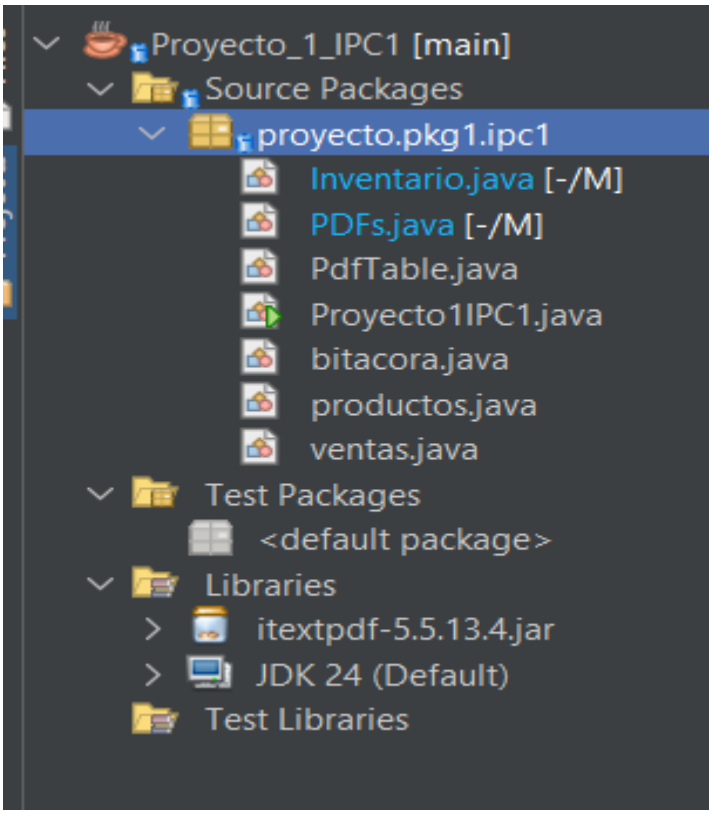
MANUAL TÉCNICO

Estructura del Proyecto

1.1 estructuración



1.2. Paquetes y Organización

- **Paquete principal**: proyecto.pkg1.ipc1
- **Clases incluidas**:
    - Proyecto1IPC1.java (Main)
    - Inventario.java
    - productos.java
    - ventas.java
    - bitacora.java
    - PDFs.java

2. REQUISITOS DE SOFTWARE Y HARDWARE

2.1. Para Desarrollo

Componente	Requerimiento Mínimo	Recomendado
JDK	Java Development Kit 8+	JDK 11 o superior
IDE	NetBeans, Eclipse, IntelliJ	IntelliJ IDEA
Librerías	iTextPDF (para generación de PDFs)	itextpdf-5.5.13.3.jar o similar
Sistema Operativo	Windows 7+, Linux, macOS	Windows 10+, Linux Ubuntu LTS

N

2.2. Hardware

Componente	Requerimiento Mínimo	Recomendado
Procesador	1.0 GHz	Dual Core 2.0 GHz o superior
RAM	512 MB	2 GB
Disco Duro	100 MB libres	500 MB libres

3. DESCRIPCIÓN DE MÉTODOS IMPORTANTES

3.1. Clase: Inventario

3.1.1 public void agragarunProducto(Scanner scanner)

- **Propósito**: Gestionar el proceso completo de registro de nuevos prodctos en el inventario.
- **Flujo Detallado**:

- **Validación de código**: Verifica que el código no exista previamente (evita duplicados)
- **Captura de catos**: Solicita de forma consecutiva nombre, precio, cantidad, categoría
- **Validación de Negocio**: Asegura que precio > 0 y cantidad >= 0
- **Creación de Instancia**: Construye objeto `productos` con los datos validados
- **Almacenamiento**: Agrega al array `productos[]` en la posición `contadorProductos`
- **Actualización de Contador**: Incrementa `contadorProductos++`
- **Registro de auditoría**: Llama a `agregarBitacora()` con resultado de la operación

- **Manejo de Errores**:
  - Código duplicado → Mensaje: "El codigo ya existe, ingrese uno valido"
  - Datos inválidos → Mensaje: "Error: El precio o la cantidad deben ser positivos"

3.1.2 eliminarProducto(Scanner scanner)

- **Propósito**: Gestionar la eliminación de productos del inventario.
- **Flujo Detallado**:
  1. **Búsqueda**: Busca producto por código (búsqueda lineal en array).
  2. **Confirmación**: solicita confirmación (SI/NO).
  3. **Eliminación física**: Reorganiza array moviedo elementos posteriores.
  4. **Actualización del array**: Decrementa `contadorProductos--`
  5. **Auditoría**: Registra resultado en bitácora.

