COSC2081 Programming 1

A. Overview of the Assignment

This project is a group programming assignment where the main goal is to develop a Store Order Management System for a technology store with order management, admin and customer functions, and product and order information. The assignment comprises of designing the system, coding it, and preparing a report and a video demonstration.

Link for full code example: https://github.com/Autopilot7/STORE-ORDER-MANAGEMENT-SYSTEM

B. Steps to complete asignment

1. Requirements Understanding
2. Design Phase
3. Detailed Planning
4. Development Phase
5. Documentation and Reporting
6. Video Demonstration

C.1. Requirements Understanding

Project Details: You will develop a Store Order Management System for a technology store. The system should handle product management, member registrations, and order processing.

Functional Requirements: Features like member registration, product listing, order creation, admin functionalities, and data persistence must be implemented.

Technical Requirements: Input validation, data encapsulation, unique ID generation, and file-based data storage are mandatory.

OOP Design and Implementation:

- Design a class hierarchy for flexibility and maintenance ease.
- Problem Solving: Apply control statements, algorithms, data structures, etc., to solve given tasks.
- Report Writing: A 5-8 page report detailing your project.
- Class Diagram: Include a class diagram showing your system's structure.
- Video Demonstration: A short video explaining your analysis, design, implementation, and a demo of the system.

C.2. Design phase

For the "COSC2081 Programming 1" group assignment at RMIT University, which involves developing a text-based Order Management System, a detailed class diagram can be designed to represent the system's structure. Below is an example of classes and methods that might be included in such a system:

Classes and Methods:

1. **Product Class**
   - **Attributes:** `productId`, `productName`, `price`, `stockQuantity`
   - **Methods:**
     - `addProduct()`
     - `updateProduct()`
     - `deleteProduct()`
     - `getProductDetails()`
2. **Order Class**
   - **Attributes:** `orderId`, `orderDate`, `memberId`, `orderDetails`
   - **Methods:**
     - `createOrder()`
     - `cancelOrder()`
     - `updateOrder()`
     - `viewOrderDetails()`
3. **Member Class**
   - **Attributes:** `memberId`, `name`, `email`, `address`
   - **Methods:**
     - `registerMember()`
     - `updateMemberDetails()`
     - `deleteMember()`
     - `getMemberDetails()`
4. **Admin Class**
   - **Attributes:** Inherits or shares some attributes from `Member` (e.g., `adminId`, `name`, `email`)
   - **Methods:**
     - `addProduct()`
     - `removeProduct()`
     - `viewAllOrders()`
     - `updateOrderStatus()`
5. **OrderDetails Class**
   - **Attributes:** `orderId`, `productId`, `quantity`, `price`
   - **Methods:**
     - `addProductToOrder()`
     - `removeProductFromOrder()`

        - `updateQuantity()`

6. **Inventory Class**
    - **Attributes:** `products` (a list or collection of `Product` objects)
    - **Methods:**
        - `addStock()`
        - `reduceStock()`
        - `checkInventoryLevels()`

Relationships:

- **Member and Order:** A Member can place multiple Orders. This is a one-to-many relationship.
- **Order and OrderDetails:** An Order can have multiple OrderDetails (each detailing a product in the order). This is a one-to-many relationship.
- **Product and Inventory:** The Inventory class manages multiple Products. This is a one-to-many relationship.
- **Admin and Product/Order:** Admin has control over Products and can manage Orders, representing a one-to-many relationship with both classes.

Additional Notes:

- **Inheritance:** `Admin` might inherit from `Member`, indicating that an Admin is a special type of Member with additional privileges.
- **Association:** There could be associations between `Product` and `OrderDetails` to track which products are in an order.

This class diagram provides a high-level view of the system, detailing the relationships between different entities and the functionalities they are expected to perform. It serves as a blueprint for development, guiding the team on how to structure the codebase.

C.3 Detailed Planning

For step 3, "Detailed Planning," which involves task breakdown and timeline development for the COSC2081 Programming 1 group project, here's a specific example:

Task Breakdown

1. **Project Initialization**
    - Task: Set up the project repository and environment.
    - Assigned to: Team Lead
    - Estimated Time: 2 days
2. **Class Design**
    - Task: Design the classes (`Product`, `Order`, `Member`, `Admin`, `OrderDetails`, `Inventory`).
    - Assigned to: Alice and Bob (team members)
    - Estimated Time: 3 days
3. **Database/File System Setup**
    - Task: Set up the file system or database for data persistence.
    - Assigned to: Charlie
    - Estimated Time: 4 days
4. **User Interface Development**
    - Task: Develop the command-line interface for user interaction.
    - Assigned to: David
    - Estimated Time: 5 days
5. **Core Functionality Implementation**
    - Task: Implement core functionalities such as member registration, product management, order processing.
    - Assigned to: Alice (member registration), Bob (product management), Charlie (order processing)
    - Estimated Time: 7 days each
6. **Integration and Testing**
    - Task: Integrate different components and perform testing.
    - Assigned to: Entire team
    - Estimated Time: 5 days
7. **Documentation and Report Writing**
    - Task: Write the project report and document the code.
    - Assigned to: Eve
    - Estimated Time: 4 days
8. **Video Demonstration Preparation**
    - Task: Prepare and record the video demonstration.
    - Assigned to: Team Lead and David
    - Estimated Time: 3 days

