Universidad de San Carlo de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Introducción a la programación y computación 1
Ing. Moisés Velásquez
Auxiliar. Pablo Oliva

PROYECTO No.1: MANUAL TECNICO

Guillermo Enrique Marroquin Morán

202103527

Guatemala 12 de ago. de 25

INDICE

Manual Técnico

Requerimientos mínimos del programa.

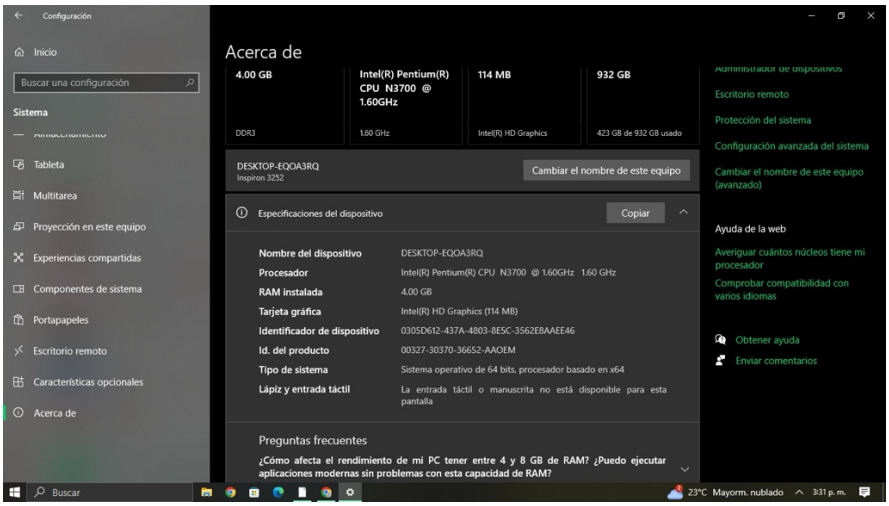
Para instalar apache NetBeans los requisitos mínimos son:

- 781 MB de Espacio Libre en el Disco Duro.
- 512 MB de RAM.
- Procesador Intel Pentium III a 800 MHz.
- Compatible con Windows, macOS y Linux.

(Uanl, 2022)

Requisitos del ordenador usado para el proyecto:

Requisitos del ordenador con sistema Windows:



Explicación de las partes del proyecto.

Librerías usadas

```
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.InputMismatchException;
import java.util.Scanner;
```

- java.time.LocalDateTime, nos permite registrar la hora del ordenador, el uso que se le dio fue para los nombres de los archivos pdf´s, y para registrar la hora y fecha en las que se realizó las acciones.
- java.time.format.DateTimeFormatter, nos permite agregar un formato para la hora y fecha en mi caso fue el formato para fecha (DD-MM-YYYY) y de hora (HH-MM-SS).
- java.util.InputMismatchException, nos permite evitar que haya algún error al “leer” un valor que no sea, el establecido, en mi caso fue valores numéricos positivos (los esperados) y los demás que no lo son, nos permite que el programa no se bloquee.
- java.util.Scanner, nos permite que el usuario ingrese algo, sea números, cadena de texto, etc. Que en mi caso fue muy esencial para que el usuario ingrese, los productos, confirme o no algo, que reingrese valores entre otros.

```
import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.io.FileWriter;
import java.io.IOException;
```

- java.io.File, nos permite la manipulación de archivos, ósea; crear, eliminar, renombrarlos, moverlos, ver que existan o no, en el proyecto se usó para la generación de los archivos de texto de la bitácora y el registro de ventas.
- java.io.FileNotFoundException, no ayuda a crear excepciones si en dado caso no exista un archivo, junto a “try-catch” ayuda a que el programa no se bloquee y pues habría que reiniciar.
- java.io.PrintWriter, nos ayuda a poder escribir o “imprimir” en archivos, con el código, pw.println(“palabras”), nos deja poder escribir en el archivo desde el programa.
- java.io.FileWriter, nos permite abrir el archivo, desde el programa hasta donde se encuentre (normalmente se encuentra por su nombre) en este caso fue el archivo de ventas (venta.txt) y la bitacora (bitácora.txt).
- java.io.FileReader, este no se ve en este bloque, pero este nos permite leer el archivo, en este caso es muy útil para generar el archivo pdf del reporte de ventas.

```
//Liberias de itext7
import com.itextpdf.kernel.pdf.PdfDocument;
import com.itextpdf.kernel.pdf.PdfWriter;
import com.itextpdf.layout.Document;
import com.itextpdf.layout.element.Paragraph;
import com.itextpdf.layout.element.Table;
```

Todas estas librerías no son oriundas de java como tal, estas se deben descargar por aparte, eso explica que empiecen con “com” y no como “.java”.

