paneles (CardLayout) y diversos paneles funcionales (usuarios, administración, ventas y membresías).

Métodos Principales

1. Constructor MainFrame(int width, int height)
   - Descripción: Inicializa la ventana principal con las dimensiones especificadas y configura su contenido.
   - Parámetros:
     - int width: Ancho de la ventana en píxeles.
     - int height: Alto de la ventana en píxeles.
   - Funcionalidad Principal:
     1 Configuración de la Ventana:
       - Ajusta el tamaño (setSize) y posiciona la ventana en el centro de la pantalla (setLocationRelativeTo(null)).
       - Quita la barra de título estándar con setUndecorated(true).
       - Configura un fondo transparente mediante setBackground(Palette.ALPHA_0).
     2 Personalización:
       - Se asigna un panel personalizado (CustomPane) como contenedor principal.
       - Se inicializan componentes personalizados como TitleBar, UserSelector, Admin, Sales y Members.
     3 Cambio de Paneles:
       - Se crea un panel principal (panelChanger) con un diseño CardLayout que permite alternar entre diferentes vistas.
       - Cada vista (usuarios, administración, ventas, membresías) se registra con un identificador.

2. Métodos de Acceso
   - getCardLayout() y getPanelChanger():
     - Devuelven el sistema de diseño y el panel que permite el cambio dinámico de vistas.
   - Otros Métodos Getters:
     - Permiten acceder a los componentes clave de la ventana, como TitleBar, UserSelector, Admin, Sales y Members.
   - getUsableWidth() y getUsableHeight():
     - Calculan el espacio utilizable dentro de la ventana, descontando los bordes externos.

Clase Interna: CustomPane
La clase CustomPane extiende de javax.swing.JPanel y se utiliza para personalizar la apariencia de la ventana principal.

1. Constructor CustomPane(MainFrame mainFrame)
   - Descripción: Define un panel personalizado con bordes y un diseño de disposición vertical.
   - Parámetros:
     - MainFrame mainFrame: Referencia a la instancia principal de la ventana.

2. Método Sobrescrito: paintComponent(Graphics g)
   - Función:
     - Dibuja un fondo con bordes redondeados.
     - Usa colores definidos en Palette para el borde y el área interna.
     - Activa suavizado de bordes con RenderingHints.

Variables Globales Principales

1 TitleBar titleBar:
   - Una barra de título personalizada que reemplaza la barra estándar del sistema.

2 CardLayout cardLayout y JPanel panelChanger:
   - Administran y contienen las diferentes vistas de la interfaz.

3 Paneles Específicos (UserSelector, Admin, Sales, Members):
   - Representan las diferentes secciones funcionales de la aplicación.

4 Colores y Estilos (Palette):

- Proporcionan constantes para la configuración visual (por ejemplo, colores de fondo y transparencia).

Bloques de Código Relevantes

1 Configuración del CardLayout:

- Permite alternar entre vistas mediante identificadores únicos como "Usuarios", "Administracion", "Ventas", y "Membresias".

2 Método paintComponent en CustomPane:

- Implementa un diseño gráfico atractivo con bordes redondeados y suavizado, lo que mejora la apariencia visual de la ventana.

Dependencias Externas

- Paquete UI.Assets:
  - Es probable que contenga recursos gráficos o constantes como Palette.
- Paquete UI.Panel:
  - Contiene las clases que definen los paneles de cada vista (UserSelector, Admin, etc.).
- Librerías de Swing y AWT:
  - Utilizadas para crear y personalizar la interfaz gráfica.

Notas Finales

La clase MainFrame está diseñada como un contenedor modular y extensible para una aplicación gráfica. Su uso de CardLayout permite una navegación eficiente entre las diferentes secciones, mientras que la personalización de los componentes garantiza una apariencia uniforme y profesional.

TitleBar

Informe Técnico del Archivo Java: TitleBar.java

Clase Principal: TitleBar

La clase TitleBar extiende de javax.swing.JPanel y actúa como una barra de título personalizada para la ventana principal de la aplicación. Incluye botones de navegación y cierre, un título dinámico y funcionalidades adicionales como el arrastre de la ventana.

Métodos Principales

1. Constructor TitleBar(MainFrame mainFrame)

- Descripción: Inicializa la barra de título con botones y un título dinámico.
- Parámetros:

- MainFrame mainFrame: Referencia al marco principal para interactuar con sus componentes.

- Funcionalidad Principal:

1 Configuración de Dimensiones:

- Define un tamaño fijo de 32 píxeles de alto, igualando el ancho utilizable de la ventana principal.

2 Configuración de Componentes:

- Botón back:
  - Navega al panel "Usuarios" y actualiza el texto del título a "Market".
  - Se desactiva tras la acción.
- Botón close:
  - Cierra la aplicación llamando a mainFrame.dispose() y terminando el proceso con System.exit(0).
- Etiqueta title:
  - Muestra el texto inicial "Market" y es ajustable dinámicamente.

3 Arrastre de Ventana:

- Implementa un MouseAdapter que permite mover la ventana arrastrando la barra de título. Calcula el desplazamiento en función de las coordenadas del ratón.

2. Métodos de Acceso

- getBackButton() y getCloseButton():
  - Devuelven referencias a los botones back y close, permitiendo acceso externo a sus propiedades y métodos.
- getTitleLabel():
  - Devuelve la etiqueta title, lo que permite modificar su texto desde otras partes de la aplicación.

Variables Globales Principales

1 MainFrame mainFrame:

- Referencia al marco principal, utilizada para cambiar paneles y acceder a métodos relevantes.

2 Botones Personalizados (CustomButton.TitleBar back, close):

- back: Botón para regresar al panel "Usuarios".
- close: Botón para cerrar la aplicación.

3 Etiqueta Personalizada (CustomLabel title):

- Muestra el título actual de la ventana y sirve como área de interacción para arrastrar la ventana.

Bloques de Código Relevantes

1 Acción del Botón back:

- Llama a mainFrame.getCardLayout().show() para cambiar al panel "Usuarios".
- Actualiza el texto del título a "Market".
- Desactiva el botón tras el uso.

2 Acción del Botón close:

- Cierra la ventana principal y finaliza el proceso de ejecución.

3 Arrastre de Ventana:

- Calcula las nuevas coordenadas de la ventana en función de la posición del ratón al ser arrastrada.
- Utiliza los métodos mousePressed y mouseDragged para manejar este comportamiento.

