Object Orientated Programming
CCS1303

Command-Line Application

Group No: 30

Student Name	Registration Number
S.M. Rajapakse	CIT-25-01-0374
S.S. Matharage	CIT-25-01-0533
A.J.M.K.S. Abeykoon	CIT-25-01-0714
G.S. Jayalath	CIT-25-01-0328
S.M.S. Thathsarani	CIT-25-01-0281
S.D.D. Dahanayke	CIT-25-01-0104

Submission Deadline: 16-02-2026

Table of Contents

1.Abstract

This Project represents practical application of Object-Oriented Programming (OOP) concepts through a command- Line based Vehicle Rental Management System using Java Programming Language. This System Illustrates the OOP concepts. Such as, Encapsulation, Inheritance, Overriding, Polymorphism and abstraction. Users of this application enables to efficiently manage different types of vehicles (Car, Bike and Van) and perform rental operations. For examples, users can rent and return the vehicles, view all vehicles and total rental income via this menu driven Command-Line Interface (CLI). Moreover, users can search specific vehicles by the vehicle ID. This project demonstrates how OOP principles work collaboratively to create a maintainable, robust software solution that address real-world business requirements.

2.Introduction

2.1. Background of the problem

These days, there are many problems in Vehicle Rental Industry. For examples, Traditional manual systems are time-consuming, prone to errors, and lacks scalability. For that, it requires an efficient management system to handle these problems.

This Vehicle Rental Management System serves as a practical solution helps to handle distinct vehicle types, track availability, calculate the rental costs.

2.2. Objectives of the assignment

Objectives of this assignment are:

- Use abstract classes and design the class hierarchy using inheritance.
- Execute runtime polymorphism and create a menu-driven application.
- Apply input validation and exception handling.
- Use encapsulation to protect data in the system.

2.3. Scope of the system

- Search the vehicle and track the availability of the vehicle using vehicle ID.
- Manage the different vehicle types (Cars, Bikes and Vans).
- Calculate rental cost and total rental income based on vehicle type and days.
- Functions through command-line interface.

3.System Design and Architecture

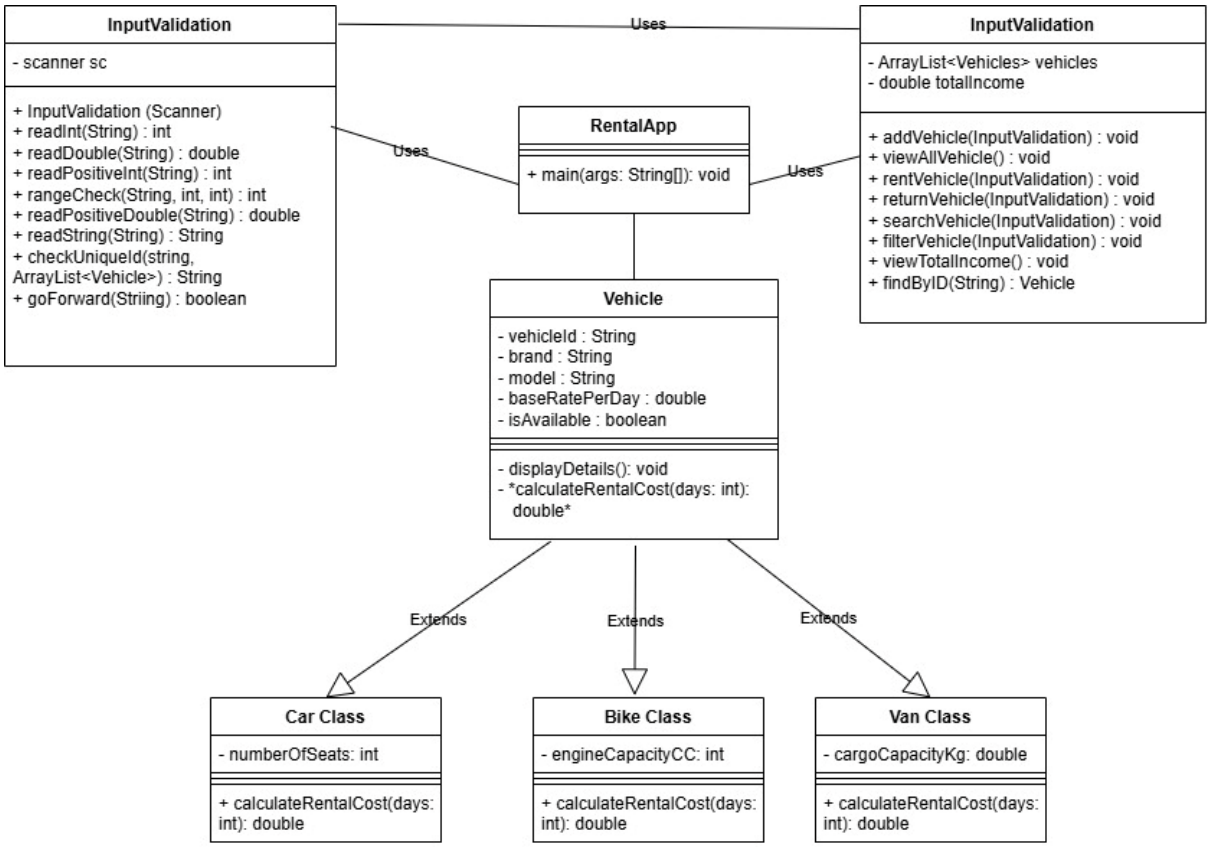
3.1. Overall system flow (menu-driven logic)

