MANUAL TECNICO APP BANCARIA

LENGUAJE: JAVA

JDK : 17

BASE DE DATOS: MySQL

Clase InterfazGrafica

Descripción General

La clase InterfazGrafica extiende JFrame y representa la ventana principal de una aplicación gráfica que simula una interfaz para un banco. Esta interfaz contiene botones, una imagen de logotipo, y funcionalidades para abrir dos tipos de entradas: un archivo de texto y una entrada manual. Además, usa un JDesktopPane para manejar múltiples ventanas internas.

Componentes y Elementos Utilizados

1. JFrame

JFrame es el componente principal de Swing que representa la ventana de la aplicación. En esta clase, la ventana es maximizada automáticamente con setExtendedState(JFrame.MAXIMIZED_BOTH).

2. JDesktopPane

Un JDesktopPane es un panel especializado que permite contener múltiples ventanas internas (JInternalFrame), simulando un entorno de ventanas múltiples dentro de una única ventana externa.

En este caso, se utiliza para agregar diferentes componentes como botones, imágenes y ventanas internas que representan las interfaces de "Entrada Manual" y "Cargar Archivo".

3. Nimbus Look and Feel

A través de un bloque static, la clase intenta configurar el tema visual de la interfaz usando "Nimbus", que es un look and feel moderno y atractivo que puede aplicarse a aplicaciones Swing.

El código recorre los temas disponibles y selecciona "Nimbus" si está presente en el sistema.

```
static {
    try {
                                    for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
```

```
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

4. ImageIcon, JLabel, y Manejo de Imágenes

Se utiliza ImageIcon para cargar y escalar una imagen (banco.jpg), que es el logotipo de la aplicación. La imagen se ajusta a un tamaño de 250x250 píxeles usando el método getScaledInstance.

El JLabel actúa como contenedor para la imagen, y luego es añadido a un JPanel que se coloca dentro del JDesktopPane.

5. JPanel

Se utilizan varios JPanel para organizar los diferentes elementos de la interfaz. Los paneles son contenedores que permiten agrupar componentes.

Por ejemplo, se utiliza un JPanel llamado logo para colocar la imagen del banco, y otro JPanel llamado panelBotones para organizar los botones de la interfaz.

6. FlowLayout y BorderLayout

FlowLayout es el administrador de diseño que se utiliza para organizar los botones en una línea horizontal dentro del panelBotones.

BorderLayout se usa en el logo para centrar la imagen dentro del JPanel.

```
panelBotones.setLayout(new FlowLayout());
logo.setLayout(new BorderLayout());
```

7. JButton y ActionListener

Se crean dos botones, uno para cargar un archivo (botonEntradaTexto) y otro para la entrada manual (botonEntradaManual).

Cada botón tiene un ActionListener que responde a los eventos de clic. Estos listeners ejecutan métodos que abren nuevas ventanas (abrirEntradaTexto() y abrirEntradaManual()), los cuales crean instancias de clases de tipo JInternalFrame que muestran las interfaces correspondientes.

```
botonEntradaTexto.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        abrirEntradaTexto();
    }
});
```

8. JInternalFrame

Las clases EntradaArchivo y EntradaManual que se mencionan son ventanas internas (no incluidas en el código mostrado). Estas ventanas son agregadas al JDesktopPane y luego se maximizan usando el método setMaximum(true) dentro de maximizarVentana().

9. JOptionPane

Se utiliza un JOptionPane para mostrar un mensaje de error en caso de que la imagen del banco no se pueda cargar.

