E-commerce Inventory Product Management System

Liwei Rong 6400257, David Silva 0036195

Programming Concepts II, Final Project Report

February 25, 2026

Contents

1 Introduction

This project presents the design and implementation of an E-commerce Inventory Management System developed in Java and using the JDBC Java API to access a relational database, allowing for connecting to databases, executing SQL statements, and processing query results. The system integrates CSV-based data ingestion with relational database persistence using Microsoft SQL Server.

The primary objective of the project is to demonstrate the integration of file-based structured data with a relational database management system through a layered software architecture, that allows reading product data from two separate CSV files (Electronics.csv and Clothing.csv), creating objects of appropriate subclasses (Electronics or Clothing), displaying them in a formatted console table, and inserting the data stored in disk into a SQL database. The project follows an object-oriented design. We use Maven for build management. We incorporate exception handling for robust error management.

2 Project Overview and Purpose

The system reads product data from CSV files, validates and converts them into domain objects, temporarily stores them in memory, and persists them into a relational database using JDBC batch operations.

The system supports two product categories:

- Electronics (weight-based shipping model)
- Clothing (volume-based shipping model)

The project demonstrates:

- Object-Oriented Design principles
- Layered software architecture
- JDBC database connectivity
- Transaction management
- Defensive programming using validation factories

3 Software Architecture

The system follows a layered architecture composed of:

3.1 Model Layer

Three classes model different product types—Clothing and Electronics—that extend a shared base class, Product. The Product class provides common functionality, while Clothing and Electronics add specific properties (volume and weight) and their own shipping cost calculations. Both subclasses ensure that only valid products are created using Optional, which safely returns objects when parameters are correct. This design centralizes common features in Product and allows specialized behavior in the subclasses.

The Product class includes common properties like category, product name, quantity, unit cost, and margin. It offers methods to calculate the inventory value and unit price, and defines an abstract method getShippingCost, which subclasses must implement. The toString method provides a formatted string of the product's details.

The Clothing class extends Product by adding a volume property to represent the space the item occupies. It includes a constructor, a getter for volume, and a static create method that validates input before creating a new Clothing object. The getShippingCost method calculates shipping based on volume and quantity, and toString includes volume in its output.

The Electronics class also extends Product, using weight instead of volume. It has a similar structure to Clothing, with a static create method for validation and object creation. The getShippingCost method calculates shipping based on weight and quantity, and toString includes the weight in its output.

The abstract class Product defines shared attributes:

- category
- product name
- quantity
- unit cost
- margin

Two subclasses extend this abstraction:

- Electronics (adds weight attribute)
- Clothing (adds volume attribute)

Polymorphism is achieved through the abstract method:

```
public abstract double getShippingCost(...)
```

3.2 IO Layer

The class, CSVProductReader, is designed to read product data from two separate CSV files—one for electronics and one for clothing—and store the data into distinct lists of Electronics and Clothing objects. The readElectronics and readClothing methods each handle the reading of a CSV file, processing its contents line by line. The first line (header) is skipped, and each subsequent line is split into comma-separated values. The data is then parsed into the appropriate types, such as String, Integer, and double, with basic validation to ensure the correct number of values are present on each line. Invalid or malformed entries are ignored. The Electronics and Clothing objects are created using the parsed values, and valid objects are added to their respective lists. The use of a BufferedReader ensures efficient reading of the file, and the try-with-resources statement ensures the file is properly closed after processing.

The `CSVProductReader` class handles:

- Reading CSV files using BufferedReader
- Parsing and type conversion
- Validation via factory methods
- Populating ArrayLists

