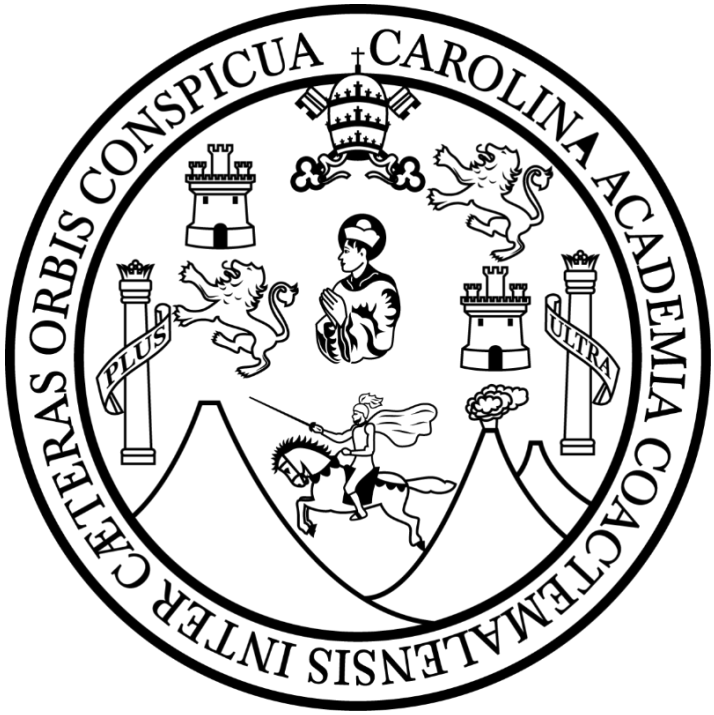
Universidad de San Carlos de Guatemala

Facultad de Ingeniería
Introducción a la
Programación y
Computación 1
Sección: F
Ing. William Escobar

Tutor Académico: Aux. Zenaida Chacón



Proyecto1: Manual Teórico

Sistema de Inventario

Nombre: Eldan André Escobar Asturias

Carné: 202303088

Fecha: 12/09/2025

INTRODUCCIÓN

El control de inventarios constituye un elemento esencial en la administración de recursos de cualquier organización, ya que permite garantizar la disponibilidad de productos, optimizar procesos de venta y mantener un registro confiable de las operaciones realizadas. En el ámbito de la programación, desarrollar un sistema de inventario implica aplicar estructuras de datos, validaciones lógicas y herramientas de generación de reportes que reflejen el estado actual de los productos y las transacciones efectuadas.

En este proyecto se presenta el diseño e implementación de un sistema de inventario en Java, el cual permite registrar, buscar, eliminar y vender productos de manera interactiva. Además, incorpora un módulo de generación de reportes en formato PDF, apoyado en la librería iText, con el objetivo de proporcionar documentación formal del stock disponible y de las ventas realizadas. El sistema también integra una bitácora de acciones que facilita el seguimiento de las operaciones, así como un control de validaciones para evitar errores comunes en el registro de datos.

Este informe tiene como finalidad describir las variables y métodos utilizados en el desarrollo del programa, explicando su funcionamiento, su relación con los objetivos del sistema y su importancia en la gestión eficiente de la información.

GLOSARIO DE PALABRAS CLAVE EN EL CÓDIGO

package: Define el paquete al que pertenece la clase. Sirve para organizar y agrupar clases relacionadas en un mismo proyecto.

import: Se utiliza para incluir librerías o clases externas necesarias para el funcionamiento del programa.

public: Modificador de acceso. Indica que la clase, método o variable puede ser utilizada desde cualquier otra clase.

class: Palabra clave que define una clase, que es la estructura principal de un programa en Java orientado a objetos.

static: Permite que un método o variable pertenezca a la clase en sí misma y no a un objeto en particular.

void: Tipo de retorno que indica que un método no devuelve ningún valor.

main: Método principal de ejecución en Java. Es el punto de entrada donde comienza a correr el programa.

String[] args: Argumentos que se pueden pasar desde la línea de comandos al ejecutar el programa.

int: Tipo de dato primitivo que almacena números enteros.

double: Tipo de dato primitivo que almacena números con decimales (punto flotante).

String: Tipo de dato que almacena cadenas de texto.

new: Se utiliza para crear nuevos objetos en memoria.

Scanner: Clase que permite la entrada de datos desde la consola.

System.out.println: Imprime datos en la consola y hace un salto de línea.