This Vehicle Rental Management System runs through a menu-driven structure. Users of this system can select the options until choose exit. Also, vehicles are stored in an array using ArrayList function. Before starting the process, system checks whether the inputs are validated.

3.2. Class hierarchy explanation

- Vehicle (Abstract Class) – Common attribute and methods.
- Car, Bike, Van (Subclasses) – Extend Vehicle and override calculateRentalCost().
- Rental App (Main class)- Controls menu and operations.
- Vehicle Services – Contains the main mechanisms of the System.
- Input Validations – Contain all input validation processes.

3.3. UML Class Diagram



3.4. Justification for using abstraction, inheritance, and polymorphism

Abstraction

- It includes the main functions (displayDeyails () and calculateRentalCost(days)) and attributes (VehicleId, brand, model, baseRatePerDay, isAvailable).

Inheritance

- Reuse the common properties (attributes and methods) for Subclasses (Car, Bike and Van).

Polymorphism

- Methods (calculteRentalCost) work differently for each subclass.

Encapsulation

- Protect the private attributes from unauthorized access.

4.Implementation Details

4.1. Description of each major class and its responsibility

Major class	Responsibility
Vehicle	It contains the common attributes and methods and defines abstract method calculateRentalCost ().
Car	Adds numberOfSeats and calculates own rental cost.
Bike	Adds engineCapacityCC and calculates own rental cost.
Van	Adds cargoCapacityKg and implements own rental costs.
RentalApp	Manages user inputs, Controls menu and tracks the rental income.
Vehicle Services	Manages all the processes happening in the system.
Input Validation	Handles all the inputs and manages exceptions using try catch.

4.2. Explanation of overridden methods and runtime polymorphism

This system uses the **Vehicle** reference to call the rental cost method and it allows the system to calculate the correct rental cost of the selected vehicle type. All the vehicle types are stored in ArrayList. However, using runtime polymorphism programme can automatically select which method to run when calculating rental cost.

4.3. Important code snippets with explanation

Using these methods, we can handle unexpected errors while reading integers, doubles and strings. These methods mainly use try catch to handle errors gracefully. (These methods are in the Input Validation Class.)

```
12      // Reads integers with try catch
13      public int readInt(String text) {  2 usages
14          while (true){
15              System.out.print(text);
16              try {
17                  int value = sc.nextInt();
18                  sc.nextLine();
19                  return value;
20              } catch (InputMismatchException e){
21                  System.out.println("Invalid input please try a whole number");
22                  sc.nextLine();
23              }
24          }
25      }
```

```java
// Reads doubles with try catch
public double readDouble(String text) {  1 usage
    while (true) {
        System.out.print(text);
        try {
            double value = sc.nextDouble();
            sc.nextLine();
            return value;
        } catch (InputMismatchException e) {
            System.out.println("Invalid input! Please try a decimal number.");
            sc.nextLine();
        }
    }
}
```

```java
// Error handling when getting strings
public String readString(String text) {  8 usages
    while (true) {
        System.out.print(text);
        try {
            String value = sc.nextLine();
            if (!value.isEmpty()) {
                return value;
            }
            System.out.println("Input cannot be empty.");
        } catch (Exception e) {
            System.out.println("Error please try again.");
        }
    }
}
```

