Java Console POS System Assignment

Course Information

Course: Programming Fundamentals / Java Programming
Assignment Type: Console Application Development
Programming Paradigm: Procedural Programming (No OOP)
Duration: 2-3 Weeks
Total Marks: 100

Assignment Overview

Students are required to develop a comprehensive Point of Sale (POS) system using Java console application. This assignment focuses on implementing a complete business management system using only procedural programming concepts, arrays, and fundamental programming constructs without object-oriented programming (OOP) principles.

Learning Objectives

Upon completion of this assignment, students will be able to:

1. **Apply procedural programming concepts** to solve real-world business problems
2. **Implement data management** using arrays and 2D arrays effectively
3. **Design and develop menu-driven console applications** with user-friendly interfaces
4. **Handle data validation and error checking** in business applications
5. **Create comprehensive reporting systems** for business analytics
6. **Manage data relationships** between multiple entities without using objects
7. **Implement CRUD operations** (Create, Read, Update, Delete) using procedural approaches

---

System Requirements

Database Schema

The system must manage the following entities using separate arrays:

Customer Entity

- **Customer ID** (int) - Primary Key
- **Name** (String) - Customer full name
- **Address** (String) - Customer address

- **Salary** (double) - Customer salary information

Product Entity

- **Product Code** (String) - Primary Key
- **Description** (String) - Product description
- **Unit Price** (double) - Price per unit
- **Quantity on Hand** (int) - Available stock

Order Entity

- **Order ID** (int) - Primary Key
- **Date** (String) - Order date
- **Amount** (double) - Total order amount
- **Customer ID** (int) - Foreign Key referencing Customer

Order Details Entity

- **Order ID** (int) - Foreign Key referencing Order
- **Product Code** (String) - Foreign Key referencing Product
- **Date** (String) - Transaction date
- **Quantity** (int) - Quantity ordered
- **Unit Price** (double) - Price at time of sale

Technical Constraints

Mandatory Requirements

1. **No Object-Oriented Programming** - No classes (except main class), objects, inheritance, or encapsulation
2. **Procedural Programming Only** - Use only static methods and functions
3. **Array-Based Data Storage** - All data must be stored in arrays or 2D arrays
4. **Loop-Based Operations** - Use loops for all data processing and searching
5. **Console Interface** - Text-based menu system only
6. **No External Libraries** - Use only standard Java libraries

Data Structure Requirements

- **Maximum Array Sizes:**
  - Customers: 100 records
  - Products: 100 records

- Orders: 200 records
- Order Details: 500 records

- **Data Persistence:** In-memory storage only (data resets on program restart)

Functional Requirements

1. Customer Management Module (20 Marks)

1.1 Add Customer (5 marks)

- Input validation for customer ID uniqueness
- Capture customer details (ID, name, address, salary)
- Prevent duplicate customer IDs
- Display success/error messages

1.2 View All Customers (3 marks)

- Display all customers in tabular format
- Show customer ID, name, address, and salary
- Handle empty customer list scenario

1.3 Search Customer (4 marks)

- Search by customer ID
- Display complete customer information
- Handle customer not found scenarios

1.4 Update Customer (8 marks)

- Find customer by ID
- Allow selective field updates
- Maintain data integrity
- Confirm successful updates

2. Product Management Module (20 Marks)

2.1 Add Product (5 marks)

- Unique product code validation
- Input product details (code, description, price, quantity)
- Prevent duplicate product codes
- Display confirmation messages

2.2 View All Products (3 marks)

- Tabular display of all products
- Show code, description, price, and stock levels
- Handle empty inventory scenarios

2.3 Search Product (4 marks)

- Search by product code
- Display complete product information
- Handle product not found cases

2.4 Update Product (4 marks)

- Modify product details except code
- Update description, price, and quantity
- Validate input data

2.5 Stock Management (4 marks)

- Add or subtract stock quantities
- Prevent negative stock levels
- Display current stock after updates

3. Order Processing Module (25 Marks)

3.1 Create New Order (15 marks)

- Generate unique order IDs
- Validate customer existence
- Multi-item order processing
- Real-time stock checking and updates
- Calculate order totals automatically
- Handle insufficient stock scenarios

3.2 Order Validation (5 marks)

- Verify customer ID exists
- Check product availability
- Validate quantities against stock
- Prevent invalid order creation

3.3 Receipt Generation (5 marks)

- Format and display detailed receipts
- Show customer information
- List all ordered items with quantities and prices
- Display order total and date

4. Order Management Module (15 Marks)

4.1 View All Orders (8 marks)

- Display order summary in tabular format
- Show order ID, date, customer details, and total amount
- Allow detailed order viewing

4.2 Order Details View (7 marks)

- Display complete order information
- Show itemized list of products
- Include customer and order metadata

5. Reporting Module (15 Marks)

5.1 Sales Summary Report (4 marks)

- Calculate total sales revenue
- Count total number of orders
- Compute average order value
- Display comprehensive sales statistics