System.out.print: Imprime datos en la consola sin salto de línea.

while: Estructura de control repetitiva que ejecuta un bloque de código mientras se cumpla una condición.

if: Estructura de decisión que ejecuta un bloque de código si la condición especificada es verdadera.

else if: Permite evaluar condiciones adicionales si la primera condición del if no se cumple.

else: Se ejecuta cuando ninguna de las condiciones anteriores es verdadera.

for: Ciclo repetitivo que se usa cuando se conoce de antemano el número de iteraciones.

break: Detiene la ejecución de un ciclo antes de que se complete normalmente.

boolean: Tipo de dato que solo puede contener los valores true o false.

true: Valor lógico que representa verdadero.

false: Valor lógico que representa falso.

try: Bloque que contiene instrucciones que pueden generar errores en tiempo de ejecución.

catch: Bloque que captura y maneja las excepciones producidas en el bloque try.

Exception e: Representa un error que puede ocurrir durante la ejecución del programa.

return: Finaliza la ejecución de un método y puede devolver un valor.

LocalDateTime: Clase de la librería java.time que permite obtener y manejar fecha y hora actuales.

DateTimeFormatter: Clase que sirve para dar formato personalizado a fechas y horas.

Document: Clase de la librería iText que representa un documento PDF en construcción.

Paragraph: Clase de iText utilizada para agregar párrafos de texto a un documento PDF.

PdfPTable: Clase de iText que permite crear tablas dentro de un documento PDF.

PdfWriter: Clase de iText que escribe el contenido del objeto Document en un archivo PDF.

docStock / docVenta: Objetos de tipo Document que representan los archivos PDF de stock y ventas.

desktopPath: Variable que almacena la ruta del escritorio del usuario, donde se guardan los reportes.

fileStock / fileVenta: Variables que contienen la ruta completa y nombre del archivo PDF generado.

fechaHora: Variable que almacena la fecha y hora actual con formato, utilizada para nombrar los reportes.

ventas[]: Arreglo que guarda el historial de ventas realizadas.

bitacora[]: Arreglo que almacena el historial de acciones hechas en el sistema (agregar, eliminar, vender).

totalProductos, totalVentas, totalAcciones: Contadores que indican la cantidad de productos, ventas o acciones registradas.

DICCIONARIO DE MÉTODOS DE ACUERDO AL CÓDIGO

El código inicia importando las librerías a utilizar:

```
import java.util.Scanner;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import com.itextpdf.text.Document;
import com.itextpdf.text.Paragraph;
import com.itextpdf.text.pdf.PdfPTable;
import com.itextpdf.text.pdf.PdfWriter;
```

Scanner: sirve para ingresar un valor por medio del teclado.

“Time” o “Date”: Sirven para acceder a la fecha y hora actual.

ITextPDF: Sirven para modificar o crear archivos PDF desde el programa.

Luego se tienen las variables:

```
Scanner sc = new Scanner(System.in);
//variables con vectores
int[] ids = new int[100];
String[] nombres = new String[100];
int[] cantidades = new int[100];
double[] precios = new double[100];
int totalProductos = 0;

String[] bitacora = new String[200];
int totalAcciones = 0;

String[] ventas = new String[200];
int totalVentas = 0;
double totalAcumuladoVentas = 0.0; // acumulador de todas las ventas
```

Cada variable se utilizará en las siguientes líneas de código para ciertos tipos de datos, y las que tienen valores entre corchetes, son vectores los cuales almacenarán información.

Después se tiene el menú principal, en el cual se eligen las respectivas funciones del sistema.

```
int opc = 0;

while (opc != 8) { //menu de opciones
    System.out.println("\nBienvenido al sistema de inventario de tienda de ropa");
    System.out.println("1. Agregar producto");
    System.out.println("2. Buscar producto");
    System.out.println("3. Eliminar producto");
    System.out.println("4. Registrar venta");
    System.out.println("5. Generar reportes");
    System.out.println("6. Ver datos de estudiante");
    System.out.println("7. Bitacora");
    System.out.println("8. Salir");
    System.out.print("Seleccione su opcion: ");
    opc = sc.nextInt();
    sc.nextLine();
```