These codes show how we override calculateRentalCost method in Car, Van, Bike Sub Classes.

```java
@Override  1 usage
public double calculateRentalCost(int days){
    totalCost = getBaseRatePerDay()*days+(engineCapacityCC*0.5*days);
    return totalCost;
}
```

```java
@Override  1 usage
public double calculateRentalCost(int days){
    totalCost = getBaseRatePerDay()*days+(cargoCapacity*0.2*days);
    return totalCost;
}
```

```
10      @Override  1 usage
        public double calculateRentalCost(int days){
12          totalCost = getBaseRatePerDay()*days+(numberOfSeats*200*days);
13          return totalCost;
14
15      }
```

5.Results and Demonstration

5.1. Adding vehicles

```
----------------- Menu -----------------
1. Add a Vehicle
2. View All Vehicles
3. Rent a Vehicle
4. Return a Vehicle
5. Search Vehicle by ID
6. Filter Vehicles
7. View Total Rental Income
8. Exit
========================================
Enter your choice: 1

--------Select Vehicle Type--------
1. Car
2. Van
3. Bike
Select option (1-3) : 1

------Add Vehicle------
vehicle ID: ABC1234
Vehicle brand : Toyota
Vehicle model : Camry
Rent Price/Day : 1500
Number of seats: 5
Vehicle added successfully!

Add another vehicle(Y/N)
N

----------------- Menu -----------------
1. Add a Vehicle
2. View All Vehicles
3. Rent a Vehicle
4. Return a Vehicle
5. Search Vehicle by ID
```

5.2. Renting a vehicle

```
---------------- Menu ----------------
1. Add a Vehicle
2. View All Vehicles
3. Rent a Vehicle
4. Return a Vehicle
5. Search Vehicle by ID
6. Filter Vehicles
7. View Total Rental Income
8. Exit
========================================
Enter your choice: 3

----- Vehicle Renting -----
Enter Vehicle ID: ABC1234
Vehicle is available.
Number of days: 5
Your rental cost: 12500.0
Vehicle rented successfully

Rent another vehicle.(Y/N)
N

---------------- Menu ----------------
1. Add a Vehicle
2. View All Vehicles
3. Rent a Vehicle
4. Return a Vehicle
5. Search Vehicle by ID
6. Filter Vehicles
7. View Total Rental Income
8. Exit
========================================
Enter your choice:
```

5.3. Returning a vehicle

```
---------------- Menu ----------------
1. Add a Vehicle
2. View All Vehicles
3. Rent a Vehicle
4. Return a Vehicle
5. Search Vehicle by ID
6. Filter Vehicles
7. View Total Rental Income
8. Exit
========================================
Enter your choice: 4
Enter vehicle ID: ABC1234
Vehicle (ABC1234) Returned Successfully

Return another vehicle(Y/N)
N
```

5.4. Searching vehicles

```
---------------- Menu ----------------
1. Add a Vehicle
2. View All Vehicles
3. Rent a Vehicle
4. Return a Vehicle
5. Search Vehicle by ID
6. Filter Vehicles
7. View Total Rental Income
8. Exit
========================================
Enter your choice: 5

----- Searching Vehicle -----
Enter Vehicle ID: ABC1234

---- Details ----
Vehicle ID : ABC1234
Brand : Toyota
Model : Camry
Rate Per Day : 1500.0
Availability : Available
```

5.5. Viewing total income

```
----------------- Menu -----------------
1. Add a Vehicle
2. View All Vehicles
3. Rent a Vehicle
4. Return a Vehicle
5. Search Vehicle by ID
6. Filter Vehicles
7. View Total Rental Income
8. Exit
========================================
Enter your choice: 7

----- Total Income -----
Total income is: Rs12500.0
```

6.Testing and Validation

6.1. Testing strategy

Different types of inputs, including valid values and invalid values used to test this system and checks how the program behaves in different situations.

- Unit testing for rental cost.
- Validation testing for user input.
- Functional testing for menu options.
- Boundary testing for edge cases.

6.2. Test case table

The following table shows the complete test cases covering all system functionalities.

