-

Java Project Report

-

-

Professor: Mohammed Omari

-

May 2nd - 2024

-

Team members:

Adnan 2022005845

Amir 2023006017

Hazim 2023005883

-

-

Table of Contents

System Design and Architecture

1. System Overview

The Stock Management System comprises several modules, including:

- **Product Management:** Adding, updating, deleting, and searching for products.
- **Customer Management:** Adding and deleting customers.
- **Supplier Management:** Adding suppliers and restocking products from suppliers.
- **Sales Reporting:** Generating sales reports by customers and suppliers.

2. Architecture Overview

The system follows a modular and layered architecture to ensure scalability, maintainability, and extensibility. The architecture consists of the following layers:

a. Presentation Layer

- The presentation layer handles user interactions through a command-line interface (CLI).
- It displays menus, prompts users for input, and presents information to users.

b. Business Logic Layer

- The business logic layer contains the core logic of the application.
- It orchestrates interactions between different modules and performs business operations such as product management, customer management, and sales reporting.

c. Data Access Layer

- The data access layer manages interactions with external data sources, such as files (e.g., ProductData.txt, CustomerData.txt).
- It reads data from files, updates data, and saves data back to files.

3. Key Components

a. Product Management

- Handles operations related to products, including adding, updating, deleting, and searching for products.
- Provides functionalities to manage product details such as ID, name, description, price, and stock quantity.

b. Customer Management

- Manages customer information, including adding and deleting customers.
- Provides functionalities to handle customer IDs and names.

c. Supplier Management

- Handles supplier-related operations, including adding suppliers and restocking products from suppliers.
- Facilitates interactions with suppliers to replenish stock quantities.

d. Sales Reporting

- Generates sales reports based on customer purchases and supplier restocks.
- Provides insights into sales trends, customer preferences, and supplier activities.

4. Interaction Flow

- The system starts with a login menu, allowing users to log in as a manager or a customer.
- Upon logging in as a manager, users are presented with a manager menu to access various functionalities.
- Managers can perform operations such as viewing stock levels, managing products, managing customers, managing suppliers, and generating sales reports.
- Customers, upon logging in, can view products and make purchases.
- Interactions with the system involve input from users through the CLI, followed by processing of inputs by the business logic layer and data access layer.

Description of key OOP concepts applied.

- **Classes and Objects:** The project defines several classes, such as StockManager, Product, Customer, Supplier, and Save. These classes represent different entities in the system, and objects are created from these classes to hold data and perform operations.
- **Encapsulation:** The classes follow the principle of encapsulation by providing private instance variables and public getter and setter methods to access and modify these variables. For example, the Product class encapsulates attributes like productID, name, description, price, stockQuantity, and sales.
- **Inheritance:** There is no direct implementation of inheritance in the provided code. However, it is a common OOP concept that could be applied to create hierarchies of related classes.
- **Static and Instance Methods:** The code utilizes both static and instance methods. Static methods, such as readProductsFromFile() and displayStockLevels() in the Product class, can be called without creating an instance of the class. Instance methods, like getProductID() and setProductID(), operate on the specific instance of an object.
- **Collections and Data Structures:** The project makes use of various Java collections and data structures, such as ArrayList and Iterator, to store and manipulate data. For example, productsList, customersList, and suppliersList are ArrayList objects that hold instances of Product, Customer, and Supplier classes, respectively.
- **File I/O:** The code implements file input/output operations to read from and write to text files. This is achieved through the use of classes like BufferedReader, BufferedWriter, FileReader, and FileWriter. The Save class contains methods to save data to files, while other classes read data from files.
- **Exception Handling:** The code handles exceptions that may occur during file operations using try-catch blocks. For example, in the readProductsFromFile() method of the Product class, exceptions related to file reading are caught and handled appropriately.

User Manual

Introduction

The Stock Management System is a console-based application designed to help businesses manage their inventory, customers, and suppliers. It provides various features for stock management, sales tracking, and reporting.

Login Options

Upon launching the application, you will be prompted to select one of the following options:

1. **Manager**: Access the manager's menu for stock management, sales reports, and managing products, customers, and suppliers.
2. **Customer**: Access the customer's menu to view and buy products.
3. **Exit**: Exit the application.

Manager Menu

If you select the "Manager" option, you will be presented with the following menu:

1. **View Stock Levels**: Display the current stock levels for all products and alert for low stock levels.
2. **View Sales Reports**: View sales reports for customers and suppliers.
3. **Manage Products**: Add, update, delete, or search for products.
4. **Manage Customers**: Add, delete, or view customers.
5. **Manage Suppliers**: Add suppliers and restock products.
6. **Logout**: Log out and return to the login menu.

View Stock Levels

This option displays the current stock quantity for each product. It also alerts you if the stock level of any product is low (less than or equal to 5 units).

View Sales Reports

This option allows you to view sales reports for customers or suppliers.

