MANUAL TÉCNICO

Requerimientos

- Java JDK 24 o superior
- Biblioteca iTextPdf versión 5.5.9
- NetBeans para un desarrollo mucho mas simple.

Estructura del proyecto

El proyecto se encuentra desarrollado en un archivo principal denominado **`Principal.java`** en el paquete org.aaronmatta.system.

Variables globales de interés:

```
String[][] inventario = new String[100][5];
String[][] ventas = new String[100][3];
String[][] bitacora = new String[200][5];
String[] categorias = { "Camisetas y blusas", ... };
```

Los vectores y matrices representan el almacenamiento del todo el programa en si.

- **Inventario:** registra código, nombre, categoría, precio y stock
- **Ventas:** registra detalles de venta, fecha y total
- **Bitácora:** registra eventos con fecha, tipo y el mensaje.

Descripción de los métodos principales

Descripción y funcionamiento de los métodos más relevantes/importantes dentro del programa.

Métodos iniciales

`public void menu()`

Punto donde empieza el programa, creando una instancia de Principal para poder inicializar estructuras y mostrar el menú.

```
public static void main(String[] args) {
    Principal programa = new Principal();
    programa.generarEspaciosVaciosInventario();
    programa.menu();
}
```

`public void menu()`

Controla el flujo principal de todo el programa por consola y llama a las funciones correspondientes que indique el usuario, es el controlador de interacción.

Pide datos de usuario (nombre y carnet), presenta un menú con opciones a elegir del 1 al 8 y se realiza un switch sobre la opción elegida para ejecutar diferentes acciones (agregar, buscar, eliminar, registrar venta, generar reportes, ver datos, ver bitácora, salir).

```
nombreUsuario = ingresarTexto("*    Nombre: ");
carnetUsuario = String.valueOf(ingresarEntero("*    Carnet: "));
...
opcion = ingresarEntero("*    Elige una opcion (1-8): ");
switch(opcion){
    case 1: agregarProducto(...); break;
    case 4: registrarVenta(); break;
    // ...
}
```

Métodos de inicialización

`public void generarEspaciosVaciosInventario()`

Inicializa la matriz de inventario marcando todas las filas vacías con -100 en la columna 0 y VACIO en las restantes.

```
for (int fila=0;fila<100;fila++){
    inventario[fila][0] = "-100";
    for (int col=1;col<5;col++) inventario[fila][col] = "VACIO";
}
```

`public String[][] generarEspaciosVaciosCarrito()`

Crea/limpia la estructura de carrito de ventas usando el mismo patrón que la funcion generarEspaciosVaciosInventario().

Métodos de entrada y validación

`public int ingresarEntero(String mensaje)`

Lee un entero desde la consola y valida que su formato sea el correcto, en caso de error se reintentará este proceso hasta que el formato sea el adecuado.

```
public int ingresarEntero(String mensaje) {
    int num = 0;
    boolean valido = false;

    do {
        try {
            System.out.print(mensaje);
            String input = scanner.nextLine();
            num = Integer.parseInt(input);
            valido = true;
        } catch (NumberFormatException e) {
            System.out.println("[?]    ERROR: Debe ingresar un número entero
válido. Porfavor ingresar un número válido.");
        }
    } while (!valido);

    return num;
}
```

`public String ingresarTexto(String mensaje)`

Lee una cadena y valida que contenga solo letras (incluye acentos y ñ) y espacios.

```
if (!texto.isEmpty() && texto.matches("[a-zA-ZáéíóúÁÉÍÓÚñÑ\\s]+")) {
    valido = true;
}
```

`public double ingresarDecimal(String mensaje)`

Lee un número decimal, lo valida y redondea a dos decimales con:

```
Math.round(decimal * 100.0) / 100.0
```

Métodos de gestión de productos

`public void agregarProducto(String nombre, String categoria, double precio, int cantidadStock)`

Agrega un nuevo producto al primer espacio libre del inventario.