- com.itextpdf.kernel.pdf.PdfDocument, con esto “invocamos” el archivo pdf como tal, para luego manipularlo.
- com.itextpdf.kernel.pdf.PdfWriter, con esto creamos el archivo en sí, sin el nombre como tal, eso ya se encarga con otros comandos.

- com.itextpdf.layout.element.Paragraph y com.itextpdf.layout.element.Table, nos permiten agregar contenido, como su nombre lo dice, son elementos como tablas y párrafos, para el reporte de stock que usamos tablas y párrafos.

```
//Las Importaciones de las clases que hice por aparte
import static proyecto1.Productos.EliminarProd;
import static proyecto1.Productos.RegisVenta; // aca vienen tambien las liberias de formatos y la de la excepcion
import static proyecto1.Productos.buscarProducto;//Lo importe porque... si ponia el codigo esto se haria muy largo :(
import static proyecto1.Reportes.generarPDF;//ahoa si xddd
import static proyecto1.ValidarAccion.LimpHistorial;// para limpiar el historial
import static proyecto1.ValidarAccion.MostarAcciones;//Esto es para  ver la bitacora
import static proyecto1.ValidarAccion.VerAccion;// y esto es para la funcion de la bitacora
```

Como tal esas no son librerías, pero son llamadas a las clases, y estas clases fueron hechas en script diferentes que se encuentran en el paquete, llamado proyecto1. Pero se ve cool ponerlo aquí.

Módulos usados.

Modulo de buscar productos.

```
public static void buscarProducto (String[][] Inventario, int CantInventario, Scanner sc, String vendedor) {
    int optBuscar;
    String valBuscar;
    boolean encontrar = false;
    System.out.println("Bienvenido a la seccion de busqueda de Productos ");
    System.out.println();//sip, un salto de linea pa que se vea mejor
    System.out.println("¿Como desea buscar el producto'");
    System.out.println("1. Por Codigo");
    System.out.println("2. Por Nombre");
    System.out.println("3. Por Categoria");
    System.out.println("Ingrese su opcion:");
    try {
        optBuscar = sc.nextInt();
        sc.nextLine();
    } catch (InputMismatchException e) {
        System.out.println("Error  006: Debe de ingresar un numero valido, intente de nuevo");
        sc.nextLine();
        VerAccion (vendedor, "Busqueda de productos, opcion invalida ", "fallida");
        return;
    }
    switch (optBuscar) {
        case 1:
            System.out.println("Ingrese el codigo a buscar:");
            valBuscar = sc.nextLine();
            for (int i = 0; i < CantInventario; i++) {
                if (Inventario[i][0] != null && Inventario[i][0].equalsIgnoreCase(valBuscar)) {
                    mostrarProd (Inventario, i);
                    encontrar = true;
                    break;
                }
            }
            break;
        case 2:
            System.out.println("Ingrese el nombre a buscar:");
            valBuscar = sc.nextLine();
            for (int i = 0; i < CantInventario; i++) {
                //esto va a ingnorar las mayusculas y minusculas, solo toma las palabras
                if (Inventario[i][1] != null && Inventario[i][1].toLowerCase().contains(valBuscar.toLowerCase())) {
                    mostrarProd (Inventario, i);
                    encontrar = true;
                }
            }
            break;
        case 3:
            System.out.println("Ingrese la categoria a buscar:");
            valBuscar = sc.nextLine();
            for (int i = 0; i < CantInventario; i++) {
                //Igual aca, solo que.. bueno hace una equivalencia entre los productos ingresado y los que encuentre, solo que ignora las mayusculas
                if (Inventario[i][2] != null && Inventario[i][2].toLowerCase().equals(valBuscar.toLowerCase())) {
                    mostrarProd (Inventario, i);
                    encontrar = true;
                }
            }
            break;
        default:
            System.out.println("Opcion no valida, intente otra vez");
            VerAccion (vendedor, "busqueeda de producto, opcion invalida ", "fallida");
            return;
    }
    if (!encontrar) {
        System.out.println("No se encontraron productos que coincidan con la busqueda");
        VerAccion (vendedor, "Busqueda de producto, no se encontró el producto ", "fallida");
    } else {
        VerAccion (vendedor, "Busqueda de producto", "Correcta");
    }
}
```

