RMIT

RMIT International University Vietnam

Assignment Cover Page

Subject Code:	COSC2081
Subject Name:	Programming 1
Location & Campus (SGS or HN) where you study:	SGS
Title of Assignment:	Group Assignment
Teachers Name:	Minh Vu Thanh
Group Name:	WAO
Group Members (names and id numbers):	S3974735 – Tran Manh Cuong S3965528 – Truong Quang Bao Loc S3979259 – Truong Tuong Hao
Assignment due date:	24/09/2023
Date of Submission:	23/09/2023
Declaration of Authorship	We declare that in submitting all work for this assessment, we have read, understood and agreed to the content and expectations of Assessment Declaration as specified in https://www.rmit.edu.vn/students/my-studies/assessment-and-exams/assessment
Consent to Use:	We give RMIT University permission to use our work as an exemplar and for showcase/exhibition display.

RMIT

Table of Contents

RMIT

1. Introduction

1.1. Definition

A collective group effort called the Container Port Management System proposed to create Java-based software that utilizes object-oriented programming (OOP) methods. Inside the port management system, the project is essentially connected with modeling and managing the operations of container ports, vehicles, containers, and lastly the user roles. An overview of the project's scope, goals, and group roles has been provided in this paper.

1.2. Role

As there are 3 members in our team, it is a bit more challenging for the project. Cuong is the leader of the team and oversees analyzing all criteria and aspects of the project so that he can build up a code template including classes, interfaces, and packages in the terminal of IntelliJ. He also develops most of the functions in the system as well as debugging any error that we encountered in the progress. Otherwise, Loc is responsible for developing vehicle functions as well as testing each function at the end to find errors in the code and supporting Cuong to fix them. Hao is accountable for developing container functions as well as designing the slides and demonstration scripts. We also share our chores to complete the UML diagram and the report.

1.3. Scope & Objectives

This project aims to simulate the management and movement of containers in ports that take place in a digital environment, replacing them with paper-based methods. The system is responsible for managing information that is relevant to vehicles, ports, containers, and port managers. Furthermore, there will be constraints that are required to be distinguished which are 2 different user roles, the system admin can access all data and functions, while port managers are limited to operate only on the port they oversee. It also requires having a proper database structure and efficiently savable all changes. All objectives must be completed such as implementation, history tracking for all databases, functional features as well and text-based interface.

2. Project Description

The Container Port Management System project is a challenging, comprehensive software that places a heavy focus on technical elements of software development. The project's architectural design, the usage of Object-Oriented Programming (OOP) concepts, data management by creating those databases in the model package, user authentication, and system scalability will all be covered in detail in this part as we dig into the technical aspects of the project. This conversation will shed light on the project's organizational structure and its use of technology to accomplish its objectives.

2.1. Architectural Design

The focus of the Container Port Management System is its architectural design, which lays the foundation for the entire software. The architecture of this system is based on a modular approach, with a clear separation of concerns, we implemented the CRUD approach to complete the progress. This statement also means that each function of the

system can function smoothly and later we can also upgrade or modify it most conveniently by taking advantage of the CRUD method.

2.2. Object-Oriented Programming (OOP) Principles

The project is heavily based on OOP regulations to build a flexible and maintainable codebase. OOP is defended as the modeling of real-world entities or objects with attributes and behaviors. This approach created a proper design of a class hierarchy that emphasizes the various entities within the system, such as ports, vehicles, containers, users, etc. Each class is designed to encapsulate data and functions that are related to a general entity, modularity, and code reusability.

2.3. Data Handling

Efficient data handling is a critical aspect of the project, given the large volume of information that the system must manage. To achieve this, we created the package name "model" that contains several ".dat" files to manage all the databases that we have created before. Manipulating with the lecture and tutorial about Java Streams and Files (I/O), we were able to be familiar with Object Output Stream and Object Output Stream that we can write and read all the Object attributes and formatted by toString() to the ".dat" files to saved it after functioning with functions.

