IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 6, November 2026
Impact Factor: 8.2

A Software Framework for Inventory and Supply Chain Management using Java

Dr. K. Kasturi[1] and Vincent Mariya Singam P.[2]
[1] Professor, Department of Applied Computing and Emerging Technologies
[2]Student, Department of Applied Computing and Emerging Technologies
School of Computing Sciences, VISTAS, Chennai, Tamil Nadu, India
[1] kasturi.scs@vistas.ac.in,[2] vincentmariyasingam2303@gmail.com

Abstract : . *The proposed application demonstrates how a software solution can maintain product records, supplier information, procurement orders, customer orders, shipment details and stock movements. These functions are connected to common activities in retail, distribution and warehouse operations.The work is organised into four tasks. Task 1 focuses on product registration and inventory maintenance using Core Java and object-oriented programming. Task 2 covers supplier registration and procurement management, including purchase-order records. Task 3 explains customer order processing and supply chain tracking. Task 4 focuses on stock monitoring, low-stock alerts and replenishment suggestions.Java classes represent entities such as Product, Supplier, PurchaseOrder, CustomerOrder and Shipment. A relational database can store the records, while Java Database Connectivity (JDBC) can connect the application to the database. SQL supports operations such as inserting, retrieving, updating and deleting records. Input validation, exception handling and testing are considered throughout the design.*

Keywords: Java, Inventory Management, Supply Chain, JDBC, SQL, Procurement, Order Processing, Stock Replenishment.

I. INTRODUCTION

Java is a general-purpose programming language used to develop desktop, web-based and enterprise applications. Its object-oriented features allow a program to be organised into classes and objects that represent real-world entities. Java also provides features for collections, exception handling, file handling and database connectivity.

Inventory management is the process of recording and controlling goods held by an organisation. It includes maintaining product details, monitoring available quantities, recording goods received and goods issued, and identifying products that need replenishment. Accurate stock information helps an organisation reduce shortages, avoid unnecessary excess inventory and respond to customer requirements.

Supply chain management involves the movement of products and information from suppliers through procurement and storage to order fulfilment and delivery. A basic information system can connect these activities by linking supplier records, purchase orders, inventory balances, customer orders and shipment status[1].

Manual registers and separate spreadsheets can lead to duplicated records, delayed updates and calculation mistakes. A computerised application can store information in a common database and apply consistent validation rules. Java can implement the application logic, while a relational database can preserve structured records.

II. TECHNOLOGIES AND TOOLS USED

Java: Java is the main programming language that supports object-oriented programming, reusable methods, exception handling and application development.JDK and JVM: The Java Development Kit provides the compiler and development tools. The Java Virtual Machine executes compiled Java bytecode.IDE: Eclipse or IntelliJ IDEA can be

Copyright to IJARSCT
www.ijarsct.co.in
DOI: 10.48175/IJARSCT-41133

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 6, November 2026 Impact Factor: 8.2

used to create, organise, run and debug Java programs.Object-Oriented Programming: Encapsulation, inheritance, abstraction and polymorphism help organise code into reusable components. For example, a Product class stores product attributes and provides methods to access or update them.JDBC: Java Database Connectivity is an API used to connect Java applications to relational databases. Connection, PreparedStatement and ResultSet are common JDBC interfaces used to execute SQL statements and process query results.

SQL and Relational Database: SQL is used to create tables and manipulate records. MySQL or PostgreSQL may be used to store product, supplier, purchase-order, customer-order and shipment information. Mention only the database actually used.Supporting tools: Git can be used for version control.

