Universidad de San Carlos de Guatemala

Introducción a la programación y computación 1

Sección: A

Manual Técnico: Sistema de Inventario Tienda de ropa

Alumno: Carlos David Recinos Chip

Carne: 2022-09463

Índice

1. Descripción General

Este sistema permite administrar productos, registrar ventas, generar reportes en PDF y mantener una bitácora de acciones. Está desarrollado en Java y opera mediante una interfaz de consola.

2. Tecnologías Utilizadas

- **Lenguaje:** Java
- **Librerías estándar:**
  - java.io.* – manejo de archivos
  - java.util.* – estructuras de datos y entrada
  - java.time.* – fechas y horas
- **Librería externa:**
  - iText – generación de documentos PDF

3. Estructura de Datos

Variable	Tipo	Descripción
nombres	String[50]	Nombres de productos
categorias	String[50]	Categorías de productos
codigos	String[50]	Códigos únicos
precios	double[50]	Precios unitarios
cantidades	int[50]	Stock disponible
contadorProductos	int	Total de productos registrados
bitacora	ArrayList<String>	Registro de acciones
ventas	ArrayList<String>	Registro de ventas

4. Funciones Principales

main()

Menú interactivo con 8 opciones que redirigen a los métodos correspondientes.

```
public static void main(String[] args) {
    Scanner mn = new Scanner(System.in);
    boolean salir = false;
    int opcion;

    while (!salir){
          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 Reporte ");
          System.out.println("6. Ver datos del estudiante ");
          System.out.println("7. Bitacora ");
          System.out.println("8. Salir ");
          System.out.println("Ingresa que opcion deseas realizar: ");
          opcion = mn.nextInt();
          try{
          switch (opcion){
              case 1 -> agregarProducto();
              case 2 -> buscarProducto();
              case 3 -> eliminarProducto();
              case 4 -> registrarVenta();
              case 5 -> generarReportesPDF();
              case 6 -> verDatosEstudiante();
              case 7 -> mostrarBitacora();
              case 8 -> salir = true;
              default -> System.out.println("Las opciones son entrer 1 y 8");

             }
             } catch (InputMismatchException e){
                 System.out.println("Debes escribir un numero");
                 mn.next();
             }
```

agregarProducto()

Valida y registra un nuevo producto. Verifica unicidad del código y entrada correcta de precio y cantidad.

```
public static void agregarProducto() {
Scanner entrada = new Scanner(System.in);
System.out.println("   AGREGAR PRODUCTO  ");

// Nombre del producto
System.out.print("Nombre del producto: ");
String nombre = entrada.nextLine();
if (nombre.trim().isEmpty()) {
    System.out.println("El nombre no puede estar vacío.");
    registrarBitacora("Agregar producto - nombre vacio", false);
    return;
}

// Categoría
System.out.print("Categoria: ");
String categoria = entrada.nextLine();
if (categoria.trim().isEmpty()) {
    System.out.println("La categoria no puede estar vacia.");
    registrarBitacora("Agregar producto - categoria vacia", false);
    return;
}

// Código único
String codigo;
boolean codigoValido;
do {
    System.out.print("Codigo unico: ");
    codigo = entrada.nextLine();
    codigoValido = true;

    for (int i = 0; i < contadorProductos; i++) {
        if (codigos[i].equalsIgnoreCase(codigo)) {
            System.out.println("Ese codigo ya existe.");
            registrarBitacora("Agregar producto - codigo repetido", false);
            codigoValido = false;
```

buscarProducto()

Permite buscar productos por nombre, categoría o código. Muestra detalles si hay coincidencia.

```
public static void buscarProducto() {
 Scanner buscar = new Scanner(System.in);
 System.out.println("  BUSCAR PRODUCTO ");
 System.out.println("1. Nombre\n2. Categoria\n3. Codigo");
 System.out.print("Opcion: ");
 int criterio = buscar.nextInt();
 buscar.nextLine();

 System.out.print("Producto a buscar: ");
 String valor = buscar.nextLine().toLowerCase();

 boolean encontrado = false;

 for (int i = 0; i < contadorProductos; i++) {
     boolean coincide = false;
     switch (criterio) {
         case 1 -> coincide = nombres[i].toLowerCase().contains(valor);
         case 2 -> coincide = categorias[i].toLowerCase().contains(valor);
         case 3 -> coincide = codigos[i].toLowerCase().equals(valor);
         default -> {
             System.out.println("Opcion invalida.");
             registrarBitacora("Buscar producto", false);
             return;
         }
     }

     if (coincide) {
         System.out.println(" Producto encontrado:");
         System.out.println("Nombre: " + nombres[i]);
         System.out.println("Categoria: " + categorias[i]);
         System.out.println("Codigo: " + codigos[i]);
         System.out.println("Precio: Q" + precios[i]);
         System.out.println("Stock: " + cantidades[i]);
         encontrado = true;
     }
```

eliminarProducto()

Busca por código y elimina el producto tras confirmación. Reordena los arreglos para mantener consistencia.

```
public static void eliminarProducto() {
Scanner eliminar = new Scanner(System.in);
System.out.println(" ELIMINAR PRODUCTO ");
System.out.print("Codigo del producto: ");
String codigoBuscar = eliminar.nextLine().toLowerCase();

boolean eliminado = false;

for (int i = 0; i < contadorProductos; i++) {
    if (codigos[i].toLowerCase().equals(codigoBuscar)) {
        System.out.println("Producto: " + nombres[i]);
        System.out.print("¿Eliminar? (si/no): ");
        String confirmacion = eliminar.nextLine().toLowerCase();

        if (confirmacion.equals("si")) {
            for (int j = i; j < contadorProductos - 1; j++) {
                nombres[j] = nombres[j + 1];
                categorias[j] = categorias[j + 1];
                codigos[j] = codigos[j + 1];
                precios[j] = precios[j + 1];
                cantidades[j] = cantidades[j + 1];
            }
            contadorProductos--;
            System.out.println("Producto eliminado.");
            eliminado = true;
        } else {
            System.out.println("Eliminacion cancelada.");
        }

        break;
    }
}

registrarBitacora("Eliminar producto", eliminado);
if (!eliminado) {
```

registrarVenta()