En este bloque de código usado para la opción 2, es buscar en la matriz inventario por el elemento que estemos buscando, si es el caso de código, buscará en la matriz por el código, si es por el nombre será por el nombre y por categoría será por categoría, en este último hay que escribir bien pantalón con su tilde que, si no encontrara nada, se hace llamado del método/modulo/función de mostrar productos, que explicaré abajo.

Módulo de mostrar productos.

```java
//Este metodo ayuda a visualizar los porductos, igual me base de la practica... sip mi creatividad llego a 0
public static void mostrarProd (String[][] Inventario, int i) {
    System.out .println("Se ha encontrado el producto");
    System.out .println("Codigo: " + Inventario[i][0]);
    System.out .println("Nombre: " + Inventario[i][1]);
    System.out .println("Categoria: " + Inventario[i][2]);
    System.out .println("Precio: " + Inventario[i][3]);
    System.out .println("Stock: " + Inventario[i][4]);
}
```

Este bloque de código muy creativamente nombrado mostrar producto, nos permite visualizar el producto que estamos buscando, este módulo lo uso en las opciones 2, 3 y 4 para mostrar en pantalla el producto solicitado, como los productos se guardan en una matriz esta va por posiciones desde i hasta el producto, (las matrices empiezan en 0, por eso en código sale 0).

Modulo Eliminar Producto.

```java
//Sip un Metodo que sea para buscar productos exclusivamante para eliminarlos, me base mucho en la de la practica yeeey
public static int EliminarProd (String[][] Inventario, int CantInventario, Scanner sc, String vendedor) {
    System.out .println("Ingrese el Codigo del producto que se quiera eliminar:  ");
    String codEli = sc.nextLine();
    int Eliminar = -1;
    //Usamos un bucle for para buscar en la matriz inventario para encontrar el producto
    for (int i = 0; i < CantInventario; i++) {
        if (Inventario[i][0] != null && Inventario[i][0].equalsIgnoreCase(codEli)) {
            Eliminar = i;
            break;
        }
    }
//Si el producto fue encontrado
    if (Eliminar != -1) {
        System.out .println("Se encontro el siguiente producto: ");
        //Hacemos un llamado para mostrar el producto a eliminar
        mostrarProd (Inventario, Eliminar);
        System.out .println();
        System.out .println("¿Esta seguro de la eliminacion del producto (s/n): ");
        String Confirmar = sc.nextLine().toLowerCase();

        if (Confirmar.equals("s")) {
            //Eliminamos la posicion donde se encuentre el producto
            for (int i = Eliminar; i < CantInventario - 1; i++) {
                Inventario[i] = Inventario[i + 1];
            }
//Este bloquecito de codigo bonito es para "limpiar" la ultima fila para que no haya duplicaciones
            Inventario[CantInventario - 1] = new String[5];
            System.out .println("El producto fue eliminado ");
            VerAccion (vendedor, "Eliminacion del producto con el codigo:  " + Eliminar, "Correcta");
            CantInventario--;//baja el "contenido" del inventario una unidad UWU
        } else {
            System.out .println("El producto no fue eliminado ");
            VerAccion (vendedor, "Eliminacion del producto con el codigo:  " + Eliminar, "fallida");
        }
    } else {
        System.out .println("El producto: " + codEli + "no fue encontrado");
        VerAccion (vendedor, "Eliminacion del producto, producto no encontrado ", "fallida");
    }
    return CantInventario;//devolvemos el valor de la nueva matriz de inventario
}
```

Al igual que en la práctica, nos solicitaban eliminar algo, acá lo que hacemos es buscar en la matriz de inventario por el producto que queremos buscar, declaramos una variable muy creativamente nombrada ELIMINAR, con el valor de -1 si esta no retorna un valor que no es ese el programa no encontró nada, de lo contrario de encontrarlo mostrara el producto con el módulo de mostrar producto (que versátil por dios), para luego ser o no eliminado, de ser eliminado, se “limpia” la columna donde se encontraba sus datos, y luego “corremos” todo una “casilla arriba” para que no hayan productos compartiendo posición, luego devolveos el valor de la nueva matriz de inventario con la casilla corrida y el producto eliminado.

