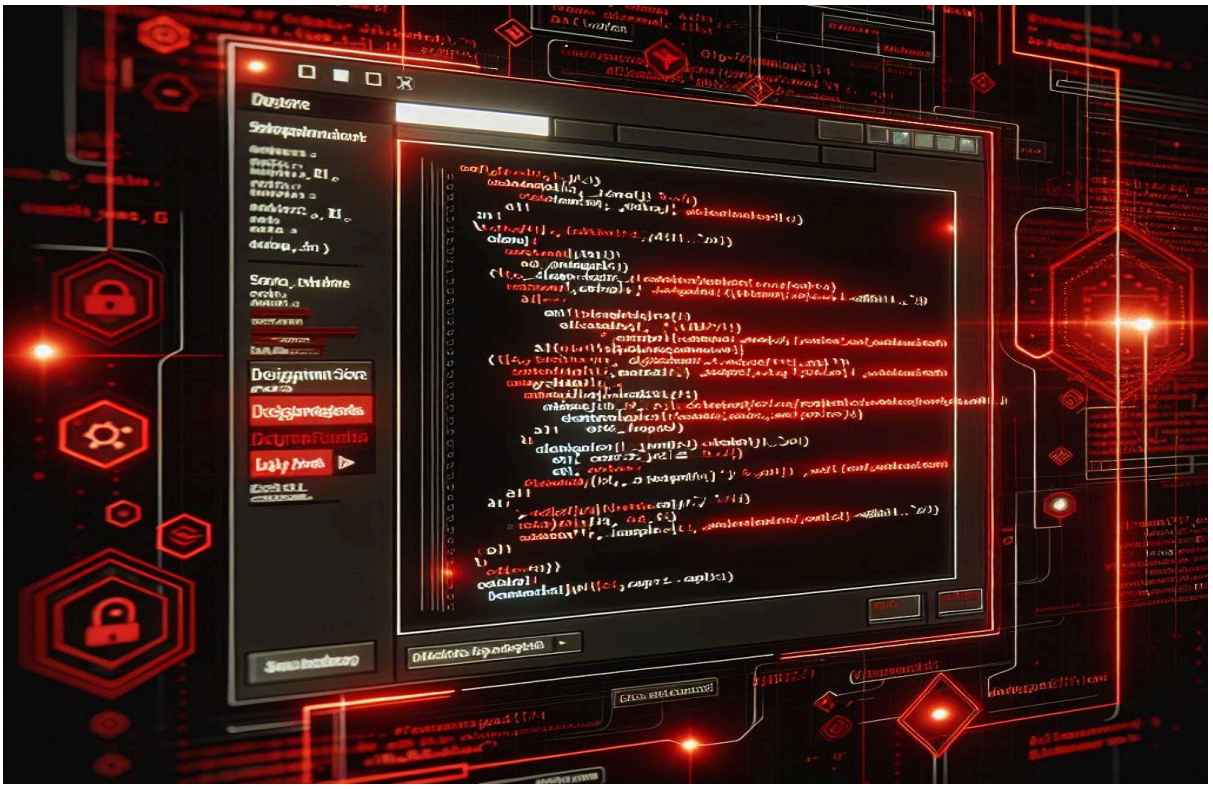

TECNOLÓGICO NACIONAL DE MEXICO

INSTITUTO TECNOLOGICO DE ZACATECAS

PROYECTO GESTOR INVENTARIO

MATERIA: TÓPICOS AVANZADOS DE PROGRAMACIÓN

DOCENTE: M.C.I.A JOSE MARTIN BARAJAS GUERRERO

CARRERA: INGENIERIA EN SISTEMAS COMPUTACIONALES

EQUIPO:

JOSE ALBERTO ESCAREÑO SANCHEZ 23450190

YOSELIN HERRERA GÓMEZ 23450200

Índice

EscribirEnBaseDeDatos.java

- La clase `EscribirEnBaseDeDatos` proporciona la funcionalidad para escribir datos en un archivo de texto que actúa como una base de datos simple.
- Comprueba si los datos ya existen en el archivo antes de agregarlos, y añade una marca de tiempo a cada entrada.
- **Librerías utilizadas:**
  - `javax.swing.JOptionPane` para mostrar mensajes al usuario.
  - `java.io.*` para operaciones de archivo.
  - `java.sql.Date` para manejar fecha y hora.
  - `java.text.SimpleDateFormat` para formatear fecha y hora.
- **Ejemplo de uso:**
  - `EscribirEnBaseDeDatos escribir = new EscribirEnBaseDeDatos("item1");`
- **Constructor:**
  - `EscribirEnBaseDeDatos(String in)`: Inicializa la clase con la cadena de entrada y la escribe en el archivo si no existe ya.
- **Parámetro:**
  - `in`: La cadena que se va a escribir en el archivo.

EscribirEnBaseDeDatosSalida.java

- La clase `EscribirEnBaseDeDatosSalida` es responsable de escribir los datos de salida del inventario en un archivo de texto.
- Añade la cadena de entrada proporcionada junto con la fecha y hora actuales al archivo "GestorInventarioSalidas.txt".
- Si el archivo no existe, crea uno nuevo.
- **Ejemplo de uso:**
  - `EscribirEnBaseDeDatosSalida escribir = new EscribirEnBaseDeDatosSalida("Item description");`
- **Parámetro:**
  - `in`: La cadena de entrada que se va a escribir en el archivo, que representa la descripción del artículo del inventario.
- **Excepciones:**
  - `NumberFormatException`: Si se produce un error al analizar números.
  - `IOException`: Si se produce un error de E/S al leer o escribir en el archivo.

EliminarDeBaseDeDatos.java

- La clase `EliminarDeBaseDeDatos` proporciona la funcionalidad para eliminar una línea específica de un archivo de texto que actúa como una base de datos.
- Lee el archivo original, copia todas las líneas excepto la especificada a un archivo temporal y, a continuación, reemplaza el archivo original con el archivo temporal.
- **Parámetro:**
    - `in`: La línea que se va a eliminar del archivo de base de datos.

Principal.java

- La clase `Principal` sirve como el **punto de entrada para la aplicación** `GestorDeInventarios`.
- Su función principal es **inicializar la ventana principal de la aplicación**.
- Crea una instancia de la clase `VentanaPrincipal`.

VentanaConsultar.java

- La clase `VentanaConsultar` proporciona una **interfaz gráfica de usuario (GUI) para buscar la disponibilidad de un `MAP`** (Material de Apoyo Promocional) en el inventario.
- **Lee de dos archivos**: `"GestorInventario.txt"` (entradas) y `"GestorInventarioSalidas.txt"` (salidas).
- Muestra un mensaje indicando si el `MAP` se encuentra en alguno de los archivos o si no se encuentra.
- **Componentes:**
    - `JFrame`: Ventana principal para la GUI.
    - `JPanel`: Contenedor para los componentes.
    - `JButton`: Botón para activar la acción de búsqueda.
    - `JTextField`: Campo de entrada para que el usuario introduzca el `MAP` a buscar.
- **Dependencias:**
    - `CustomFontLoader`: Clase auxiliar para cargar fuentes personalizadas.
    - `RoundedBorder`: Clase auxiliar para crear bordes redondeados para los componentes.
    - `OnClickEventHelper`: Clase auxiliar para manejar los cambios de color al hacer clic en los botones.