Diseño Visual

- Diseño de Disposición:
  - Utiliza un GridBagLayout para organizar los componentes en una sola fila:
    - Botón back a la izquierda.
    - Etiqueta title centrada.
    - Botón close a la derecha.
- Estilo Visual:
  - Se emplean componentes personalizados (CustomButton, CustomLabel) para mantener una estética uniforme con el resto de la aplicación.

Dependencias Externas

1 Paquete UI:

- Contiene la clase MainFrame.

2 Paquete UI.Assets:

- Incluye componentes personalizados como CustomButton y CustomLabel.

3 Librerías Swing y AWT:

- Utilizadas para la construcción de la interfaz gráfica y la gestión de eventos.

Notas Finales

La clase TitleBar proporciona una barra de título personalizada que reemplaza la barra predeterminada de la ventana. Esto permite un mayor control sobre la funcionalidad y el diseño visual, incluyendo botones configurables y arrastre de ventana. La modularidad de sus métodos de acceso y el uso de componentes personalizados facilitan la integración con el resto de la aplicación.

UserSelector

Informe Técnico del Archivo Java: UserSelector.java

Clase Principal: UserSelector

La clase UserSelector extiende de javax.swing.JPanel y se utiliza para representar un panel de selección de perfiles en la interfaz gráfica. Proporciona opciones visuales y botones interactivos para que el usuario seleccione un perfil entre "Administración", "Ventas" y "Membresías".

Métodos Principales

1. Constructor UserSelector(MainFrame mainFrame)
   - Descripción: Configura y organiza los componentes del panel de selección de perfiles.
   - Parámetros:
     - MainFrame mainFrame: Referencia al marco principal para futuras interacciones.
   - Funcionalidad Principal:
     1 Configuración del Panel:
       - Establece bordes y diseño mediante BorderFactory.createEmptyBorder y GridBagLayout.
       - Define un fondo transparente con setOpaque(false).
     2 Componentes Principales:
       - Etiqueta title: Muestra el título "Elija un perfil" en la parte superior.
       - Etiquetas de Descripción:

- adminText muestra "Administración".
- sellText muestra "Ventas".
- memberText muestra "Membresías".

- Botones:
  - admin: Representa la opción para el perfil de administración.
  - sale: Representa la opción para el perfil de ventas.
  - member: Representa la opción para el perfil de membresías.

3 Disposición de Componentes:

- Utiliza GridBagConstraints para ubicar los elementos en una cuadrícula.
- Define márgenes y alineaciones para cada componente usando Insets.

2. Métodos de Acceso

- getAdminButton():
  - Devuelve el botón admin correspondiente al perfil de administración.
- getSaleButton():
  - Devuelve el botón sale correspondiente al perfil de ventas.
- getMemberButton():
  - Devuelve el botón member correspondiente al perfil de membresías.

Variables Globales Principales

1 Etiquetas Personalizadas (CustomLabel):

- title: Encabezado principal del panel.
- adminText, sellText, memberText: Etiquetas descriptivas asociadas a cada botón.

2 Botones Personalizados (CustomButton):

- admin: Botón con diseño específico para el perfil de administración.
- sale: Botón para el perfil de ventas.
- member: Botón para el perfil de membresías.

Bloques de Código Relevantes

1 Configuración de GridBagConstraints:

- Se utiliza para alinear los componentes en una cuadrícula de tres columnas:
  - Primera fila: Muestra la etiqueta title.
  - Segunda fila: Coloca los botones admin, sale, y member con márgenes diferenciados.
  - Tercera fila: Coloca las etiquetas descriptivas bajo cada botón.

2 Espaciado y Estilo:

- El panel se configura con márgenes internos uniformes de 130 píxeles en los bordes superior e inferior, y de 180 píxeles a los lados.
- Los márgenes entre componentes se manejan mediante Insets.

Diseño Visual

- Estilo Uniforme:
  - Se emplean componentes personalizados (CustomButton, CustomLabel) para mantener una apariencia consistente.
- Distribución de Espacio:
  - Proporciones bien definidas mediante márgenes (Insets) y dimensiones específicas (setPreferredSize).

Dependencias Externas

1 Paquete UI:

- Contiene la clase MainFrame, necesaria para referencia y posibles interacciones futuras.

2 Paquete UI.Assets:

- Proporciona componentes personalizados como CustomButton y CustomLabel.

3 Librerías de Swing:

- Usadas para crear y organizar los componentes gráficos.

Notas Finales

La clase UserSelector está diseñada para ser un panel central en la selección de roles de usuario dentro de la aplicación. Su diseño modular permite una fácil integración con el resto del sistema, y los botones personalizados ofrecen una interfaz intuitiva y visualmente atractiva para la navegación entre perfiles.

Admin

Informe Técnico del Archivo Java: Admin.java

Clase Principal: Admin

La clase Admin extiende de javax.swing.JPanel y representa un panel administrativo dentro de la interfaz gráfica de usuario. Este panel permite gestionar elementos (ítems), proporcionando funciones para añadir, editar y eliminar registros.

Métodos Principales

1. Constructor Admin(MainFrame mainFrame)
   - Descripción: Inicializa el panel de administración y configura sus componentes principales.
   - Parámetros:
     - MainFrame mainFrame: Referencia al marco principal de la aplicación para integración y navegación.
   - Funcionalidad Principal:
     1 Configuración del Panel:
       - Se define un diseño de cuadrícula con bordes uniformes mediante GridBagLayout y BorderFactory.createEmptyBorder.
       - Se establece un fondo transparente utilizando setOpaque(false).
     2 Componentes Principales:
       - Tabla Desplazable (CustomScroll):
         - Se inicializa con un ancho de 720 píxeles y un alto de 398 píxeles.
         - Los datos iniciales se cargan desde un archivo utilizando el método Item.leerItemsDesdeArchivo().
       - Botones Funcionales (CustomButton.Option):
         - addItem: Botón para añadir nuevos ítems.
         - editItem: Botón para editar ítems seleccionados en la tabla.
         - deleteItem: Botón para eliminar ítems seleccionados en la tabla.
     3 Manejadores de Eventos:
       - addItem: Abre una nueva ventana de entrada (FieldFrame.AddItem) para añadir ítems.

- editItem: Llama a FieldFrame.EditItem con la fila seleccionada para editarla. No hace nada si no se selecciona una fila.
- deleteItem: Llama a FieldFrame.DeleteItem con la fila seleccionada para eliminarla. No hace nada si no se selecciona una fila.