```
JOptionPane.showMessageDialog(this, "Error al cargar la imagen.", "Error", JOptionPane.ERROR_MESSAGE);
```

Funcionalidad de los Métodos
1. Constructor InterfazGrafica()
El constructor inicializa la ventana principal (JFrame) y establece su título, tamaño y comportamiento de cierre. También organiza los paneles y los botones, y carga la imagen del banco en la interfaz. Finalmente, se hace visible la ventana con setVisible(true).
2. Métodos abrirEntradaTexto() y abrirEntradaManual()
Estos métodos crean instancias de EntradaArchivo y EntradaManual respectivamente, que son clases de ventanas internas. Ambas ventanas se agregan al JDesktopPane y se traen al frente con toFront(). Además, se maximiza su tamaño usando el método maximizarVentana().
3. Método maximizarVentana(JInternalFrame frame)
Se encarga de maximizar y traer al frente cualquier ventana interna (JInternalFrame) que se pase como parámetro. Este método también maneja excepciones cuando intenta maximizar la ventana.
Elementos Técnicos Propios
ArrayList (no se utiliza directamente en esta clase, pero es común en Swing)

Un ArrayList es una estructura de datos utilizada para almacenar elementos de forma dinámica, y es útil cuando el número de elementos puede cambiar en tiempo de ejecución.
JPanel

Un contenedor flexible para organizar componentes gráficos. En este caso, se usa para organizar tanto la imagen como los botones.
ActionEvent

Un tipo de evento que ocurre cuando se realiza una acción, como hacer clic en un botón. Cada ActionEvent es manejado por un ActionListener, que ejecuta el código asociado.
Consideraciones Finales
Este código utiliza la librería gráfica de Swing para crear una interfaz gráfica de usuario rica con manejo de imágenes, múltiples ventanas internas, y botones interactivos.

Clase EntradaManual
1. Propósito de la Clase:
La clase EntradaManual extiende de JInternalFrame y tiene como finalidad actuar como la ventana principal de una aplicación que gestiona diferentes funcionalidades relacionadas con tarjetas, tales como la solicitud, autorización, cancelación, y generación de reportes. A través de esta clase, se abren y gestionan múltiples sub-ventanas dentro de un área de trabajo principal (JDesktopPane), permitiendo una navegación organizada y controlada dentro de la interfaz de usuario.

2. Herramientas Utilizadas en la Clase:
2.1 JInternalFrame
Un JInternalFrame es una ventana que puede ser anidada dentro de un JDesktopPane. Es similar a un marco (frame) independiente, pero que reside dentro de una ventana principal.
Para qué se usa: EntradaManual hereda de JInternalFrame para permitir que se comporten como una ventana interna en la aplicación principal. Todas las sub-opciones (por ejemplo, la ventana de solicitudes, autorizaciones, etc.) también se crean como JInternalFrame para ser gestionadas desde la ventana principal.

2.2 JDesktopPane

Un JDesktopPane es un contenedor que permite organizar múltiples ventanas internas (JInternalFrame) dentro de una aplicación.

Para qué se usa: En EntradaManual, el JDesktopPane es el componente que organiza las diferentes sub-ventanas que la clase abre. Cada sub-opción que el usuario selecciona del menú (solicitudes, autorizaciones, reportes, etc.) se abre como una ventana interna en este contenedor.

2.3 JMenuBar

Un JMenuBar es un componente que permite crear un menú dentro de una aplicación gráfica. Cada menú puede contener elementos o submenús.

Para qué se usa: En EntradaManual, el JMenuBar gestiona las opciones principales de la aplicación, como abrir ventanas de solicitudes, autorizaciones, o reportes. Los usuarios interactúan con el menú para navegar por la aplicación y acceder a las diferentes funcionalidades.

2.4 JMenu

Un JMenu es un componente de la barra de menús (JMenuBar) que contiene opciones o submenús que el usuario puede seleccionar.

Para qué se usa: En EntradaManual, los JMenu definen las categorías dentro del menú, como "Solicitudes", "Tarjetas", "Reportes", etc. Cada categoría contiene opciones relacionadas con la funcionalidad específica.

2.5 JMenuItem

Un JMenuItem es una opción específica dentro de un menú. Al seleccionar un JMenuItem, se ejecuta una acción definida.

Para qué se usa: Cada sub-opción, como "Solicitar Tarjeta", "Autorizar Tarjeta", "Consultar Tarjeta", entre otras, está definida como un JMenuItem en el JMenuBar. Cuando el usuario selecciona una opción, se abre la ventana correspondiente a esa funcionalidad.

2.6 JTextField

Un JTextField es un campo de entrada de texto que permite al usuario ingresar una línea de texto.

Para qué se usa: En algunas sub-ventanas internas que se abren desde EntradaManual, como "Solicitar Tarjeta", los JTextField permiten al usuario ingresar datos (nombre, número de tarjeta, etc.). Aunque no están directamente en EntradaManual, estas sub-opciones dependen de su correcta gestión.

2.7 JOptionPane

Un JOptionPane es un componente que muestra cuadros de diálogo estándar, como mensajes de advertencia, errores o confirmaciones.

Para qué se usa: En algunas sub-opciones abiertas desde EntradaManual, se usa para mostrar mensajes de confirmación o error al usuario, como cuando una operación ha fallado o se ha realizado con éxito.

2.8 ActionListener

Un ActionListener es una interfaz que define un comportamiento que debe ejecutarse cuando ocurre una acción, como hacer clic en un botón o seleccionar un elemento de menú.

Para qué se usa: En EntradaManual, los ActionListeners están asociados a cada JMenuItem para que, cuando el usuario seleccione una opción del menú, se ejecute el método que abre la sub-ventana correspondiente (por ejemplo, abrir la ventana de solicitudes o reportes).

2.9 JTable

Una JTable es un componente de tabla que permite mostrar datos organizados en filas y columnas.
Para qué se usa: En algunas sub-ventanas abiertas desde EntradaManual, las JTable se utilizan para mostrar listas de tarjetas, solicitudes, o reportes al usuario. EntradaManual facilita la apertura de estas ventanas, y la JTable es una herramienta crucial en las mismas.
2.10 SQLException
Una SQLException es una excepción que se lanza cuando ocurre un error en una operación de base de datos.
Para qué se usa: En algunos métodos de EntradaManual, como al abrir sub-opciones que interactúan con la base de datos, se puede lanzar esta excepción si hay problemas de conexión o consulta. Se maneja para evitar que el programa falle.
3. Métodos Principales de la Clase:
abrirSubOpcion1()

Descripción: Este método abre la ventana interna SubOpcion1Solicitud dentro del desktopPane. Facilita la gestión de solicitudes de tarjeta.
Elementos involucrados:
JInternalFrame: Para abrir la ventana.
desktopPane: Donde se añade la nueva ventana.
maximizarVentana(JInternalFrame frame): Método que maximiza la ventana interna.
abrirSubOpcion2()

Descripción: Este método abre la ventana SubOpcion2AutorizacionDeTarjeta, que permite autorizar tarjetas.
Elementos involucrados:
JInternalFrame: Para abrir la ventana.
desktopPane: Para añadir la ventana.
abrirSubOpcion3()

Descripción: Abre la ventana de consulta de tarjetas, permitiendo al usuario verificar el estado de una tarjeta existente.
Elementos involucrados:
JInternalFrame, desktopPane.
abrirSubOpcionEstadoDeCuenta()

Descripción: Abre la ventana que muestra el estado de cuenta de las tarjetas.
Elementos involucrados:
JInternalFrame, desktopPane, JTable.
4. Finalidad General de la Clase:
La clase EntradaManual funciona como el marco principal para gestionar y organizar las diferentes funcionalidades de la aplicación. Facilita la navegación a través de las diversas operaciones del sistema, como solicitudes, autorizaciones, cancelaciones, y reportes. Además, asegura una interfaz gráfica interactiva donde se pueden gestionar múltiples sub-ventanas relacionadas con las operaciones de tarjetas de crédito.

Clase EntradaArchivo

1. Propósito de la Clase:

La clase EntradaArchivo extiende de JInternalFrame y proporciona una interfaz gráfica para cargar y procesar archivos de texto. Permite al usuario seleccionar un archivo, definir una ruta para generar informes, establecer un intervalo entre la ejecución de las líneas del archivo, y visualizar el proceso de carga en un área de texto. La clase ejecuta las operaciones de lectura de archivos en un hilo separado para evitar bloquear la interfaz gráfica.

2. Herramientas Utilizadas en la Clase:

2.1 JInternalFrame

JInternalFrame es un componente de ventana interna que permite a los usuarios abrir múltiples ventanas dentro de una ventana principal.

Para qué se usa: EntradaArchivo hereda de JInternalFrame para comportarse como una ventana interna dentro de la aplicación principal. Permite al usuario interactuar con la funcionalidad de carga de archivos sin interferir con otras ventanas.

2.2 JButton

Un JButton es un botón en la interfaz gráfica que el usuario puede presionar para realizar una acción.

Para qué se usa: La clase utiliza tres botones:

seleccionarArchivo: Para abrir un cuadro de diálogo que permita seleccionar un archivo de texto.

seleccionarRuta: Para abrir un cuadro de diálogo que permita seleccionar una ruta de destino para los informes.

cargarArchivo: Para iniciar el proceso de lectura y carga del archivo seleccionado.

2.3 JTextField

Un JTextField es un campo de entrada de una sola línea que permite al usuario ingresar o visualizar texto.

Para qué se usa:

rutaArchivo: Muestra la ruta del archivo seleccionado.

rutaReporte: Muestra la ruta seleccionada para guardar los informes.

2.4 JTextArea

Un JTextArea es un área de texto que permite la entrada o visualización de múltiples líneas de texto.

Para qué se usa: La clase utiliza una JTextArea llamada logArea para mostrar mensajes de estado, como el progreso de la lectura de líneas del archivo, errores, y mensajes de confirmación. Este componente no es editable por el usuario.

2.5 JSpinner

Un JSpinner es un componente que permite seleccionar un valor numérico incrementándolo o disminuyéndolo mediante flechas.

Para qué se usa: El JSpinner intervaloSpinner permite al usuario especificar el intervalo en milisegundos entre la ejecución de cada línea del archivo cargado.

2.6 GridBagLayout

Un GridBagLayout es un administrador de diseño flexible que organiza los componentes en una cuadrícula, permitiendo control detallado sobre el tamaño y posición de cada componente.

Para qué se usa: La clase utiliza GridBagLayout para organizar los componentes gráficos (botones, campos de texto, etiquetas) de manera que se adapten de manera flexible a los cambios de tamaño de la ventana.

2.7 JFileChooser

Un JFileChooser es un componente que permite a los usuarios navegar por el sistema de archivos y seleccionar archivos o directorios.

Para qué se usa:

Para seleccionar archivos: JFileChooser se usa en el método seleccionarArchivoAction() para que el usuario pueda seleccionar un archivo de texto (.txt) que se va a cargar.

Para seleccionar directorios: JFileChooser se usa en el método seleccionarRutaAction() para permitir al usuario seleccionar un directorio en el cual guardar los informes.

2.8 JLabel

Un JLabel es un componente que muestra texto o una imagen estática en la interfaz.

Para qué se usa: En EntradaArchivo, se utiliza un JLabel llamado intervaloLabel para etiquetar el campo de intervalo que define el tiempo en milisegundos entre la ejecución de líneas del archivo.

2.9 JScrollPane

Un JScrollPane es un contenedor que añade barras de desplazamiento a otro componente cuando este excede el tamaño del área visible.

Para qué se usa: La clase envuelve la JTextArea logArea en un JScrollPane para permitir que el usuario desplace verticalmente el contenido del área de texto a medida que se añaden mensajes de estado.

2.10 ActionListener

ActionListener es una interfaz que define un comportamiento a ejecutar cuando se produce una acción, como hacer clic en un botón.

Para qué se usa: En la clase, los JButton (seleccionarArchivo, seleccionarRuta, cargarArchivo) tienen asociados ActionListener que ejecutan acciones específicas cuando el usuario hace clic en los botones.

2.11 Thread

Thread es una clase que permite la ejecución de tareas en segundo plano sin bloquear el hilo principal (la interfaz gráfica).

Para qué se usa: En el método cargarArchivoAction(), se usa un Thread para leer el archivo y procesar cada línea, permitiendo que la interfaz gráfica siga siendo interactiva mientras el archivo se carga.

2.12 BufferedReader

BufferedReader es una clase que permite leer texto de un archivo de manera eficiente.

Para qué se usa: En el método cargarArchivoAction(), el BufferedReader se utiliza para leer el archivo línea por línea y procesar su contenido.

2.13 File

File es una clase que representa archivos y directorios en el sistema de archivos.

Para qué se usa: En el método cargarArchivoAction(), la clase File se utiliza para obtener la ruta del archivo que se va a procesar.

2.14 operadorDeArchivo

Presumiblemente, operadorDeArchivo es una clase personalizada encargada de ejecutar acciones sobre cada línea del archivo cargado.

Para qué se usa: En el método cargarArchivoAction(), se instancia operadorDeArchivo para procesar cada línea leída del archivo, ejecutando la lógica definida en esa clase.

3. Métodos Principales de la Clase:

seleccionarArchivoAction()

Descripción: Abre un JFileChooser para seleccionar un archivo de texto (.txt). La ruta seleccionada se muestra en el campo rutaArchivo.
Elementos involucrados: JFileChooser, JTextField.
seleccionarRutaAction()

Descripción: Abre un JFileChooser en modo de selección de directorios para elegir la ruta donde se guardará el informe. La ruta seleccionada se muestra en rutaReporte.
Elementos involucrados: JFileChooser, JTextField.
cargarArchivoAction()

Descripción: Inicia un proceso en segundo plano que lee el archivo seleccionado línea por línea, mostrando el contenido en logArea y procesando cada línea mediante operadorDeArchivo. El intervalo entre la ejecución de cada línea es controlado por intervaloSpinner.
Elementos involucrados: BufferedReader, Thread, JTextArea, JSpinner, operadorDeArchivo.
4. Finalidad General de la Clase:
La clase EntradaArchivo proporciona una interfaz gráfica que permite a los usuarios cargar y procesar archivos de texto. Facilita la selección de archivos y directorios, la configuración de intervalos de procesamiento, y el registro de las acciones realizadas en un área de texto. Todo el proceso de carga y ejecución de archivos se maneja en un hilo separado para asegurar que la interfaz gráfica continúe respondiendo durante el procesamiento.

Clase SubOpcion1Solicitud
Descripción General
