Manual Técnico – Práctica 01

Jonathan López

Carne. 202404730

Objetivos

Objetivo General

Analizar y comprender los diferentes procesos, métodos y ciclos lógicos para llevar a cabo la solución a un problema que se nos presenta donde se tienen diferentes necesidades. Además de la descripción del código fuente para el correcto funcionamiento de la aplicación.

Objetivos Específicos

- Explicar de manera detallada el código fuente utilizado para el sistema del inventario
- Mostrar los procesos lógicos llevados a cabo

Explicación del Cliente

Las librerías utilizadas son java.io. y java.util.

- **java.io** incluye clases para leer y escribir archivos, manejar flujos de datos, etc.

- **java.utill** incluye clases para trabajar con colecciones, fechas, números aleatorios, etc.

```
import java.io.*;
import java.util.*;
```

```java
public class Practica01 {

    private static final String USUARIO_VALIDO = "sensei_<3020059940101>";
    private static final String CONTRASEÑA_VALIDA = "ipc1_<3020059940101>";
    private static Producto[] productos = new Producto[100];
    private static Venta[] ventas = new Venta[100];
    private static int contadorProductos = 0;
    private static int contadorVentas = 0;
    private static int contadorFacturas = 1;
```

- Declaramos el usuario y las contraseñas válidas para ingresar al sistema, la palabra clave final indica que estos valores no pueden ser modificados después de su inicialización.

- Productos y ventas son arreglos estáticos que pueden almacenar hasta 100 objetos de tipo Producto y Venta, respectivamente.

- contadorProductos, contadorVentas y contadorFacturas son variables estáticas que se utilizan para llevar un registro del número de productos, ventas y facturas, respectivamente.

```
Run | Debug
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    boolean autenticado = false;

    while (!autenticado) {
        System.out.println(x:"Ingrese su usuario:");
        String usuario = scanner.nextLine();

        System.out.println(x:"Ingrese su contraseña:");
        String contraseña = scanner.nextLine();

        if (autenticarUsuario(usuario, contraseña)) {
            autenticado = true;
            mostrarMenu();
        } else {
            System.out.println(x:"Error: Usuario o contraseña incorrectos. Inténtelo nuevamente.");
        }
    }
}

private static boolean autenticarUsuario(String usuario, String contraseña) {
    return usuario.equals(USUARIO_VALIDO) && contraseña.equals(CONTRASEÑA_VALIDA);
}
```

Es el lugar donde comienza la ejecución del programa, creamos el login para que ingrese el usuario

```java
private static void mostrarMenu() {
    Scanner scanner = new Scanner(System.in);
    int opcion;

    do {
        System.out.println(x:"\nBienvenido al menú:");
        System.out.println(x:"1. Agregar nuevos productos al sistema (Individuales)");
        System.out.println(x:"2. Agregar nuevos productos al sistema (Carga masiva)");
        System.out.println(x:"3. Realizar Ventas");
        System.out.println(x:"4. Reportes");
        System.out.println(x:"5. Salir");

        opcion = scanner.nextInt();
        scanner.nextLine(); // Limpiar el buffer

        switch (opcion) {
            case 1:
                agregarProductoIndividual(scanner);
                break;
            case 2:
                cargarProductosMasivamente(scanner);
                break;
            case 3:
                realizarVenta(scanner);
                break;
            case 4:
                generarReportes(scanner);
                break;
            case 5:
                System.out.println(x:"Saliendo del sistema...");
                break;
            default:
                System.out.println(x:"Opción inválida. Intente nuevamente.");
        }
    } while (opcion != 5);
}
```

Creamos el menú llamando la librería Scanner usamos la estructura Switch and Case para acceder a cualquier opción en el menú

```java
private static void agregarProductoIndividual(Scanner scanner) {
    System.out.println(x:"\nAgregar nuevo producto:");
    System.out.print(s:"Ingrese el nombre del producto: ");
    String nombre = scanner.nextLine();

    System.out.print(s:"Ingrese el precio del producto: ");
    double precio = scanner.nextDouble();
    scanner.nextLine(); // Limpiar el buffer

    if (precio <= 0) {
        System.out.println(x:"El precio debe ser mayor a 0. Intente nuevamente.");
        return;
    }

    for (Producto producto : productos) {
        if (producto != null && producto.getNombre().equalsIgnoreCase(nombre)) {
            System.out.println(x:"El producto ya existe en el sistema.");
            return;
        }
    }

    productos[contadorProductos++] = new Producto(nombre, precio);
    System.out.println(x:"Producto agregado exitosamente.");
}
```