4 Distribución de Componentes:

- Se utiliza GridBagConstraints para organizar la tabla desplazable y los botones en una cuadrícula con márgenes específicos.

2. Métodos de Acceso

- getMainFrame():
  - Devuelve la referencia al marco principal de la aplicación (MainFrame).
- getScroll():
  - Devuelve el componente de desplazamiento (CustomScroll), que contiene la tabla de ítems.

Variables Globales Principales

1 CustomScroll scroll:

- Contenedor desplazable que presenta una tabla de datos cargados desde el archivo de ítems.

2 Botones Funcionales (CustomButton.Option):

- addItem: Botón para añadir nuevos ítems al sistema.
- editItem: Botón para modificar un ítem existente seleccionado en la tabla.
- deleteItem: Botón para eliminar un ítem existente seleccionado en la tabla.

Bloques de Código Relevantes

1 Carga de Datos en CustomScroll:

- Método Item.leerItemsDesdeArchivo():
  - Proporciona la información inicial que se mostrará en la tabla. Este método probablemente lee y devuelve los datos de un archivo externo.

2 Control de Fila Seleccionada:

- Los botones editItem y deleteItem validan que se haya seleccionado una fila en la tabla antes de realizar cualquier acción.

3 Configuración de GridBagConstraints:

- Se establece una cuadrícula con tres columnas:

- Primera fila: La tabla desplazable ocupa las tres columnas.
- Segunda fila: Los botones se distribuyen uniformemente con márgenes laterales.

Diseño Visual

- Estilo Minimalista y Ordenado:
  - El panel utiliza bordes de 40 píxeles para crear un espacio uniforme alrededor de los elementos.
  - Los botones tienen un diseño limpio y están espaciados uniformemente mediante Insets.
- Interactividad Intuitiva:
  - Los botones proporcionan accesos directos claros para la gestión de ítems.
  - El uso de ventanas adicionales (FieldFrame.AddItem, FieldFrame.EditItem, FieldFrame.DeleteItem) mejora la experiencia del usuario al segmentar las tareas.

Dependencias Externas

1. Paquete UI:
   - Contiene la clase MainFrame, necesaria para la integración con el resto de la aplicación.
2. Paquete UI.Assets:
   - Proporciona componentes personalizados como CustomScroll y CustomButton.Option.
3. Clase Item:
   - Proporciona el método leerItemsDesdeArchivo() para cargar los datos de los ítems en el panel.
4. Clases FieldFrame.AddItem, FieldFrame.EditItem, FieldFrame.DeleteItem:
   - Gestionan las acciones específicas de añadir, editar y eliminar ítems.

Notas Finales

La clase Admin es fundamental para la gestión administrativa dentro de la aplicación. Su diseño modular permite un manejo claro y eficiente de los ítems, mientras que la distribución de sus componentes asegura una interfaz organizada y fácil de usar.

Members

Informe Técnico del Archivo Java: Members.java

Clase Principal: Members

La clase Members extiende de javax.swing.JPanel y representa un panel para la gestión de miembros dentro de la interfaz gráfica de usuario. Permite visualizar, agregar, editar y eliminar miembros de un sistema, interactuando con una lista de datos cargados desde un archivo CSV.

Métodos Principales

1. Constructor Members(MainFrame mainFrame)
   - Descripción: Inicializa el panel de miembros y configura sus componentes principales.
   - Parámetros:
     - MainFrame mainFrame: Referencia al marco principal de la aplicación para integración y navegación.
   - Funcionalidad Principal:
     1 Configuración del Panel:
       - Utiliza un diseño de cuadrícula con bordes de 40 píxeles mediante GridBagLayout y BorderFactory.createEmptyBorder.
       - Se establece un fondo transparente mediante setOpaque(false).
     2 Componentes Principales:
       - Tabla Desplazable (CustomScroll):
         - Se inicializa con un ancho de 720 píxeles y un alto de 398 píxeles.
         - Los datos iniciales se cargan desde un archivo CSV utilizando el método Miembros.leerSociosDesdeArchivo("src\\data\\DatosM.csv").
       - Botones Funcionales (CustomButton.Option):
         - addMember: Botón para añadir nuevos miembros.
         - editMember: Botón para editar miembros seleccionados en la tabla.
         - deleteMember: Botón para eliminar miembros seleccionados en la tabla.

3 Manejadores de Eventos:

- addMember: Abre una nueva ventana de entrada (FieldFrame.AddMember) para añadir miembros.
- editMember: Llama a FieldFrame.EditMember con la fila seleccionada para editarla. No hace nada si no se selecciona una fila.
- deleteMember: Llama a FieldFrame.DeleteMember con la fila seleccionada para eliminarla. No hace nada si no se selecciona una fila.

4 Distribución de Componentes:

- Se utiliza GridBagConstraints para organizar la tabla desplazable y los botones en una cuadrícula con márgenes específicos.

2. Métodos de Acceso

- getMainFrame():
  - Devuelve la referencia al marco principal de la aplicación (MainFrame).
- getScroll():
  - Devuelve el componente de desplazamiento (CustomScroll), que contiene la tabla de miembros.

Variables Globales Principales

1 CustomScroll scroll:

- Contenedor desplazable que presenta una tabla de datos cargados desde el archivo CSV de miembros.

2 Botones Funcionales (CustomButton.Option):

- addMember: Botón para añadir nuevos miembros al sistema.
- editMember: Botón para modificar un miembro existente seleccionado en la tabla.
- deleteMember: Botón para eliminar un miembro existente seleccionado en la tabla.

Bloques de Código Relevantes

1 Carga de Datos en CustomScroll:

- Método Miembros.leerSociosDesdeArchivo():
  - Este método lee los datos de miembros desde un archivo CSV y los pasa a la tabla de miembros.

2 Control de Fila Seleccionada:

- Los botones editMember y deleteMember validan que se haya seleccionado una fila en la tabla antes de realizar cualquier acción.

3 Configuración de GridBagConstraints:

- Se establece una cuadrícula con tres columnas:
  - Primera fila: La tabla desplazable ocupa las tres columnas.
  - Segunda fila: Los botones se distribuyen uniformemente con márgenes laterales.

