

Project-Based Learning in Java
A PROJECT REPORT

Submitted by

Harsh Ranjan – 23BCS11745

Simple Inventory Management System
PROJECT REPORT

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCINENCE AND ENGINEERING



Chandigarh University



BONAFIDE CERTIFICATE

This is to certify that the project report entitled “University Exam Result Processing System” is the Bonafide and genuine work carried out by Harsh Ranjan, Pranya Garg, and Gunmeet Kaur, students of Computer Science and Engineering – 3rd year, in partial fulfillment of the requirements for the award of the Bachelor’s Degree in Engineering from Chandigarh University.

The project work embodied in this report has been carried out under my supervision and guidance. The work presented herein is an outcome of the students’ original effort, sincere research, and practical understanding of the subject.

I hereby certify that the project fulfills all the necessary standards and requirements of academic work prescribed by the university and that the students have shown commendable performance, dedication, and teamwork throughout the completion of this project.

Supervisor’s Name: _____

Department: _____

Signature: _____

Date: _____

ABSTRACT

This report presents a comprehensive overview of the design, development, and implementation of a Simple Inventory Management System, which is a console-based Java application created to illustrate the use of fundamental Object-Oriented Programming (OOP) principles and core data structures in a practical environment. The primary goal of this project is to provide an efficient, easy-to-use, and scalable solution for managing inventory data within a small-scale business or educational context.

At the heart of the system lies the `Item` class, which encapsulates all essential product-related attributes such as Item ID, Name, Quantity, and Price. This class demonstrates key OOP concepts, including encapsulation, data abstraction, and class-based modularity, ensuring that each item is represented as a well-defined object with clearly structured data. The `InventoryManager` class serves as the central control unit of the system, maintaining a dynamic collection of `Item` objects through an `ArrayList`, a flexible data structure from the Java Collections Framework that allows efficient insertion, deletion, and traversal operations.

The system provides several core functionalities that enhance user interaction and demonstrate the practical use of loops, conditional statements, and data handling techniques. These include:

1. Adding new inventory records, where each item is automatically assigned a unique ID to avoid duplication and maintain data consistency.
2. Viewing the complete stock list, which displays all stored items in a formatted tabular structure, allowing users to easily monitor inventory levels.
3. Updating the quantity or details of existing items using an ID-based search mechanism, ensuring that updates are precise and restricted to valid records.

To manage user interaction effectively, the project adopts a menu-driven interface. It employs a combination of a `while` loop for continuous operation and a `switch` statement to handle user choices systematically. Although the current version of the system operates entirely in-memory, without saving data to permanent storage, it lays a strong foundation for future development. Potential enhancements include integrating file handling (I/O) for data persistence, enabling the application to store and retrieve records even after the program is closed. Additional improvements may involve implementing advanced search and sorting functionalities, a graphical user interface (GUI) for better usability, and database connectivity for real-world deployment.

ACKNOWLEDGEMENT

We, the members of this project team, would like to express our heartfelt gratitude and deep appreciation to all those who have supported, guided, and inspired us throughout the completion of our project, “Simple Inventory Management System”, in November 2025. This project marks an important milestone in our academic journey, and it would not have been possible without the help and encouragement of several individuals and institutions who contributed to our success.

First and foremost, we would like to thank God Almighty for His continuous blessings, wisdom, and guidance. His grace has been our strength throughout this endeavor, helping us stay motivated, focused, and determined even in challenging times. We are truly grateful for the perseverance and clarity that our faith has instilled in us, allowing us to complete this project successfully.

We extend our sincere thanks to our teachers and mentors for their invaluable guidance, encouragement, and constant supervision. Their expert advice, constructive feedback, and patient support have been instrumental in shaping our understanding and improving the quality of this project. Their enthusiasm for teaching and research has inspired us to think critically and strive for excellence in every aspect of our work.

We are also deeply thankful to our families for their unconditional love, patience, and encouragement. Their constant moral support and belief in our abilities have been a source of immense motivation throughout this project. Their understanding during long hours of research, coding, and documentation has been crucial to our success.

We would also like to extend our gratitude to the Department of Computer Science and Engineering, Chandigarh University, Gharuan, Mohali, Punjab, for providing us with the necessary facilities, infrastructure, and academic environment to carry out our project effectively. The department’s continuous emphasis on innovation, teamwork, and technical learning has greatly enriched our experience and helped us gain practical insights into software development. Finally, we wish to acknowledge all our friends and peers who provided valuable suggestions, shared knowledge, and encouraged us during this project. Their collaboration and insights have contributed immensely to improving our work and keeping our spirits high throughout this journey. We wholeheartedly thank every individual who, directly or indirectly, contributed to the successful completion of the “Simple Inventory Management System”. Your guidance, encouragement, and support have made this project a rewarding and memorable experience, helping us grow both personally and professionally.