Si se elige la primera opción, se tendrá que ingresar los datos solicitados, y también se puede validar si el ID ingresado ya existe, y si el precio es positivo, esto gracias a

las condiciones programadas.

```
if (opc == 1) { //agregar producto
    System.out.print("Ingrese ID del producto: ");
    int id = sc.nextInt();
    sc.nextLine();

    boolean repetido = false; //verificador de id
    for (int i = 0; i < totalProductos; i++) {
        if (ids[i] == id) {
            repetido = true;
            break;
        }
    }

    if (repetido) {
        System.out.println("Error, ya existe un producto con ese ID");
    } else {
        System.out.print("Ingrese producto: ");
        String producto = sc.nextLine();
        System.out.print("Ingrese cantidad: ");
        int cantidad = sc.nextInt();
        System.out.print("Ingrese precio: ");
        double precio = sc.nextDouble();

        if (precio < 0) {
            System.out.println("Error, el precio tiene que ser positivo");
        } else { //almacenar datos para reporte y bitacora
            ids[totalProductos] = id;
            nombres[totalProductos] = producto;
            cantidades[totalProductos] = cantidad;
            precios[totalProductos] = precio;
            totalProductos++;

            String fecha = LocalDateTime.now().format(DateTimeFormatter.ofPattern("dd/MM/yyyy HH:mm:ss"));
            bitacora[totalAcciones++] = fecha + " Se agrego el producto: " + producto;

            System.out.println("Producto agregado exitosamente: " + producto);
        }
    }
```

Para la opción 2, el código pedirá ingresar el ID, para buscar el producto por el mismo, y en caso no exista, lo mostrará; pero si existe, lo imprimirá en pantalla haciendo uso de las variables.

```
} else if (opc == 2) { //buscar producto
    System.out.print("Ingrese ID del producto a buscar: ");
    int id = sc.nextInt();
    boolean encontrado = false;
    for (int i = 0; i < totalProductos; i++) {
        if (ids[i] == id) { //buscar por medio de id
            System.out.println("Producto encontrado: " + nombres[i]);
            System.out.println("ID: " + ids[i] + " | Nombre: " + nombres[i] + " | Cantidad: " + cantidades[i] + " | Precio: Q" + precios[i]);
            encontrado = true;
        }
    }
    if (!encontrado) {
        System.out.println("Error, no se encontro el producto");
    }
```

La opción 3 eliminará un producto, y funciona igual que la anterior, que por medio de ID elimina el producto, solo que aquí se pedirá una confirmación de eliminación, en la cual si se ingresa la tecla “s” correspondiente al si, se eliminará el producto por medio de variables.

```
} else if (opc == 3) { //eliminar producto
    System.out.print("Ingrese ID del producto a eliminar: ");
    int id = sc.nextInt();
    boolean eliminado = false;
    for (int i = 0; i < totalProductos; i++) {
        if (ids[i] == id) { //buscar producto para eliminar por medio de id
            System.out.println("Producto encontrado:");
            System.out.println("ID: " + ids[i] + " | Nombre: " + nombres[i] + " | Cantidad: " + cantidades[i] + " | Precio: Q" + precios[i]);
            System.out.print("Confirmar eliminacion (s/n): "); //confirmacion de eliminacion
            sc.nextLine();
            String confirmacion = sc.nextLine();

            if (confirmacion.equalsIgnoreCase("s")) { //si el usuario marca la tecla, se eliminará
                String nombre = nombres[i];
                for (int j = i; j < totalProductos - 1; j++) {
                    ids[j] = ids[j + 1];
                    nombres[j] = nombres[j + 1];
                    cantidades[j] = cantidades[j + 1];
                    precios[j] = precios[j + 1];
                }
                totalProductos--;
                String fecha = LocalDateTime.now().format(DateTimeFormatter.ofPattern("dd/MM/yyyy HH:mm:ss"));
                bitacora[totalAcciones++] = fecha + " Se elimino producto: " + nombre;
                System.out.println("El producto se elimino correctamente");
            } else {
                System.out.println("El producto no se elimino");
            }
            eliminado = true;
            break;
        }
    }
    if (!eliminado) {
        System.out.println("Error, producto no encontrado");
    }
```

La cuarta opción solicita ingresar el ID para realizar una venta, y esta analizará si la venta es posible o no.

```
} else if (opc == 4) { //hacer venta
    System.out.print("Ingrese ID del producto a vender: ");
    int id = sc.nextInt();
    int indice = -1; //contar existencia de producto
    for (int i = 0; i < totalProductos; i++) {
        if (ids[i] == id) {
            indice = i;
            break;
        }
    }
```