2.4. User Authentication

This project also implemented user authentication as username and password that we usually used as 'admin' and 'port manager' to confirm access and allow permissions relying on user roles. So, there are two primary user roles in our system: The system administrator and port manager. The authentication process verifies the credentials provided by the username and password that I mentioned above to corresponding functionalities based on its user's role. Role-Based Access Control (RBAC) was implemented to define and manage each user's roles and their associated permissions. The system admin has access to all the functions that we created, while port managers are limited to interacting with fewer functions that are only designed for designated their port.

2.5. System Scalability

Scalability is an important deliberation in the prototype of the Container Port Management System. The architecture and data structures are built to provide potential growth and enlarge the complexity. Several aspects that contributed to the system's scalability:

- **Modular Design**: The modular design allows for the easy recycling of new features or entities. For instance, we designed the System with abstract classes that outlet several child classes such as "Container, User, Vehicle" so that it lets us inherit methods and fields that have been used before in the parent class.
- **Efficient Algorithms**: The effectiveness and scalability of the algorithms employed for data processing and computations are factors in their selection. This guarantees that the system will be able to process an increasing amount of data and carry out complicated calculations without suffering a severe performance hit.

2.6. Text-Based User Interface

The system's user interface is completely a text-based system out in the code to simplify the implementation of the application. Despite the lack of graphical elements, we formatted a good-looking text space for users to interact with, this interacting interface offers a command line that requires users to input from 1 to 8 to use functions as well as calculate formulas (press 0 to move backward or exit the system). Users will navigate through the system by using the num key as mentioned above, after pressing the num key user will be navigated to a new terminal that contains further interaction methods, and they can also press 0 to return one step as before.

2.7. Testing & Debugging

The project has a tight coherence between testing and debugging practices to ensure the reliability and accuracy of the application. Unit testing is manipulated on individual components, but when it comes to interacting with databases, it is extremely complex and requires us to initialize the database multiple times since it conflicts with bugs.

2.8. Documentation

Documentation plays an essential role in our project, aiding in both the development and future maintenance process. The following documentation aspects are critical:

- **Code Comments**: The codebase is well-documented with comments to explain the purpose and functionality of classes, methods, and variables. For example, we must comment on all the names of functions, we even jot out the definition for each step of that function in the development process.
- **User Guidance**: we printed out to the interacting interface numerous guides such as "enter the username, enter the password" and attached examples for object IDs like "SH0001".
- **Class Diagram**: The diagram for Container Port Management System is quite complicated and it took days to complete, for the sake of high resolution, it must be defined with relationships such as Dependency, Aggregation, Composition, etc.