Implementamos la primera opción que es agregar productos de manera individual, donde ponemos la condición if, donde el precio debe ser mayor a 0, y también usamos un ciclo for para ir acumulando productos.

```java
private static void cargarProductosMasivamente(Scanner scanner) {
    System.out.println(x:"\nHas seleccionado la opción de carga masiva.");
    System.out.print(s:"Ingrese la ruta del archivo (por ejemplo, C:/ruta/al/archivo.txt): ");
    String rutaArchivo = scanner.nextLine();

    File archivo = new File(rutaArchivo);
    if (!archivo.exists()) {
        System.out.println(x:"El archivo no existe en la ruta proporcionada.");
        return;
    }

    try (BufferedReader br = new BufferedReader(new FileReader(archivo))) {
        String linea;
        boolean esPrimerLinea = true;

        while ((linea = br.readLine()) != null) {
            if (esPrimerLinea) {
                esPrimerLinea = false; // Omitimos la primera línea
                continue;
            }

            String[] partes = linea.split(regex:";");
            if (partes.length != 2) {
                System.out.println("Línea inválida en el archivo: " + linea);
                continue;
            }

            String nombreProducto = partes[0].trim();
            String precioStr = partes[1].trim();

            try {
                double precio = Double.parseDouble(precioStr);
                if (precio > 0) {
                    productos[contadorProductos++] = new Producto(nombreProducto, precio);
                    System.out.println("Producto agregado: " + nombreProducto + " con precio: " + precio);
                } else {
                    System.out.println("El precio de " + nombreProducto + " no es válido.");
                }
            } catch (NumberFormatException e) {
                System.out.println("Error al procesar el precio de " + nombreProducto + ": " + precioStr);
            }
        }
    } catch (IOException e) {
        System.out.println("Error al leer el archivo: " + e.getMessage());
    }
}
```

Implementamos la segunda función del menú, donde cargamos productos masivamente, a traves de la ruta de un archivo txt. Se utiliza la estructura de control para manejar excepciones

- try: Se utiliza para encapsular el código que puede generar una excepción.
- catch: Se utiliza para capturar y manejar la excepción generada en el bloque try.

```java
private static void realizarVenta(Scanner scanner) {
    System.out.println(x:"\nRealizar venta:");
System.out.print(s:"Ingrese el nombre del cliente: ");
String cliente = scanner.nextLine();

System.out.print(s:"Ingrese el NIT del cliente (o C/F si no tiene): ");
String nit = scanner.nextLine();

double totalVenta = 0;
StringBuilder detalleVenta = new StringBuilder();

while (true) {
    System.out.println(x:"\nProductos disponibles:");
    for (int i = 0; i < contadorProductos; i++) {
        Producto producto = productos[i];
        if (producto != null) {
            System.out.println((i + 1) + ". " + producto.getNombre() + " - Q" + producto.getPrecio());
        }
    }

    System.out.print(s:"Seleccione el número del producto que desea comprar (0 para finalizar): ");
    int seleccion = scanner.nextInt();
    scanner.nextLine(); // Limpiar el buffer

    if (seleccion == 0) {
        break;
    }

    if (seleccion < 1 || seleccion > contadorProductos) {
        System.out.println(x:"Selección inválida. Intente nuevamente.");
        continue;
    }
```

Generamos la 3ra opción, que es realizar ventas y generamos la factura

```java
    Producto productoSeleccionado = productos[seleccion - 1];
    System.out.print(s:"Ingrese la cantidad: ");
    int cantidad = scanner.nextInt();
    scanner.nextLine(); // Limpiar el buffer

    double subtotal = productoSeleccionado.getPrecio() * cantidad;
    totalVenta += subtotal;

    detalleVenta.append(productoSeleccionado.getNombre())
                .append(str:" (x").append(cantidad).append(str:") - Q").append(subtotal).append(str:"\n");

    productoSeleccionado.incrementarVentas(cantidad);
}

System.out.println("\nTotal de la venta: Q" + totalVenta);

// Generar un nombre único para la factura basado en el contador global
String nombreFactura = "factura_" + contadorFacturas++ + ".html";

try (FileWriter writer = new FileWriter(nombreFactura)) {
    writer.write(str:"<html><body>");
    writer.write(str:"<h1>Factura</h1>");
    writer.write("<p>Cliente: " + cliente + "</p>");
    writer.write("<p>NIT: " + nit + "</p>");
    writer.write(str:"<p>Detalle de la compra:</p>");
    writer.write("<pre>" + detalleVenta + "</pre>");
    writer.write("<p>Total: Q" + totalVenta + "</p>");
    writer.write(str:"</body></html>");
    System.out.println("Factura generada: " + nombreFactura);
} catch (IOException e) {
    System.out.println(x:"Error al generar la factura.");
}

ventas[contadorVentas++] = new Venta(cliente, nit, totalVenta);
}
```

```java
private static void generarReportes(Scanner scanner) {
    System.out.println(x:"\nGenerar reportes:");
    System.out.println(x:"1. Top 5 productos más vendidos");
    System.out.println(x:"2. Reporte histórico de ventas");
    System.out.print(s:"Seleccione una opción: ");

    int opcion = scanner.nextInt();
    scanner.nextLine(); // Limpiar el buffer

    switch (opcion) {
        case 1:
            generarReporteTopProductos();
            break;
        case 2:
            generarReporteHistoricoVentas();
            break;
        default:
            System.out.println(x:"Opción inválida.");
    }
}
```