Módulo de Registrar ventas

```
        // Actualizamos el nuevo stock con la cantidad vendida
        int Stock3 = Stock2 - cantidad;
        Inventario[Prod][4] = String.valueOf (Stock3);

        //calciulamos el total de la venta :3
        double precio = Double.parseDouble (Inventario[Prod][3]);
        double Total = precio * cantidad;

        //esto es para la fecha y hora de la ventas yuju :3
        LocalDateTime actual = LocalDateTime.now ();
        DateTimeFormatter formato = DateTimeFormatter.ofPattern ("DD-MM-YYYY HH-:mm:ss");
        String FechaHora = actual.format(formato);

        // Registramos la venta en un archibo de texto
        try {
            FileWriter fw = new FileWriter("Venta.txt", true);
            PrintWriter pw = new PrintWriter(fw);
            pw.println("--------------------");
            pw.println("Ventas Efectuadas: ");
            pw.println("Hora y Fecha de la transacción: " + FechaHora);
            pw.println("Producto vendido: " + Inventario[Prod][1]);
            pw.println("Cantidad vendida del producto es : " + cantidad);
            pw.println("el total a pagar es de: " + Total + " Qeutzales");
            pw.println("----------------------");
            pw.close();
        } catch (IOException e) {
            System.out .println("Error 007: ocurrio un error al registrar la venta en el archivo");
            e.printStackTrace();
        }
        System.out .println("La venta fue efectuada");
        System.out .println("Fueron vendidas: " + cantidad + " unidades de :" + Inventario[Prod][1]);
        System.out .println("La cantidad actual del producto es: " + Stock3);
        System.out .println("La cantidad a pagar es :" + Total + "Quetzales");
        VerAccion (vendedor, "venta de:  " + cantidad + "unidades " + Inventario[Prod][1], "corecta");
    } else {
        System.out .println("Error 008: el producto con el codigo : " + CodVenta + " No fue encontrado");
        VerAccion (vendedor, "Venta del producto, producto no encontrado ", "fallida");
    }
  }
}
```

```
public static void RegisVenta (String[][] Inventario, int CantInventario, Scanner sc, String vendedor) {
    int Prod = -1;
    System.out .println("Ingrese el codigo del producto a vender: ");
    String CodVenta = sc.nextLine();
    for (int i = 0; i < CantInventario; i++) { // Buscmos en el inventario por el codigo con un ciclo for
        if (Inventario[i][0] != null && Inventario[i][0].equalsIgnoreCase(CodVenta)) {
            Prod = i;
            break;
        }
    }
    if (Prod != -1) {// Esto es porque si  en dado caso el indice a buscar es -1, por la posicion -1 de la matriz esto no va afuncionar
        mostrarProd (Inventario, Prod);

        System.out .println("Ingrese la cantidad del produco que se va a vender");
        int cantidad = Validar.verNumPos (sc);
        int Stock2 = Integer.parseInt (Inventario[Prod][4]);

        //Validadmos que la venta sea exitosa
        if (cantidad > Stock2) {
            System.out .println("Error 006: La cantidad a vender es superior al stock");
            VerAccion (vendedor, "Venta, Stock INSUFICIENTE ", "fallida");
            return;
        }
```

En este método/función lo que sea, hacemos lo mismo de buscar en la matriz de inventario por el código del producto que deseamos buscar, si este se encuentra y nos devuelve -1, proseguimos evaluando si la cantidad que deseamos vender es igual o menor al stock disponible, si en dado caso no pasa, hacemos que vuelva a ingresar el valor, luego, declaramos otra variable de stock, que será como un llamado al stock temporal que será el valor convertido a entero directamente de la matriz de inventario, luego llamaos a otra variable que será el stock restante de la venta, llamamos a otra variable de precio, igual convertida a entero de la matriz de inventario, declaramos una variable del total de la venta que es el valor del producto*cantidad(no pidieron calcular impuestos obviamente no lo hice), luego hacemos llamado de la librería para crear el documento de texto y luego llenarlo con los datos de la compra, invocando el nombre del producto directamente de la matriz de inventario, y la cantidad vendida (en quetzales), luego si en dado caso no encuentra el archivo el programa no se va a bloquear por la librería de excepción para archivos, luego el ultimo bloque de código, es para imprimir en pantalla lo que fue vendido, ya se la saben, la cantidad vendida, el producto vendido, la cantidad actual del stock, y abajo un llamado a un método que luego explicare.