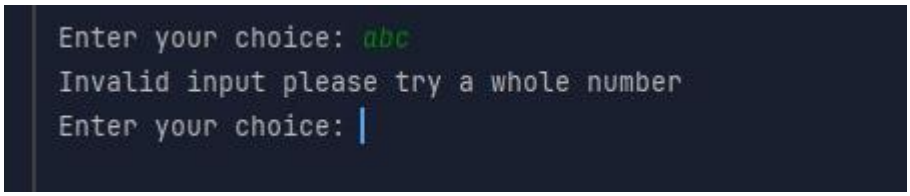
Test ID	Input	Expected Output	Actual Output	Status
TC-01	Add Car: Type: 1 ID: C001 Brand: Toyota Model: Camry Rate: 5000 Seats: 5	Vehicle added successfully!	Vehicle added successfully!	Pass
TC-02	Add Van: Type: 2 ID: V001 Brand: Ford Model: Transit Rate: 7000 Cargo: 1500kg	Vehicle added successfully!	Vehicle added successfully!	Pass
TC-03	Add Bike: Type: 3 ID: B001 Brand: Honda Model: CBR Rate: 2000 Engine: 150cc	Vehicle added successfully!	Vehicle added successfully!	Pass
TC-04	Duplicate ID: Add vehicle with existing ID 'C001'	Error: 'Vehicle ID: C001 already exists. Try a different ID.'	'Vehicle ID: C001 already exists. Try a different ID.'	Pass
TC-05	View All Vehicles (after adding 3 vehicles)	Display all 3 vehicles with correct details All show 'Availability: Available'	Display all 3 vehicles with correct details All show 'Availability: Available'	Pass

TC-06	Rent Car C001 for 2 days	Rental Cost: Rs 12,000 Formula: (5000× 2) + (5×200×2) Status changes to 'Not Available'	Your Rental Cost: Rs 12,000 Vehicle rented successfully	Pass
TC-07	Rent Van V001 for 3 days	Rental Cost: Rs 21,900 Formula: (7000×3)+(1500×0.2×3) Status changes to 'Not Available'	Your Rental Cost: Rs 21,900 Vehicle rented successfully	Pass
TC-08	Rent Bike B001 for 1 day	Rental Cost: Rs 2,075 Formula: (2000×1)+(150×0.5×1) Status changes to 'Not Available'	Your Rental Cost: Rs 2,075 Vehicle rented successfully	Pass
TC-09	View Total Income (after 3 rentals)	Total Income: Rs 35,975 (12,000 + 21,900 + 2,075)	Total Income is: Rs 35,975	Pass
TC-10	Try to rent already rented vehicle C001	Error: 'Already Rented Out.'	'Already Rented Out'	Pass
TC-11	Search Vehicle by ID: C001	Display complete details of C001 Show current availability status	'Not Available'	Pass
TC-12	Filter by Brand: Toyota	Display only C001 (Toyota Camry)	Display only C001 (Toyota Camry)	Pass
TC-13	Filter by Availability: Not Available	Display C001, V001, B001 (3 rented vehicles)	Display C001, V001, B001 (3 rented vehicles)	Pass
TC-14	Return Vehicle C001	Message: 'Vehicle (C001) Returned Successfully' Status changes to 'Available'	'Vehicle (C001) Returned Successfully'	Pass
TC-15	Try to return already returned vehicle C001	Error: 'Vehicle(C001) Already Returned'	'Vehicle(C001) Already Returned'	Pass
TC-16	View Total Income (after return)	Total Income: Rs 35,975 (Income unchanged after return)	Total Income: Rs 35,975	Pass
TC-17	Rent returned vehicle C001 again for 5 days	Rental Cost: Rs 30,000 Formula: (5000×5)+(5×200×5)	Rental Cost: Rs 30,000	Pass

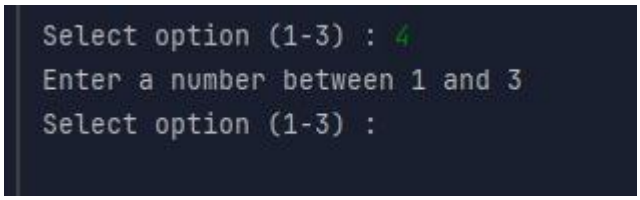
TC-18	View Final Total Income	Total Income: Rs 65,975 (35,975 + 30,000)	Total Income: Rs 65,975	Pass
TC-19	Invalid input: Menu choice: 10	Error: 'Enter a number between 1 and 8'	'Enter a number between 1 and 8'	Pass
TC-20	Invalid input: Negative rate: -5000	Error: 'Enter a positive decimal number'	‘Enter positive decimal number’	Pass