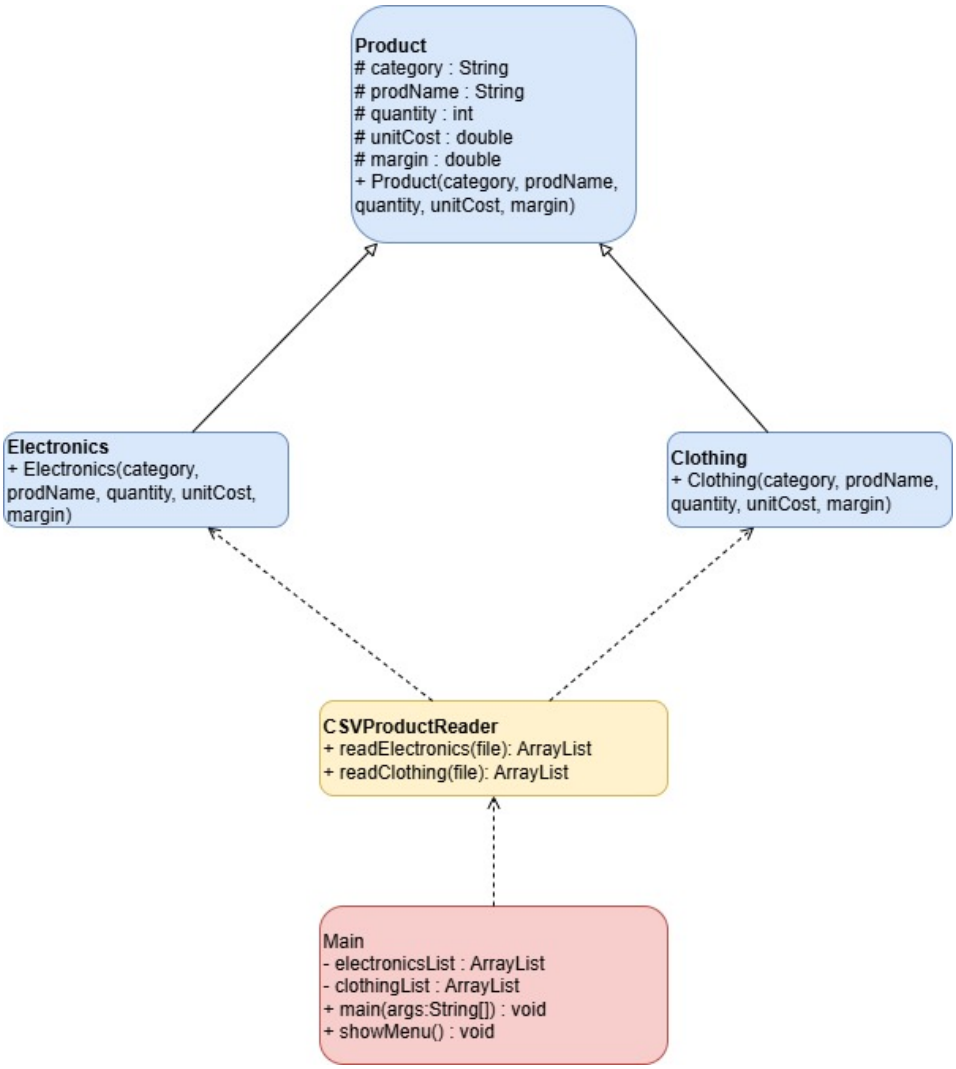


Figure 1: The Model, CSV Data Reader, and UI

3.3 Data Access Layer

The ClothingDAO and ElectronicsDAO classes manage database operations for Clothing and Electronics products, respectively. Both classes provide two key methods:

Batch Insertion: Both classes have a batchInsert method that inserts lists of product objects into the database in batches for efficiency. It uses a PreparedStatement to set product attributes and execute the batch, committing the transaction once done. Errors during insertion are caught and rethrown.

Retrieve All Products: The getAll methods query the database to fetch all product records. Each row is processed by the respective create method (from the Clothing or Electronics class) to validate and convert it into an object, which is then added to a list.

Both classes ensure valid product data is inserted and fetched by using the create method from their respective model classes, and they handle SQL exceptions by logging them without interrupting execution.

The DAO layer consists of:

- ElectronicsDAO
- ClothingDAO

Responsibilities include:

- Batch insertion
- Data retrieval
- Transaction management
- SQL execution
- Utility Layer