Aquí generamos los reportes y usamos switch and case para dar 2 opciones, que son generar el reporte de los Top 5 productos, y el reporte histórico de ventas

```java
private static void generarReporteTopProductos() {
    try (FileWriter writer = new FileWriter(fileName:"reporte_top_productos.html")) {
        writer.write(str:"<html><body><h1>Top 5 productos más vendidos</h1><table border='1'>");
        writer.write(str:"<tr><th>Producto</th><th>Precio</th><th>Cantidad Vendida</th></tr>");

        Producto[] topProductos = obtenerTopProductos(); // Obtener los productos más vendidos
        for (Producto producto : topProductos) {
            writer.write("<tr><td>" + producto.getNombre() + "</td><td>Q" + producto.getPrecio() + "</td><td>" + producto.getCantidadVendida() + "</td></tr>");
        }

        writer.write(str:"</table></body></html>");
        System.out.println(x:"Reporte generado: reporte_top_productos.html");
    } catch (IOException e) {
        System.out.println("Error al generar el reporte: " + e.getMessage());
    }
}
```

```java
    private static void generarReporteHistoricoVentas() {
        try (FileWriter writer = new FileWriter(fileName:"reporte_historico_ventas.html")) {
            writer.write(str:"<html><body><h1>Reporte histórico de ventas</h1><table border='1'>");
            writer.write(str:"<tr><th>Cliente</th><th>NIT</th><th>Total</th></tr>");

            for (Venta venta : ventas) {
                if (venta != null) {
                    writer.write("<tr><td>" + venta.getCliente() + "</td><td>" + venta.getNit() + "</td><td>Q" + venta.getTotal() + "</td></tr>");
                }
            }

            writer.write(str:"</table></body></html>");
            System.out.println(x:"Reporte generado: reporte_historico_ventas.html");
        } catch (IOException e) {
            System.out.println(x:"Error al generar el reporte.");
        }
    }

    private static Producto[] obtenerTopProductos() {
        Producto[] productosNoNulos = Arrays.stream(productos)
                                            .filter(Objects::nonNull) // Eliminar nulos
                                            .toArray(Producto[]::new);
    // Ordenar por cantidad vendida en orden descendente
    Arrays.sort(productosNoNulos, (p1, p2) -> Integer.compare(p2.getCantidadVendida(), p1.getCantidadVendida()));

    // Limitar a los 5 productos más vendidos
    int limite = Math.min(a:5, productosNoNulos.length); // Evita IndexOutOfBounds si hay menos de 5 productos
    return Arrays.copyOfRange(productosNoNulos, from:0, limite);
    }
}
```

Aquí cerramos el main

```java
class Producto {
    private String nombre;
    private double precio;
    private int cantidadVendida;

    public Producto(String nombre, double precio) {
        this.nombre = nombre;
        this.precio = precio;
        this.cantidadVendida = 0;
    }

    public String getNombre() {
        return nombre;
    }

    public double getPrecio() {
        return precio;
    }

    public int getCantidadVendida() {
        return cantidadVendida;
    }

    public void incrementarVentas(int cantidad) {
        this.cantidadVendida += cantidad;
    }
}

class Venta {
    private String cliente;
    private String nit;
    private double total;

    public Venta(String cliente, String nit, double total) {
        this.cliente = cliente;
        this.nit = nit;
        this.total = total;
    }

    public String getCliente() {
        return cliente;
    }

    public String getNit() {
        return nit;
    }

    public double getTotal() {
        return total;
    }
}
```

Las clases que usamos anteriormente, Producto y Venta.