III. DESCRIPTION OF TASKS

3.1 Task 1: Inventory Management Using Java

Objective. The first task is to design the product inventory module. Its purpose is to maintain product details and provide operations to add, view, search, update and remove product records.Functional requirements[2]. A product record can contain a unique product ID, product name, category, unit price and available quantity. The application should reject missing product names, negative prices and negative stock quantities. The product ID is used to identify a record during search, update and deletion.Program structure. A Product class represents a product. An InventoryService class manages product records and prov*ides methods such as addProduct(), findProduct(), updateQuantity() and removeProduct(). A main class can display a menu and accept user input. A database version can use a ProductDAO class to perform JDBC operations. A Software Framework for Inventory and Supply Chain Management Using Java

Object-oriented concepts. Product attributes can be declared private to apply encapsulation. A constructor assigns initial values, while getter and setter methods provide controlled access. Validation can be placed in setter methods or service classes to prevent invalid values from entering the system.Example workflow. The user selects Add Product, enters the product ID, name, category, price and quantity. The program validates the input and stores the record. The user can then display all products or search for a particular product by ID.

Illustrative Java code:

```
public class Product {
private int productId;
private String productName;
private double unitPrice;
private int quantity;

public Product(int id, String name,
double price, int quantity) {
this.productId = id;
this.productName = name;
this.unitPrice = price;
setQuantity(quantity);
}

public int getProductId() { return productId; }

public void setQuantity(int quantity) {
if (quantity < 0)
throw new IllegalArgumentException(
"Quantity cannot be negative");
this.quantity = quantity;
}
}
```

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 6, November 2026
Impact Factor: 8.2

The code demonstrates a Product class with private fields, a constructor, a getter and a setter. The setter checks that the quantity is not negative. Similar validation should be applied to the product name and unit price. Testing. Test adding a valid product, searching for an existing and non-existing product ID, updating quantity and entering a negative quantity. The program should display a clear message when an operation fails.

3.2 Task 2: Supplier and Procurement Management

Objective. The second task extends the application to supplier records and procurement activities. Its purpose is to maintain supplier information and record purchase orders raised to obtain products.Supplier details. A supplier record may include supplier ID, supplier name, contact number, email address and business address. Required fields should be validated, and duplicate supplier IDs should be prevented. Supplier records should be searchable so users can identify the supplier associated with a purchase order.

Purchase order details. A purchase order may contain a purchase order number, supplier ID, order date and order status. The order items can contain product ID, ordered quantity and unit purchase price. Typical order statuses include Draft, Placed, Partially Received, Received and Cancelled.Suggested Java classes. Supplier represents supplier details. PurchaseOrder represents an order placed with a supplier. ProcurementService contains the business rules for creating and updating purchase orders. SupplierDAO and PurchaseOrderDAO can contain database access methods.

Database relationship. The supplier ID in a purchase-order record can refer to the primary key of the Supplier table. This foreign-key relationship ensures that a purchase order is associated with a supplier record.

Illustrative SQL structure:

```
CREATE   TABLE   Supplier   (
supplier_id INT PRIMARY KEY,
supplier_name   VARCHAR(100)   NOT   NULL,
phone VARCHAR(20),
email VARCHAR(120)
);

CREATE   TABLE   Purchase_Order   (
purchase_order_id  INT  PRIMARY  KEY,
supplier_id     INT     NOT     NULL,
order_date DATE NOT NULL,
status VARCHAR(30) NOT NULL,
FOREIGN      KEY      (supplier_id)
REFERENCES Supplier(supplier_id)
);
```

The Supplier table stores supplier contact information. The Purchase_Order table stores the order reference, supplier, date and status. In a complete application, a separate Purchase_Order_Item table can store the products and quantities included in each purchase order.Workflow. The user registers a supplier, selects that supplier and creates a purchase order. The system validates the supplier ID and order details before saving the record. When goods arrive, the order status can be changed to Received or Partially Received, and the received quantity can be added to inventory.

Testing and expected outcome. Test supplier registration, supplier search, purchase-order creation and status updates. Check that a purchase order cannot refer to a supplier that does not exist. This task introduces database relationships, primary keys, foreign keys and procurement workflow management.

