CSE 108: Offline Assignment 3

Vehicle Management System for a Car Workshop

(Using Inheritance, Interface, and Exception Handling in Java)

Bangladesh University of Engineering and Technology, BUET
Department of CSE

Submission Deadline: February 6th, 2026, 11.55 PM

1. Introduction

Object-Oriented Programming (OOP) is a fundamental paradigm in Java that enables developers to design modular, reusable, and robust software systems. This assignment focuses on applying three key OOP concepts:

- Inheritance
- Interface
- Exception Handling

Students are required to implement these concepts through a real-world use case involving a Vehicle Management System for a car workshop.

2. Problem Description

A car workshop provides maintenance, repair, and inspection services for different types of vehicles such as cars, bikes, and trucks. While all vehicles share some common characteristics, the service charges and behavior vary depending on the vehicle type.

The workshop requires a Java-based Vehicle Management System that can:

- Manage multiple types of vehicles
- Calculate service costs based on vehicle-specific logic
- Use inheritance to avoid code duplication
- Use interfaces to enforce service-related behavior
- Handle invalid inputs and runtime errors gracefully using exception handling

3. System Requirements

The system must support the following vehicle types:

- Car
- Bike
- Truck

Each vehicle must have:

- A unique registration number

- Owner name
- Base service cost

The system should validate inputs and ensure that invalid operations do not crash the program.

4. Functional Requirements

4.1 Task 1: Base Class Using Inheritance

Create an abstract class named `Vehicle` with:

- Attributes:
  - Registration Number
  - Owner Name
  - Base Service Cost
- A constructor to initialize attributes
- An abstract method to calculate service cost
- A concrete method to display vehicle details

4.2 Task 2: Derived Classes

Create the following subclasses that extend the `Vehicle` class:

- **Car**
  - Additional attribute: Number of doors
  - Service cost = Base cost + (Number of doors × 500)
- **Bike**
  - Additional attribute: Engine capacity
  - Service cost = Base cost + (Engine capacity × 2)
- **Truck**
  - Additional attribute: Load capacity
  - Service cost = Base cost + (Load capacity × 1000)

Each subclass must override the service cost calculation method.

4.3 Task 3: Interface Implementation

Create an interface named `Serviceable` with methods to:

- Service a vehicle
- Return the service cost

All vehicle subclasses must implement this interface.

4.4 Task 4: Exception Handling

Create a custom checked exception class named `ServiceException`. This exception should be raised when:

- Base service cost is negative
- Registration number is empty or null
- Service cost exceeds a defined maximum limit

Use `try-catch-finally` blocks to handle exceptions in the application.

4.5 Task 5: Workshop Management Class

Create a `WorkshopManager` class that:

- Stores multiple vehicles using a collection
- Adds new vehicles
- Displays all vehicle details
- Calculates total service revenue

Ensure proper exception handling when managing vehicles.

4.6 Task 6: Main Application

Create a main application class that:

- Demonstrates polymorphism using vehicle references
- Calls interface methods
- Handles exceptions gracefully
- Displays meaningful output messages

5. Submission Guidelines

- Submit all `.java` source files as a zip file. The zip file name should be like "2405001.zip" for the student with ID "2405001" and similarly for others.

6. Evaluation Criteria

Criteria	**Marks**
Use of Inheritance	25
Interface Implementation	25
Exception Handling	25
Program Correctness	15
Code organization and clarity	10
Total	**100**

No marks will be provided for the code which you fail to explain