La variable “índice” indicará si existen tales productos para que la venta pueda hacerse exitosamente, por lo que si no hay objetos (representados como -1), no se hará la venta. Y en caso si haya suficientes productos, se realizará la venta, no sin antes pedir una confirmación, y luego imprimirá los datos.

```java
if (indice == -1) {
    System.out.println("Error, ese producto no existe");
} else {
    System.out.println("Producto encontrado:");
    System.out.println("ID: " + ids[indice] + " | Nombre: " + nombres[indice] + " | Cantidad disponible: " + cantidades[indice] + " | Precio

    System.out.print("Ingrese cantidad a vender: ");
    int cantVender = sc.nextInt();

    if (cantVender <= 0) { //validacion de existencia de producto
        System.out.println("Error, la cantidad a vender debe ser mayor que cero");
    } else if (cantVender > cantidades[indice]) {
        System.out.println("Error, no hay suficiente stock disponible, el stock actual es: " + cantidades[indice]);
    } else {
        System.out.print("Confirmar la venta (s/n): "); //confirmacion de venta
        sc.nextLine();
        String confirmacion = sc.nextLine();

        if (confirmacion.equalsIgnoreCase("s")) {
            cantidades[indice] -= cantVender;
            double totalGastado = cantVender * precios[indice];
            totalAcumuladoVentas += totalGastado; // acumular datos en los registros
            String fecha = LocalDateTime.now().format(DateTimeFormatter.ofPattern("dd/MM/yyyy HH:mm:ss"));

            bitacora[totalAcciones++] = fecha + "Se vendio " + cantVender + " de " + nombres[indice];
            ventas[totalVentas++] = fecha + " | " + nombres[indice] + " | Cantidad: " + cantVender + " | Total: Q" + totalGastado;

            System.out.println("Venta registrada: " + nombres[indice] + " | Total: Q" + totalGastado);
            System.out.println("Stock restante: " + cantidades[indice]);
        } else {
            System.out.println("La venta no se concreto");
        }
    }
}
```

La quinta opción corresponde a la generación de reportes. Para el primer reporte se mostrarán los productos en stock, pero primero se tiene que definir la ruta en que se guardarán los reportes, luego se elegirá el nombre del archivo, en este caso es la hora y fecha de generación, junto a la palabra “stock”, y también se programará lo que se desea imprimir en el archivo, haciendo uso de una tabla.

```java
} else if (opc == 5) { //generacion de reportes
    try {
        String desktopPath = System.getProperty("user.home") + "/Desktop/"; //ruta de almacenamiento de reportes
        //nombre de archivos
        String fechaHora = LocalDateTime.now().format(DateTimeFormatter.ofPattern("dd_MM_yyyy_HH_mm_ss"));
        String fileStock = desktopPath + fechaHora + "_Stock.pdf";
        Document docStock = new Document();
        PdfWriter.getInstance(docStock, new java.io.FileOutputStream(fileStock));
        docStock.open();
        docStock.add(new Paragraph("Reporte de Stock"));
        docStock.add(new Paragraph("Fecha: " + LocalDateTime.now().format(DateTimeFormatter.ofPattern("dd/MM/yyyy HH:mm:ss"))));
        docStock.add(new Paragraph("\n"));

        PdfPTable tableStock = new PdfPTable(4); //datos incluidos en el pdf
        tableStock.addCell("Codigo");
        tableStock.addCell("Nombre");
        tableStock.addCell("Precio");
        tableStock.addCell("Cantidad");

        for (int i = 0; i < totalProductos; i++) {
            tableStock.addCell(String.valueOf(ids[i]));
            tableStock.addCell(nombres[i]);
            tableStock.addCell("Q" + String.format("%.2f", precios[i]));
            tableStock.addCell(String.valueOf(cantidades[i])); //total de productos
        }
        docStock.add(tableStock);
        docStock.close();
```

Para el reporte de ventas es el mismo procedimiento, solo cambian los datos y el nombre del archivo.

```
//reporte de ventas
String fileVenta = desktopPath + fechaHora + "_Venta.pdf";
Document docVenta = new Document();
PdfWriter.getInstance(docVenta, new java.io.FileOutputStream(fileVenta));
docVenta.open();
docVenta.add(new Paragraph("Reporte de Ventas"));
docVenta.add(new Paragraph("Fecha: " + LocalDateTime.now().format(DateTimeFormatter.ofPattern("dd/MM/yyyy HH:mm:ss"))));
docVenta.add(new Paragraph("\n"));

PdfPTable tableVenta = new PdfPTable(4);
tableVenta.addCell("Fecha");
tableVenta.addCell("Producto");
tableVenta.addCell("Cantidad");
tableVenta.addCell("Total");

for (int i = 0; i < totalVentas; i++) {
    String[] partes = ventas[i].split("\\|");
    tableVenta.addCell(partes[0].trim());
    tableVenta.addCell(partes[1].trim());
    tableVenta.addCell(partes[2].replace("Cantidad:", "").trim());
    tableVenta.addCell(partes[3].replace("Total:", "").trim());
}
docVenta.add(tableVenta);
docVenta.add(new Paragraph("\nTotal general de ventas: Q" + String.format("%.2f", totalAcumuladoVentas)));
docVenta.close();
```