5.2 Product Sales Report (4 marks)

- Show quantity sold for each product
- Calculate revenue per product
- Display only products with sales activity

5.3 Customer Purchase Report (4 marks)

- List customer purchase history
- Show total orders per customer
- Calculate total spending per customer

5.4 Low Stock Alert (3 marks)

- User-defined minimum stock level
- Identify products below threshold
- Display products requiring restocking

6. User Interface and Navigation (5 Marks)

6.1 Menu System (3 marks)

- Clear and intuitive main menu
- Proper sub-menu navigation
- Consistent interface design

6.2 Error Handling (2 marks)

- Validate user inputs
- Handle invalid menu selections
- Display appropriate error messages

Implementation Guidelines

Data Management Strategy

1. **Use separate arrays** for each entity attribute
2. **Implement counter variables** to track current array usage
3. **Create search functions** for finding records by key fields
4. **Maintain data relationships** through foreign key references

Menu Structure

```
Main Menu
├── 1. Manage Customers
│   ├── Add Customer
│   ├── View All Customers
│   ├── Search Customer
│   └── Update Customer
├── 2. Manage Products
│   ├── Add Product
│   ├── View All Products
│   ├── Search Product
│   ├── Update Product
│   └── Update Stock
├── 3. Process Order
├── 4. View Orders
├── 5. Generate Reports
│   ├── Sales Summary
│   ├── Product Sales Report
│   ├── Customer Purchase Report
│   └── Low Stock Alert
└── 6. Exit
```

Sample Data Requirements

Initialize the system with:

- **3 sample customers** with varied salary ranges
- **5 sample products** with different categories and stock levels
- Display initialization confirmation message

Evaluation Criteria

Technical Implementation (60%)

- **Correct use of arrays and loops** (15%)
- **Proper data validation and error handling** (15%)
- **Complete CRUD operations implementation** (15%)
- **Accurate calculations and business logic** (15%)

Functionality (25%)

- **All required features implemented** (10%)
- **Menu navigation and user experience** (8%)
- **Report generation accuracy** (7%)

Code Quality (15%)

- **Code organization and readability** (5%)
- **Proper variable naming and comments** (5%)
- **Efficient algorithm implementation** (5%)

Submission Requirements

Deliverables

1. **Source Code File:** `POSSystem.java`
2. **Documentation:** Assignment report with:
    - System overview and features
    - Data structure explanation
    - Sample output screenshots
    - Testing scenarios and results
    - Challenges faced and solutions

Submission Format

- **File naming:** `StudentID_POSSystem.zip`
- **Include:** Source code, documentation, and any additional files
- **Submission deadline:** [Insert specific date]

Testing Requirements

Students must demonstrate:

1. **All menu options working correctly**
2. **Data validation preventing invalid inputs**
3. **Order processing with stock updates**
4. **Report generation with accurate calculations**
5. **Error handling for edge cases**

Grading Rubric

Component	Excellent (90-100%)	Good (80-89%)	Satisfactory (70-79%)	Needs Improvement (60-69%)	Unsatisfactory (<60%)
Customer Management	All CRUD operations work flawlessly with proper validation	Most operations work with minor issues	Basic operations work with some validation missing	Limited functionality with several bugs	Major functionality missing
Product Management	Complete inventory management with stock control	Good product management with minor gaps	Basic product operations working	Limited product features	Poor implementation
Order Processing	Seamless order creation with real-time updates	Good order processing with minor issues	Basic order functionality	Limited order features	Major order system flaws
Reporting System	Comprehensive reports with accurate calculations	Good reports with minor calculation issues	Basic reports working	Limited reporting features	Poor or missing reports
Code Quality	Clean, well-documented, efficient code	Good code structure with minor improvements needed	Acceptable code quality	Code needs significant improvement	Poor code quality

---

Additional Resources

Recommended Study Materials

1. **Java Arrays and Loops** - Review multidimensional arrays and nested loops
2. **Input Validation** - Scanner class usage and error handling
3. **String Manipulation** - Formatting output and string comparisons
4. **Mathematical Operations** - Calculations and aggregations

Development Tips

1. **Start with basic menu structure** before implementing features
2. **Test each module independently** before integration
3. **Use meaningful variable names** for better code readability
4. **Implement data validation** at input points

5. **Test edge cases** like empty arrays and invalid inputs

Academic Integrity

This assignment must be completed individually. While students may discuss general programming concepts, sharing code or copying solutions will result in academic penalties. All submitted work must be original and properly attributed.

Support and Questions

For technical questions or clarification on requirements, students should:

1. **Review assignment guidelines** thoroughly
2. **Consult course materials** and lecture notes
3. **Use office hours** for instructor assistance
4. **Post general questions** on course discussion forum

Assignment Coordinator: [Instructor Name]
Contact: [Email Address]
Office Hours: [Schedule]

This assignment is designed to reinforce fundamental programming concepts while building practical software development skills. Good luck with your implementation!