Módulo de Reportes.

```java
public class Reportes {
    //Sip un metodo para generar reportes, (no lo iba a poner en la de productos que desmadre seria)
    //dios que miedo

    public static void generarPDF(String[][] Inventario, int CantInventario, Scanner sc, String vendedor) {
        System.out.println("¿Qué reporte se desea generar?");
        System.out.println("1. Reporte de Stock");
        System.out.println("2. Reporte de ventas");
        System.out.println("Ingresa tu opcion: ");

        try {
            int optRep = sc.nextInt();
            sc.nextLine();

            switch (optRep) {
                case 1:
                    ReporteStock(Inventario, CantInventario, vendedor);
                    break;
                case 2:
                    ReporteVenta(vendedor);
                    break;
                default:
                    System.out.println("Opcion no es valida, intente de nuevo");
                    VerAccion(vendedor, "Generar reporte, opcion no valida ", "fallida");
            }
        } catch (InputMismatchException e) {
            System.out.println("Error 009: Debes de ingresar un numero valido, intente de nuevo");
            sc.nextLine();
            VerAccion(vendedor, "Generar reporte, valor ingresado no valido ", "fallida");
        }
    }
```

Este módulo trabaja para decirlo fácilmente como un menú principal donde hago las llamadas a sus respectivos módulos, si es el módulo de reporte de stock ese llamara y si es de reporte de ventas pos ese va a llamar, es un menú sencillito, con su validación con el ya famoso try catch, para evitar que el programa fallezca.

Módulo de reporte de stock.

```
public static void ReporteStock (String[][] Inventario, int CantInventario, String vendedor) {
    LocalDateTime Hactual = LocalDateTime.now ();
    //formato de dias_meses_año, Hora_Minuto_ segundo
    DateTimeFormatter formato = DateTimeFormatter.ofPattern ("dd-MM-YYYY_HH_mm_ss");
    //nombramos al archivo :3
    String nomArchi = Hactual.format(formato) + "_Stock.pdf";

    try {
      PdfWriter Esc = new PdfWriter(nomArchi);
      PdfDocument pdf = new PdfDocument(Esc);
      Document docu = new Document(pdf);
      docu.add(new Paragraph("Reporte de Stock - " + Hactual.format(DateTimeFormatter.ofPattern ("dd/MM/yyyy HH:mm:ss"))));
//Creamos celdas con el el atributo de cada producto
      Table tabla = new Table(new float[]{1, 2, 2, 1, 1});
      tabla.addCell("Codigo");
      tabla.addCell("Nombre");
      tabla.addCell("Categoría");
      tabla.addCell("Precio");
      tabla.addCell("Stock");

//LLenamos las celdas recorriendo con el ciclo for, lo productos con sus atributos registrados
      for (int i = 0; i < CantInventario; i++) {
        if (Inventario[i][0] != null) {
          tabla.addCell(Inventario[i][0]);
          tabla.addCell(Inventario[i][1]);
          tabla.addCell(Inventario[i][2]);
          tabla.addCell("Q. " + Inventario[i][3]);
          tabla.addCell(Inventario[i][4]);
        }
      }
      docu.add(tabla);
      docu.close();
      //Se hace aviso de que el reporte si se creo, o bueno eso espero que funcione
      System.out .println("El reporte de stock fue generado en :" + nomArchi);
      VerAccion (vendedor, "Generar reporte de stock ", "Correcta");
    } catch (FileNotFoundException e) {
      System.out .println("Error 010: no se pudo crear el archivo:");
      VerAccion (vendedor, "Generar reporte de stock , no se encontro el archivo", "fallida");
      e.printStackTrace();
    } catch (Exception e) {
      System.out .println("Error 011: ocurrio un problema al general el reporte");
      e.printStackTrace();
      VerAccion (vendedor, "Generar reporte de stock ", "fallida");
```