INTRODUCTION AND PROJECT GOAL

The Simple Inventory Management System is a console-based Java application developed to effectively demonstrate the practical use of fundamental Object-Oriented Programming (OOP) concepts, core data structures, and interactive user interfaces through a text-based command-line environment. This project serves as an excellent example of how object-oriented design principles can be applied to create a modular, maintainable, and efficient real-world system for managing inventory in small-scale business environments or academic simulations.

The main objective of developing this project was to build a functional and user-friendly system that can efficiently handle basic inventory management operations while maintaining simplicity and clarity in both design and execution. The system focuses on organizing and manipulating product data using an object-oriented approach that emphasizes encapsulation, reusability, and data abstraction.

The project is designed to achieve the following key goals:

- Storing and managing a collection of product records: The system utilizes an ArrayList from the Java Collections Framework to dynamically store product data. This structure allows efficient insertion, deletion, and retrieval operations, enabling smooth management of inventory items without the need for static data structures.
- Allowing users to add new items with unique details: Each product in the inventory is represented by an instance of the Item class, which contains attributes such as unique ID, item name, quantity, and price. The system automatically generates a unique ID for every new entry, ensuring that no two products share the same identifier. This feature enhances data accuracy and prevents duplication errors.
- Displaying the entire inventory list: The system provides a clear and organized view of all stored items, displaying them in a tabular format that includes the ID, name, quantity, and price of each product. This feature helps users keep track of available stock and monitor overall inventory status at a glance.
- Checking and updating stock quantities: Users can search for any product using its unique ID and update the quantity whenever stock changes occur. This ensures that the system always reflects accurate and up-to-date information, which is essential for inventory control and decision-making.

Beyond these core functionalities, the Simple Inventory Management System also emphasizes ease of use and interactivity. It features a menu-driven interface, allowing users to navigate

through various options using a combination of while loops and switch statements. This control structure not only ensures continuous operation until the user decides to exit but also enhances program readability and maintainability. Additionally, basic input validation has been implemented to prevent runtime errors and ensure that only valid data is processed by the system.

From a technical perspective, the project effectively demonstrates how OOP principles like encapsulation, modularity, and data abstraction can be used to organize and protect data. The design ensures that each class has a specific responsibility—making the program easier to understand, debug, and extend in the future.

While the current implementation maintains data in temporary memory (RAM) during execution, it provides a solid foundation for future improvements. Possible extensions include implementing file handling to achieve data persistence, allowing records to be saved and retrieved between sessions. Additional enhancements could involve adding features such as item deletion, search by name or price range, sorting options, or even developing a Graphical User Interface (GUI) using JavaFX or Swing for improved user experience.

TECHNOLOGY AND CORE CONCEPTS

Component	Description	Java Concept Demonstrated
Item Class	Serves as the blueprint for an inventory record. It utilizes private fields and public Getters and Setters.	Object-Oriented Programming (OOP), Encapsulation
InventoryManager Class	Contains the main application logic, the user menu loop, and the methods for data manipulation.	Modularity, Program Entry Point (main)
List<Item> (ArrayList)	The primary data structure used to store the items. It allows for dynamic resizing as items are added.	Collections Framework, Dynamic Data Structures
Scanner	Used to facilitate user input from the console for menu navigation and data entry.	User Input Handling
while Loop & switch	Implements the continuous Menu Loop and controls the program flow based on the user's menu choice.	Control Flow

KEY DESIGN AND STRUCTURE

The Simple Inventory Management System was designed with a strong emphasis on modularity, data organization, and maintainability. The system's architecture is divided into distinct components, each responsible for a specific aspect of the program's functionality. This design ensures clean separation of concerns, reduces code redundancy, and makes the overall system easier to extend in the future.

A. Data Structure

The entire inventory is managed using a static `List<Item>` named `inventory`. The decision to use an `ArrayList` as the underlying data structure was made because the total number of items in the inventory is dynamic — it can grow or shrink during runtime depending on user operations such as adding or removing items. Unlike arrays, which have a fixed size, `ArrayList` provides the flexibility to handle a variable amount of data efficiently. It supports operations such as insertion, deletion, and traversal without the need for manual memory management. Moreover, its built-in methods and dynamic resizing capabilities make it an ideal choice for this kind of real-time data management system.

The use of a static list also ensures that the inventory data remains accessible across different methods within the `InventoryManager` class without requiring repeated instantiation or data passing.

B. Item Class (`Item.java`)

The `Item` class plays a central role in defining the structure and behavior of each product within the inventory. It serves as the blueprint for creating item objects and effectively demonstrates core Object-Oriented Programming principles such as encapsulation and abstraction.

