Advanced Programming (OOP) Lab

Assignment–3

Report Instructions

This document defines the mandatory structure and assignment-specific questionnaires for the handwritten report corresponding to Assignment–3: Inheritance, Method Overriding, and Abstract Classes. The report will be assessed for depth of object-oriented reasoning, correctness of design choices, and evidence of individual understanding.

General Submission Rules

- The report must be handwritten and submitted individually.
- Do not rewrite complete class definitions; include only selective and relevant code snippets.
- All explanations must be original and concept-focused, not syntax-focused.
- Every implemented requirement and attempted bonus challenge must be reflected in the report.
- Reports with generic or templated explanations will be penalized.

Tip: *Use multiple colors for highlighting your explanations, references and annotations.*

Section A: Object-Oriented Design Overview

Explain the overall class hierarchy used in this assignment. Clearly describe the role of the Vehicle base class, its subclasses, and the abstract class. Justify why inheritance is an appropriate choice for this problem domain.

Section B: Inheritance and Attribute Sharing

Answer the following:

1. Which attributes and methods are inherited by subclasses, and why?
2. Identify one attribute that should not be duplicated in subclasses and explain why.
3. Explain how constructor chaining works in your class hierarchy.

Section C: Method Overriding Analysis

For the accelerate() and brake() methods:

- Explain why overriding is required instead of method overloading
- Describe how runtime method binding occurs when a Vehicle reference points to a subclass object
- Include one short code snippet showing an overridden method and explain its behavior

Section D: Abstract Class and Abstract Method Reasoning

Explain the purpose of the abstract ElectricVehicle class. Answer the following:

1. Why is ElectricVehicle declared abstract?

2. What design guarantee is provided by the abstract chargeBattery() method?
3. What would happen if chargeBattery() were implemented in the base Vehicle class?

Section E: Interface and Multiple Inheritance of Behavior

If you implemented the FuelEfficient interface:
- Explain how interfaces differ from abstract classes
- Justify why fuel efficiency is modeled as an interface
- Describe how a class implementing both inheritance and interfaces is resolved at compile time

Section F: Polymorphism Demonstration

Using the TestVehicle class:
- Explain how polymorphism is demonstrated using a Vehicle reference
- Describe the sequence of method calls when different vehicle objects are stored in a collection
- State one advantage of polymorphism shown by this design

Section G: What-If Design Variations

Answer any two of the following:
1. What design issues arise if Vehicle is declared final?
2. What changes are required to remove inheritance and use composition instead?
3. How would behavior change if overridden methods call super implementations?
4. What breaks if abstract methods are replaced with empty method bodies?

Section H: Individual Extension Explanation

Describe one individual enhancement you implemented (for example: battery state tracking, input validation, or extended polymorphic behavior). Explain how this change affects class relationships and method execution.

Section I: Error, Debugging, and Learning Reflection

Describe one design or implementation issue faced while completing this assignment. Explain how it was resolved and what deeper understanding of inheritance or polymorphism was gained.

Section J: AI Usage Disclosure

Declare whether AI or online resources were used. Clearly state:
- Which part of the design or implementation was assisted
- How you validated or modified the suggested solution