La clase SubOpcion1Solicitud es una subventana (JInternalFrame) que gestiona la interfaz para el registro de nuevas solicitudes de tarjeta de crédito. Esta clase está diseñada para permitir al usuario ingresar los datos relacionados con una solicitud, como el tipo de tarjeta, nombre completo, salario y dirección, y luego validar y guardar dicha información en una base de datos.

Atributos
campoNumeroSolicitud (JTextField): Muestra el número de solicitud generado automáticamente.
tipoCombo (JComboBox<String>): Selección del tipo de tarjeta (Nacional, Regional, Internacional).
campoNombreCompleto (JTextField): Campo de texto para ingresar el nombre completo del solicitante.
campoSalario (JTextField): Campo de texto para ingresar el salario del solicitante.
campoDireccion (JTextField): Campo de texto para ingresar la dirección del solicitante.
Constructor

public SubOpcion1Solicitud() throws SQLException

Inicializa los componentes de la interfaz gráfica y organiza los elementos visuales utilizando GridBagLayout.
El número de solicitud es calculado y mostrado automáticamente usando el método Solicitud.obtenerUltimoNumeroSolicitud().
Métodos
validarCampos()
Este método es llamado al hacer clic en el botón "Enviar". Realiza la validación de los campos para asegurarse de que no estén vacíos antes de procesar la solicitud. Si algún campo está vacío, muestra un mensaje de error.

private void validarCampos()
Si los campos son válidos, crea una instancia de la clase Solicitud, asigna los valores de los campos a los atributos de la solicitud, y guarda la solicitud en la base de datos llamando a solicitud.guardarEnBaseDeDatos().
limpiarCampos()
Después de que una solicitud ha sido enviada y guardada en la base de datos, este método limpia todos los campos para preparar el formulario para una nueva entrada.

private void limpiarCampos() throws SQLException
Actualiza el número de solicitud para la siguiente entrada usando el método Solicitud.obtenerUltimoNumeroSolicitud().
Interacciones con la Base de Datos
Solicitud.guardarEnBaseDeDatos(): Este método de la clase Solicitud se utiliza para insertar los datos de la nueva solicitud en la base de datos.
Validación
La clase implementa un mecanismo de validación básica que asegura que todos los campos estén llenos antes de intentar guardar los datos en la base de datos. Si alguno de los campos es inválido, se despliega un mensaje de error mediante un JOptionPane.

Elementos de la Interfaz Gráfica
Formulario (JPanel): Contiene todos los campos de entrada y el botón de envío, organizado con GridBagLayout.
Campos de Texto (JTextField): Para el ingreso de información como el número de solicitud, nombre, salario y dirección.
ComboBox (JComboBox<String>): Para seleccionar el tipo de tarjeta.
Botón Enviar (JButton): Ejecuta la validación de los campos y el envío de los datos a la base de datos.
Mantenimiento y Extensibilidad
La clase está diseñada de manera modular, lo que facilita la extensión y el mantenimiento:

Validación de campos: Actualmente, la validación es básica. Puede mejorarse incluyendo validaciones más específicas para el salario o la dirección.
Interfaz Base de Datos: Si en el futuro se necesitan cambios en la estructura de la base de datos, solo se debe modificar la clase Solicitud en la parte que maneja la persistencia de datos.

Resumen
La clase SubOpcion1Solicitud implementa una interfaz amigable para la creación de solicitudes de tarjetas de crédito. Proporciona una manera eficiente de validar, procesar y guardar solicitudes, con interacciones simples con la base de datos y la posibilidad de mejorar o ampliar sus funcionalidades en el futuro.

Clase SubOpcion2AutorizacionDeTarjeta
Descripción General
La clase SubOpcion2AutorizacionDeTarjeta es una subventana (JInternalFrame) encargada de gestionar el proceso de autorización de tarjetas de crédito basadas en solicitudes previamente ingresadas. La clase ofrece la funcionalidad para autorizar una tarjeta o cancelar una solicitud, mediante la interacción con la base de datos.

Atributos
numeroSolicitud (JTextField): Campo de texto donde el usuario ingresa el número de la solicitud a autorizar o cancelar.
Constructor

public SubOpcion2AutorizacionDeTarjeta()
El constructor configura la interfaz gráfica de la clase, inicializando la ventana con las siguientes características:

Título: "Autorización de tarjetas".
Tamaño: 500x400 píxeles.
Configuraciones comunes de un JInternalFrame, como ser cerrable, maximizable, iconificable, y redimensionable.
El formulario utiliza un JPanel con un layout de GridBagLayout, lo que permite una distribución flexible de los componentes. Este layout es especialmente útil para organizar elementos en una cuadrícula de manera dinámica y ajustable.

Métodos
validarCampos()
Este método es ejecutado cuando el usuario hace clic en el botón "Autorizar Tarjeta". Su función es validar que el número de solicitud no esté vacío antes de intentar procesar la autorización de una tarjeta.

protected void validarCampos()
Verifica si el campo numeroSolicitud está vacío. Si lo está, muestra un mensaje de error.
Si el campo contiene un valor, intenta convertirlo a un número entero y proceder a la autorización de la tarjeta:
Se crea una instancia de la clase Autorizacion con el número de solicitud.
Se llama al método autorizar() de Autorizacion, que procesa la autorización y devuelve una instancia de Tarjeta.
Si la tarjeta es autorizada, se muestra un mensaje de éxito con el número de la tarjeta generada.

Si ocurre algún error durante el proceso (como un problema en la autorización), se captura la excepción y se muestra un mensaje de error.
cancelarSolicitud()
Este método es ejecutado cuando el usuario hace clic en el botón "Cancelar Solicitud". Permite al usuario cancelar una solicitud en estado pendiente.

protected void cancelarSolicitud()
Similar a validarCampos(), se verifica que el número de solicitud no esté vacío.
Si el número es válido, intenta buscar la solicitud en la base de datos:
Usa la clase Autorizacion para obtener la solicitud asociada.
Verifica el estado actual de la solicitud. Si la solicitud ya ha sido aprobada o rechazada, no se puede cancelar y muestra un mensaje de error.
Si el estado es diferente, actualiza el estado de la solicitud a "RECHAZADA" y guarda el motivo en la base de datos.
Si ocurre algún error en el proceso de cancelación, se captura la excepción y se muestra un mensaje de error.
Interacciones con la Base de Datos
Autorización de Tarjeta: Utiliza la clase Autorizacion para autorizar una tarjeta basada en una solicitud válida. Al autorizar, también se genera un número de tarjeta.
Cancelación de Solicitud: También utiliza la clase Autorizacion para acceder a la solicitud en la base de datos y, si es posible, actualizar su estado a "RECHAZADA" mediante una conexión con la base de datos (ConexionMySQL).
Componentes de Interfaz Gráfica
JPanel con GridBagLayout: Organiza los componentes de manera estructurada y permite distribuir el formulario de manera flexible.

JPanel autorizacion = new JPanel(new GridBagLayout());
JTextField (numeroSolicitud): Permite la entrada del número de solicitud que se va a procesar. Se encuentra en la primera fila del JPanel.

JButton (autorizar): Botón que activa la funcionalidad de autorización de tarjetas. Al hacer clic, ejecuta el método validarCampos().

JButton autorizar = new JButton("Autorizar Tarjeta");
autorizar.addActionListener(e -> validarCampos());
JButton (cancelar): Botón que activa la funcionalidad para cancelar una solicitud. Al hacer clic, ejecuta el método cancelarSolicitud().

JButton cancelar = new JButton("Cancelar Solicitud");
cancelar.addActionListener(e -> cancelarSolicitud());
Validación y Manejo de Errores
La clase implementa un sistema de validación básico para asegurarse de que el número de solicitud sea ingresado antes de realizar cualquier acción. Los errores durante la autorización o cancelación de solicitudes son capturados y gestionados con un JOptionPane, mostrando mensajes informativos al usuario.

Uso de JInternalFrame
El uso de JInternalFrame permite que esta subopción de autorización se maneje dentro de una aplicación de escritorio más grande con un contenedor de múltiples ventanas internas, comúnmente usada en aplicaciones empresariales con interfaces complejas.

Posibles Mejoras y Extensibilidad
Mejor validación: Actualmente, el sistema solo valida si el campo de número de solicitud está vacío. Se podría mejorar para verificar también que el número de solicitud exista antes de proceder.
Manejo avanzado de errores: Los mensajes de error pueden ser más detallados, informando sobre los posibles problemas que ocurrieron durante la interacción con la base de datos.
Optimización de consultas: El manejo de las consultas a la base de datos a través de ConexionMySQL puede ser optimizado para reducir la latencia y mejorar el rendimiento en escenarios con gran volumen de solicitudes.
Resumen
La clase SubOpcion2AutorizacionDeTarjeta proporciona una interfaz intuitiva para gestionar la autorización o cancelación de solicitudes de tarjetas de crédito. El diseño modular de la clase facilita su mantenimiento y mejora futura, mientras que su interacción con la base de datos está centralizada en las clases Autorizacion y Solicitud, lo que permite una fácil modificación en caso de cambios en el modelo de datos.

Clase SubOpcion4Movimientos
Descripción General
La clase SubOpcion4Movimientos extiende JInternalFrame de Swing y representa una interfaz gráfica para registrar movimientos asociados a una tarjeta. El formulario incluye campos para el número de tarjeta, fecha del movimiento, tipo de movimiento (CARGO o ABONO), descripción, código de establecimiento y monto de la operación. También incluye un botón para registrar el movimiento, que valida los campos ingresados por el usuario.

Atributos
numeroTarjeta (JTextField): Campo de texto que permite al usuario ingresar el número de tarjeta.
fechaOperacion (JFormattedTextField): Campo formateado para ingresar la fecha de operación en formato dd/MM/yyyy.
tipoMovimiento (JComboBox<String>): Desplegable para seleccionar el tipo de movimiento (CARGO o ABONO).
descripcion (JTextArea): Área de texto para la descripción del movimiento.
codigoEstablecimiento (JTextField): Campo de texto para ingresar el código del establecimiento donde se realizó la operación.
monto (JTextField): Campo de texto para ingresar el monto de la operación.

Constructor
El constructor de la clase configura los atributos visuales y organiza los componentes en un JPanel utilizando GridBagLayout para estructurar la disposición de los campos en filas y columnas.

public SubOpcion4Movimientos() { ... }
Este método:

Configura las propiedades básicas de la ventana (título, tamaño, etc.).
Inicializa los componentes de la interfaz gráfica.
Agrega los componentes al panel usando GridBagLayout y luego los posiciona dentro de la ventana.
Componentes y Herramientas Usadas
JInternalFrame: Esta clase es utilizada para crear una ventana interna dentro de una aplicación que usa un JDesktopPane. Es útil para aplicaciones de múltiples ventanas que funcionan dentro de un marco de ventana principal.

GridBagLayout: Proporciona una disposición flexible para los componentes, permitiendo que se posicionen en una cuadrícula con control sobre cómo se expande cada componente. Los parámetros importantes usados aquí son:

GridBagConstraints.gridx y GridBagConstraints.gridy: Definen la posición del componente en la cuadrícula.
GridBagConstraints.fill: Controla cómo se debe expandir el componente dentro de su celda, en este caso, horizontalmente.
GridBagConstraints.insets: Controla el espaciado alrededor de cada componente.
JLabel: Muestra texto estático para identificar los campos del formulario.

JTextField: Permite la entrada de texto por parte del usuario. Aquí se usa para ingresar el número de tarjeta, el código del establecimiento y el monto.

JFormattedTextField: Campo de texto formateado, en este caso se usa para ingresar la fecha en formato dd/MM/yyyy, utilizando SimpleDateFormat.

JComboBox: Desplegable que permite al usuario elegir entre "CARGO" y "ABONO" como tipo de movimiento.

JTextArea: Área de texto que permite ingresar descripciones de múltiples líneas. En este caso, está utilizada para la descripción del movimiento.

JButton: Botones interactivos que permiten ejecutar las acciones de registro de movimientos, validación de campos, y mostrar mensajes de retroalimentación.

JOptionPane: Muestra mensajes emergentes para informar al usuario de errores o resultados de las acciones, como la validación de los campos o el éxito del registro.

Métodos

validarCampos(): Este método es llamado cuando se presiona el botón de "Registrar Movimiento". Se encarga de validar que todos los campos del formulario estén completos. Si falta algún campo, muestra un mensaje de error utilizando JOptionPane. Si los campos son válidos, intenta crear un nuevo objeto de la clase Movimiento con los datos ingresados y lo guarda en la base de datos.

protected void validarCampos() { ... }
El método:

Valida que ningún campo esté vacío.
Convierte y valida el valor del campo monto a tipo double.
Crea una instancia de la clase Movimiento y llama al método guardarEnBaseDeDatos() para registrar el movimiento en la base de datos.
Maneja excepciones mostrando mensajes de error si ocurre algún problema durante el registro del movimiento.
Manejo de Excepciones
El método validarCampos() atrapa las excepciones generales mediante try-catch, lo que permite que el programa no se detenga si ocurre algún error durante la conversión de tipos o el guardado en la base de datos. Los mensajes de error se muestran usando JOptionPane.

Ejemplo de Uso
Al crear una instancia de SubOpcion4Movimientos, se abrirá una ventana con un formulario donde el usuario podrá ingresar la información de un movimiento y registrarlo.

SubOpcion4Movimientos ventanaMovimientos = new SubOpcion4Movimientos();
ventanaMovimientos.setVisible(true);
Conclusión
Esta clase proporciona una interfaz gráfica sencilla pero efectiva para registrar movimientos de tarjetas. Hace uso de varios componentes clave de JSwing, como JTextField, JComboBox, JFormattedTextField y JTextArea, junto con la disposición flexible que permite GridBagLayout. Además, implementa validación básica de datos y manejo de excepciones para garantizar una buena experiencia de usuario al momento de registrar un movimiento.

clase SubOpcion5CancelacionDeTarjeta
1. Descripción General
La clase SubOpcion5CancelacionDeTarjeta extiende de JInternalFrame y se utiliza para implementar la funcionalidad de cancelación de tarjetas en una aplicación de escritorio Java. Esta ventana permite al usuario consultar el estado de una tarjeta ingresando su número, y posteriormente, cancelar dicha tarjeta si se desea. Los elementos de la interfaz gráfica están organizados en un formulario usando GridBagLayout y son controlados mediante acciones asociadas a botones.