3.3 Task 3: Order Processing and Supply Chain Tracking

Objective. The third task covers customer order processing and basic shipment tracking. Its purpose is to connect customer demand with available inventory and maintain a record of order progress.

Copyright to IJARSCT
www.ijarsct.co.in
DOI: 10.48175/IJARSCT-41133

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 6, November 2026 Impact Factor: 8.2

Customer order. An order can contain an order ID, customer ID, order date and order status. Each order may contain one or more order items. An order item identifies a product, requested quantity and selling price. Before confirmation, the application checks whether enough stock is available.Stock control. When an order is confirmed, the system reduces available stock by the fulfilled quantity. Related database updates should be handled consistently. A database transaction can commit all related changes together or roll them back if an operation fails.

Shipment tracking. Shipment information may include shipment ID, order ID, dispatch date, carrier reference and delivery status. Example status values include Processing, Packed, Dispatched, In Transit and Delivered.Validation. The application should reject zero or negative order quantities, unknown product IDs and quantities greater than available stock. It should also ensure that rejected orders do not change the inventory balance.

Illustrative Java logic:

```
public int confirmOrder(int requestedQty,
int availableQty) {
if (requestedQty <= 0) {
throw new IllegalArgumentException(
"Quantity must be positive");
}
if (requestedQty > availableQty) {
throw new IllegalStateException(
"Insufficient stock");
}
return availableQty - requestedQty;
}
```

The method checks whether the requested quantity is valid and whether sufficient stock is available. If both checks pass, it returns the remaining quantity. In a complete application, the result must be saved to the database as part of a controlled order-confirmation transaction.Sample scenario. If 25 wireless mice are available and a customer orders 2, the order can be accepted and the remaining quantity becomes 23. If the customer requests 30, the system should reject the order and leave the stock quantity unchanged.

Testing and expected outcome. Test valid orders, insufficient-stock orders, zero and negative quantities, and orders for unknown products. Test shipment status updates and verify that each shipment is linked to the correct customer order. This task demonstrates business logic, validation and linked transaction records.

3.4 Task 4: Inventory Monitoring and Stock Replenishment

Objective. The fourth task focuses on monitoring stock levels and identifying products that need replenishment. It aims to help users recognise low-stock items before inventory becomes unavailable.

Reorder threshold. Each product can be assigned a reorder level or minimum stock threshold. When the available quantity is equal to or below this level, the application displays a low-stock warning. The reorder quantity may be configured by the organisation or determined from expected demand and supplier lead time. Illustrative scenario. A product has 10 units available and a reorder level of 15 units. Since the available quantity is below the threshold, the system identifies the product as low stock. If the configured reorder quantity is 20, the application can prepare a draft purchase order for 20 units.

Stock movement history. A stock movement record can capture each increase or decrease in quantity. It may include movement ID, product ID, movement type, quantity, movement date and reference ID. Movement types can include Purchase Receipt, Customer Dispatch and Stock Adjustment.Business value. A low-stock list gives users a quick view of products that may need attention. Recording stock movements also makes it easier to review why a product balance changed.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 6, November 2026
Impact Factor: 8.2

Illustrative Java logic:

```
public int confirmOrder(int requestedQty,
int availableQty) {
if (requestedQty <= 0) {
throw new IllegalArgumentException(
"Quantity must be positive");
}
if (requestedQty > availableQty) {
throw new IllegalStateException(
"Insufficient stock");
}
return availableQty - requestedQty;
}
```

Illustrative SQL query:

```
public int confirmOrder(int requestedQty,
int availableQty) {
if (requestedQty <= 0) {
throw new IllegalArgumentException(
"Quantity must be positive");
}
if (requestedQty > availableQty) {
throw new IllegalStateException(
"Insufficient stock");
}
return availableQty - requestedQty;
}
```

The Java method compares available quantity with the reorder level. The SQL query retrieves products whose quantity is at or below the configured threshold. A complete application can display these products in a low-stock report and allow an authorised user to prepare a purchase order.Testing and expected outcome. Test quantities above, equal to and below the reorder threshold. Verify that only the correct products appear in the low-stock list. After a purchase receipt, verify that the inventory quantity and stock movement history are updated correctly.