Diseño Visual

- Estilo Minimalista y Ordenado:
  - El panel utiliza bordes de 40 píxeles para crear un espacio uniforme alrededor de los elementos.
  - Los botones tienen un diseño limpio y están espaciados uniformemente mediante Insets.
- Interactividad Intuitiva:
  - Los botones proporcionan accesos directos claros para la gestión de miembros.
  - El uso de ventanas adicionales (FieldFrame.AddMember, FieldFrame.EditMember, FieldFrame.DeleteMember) mejora la experiencia del usuario al segmentar las tareas.

Dependencias Externas

1 Paquete UI:

- Contiene la clase MainFrame, necesaria para la integración con el resto de la aplicación.

2 Paquete UI.Assets:

- Proporciona componentes personalizados como CustomScroll y CustomButton.Option.

3 Clases Miembros y Item:

- Miembros proporciona el método leerSociosDesdeArchivo() para cargar los datos de los miembros desde el archivo CSV.

4 Clases FieldFrame.AddMember, FieldFrame.EditMember, FieldFrame.DeleteMember:

- Gestionan las acciones específicas de añadir, editar y eliminar miembros.

Notas Finales

La clase Members es fundamental para la gestión de los miembros dentro de la aplicación. Su diseño modular permite un manejo eficiente de los miembros, mientras que la distribución de sus componentes asegura una interfaz organizada y fácil de usar.

Sales

Reporte Técnico: Clase Sales

Descripción General

La clase Sales extiende JPanel y forma parte de la interfaz de usuario de una aplicación de ventas. Esta clase se encarga de gestionar la interacción con los componentes visuales, como botones, tablas, y campos de texto, relacionados con el proceso de venta de productos, así como de controlar eventos relacionados con la selección de productos, miembros y la actualización de la base de datos de inventario.

Componentes Principales

- Variables Globales:
  - mainFrame: Una instancia de la clase MainFrame, que representa la ventana principal de la aplicación.
  - scroll: Componente CustomScroll que contiene una tabla de ventas.
  - member: Componente CustomCombo para seleccionar un miembro del cliente.
  - item: Componente CustomCombo para seleccionar un producto.
  - text: Componente CustomLabel que muestra el texto "Total acumulado".
  - total: Componente CustomLabel que muestra el total acumulado de la venta.
  - reset, cashback, complete: Botones de tipo CustomButton.Option que permiten reiniciar la venta, aplicar un cashback, y completar la venta, respectivamente.
  - delete, add: Botones de tipo CustomButton.Decision para eliminar o agregar productos a la venta.
  - totalValue: Variable de tipo double que guarda el total acumulado de la venta.

Métodos Importantes

Constructor Sales(MainFrame mainFrame)

Este constructor configura el panel, inicializando los componentes de la interfaz de usuario y estableciendo las configuraciones iniciales, como el diseño y el tamaño de los

componentes visuales. Además, se agregan listeners a los botones para manejar eventos de acción como agregar, eliminar productos, reiniciar la venta o aplicar un cashback.

cleanTable()
Este método limpia la tabla de ventas eliminando todas las filas de la misma.

Métodos de Acción:

1. reset.addActionListener: Este listener es asignado al botón de "Reiniciar". Su función es limpiar la tabla de ventas y restablecer el valor acumulado a cero.
2. cashback.addActionListener: Este listener se activa al presionar el botón "Usar cashback". Si el miembro seleccionado es de tipo "PREMIUM", permite aplicar un descuento en forma de cashback al total de la venta.
3. complete.addActionListener: Este listener se activa al presionar el botón "Completar venta". Su propósito es actualizar el inventario reduciendo el stock de los productos vendidos, aplicar el cashback (si corresponde), y guardar los cambios en el archivo de miembros y productos.
4. delete.addActionListener: Este listener es activado por el botón "Eliminar". Permite eliminar un producto de la tabla de ventas, actualizando la cantidad y el precio final, además de ajustar el total acumulado.
5. add.addActionListener: Este listener se activa al presionar el botón "Añadir". Permite agregar un producto a la tabla de ventas. Si el producto ya existe, incrementa su cantidad, actualizando el precio final y el total acumulado.

Bloques de Código Importantes

Bloques try-catch
Aunque el código proporcionado no tiene bloques explícitos try-catch para manejo de excepciones, se realiza una validación y gestión de errores indirecta al manejar acciones como la reducción de stock o la aplicación de cashback, en los cuales se verifican varias condiciones (por ejemplo, si el miembro es premium y si existen productos en stock).

Ciclos for y Condicionales
Los ciclos for se utilizan en varios puntos del código, especialmente para recorrer las filas de las tablas y verificar si los productos ya están presentes en la venta o en el inventario. Los condicionales dentro de estos ciclos permiten gestionar las modificaciones de los datos, como agregar o eliminar productos, o aplicar el cashback.

Por ejemplo:

- Ciclo para reducir stock en complete.addActionListener: Se recorre la tabla de inventario y, por cada producto vendido, se reduce la cantidad de stock disponible.

- Ciclo para buscar un miembro premium en cashback.addActionListener: Se verifica si el miembro seleccionado es premium y, en caso afirmativo, se calcula el cashback a aplicar.

Manejo de Datos y Persistencia

- Actualización de Inventario: Cuando se completa la venta (complete.addActionListener), se recorre la tabla de ventas y se actualiza el inventario reduciendo el stock de los productos vendidos.
- Guardar Datos: Tras completar la venta o aplicar un cashback, los cambios en el inventario y los miembros se guardan en archivos correspondientes mediante los métodos Item.guardarItems y Miembros.guardarTableModelEnArchivo.

Conclusiones

La clase Sales se encarga de manejar la lógica relacionada con las ventas en la interfaz de usuario. Gestiona las interacciones con los productos, los miembros, y el proceso de actualización de datos tanto en la interfaz como en los archivos de persistencia. La clase hace uso de componentes personalizados como CustomScroll, CustomCombo, y CustomButton, facilitando la interacción del usuario. El manejo de eventos de acción es un componente clave para las funcionalidades, permitiendo una gestión dinámica y reactiva de las ventas.

FieldFrame

Reporte Técnico: Clase FieldFrame

1. Nombre de la clase:

- FieldFrame

2. Propósito de la clase: La clase FieldFrame representa una ventana de formulario dentro de una aplicación de escritorio basada en Java Swing. Su propósito es proporcionar una interfaz para gestionar la creación, edición y eliminación de elementos (items) en una tabla. Esta clase sirve como base para varias ventanas (diálogos) de interacción con los ítems, como la adición, edición y eliminación de los mismos.