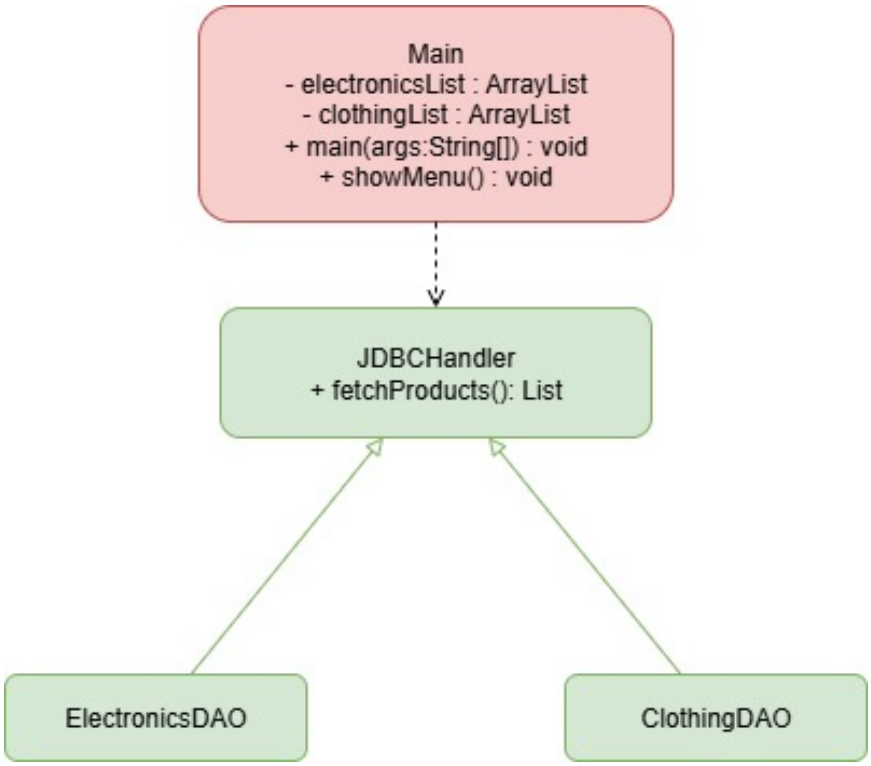


Figure 2: The DAO Layer

The DBUtil class is designed to provide a single, centralized way to access the database connection. It follows the Singleton Pattern, ensuring that only one instance of the class is used throughout the application. This is achieved by making the constructor private, preventing direct instantiation of the class. Instead, the connection is accessed through the static getConnection() method.

The class loads database connection details (URL, username, and password) from environment variables using the dotenv library. If any required variables are missing, an IllegalStateException is thrown. If the environment variables are valid, the method establishes and returns a connection to the database using DriverManager.

By using the Singleton Pattern, the DBUtil class ensures that the database connection logic is centralized, and no unnecessary instances of the class are created. This helps manage the connection in a consistent and efficient way across the application.

The `DBUtil` class encapsulates:

- Environment-based credential loading
- JDBC connection management

3.4 Utility Layer

The Main class serves as the entry point for an inventory system that allows the user to manage and process electronics and clothing products. The program presents a text-based menu to the user, with options to import product data from CSV files, insert data into the database, display stored products, compute inventory values, and calculate shipping costs.

Here's how it works:

1. Import CSV: The user can import electronics or clothing data from CSV files. Once imported, the data is stored in respective lists (electronicsCSVList, clothingCSVList).
2. Insert Data into Database: After importing, the user can insert the product data into the database. If no data is imported, the system prompts the user to import the CSV first.
3. Display Products: The user can view electronics or clothing products stored in the database, with detailed information like category, product name, quantity, unit cost, and more.
4. Inventory Value Calculation: The system can compute the total inventory value, considering both imported data and the data already in the database.
5. Shipping Costs: The user can input a shipping cost per unit (weight for electronics, volume for clothing), and the system will calculate and display the shipping cost for each product, along with the total shipping cost for each category.
6. Exit: The program ends when the user selects the exit option.

Each operation interacts with the database using the ElectronicsDAO and ClothingDAO classes to perform batch inserts and retrieve stored data. Additionally, utility

methods are provided to display product details and calculate shipping costs. The program runs in a loop, waiting for user input until the exit option is chosen.

The `Main` class provides a command-line interface enabling:

- CSV import
- Database insertion
- Inventory computation
- Shipping cost calculation

4 UML Class Diagram

The full system class structure is summarized in the following UML diagram:

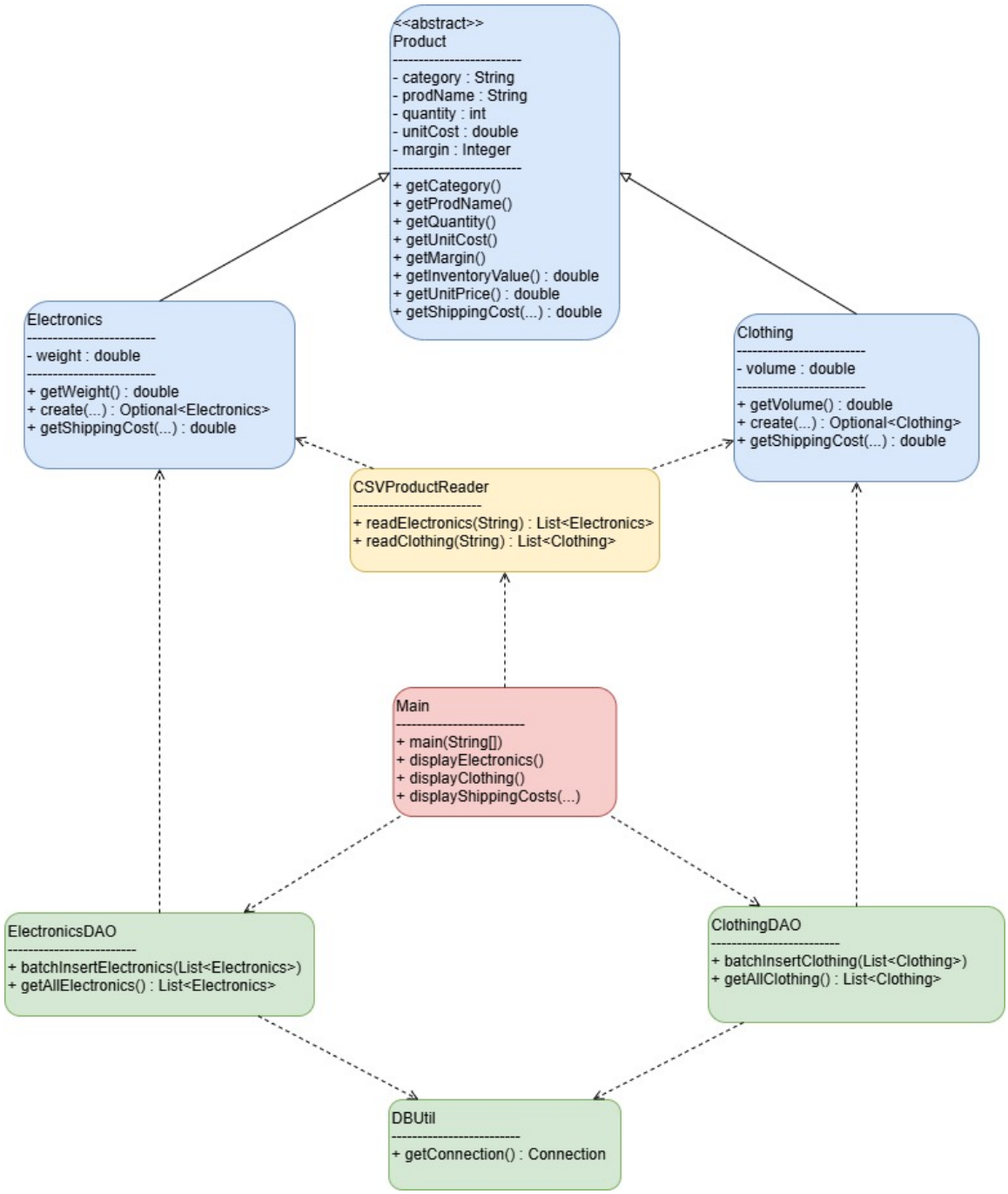


Figure 3: UML Class Diagram

5 Database Design

Two tables are used, one for each product category, with sku number generated automatically.

```sql
CREATE TABLE electronics (
    sku VARCHAR(20) PRIMARY KEY,
    category VARCHAR(50),
    prodName VARCHAR(100),
    quantity INT,
    unitCost DECIMAL(10,2),
    margin INT,
    weight DECIMAL(5,2)
);
```

```sql
CREATE TABLE clothing (
    sku VARCHAR(20) PRIMARY KEY,
    category VARCHAR(50),
    prodName VARCHAR(100),
    quantity INT,
    unitCost DECIMAL(10,2),
    margin INT,
    volume Decimal(10,2)
);
```

	sku	category	prodName	quantity	unitCost	margin	weight
1	SKU_E-0001	Laptop	MacBook Pro	10	1300.00	50	2.00

	sku	category	prodName	quantity	unitCost	margin	volume
1	SKU_C-0001	T-Shirt	Basic White Tee	100	15.00	40	1500.00

Figure 4: SQL Tables

The DBUtil class loads database credentials from a .env file using the dotenv library and provides a JDBC connection.

6 Console Output

After importing a CSV file, the user can display the products stored in the database using options 3 and 4. The output is formatted as a table with clear column headers.