2. Atributos
numeroTarjeta (JTextField): Campo de texto donde el usuario ingresa el número de la tarjeta que desea consultar o cancelar.

infoTarjeta (JLabel): Etiqueta que muestra la información detallada de la tarjeta consultada, incluyendo número, tipo, cliente, dirección, y saldo total.
confirmarCancelacion (JButton): Botón para confirmar la cancelación de la tarjeta una vez que ha sido consultada. Está oculto inicialmente y se muestra solo después de realizar la consulta.
3. Constructor SubOpcion5CancelacionDeTarjeta()
El constructor configura la ventana de cancelación de tarjetas con las siguientes características: Título: "Cancelación de Tarjetas".
Dimensiones: 500 x 400 píxeles.
Propiedades: Habilita las opciones de cerrar, maximizar, minimizar y redimensionar la ventana.
Panel: Se utiliza un JPanel con GridBagLayout para colocar los componentes del formulario.
Botones: Se incluyen botones para consultar la tarjeta y para confirmar su cancelación.
4. Métodos
4.1. validarCampos()
Este método se ejecuta al hacer clic en el botón "Consultar Tarjeta". Valida si el campo de texto numeroTarjeta está vacío. Si lo está, muestra un mensaje de error. De lo contrario:

Intenta consultar la tarjeta mediante la clase EstadoDeCuenta pasándole el número de tarjeta.
Si la consulta es exitosa, llama al método mostrarInformacionTarjeta() para desplegar la información de la tarjeta en la interfaz.
Si hay errores, como una tarjeta no válida o problemas con la base de datos, se captura y muestra el mensaje de error correspondiente.
4.2. mostrarInformacionTarjeta(EstadoDeCuenta estadoDeCuenta)
Este método recibe un objeto de tipo EstadoDeCuenta y genera una cadena de HTML que muestra la información de la tarjeta consultada, incluyendo el número de tarjeta, tipo, cliente, dirección, y saldo total. La información se asigna al componente infoTarjeta para ser mostrada en pantalla.

4.3. confirmarCancelacion()
Este método se ejecuta cuando el usuario hace clic en el botón "Confirmar Cancelación". Intenta cancelar la tarjeta utilizando la clase Cancelacion. Si la operación es exitosa, muestra un mensaje de confirmación. Si ocurre un error, se maneja mediante un mensaje de error que se muestra al usuario.

5. Manejo de Excepciones
IllegalArgumentException: Se captura cuando se ingresa un número de tarjeta inválido. El mensaje de error asociado se muestra al usuario.
SQLException: Se captura cuando hay un problema con la consulta o cancelación en la base de datos. El mensaje del error se muestra al usuario.
6. Interacción con otras clases
EstadoDeCuenta: Clase utilizada para obtener el estado de cuenta de una tarjeta, como su número, tipo, nombre del cliente, dirección, y saldo total. Se invoca en el método validarCampos().
Cancelacion: Clase encargada de manejar la lógica para cancelar una tarjeta. Se invoca en el método confirmarCancelacion().
7. Interfaz de Usuario

La interfaz de usuario consta de:

Un campo de texto para ingresar el número de tarjeta.
Un botón para consultar la tarjeta.
Un área de información que muestra los detalles de la tarjeta consultada.
Un botón de confirmación para cancelar la tarjeta (visible solo después de una consulta exitosa).

reportesSubOpcion1EstadoDeCuenta
Descripción General
La clase reportesSubOpcion1EstadoDeCuenta es una subventana interna (JInternalFrame) que permite a los usuarios consultar el estado de cuenta de tarjetas mediante filtros específicos. La interfaz gráfica incluye campos para ingresar filtros, botones para aplicar estos filtros, y un área para mostrar los resultados de la búsqueda.

Atributos
private JPanel mainPanel: Panel principal que contiene el contenido dinámico de la ventana.
private JTextField filtro1: Campo de texto para ingresar el número de tarjeta como filtro.
private JTextField filtro3: Campo de texto para ingresar el saldo mayor a como filtro.
private JTextField filtro4: Campo de texto para ingresar los intereses mayores a como filtro.
private JComboBox<String> filterComboBox: Combo box para seleccionar el tipo de tarjeta.
private JButton botonFiltrar: Botón para aplicar los filtros seleccionados.
private JButton botonFiltrarTodo: Botón para mostrar todos los resultados sin aplicar filtros.
Constructor
public reportesSubOpcion1EstadoDeCuenta(): Configura la ventana con un título, tamaño y opciones de cierre/máximo/minimización. Inicializa los componentes gráficos, configura la disposición de los filtros, y agrega los componentes al panel principal.
Métodos
private void aplicarFiltros(): Aplica los filtros especificados por el usuario. Recoge los datos de los campos de texto y el combo box, y realiza una consulta para filtrar los estados de cuenta. Luego, actualiza el panel principal con los resultados de la búsqueda.

Manejo de Excepciones:
SQLException: Muestra un mensaje de error si ocurre un problema al consultar los datos.
private void mostrarTodo(): Muestra todos los estados de cuenta activos sin aplicar filtros. Actualiza el panel principal con la información de todas las tarjetas.

Manejo de Excepciones:
SQLException: Muestra un mensaje de error si ocurre un problema al obtener los datos.
Diseño de la Interfaz
Panel Principal:

Disposición: BoxLayout vertical para organizar los componentes.
ScrollPane: Permite desplazarse por los resultados cuando hay muchos datos.
Panel de Filtros:

Componentes:
Botón "Filtrar": Aplica los filtros especificados.

Campo de Texto para Número de Tarjeta: Permite ingresar el número de tarjeta como filtro.
ComboBox para Tipo de Tarjeta: Permite seleccionar el tipo de tarjeta como filtro.
Campo de Texto para Saldo Mayor a: Permite ingresar el saldo mayor a como filtro.
Campo de Texto para Intereses Mayores a: Permite ingresar los intereses mayores a como filtro.
Botón "Mostrar Todo": Muestra todos los estados de cuenta sin aplicar filtros.
Panel de Resultados:

Panel para Cada Tarjeta:
Información de la Tarjeta: Muestra número de tarjeta, tipo, nombre y dirección del cliente.
Tabla de Movimientos: Muestra los movimientos asociados a la tarjeta en una tabla con columnas para fecha, tipo de movimiento, descripción, establecimiento y monto.
Totales: Muestra el monto total, los intereses y el saldo total.
Uso
Aplicar Filtros:

El usuario ingresa los criterios de búsqueda en los campos correspondientes y hace clic en el botón "Filtrar". La ventana muestra solo los resultados que cumplen con los filtros especificados.
Mostrar Todos los Resultados:

El usuario hace clic en el botón "Mostrar Todo" para ver todos los estados de cuenta activos sin aplicar filtros.
Ejemplos de Mensajes de Error
"Por favor, ingrese un filtro.": Muestra si el usuario intenta aplicar filtros sin ingresar ningún criterio.
"No hay resultados para el reporte.": Muestra si la consulta no devuelve resultados.
"Error al obtener los datos: [mensaje de error]": Muestra si ocurre un error al obtener datos de la base de datos.
Este manual proporciona una visión clara y detallada del funcionamiento y la estructura de la clase reportesSubOpcion1EstadoDeCuenta, ayudando en su mantenimiento y desarrollo futuro.

reportesSubOpcion2ListadoDeTarjetas
Descripción General
La clase reportesSubOpcion2ListadoDeTarjetas es un componente de la interfaz gráfica de usuario (GUI) en una aplicación basada en Java Swing. Esta clase extiende JInternalFrame y proporciona una vista para listar tarjetas de crédito basadas en varios filtros. Permite al usuario consultar y filtrar tarjetas de crédito según distintos criterios y visualizar los resultados en una tabla.

Estructura de la Clase
Atributos
JTable table: Tabla que muestra el listado de tarjetas.
DefaultTableModel listadoTarjetas: Modelo de la tabla para mostrar los datos de las tarjetas.

JTextField filtro2: Campo de texto para filtrar por nombre de cliente.
JTextField filtro3: Campo de texto para filtrar por límite de tarjeta.
JTextField filtroFechaInicio: Campo de texto para ingresar la fecha de inicio del filtro.
JTextField filtroFechaFin: Campo de texto para ingresar la fecha de fin del filtro.
JComboBox<String> filtro1: ComboBox para seleccionar el tipo de tarjeta.
JComboBox<String> filtro5: ComboBox para seleccionar el estado de la tarjeta.
JButton consultar: Botón para aplicar los filtros y consultar los datos.

Métodos
aplicarFiltros()
Este método aplica los filtros especificados por el usuario y actualiza la tabla con los resultados obtenidos.

Obteniendo Filtros

Obtiene el tipo, nombre, límite, fechas y estado desde los campos de filtro.
Valida el límite para asegurarse de que es un número válido.
Convierte las fechas al formato yyyy-MM-dd para su uso en consultas SQL.
Consultando Datos

Llama al método obtenerListadoTarjetas de la clase ConexionMySQL para obtener los resultados filtrados.
Maneja excepciones relacionadas con SQL y formato de fecha.
Limpia la tabla actual y agrega nuevas filas con los datos obtenidos.
Muestra mensajes de error o de no resultados si es necesario.

Dependencias
JTable y DefaultTableModel para la visualización y manejo de datos tabulares.
JTextField y JComboBox para la entrada y selección de filtros.
JButton para activar la consulta de datos.
JOptionPane para mostrar mensajes de error y notificaciones.
SimpleDateFormat para la conversión de fechas.
ConexionMySQL para la conexión a la base de datos y ejecución de consultas.
Manejo de Errores
Validación de campos numéricos: Se verifica que el límite ingresado sea un número válido.
Formato de fecha: Se asegura de que las fechas estén en el formato correcto.
Excepciones de SQL: Se manejan posibles errores en la consulta y acceso a la base de datos.
Uso
Esta clase se utiliza en el contexto de una aplicación de gestión de tarjetas, donde se requiere visualizar y filtrar datos de tarjetas de crédito. Se integra dentro de un JDesktopPane y proporciona una interfaz amigable para los usuarios finales que necesitan generar reportes y visualizar información específica de tarjetas.

reportesSubOpcion3ListadoDeSolicitudes

Descripción General

La clase reportesSubOpcion3ListadoDeSolicitudes extiende JInternalFrame y proporciona una interfaz gráfica para visualizar un listado de solicitudes. Permite a los usuarios filtrar las solicitudes por fecha, tipo, salario y estado. La información se muestra en una tabla, que se actualiza de acuerdo a los filtros aplicados.

Componentes

JTable: Utilizada para mostrar el listado de solicitudes. Se integra con un modelo de tabla (DefaultTableModel) para manejar los datos.
DefaultTableModel: Modelo de datos de la tabla que define las columnas y maneja las filas de datos.
JTextField: Campos de texto para ingresar fechas y salario.
JComboBox: Menús desplegables para seleccionar tipo y estado de la solicitud.
JButton: Botón que inicia la consulta de solicitudes basadas en los filtros aplicados.

Constructor

El constructor de la clase realiza las siguientes tareas:

Configuración del JInternalFrame: Establece el título, tamaño, y otras propiedades de la ventana interna.
Creación de la Interfaz: Configura y organiza los componentes de la interfaz gráfica utilizando un diseño GridBagLayout.
Panel de Filtros: Incluye campos de texto y menús desplegables para aplicar filtros. Agrega un botón "Consultar" para ejecutar la consulta de datos.
Modelo de la Tabla: Define las columnas de la tabla y añade la tabla a un JScrollPane para permitir el desplazamiento.
Panel Principal: Agrupa los componentes y los añade a la ventana interna.

Funcionalidad

aplicarFiltros()

Este método se encarga de:

Recopilar los Datos de los Filtros: Obtiene los valores de los campos de texto y menús desplegables.
Conversión de Fechas: Convierte las fechas del formato dd/MM/yyyy al formato yyyy-MM-dd requerido por la base de datos.
Consulta a la Base de Datos: Llama al método obtenerListadoSolicitudes de la clase ConexionMySQL para obtener los resultados filtrados.
Actualización de la Tabla: Limpia la tabla y agrega nuevas filas con los resultados obtenidos.
Maneja posibles errores en la consulta y en el formato de fechas.

Dependencias

JTable y DefaultTableModel para la visualización y manejo de datos tabulares.
JTextField y JComboBox para la entrada y selección de filtros.
JButton para activar la consulta de datos.
JOptionPane para mostrar mensajes de error y notificaciones.
SimpleDateFormat para la conversión de fechas.
ConexionMySQL para la conexión a la base de datos y la ejecución de consultas.

Manejo de Errores

Formato de Fecha: Se valida que las fechas tengan el formato correcto. Si el formato es incorrecto, se muestra un mensaje de error.
Consulta SQL: Se manejan excepciones relacionadas con la base de datos, mostrando mensajes adecuados si ocurre un error durante la consulta o al procesar los datos.
Uso
La clase reportesSubOpcion3ListadoDeSolicitudes se utiliza en el contexto de una aplicación de gestión de solicitudes. Ofrece una interfaz para filtrar y visualizar solicitudes basadas en criterios específicos, facilitando la gestión y el análisis de las solicitudes dentro de la aplicación.

Backend

Clase Autorizacion
Descripción General
La clase Autorizacion es responsable de manejar el proceso de autorización de solicitudes de tarjetas de crédito. Se encarga de validar y autorizar una solicitud basada en varios criterios, generar un número de tarjeta, y actualizar la base de datos con la información relevante.

Componentes y Métodos
Atributos:

idAutorizacion: Identificador único para la autorización (actualmente no utilizado en el código proporcionado).
numeroTarjeta: Número de tarjeta generado tras la autorización.
numeroSolicitud: Identificador de la solicitud que se está autorizando.
fechaAutorizacion: Fecha en la que se realiza la autorización.
Constructor:

Autorizacion(int numeroSolicitud): Inicializa una nueva instancia de Autorizacion con el número de solicitud proporcionado y establece la fecha de autorización a la fecha actual.
Métodos Públicos:

Tarjeta autorizar():
Descripción: Autoriza la solicitud basada en el número de solicitud. Verifica si la solicitud existe, si ya está aprobada, y si cumple con los criterios para ser autorizada. Genera una nueva tarjeta si la autorización es exitosa.
Excepciones: Lanza SQLException en caso de errores con la base de datos, IllegalArgumentException para casos de errores de autorización.
Solicitud obtenerSolicitud(Connection connection, int numeroSolicitud):
Descripción: Obtiene una solicitud de la base de datos usando el número de solicitud proporcionado.
Excepciones: Lanza SQLException en caso de errores con la base de datos.