6.3. Validation tests and exception-handling scenarios

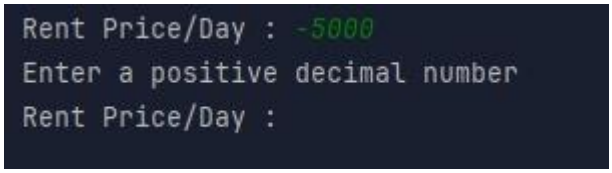
Entering a non-integer value when expecting an integer.



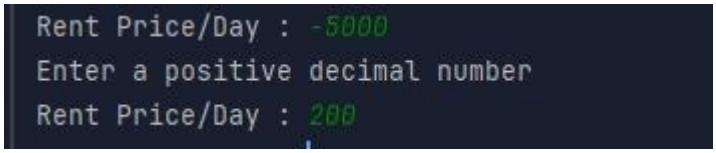
Entering out of range value



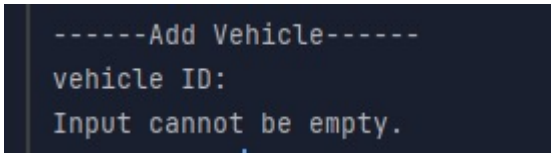
Entering a negative value when expecting a positive integer.



When proceeding forward without entering a value.



When proceeding forward without entering a value.



When entering a text instead of a double value.

7. Challenges and Solutions

7.1. Technical challenges faced during development

1. Input Validation and error handling
    - We have to take lot of user inputs in this programme. Every input must be validated and need to handle exception gracefully.
2. Managing vehicle availability.
    - There are few factors that affect vehicle availability. They all need to work smoothly without interfering each other.

7.2. How OOP concepts helped resolve these challenges

1. Input Validation and error handling
    - There is a dedicated class called InputValidation to handle input validation. All of the validation methods in this class are reusable.
2. Managing vehicle availability.
    - We use inheritance and runtime polymorphism to connect all the things that affects vehicle details and availability.

8.Limitation and Future Enhancements

8.1. Current limitation of the system

The current system cannot store vehicle data when the programme closes all the data vanishes. Also, the programme currently runs on a command-line interface. Moreover, anyone can access the system. The system cannot run directly on a user's computer because, it needs a coding platform like IntelliJ Idea.

8.2. Proposed improvements and extensions

We need to connect the programme with a database to ensure the data availability even after the programme is closed. Also, we need to create a Graphical User Interface (GUI) for this programme, and there is also a need for separate menus to admins and users. This will ensure data safety in the system. The programme must build as an executable programme to run on any personal or work computer.

9. Team Contribution

9.1. Individual responsibilities and effort percentage

Team Member		Responsibilities	Contribution
Index No	Name		
CIT-25-01-0374	S.M. Rajapakse	Team Leader – Coded most of the rental app code and supported each and every member to code their part.	20%
CIT-25-01-0533	S.S. Matharage	Completed the polymorphism part by coding the subclasses, helped with the rental main code and implemented try catch.	16%
CIT-25-01-0714	A.J.M.K.S. Abeykoon	Checked the code for validation and coded the challenging component of the code (Advanced search and filtering).	16%
CIT-25-01-0328	G.S. Jayalath	Coded the part where vehicles are stored uniquely in the array list.	16%
CIT-25-01-0281	S.M.S. Thathsarani	Created the report with all the explanations and tested the code for bugs and errors.	16%
CIT-25-01-0104	S.D.D. Dahanayake	Declared the variables and the methods in the super class.	16%

10. Conclusion

In conclusion, this assignment helps us to understand how object-Oriented Programming concepts applied in a real application. We learned how to design a proper class hierarchy, using abstraction and inheritance. Also, we learned to create runtime polymorphism through method overriding and parent class references. Encapsulation helps to protect the data using private attributes and public methods. We used input validation and exception handling to develop a successful menu-driven command-line application. The Vehicle Rental Management System met all functional requirements. So, this project helps us to improve our Java programming skills and understand the OOP concepts through practical implementation.