Empezamos llamado las librerías para la hora del dispositivo y para el formato de la misma, luego en el bloque de try catch, llamamos al documento pdf para crearlo, le agregamos los párrafos para luego poder agregar la tabla con las celdas para: código, nombre categoría, precio y stock, y llenamos las mismas con los datos llamando a sus valores directamente de la matriz(i siendo el número de producto y el valor de 0 a 4 las columnas que ocupa cada dato), el código "*Table tabla = new Table(new float[]{1, 2, 2, 1, 1});*" indica que la tabla va a explicar que el ancho de la tabla es 7, para que cada apartado de cada dato quepa allí (1+2+2+1+1=7) podría ser más grande pos sí. Luego de eso añadimos la tabla en el documento para luego cerrarlo, para luego imprimir en pantalla que el documento fue guardado con el nombre de la hora y fecha seguido de stock.pdf, luego con el catch hacemos una excepción para evitar que el programa se bloquee si no se pudo crear el archivo, con la llamada de modulo para registrar acción que luego explicare.

Módulo de Reporte de Ventas.

Como siempre hacemos llamado de la librería de la hora local y del formato de la misa, y colocamos el nombre del archivo que es la hora y fecha_ventas.pdf, en el bloque "try-catch" hacemos llamado para crear el documento en pdf, para escribir en él y colocarlo en la carpeta del proyecto, y hacemos algo diferente, hacemos llamado del *file reader* para leer el documento creado anteriormente de ventas, y agregarle el contenido del archivo de texto al archivo en pdf, luego creamos el título de "historial de ventas", imprimimos el mensaje que si se pudo crear el archivo en pdf y luego cerramos el documento, con el catch agregamos una excepción si en dado caso no encontramos el archivo de texto y con otro catch agregamos una nueva excepción para si en dado caso no se genere el documento y el programa no se bloquee.

```java
//Una funcion para el reporte de ventas, asi mejor, y no se ve todo desordenado
public static void ReporteVenta (String vendedor) {
    LocalDateTime HFactual = LocalDateTime.now ();
    //ya lo dije arriba :P
    DateTimeFormatter HFormato = DateTimeFormatter.ofPattern ("dd_MM_yyyy_HH_mm_ss");
    //nombramos al otro archivo
    String nomArchiv = HFactual.format(HFormato) + "_Ventas.pdf";

    try {
        // se ordeno muy feo... asi que se vaya
        // aca declaro las variables de escribir en el archivo, de documento, de lectura y escrituta en archivo de texto
        PdfWriter Escribir = new PdfWriter(nomArchiv);
        PdfDocument pdf = new PdfDocument(Escribir);
        Document documento = new Document(pdf);
        Scanner Fs = new Scanner(new FileReader("Venta.txt"));
        {
        }
        documento.add(new Paragraph("Historial de Ventas - " + HFactual.format(DateTimeFormatter.ofPattern ("dd7mm/yyyy HH:mm:ss"))));
        while (Fs.hasNextLine()) {
            documento.add(new Paragraph(Fs.nextLine()));
        }
        documento.close();
        System.out .println("Reporte de venta generdado en : " + nomArchiv);
        VerAccion (vendedor, "Generar reporte de ventas ", "Correcta");
    } catch (FileNotFoundException e) {
        System.out .println("Error 012: el archivo de ventas.txt no se encontró, no hay ventas por reportar");
        VerAccion (vendedor, "Generar reporte de venta, no se encontro el archivo ", "fallida");
    } catch (IOException e) {
        System.out .println("Error 013: ocurrio un error de lectura o escritura");
        VerAccion (vendedor, "Generar reporte de ventas ", "fallida");
        e.printStackTrace();
    }
  }
}
```