Métodos Privados:

double calcularLimiteCredito(double salario):

Descripción: Calcula el límite de crédito basado en el salario del solicitante.
boolean verificarAutorizacion(double limite, String tipoSolicitud):

Descripción: Verifica si el límite de crédito cumple con los requisitos mínimos según el tipo de solicitud (NACIONAL, REGIONAL, INTERNACIONAL).
String obtenerMotivoRechazo(String tipoSolicitud, double limite):

Descripción: Obtiene el motivo del rechazo en caso de que la solicitud no cumpla con los requisitos de límite de crédito.
boolean numeroTarjetaExiste(Connection connection, String numeroTarjeta):

Descripción: Verifica si el número de tarjeta ya existe en la base de datos.
Excepciones: Lanza SQLException en caso de errores con la base de datos.
void guardarAutorizacionEnBaseDeDatos(Connection connection, String numeroTarjeta, int numeroSolicitud, LocalDate fechaAutorizacion):

Descripción: Guarda la información de la autorización en la base de datos.
Excepciones: Lanza SQLException en caso de errores con la base de datos.
void actualizarEstadoSolicitud(Connection connection, int numeroSolicitud, String estado, String motivoRechazo):

Descripción: Actualiza el estado de la solicitud en la base de datos.
Excepciones: Lanza SQLException en caso de errores con la base de datos.
void actualizarFechaUltimoEstado(Connection connection, String numeroTarjeta, LocalDate fechaUltimoEstado):

Descripción: Actualiza la fecha del último estado de la tarjeta en la base de datos.
Excepciones: Lanza SQLException en caso de errores con la base de datos.
Flujo de Trabajo
Inicialización:

Se crea una instancia de Autorizacion con el número de solicitud que se desea autorizar.
Autorización:

Se llama al método autorizar(), que realiza los siguientes pasos:
Obtiene la solicitud desde la base de datos.
Verifica si la solicitud ya está aprobada.
Calcula el límite de crédito basado en el salario del solicitante.
Verifica si el límite cumple con los requisitos para el tipo de solicitud.
Genera un número de tarjeta único.
Guarda la nueva tarjeta en la base de datos.
Actualiza la fecha de autorización y el estado de la solicitud.
Maneja posibles errores y rechazos.
Actualización y Validación:

El método aplicarFiltros() gestiona la validación de los datos y actualiza la base de datos en función de la autorización o rechazo de la solicitud.
Interacción con la Base de Datos
La clase utiliza ConexionMySQL para establecer conexiones con la base de datos y ejecutar consultas SQL.
Las operaciones de base de datos incluyen la obtención de solicitudes, la verificación de existencia de números de tarjeta, la actualización de estados, y el almacenamiento de autorizaciones.
Consideraciones de Seguridad
Validación de Datos: Se valida que los datos sean correctos y que las solicitudes cumplan con los criterios antes de autorizarlas.
Manejo de Excepciones: Se capturan y manejan excepciones para evitar fallos en la aplicación y proporcionar mensajes de error adecuados.

Clase Cancelacion

Descripción General

La clase Cancelacion gestiona la cancelación de una tarjeta. Verifica si la tarjeta está activa y si tiene saldo pendiente antes de proceder con la cancelación. Si las condiciones se cumplen, actualiza el estado de la tarjeta a “CANCELADA” y guarda la cancelación en la base de datos.

Atributos

numeroTarjeta (String): Número de la tarjeta que se desea cancelar.

Constructor

El constructor de la clase inicializa el atributo numeroTarjeta.

Métodos

cancelar(): Realiza la cancelación de la tarjeta. Verifica si la tarjeta está activa y si no tiene saldo pendiente antes de proceder con la cancelación.

tarjetaActiva(Connection connection): Verifica si la tarjeta está activa en la base de datos.

actualizarEstadoTarjeta(Connection connection): Actualiza el estado de la tarjeta a “CANCELADA” en la base de datos.

actualizarFechaUltimoEstado(Connection connection, LocalDate fechaUltimoEstado): Actualiza la fecha del último estado de la tarjeta en la base de datos.

guardarCancelacionEnBaseDeDatos(Connection connection): Guarda la cancelación de la tarjeta en la base de datos.

Componentes y Herramientas Usadas

Connection: Utilizado para establecer la conexión con la base de datos.

PreparedStatement: Utilizado para ejecutar consultas SQL parametrizadas.

ResultSet: Utilizado para almacenar los resultados de una consulta SQL.

SQLException: Excepción lanzada en caso de errores relacionados con la base de datos.

Manejo de Excepciones

El método cancelar() maneja las excepciones SQL mediante bloques try-catch, asegurando que el programa no se detenga si ocurre un error durante la ejecución de las consultas SQL. Los mensajes de error se imprimen en la consola.

Ejemplo de Uso

Para cancelar una tarjeta, se crea una instancia de la clase Cancelacion y se llama al método cancelar().

Conclusión

La clase Cancelacion proporciona una funcionalidad esencial para la gestión de tarjetas en el sistema bancario. Verifica las condiciones necesarias antes de cancelar una tarjeta y maneja adecuadamente las excepciones para asegurar una operación robusta.

Clase ConexionMySQL

Descripción General

La clase ConexionMySQL gestiona la conexión a la base de datos MySQL utilizada por el sistema bancario. Proporciona métodos para obtener y cerrar conexiones, así como para realizar consultas específicas sobre solicitudes y tarjetas.

Atributos

URL_MYSQL (String): URL de conexión a la base de datos MySQL.
USER (String): Nombre de usuario para la conexión a la base de datos.
PASSWORD (String): Contraseña para la conexión a la base de datos.
connection (Connection): Objeto de conexión a la base de datos.

Constructor

La clase no tiene un constructor explícito, ya que todos los métodos y atributos son estáticos.

Métodos

getConnection(): Obtiene una conexión a la base de datos. Si la conexión actual está cerrada o no existe, se crea una nueva conexión.
isClosed(Connection conn): Verifica si una conexión está cerrada.
cerrarConexion(): Cierra la conexión a la base de datos si está abierta.
obtenerListadoSolicitudes(String fechaInicio, String fechaFin, String tipo, double salario, String estado): Realiza una consulta a la base de datos para obtener un listado de solicitudes filtradas por los parámetros proporcionados.
obtenerListadoTarjetas(String tipo, String nombre, double limite, String fechaInicio, String fechaFin, String estado): Realiza una consulta a la base de datos para obtener un listado de tarjetas filtradas por los parámetros proporcionados.

Componentes y Herramientas Usadas

DriverManager: Utilizado para obtener la conexión a la base de datos.
Connection: Representa una conexión a la base de datos.
PreparedStatement: Utilizado para ejecutar consultas SQL parametrizadas.
ResultSet: Utilizado para almacenar los resultados de una consulta SQL.
SQLException: Excepción lanzada en caso de errores relacionados con la base de datos.

Manejo de Excepciones

Los métodos getConnection(), cerrarConexion(), obtenerListadoSolicitudes() y obtenerListadoTarjetas() manejan las excepciones SQL mediante bloques try-catch, asegurando que el programa no se detenga si ocurre un error durante la ejecución de las consultas SQL. Los mensajes de error se imprimen en la consola.

Ejemplo de Uso

Para obtener una conexión a la base de datos y realizar una consulta de solicitudes, se puede utilizar el siguiente flujo:

Llamar al método getConnection() para obtener una conexión.
Utilizar la conexión para ejecutar una consulta mediante obtenerListadoSolicitudes().
Procesar los resultados obtenidos en un ResultSet.
Cerrar la conexión utilizando cerrarConexion().

Conclusión

La clase ConexionMySQL proporciona una funcionalidad esencial para la gestión de conexiones y consultas a la base de datos en el sistema bancario. Facilita la obtención y cierre de conexiones, así como la ejecución de consultas específicas, manejando adecuadamente las excepciones para asegurar una operación robusta.

Clase EstadoDeCuenta

Descripción General

La clase EstadoDeCuenta se encarga de gestionar y calcular el estado de cuenta de una tarjeta específica. Al instanciarse, carga los datos de la tarjeta, los movimientos asociados y calcula los totales correspondientes, incluyendo el saldo total y los intereses.

Atributos

numeroTarjeta (String): Número de la tarjeta para la cual se genera el estado de cuenta.
tipoTarjeta (String): Tipo de la tarjeta (NACIONAL, REGIONAL, INTERNACIONAL).
nombreCliente (String): Nombre del cliente titular de la tarjeta.
direccionCliente (String): Dirección del cliente titular de la tarjeta.
movimientos (List<Movimiento>): Lista de movimientos asociados a la tarjeta.
montoTotal (double): Monto total de los movimientos.
intereses (double): Intereses calculados sobre el monto total.
saldoTotal (double): Saldo total de la tarjeta, incluyendo intereses.

Constructor

El constructor de la clase inicializa el atributo numeroTarjeta y llama a métodos privados para cargar los datos de la tarjeta, los movimientos y calcular los totales.

Métodos

cargarDatosTarjeta(): Carga los datos básicos de la tarjeta (tipo, nombre del cliente, dirección) desde la base de datos, siempre y cuando la tarjeta esté activa.
cargarMovimientos(): Carga todos los movimientos asociados a la tarjeta desde la base de datos y los almacena en la lista movimientos.
calcularTotales(): Calcula el monto total de los movimientos, los intereses basados en el tipo de tarjeta y el saldo total.
calcularIntereses(double monto, String tipoTarjeta): Calcula los intereses sobre el monto total basado en el tipo de tarjeta.
getSaldoTotal(): Devuelve el saldo total de la tarjeta.
getMovimientos(): Devuelve la lista de movimientos asociados a la tarjeta.
filtrarPorNumeroTarjeta(String numeroTarjeta): Filtra y devuelve una lista de estados de cuenta de tarjetas cuyo número coincida parcialmente con el número proporcionado.
filtrarPorTipoTarjeta(String tipoTarjeta): Filtra y devuelve una lista de estados de cuenta de tarjetas del tipo especificado.
obtenerTodasLasTarjetasActivas(): Devuelve una lista de estados de cuenta de todas las tarjetas activas.
filtrarPorSaldoMayorA(double monto): Filtra y devuelve una lista de estados de cuenta de tarjetas cuyo saldo total sea mayor al monto especificado.
filtrarPorInteresesMayorA(double intereses): Filtra y devuelve una lista de estados de cuenta de tarjetas cuyos intereses sean mayores al valor especificado.

Componentes y Herramientas Usadas

Connection: Utilizado para establecer la conexión con la base de datos.
PreparedStatement: Utilizado para ejecutar consultas SQL parametrizadas.
ResultSet: Utilizado para almacenar los resultados de una consulta SQL.
SQLException: Excepción lanzada en caso de errores relacionados con la base de datos.
Manejo de Excepciones
Los métodos que interactúan con la base de datos manejan las excepciones SQL mediante bloques try-catch, asegurando que el programa no se detenga si ocurre un error durante la ejecución de las consultas SQL. Los mensajes de error se imprimen en la consola.

Ejemplo de Uso
Para generar un estado de cuenta, se crea una instancia de la clase EstadoDeCuenta proporcionando el número de tarjeta. El constructor automáticamente carga los datos necesarios y calcula los totales. Además, se pueden utilizar los métodos estáticos para filtrar estados de cuenta según diferentes criterios.

Conclusión
La clase EstadoDeCuenta proporciona una funcionalidad esencial para la gestión de estados de cuenta en el sistema bancario. Facilita la carga de datos, el cálculo de totales y la filtración de estados de cuenta según diversos criterios, manejando adecuadamente las excepciones para asegurar una operación robusta.

Clase GeneradorNumeroTarjeta
Descripción General
La clase GeneradorNumeroTarjeta se encarga de generar números de tarjeta secuenciales para diferentes tipos de tarjetas (NACIONAL, REGIONAL, INTERNACIONAL). Inicializa contadores para cada tipo de tarjeta desde la base de datos y proporciona un método para generar nuevos números de tarjeta basados en estos contadores.

Atributos
contadorNacional (int): Contador para las tarjetas de tipo NACIONAL.
contadorRegional (int): Contador para las tarjetas de tipo REGIONAL.
contadorInternacional (int): Contador para las tarjetas de tipo INTERNACIONAL.
Constructor
El constructor de la clase inicializa los contadores de tarjetas desde la base de datos utilizando el método inicializarContadores.

Métodos
inicializarContadores(ConexionMySQL conexion): Este método se conecta a la base de datos y cuenta el número de tarjetas de cada tipo (NACIONAL, REGIONAL, INTERNACIONAL). Los resultados se utilizan para inicializar los contadores correspondientes.
SQL Query: Ejecuta una consulta SQL que agrupa las tarjetas por tipo y cuenta cuántas hay de cada tipo.
Proceso:
Se establece una conexión a la base de datos.
Se ejecuta la consulta SQL.

Se recorren los resultados y se asignan los contadores según el tipo de tarjeta.
Si se encuentra un tipo de tarjeta desconocido, se lanza una excepción.
generarNumeroSecuencial(TipoTarjeta tipo): Este método genera un nuevo número de tarjeta secuencial basado en el tipo de tarjeta proporcionado.
Proceso:
Incrementa el contador correspondiente al tipo de tarjeta.
Formatea el nuevo número de tarjeta utilizando el contador incrementado.
Devuelve el número de tarjeta en el formato adecuado.
Formato del Número de Tarjeta:
NACIONAL: “4256 3102 654XX XXXX”
REGIONAL: “4256 3102 656XX XXXX”
INTERNACIONAL: “4256 3102 658XX XXXX”
Excepción: Si se proporciona un tipo de tarjeta desconocido, se lanza una excepción.
Componentes y Herramientas Usadas
Connection: Utilizado para establecer la conexión con la base de datos.
Statement: Utilizado para ejecutar consultas SQL sin parámetros.
PreparedStatement: Utilizado para ejecutar consultas SQL parametrizadas.
ResultSet: Utilizado para almacenar los resultados de una consulta SQL.
SQLException: Excepción lanzada en caso de errores relacionados con la base de datos.
IllegalArgumentException: Excepción lanzada en caso de argumentos inválidos, como un tipo de tarjeta desconocido.
Manejo de Excepciones
Los métodos que interactúan con la base de datos manejan las excepciones SQL mediante bloques try-catch, asegurando que el programa no se detenga si ocurre un error durante la ejecución de las consultas SQL. Los mensajes de error se imprimen en la consola. Además, se manejan excepciones de tipo IllegalArgumentException para casos de argumentos inválidos.

Ejemplo de Uso
Para generar un nuevo número de tarjeta, se crea una instancia de la clase GeneradorNumeroTarjeta proporcionando una conexión a la base de datos. Luego, se llama al método generarNumeroSecuencial con el tipo de tarjeta deseado.