Verifica existencia y stock del producto. Registra la venta y actualiza inventario. Guarda la transacción con fecha.

```
public static void registrarVenta() {
    Scanner venta = new Scanner(System.in);
    System.out.println("   REGISTRAR VENTA  ");
    System.out.print("Codigo del producto: ");
    String codigoVenta = venta.nextLine().toLowerCase();

    boolean ventaExitosa = false;
    int indice = -1;

    for (int i = 0; i < contadorProductos; i++) {
        if (codigos[i].toLowerCase().equals(codigoVenta)) {
            indice = i;
            break;
        }
    }

    if (indice == -1) {
        System.out.println(" Producto no encontrado.");
        registrarBitacora("Registrar venta", false);
        return;
    }

    System.out.println("Producto: " + nombres[indice]);
    System.out.println("Stock: " + cantidades[indice]);
    System.out.print("Cantidad a vender: ");
    if (!venta.hasNextInt()) {
    System.out.println(" Entrada inválida. Debe ser un número entero.");
    venta.next(); // limpiar entrada
    registrarBitacora("Registrar venta - cantidad no numérica", false);
    return;
}

    int cantidadVendida = venta.nextInt();

    if (cantidadVendida <= 0 || cantidadVendida > cantidades[indice]) {
```

generarReportesPDF()

Genera dos archivos PDF:

- **Stock:** Lista de productos con sus datos.
- **Ventas:** Historial de ventas con fecha y total.

```
public static void generarReportesPDF() {
    try {
        //  Crear carpeta "reportes" si no existe
        File carpeta = new File("reportes");
        if (!carpeta.exists()) {
            carpeta.mkdir();
        }

        // Generar timestamp para nombrar los archivos
        DateTimeFormatter formato = DateTimeFormatter.ofPattern("dd_MM_yyyy_HH_mm_ss");
        String timestamp = LocalDateTime.now().format(formato);

        String archivoStock = "reportes/" + timestamp + "_Stock.pdf";
        String archivoVenta = "reportes/" + timestamp + "_Venta.pdf";

        //  Reporte de Stock
        Document docStock = new Document();
        PdfWriter.getInstance(docStock, new FileOutputStream(archivoStock));
        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(" "));

        PdfPTable tablaStock = new PdfPTable(5);
        tablaStock.addCell("Nombre");
        tablaStock.addCell("Codigo");
        tablaStock.addCell("Categoria");
        tablaStock.addCell("Precio");
        tablaStock.addCell("Cantidad");

        for (int i = 0; i < contadorProductos; i++) {
            tablaStock.addCell(nombres[i]);
            tablaStock.addCell(codigos[i]);
            tablaStock.addCell(categorias[i]);
```

registrarBitacora(String tipoAccion, boolean esCorrecta)

Registra cada acción con fecha, resultado y usuario.

```
public static void registrarBitacora(String tipoAccion, boolean esCorrecta) {
    LocalDateTime ahora = LocalDateTime.now();
    DateTimeFormatter formato = DateTimeFormatter.ofPattern("dd/MM/yyyy HH:mm:ss");
    String fechaHora = ahora.format(formato);

    String resultado = esCorrecta ? "Correcta" : "Erronea";

    String registro = "\nFecha y hora: " + fechaHora +
                      "\nAccion: " + tipoAccion +
                      "\nResultado: " + resultado +
                      "\nUsuario: " + usuarioActual;

    bitacora.add(registro);
}
```

mostrarBitacora()

Muestra todas las acciones registradas en consola.

```
public static void mostrarBitacora() {
    System.out.println(" BITACORA ");
    if (bitacora.isEmpty()) {
        System.out.println("No hay acciones registradas.");
    } else {
        for (String registro : bitacora) {
            System.out.println("————————————");
            System.out.println(registro);
            System.out.println("————————————");
        }
    }
}
// terminan metodos para la bitacora
```

verDatosEstudiante()

Muestra los datos del autor del sistema.

```
public static void verDatosEstudiante() {
    System.out.println(" DATOS DEL ESTUDIANTE ");
    System.out.println("Nombre completo: Carlos David Recinos Chip");
    System.out.println("Carnet: 202209463");
    System.out.println("Laboratorio: Introduccion a la computacion y programacion 1");
    System.out.println("Seccion: A");
    System.out.println("GitHub: chipcx");
}
```

5. Validaciones y Seguridad

- Validación de entradas numéricas con hasNextInt() y hasNextDouble().
- Confirmación de eliminación para evitar errores.
- Registro de errores y acciones en bitácora.
- Control de duplicados por código único.

6. Estructura de Archivos

- Carpeta reportes/ se crea automáticamente si no existe.
- Archivos PDF nombrados con timestamp: dd_MM_yyyy_HH_mm_ss_Stock.pdf dd_MM_yyyy_HH_mm_ss_Venta.pdf

7. Recomendaciones para Programadores

- Modularizar funciones si se desea escalar el sistema.
- Reemplazar arreglos por estructuras dinámicas como ArrayList para mayor flexibilidad.
- Implementar persistencia con archivos o bases de datos si se requiere guardar datos entre sesiones.
- Añadir interfaz gráfica para mejorar experiencia de usuario.