IV. OUTCOMES

The four modules in this application demonstrate how Java programming can be applied to a business-oriented inventory and supply chain problem. The inventory module shows how product information can be organized into classes and maintained through add, search, update and delete operations. Input validation helps prevent invalid product names, prices and quantities.

The supplier and procurement module introduce supplier records and purchase-order information. It demonstrates how related records can be connected through identifiers and database relationships. The order-processing module explains how customer orders can be checked against available inventory and how shipment status can be associated with an order. The stock-monitoring module demonstrates conditional logic and database queries for identifying products that have reached their reorder threshold. A stock-movement history can provide a record of goods received, goods dispatched and manual adjustments. Together, these modules illustrate how separate business activities can be represented in one software application.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 6, November 2026
Impact Factor: 8.2

V. CONCLUSION

This paper outlined a Java-based framework for connecting core inventory and supply chain activities. Its four modules show how object-oriented classes can represent products, suppliers, purchase orders, customer orders, shipments, and stock movements. Validation rules, relational database relationships, and transaction handling provide a foundation for maintaining consistent records as goods are received, ordered, and dispatched.

APPENDIX A – Colour Application Screenshots

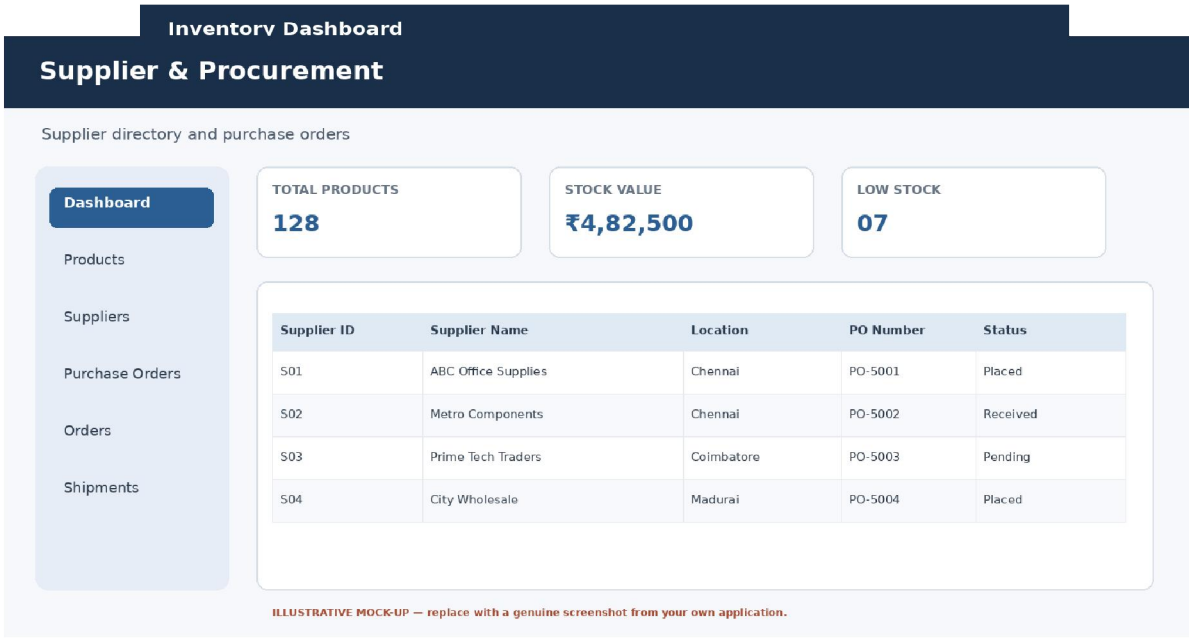


Figure A1. Inventory product dashboard – illustrative mock-up.