Crear una instancia de GeneradorNumeroTarjeta:
Inicializa los contadores desde la base de datos.
Llamar al método generarNumeroSecuencial con el tipo de tarjeta:
Genera y devuelve un nuevo número de tarjeta secuencial.
Conclusión
La clase GeneradorNumeroTarjeta proporciona una funcionalidad esencial para la generación de números de tarjeta en el sistema bancario. Inicializa los contadores desde la base de datos y genera números de tarjeta secuenciales basados en estos contadores, manejando adecuadamente las excepciones para asegurar una operación robusta.

Clase Movimiento

Descripción General

La clase Movimiento representa una transacción financiera asociada a una tarjeta. Cada movimiento incluye detalles como el número de tarjeta, la fecha de la transacción, el tipo de movimiento (CARGO o ABONO), una descripción, el establecimiento donde se realizó la transacción y el monto. La clase también proporciona métodos para validar la tarjeta y guardar el movimiento en la base de datos.

Atributos

idMovimiento (int): Identificador único del movimiento.
numeroTarjeta (String): Número de la tarjeta asociada al movimiento.
fecha (Date): Fecha en que se realizó el movimiento.
tipoMovimiento (String): Tipo de movimiento (CARGO o ABONO).
descripcion (String): Descripción del movimiento.
establecimiento (String): Establecimiento donde se realizó el movimiento.
monto (double): Monto de la transacción.

Constructor

El constructor de la clase inicializa los atributos del movimiento con los valores proporcionados.

Métodos

guardarEnBaseDeDatos(): Guarda el movimiento en la base de datos. Antes de guardar, valida que la tarjeta exista y esté activa.
Proceso:
Valida la tarjeta llamando al método tarjetaValida().
Si la tarjeta es válida, construye una consulta SQL para insertar el movimiento en la base de datos.
Ejecuta la consulta SQL utilizando un Statement.
Maneja excepciones SQL y lanza una excepción en caso de error.
Excepción: Si la tarjeta no es válida, lanza una excepción IllegalArgumentException.
tarjetaValida(): Verifica si la tarjeta asociada al movimiento existe y está activa.
Proceso:
Construye una consulta SQL para verificar el estado de la tarjeta.
Ejecuta la consulta SQL utilizando un Statement.
Recorre los resultados y verifica si el estado de la tarjeta es “ACTIVA”.
Devuelve true si la tarjeta es válida, de lo contrario, devuelve false.
Excepción: Maneja excepciones SQL y lanza una excepción en caso de error.
getMonto(): Devuelve el monto del movimiento.
getIdMovimiento(): Devuelve el identificador del movimiento.
getNumeroTarjeta(): Devuelve el número de la tarjeta asociada al movimiento.
getFecha(): Devuelve la fecha del movimiento.
getTipoMovimiento(): Devuelve el tipo de movimiento.
getDescripcion(): Devuelve la descripción del movimiento.
getEstablecimiento(): Devuelve el establecimiento donde se realizó el movimiento.

Componentes y Herramientas Usadas

Connection: Utilizado para establecer la conexión con la base de datos.
Statement: Utilizado para ejecutar consultas SQL sin parámetros.
ResultSet: Utilizado para almacenar los resultados de una consulta SQL.

SQLException: Excepción lanzada en caso de errores relacionados con la base de datos.
IllegalArgumentException: Excepción lanzada en caso de argumentos inválidos, como una tarjeta no válida.
Manejo de Excepciones
Los métodos que interactúan con la base de datos manejan las excepciones SQL mediante bloques try-catch, asegurando que el programa no se detenga si ocurre un error durante la ejecución de las consultas SQL. Los mensajes de error se imprimen en la consola. Además, se manejan excepciones de tipo IllegalArgumentException para casos de argumentos inválidos.

Ejemplo de Uso
Para registrar un movimiento, se crea una instancia de la clase Movimiento proporcionando los detalles del movimiento. Luego, se llama al método guardarEnBaseDeDatos para guardar el movimiento en la base de datos.

Crear una instancia de Movimiento:
Inicializa los atributos del movimiento con los valores proporcionados.
Llamar al método guardarEnBaseDeDatos:
Valida la tarjeta y guarda el movimiento en la base de datos.
Conclusión
La clase Movimiento proporciona una funcionalidad esencial para la gestión de transacciones financieras en el sistema bancario. Facilita la creación y almacenamiento de movimientos, validando la tarjeta asociada y manejando adecuadamente las excepciones

Clase Solicitud
Descripción General
La clase Solicitud representa una solicitud de servicio o producto en el sistema bancario. Cada solicitud incluye detalles como el número de solicitud, la fecha, el tipo de solicitud, el nombre del solicitante, el salario del solicitante, la dirección y el estado de la solicitud. La clase también proporciona métodos para guardar la solicitud en la base de datos y obtener el último número de solicitud registrado.

Atributos
numeroSolicitud (int): Identificador único de la solicitud.
fecha (LocalDate): Fecha en que se realizó la solicitud.
tipo (String): Tipo de solicitud.
nombre (String): Nombre del solicitante.
salario (Double): Salario del solicitante.
direccion (String): Dirección del solicitante.
estado (String): Estado de la solicitud (por ejemplo, PENDIENTE, APROBADA, RECHAZADA).
Constructor
El constructor de la clase inicializa los atributos de la solicitud con los valores proporcionados.

Métodos
guardarEnBaseDeDatos(): Guarda la solicitud en la base de datos.

Proceso:
Construye una consulta SQL para insertar la solicitud en la base de datos.
Ejecuta la consulta SQL utilizando un Statement.
Maneja excepciones SQL y lanza una excepción en caso de error.
Excepción: Si ocurre un error durante la ejecución de la consulta SQL, se lanza una excepción SQLException.
obtenerUltimoNumeroSolicitud(): Método estático que obtiene el último número de solicitud registrado en la base de datos.
Proceso:
Construye una consulta SQL para obtener el número máximo de solicitud.
Ejecuta la consulta SQL utilizando un Statement.
Recorre los resultados y obtiene el último número de solicitud.
Devuelve el último número de solicitud.
Excepción: Si ocurre un error durante la ejecución de la consulta SQL, se lanza una excepción SQLException.
toString(): Devuelve una representación en cadena de la solicitud.
Formato: La cadena devuelta sigue el formato SOLICITUD(numeroSolicitud, "fecha", tipo, "nombre", salario, "direccion", "estado").
Componentes y Herramientas Usadas
Connection: Utilizado para establecer la conexión con la base de datos.
Statement: Utilizado para ejecutar consultas SQL sin parámetros.
ResultSet: Utilizado para almacenar los resultados de una consulta SQL.
SQLException: Excepción lanzada en caso de errores relacionados con la base de datos.
Manejo de Excepciones
Los métodos que interactúan con la base de datos manejan las excepciones SQL mediante bloques try-catch, asegurando que el programa no se detenga si ocurre un error durante la ejecución de las consultas SQL. Los mensajes de error se imprimen en la consola.

Ejemplo de Uso
Para registrar una solicitud, se crea una instancia de la clase Solicitud proporcionando los detalles de la solicitud. Luego, se llama al método guardarEnBaseDeDatos para guardar la solicitud en la base de datos.

Crear una instancia de Solicitud:
Inicializa los atributos de la solicitud con los valores proporcionados.
Llamar al método guardarEnBaseDeDatos:
Guarda la solicitud en la base de datos.
Para obtener el último número de solicitud registrado, se llama al método estático obtenerUltimoNumeroSolicitud.

Conclusión
La clase Solicitud proporciona una funcionalidad esencial para la gestión de solicitudes en el sistema bancario. Facilita la creación y almacenamiento de solicitudes, así como la obtención del último número de solicitud registrado, manejando adecuadamente las excepciones para asegurar una operación

Clase Tarjeta

Descripción General

La clase Tarjeta representa una tarjeta de crédito o débito en el sistema bancario. Cada tarjeta incluye detalles como el número de tarjeta, el tipo de tarjeta, el límite de crédito, el nombre y la dirección del cliente, el estado de la tarjeta y el número de solicitud asociado.

Atributos

numeroTarjeta (String): Número único de la tarjeta.
tipoTarjeta (TipoTarjeta): Tipo de la tarjeta (NACIONAL, REGIONAL, INTERNACIONAL).
limite (double): Límite de crédito de la tarjeta.
nombreCliente (String): Nombre del cliente titular de la tarjeta.
direccionCliente (String): Dirección del cliente titular de la tarjeta.
estadoTarjeta (String): Estado actual de la tarjeta (por ejemplo, ACTIVA, CANCELADA).
numeroSolicitud (int): Número de solicitud asociado a la tarjeta.
Constructor

El constructor de la clase inicializa los atributos de la tarjeta con los valores proporcionados.

Métodos

getTipoTarjeta(): Devuelve el tipo de tarjeta.
getLimite(): Devuelve el límite de crédito de la tarjeta.
getNombreCliente(): Devuelve el nombre del cliente titular de la tarjeta.
getDireccionCliente(): Devuelve la dirección del cliente titular de la tarjeta.
getEstadoTarjeta(): Devuelve el estado actual de la tarjeta.
getNumeroSolicitud(): Devuelve el número de solicitud asociado a la tarjeta.
getNumeroTarjeta(): Devuelve el número único de la tarjeta.
guardarEnBaseDeDatos(Connection connection): Guarda la tarjeta en la base de datos.
Proceso:
Construye una consulta SQL para insertar la tarjeta en la base de datos.
Ejecuta la consulta SQL utilizando un Statement.
Maneja excepciones SQL y lanza una excepción en caso de error.
Excepción: Si ocurre un error durante la ejecución de la consulta SQL, se lanza una excepción SQLException.
Componentes y Herramientas Usadas

Connection: Utilizado para establecer la conexión con la base de datos.
Statement: Utilizado para ejecutar consultas SQL sin parámetros.
SQLException: Excepción lanzada en caso de errores relacionados con la base de datos.
Manejo de Excepciones

El método guardarEnBaseDeDatos maneja las excepciones SQL mediante bloques try-catch, asegurando que el programa no se detenga si ocurre un error durante la ejecución de las consultas SQL. Los mensajes de error se imprimen en la consola.

Ejemplo de Uso

Para registrar una tarjeta, se crea una instancia de la clase Tarjeta proporcionando los detalles de la tarjeta. Luego, se llama al método guardarEnBaseDeDatos para guardar la tarjeta en la base de datos.

Crear una instancia de Tarjeta:
Inicializa los atributos de la tarjeta con los valores proporcionados.

Llamar al método guardarEnBaseDeDatos:
Guarda la tarjeta en la base de datos.
Conclusión
La clase Tarjeta proporciona una funcionalidad esencial para la gestión de tarjetas en el sistema bancario. Facilita la creación y almacenamiento de tarjetas

Enum TipoTarjeta
Descripción General
TipoTarjeta es una enumeración que define los diferentes tipos de tarjetas disponibles en el sistema bancario. Esta enumeración se utiliza para categorizar las tarjetas en tres tipos distintos: NACIONAL, REGIONAL e INTERNACIONAL.

Valores de la Enumeración
NACIONAL: Representa una tarjeta que se utiliza principalmente dentro del país.
REGIONAL: Representa una tarjeta que se puede utilizar en una región específica, que puede incluir varios países.
INTERNACIONAL: Representa una tarjeta que se puede utilizar a nivel mundial, sin restricciones geográficas.
Uso en el Sistema
La enumeración TipoTarjeta se utiliza en varias clases del sistema bancario para definir el tipo de tarjeta. Por ejemplo, en la clase Tarjeta, el atributo tipoTarjeta es de tipo TipoTarjeta, lo que permite especificar si una tarjeta es NACIONAL, REGIONAL o INTERNACIONAL.

Ejemplo de Uso
Para crear una instancia de una tarjeta con un tipo específico, se puede utilizar la enumeración TipoTarjeta de la siguiente manera:

Definir el tipo de tarjeta al crear una nueva tarjeta:
TipoTarjeta tipo = TipoTarjeta.NACIONAL;
Utilizar el tipo de tarjeta en el constructor de la clase Tarjeta:
Tarjeta tarjeta = new Tarjeta("1234567890123456", tipo, 5000.0, "Juan Pérez", "Calle Falsa 123", "ACTIVA", 1);
Conclusión
La enumeración TipoTarjeta proporciona una forma clara y concisa de definir los tipos de tarjetas en el sistema bancario. Facilita la categorización y el manejo de tarjetas, asegurando que se utilicen valores consistentes y predefinidos en todo el sistema.

Clase operadorAutorizacionTarjeta

Descripción General

La clase operadorAutorizacionTarjeta se encarga de procesar las solicitudes de autorización de tarjetas. Utiliza un área de texto (JTextArea) para registrar los resultados de las autorizaciones, incluyendo tanto los éxitos como los errores.

Atributos

logArea (JTextArea): Área de texto donde se registran los resultados de las autorizaciones.

Constructor

El constructor de la clase inicializa el atributo logArea con el área de texto proporcionada.

Métodos

ejecutarAutorizacionTarjeta(String linea): Procesa una línea de texto que contiene una solicitud de autorización de tarjeta.

Proceso:

Descompone la línea para extraer el número de solicitud.

Crea una instancia de la clase Autorizacion utilizando el número de solicitud.

Intenta autorizar la tarjeta llamando al método autorizar de la clase Autorizacion.

Si la autorización es exitosa, registra el éxito y el número de tarjeta generado en logArea.

Si ocurre un error durante la autorización (por ejemplo, una excepción IllegalArgumentException), registra el error en logArea.

Si ocurre cualquier otra excepción durante el procesamiento, registra un mensaje de error en logArea y muestra la traza de la excepción.

Excepción: Maneja excepciones IllegalArgumentException y otras excepciones generales para asegurar que los errores se registren adecuadamente.

Componentes y Herramientas Usadas

JTextArea: Utilizado para mostrar y registrar los resultados de las autorizaciones.

Autorizacion: Clase utilizada para autorizar las tarjetas.

Tarjeta: Clase que representa la tarjeta generada tras una autorización exitosa.

IllegalArgumentException: Excepción lanzada en caso de argumentos inválidos durante la autorización.

Exception: Excepción general manejada para capturar cualquier otro error durante el procesamiento.

Manejo de Excepciones

El método ejecutarAutorizacionTarjeta maneja las excepciones IllegalArgumentException y otras excepciones generales mediante bloques try-catch, asegurando que los errores se registren en logArea y que el programa no se detenga si ocurre un error durante el procesamiento.

Ejemplo de Uso

Para procesar una solicitud de autorización de tarjeta, se crea una instancia de la clase operadorAutorizacionTarjeta proporcionando un área de texto para los registros. Luego, se llama al método ejecutarAutorizacionTarjeta con una línea de texto que contiene la solicitud.