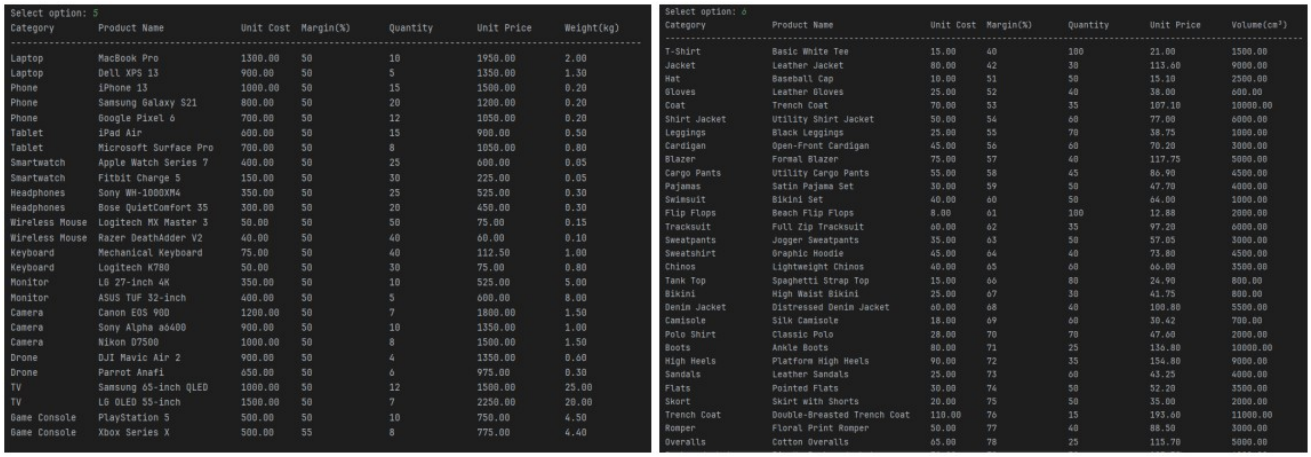
```
Select option: 5
Category        Product Name            Unit Cost  Margin(%)      Quantity       Unit Price     Weight(kg)
-------------------------------------------------------------------------------------------------------
Laptop          MacBook Pro             1300.00    50             10             1950.00        2.00
Laptop          Dell XPS 13             900.00     50             5              1350.00        1.30
Phone           iPhone 13               1000.00    50             15             1500.00        0.20
Phone           Samsung Galaxy S21      800.00     50             20             1200.00        0.20
Phone           Google Pixel 6          700.00     50             12             1050.00        0.20
Tablet          iPad Air                600.00     50             15             900.00         0.50
Tablet          Microsoft Surface Pro   700.00     50             8              1050.00        0.80
Smartwatch      Apple Watch Series 7    400.00     50             25             600.00         0.05
Smartwatch      Fitbit Charge 5         150.00     50             30             225.00         0.05
Headphones      Sony WH-1000XM4         350.00     50             25             525.00         0.30
Headphones      Bose QuietComfort 35    300.00     50             20             450.00         0.30
Wireless Mouse  Logitech MX Master 3    50.00      50             50             75.00          0.15
Wireless Mouse  Razer DeathAdder V2     40.00      50             40             60.00          0.10
Keyboard        Mechanical Keyboard     75.00      50             40             112.50         1.00
Keyboard        Logitech K780           50.00      50             30             75.00          0.80
Monitor         LG 27-inch 4K           350.00     50             10             525.00         5.00
Monitor         ASUS TUF 32-inch        400.00     50             5              600.00         8.00
Camera          Canon EOS 90D           1200.00    50             7              1800.00        1.50
Camera          Sony Alpha a6400        900.00     50             10             1350.00        1.00
Camera          Nikon D7500             1000.00    50             8              1500.00        1.50
Drone           DJI Mavic Air 2         900.00     50             4              1350.00        0.60
Drone           Parrot Anafi            650.00     50             6              975.00         0.30
TV              Samsung 65-inch QLED    1000.00    50             12             1500.00        25.00
TV              LG OLED 55-inch         1500.00    50             7              2250.00        20.00
Game Console    PlayStation 5           500.00     50             10             750.00         4.50
Game Console    Xbox Series X           500.00     55             8              775.00         4.40
```

```
Select option: 6
Category        Product Name                Unit Cost  Margin(%)      Quantity       Unit Price     Volume(cm³)
-------------------------------------------------------------------------------------------------------
T-Shirt         Basic White Tee             15.00      40             100            21.00          1500.00
Jacket          Leather Jacket              80.00      42             30             113.60         9000.00
Hat             Baseball Cap                10.00      51             50             15.10          2500.00
Gloves          Leather Gloves              25.00      52             40             38.00          600.00
Coat            Trench Coat                 70.00      53             35             107.10         10000.00
Shirt Jacket    Utility Shirt Jacket        50.00      54             60             77.00          6000.00
Leggings        Black Leggings              25.00      55             70             38.75          1000.00
Cardigan        Open-Front Cardigan         45.00      56             60             70.20          3000.00
Blazer          Formal Blazer               75.00      57             40             117.75         5000.00
Cargo Pants     Utility Cargo Pants         55.00      58             45             86.90          4500.00
Pajamas         Satin Pajama Set            30.00      59             50             47.70          4000.00
Swimsuit        Bikini Set                  40.00      60             50             64.00          1000.00
Flip Flops      Beach Flip Flops            8.00       61             100            12.80          2000.00
Tracksuit       Full Zip Tracksuit          60.00      62             35             97.20          6000.00
Sweatpants      Jogger Sweatpants           35.00      63             50             57.05          3000.00
Sweatshirt      Graphic Hoodie              45.00      64             40             73.80          4500.00
Chinos          Lightweight Chinos          40.00      65             60             66.00          3500.00
Tank Top        Spaghetti Strap Top         15.00      66             80             24.90          800.00
Bikini          High Waist Bikini           25.00      67             30             41.75          800.00
Denim Jacket    Distressed Denim Jacket     60.00      68             40             100.80         5500.00
Camisole        Silk Camisole               18.00      69             60             30.42          700.00
Polo Shirt      Classic Polo                28.00      70             70             47.60          2000.00
Boots           Ankle Boots                 80.00      71             25             136.80         10000.00
High Heels      Platform High Heels         90.00      72             35             154.80         9000.00
Sandals         Leather Sandals             25.00      73             60             43.25          4000.00
Flats           Pointed Flats               30.00      74             50             52.20          3500.00
Skort           Skirt with Shorts           20.00      75             50             35.00          2000.00
Trench Coat     Double-Breasted Trench Coat 110.00     76             15             193.60         11000.00
Romper          Floral Print Romper         50.00      77             40             88.50          3000.00
Overalls        Cotton Overalls             65.00      78             25             115.70         5000.00
```