3. Variables Globales Importantes:

- shadow (Shadow): Representa un objeto que genera una sombra visual alrededor de la ventana de la aplicación, lo que mejora la apariencia estética de la interfaz. Esta variable se inicializa en las clases hijas de FieldFrame.
- money, percentage, letter, digit, slash (KeyAdapter): Son instancias de la clase KeyAdapter, que sirven para restringir los tipos de caracteres que un usuario puede ingresar en campos específicos (por ejemplo, solo números o solo letras). Estas variables son claves para la validación de los datos ingresados en los formularios.

- data (DefaultTableModel): Es el modelo de la tabla donde se almacenan los elementos. Se utiliza en las clases hijas para acceder y modificar los datos de la tabla.
- title, name, id, desc, buy, sell, discount, stock (CustomLabel, CustomField): Son componentes personalizados (como etiquetas y campos de texto) que conforman los formularios en las ventanas de la aplicación. Estos elementos se utilizan para mostrar información al usuario y recibir sus entradas.
- save, cancel (CustomButton): Son botones personalizados que permiten al usuario guardar los cambios realizados o cancelar la acción. Se asocian con eventos de acción que se manejan mediante ActionListener.

4. Métodos Principales: checkDiscount(CustomField discount, String discountS): Este método verifica y valida el valor de un campo de descuento (CustomField). Si el valor es mayor a 100%, el descuento se ajusta a "100%". Si el campo contiene un valor válido, lo formatea adecuadamente. Este método es crucial para asegurar que los valores ingresados en el campo de descuento no excedan un límite lógico.

- addZeros(String idS): Método que asegura que el ID de un item tenga al menos 4 caracteres, agregando ceros al principio si es necesario. Este método es útil para mantener una estructura uniforme en los ID de los ítems.
- AddItem(Admin adminPanel) (Subclase de FieldFrame): Constructor y método principal para crear un nuevo ítem. Inicializa los campos del formulario, configura los listeners para cada campo y define el comportamiento de los botones "Cancelar" y "Añadir".
    - El botón "Añadir" valida que todos los campos estén completos y que el ID del ítem no esté duplicado, agregando el nuevo ítem al modelo de la tabla si pasa la validación.
- EditItem(Admin adminPanel, int row) (Subclase de FieldFrame):
    - Constructor y método principal para editar un ítem existente. Carga los valores del ítem seleccionado en los campos del formulario, permite modificarlos y guarda los cambios en la tabla. Similar a AddItem, pero con la diferencia de que modifica una fila ya existente en lugar de agregar una nueva.
- DeleteItem(Admin adminPanel, int row) (Subclase de FieldFrame): Constructor y método principal para eliminar un ítem de la tabla. Muestra una ventana de confirmación y, si el usuario acepta, elimina la fila correspondiente del modelo de datos.

5. Excepciones y Manejo de Errores: La clase FieldFrame y sus subclases no utilizan bloques try-catch de forma explícita en el código proporcionado. Sin embargo, hay validaciones importantes que previenen la entrada incorrecta de datos, como por ejemplo:

- Validación de campos: Se asegura que los campos no estén vacíos y que los valores ingresados sean válidos, como en los campos de porcentaje (descuento) o dinero (precio de compra y venta).
- Verificación de ID único: En el método AddItem, se comprueba que el ID ingresado no exista ya en la lista de ítems antes de permitir que se añada un nuevo ítem, lo cual previene duplicados.

Aunque no hay un manejo explícito de excepciones, el código depende de una validación rigurosa de los datos antes de realizar operaciones en la tabla, lo que reduce la probabilidad de errores o excepciones durante la ejecución.

6. Explicación de Bloques de Código:

- Bloques KeyListener: Cada campo tiene un KeyListener asociado que limita el tipo de caracteres que se pueden escribir. Estos listeners son definidos como instancias de KeyAdapter, una clase abstracta que permite interceptar eventos de teclas y restringir entradas específicas, como números, letras, o símbolos. Por ejemplo, el money listener solo permite números, el símbolo de dólar y puntos, y el percentage listener solo permite números y el símbolo %.
- GridBagLayout: El diseño de la interfaz gráfica de los formularios se organiza usando GridBagLayout. Este layout permite posicionar los componentes de manera flexible y controlada, utilizando restricciones (GridBagConstraints) que definen cómo y dónde se ubican los elementos dentro de la ventana.
- Interacción con la Tabla (DefaultTableModel): Los ítems se gestionan a través de una tabla cuyo modelo es un DefaultTableModel. Este modelo permite agregar, editar y eliminar filas. Los métodos como addRow(), removeRow(), y insertRow() son fundamentales para modificar el contenido de la tabla. Cada vez que se realiza una modificación en la tabla, se guarda el estado de los ítems utilizando el método Item.guardarItems().

7. Conclusión: La clase FieldFrame y sus subclases proporcionan una estructura sólida para la gestión de ítems en una interfaz gráfica. A través de métodos como AddItem, EditItem y DeleteItem, los usuarios pueden interactuar con una tabla de datos, validando y manipulando los ítems de acuerdo con las reglas definidas. Las principales funcionalidades se basan en la validación de entrada y la manipulación del modelo de datos de la tabla, asegurando que los datos sean consistentes y correctos.

Reporte Técnico: Análisis del código fuente

Nombre de la clase: AddMember, EditMember, DeleteMember, UseCashback y FieldFrame (Clase base abstracta)

Este reporte describe las clases y métodos principales de los fragmentos de código proporcionados, destacando las variables y bloques de código relevantes para el funcionamiento de cada una de las clases.

Clase: AddMember

Propósito: Esta clase tiene como objetivo gestionar la adición de un nuevo miembro a través de un formulario interactivo. Utiliza una ventana emergente (con sombra) para ingresar datos y luego actualizar la tabla de miembros.

Métodos principales:

1. Constructor (AddMember(Members memberPanel)):
   - Inicializa la ventana emergente y los componentes visuales.
   - Asocia componentes visuales (CustomLabel, CustomField, CustomButton) a los elementos de la interfaz, como el nombre, tipo, dirección, teléfono, RFC, entre otros.
   - Configura los KeyListeners para validar los campos (letras, dígitos, fechas).
   - Implementa los botones "Cancelar" y "Añadir" con sus respectivas acciones:
     - Cancelar: Cierra la ventana.
     - Añadir: Valida los campos ingresados y agrega los datos a un DefaultTableModel. Luego guarda la tabla en un archivo CSV.