1. **Sales Report of Customers**: View the sales report for each customer, displaying the quantities of each product they have purchased.
2. **Sales Report of Suppliers**: View the sales report for each supplier, displaying the quantities of each product they have supplied.

Manage Products

This option provides the following sub-options for managing products:

1. **Add Product**: Add a new product to the system by providing the product ID, name, description, price, and stock quantity.
2. **Update Product**: Update the details of an existing product, such as ID, name, description, or price.
3. **Delete Product**: Delete an existing product from the system.
4. **Search for Product**: Search for a product by name, category, or ID.
5. **View Products**: Display a list of all products in the system.

Manage Customers

This option provides the following sub-options for managing customers:

1. **Add Customer**: Add a new customer to the system by providing the customer ID and name.
2. **Delete Customer**: Delete an existing customer from the system.
3. **View Customers**: Display a list of all customers in the system.

Manage Suppliers

This option provides the following sub-options for managing suppliers:

1. **Add Supplier**: Add a new supplier to the system by providing the supplier ID and name.

2. **Restock Products**: Restock a product by specifying the product name, supplier name, and quantity.
3. **View Suppliers**: Display a list of all suppliers in the system.

Customer Menu

If you select the "Customer" option, you will be presented with the following menu:

1. **View Products**: Display a list of all available products in the system.
2. **Buy Products**: Purchase a product by specifying the product name and quantity.
3. **Logout**: Log out and return to the login menu.

View Products

This option displays a list of all available products in the system, showing the product ID, name, description, price, and stock quantity.

Buy Products

This option allows you to purchase a product by entering the product name and desired quantity. The system will check if the requested quantity is available and complete the transaction if possible. After a successful purchase, the product's stock quantity will be updated accordingly.

Data Storage

The application stores data in the following text files:

- `ProductData.txt`: Contains information about products (product ID, name, description, price, and stock quantity).
- `CustomerData.txt`: Contains information about customers (customer ID and name).
- `SupplierData.txt`: Contains information about suppliers (supplier ID and name).
- `SalesData.txt`: Stores sales transactions, including purchases made by customers and restocking by suppliers.

These files are automatically updated whenever changes are made to the corresponding data (products, customers, suppliers, or sales transactions).

- ## Exiting the Application

To exit the Stock Management System, simply select the "Exit" option from the login menu or the "Logout" option from the manager or customer menus.

Note: Any unsaved changes made during the current session will be lost upon exiting the application.

Challenges faced and Lessons learnt

1. **Understanding File I/O Operations**: One of the biggest challenges would have been understanding how to read and write data from/to files. Working with files and handling file operations is a fundamental skill in programming, but it can be daunting for a beginner. Lessons learned would include:
    - How to create, open, read, and write files using Java's File I/O classes (e.g., `FileReader`, `FileWriter`, `BufferedReader`, `BufferedWriter`).
    - Handling exceptions and error cases that may occur during file operations.
    - Understanding file paths and working with different directories.
2. **Data Structures and Collections**: Effectively managing and storing data is crucial in a system like this. Working with data structures and collections would have been a learning experience, such as:
    - Understanding and using collections like `ArrayList` to store and manipulate data.
    - Learning how to iterate over collections using loops or iterators.
    - Implementing search, update, and deletion operations on collections.
3. **Object-Oriented Programming (OOP) Concepts**: Building a system like this requires a good understanding of OOP principles. Lessons learned would include:
    - Creating classes and objects to represent different entities (e.g., `Product`, `Customer`, `Supplier`).

    - Applying encapsulation by defining instance variables and providing getter and setter methods.
    - Understanding the purpose and implementation of static and instance methods.
4. **User Interface and Input Handling**: Although this system uses a console-based interface, handling user input and providing a user-friendly experience can be challenging. Lessons learned would include:
    - Implementing menu systems using loops and conditional statements.
    - Validating and handling user input to prevent errors and unexpected behavior.
    - Providing clear instructions and feedback to the user.
5. **Code Organization and Modularity**: As the codebase grows, organizing and structuring code becomes increasingly important. Lessons learned would include:
    - Separating concerns by creating different classes for different functionalities (e.g., `StockManager`, `Product`, `Customer`, `Supplier`, `Save`).
    - Understanding the importance of code reusability and modular design.
    - Implementing methods and functions to encapsulate specific tasks or operations.
6. **Debugging and Testing**: Debugging and testing are essential skills for any programmer. Lessons learnt:
    - Writing test cases to ensure the correctness of individual components or the entire system.
    - Learning how to effectively troubleshoot and resolve errors or unexpected behavior.
7. **Problem-Solving and Logical Thinking**: Developing a system like this requires breaking down complex problems into smaller, manageable tasks. Lessons learnt:
    - Applying problem-solving techniques to understand requirements and design solutions.
    - Developing logical thinking skills to analyze and implement algorithms or processes.
    - Learning how to research and seek help from documentation, online resources, or more experienced programmers and strengthen team communication skills.