RMIT

3. Implementation Details

3.1. Class Diagram Design

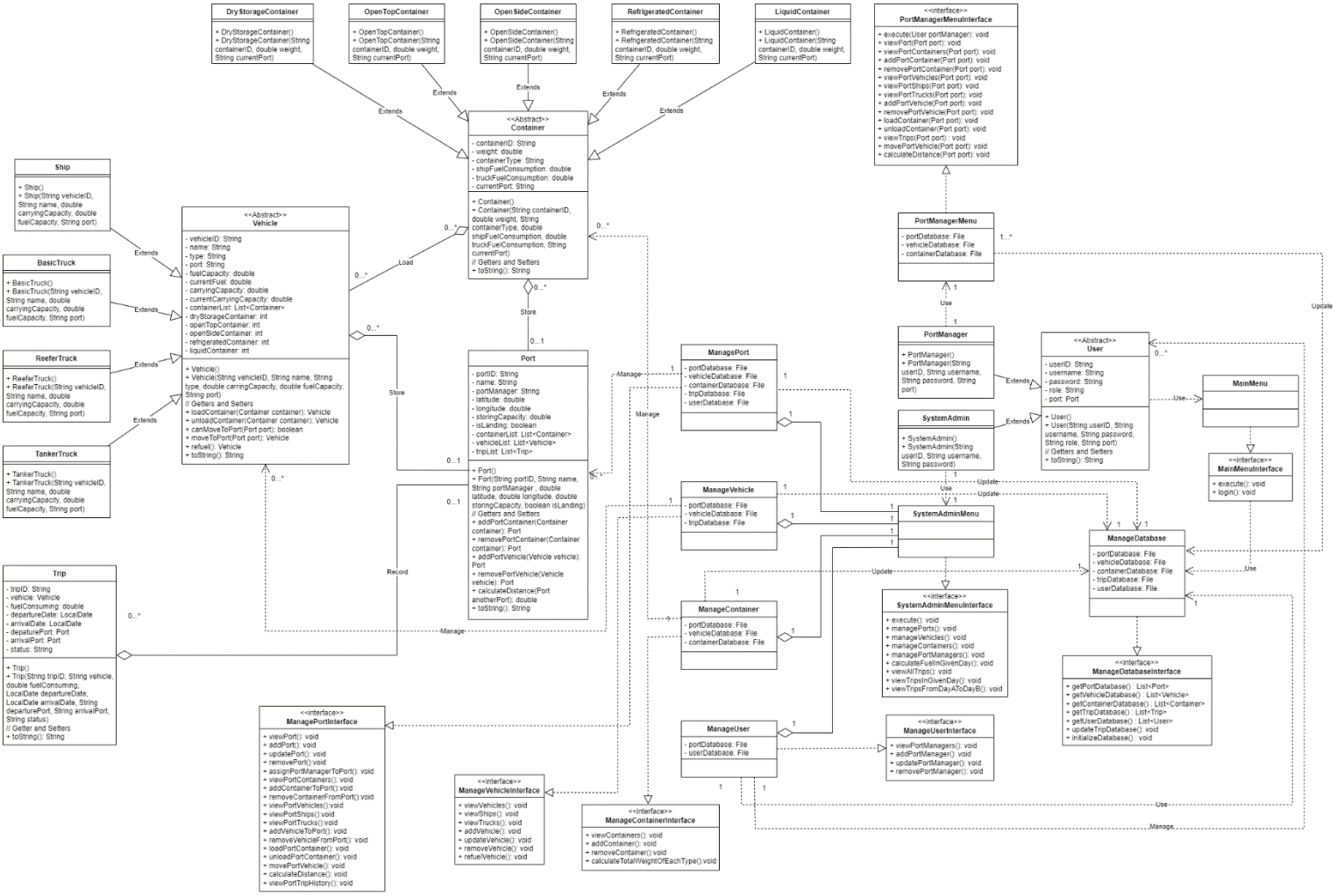

Figure 1: Class Diagram

We spent 3 days designing the class diagram for the system and we also modified this diagram during our development process. Overall, our class diagram has 23 classes and 8 interfaces which were designed based on 5 main classes: Port, Vehicle, Container, User, and Trip. Every class has its own attributes, including a unique ID, name, and other attributes related to that object. Moreover, it also has its own constructor, getters, setters, toString(), and other methods to manage that object as the Port class has some methods like addPortContainer(), addPortVehicle(), the Vehicle class has some methods like loadContainer(), unloadContainer(), etc. We also show the inheritance between the Vehicle class with 4 types of vehicles, the Container class with 5 types of containers, and the User class with 2 types of users. After finishing defining all the main objects, we started to design the menu for users. Our system has 2 types of users: system admin and port manager. We created 2 different menus for the admin and port manager, and those menus will interact with the ManageDatabase, ManagePort, ManageVehicle, ManageContainer, and

ManageUser classes. All users will have to access the MainMenu to log in to their account and process the system functions based on their role.

3.2. System Design

In our Port Management System, we defined 8 interfaces, 25 classes, and 5 database files and divided those into 7 packages which are: port, vehicle, container, user, menu, manage, and model. Here are the details of each package:

- The port package includes “Port” and “Trip” classes.
- The vehicle package includes “Vehicle”, “Ship”, “BasicTruck”, “ReeferTruck”, and “TankerTruck” classes.
- The container package includes “Container”, “DryStorageContainer”, “LiquidContainer”, “OpenSideContainer”, “OpenTopContainer”, and “RefrigeratedContainer” classes.
- The user package includes “User”, “SystemAdmin”, and “PortManager” classes.
- The menu package includes “MainMenu”, “SystemAdminMenu”, and “PortManagerMenu” interfaces and classes.
- The manage package includes “ManageDatabase”, “ManagePort”, “ManageVehicle”, “ManageContainer”, and “ManageUser” interfaces and classes.
- The model package includes the database files of ports, vehicles, containers, users, and trips.