3.1.3 registroVentas(Scanner scanner)

- **Propósito**: Gestionar el proceso completo de ventas con actualización del stock
- **Flujo Detallado**:
  1. **Validación de existencia**: Busca producto por código.
  2. **Validación de Stock**: Verifica que la cantidad solicitada ≤ stock disponible.
  3. **Cálculo Automático**: Total = cantidad * precio unitario.
  4. **Actualización de Stock**: Reduce el stock del producto vendido.

5. **Registro de Venta**: Crea objeto ventas y lo almacena en array.
6. **Actualización de Contador**: Incrementa contadorVentas++

3.1.4 buscarProducto(Scanner scanner)

- **Propósito:** Búsqueda de productos por múltiples criterios.
- **Algoritmo de Búsqueda:**
  - Búqueda por cdigo (coincidencia exacta o por referrencia)
  - Búsqueda por nombre (por coincidencia exacta o por referencia)
- **Resultados:**
  - Muestra todos los productos que coincidan con el criterio.
  - Utiliza MostrarProducto() para visualizar el producto que se encontró.

3.1.5 generacionReportes()

- **Propósito**: Coordinar la generación de todos los reportes disponibles
- **Flujo**:
  1. **Validación de datos**: Verifica que existan productos o ventas
  2. **Llamada a PDFs**: Invoca métodos estáticos de la clase PDFs
  3. **Generación**: Crea reportes de stock y ventas
  4. **Confirmación**: Muestra mensajes de éxito/error al usuario deendiendo
- **Integración**:
  - PDFs.generadorreportesStock(productos, contadorProductos)
  - PDFs.generarReporteVentas(ventas, contadorVentas)

3.1.6 agregarBitacora(String nombreAccion, String estadoResultado)

- **Propósito**: Sistema centralizado de logging y auditoría
- **Automático**: Genera fecha y hora de la acción en formato "dd/MM/yyyy HH:mm:ss"
- **Estructura de la bitácora**:
  - Fecha/hora

- Nombre de la acción
- Resultado (Éxito/Error específico)
- Usuario fijo "estudiante"

3.2. Clase: productos.java

- **Constructor:**

public productos(String nombre, String categoria, String ID, double precio, int cantidad)

- **Inicialización**: Todos los parámetros son requeridos
- **Validaciones**: Los setters pueden incluir validacione

3.2.1 MostrarProducto()

- **proposito**: Se usa para mostrar la información de los productos creados.

3.3. Clase: PDFs

3.3.1 generadorreportesStock(productos[] productos, int contadorProductos)

- **Librería usada**: iTextPDF 5.5.13 para generación de PDFs
- **Estructura del documento**:
  - Título centrado: "REPORTE DE STOCK TIENDA"
  - Tabla con 5 columnas: Código, Nombre, Categoría, Precio, Stock
  - Timestamp de generación automático
- **Nombre del archivo**: dd_MM_yyyy_HH_mm_ss_Stock.pdf (ejemplo: 12_03_2024_14_30_45_Stock.pdf)

3.3.2 generarReporteVentas(ventas[] ventas, int contadorVentas)

- **Contenido**:
  - Tabla con 4 colunas: Código, Cantidad, Total, Fecha
  - Total general acumulado de todas las ventas
  - Timestamp de cada venta
- **Validaciones**:

- Verifica que el array no sea null
- Confirma que existan ventas para reportar

3.4. Clase: Proyecto1IPC1.java (Main)

3.4.1datosEstudiante()

- **Propósito**: Mostrar información del desarrolador (ósea yo xd)

GRACIAS 😊