Figure 5: Display CSV files

The display methods (displayElectronics() and displayClothing()) retrieve data from the database via the DAO classes and use System.out.printf with fixed column widths for

alignment. The user can display the products retrieved from the database using options 7 and 8.



Select option: 5

Category	Product Name	Unit Cost	Margin(%)	Quantity	Unit Price	Weight(kg)
Laptop	MacBook Pro	1300.00	50	10	1950.00	2.00
Laptop	Dell XPS 13	900.00	50	5	1350.00	1.30
Phone	iPhone 13	1000.00	50	15	1500.00	0.20
Phone	Samsung Galaxy S21	800.00	50	20	1200.00	0.20
Phone	Google Pixel 6	700.00	50	12	1050.00	0.20
Tablet	iPad Air	600.00	50	15	900.00	0.50
Tablet	Microsoft Surface Pro	700.00	50	8	1050.00	0.80
Smartwatch	Apple Watch Series 7	400.00	50	25	600.00	0.05
Smartwatch	Fitbit Charge 5	150.00	50	30	225.00	0.05
Headphones	Sony WH-1000XM4	350.00	50	25	525.00	0.30
Headphones	Bose QuietComfort 35	300.00	50	20	450.00	0.30
Wireless Mouse	Logitech MX Master 3	50.00	50	50	75.00	0.15
Wireless Mouse	Razer DeathAdder V2	40.00	50	40	60.00	0.10
Keyboard	Mechanical Keyboard	75.00	50	40	112.50	1.00
Keyboard	Logitech K780	50.00	50	30	75.00	0.80
Monitor	LG 27-inch 4K	350.00	50	10	525.00	5.00
Monitor	ASUS TUF 32-inch	400.00	50	5	600.00	8.00
Camera	Canon EOS 90D	1200.00	50	7	1800.00	1.50
Camera	Sony Alpha a6400	900.00	50	10	1350.00	1.00
Camera	Nikon D7500	1000.00	50	8	1500.00	1.50
Drone	DJI Mavic Air 2	900.00	50	4	1350.00	0.60
Drone	Parrot Anafi	650.00	50	6	975.00	0.30
TV	Samsung 65-inch QLED	1000.00	50	12	1500.00	25.00
TV	LG OLED 55-inch	1500.00	50	7	2250.00	20.00
Game Console	PlayStation 5	500.00	50	10	750.00	4.50
Game Console	Xbox Series X	500.00	55	8	775.00	4.40