Variables globales relevantes:

- memberPanel: Panel que contiene la lista de miembros.
- title, name, type, address, tel, rfc, aditional1, aditional2, start: Componentes de la interfaz gráfica.
- cancel, add: Botones para cancelar o agregar un miembro.

Bloques de código:

- Validación de campos: El campo "Fecha de inicio" se establece por defecto con la fecha actual y se formatea con un KeyListener.
- Acción del botón "Añadir": Verifica si todos los campos son válidos y luego agrega una nueva fila a la tabla.

Clase: EditMember

Propósito: Permite editar los detalles de un miembro existente en la lista, mostrando los valores actuales para que el usuario los modifique.

Métodos principales:

1 Constructor (EditMember(Members memberPanel, int row)):
   - Similar al de AddMember, pero en lugar de crear un nuevo miembro, edita los valores existentes de un miembro seleccionando una fila de la tabla.
   - Actualiza los campos de texto con los valores actuales de la fila seleccionada.

Variables globales relevantes:

- memberPanel: Panel que contiene la lista de miembros.
- title, name, type, address, tel, rfc, aditional1, aditional2, start: Componentes de la interfaz gráfica, con los valores preestablecidos de la fila seleccionada.
- cancel, add: Botones para cancelar o guardar los cambios.

Bloques de código:

- Acción del botón "Guardar": Similar a la de AddMember, valida los campos y reemplaza la fila seleccionada con los nuevos valores.

Clase: DeleteMember

Propósito: Muestra una ventana de confirmación para eliminar un miembro de la lista de miembros.

Métodos principales:

1 Constructor (DeleteMember(Members memberPanel, int row)): Muestra un mensaje de confirmación para eliminar un miembro de la fila seleccionada en la tabla.

Variables globales relevantes:

- memberPanel: Panel que contiene la lista de miembros.
- data: Modelo de datos de la tabla de miembros.
- title, text1, text2: Componentes de la interfaz para mostrar el mensaje de confirmación.

- cancel, accept: Botones para cancelar o aceptar la eliminación.

Bloques de código:

- Acción del botón "Aceptar": Elimina la fila correspondiente en el DefaultTableModel.

Clase: UseCashback

Propósito: Permite a un miembro utilizar su cashback disponible en una compra, actualizando tanto el total de la compra como el valor del cashback del miembro.

Métodos principales:

1 Constructor (UseCashback(Sales salesPanel, String name)): Busca el valor de cashback de un miembro específico basado en su nombre.

- Muestra una ventana de confirmación para usar el cashback en una compra.

Variables globales relevantes:

- salesPanel: Panel que maneja la venta y el total.
- data: Modelo de datos de la tabla de ventas.
- cashback: Monto disponible para el miembro.
- row: Índice de la fila del miembro en la tabla.

Bloques de código:

- Búsqueda de cashback: Recorre las filas del DefaultTableModel para encontrar el miembro con el nombre proporcionado y extraer su cashback.
- Acción del botón "Aceptar": Actualiza el total de la compra restando el cashback disponible y ajusta el valor del cashback del miembro.

Clase base: FieldFrame

Propósito: Clase base para las ventanas emergentes utilizadas en AddMember, EditMember, DeleteMember y UseCashback, proporcionando funcionalidades comunes como la creación de sombra (efecto visual).

Métodos principales:

1 setGeneralSettings(MainFrame mainFrame):

- Configura la ventana para que se muestre sin decoración, establezca el tamaño y se centre sobre el MainFrame.

Variables globales relevantes:

- shadow: Representa la sombra que acompaña a cada ventana emergente.

Excepciones y control de errores:

No se manejan explícitamente excepciones en este fragmento de código. Sin embargo, es recomendable agregar validaciones adicionales para manejar posibles errores en la entrada de datos (por ejemplo, formato de fecha incorrecto o valores numéricos no válidos).

Este análisis cubre las funcionalidades principales de las clases proporcionadas y sus interacciones dentro de la interfaz de usuario para la gestión de miembros y sus operaciones relacionadas.

Excepciones y control de errores

No se manejan explícitamente excepciones en este fragmento de código. Sin embargo, es recomendable agregar validaciones adicionales para manejar posibles errores en la entrada de datos (por ejemplo, formato de fecha incorrecto o valores numéricos no válidos).

Este análisis cubre las funcionalidades principales de las clases proporcionadas y sus interacciones dentro de la interfaz de usuario para la gestión de miembros y sus operaciones relacionadas.

Baja de producto

El proceso para eliminar un producto o ítem del inventario requiere llevar a cabo los siguientes pasos:

1. Abrir la aplicación: Iniciar la aplicación y asegurarse de estar en la ventana principal.
2. Acceder al *Panel de Administración*: Navegar al panel de administración desde el menú principal o la barra de navegación.
3. Seleccionar el producto: Buscar y seleccionar el producto que se desea dar de baja en la lista de ítems.
4. Verificar la eliminación: Verificar que el producto haya sido eliminado correctamente de la lista de ítems en el panel de administración.

Lógica y código

Informe Técnico del Archivo Java: Files.java

Clase Principal: Files

La clase Files se utiliza para gestionar archivos relacionados con los objetos Item y Socio. Permite leer, guardar, y verificar la existencia de estos archivos, además de crear directorios si es necesario.

Variables Globales:

1. itemsFilePath (String): Ruta del archivo que almacena la información de los objetos Item. Se inicializa mediante el constructor y puede modificarse con el método setItemsFilePath.

2. sociosFilePath (String): Ruta del archivo que contiene la información de los objetos Socio. Se inicializa mediante el constructor y puede modificarse con el método setSociosFilePath.

3. DELIMITER (String - static final): Separador de campos utilizado para procesar los datos en los archivos (",").

Métodos Principales

1. readFileItems()

2. readFileSocios()

3. saveFileItems(List)

4. saveFileSocios(List)

5. filesExist()

6. createFiles()

7. createDirectories(Path carpetaPath)

8. exists(Path carpetaPath)

Bloques de Código Destacados

1. Estructuras Try-With-Resources: Se utilizan en los métodos de lectura y escritura (readFileItems, readFileSocios, saveFileItems, saveFileSocios) para asegurar que los recursos como BufferedReader y BufferedWriter se cierren automáticamente, evitando fugas de memoria.