3.3. System Features Details

Before using the system features, users will have to access the Main Menu to log in to their account. Users can go back to the previous menu or exit the system by entering number 0.

*** Main Menu**

- **Login**: Users must enter the username and password which exist in the database. If they enter incorrectly, the system will send an error message and ask them to enter again.

*** System Admin Menu:**

- **View All Ports**: The system displays all the ports in the database.
- **Add New Port**: Admin can add a new port to the database by entering the port’s name, latitude, longitude, storing capacity, and assign a port manager to that port. The system will automatically generate a unique ID for that port.
- **Update Port’s Information**: Admin can update a port’s name, latitude, longitude, and port manager.
- **Remove Port**: Admin can remove a port from the ID by entering that port’s ID.
- **Assign a Port Manager to a Port**: Admin can assign a port manager to a port by entering the port’s ID and port manager’s ID. The port manager’s port and the port’s manager will be updated.

- **View All Containers in a Port**: Admin can view all containers in a specific port by entering the port's ID.
- **Add a Container to a Port**: Admin can add a container to a port by entering the port's ID and container's ID. The container's current port and port's container list will be updated.
- **Remove a Container from a Port**: Admin can remove a container from a port by entering the port's ID and container's ID. The container's current port and port's container list will be updated.
- **View All Vehicles in a Port**: Admin can view all vehicles in a specific port by entering the port's ID.
- **View All Ships in a Port**: Admin can view all ships in a specific port by entering the port's ID.
- **View All Trucks in a Port**: Admin can view all trucks in a specific port by entering the port's ID.
- **Add a Vehicle to a Port**: Admin can add a vehicle to a port by entering the port's ID and vehicle's ID. The vehicle's current port and port's vehicle list will be updated.
- **Remove a Vehicle from a Port**: Admin can remove a vehicle from a port by entering the port's ID and vehicle's ID. The vehicle's current port and port's vehicle list will be updated.
- **Load a Container to a Vehicle in a Port**: Admin can load a container to a vehicle in a port by entering the port's ID, vehicle's ID, and container's ID. The vehicle's container list will be updated.
- **Unload a Container from a Vehicle in a Port**: Admin can unload a container from a vehicle in a port by entering the port's ID, vehicle's ID, and container's ID. The vehicle's container list will be updated.
- **Move a Vehicle from a Port to another Port**: Admin can move a vehicle from a port to another port by entering the departure port's ID, vehicle's ID, arrival port's ID, departure date, and arrival date. The system will automatically create a trip with a unique ID. The 2 ports' vehicle list, container list, trip list, the vehicle and container's current port, and the trip database will be updated.
- **View a Port's Trip History**: Admin can view a port's trip history by entering the port's ID.
- **View All Vehicles**: The system displays all vehicles in the database.
- **View All Ships**: The system displays all ships in the database.
- **View All Trucks**: The system displays all trucks in the database.
- **Add New Vehicles**: Admin can add a new vehicle to the database by entering the vehicle's type, name, fuel capacity, carrying capacity, and add that vehicle to a port. The system will automatically generate a unique ID for that vehicle.
- **Update Vehicle's Information**: Admin can update a vehicle's name.

- **Remove Vehicle**: Admin can remove a vehicle from the database by entering that vehicle's ID.
- **Refuel a Vehicle**: Admin can refuel a vehicle by entering that vehicle's ID and that vehicle's current fuel will be updated to be equal to the vehicle's fuel capacity.
- **View All Containers**: The system displays all containers in the database.
- **Add New Container**: Admin can add a new container to the database by entering the container's type, name, weight, and add that vehicle to a port. The system will automatically generate a unique ID for that container.
- **Remove Container**: Admin can remove a container from the database by entering that container's ID.
- **Calculate Total Weight of Each Type of All Containers**: Admin can view the total weight of a specific container type by entering that type's name.
- **View All Port Managers**: The system displays all port managers in the database.
- **Add New Port Manager**: Admin can add a new port manager to the database by entering the port manager's username, password, and assign that port manager to a port. The system will automatically generate a unique ID for that port manager.
- **Update Port Manager's Information**: Admin can update a port manager's username and password.
- **Remove Port Manager**: Admin can remove a port manager from the database by entering that port manager's ID.
- **Calculate How Much Fuel Has Been Use in a Given Day**: Admin can view the total consuming fuel in a given day by entering that day.
- **View All Trips**: The system displays trips in the database.
- **View All Trips in a Given Day**: Admin can view all trips in a given day by entering that day.
- **View All Trips from Day A to Day B**: Admin can view all trips from day A to day B by entering 2 different days.