Select option: 8

Category	Product Name	Unit Cost	Margin(%)	Quantity	Unit Price	Volume(cm³)
T-Shirt	Basic White Tee	15.00	40	100	21.00	1500.00
Jacket	Leather Jacket	80.00	42	30	113.60	9000.00
Hat	Baseball Cap	10.00	51	50	15.10	2500.00
Gloves	Leather Gloves	25.00	52	40	38.00	600.00
Coat	Trench Coat	70.00	53	35	107.10	10000.00
Shirt Jacket	Utility Shirt Jacket	50.00	54	60	77.00	6000.00
Leggings	Black Leggings	25.00	55	70	38.75	1000.00
Cardigan	Open-Front Cardigan	45.00	56	60	70.20	3000.00
Blazer	Formal Blazer	75.00	57	40	117.75	5000.00
Cargo Pants	Utility Cargo Pants	55.00	58	45	86.90	4500.00
Pajamas	Satin Pajama Set	30.00	59	50	47.70	4000.00
Swimsuit	Bikini Set	40.00	60	50	64.00	1000.00
Flip Flops	Beach Flip Flops	8.00	61	100	12.88	2000.00
Tracksuit	Full Zip Tracksuit	60.00	62	35	97.20	6000.00
Sweatpants	Jogger Sweatpants	35.00	63	50	57.05	3000.00
Sweatshirt	Graphic Hoodie	45.00	64	40	73.80	4500.00
Chinos	Lightweight Chinos	40.00	65	60	66.00	3500.00
Tank Top	Spaghetti Strap Top	15.00	66	80	24.90	800.00
Bikini	High Waist Bikini	25.00	67	30	41.75	800.00
Denim Jacket	Distressed Denim Jacket	60.00	68	40	100.80	5500.00
Camisole	Silk Camisole	18.00	69	60	30.42	700.00
Polo Shirt	Classic Polo	28.00	70	70	47.60	2000.00
Boots	Ankle Boots	80.00	71	25	136.80	10000.00
High Heels	Platform High Heels	90.00	72	35	154.80	9000.00
Sandals	Leather Sandals	25.00	73	60	43.25	4000.00
Flats	Pointed Flats	30.00	74	50	52.20	3500.00
Skort	Skirt with Shorts	20.00	75	50	35.00	2000.00
Trench Coat	Double-Breasted Trench Coat	110.00	76	15	193.60	11000.00
Romper	Floral Print Romper	50.00	77	40	88.50	3000.00
Overalls	Cotton Overalls	65.00	78	25	115.70	5000.00

Figure 6: Display DB files

7 Batch Insertion

After importing a CSV file, the user can insert the in-memory list into the database using options 5 and 6. The DAO classes (ElectronicsDAO and ClothingDAO) provide batch insertion:

- batchInsertElectronics(List¡Electronics¿ list)
- batchInsertClothing(List¡Clothing¿ list)

Process:

1. Obtain a connection from DBUtil.
2. Disable auto-commit (conn.setAutoCommit(false)).
3. Prepare an INSERT statement for the respective table.
4. For each product, set the parameters and add to the batch.
5. Execute the batch and commit the transaction.
6. If an SQLException occurs, it is thrown to the caller (no explicit rollback is needed because the connection will be closed without commit).

After insertion, the in-memory dbElectronics and dbClothing lists are refreshed by calling getAllElectronics() and getAllClothing().

8 Exception Handling

Proper error handling makes our application robust. Here's our strategy: For IOExceptions during file reading, we catch them in Main and print a friendly message. In the CSVProductReader, if a line has malformed data causing a NumberFormatException, we silently skip that line and keep processing—one bad row won't crash the whole import. SQLExceptions are caught in Main, but in the DAO classes they're rethrown for higher-level handling. And if environment variables are missing, DBUtil throws an exception that propagates to Main.

The project handles several exception types, as described in the following table:

Table 1: Exception Handling Strategy in the CSV–JDBC–MS SQL Project

Exception Type	**Handling Strategy**
`IOException`	Caught in `Main` when reading CSV files; a user-friendly message is printed.
`NumberFormatException`	Caught in `CSVProductReader` during parsing; the malformed line is silently skipped (not added to the `ArrayList`).
`SQLException`	Caught in `Main` during database operations; the error message is printed. In DAO batch insert operations, the exception is rethrown to the caller for higher-level handling.
`IllegalStateException`	Thrown by `DBUtil` if required environment variables are missing; propagated to `Main`.