Crear una instancia de operadorAutorizacionTarjeta:

Inicializa el atributo logArea con el área de texto proporcionada.

Llamar al método ejecutarAutorizacionTarjeta con una línea de texto:

Procesa la solicitud de autorización y registra los resultados en logArea.

Conclusión
La clase operadorAutorizacionTarjeta proporciona una funcionalidad esencial para el procesamiento y registro de solicitudes de autorización de tarjetas en el sistema bancario. Facilita la autorización de tarjetas y el registro de resultados, manejando adecuadamente las excepciones

Clase operadorCancelacionTarjeta
Descripción General
La clase operadorCancelacionTarjeta se encarga de procesar las solicitudes de cancelación de tarjetas. Utiliza un área de texto (JTextArea) para registrar los resultados de las cancelaciones, incluyendo tanto los éxitos como los errores.

Atributos
logArea (JTextArea): Área de texto donde se registran los resultados de las cancelaciones.
Constructor
El constructor de la clase inicializa el atributo logArea con el área de texto proporcionada.

Métodos
ejecutarCancelacionTarjeta(String linea): Procesa una línea de texto que contiene una solicitud de cancelación de tarjeta.
Proceso:
Descompone la línea para extraer el número de tarjeta.
Crea una instancia de la clase Cancelacion utilizando el número de tarjeta.
Intenta cancelar la tarjeta llamando al método cancelar de la clase Cancelacion.
Si la cancelación es exitosa, registra el éxito en logArea.
Si ocurre un error durante la cancelación (por ejemplo, una excepción IllegalArgumentException), registra el error en logArea.
Si ocurre cualquier otra excepción durante el procesamiento, registra un mensaje de error en logArea y muestra la traza de la excepción.
Excepción: Maneja excepciones IllegalArgumentException y otras excepciones generales para asegurar que los errores se registren adecuadamente.
Componentes y Herramientas Usadas
JTextArea: Utilizado para mostrar y registrar los resultados de las cancelaciones.
Cancelacion: Clase utilizada para cancelar las tarjetas.
IllegalArgumentException: Excepción lanzada en caso de argumentos inválidos durante la cancelación.
Exception: Excepción general manejada para capturar cualquier otro error durante el procesamiento.
Manejo de Excepciones
El método ejecutarCancelacionTarjeta maneja las excepciones IllegalArgumentException y otras excepciones generales mediante bloques try-catch, asegurando que los errores se registren en logArea y que el programa no se detenga si ocurre un error durante el procesamiento.

Ejemplo de Uso
Para procesar una solicitud de cancelación de tarjeta, se crea una instancia de la clase operadorCancelacionTarjeta proporcionando un área de texto para los registros. Luego, se llama al método ejecutarCancelacionTarjeta con una línea de texto que contiene la solicitud.

Crear una instancia de operadorCancelacionTarjeta:
Inicializa el atributo logArea con el área de texto proporcionada.
Llamar al método ejecutarCancelacionTarjeta con una línea de texto:
Procesa la solicitud de cancelación y registra los resultados en logArea.

Clase operadorConsultarTarjeta
Descripción General
La clase operadorConsultarTarjeta se encarga de procesar las solicitudes de consulta de tarjetas. Utiliza un área de texto (JTextArea) para registrar los resultados de las consultas, incluyendo tanto los éxitos como los errores. Además, genera un archivo HTML con los datos de la tarjeta consultada.

Atributos
logArea (JTextArea): Área de texto donde se registran los resultados de las consultas.
Constructor
El constructor de la clase inicializa el atributo logArea con el área de texto proporcionada.

Métodos
ejecutarConsultarTarjeta(String linea): Procesa una línea de texto que contiene una solicitud de consulta de tarjeta.
Proceso:
Descompone la línea para extraer el número de tarjeta.
Construye una consulta SQL para obtener los datos de la tarjeta desde la base de datos.
Ejecuta la consulta SQL utilizando un PreparedStatement.
Si la tarjeta se encuentra, llama al método generarArchivoHTML1 para generar un archivo HTML con los datos de la tarjeta y registra el éxito en logArea.
Si la tarjeta no se encuentra, registra un mensaje de error en logArea.
Si ocurre cualquier otra excepción durante el procesamiento, registra un mensaje de error en logArea y muestra la traza de la excepción.
Excepción: Maneja excepciones generales para asegurar que los errores se registren adecuadamente.
generarArchivoHTML1(String numeroTarjeta, String tipoTarjeta, double limite, String nombreCliente, String direccionCliente, String estadoTarjeta): Genera un archivo HTML con los datos de la tarjeta.
Proceso:
Construye el contenido HTML con los datos de la tarjeta.
Obtiene la ruta seleccionada desde la clase EntradaArchivo.
Escribe el contenido HTML en un archivo utilizando un BufferedWriter.
Excepción: Maneja excepciones IOException para asegurar que los errores se registren adecuadamente.
Componentes y Herramientas Usadas

JTextArea: Utilizado para mostrar y registrar los resultados de las consultas.
Connection: Utilizado para establecer la conexión con la base de datos.
PreparedStatement: Utilizado para ejecutar consultas SQL parametrizadas.
ResultSet: Utilizado para almacenar los resultados de una consulta SQL.
IOException: Excepción lanzada en caso de errores relacionados con la escritura de archivos.
Exception: Excepción general manejada para capturar cualquier otro error durante el procesamiento.
Manejo de Excepciones
El método ejecutarConsultarTarjeta maneja las excepciones generales mediante bloques try-catch, asegurando que los errores se registren en logArea y que el programa no se detenga si ocurre un error durante el procesamiento. El método generarArchivoHTML1 maneja las excepciones IOException para asegurar que los errores se registren adecuadamente.

Ejemplo de Uso
Para procesar una solicitud de consulta de tarjeta, se crea una instancia de la clase operadorConsultarTarjeta proporcionando un área de texto para los registros. Luego, se llama al método ejecutarConsultarTarjeta con una línea de texto que contiene la solicitud.

Crear una instancia de operadorConsultarTarjeta:
Inicializa el atributo logArea con el área de texto proporcionada.
Llamar al método ejecutarConsultarTarjeta con una línea de texto:
Procesa la solicitud de consulta y registra los resultados en logArea.
Conclusión
La clase operadorConsultarTarjeta proporciona una funcionalidad esencial para el procesamiento y registro de solicitudes de consulta de tarjetas en el sistema bancario.

Clase operadorDeArchivo
Descripción General
La clase operadorDeArchivo se encarga de procesar diferentes tipos de líneas de comandos relacionadas con operaciones bancarias. Utiliza un área de texto (JTextArea) para registrar los resultados de las operaciones, incluyendo tanto los éxitos como los errores. Dependiendo del tipo de línea de comando, delega la ejecución a la clase correspondiente.

Atributos
logArea (JTextArea): Área de texto donde se registran los resultados de las operaciones.
Constructor
El constructor de la clase inicializa el atributo logArea con el área de texto proporcionada.

Métodos
ejecutarLinea(String linea): Procesa una línea de texto que contiene un comando para ejecutar una operación específica.
Proceso:
Verifica el tipo de comando contenido en la línea.

Crea una instancia de la clase correspondiente y llama al método adecuado para ejecutar la operación.
Registra los resultados de la operación en logArea.
Si la línea no coincide con ningún comando válido, registra un mensaje de error en logArea.
Tipos de Comandos:
SOLICITUD(: Procesa una solicitud utilizando la clase operadorSolicitud.
MOVIMIENTO(: Procesa un movimiento utilizando la clase operadorMovimiento.
CONSULTAR_TARJETA(: Consulta una tarjeta utilizando la clase operadorConsultarTarjeta.
AUTORIZACION_TARJETA(: Autoriza una tarjeta utilizando la clase operadorAutorizacionTarjeta.
CANCELACION_TARJETA(: Cancela una tarjeta utilizando la clase operadorCancelacionTarjeta.
ESTADO_CUENTA(: Consulta el estado de cuenta utilizando la clase operadorEstadoDeCuenta.
LISTADO_TARJETAS(: Genera un listado de tarjetas utilizando la clase operadorListadoTarjetas.
LISTADO_SOLICITUDES(: Genera un listado de solicitudes utilizando la clase operadorListadoSolicitudes.
Excepción: Maneja excepciones generales para asegurar que los errores se registren adecuadamente.
Componentes y Herramientas Usadas
JTextArea: Utilizado para mostrar y registrar los resultados de las operaciones.
operadorSolicitud: Clase utilizada para procesar solicitudes.
operadorMovimiento: Clase utilizada para procesar movimientos.
operadorConsultarTarjeta: Clase utilizada para consultar tarjetas.
operadorAutorizacionTarjeta: Clase utilizada para autorizar tarjetas.
operadorCancelacionTarjeta: Clase utilizada para cancelar tarjetas.
operadorEstadoDeCuenta: Clase utilizada para consultar el estado de cuenta.
operadorListadoTarjetas: Clase utilizada para generar un listado de tarjetas.
operadorListadoSolicitudes: Clase utilizada para generar un listado de solicitudes.
Exception: Excepción general manejada para capturar cualquier otro error durante el procesamiento.
Manejo de Excepciones
El método ejecutarLinea maneja las excepciones generales mediante bloques try-catch, asegurando que los errores se registren en logArea y que el programa no se detenga si ocurre un error durante el procesamiento.

Ejemplo de Uso
Para procesar una línea de comando, se crea una instancia de la clase operadorDeArchivo proporcionando un área de texto para los registros. Luego, se llama al método ejecutarLinea con una línea de texto que contiene el comando.

Crear una instancia de operadorDeArchivo:
Inicializa el atributo logArea con el área de texto proporcionada.
Llamar al método ejecutarLinea con una línea de texto:
Procesa la línea de comando y registra los resultados en logArea.
Conclusión

La clase operadorDeArchivo proporciona una funcionalidad esencial para el procesamiento y registro de diferentes tipos de comandos relacionados con operaciones bancarias en el sistema.

Clase operadorEstadoDeCuenta

Descripción General

La clase operadorEstadoDeCuenta se encarga de procesar las solicitudes de estado de cuenta. Utiliza un área de texto (JTextArea) para registrar los resultados de las consultas, incluyendo tanto los éxitos como los errores. Además, genera un archivo HTML con los estados de cuenta filtrados según los criterios proporcionados.

Atributos

logArea (JTextArea): Área de texto donde se registran los resultados de las consultas.

Constructor

El constructor de la clase inicializa el atributo logArea con el área de texto proporcionada.

Métodos

ejecutarEstadoCuenta(String linea): Procesa una línea de texto que contiene una solicitud de estado de cuenta.

Proceso:

Descompone la línea para extraer los filtros (número de tarjeta, tipo de tarjeta, saldo mayor a, intereses mayor a).

Obtiene los estados de cuenta aplicando los filtros llamando al método obtenerEstadosDeCuentaFiltrados.

Genera un archivo HTML con los estados de cuenta llamando al método generarArchivoHTML2.

Registra el éxito de la operación en logArea.

Si ocurre cualquier otra excepción durante el procesamiento, registra un mensaje de error en logArea y muestra la traza de la excepción.

Excepción: Maneja excepciones generales para asegurar que los errores se registren adecuadamente.

obtenerEstadosDeCuentaFiltrados(String numeroTarjeta, String tipoTarjeta, double saldoMayorA, double interesesMayorA): Obtiene los estados de cuenta aplicando los filtros proporcionados.

Proceso:

Obtiene todas las tarjetas activas llamando a EstadoDeCuenta.obtenerTodasLasTarjetasActivas.

Aplica los filtros de número de tarjeta, tipo de tarjeta, saldo mayor a, e intereses mayor a utilizando streams y colecciones.

Devuelve la lista de estados de cuenta filtrados.

Excepción: Maneja excepciones SQLException para asegurar que los errores se registren adecuadamente.

generarArchivoHTML2(List<EstadoDeCuenta> estadosDeCuenta, String numeroTarjeta, String tipoTarjeta, double saldoMayorA, double interesesMayorA): Genera un archivo HTML con los estados de cuenta.

Proceso:

Construye el contenido HTML con los datos de los estados de cuenta.

Genera el nombre del archivo basado en los filtros aplicados.
Obtiene la ruta seleccionada desde la clase EntradaArchivo.
Escribe el contenido HTML en un archivo utilizando un BufferedWriter.
Excepción: Maneja excepciones IOException para asegurar que los errores se registren adecuadamente.
Componentes y Herramientas Usadas
JTextArea: Utilizado para mostrar y registrar los resultados de las consultas.
Connection: Utilizado para establecer la conexión con la base de datos.
PreparedStatement: Utilizado para ejecutar consultas SQL parametrizadas.
ResultSet: Utilizado para almacenar los resultados de una consulta SQL.
BufferedWriter: Utilizado para escribir el contenido HTML en un archivo.
IOException: Excepción lanzada en caso de errores relacionados con la escritura de archivos.
SQLException: Excepción lanzada en caso de errores relacionados con la base de datos.
Exception: Excepción general manejada para capturar cualquier otro error durante el procesamiento.
Manejo de Excepciones
El método ejecutarEstadoCuenta maneja las excepciones generales mediante bloques try-catch, asegurando que los errores se registren en logArea y que el programa no se detenga si ocurre un error durante el procesamiento. El método obtenerEstadosDeCuentaFiltrados maneja las excepciones SQLException y el método generarArchivoHTML2 maneja las excepciones IOException para asegurar que los errores se registren adecuadamente.

Ejemplo de Uso
Para procesar una solicitud de estado de cuenta, se crea una instancia de la clase operadorEstadoDeCuenta proporcionando un área de texto para los registros. Luego, se llama al método ejecutarEstadoCuenta con una línea de texto que contiene la solicitud.

Crear una instancia de operadorEstadoDeCuenta:
Inicializa el atributo logArea con el área de texto proporcionada.
Llamar al método ejecutarEstadoCuenta con una línea de texto:
Procesa la solicitud de estado de cuenta y registra los resultados en logArea.

Clase operadorListadoSolicitudes
Descripción General
La clase operadorListadoSolicitudes se encarga de procesar las solicitudes para generar un listado de solicitudes. Utiliza un área de texto (JTextArea) para registrar los resultados de las operaciones, incluyendo tanto los éxitos como los errores. Además, genera un archivo HTML con el listado de solicitudes filtradas según los criterios proporcionados.

Atributos
logArea (JTextArea): Área de texto donde se registran los resultados de las operaciones.