*** Port Manager Menu**

- **View Port's Information**: Port Manager can view his/her port's information.
- **View All Containers**: Port Manager can view his/her port's container list.
- **Add a Container to Port**: Port Manager can add a container to his/her port by entering an available container's ID. The container's current port and port's container list will be updated.
- **Remove a Container from Port**: Port Manager can remove a container from his/her port by entering that container's ID. The container's current port and port's container list will be updated.
- **View All Vehicles**: Port Manager can view his/her port's vehicle list.
- **View All Ships**: Port Manager can view his/her port's ship list.
- **View All Trucks**: Port Manager can view his/her port's truck list.

- **Add a Vehicle to Port**: Port Manager can add a vehicle to his/her port by entering an available vehicle's ID. The vehicle's current port and port's vehicle list will be updated.
- **Remove a Vehicle from Port**: Port Manager can remove a vehicle from his/her port by entering that vehicle's ID. The vehicle's current port and port's vehicle list will be updated.
- **Load a Container to a Vehicle**: Port Manager can load a container to a vehicle by entering the vehicle's ID and container's ID. The vehicle's container list will be updated.
- **Unload a Container from a Vehicle**: Port Manager can unload a container from a vehicle by entering the vehicle's ID and container's ID. The vehicle's container list will be updated.
- **Move a Vehicle to Another Port**: Port Manager can move a vehicle from his/her port to another port by entering vehicle's id, arrival port's ID, departure date, and arrival date. The system will automatically create a trip with a unique ID. The 2 ports' vehicle list, container list, trip list, the vehicle and container's current port, and the trip database will be updated.
- **Calculate The Distance Between 2 Ports**: Port Manager can calculate the distance between his/her port to another port by entering another port's ID.
- **View Port's Trip History**: Port Manager can view his/her port's trip history.

4. Project Planning Report

*** Project Timeline**

Time	Task
04/09 – 07/09	Analyze the project requirement, design the UML diagram, identify all the system's functions, and assign tasks for team members.
08/09 – 15/09	Develop the system's function and create the database.
16/09 – 17/09	Improve the code by creating separate interfaces for specific aspects of the system and adding exception messages if users enter invalid inputs.
18/09 – 21/09	Test all the functions and debug the errors.
22/09 – 23/09	Record the demonstration video and write the report

*** Individual Contribution:**

All Team Members	Role and Task Given	Individual Contributions (%)
1. Tran Manh Cuong	Leader. Develop most of the functions of the system, test the system, debug the errors, analyze the project requirements, assign tasks for members, draw the class diagram, record the demonstration video, write, and format the report.	40

2. Truong Quang Bao Loc	Team Member. Develop the vehicle's functions, test the system, debug the errors, draw the class diagram, record the demonstration video, and write the report.	30
3. Truong Tuong Hao	Team Member. Develop the container's functions, draw the class diagram, design the demonstration slide, record the demonstration video, and write the report.	30

5. Conclusion

The container port management system project is a great opportunity for our team to learn and practice Java programming and object-oriented programming. We have learned how to design and develop the system with the class diagram and the OOP principles including encapsulation, inheritance, polymorphism, and abstraction. Although we still had to face some challenges during the development process, the lecture and tutorial sessions helped us a lot to solve our problems. It included most of the knowledge and example codes to handle this assignment such as database handling, OOP principles, and Java libraries. Moreover, we also have a chance to improve our research, teamwork, and problem-solving skills and learn how to manage the project workflow and timeline, which are very important skills for our future careers. After this assessment, we will continue to improve this project by developing a GUI for the system and extending the project scope by adding more features.