Universidad de San Carlos de Guatemala
Facultad de ingeniería
Matemática para computación 1
Segundo semestre 2025

Manual técnico

Carne: 202104851

Nombre: Jonnathan David Hernandez

Fecha: 14/09/2025

Estructura del Proyecto

1. **Arquitectura General**

El sistema está desarrollado siguiendo una arquitectura modular con las siguientes clases:

Sistema de Inventario/

Producto.java (Clase modelo para productos)

Venta.java (Clase modelo para ventas)

SistemaInventario.java (Clase principal con lógica del sistema)

2. **Estructura de Datos**

- **Arrays estáticos**: Utilizados para almacenar productos, ventas y bitácora
- **Capacidades máximas**:
    - Productos: 100 elementos
    - Ventas: 500 elementos
    - Bitácora: 200 registros

Requisitos del Sistema

1. **Software Requerido**

Para Desarrollo:

- **JDK (Java Development Kit)**: Versión 8 o superior
- **IDE recomendado**: NetBeans, Eclipse, IntelliJ IDEA, o VS Code
- **Sistema Operativo**: Windows 7+, macOS 10.12+, Linux Ubuntu 16.04+
- **Git**: Para control de versiones

Para Ejecución:

- **JRE (Java Runtime Environment)**: Versión 8 o superior
- **Terminal/Consola**: Para ejecutar la aplicación
- **Mínimo 50MB** de espacio en disco para archivos generados

2. **Hardware Requerido**

Mínimo:

- **Procesador**: Intel Pentium 4 o AMD equivalente
- **RAM**: 512 MB
- **Disco Duro**: 100 MB de espacio libre
- **Resolución**: 800x600

Recomendado:

- **Procesador**: Intel Core i3 o AMD equivalente
- **RAM**: 2 GB o más
- **Disco Duro**: 1 GB de espacio libre
- **Resolución**: 1024x768 o superior

Descripción de Métodos Principales

1. **Clase Producto**

```
public class Producto {
  // Atributos públicos para acceso directo
  public String nombre, categoria, codigo;
  public double precio;
  public int cantidad;
}
```

Propósito: Modelo de datos para representar un producto en el inventario.

Métodos importantes:

- Producto(...): Constructor que inicializa todos los atributos
- mostrar(): Formatea y muestra la información del producto

2. **Clase Venta**

```
public class Venta {
  public String codigo, nombre, fecha;
```

```
    public int cantidad;
    public double total;
}
```

Propósito: Modelo de datos para registrar transacciones de venta.

Características:

- Calcula automáticamente el total (cantidad × precio unitario)
- Registra fecha y hora automáticamente usando LocalDateTime

Clase SistemaInventario (Métodos Principales)

main(String[] args)

- **Propósito**: Punto de entrada del programa
- **Funcionalidad**:
    - Inicializa el sistema
    - Controla el bucle principal del menú
    - Maneja excepciones globales para evitar cierre inesperado

agregarProducto()

- **Propósito**: Registra nuevos productos en el inventario
- **Validaciones**:
    - Capacidad máxima del inventario (100 productos)
    - Campos no vacíos
    - Código único (no duplicado)
    - Precio y cantidad positivos
- **Manejo de errores**: Try-catch para entrada de datos incorrectos

buscarProducto()

- **Propósito**: Localiza productos por código
- **Algoritmo**: Búsqueda lineal en el array de productos
- **Complejidad**: O(n) donde n = número de productos

eliminarProducto()

- **Propósito**: Remueve productos del inventario
- **Algoritmo**:
    1. Busca el producto por código
    2. Solicita confirmación del usuario
    3. Desplaza elementos hacia la izquierda para eliminar
- **Complejidad**: O(n) para búsqueda + O(n) para desplazamiento

registrarVenta()

- **Propósito**: Procesa transacciones de venta
- **Flujo**:
    1. Verifica existencia del producto
    2. Valida stock suficiente
    3. Calcula total de la venta
    4. Actualiza inventario
    5. Persiste la venta en archivo
- **Atomicidad**: La operación es completa o no se realiza

generarReporte(String tipo)

- **Propósito**: Genera reportes en formato texto
- **Tipos**: "Stock" y "Ventas"
- **Formato de archivo**: DD_MM_YYYY_HH_mm_ss_[Tipo].txt
- **Contenido**:
    - Stock: Lista completa de productos con disponibilidad
    - Ventas: Historial con totales generales

agregarBitacora(String accion)

- **Propósito**: Registra actividades del sistema
- **Información capturada**:

- Timestamp (fecha y hora)
- Descripción de la acción
- Estado (exitosa/fallida)

Algoritmos y Estructuras de Datos

Algoritmos de Búsqueda

- **Tipo**: Búsqueda lineal secuencial
- **Criterio**: Comparación exacta de códigos de producto
- **Eficiencia**: O(n) en el peor caso

Algoritmo de Eliminación

```
// Desplazamiento hacia la izquierda
for (int i = indice; i < totalProductos - 1; i++) {
   productos[i] = productos[i + 1];
}
productos[--totalProductos] = null;
```

Gestión de Memoria

- **Arrays estáticos**: Tamaño fijo en tiempo de compilación
- **Gestión manual**: Contadores para elementos activos
- **Limpieza**: Referencias null para elementos eliminados

Manejo de Archivos

Persistencia de Ventas

- **Archivo**: ventas.txt
- **Formato**: CSV (valores separados por comas)
- **Modo**: Append (agregar al final)
- **Estructura**: codigo,nombre,cantidad,total,fecha

Generación de Reportes

- **Formato**: Texto plano (.txt)

- **Nomenclatura**: Timestamp + tipo de reporte
- **Contenido**: Estructurado con headers y separadores

Validaciones y Manejo de Errores

Validaciones de Entrada

- **Campos obligatorios**: No vacíos, no nulos
- **Valores numéricos**: Positivos para precios y cantidades
- **Códigos únicos**: Verificación antes de insertar
- **Rangos**: Respeto a capacidades máximas de arrays

Manejo de Excepciones

- **InputMismatchException**: Entrada de tipo incorrecto
- **IOException**: Errores de archivos
- **ArrayIndexOutOfBoundsException**: Desbordamiento de arrays
- **Strategy**: Try-catch con mensajes informativos y continuidad del programa

Consideraciones de Mantenimiento

Escalabilidad

- **Limitación actual**: Arrays de tamaño fijo
- **Mejora futura**: Migrar a estructuras dinámicas
- **Modularidad**: Fácil expansión de funcionalidades

Debugging y Logging

- **Bitácora integrada**: Rastrea todas las operaciones
- **Mensajes informativos**: Feedback claro al usuario
- **Manejo graceful**: Errores no terminan la aplicación

Versionado

- **Control de versiones**: Git requerido
- **Commits regulares**: Mínimo 2 por semana
- **Documentación**: Actualización paralela al código

Compilación y Ejecución

Compilación

javac *.java

Ejecución

java SistemaInventario