- Attributes:

The class contains the following fields:

- `id (int)` – a unique identifier automatically assigned to each item.
- `name (String)` – the product name.
- `quantity (int)` – the available stock quantity.
- `price (double)` – the cost per unit of the item.

- Encapsulation:

All attributes are declared as `private`, meaning they cannot be directly accessed from

outside the class. This design prevents unauthorized modification of internal data and maintains data integrity. Access and updates to these attributes are managed exclusively through getter (get...()) and setter (setQuantity()) methods. This ensures controlled and validated data handling, which is a hallmark of good OOP design.

- toString() Override:
The toString() method has been overridden to generate a neatly formatted and human-readable string representation of each item. When the inventory is displayed, this method ensures that the output is clear, consistent, and easy to interpret by users, eliminating the need for manual string formatting during display operations.

The Item class, therefore, not only encapsulates data but also provides the necessary interface for interaction between the program logic and the data model, establishing a clean and modular system architecture.

C. Management Logic (InventoryManager.java)

The InventoryManager class acts as the main control center of the application, containing the logic that drives all major operations. It integrates user input, performs validation, and manipulates the inventory data through appropriate method calls. The key methods and their responsibilities are as follows:

1. addItem():
This method prompts the user to enter details such as item name, quantity, and price. It includes input validation to ensure that the entered quantity and price are numeric and valid. Once verified, a new Item object is created with an auto-generated unique ID (nextId). This ID is incremented automatically for each new item, preventing duplication. The new item is then added to the inventory list, effectively updating the stock records.
2. viewInventory():
This method provides an overview of all the products stored in the inventory. It uses an enhanced for loop to iterate through each element in the list and print its details using the toString() method from the Item class. The result is a clean and structured display of all available items, which helps the user monitor inventory levels easily.
3. checkStock():
This method allows the user to search for an item by its unique ID. Once the ID is entered, the program scans through the inventory list to find a matching record. If the item exists,

the system displays its current quantity and provides an option to update the stock using the `setQuantity()` method. This function ensures that stock management remains accurate and up-to-date, which is crucial for maintaining real-time inventory control.

Overall, the Key Design and Structure of the project reflects thoughtful planning and application of software engineering principles. The combination of modular class design, robust data handling, and logical program flow ensures that the system is efficient, easy to use, and scalable for future enhancements such as file I/O operations, sorting, searching, or even integration with databases or graphical interfaces.

PROJECT LIMITATIONS AND FUTURE ENHANCEMENTS

While functional, this console application has several limitations that can be addressed in future development:

Limitation	Suggested Enhancement
Data Volatility	The inventory data is lost every time the program closes (data is only held in memory).
Limited Search	Users can only look up items by their unique ID.
No Item Removal	There is no option to permanently remove a discontinued item from the inventory.
User Experience	The application is entirely text-based.

CONCLUSION

The Simple Inventory Management System stands as a clear demonstration of how core Java programming concepts can be effectively applied to design and implement a functional, modular, and user-friendly application. Throughout its development, the project successfully integrates essential elements such as class design, object-oriented programming principles, data encapsulation, and dynamic data structures like the ArrayList, which together create a structured and maintainable codebase. The use of a menu-driven control flow, managed through loops and switch-case constructs, ensures smooth user interaction and logical navigation between different functionalities.

This project not only fulfills its immediate purpose of managing inventory records — including adding new items, viewing the complete inventory list, and updating existing stock quantities — but also serves as an educational tool that deepens the understanding of how Java's OOP features can be applied to solve real-world problems. By structuring data within well-defined classes and using private attributes with getter and setter methods, the system emphasizes data integrity and controlled access, reinforcing the importance of clean code practices and modular design.

In addition, the program lays a strong foundation for future development and scalability. Several enhancements can be introduced to elevate its functionality, such as data persistence using file handling or databases, search and sort operations, item deletion, and report generation to analyze inventory trends. Further improvements could also include the integration of a Graphical User Interface (GUI) using JavaFX or Swing, making the system more interactive and visually appealing for end-users. Implementing features like user authentication, real-time notifications for low stock, and exporting data to files would transform it into a more robust and professional-grade inventory management solution.

Overall, the Simple Inventory Management System demonstrates not only the technical proficiency required to build a complete Java application but also highlights the developer's ability to apply theoretical knowledge to practical, real-world scenarios. It represents an excellent stepping stone for understanding software design patterns, system organization, and iterative enhancement. With its clean structure, logical flow, and potential for expansion, this project is a testament to the power and versatility of Java as a programming language for developing efficient, reliable, and scalable software systems.