9 Error Handling

We built a safety container using Optional. When reading a CSV row, we don't blindly create a product. We pass the data to a factory method like Electronics.create(), which acts like a bouncer. It checks if the data is valid—no blank names, no negative quantities. If anything's wrong, it returns an empty Optional. Back in the file reader, we ask, Is there a product inside? If yes, we add it. If not, we skip it. This cleanly handles bad data without throwing exceptions.

```
category,prodName,quantity,unitCost,margin,volume
Jeans, ,50,40.00,41,4000,three
Jacket,Leather Jacket,30,80.00,42,9000,
Jeans,Slim Fit Jeans,sixty,45.00,43,3800
T-Shirt,Graphic Print Tee,70,-20.00,44,1600
Blouse,Button-Down Blouse,80,30.00,-45,2000
Dress,Sundress,40,60.00,46,-5000
00,Mini Skirt,50,25.00,47,1500
Shirt,0,90,22.00,48,2200
Socks,Ankle Socks,150,5.00,49,
Scarf,Knitted Scarf,60,15.00,50
Hat,Baseball Cap,50,10.00,51,2500
Gloves,Leather Gloves,40,25.00,52,600
Coat,Trench Coat,35,70.00,53,10000
Shirt Jacket,Utility Shirt Jacket,60,50.00,54,6000
```

	sku	category	prodName	quantity	unitCost	margin	volume
1	SKU-0001	T-Shirt	Basic White Tee	100	15.00	40	1500.00
2	SKU-0002	Jacket	Leather Jacket	30	80.00	42	9000.00
3	SKU-0003	Hat	Baseball Cap	50	10.00	51	2500.00
4	SKU-0004	Gloves	Leather Gloves	40	25.00	52	600.00
5	SKU-0005	Coat	Trench Coat	35	70.00	53	10000.00
6	SKU-0006	Shirt Jacket	Utility Shirt Jacket	60	50.00	54	6000.00
7	SKU-0007	Leggings	Black Leggings	70	25.00	55	1000.00
8	SKU-0008	Cardigan	Open-Front Cardigan	60	45.00	56	3000.00
9	SKU-0009	Blazer	Formal Blazer	40	75.00	57	5000.00
10	SKU-0010	Cargo Pants	Utility Cargo Pants	45	55.00	58	4500.00
11	SKU-0011	Pajamas	Satin Pajama Set	50	30.00	59	4000.00
12	SKU-0012	Swimsuit	Bikini Set	50	40.00	60	1000.00
13	SKU-0013	Flip Flops	Beach Flip Flops	100	8.00	61	2000.00
14	SKU-0014	Tracksuit	Full Zip Tracksuit	35	60.00	62	6000.00
15	SKU-0015	Sweatpants	Jogger Sweatpants	50	35.00	63	3000.00
16	SKU-0016	Sweatshirt	Graphic Hoodie	40	45.00	64	4500.00
17	SKU-0017	Chinos	Lightweight Chinos	60	40.00	65	3500.00
18	SKU-0018	Tank Top	Spaghetti Strap Top	80	15.00	66	800.00
19	SKU-0019	Bikini	High Waist Bikini	30	25.00	67	800.00
20	SKU-0020	Denim Jac...	Distressed Denim J...	40	60.00	68	5500.00
21	SKU-0021	Camisole	Silk Camisole	60	18.00	69	700.00

Figure 7: Database Insertion

10 Files Submitted

- Java Source Files:
  - model/Product.java
  - model/Electronics.java
  - model/Clothing.java
  - dao/ElectronicsDAO.java
  - dao/ClothingDAO.java
  - util/DBUtil.java
  - io/CSVProductReader.java
  - Main.java
- Maven Configuration: pom.xml
- CSV Files: Electronics.csv, Clothing.csv (in src/main/resources/)
- SQL Script: pcii_SQLQuery17.sql
- Report: (this document)

11 Conclusion

This project successfully demonstrates object-oriented design (abstraction, inheritance) together with file I/O, database connectivity, batch processing, and exception handling. The system reads CSV data, validates it via factory methods, displays it in a clean tabular format, and persists it to a SQL database. All core requirements have been met, and the code is structured with readability and maintainability. Furthermore, the implementation provides a scalable foundation for further extension into enterprise-level applications.