Verifica si hay espacio disponible, que precio y stock sea positivos con **`validarPositivo(…)`**. Genera un código con **`generarCodigoUnicoProducto()`** y actualiza el inventario.

```
codigoUnico = generarCodigoUnicoProducto();
for(int fila=0;fila<100;fila++){
    if(inventario[fila][0].equals("-100")){
        inventario[fila][0] = String.valueOf(codigoUnico);
        inventario[fila][1] = nombre;
        inventario[fila][2] = categoria;
        inventario[fila][3] = String.valueOf(precio);
        inventario[fila][4] = String.valueOf(cantidadStock);
        return;
    }
}
```

`public void verProducto(int opcion, String textoIngresado)`

Busca un producto ya sea por su código, nombre o categoría e imprime los resultados en consola que coincidan con el filtro establecido.

```
switch(opcion){
    case 1:
        for (int fila=0;fila<100;fila++){
            if(inventario[fila][0].equals(String.valueOf(textoIngresado))){ //Si el CODIGO es encontrado muestra los datos en esa posicion
                System.out.println("-    Código: "+inventario[fila][0]);
                System.out.println("-    Nombre: "+inventario[fila][1]);
                System.out.println("-    Categoría: "+inventario[fila][2]);
                System.out.println("-    Precio: "+inventario[fila][3]);
                System.out.println("-    Stock: "+inventario[fila][4]);
                contador++;
                break;
            }
        }
        if(contador==0){
            System.out.println("[?]   NO SE HA ENCONTRADO NINGUN PRODUCTO.");
            break;
        }
        break;
    case 2:
        //Buscar por nombre...
    case 3:
        //Buscar por categoria...
```

`public void eliminarProducto(int codigo)`

Busca y solicita un producto al usuario, una vez se confirma y coincide, marca la fila vacia con (-100 y VACIO) y llama a **`ordenarProductos()`** para compactar el inventario y dejar los espacios vacíos hasta de último.

```
inventario[fila][0] = "-100";
inventario[fila][1] = "VACIO";
ordenarProductos();
```

Métodos de ventas

`public void registrarVenta()`

Interactúa con el usuario para crear y gestionar un carrito de compras, agrega los productos verificando el stock, finaliza la venta y registra la transacción en ventas.

Se crea `carrito` con **`generarEspaciosVaciosCarrito()`**, permitiendo agregar productos por código, si ya están en el carrito actualiza la cantidad y su monto, al finalizar se reduce el stock en `inventario` y guarda la venta en `ventas` con fecha y total.

```
int existeId = buscarProducto(codigoUnico);
if(existeId!=-100){
    cantidad = ingresarEntero("*    Cantidad: ");
    if(validarStock(existeId, cantidad)){
        monto = Double.parseDouble(inventario[existeId][3])*cantidad;
        total = total + monto;
        carrito[contador][0] = inventario[existeId][0];
        carrito[contador][1] = inventario[existeId][1];
        carrito[contador][2] = String.valueOf(cantidad);
        carrito[contador][3] = String.valueOf(monto);
    }
}
```

`public boolean validarStock(int codProdInventario, int cantidadVenta)`

Verifica que el `inventario` contenga suficiente cantidad para la venta. Muestra un mensaje de error si es insuficiente.

Métodos de reportes (PDF)

`public void generarPdfStock()` y `public void generarPdfVentas()`

Estos métodos generan un documento PDF usando iTextPdf con tablas que muestran inventario o ventas. Crean un archivo en la ruta de Descargas.

```
PdfPTable tabla = new PdfPTable(5);
float[] anchosColumnas = {1f, 3f, 2f, 1.5f, 1.5f};
tabla.setWidthPercentage(100);
tabla.setWidths(anchosColumnas);
PdfPCell h1 = new PdfPCell(new Paragraph("ID", encabezadoFont));
// ... código de agregar celdas
```

Método de Bitácora

Todos los métodos de funcionamiento del programa (agregar, buscar, eliminar, registrar ventas, generar reportes, ver datos del estudiante), al terminar cada una su funcion llaman al método **`registrarBitacora(…)`** para guardar el registrar de dicha acción y la almacena en `bitacora`.

`public void registrarBitacora(String tipoAccion, String accion, String usuario, String mensaje)`

Almacena un registro en `bitacora` con la fecha actual obtenida por **`generarFechaHoraActual()`** y un código único por **`generarCodigoUnicoBitacora()`**.

```
bitacora[posicion][0]= fecha;
bitacora[posicion][1]= tipoAccion;
bitacora[posicion][2]= accion;
bitacora[posicion][3]= usuario;
bitacora[posicion][4]= mensaje;
```

Métodos de impresión de datos

Los métodos:

- verProducto(…)
- verListadoBitacora()

Imprimen en consola los registros solicitados.