2. Manejo de Excepciones: Las excepciones comunes (IOException, NumberFormatException, IllegalArgumentException) son capturadas e informadas al usuario a través de mensajes de error.

3. Uso de Métodos Estáticos: Métodos como createDirectories y exists facilitan tareas comunes relacionadas con archivos y directorios.

Notas Finales

La clase Files está diseñada para gestionar de manera robusta y modular los archivos de datos, asegurando la persistencia y la integridad de los objetos Item y Socio. Hace un uso eficiente de las herramientas estándar de Java para manejo de archivos y excepciones, lo que refuerza su fiabilidad y legibilidad.

Reporte Técnico

Nombre de la Clase: Item

Resumen General:

La clase Item representa un artículo en un sistema de gestión de inventarios. Define atributos relacionados con la descripción del artículo, métodos para manipular datos como compra, cálculo de precios, gestión de stock, y funciones estáticas para interactuar con un archivo CSV. Además, proporciona utilidades para integrarse con interfaces gráficas mediante el uso de DefaultTableModel.

Atributos Globales Importantes:

Atributos de instancia: nombre (String): Nombre del artículo, id (String): Identificador único del artículo, descripcion (String): Descripción del artículo, precioCompra (double): Precio de compra del artículo, precioVenta (double): Precio de venta del artículo, descuento (int): Porcentaje de descuento aplicado al artículo, stock (int): Cantidad disponible en inventario.

Constantes:

1. CARPETA (String): Ruta de la carpeta que contiene los archivos de datos.

2. RUTA_ARCHIVO (String): Ruta completa al archivo CSV donde se almacenan los datos de los artículos.

3. STOCK_MINIMO (int): Umbral de stock mínimo para generar una alerta.

Métodos Principales:

1. Constructor: Item(String nombre, String id, String descripcion, double precioCompra, double precioVenta, int descuento, int stock): Inicializa una instancia con los valores especificados.

2. Gestión de stock: boolean compraArticulo(int cantidad) Reduce el stock si hay suficiente inventario. Genera una alerta si el stock cae por debajo de STOCK_MINIMO.

3. Excepciones manejadas: IllegalArgumentException si la cantidad es menor o igual a cero.

4. static boolean reducirStock(String id, int cantidad): Busca un artículo por su ID en el archivo CSV y reduce el stock en la cantidad especificada.

Bloques clave:

1. Lee el archivo línea por línea para actualizar el stock: Escribe nuevamente el archivo tras realizar cambios.

2. Gestión de precios: String getPrecioConDescuento(): Calcula y retorna el precio de venta tras aplicar el descuento.

3. String[] obtenerDetallesItem(String id): Devuelve información detallada del artículo, incluyendo precio con descuento.

4. Interacción con archivos: static void guardarItems(DefaultTableModel tableModel): Escribe los datos de un modelo de tabla en el archivo CSV. Verifica y crea la carpeta necesaria si no existe.

5. static DefaultTableModel leerItemsDesdeArchivo(): Lee el archivo CSV y retorna un modelo de tabla (DefaultTableModel) con los datos cargados.

6. static boolean existeID(String id): Busca un ID específico en el archivo para determinar si existe.

7. Gestión de alertas: private void verificarStockMinimo(): Imprime un mensaje de alerta en consola si el stock del artículo cae por debajo de STOCK_MINIMO.

8. Métodos auxiliares: @Override String toString(): Retorna una representación en formato texto de los datos del artículo.

Explicación de Bloques Importantes:

1. Bloque de manejo de archivos: Los métodos guardarItems y leerItemsDesdeArchivo usan estructuras try-with-resources para garantizar el cierre de los recursos de escritura y lectura. Verifican la existencia de directorios y crean nuevos si es necesario, utilizando utilidades como Paths y Files.

2. Bloque de verificación de stock: El método verificarStockMinimo se ejecuta tras cada compra para asegurar que el stock no baje del nivel mínimo definido. Este diseño garantiza la detección y notificación inmediata de problemas de inventario.

3. Estructuras de control: Ciclo en reducirStock: Recorre todas las líneas del archivo CSV, identifica el artículo correspondiente y actualiza su stock, asegurando consistencia en los datos.

4. Bloque condicional en existeID: Implementa un flujo para detectar la existencia del ID con comprobaciones de línea vacía o encabezados.

Reporte Técnico

Nombre de la Clase: Miembros

Esta clase pertenece al paquete Item y se encarga de la gestión de socios, incluyendo lectura y escritura de datos desde/para un archivo CSV, manejo de fechas de membresía, y representación de los datos en una tabla (DefaultTableModel).

Variables Globales más importantes:

1. RUTA_ARCHIVO: Tipo: String Descripción: Define la ruta del archivo CSV donde se almacenan los datos de los socios. Uso: Se utiliza como valor predeterminado para la ruta de los archivos en operaciones de entrada y salida.

Métodos Más Importantes:

1. calcularFechaVencimiento

a) Propósito: Calcula la fecha de vencimiento de una membresía, sumando un año a una fecha de inicio dada. Parámetros: LocalDate fechaInicio: Fecha de inicio de la membresía. Retorno: Una cadena formateada (dd/MM/yyyy) que representa la fecha de vencimiento. Retorna una cadena vacía si la fecha de inicio es null.

2. leerSociosDesdeArchivo

a) Propósito: Lee un archivo CSV que contiene información de los socios y construye un modelo de tabla (DefaultTableModel) para representar los datos. Parámetros: String rutaArchivo: Ruta del archivo CSV a leer.

b) Retorno: Un objeto DefaultTableModel que contiene los datos de los socios.

Detalles Importantes: Control de errores: Maneja excepciones relacionadas con la lectura del archivo usando un bloque try-catch. Registra errores en líneas específicas del archivo con mensajes detallados. Lógica de lectura: Se asegura de que cada línea tenga al menos 9 campos (para incluir fechas y cashback). Se implementa soporte para múltiples formatos de fecha (dd/MM/yyyy, yyyy-MM-dd, yyyy/MM/dd) mediante un ciclo de prueba.

Cálculo de fecha de fin: Si no se proporciona una fecha de fin válida, se calcula como un año después de la fecha de inicio.