    - `OnFocusEventHelper`: Clase auxiliar para manejar los eventos de foco en los campos de texto.
- **ActionListener:**
    - El botón `"Registrar"` activa la acción de búsqueda.
    - Lee la entrada del campo de texto y busca el `MAP` en los archivos de inventario y salidas.
    - Muestra mensajes apropiados basados en los resultados de la búsqueda.
- **Constructor:**
    - Inicializa los componentes de la GUI y configura el `ActionListener` para el botón de búsqueda.
    - Cuando se hace clic en el botón "Buscar MAP", el texto ingresado en el campo `element2` se utiliza para buscar en los archivos "GestorInventario.txt" y "GestorInventarioSalidas.txt".
    - Si se encuentra el `MAP` en el archivo de inventario, se muestra un mensaje indicando que "La MAP se encuentra en Inventario".
    - Si se encuentra el `MAP` en el archivo de salidas, se muestra un mensaje indicando que "La MAP se encuentra en Salidas".
    - Si no se encuentra el `MAP` en ninguno de los archivos, se muestra un mensaje indicando "MAP no encontrada!!!".
    - Si no hay elementos en la base de datos, se muestra un mensaje indicando "No hay elementos en la base de datos".

VentanaPrincipal.java

- `VentanaPrincipal` es la **ventana principal de la aplicación** `Gestor de Inventarios PTS`.
- Inicializa el marco principal y sus componentes, incluyendo las listas de inventario y salidas.
- Proporciona botones para **registrar entradas, consultar disponibilidad, actualizar las listas y registrar salidas**.
- **Parámetro:**
    - `args`: Argumentos de la línea de comandos.
- Muestra una lista del inventario y una lista de salidas.
- El botón "Registrar Entrada" abre la ventana `VentanaRegistrarEntrada`.
- El botón "Consultar Disponibilidad" abre la ventana `VentanaConsultar`.
- El botón "Actualizar" actualiza las listas de inventario y salidas leyendo los datos de los archivos correspondientes.
- El botón "Registrar Salida" registra la salida de un elemento, lo elimina de la lista de inventario y lo añade a la lista de salidas.
- Utiliza `DefaultListModel` para gestionar las listas de inventario (`modeloLista`) y salidas (`modeloListaSalidas`).

VentanaRegistrarEntrada.java

- `VentanaRegistrarEntrada` es una clase que crea una **ventana GUI para registrar entradas de material**.
- La ventana contiene un campo de texto para la entrada y un botón para enviar la entrada.
- La entrada se procesa y se escribe en una base de datos al hacer clic en el botón o al presionar la tecla `Enter`.
- **Componentes:**
    - `JFrame`: El marco de la ventana principal.
    - `JPanel`: El panel que contiene todos los componentes.
    - `JTextField`: Un campo de texto para la entrada del usuario.
    - `JButton`: Un botón para enviar la entrada.
- **Características:**
    - Fuentes y colores personalizados para los componentes de la interfaz de usuario.
    - Bordes redondeados para el campo de texto y el botón.
    - Manejo de eventos de foco y clic para una mejor experiencia del usuario.
    - `ActionListener` para el botón para manejar el envío de la entrada.
    - `KeyListener` para el campo de texto para manejar la pulsación de la tecla `Enter` para el envío de la entrada.
- **Uso:**
    - Crea una instancia de `VentanaRegistrarEntrada` para mostrar la ventana.
    - El texto de entrada se escribe en la base de datos cuando se hace clic en el botón o se presiona la tecla `Enter`.
- El campo de texto `element2` permite al usuario ingresar el material a registrar.
- El botón "Registrar MAP" activa la acción de registrar el material ingresado.
- Al presionar la tecla `Enter` en el campo de texto, también se activa el registro del material.

Clases Helper

- `CustomFontLoader`: Carga fuentes personalizadas.
- `OnClickEventHelper`: Maneja los eventos de clic en los botones, cambiando su color al ser presionados.
- `OnFocusEventHelper`: Maneja los eventos de foco en los componentes de texto, como los `JTextField`, cambiando el texto y el color cuando el componente gana o pierde el foco.

- `RoundedBorder`: Implementa bordes redondeados para los componentes de la GUI.