Constructor
El constructor de la clase inicializa el atributo logArea con el área de texto proporcionada.

Métodos

ejecutarListadoSolicitudes(String linea): Procesa una línea de texto que contiene una solicitud para generar un listado de solicitudes.
Proceso:
Descompone la línea para extraer los filtros (fecha de inicio, fecha de fin, tipo, monto, estado).
Construye la consulta SQL con los filtros aplicados.
Obtiene la conexión a la base de datos y prepara la consulta utilizando un PreparedStatement.
Ejecuta la consulta SQL y construye el contenido HTML con los resultados.
Genera el archivo HTML con el listado de solicitudes llamando al método generarArchivoHTML4.
Registra el éxito de la operación en logArea.
Si ocurre cualquier otra excepción durante el procesamiento, registra un mensaje de error en logArea y muestra la traza de la excepción.
Excepción: Maneja excepciones generales para asegurar que los errores se registren adecuadamente.
generarArchivoHTML4(String htmlContent, String fechaInicioStr, String fechaFinStr, String tipo, String montoStr, String estado): Genera un archivo HTML con el listado de solicitudes.
Proceso:
Limpia comillas especiales de las fechas.
Genera el nombre del archivo basado en los filtros aplicados.
Obtiene la ruta seleccionada desde la clase EntradaArchivo.
Escribe el contenido HTML en un archivo utilizando un BufferedWriter.
Registra el éxito de la operación en logArea.
Excepción: Maneja excepciones IOException para asegurar que los errores se registren adecuadamente.
parseDate(String dateStr): Convierte una cadena de texto en un objeto Date.
Proceso:
Utiliza SimpleDateFormat para parsear la cadena de texto en el formato dd/MM/yyyy.
Devuelve el objeto Date correspondiente.
Excepción: Maneja excepciones ParseException para asegurar que los errores se registren adecuadamente.
Componentes y Herramientas Usadas
JTextArea: Utilizado para mostrar y registrar los resultados de las operaciones.
Connection: Utilizado para establecer la conexión con la base de datos.
PreparedStatement: Utilizado para ejecutar consultas SQL parametrizadas.
ResultSet: Utilizado para almacenar los resultados de una consulta SQL.
BufferedWriter: Utilizado para escribir el contenido HTML en un archivo.
IOException: Excepción lanzada en caso de errores relacionados con la escritura de archivos.
SQLException: Excepción lanzada en caso de errores relacionados con la base de datos.
ParseException: Excepción lanzada en caso de errores relacionados con el parseo de fechas.
Exception: Excepción general manejada para capturar cualquier otro error durante el procesamiento.
Manejo de Excepciones
El método ejecutarListadoSolicitudes maneja las excepciones generales mediante bloques try-catch, asegurando que los errores se registren en logArea y que el programa no se

detenga si ocurre un error durante el procesamiento. El método generarArchivoHTML4 maneja las excepciones IOException y el método parseDate maneja las excepciones ParseException para asegurar que los errores se registren adecuadamente.

Ejemplo de Uso
Para procesar una solicitud para generar un listado de solicitudes, se crea una instancia de la clase operadorListadoSolicitudes proporcionando un área de texto para los registros. Luego, se llama al método ejecutarListadoSolicitudes con una línea de texto que contiene la solicitud.

Crear una instancia de operadorListadoSolicitudes:
Inicializa el atributo logArea con el área de texto proporcionada.
Llamar al método ejecutarListadoSolicitudes con una línea de texto:
Procesa la solicitud y registra los resultados en logArea.
Conclusión
La clase operadorListadoSolicitudes proporciona una funcionalidad esencial para el procesamiento y registro de solicitudes para generar listados de solicitudes en el sistema bancario. Facilita la generación de listados y el registro de resultados, manejando adecuadamente las excepciones para asegurar una operación robusta.

Clase operadorListadoTarjetas
Descripción General
La clase operadorListadoTarjetas se encarga de procesar las solicitudes para generar un listado de tarjetas. Utiliza un área de texto (JTextArea) para registrar los resultados de las operaciones, incluyendo tanto los éxitos como los errores. Además, genera un archivo HTML con el listado de tarjetas filtradas según los criterios proporcionados.

Atributos
logArea (JTextArea): Área de texto donde se registran los resultados de las operaciones.
Constructor
El constructor de la clase inicializa el atributo logArea con el área de texto proporcionada.

Métodos
ejecutarListadoTarjetas(String linea): Procesa una línea de texto que contiene una solicitud para generar un listado de tarjetas.
Proceso:
Descompone la línea para extraer los filtros (tipo, límite, fecha inicial, fecha final, estado).
Construye la consulta SQL con los filtros aplicados.
Obtiene la conexión a la base de datos y prepara la consulta utilizando un PreparedStatement.
Ejecuta la consulta SQL y construye el contenido HTML con los resultados.
Genera el archivo HTML con el listado de tarjetas llamando al método generarArchivoHTML3.
Registra el éxito de la operación en logArea.
Si ocurre cualquier otra excepción durante el procesamiento, registra un mensaje de error en logArea y muestra la traza de la excepción.

Excepción: Maneja excepciones generales para asegurar que los errores se registren adecuadamente.
parseDate(String dateStr): Convierte una cadena de texto en un objeto Date.
Proceso:
Elimina comillas especiales de la cadena de texto.
Utiliza SimpleDateFormat para parsear la cadena de texto en el formato dd/MM/yyyy.
Devuelve el objeto Date correspondiente.
Excepción: Maneja excepciones ParseException para asegurar que los errores se registren adecuadamente.
generarArchivoHTML3(String htmlContent, String tipo, String limiteStr, String fechaInicialStr, String fechaFinalStr, String estado): Genera un archivo HTML con el listado de tarjetas.
Proceso:
Limpia comillas especiales de las fechas.
Genera el nombre del archivo basado en los filtros aplicados.
Obtiene la ruta seleccionada desde la clase EntradaArchivo.
Escribe el contenido HTML en un archivo utilizando un BufferedWriter.
Registra el éxito de la operación en logArea.
Excepción: Maneja excepciones IOException para asegurar que los errores se registren adecuadamente.
Componentes y Herramientas Usadas
JTextArea: Utilizado para mostrar y registrar los resultados de las operaciones.
Connection: Utilizado para establecer la conexión con la base de datos.
PreparedStatement: Utilizado para ejecutar consultas SQL parametrizadas.
ResultSet: Utilizado para almacenar los resultados de una consulta SQL.
BufferedWriter: Utilizado para escribir el contenido HTML en un archivo.
IOException: Excepción lanzada en caso de errores relacionados con la escritura de archivos.
SQLException: Excepción lanzada en caso de errores relacionados con la base de datos.
ParseException: Excepción lanzada en caso de errores relacionados con el parseo de fechas.
Exception: Excepción general manejada para capturar cualquier otro error durante el procesamiento.
Manejo de Excepciones
El método ejecutarListadoTarjetas maneja las excepciones generales mediante bloques try-catch, asegurando que los errores se registren en logArea y que el programa no se detenga si ocurre un error durante el procesamiento. El método generarArchivoHTML3 maneja las excepciones IOException y el método parseDate maneja las excepciones ParseException para asegurar que los errores se registren adecuadamente.

Ejemplo de Uso
Para procesar una solicitud para generar un listado de tarjetas, se crea una instancia de la clase operadorListadoTarjetas proporcionando un área de texto para los registros. Luego, se llama al método ejecutarListadoTarjetas con una línea de texto que contiene la solicitud.

Crear una instancia de operadorListadoTarjetas:
Inicializa el atributo logArea con el área de texto proporcionada.
Llamar al método ejecutarListadoTarjetas con una línea de texto:
Procesa la solicitud y registra los resultados en logArea.

Conclusión
La clase operadorListadoTarjetas proporciona una funcionalidad esencial para el procesamiento y registro de solicitudes para generar listados de tarjetas en el sistema bancario.

Clase operadorMovimiento
Descripción General
La clase operadorMovimiento se encarga de procesar las solicitudes de movimientos en tarjetas de crédito. Utiliza un área de texto (JTextArea) para registrar los resultados de las operaciones, incluyendo tanto los éxitos como los errores. Además, valida la tarjeta antes de registrar el movimiento en la base de datos.

Atributos
logArea (JTextArea): Área de texto donde se registran los resultados de las operaciones.
Constructor
El constructor de la clase inicializa el atributo logArea con el área de texto proporcionada.

Métodos
ejecutarMovimiento(String linea): Procesa una línea de texto que contiene una solicitud de movimiento.
Proceso:
Descompone la línea para extraer los parámetros del movimiento (número de tarjeta, fecha, tipo de movimiento, descripción, establecimiento, monto).
Valida la tarjeta llamando al método tarjetaValida.
Si la tarjeta es válida, crea una nueva instancia de Movimiento y guarda el movimiento en la base de datos.
Registra el éxito de la operación en logArea.
Si la tarjeta no es válida, registra un mensaje de error en logArea.
Si ocurre cualquier otra excepción durante el procesamiento, registra un mensaje de error en logArea y muestra la traza de la excepción.
Excepción: Maneja excepciones generales para asegurar que los errores se registren adecuadamente.
tarjetaValida(String numeroTarjeta): Valida si una tarjeta es válida y está activa.
Proceso:
Construye una consulta SQL para verificar el estado de la tarjeta.
Ejecuta la consulta SQL utilizando un Statement.
Si la tarjeta se encuentra y está activa, devuelve true.
Si la tarjeta no se encuentra o no está activa, devuelve false y registra un mensaje en logArea.
Excepción: Maneja excepciones SQLException para asegurar que los errores se registren adecuadamente.
Componentes y Herramientas Usadas
JTextArea: Utilizado para mostrar y registrar los resultados de las operaciones.
Connection: Utilizado para establecer la conexión con la base de datos.
Statement: Utilizado para ejecutar consultas SQL.
ResultSet: Utilizado para almacenar los resultados de una consulta SQL.

SimpleDateFormat: Utilizado para parsear fechas en el formato dd/MM/yyyy.
SQLException: Excepción lanzada en caso de errores relacionados con la base de datos.
ParseException: Excepción lanzada en caso de errores relacionados con el parseo de fechas.
Exception: Excepción general manejada para capturar cualquier otro error durante el procesamiento.
Manejo de Excepciones
El método ejecutarMovimiento maneja las excepciones generales mediante bloques try-catch, asegurando que los errores se registren en logArea y que el programa no se detenga si ocurre un error durante el procesamiento. El método tarjetaValida maneja las excepciones SQLException para asegurar que los errores se registren adecuadamente.

Ejemplo de Uso
Para procesar una solicitud de movimiento, se crea una instancia de la clase operadorMovimiento proporcionando un área de texto para los registros. Luego, se llama al método ejecutarMovimiento con una línea de texto que contiene la solicitud.

Crear una instancia de operadorMovimiento:
Inicializa el atributo logArea con el área de texto proporcionada.
Llamar al método ejecutarMovimiento con una línea de texto:
Procesa la solicitud de movimiento y registra los resultados en logArea.
Conclusión
La clase operadorMovimiento proporciona una funcionalidad esencial para el procesamiento y registro de solicitudes de movimientos en tarjetas de crédito en el sistema bancario.

Clase operadorSolicitud
Descripción General
La clase operadorSolicitud se encarga de procesar las solicitudes de creación de nuevas solicitudes de tarjetas de crédito. Utiliza un área de texto (JTextArea) para registrar los resultados de las operaciones, incluyendo tanto los éxitos como los errores.

Atributos
logArea (JTextArea): Área de texto donde se registran los resultados de las operaciones.
Constructor
El constructor de la clase inicializa el atributo logArea con el área de texto proporcionada.

Métodos
ejecutarSolicitud(String linea): Procesa una línea de texto que contiene una solicitud de creación de una nueva solicitud.
Proceso:
Descompone la línea para extraer los parámetros de la solicitud (fecha, tipo, nombre, salario, dirección).
Obtiene el último número de solicitud y lo incrementa en uno para asignar un nuevo número de solicitud.
Limpia las comillas de la fecha y la convierte a un objeto LocalDate.

Crea una nueva instancia de Solicitud con los parámetros extraídos y el estado inicial “PENDIENTE”.
Guarda la solicitud en la base de datos.
Registra el éxito de la operación en logArea.
Si ocurre cualquier otra excepción durante el procesamiento, registra un mensaje de error en logArea y muestra la traza de la excepción.
Excepción: Maneja excepciones generales para asegurar que los errores se registren adecuadamente.
Componentes y Herramientas Usadas
JTextArea: Utilizado para mostrar y registrar los resultados de las operaciones.
LocalDate: Utilizado para manejar fechas.
DateTimeFormatter: Utilizado para formatear y parsear fechas.
SQLException: Excepción lanzada en caso de errores relacionados con la base de datos.
ParseException: Excepción lanzada en caso de errores relacionados con el parseo de fechas.
Exception: Excepción general manejada para capturar cualquier otro error durante el procesamiento.
Manejo de Excepciones
El método ejecutarSolicitud maneja las excepciones generales mediante bloques try-catch, asegurando que los errores se registren en logArea y que el programa no se detenga si ocurre un error durante el procesamiento.

Ejemplo de Uso
Para procesar una solicitud de creación de una nueva solicitud, se crea una instancia de la clase operadorSolicitud proporcionando un área de texto para los registros. Luego, se llama al método ejecutarSolicitud con una línea de texto que contiene la solicitud.

Crear una instancia de operadorSolicitud:
Inicializa el atributo logArea con el área de texto proporcionada.
Llamar al método ejecutarSolicitud con una línea de texto:
Procesa la solicitud de creación de una nueva solicitud y registra los resultados en logArea.
Conclusión
La clase operadorSolicitud proporciona una funcionalidad esencial para el procesamiento y registro de solicitudes de creación de nuevas solicitudes de tarjetas de crédito en el sistema bancario. Facilita la creación de solicitudes y el registro de resultados, manejando adecuadamente las excepciones para asegurar una operación robusta.