Figure A2. Supplier and procurement management – illustrative mock-up.

APPENDIX B – Java Code and Application Screenshots



Figure B1. Product.java – product data model and validation.

Copyright to IJARSCT
www.ijarsct.co.in
DOI: 10.48175/IJARSCT-41133

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 6, November 2026
Impact Factor: 8.2

InventoryService.java

```java
import java.util.*;

public class InventoryService {
    private Map<Integer, Product> products =
        new HashMap<>();

    public void addProduct(Product product) {
        int id = product.getProductId();
        if (products.containsKey(id))
            throw new IllegalArgumentException(
                "Product ID already exists");
        products.put(id, product);
    }

    public Product findProduct(int id) {
        return products.get(id);
    }
}
```

ILLUSTRATIVE CODE — compile and test; use your own IDE screenshots for final evidence.

Figure B2. InventoryService.java – product collection operations.

StockMonitor.java

```java
public class StockMonitor {
    public static boolean isLowStock(
            int available, int reorderLevel) {
        return available <= reorderLevel;
    }

    public static void main(String[] args) {
        int available = 10;
        int reorderLevel = 15;

        if (isLowStock(available, reorderLevel))
            System.out.println("LOW STOCK ALERT");
    }
}
```

ILLUSTRATIVE CODE — compile and test; use your own IDE screenshots for final evidence.

Figure B3. StockMonitor.java – low-stock check.

REFERENCES

1. Kasturi, K., & Jebathangam, J. (2023). Supply chain management for business process optimization using decision tree regression model. Supply Chain Management, 3(5),DOI: 0.48175/IJARSCT-11683
2. Kasturi, K., & Mukaesh.K., (2026). Demand Forecasting and Inventory Intelligence Platform Using Prophet and XGBoost for Retail Analytics.DOI: 10.48175/IJARSCT-34684
3. Oracle. "Classes and Objects." The Java Tutorials. https://docs.oracle.com/javase/tutorial/java/javaOO/index.html
4. Oracle. "Collections." The Java Tutorials. https://docs.oracle.com/javase/tutorial/collections/index.html
5. Oracle. "Exceptions." The Java Tutorials. https://docs.oracle.com/javase/tutorial/essential/exceptions/index.html
6. Oracle. "JDBC Basics." The Java Tutorials. https://docs.oracle.com/javase/tutorial/jdbc/basics/
7. Oracle. "Using Transactions." The Java Tutorials. https://docs.oracle.com/javase/tutorial/jdbc/basics/transactions.html
8. MySQL. "Foreign Key Constraints." MySQL 8.4 Reference Manual. https://dev.mysql.com/doc/refman/8.4/en/create-table-foreign-keys.html
9. MySQL. "START TRANSACTION, COMMIT, and ROLLBACK Statements." MySQL 8.4 Reference Manual. https://dev.mysql.com/doc/refman/8.4/en/commit.html
10. Council of Supply Chain Management Professionals. "SCM Definitions and Glossary of Terms." https://cscmp.org/CSCMP/cscmp/educate/scm_definitions_and_glossary_of_terms.aspx
11. GS1. GS1 Global Traceability Standard.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 6, November 2026
Impact Factor: 8.2

https://www.gs1.org/sites/default/files/docs/traceability/GS1_Global_Traceability_Standard_i2.pdf

12. Chopra, S. (2020). Supply Chain Management: Strategy, Planning, and Operation (7th ed.). Pearson. Publisher page
13. Silver, E. A., Pyke, D. F., & Thomas, D. J. (2017). Inventory and Production Management in Supply Chains (4th ed.). CRC Press. Publisher page
14. DeHoratius, N., & Raman, A. (2008). Inventory record inaccuracy: An empirical analysis. Management Science, 54(4), 627–641. https://doi.org/10.1287/mnsc.1070.0789