3. convertirSociosATableModel

a) Propósito: Convierte una lista de objetos Socio en un modelo de tabla (DefaultTableModel).

b) Parámetros: List<Socio> socios: Lista de objetos Socio. Retorno: Un objeto DefaultTableModel con las columnas {"Nombre", "Tipo", "Dirección", "Teléfono", "RFC", "Add1", "Add2", "Fecha I.", "Fecha F.", "Cashback"}.

c) Detalles Importantes: Cada fila de la tabla se genera usando los datos del objeto Socio y su formato. El método calcularFechaVencimiento se invoca para calcular dinámicamente la fecha de vencimiento.

4. guardarTableModelEnArchivo

a) Propósito: Guarda los datos de un modelo de tabla (DefaultTableModel) en un archivo CSV. Parámetros: String rutaArchivo: Ruta del archivo donde se guardará la información. DefaultTableModel tableModel: Modelo de tabla con los datos a guardar. Detalles Importantes: Escritura: Los nombres de las columnas se escriben

como la primera línea del archivo. Cada fila del modelo se recorre y escribe en formato CSV. Control de errores: Maneja posibles errores de escritura con un bloque try-catch.

Estructuras de Control Relevantes:

1. Bloques try-catch: Se utilizan extensivamente para manejar excepciones en: Lectura de archivos (FileReader): Captura excepciones de tipo IOException y registra un mensaje de error detallado. Parsing de fechas (LocalDate.parse): Implementados en un ciclo, intentan diferentes formatos de fecha y continúan en caso de error.

2. Ciclo para Formatos de Fecha:

a) Propósito: Probar múltiples formatos de fecha para asegurar compatibilidad.

3. Resumen del Flujo de Trabajo

Lectura: El método leerSociosDesdeArchivo lee un archivo CSV y procesa cada línea para extraer datos, manejando errores en el formato o contenido. Procesamiento: Los datos se convierten a objetos Socio y se almacenan en una lista. Conversión: La lista de objetos Socio se transforma en un DefaultTableModel con columnas predefinidas mediante convertirSociosATableModel. Escritura: Los datos del DefaultTableModel se guardan nuevamente en un archivo CSV mediante guardarTableModelEnArchivo.

Reporte Técnico: Clase Socio

Nombre de la Clase: Socio

Esta clase pertenece al paquete Item y modela la información y comportamientos asociados a un socio, incluyendo atributos personales, estado de la membresía, fechas importantes, y funcionalidades adicionales como manejo de cashback y renovación.

Variables Globales Más Importantes:

1. nombre, direccion, telefono, rfc: Tipo: String

2. Descripción: Datos básicos del socio.

3. usuarioAdicional1, usuarioAdicional2: Tipo: String Descripción: Usuarios adicionales permitidos bajo la misma membresía.

4. fechaInicio, fechaRenovacion: Tipo: LocalDate Descripción: Fecha de inicio de la membresía y la próxima fecha de renovación. activa: Tipo: boolean Descripción: Indica si la membresía del socio está activa. tipoMembresia: Tipo: TipoMembresia (enumeración) Descripción: Tipo de membresía del socio, que puede ser NORMAL o PREMIUM. cashback: Tipo: double Descripción: Monto acumulado en cashback para socios con membresía PREMIUM.

Enumeración TipoMembresia

Propósito: Representa los tipos de membresía disponibles: NORMAL y PREMIUM. Atributo Asociado: costo: Representa el costo anual de la membresía. Método Importante: getCosto: Retorna el costo asociado al tipo de membresía.

Métodos Más Importantes

1. Constructores

a) Constructor Principal: Recibe todos los atributos necesarios para crear un objeto Socio. Inicializa fechas (fechaInicio y fechaRenovacion) y activa la membresía por defecto. Constructor Simplificado: Excluye los campos usuarioAdicional1, usuarioAdicional2 y permite usar la fecha actual como fecha de inicio.

2. convertirAMembresia

a) Propósito: Convierte una cadena de texto a un valor de la enumeración TipoMembresia. Parámetros: String entrada: Texto que representa el tipo de membresía. Retorno: Un valor de TipoMembresia correspondiente a la entrada. Detalles: Reconoce variantes comunes del texto (NORMAL, NORM, NRM, PREMIUM, PREM, PRM). Lanza una excepción IllegalArgumentException si la entrada es inválida.

3. isMembresiaActiva

a) Propósito: Verifica si la membresía sigue activa, comparando la fecha de renovación con la fecha actual. Retorno: Un valor booleano indicando si la membresía está activa.

4. agregarCashback

a) Propósito: Calcula y agrega cashback al socio si su membresía es PREMIUM. Parámetros: double montoCompra: Monto de la compra realizada. Detalles: El cashback se calcula como el 5% del monto de la compra.

5. usarCashback

a) Propósito: Permite al socio usar parte de su cashback acumulado. Parámetros: double monto: Monto a descontar del cashback. Retorno: Un valor booleano que indica si la operación fue exitosa (el cashback disponible es suficiente).

6. renovarMembresia

a) Propósito: Extiende la fecha de renovación de la membresía por un año y la activa nuevamente.

7. getDiasHastaRenovacion

a) Propósito: Calcula la cantidad de días restantes hasta la fecha de renovación de la membresía. Retorno: Un valor long que representa los días hasta la renovación.

8. toString

a) Propósito: Retorna una representación en texto del objeto Socio, incluyendo datos clave como el nombre, tipo de membresía, fecha de renovación y estado.

Estructuras de Control Relevantes:

1. Control de Inicialización en Constructores

a) Propósito: Garantizar valores predeterminados válidos.

2. Bloque switch en convertirAMembresia

a) Propósito: Validar y transformar las cadenas de texto en valores de la enumeración TipoMembresia.

Resumen del Flujo de Trabajo:

1. Creación del Objeto: Un objeto Socio se inicializa con datos básicos y, opcionalmente, datos adicionales.

2. Estado y Renovación: La clase permite verificar el estado de la membresía, calcular días restantes hasta la renovación y extender la membresía.

Manejo de Cashback: Incluye funcionalidades para acumular y utilizar cashback exclusivamente para socios con membresía PREMIUM.

Ping

Conclusión

El proyecto de un sistema de administración de comercios mayoristas, con capacidad de manejo de ofertas, cashback, membresías y un inventario a través de archivos es capaz de reflejar un dominio y aprendizaje de la Programación Orientada a Objetos (POO), paradigma abordado en el curso de *Programación II* a lo largo de este semestre. Resulta así en una interesante aplicación real de lo aprendido.