La sexta opción únicamente imprimirá los datos del estudiante en la consola, los cuales fueron declarados previamente.

```
} else if (opc == 6) { //mostrar datos del estudiante
    System.out.println("Datos del estudiante");
    System.out.println("Nombre: Eldan Andre Escobar Asturias");
    System.out.println("Carnet: 202303088");
    System.out.println("Curso: Introduccion a la programacion y computacion 1");
```

La séptima opción corresponde a la bitácora, la cual a diferencia de los reportes, muestra en la consola todas las acciones realizadas en el sistema, gracias a que en cada condición se generó una variable homónima para almacenar los datos, así luego se imprimen junto a su fecha y hora de acción.

```
} else if (opc == 7) { //mostrar bitácora mediante datos almacenados en cada opción
    if (totalAcciones == 0) {
        System.out.println("Error, la bitacora está vacía");
    } else {
        System.out.println("Bitacora");
        for (int i = 0; i < totalAcciones; i++) {
            System.out.println(bitacora[i]);
        }
    }
```

La octava y última opción es únicamente para terminar la ejecución del programa por medio del ciclo repetitivo while; también se mostrará un error si se ingresa un valor no correspondiente al menú de opciones.

```
} else if (opc == 8) { //fin del programa
    System.out.println("Gracias por usar el sistema");
} else {
    System.out.println("Error, ingrese una opcion valida");
}
```

POSIBLES MEJORAS AL SISTEMA EN EL FUTURO

Integración con una base de datos: Actualmente, los productos, ventas y bitácora se almacenan en arreglos dentro del programa, lo que limita la persistencia de la información. Una mejora sería conectar el sistema a una base de datos como: MySQL, PostgreSQL, etc, para que los datos se conserven incluso después de cerrar la aplicación.

Interfaz gráfica de usuario (GUI): Implementar una interfaz gráfica permitiría al usuario interactuar de forma más amigable con el sistema, facilitando la gestión del inventario sin depender únicamente de la consola.

Validaciones avanzadas: Se pueden agregar más controles de validación, como límites máximos de stock, restricciones en precios demasiado bajos o altos, y mensajes de advertencia cuando el inventario de un producto esté por agotarse.

Reportes más completos: Los reportes en PDF podrían incluir gráficos estadísticos, por ejemplo: ventas mensuales, productos más vendidos, niveles de stock crítico; utilizando librerías adicionales de visualización.

Gestión de usuarios y roles: Se puede implementar un sistema de autenticación con distintos niveles de acceso (por ejemplo: administrador, vendedor, auditor) para reforzar la seguridad y el control de las operaciones.

Exportación en diferentes formatos: Además de PDF, se podría permitir la exportación de reportes en otros formatos como Excel o CSV, para facilitar el análisis de datos en otras herramientas.

Notificaciones automáticas: El sistema podría enviar alertas ya sea por correo electrónico o dentro de la aplicación, cuando el inventario de un producto alcance un nivel mínimo establecido.

Migración a aplicación web o móvil: Como evolución natural, este sistema podría convertirse en una aplicación web con acceso remoto o incluso en una aplicación móvil, permitiendo la gestión del inventario desde cualquier lugar.

CONCLUSIÓN

El desarrollo del sistema de inventario en Java permitió poner en práctica los fundamentos de la programación estructurada y orientada a objetos, así como la correcta utilización de estructuras de control, arreglos y manejo de excepciones. A través de las distintas funcionalidades implementadas —registro de productos, búsqueda, eliminación, ventas y generación de reportes— se logró construir una aplicación que resuelve de manera eficiente la gestión básica de un inventario en una tienda de ropa.

La integración de la librería iText para la generación de reportes en formato PDF añadió un valor significativo al proyecto, ya que proporciona documentación clara y estructurada sobre el stock disponible y las transacciones realizadas. Además, el uso de una bitácora de acciones permite llevar un control detallado de las operaciones, reforzando la trazabilidad del sistema.

En conclusión, este proyecto no solo representa una aplicación práctica de los conceptos aprendidos en el curso, sino también un ejemplo de cómo la programación puede ofrecer soluciones útiles y escalables para problemas reales en el ámbito empresarial.