Módulo de Registrar y mostrar acciones.

```java
public class ValidarAccion {
   // una funcion para registrar cada accion e imprimirlas en un archivo de texto

   public static void VerAccion (String vendedor, String accion, String estado) {
      try (FileWriter Fw = new FileWriter("Bitacora.txt", true); PrintWriter pw = new PrintWriter(Fw)) {

         LocalDateTime Ahora = LocalDateTime.now ();
         DateTimeFormatter Formato = DateTimeFormatter.ofPattern ("dd_mm_yyyy HH:mm:ss");
         String fechaHora = Ahora.format(Formato);

         pw.println("-----------------------------------------------");
         pw.println("--------BITACORA DE ACCIONES--------");
         pw.println("Fecha y Hora: " + fechaHora);
         pw.println("Usuario: " + vendedor);
         pw.println("Accion :" + accion);
         pw.println("Estado :" + estado);
         pw.println("-----------------------------------------------");

      } catch (IOException e){
         System.out .println("Error 013: Ocurrio un error al registrar la accion en la bitaora");
      }
   }
   //funcion para mostar la Bitacora
   public static void MostarAcciones (){
      System.out .println("-------Historial de acciones----------");
      try(Scanner Fs = new Scanner(new File("Bitacora.txt"))){
         while(Fs.hasNextLine()) {
            System.out .println(Fs.nextLine());
         }
      } catch(FileNotFoundException e){
            System.out .println("Error 014: No se ha encontrado el archivo de la bitacora");
      }
   }
```

Con este nuevo módulo, hacemos llamado de crear archivo, guardarlo, y escribir en él, con el formato de hora y fecha del dispositivo, con el pw.println(), imprimimos en el documento lo que queremos saber, que es la hora y fecha, quien hizo la acción (yop), la acción que se realizó y si salió bien o mal, para luego, si en dado caso, no se pudo crear el archivo que el programa no se bloquee.

El módulo de mostrar acciones permite la visualización, de todas las acciones en la consola, leyendo desde el archivo de texto que se llama bitácora.

En cada acción realizada por el usuario se hace llamado al módulo y en TODOS LOS MODULOS agregue el parámetro “vendedor” para que registre el nombre en cada acción para imprimirla en el documento de texto y en la consola.

Módulo de Limpiar Historial

Con el “*new FileWriter("Bitacora.txt", false).close();*” indicamos, que el archivo con el nombre “Bitácora.txt” indicamos que el *false* es para que se sobrescriba, el archivo desde 0 en lugar de agregarle algo, igual con el archivo de texto, al sobrescribirlo de 0 sin agregar nada “limpiar “el archivo de texto”, con su validación de si en dado caso no encontrarlo que el programa no se bloquee.

```java
    //Pense que como TECNICAMENTE los archivos son temporales....
//me saque del coco una forma para limpiar los archivos de texto, me regañaran? lo dudo
   public static void LimpHistorial (){
      try {
        //  limpiar la bitacora
        new FileWriter("Bitacora.txt", false).close();
        System.out .println("La bitacora fue limpiada :D");

        //Limpiar las ventas
        new FileWriter("Venta.txt", false).close();
        System.out .println("Historial de ventas fue limpiado :D");
      } catch(IOException e){
         System.out .println("Erro 015: Ocurrio un error al intentar limpiar los archivos");
      }
   }
}
```

Módulo de verifica número y numero positivo

```java
public static int VerificarNum (Scanner sc) throws InputMismatchException {
    if (!sc.hasNextInt()) {
        throw new InputMismatchException("Error 002: Por favor ingresa un numero del 1 al 8");
    }
    return sc.nextInt();
}
//Un nuevo metodo para ver si la cantidad de estock y lo otro sea positivos, o en general numeros vaya
public static int verNumPos (Scanner sc) {
    int num;
    while (true) {
        try {
            num = sc.nextInt();
            if (num < 0) {
                System.out .println("Error 003: por favor ingrese un valor positivo");
                System.out .println("Por favor elija un numero postivo");
                System.out .println();
            } else {
                sc.nextLine();
                return num;
            }
        } catch (InputMismatchException e) {
            System.out .println("Error 004: por favor ingrese un valor numerico que sea valido");
            sc.nextLine();
        }
    }
}
```

En este módulo usando el "*hasNextInt"* indica que, si se recibe un valor no indicado, crea una excepción para que el programa no se cierre,

El módulo de ver número positivo, en un bucle "do-While" y un "try-catch", para leer el numero ingresado y crear una excepción para que el programa se cierre, e imprima en consola que ingrese un valor valido.

SIP lo use bastante.