Timeline Development

- **Week 1:**
    - Project Initialization
    - Start of Class Design
- **Week 2:**
    - Completion of Class Design
    - Start of Database/File System Setup and User Interface Development

- **Week 3:**
    - Continue with Database/File System Setup and User Interface Development
    - Start Core Functionality Implementation
- **Week 4-5:**
    - Core Functionality Implementation
- **Week 6:**
    - Integration and Testing
- **Week 7:**
    - Documentation and Report Writing
    - Start Video Demonstration Preparation
- **Week 8:**
    - Completion of Video Demonstration
    - Final Review and Submission

This plan ensures that each team member has a clear understanding of their responsibilities and deadlines, promoting efficient project progress and collaboration. It also allows for buffer time towards the end of the project for unexpected delays or additional refinements.

C.4. Development Phase

Certainly! For Step 4, "Development Phase," of the COSC2081 Programming 1 group assignment at RMIT University, a comprehensive guide would encompass several key aspects: setting up the development environment, implementing functionalities, testing, and version control. Here's a detailed guide:

Development Phase Guide for COSC2081 Programming 1

1. Setting Up the Development Environment

- **Choose an IDE:** Select a Java Integrated Development Environment (IDE) like IntelliJ IDEA or Eclipse.
- **Version Control:** Set up a Git repository (e.g., on GitHub) for collaborative development and version tracking.
- **Coding Standards:** Agree on coding standards and naming conventions for consistency across the team.

2. Implementation of Core Functionalities

- **Divide and Conquer:** Based on the task breakdown, each team member should start working on their assigned parts.
- **Example Tasks:**
    - Implement the `Member` class with registration functionality.
    - Develop the `Product` and `ProductManager` classes for product management.
    - Create the `Order` and `OrderManager` classes for handling orders.
    - Write the `Admin` class with extended privileges for managing the system.
- **Coding Practices:** Focus on writing clean, readable, and efficient code. Regularly commit changes to the repository with meaningful commit messages.

3. Continuous Integration and Testing

- **Unit Testing:** Write unit tests for each class using JUnit. Ensure that each method is tested for both expected and edge-case scenarios.
- **Test as You Go:** Regularly test the application as new features are implemented to catch bugs early.

4. Integration

- **Combine Components:** Gradually integrate the separately developed components (e.g., integrating `ProductManager` with `OrderManager`).
- **Integration Testing:** Test the system as a whole to ensure that integrated components work together seamlessly.

5. Debugging and Refinement

- **Debugging:** Use debugging tools in your IDE to identify and fix any issues.
- **Code Review:** Conduct code reviews with team members to ensure code quality and adherence to project requirements.

6. Final Testing and Validation

- **User Acceptance Testing (UAT):** Perform final testing that simulates real-world usage of the system.
- **Validation Against Requirements:** Ensure that the final product meets all the specified requirements of the assignment.

7. Preparing for Submission

- **Finalize Code:** Ensure that the final version of the code is well-commented, formatted, and adheres to the project guidelines.
- **Documentation:** Prepare necessary documentation that accompanies the code, explaining the structure and functionalities.
- **Submission Check:** Perform a final check before submission to ensure that all required elements are included and functioning.

Additional Tips

- **Regular Meetings:** Hold regular meetings to track progress, discuss challenges, and plan next steps.
- **Time Management:** Adhere to the timeline developed in the planning phase to ensure timely completion.

This guide should help structure the development process, ensuring a systematic and organized approach to building the Order Management System. Remember, effective communication and collaboration within the team are key to a successful project.

```
#Outline example for coding

#Member Registration

import java.util.HashMap;

public class Member {
    private String memberId;
    private String name;
    private String email;

    public Member(String memberId, String name, String email) {
        this.memberId = memberId;
        this.name = name;
        this.email = email;
    }

    // Getters and Setters
    // ...
}

public class MemberManager {
    private HashMap<String, Member> members = new HashMap<>();

    public boolean registerMember(String memberId, String name, String email) {
        if (!members.containsKey(memberId)){
            members.put(memberId, new Member(memberId, name, email));
            return true;
        }
        return false;
    }

    // Additional methods like updateMemberInfo, deleteMember can be added
}

#Product management

public class Product {
    private String productId;
    private String productName;
    private double price;

    public Product(String productId, String productName, double price) {
        this.productId = productId;
        this.productName = productName;
        this.price = price;
    }

    // Getters and Setters
    // ...
}

public class ProductManager {
    private HashMap<String, Product> products = new HashMap<>();

    public boolean addProduct(String productId, String productName, double price) {
        if (!products.containsKey(productId)) {
            products.put(productId, new Product(productId, productName, price));
            return true;
        }
        return false;
    }

    // Additional methods for updating and removing products
}


import java.util.ArrayList;
import java.util.List;

public class Order {
    private String orderId;
    private List<Product> products;
    private String memberId;

    public Order(String orderId, String memberId) {
        this.orderId = orderId;
        this.products = new ArrayList<>();
        this.memberId = memberId;
    }

    public void addProduct(Product product) {
        products.add(product);
    }

    // Getters and Setters
    // ...
}

#Order Processing
public class OrderManager {
    private HashMap<String, Order> orders = new HashMap<>();

    public boolean createOrder(String orderId, String memberId) {
        if (!orders.containsKey(orderId)) {
            orders.put(orderId, new Order(orderId, memberId));
            return true;
        }
        return false;
    }

    // Methods for updating, canceling, and viewing orders
}
```

```
#Admin function
public class Admin extends Member {
    public Admin(String memberId, String name, String email) {
        super(memberId, name, email);
    }

    // Admin-specific functionalities like reviewing all orders or members
}


#main application
public class OrderManagementSystem {
    public static void main(String[] args) {
        // This would be the entry point of your application
        // Here you can initialize your managers and start the application loop
    }
}


```

C.5. Documentation and Reporting

For step 5, "Documentation and Reporting," of the COSC2081 Programming 1 group assignment, I'll provide a guide for both the project report and the video demonstration.

Project Report Writing Guide

1. **Introduction:**
    - Summarize the project's goal and the main functionalities implemented.
    - Example: "This project implements a text-based Order Management System for a technology store, focusing on member registration, product management, and order processing."
2. **Project Description:**
    - Detail the technical aspects, such as the classes used, their interactions, and key functionalities.
    - Include diagrams or code snippets where relevant.
    - Example: "The system consists of several core classes: `Member`, `Product`, `Order`, etc., each handling specific functionalities. The `ProductManager` class, for instance, manages product additions, updates, and removals."
3. **Implementation Details:**
    - Explain how you implemented various features.
    - Discuss challenges faced and how they were resolved.
    - Example: "Implementing the file-based persistence posed a challenge, which was resolved by using Java's `FileWriter` and `BufferedReader` classes for data storage and retrieval."
4. **Project Planning:**
    - Outline how the team was organized, the roles and responsibilities, and how tasks were distributed.
    - Example: "The team was divided into three subgroups, each responsible for a specific set of functionalities, such as member management, product management, and order processing."
5. **Conclusion:**
    - Summarize the project outcomes and learning experiences.
    - Suggest potential improvements or future work.
    - Example: "The project successfully implements the basic functionalities of an Order Management System. Future work could include adding a graphical user interface and integrating a database for data management."
6. **Appendices and References:**
    - Include any additional diagrams, code snippets, and references to external sources.

Video Demonstration Guide

1. **Planning the Content:**
    - Outline the key points to cover: introduction, demonstration of features, and conclusion.
    - Example: Start with an overview of the project, demonstrate each core functionality (e.g., how to register a new member, add a product, create an order), and conclude with a brief discussion on the project's learning outcomes.
2. **Scripting the Demonstration:**
    - Prepare a script or bullet points to ensure all important aspects are covered clearly.
    - Example: Write down the key steps and expected outcomes for each functionality you will demonstrate.
3. **Recording the Video:**
    - Use screen recording software to capture the system in action.
    - Narrate or use subtitles to explain what's happening on the screen.
    - Example: While showing the member registration feature, explain the process and the underlying logic.
4. **Editing:**
    - Edit the video for clarity, pacing, and to add any necessary annotations or highlights.
    - Keep the video concise and focused on the key functionalities.
5. **Conclusion:**
    - End with a summary of what was demonstrated and the project's significance.
    - Example: "This concludes our demonstration of the Order Management System, highlighting its key features and our team's collaborative effort."
6. **Uploading and Sharing:**

    - Upload the video to a platform like YouTube if required.
    - Include the link in your project documentation.

Remember, the video should complement your written report, providing a practical demonstration of your system, while the report offers the detailed technical and theoretical background.