Research Project Initial Plan: Composition vs. Inheritance

Show Wai Yan/105293041

Overview

This research project delves into the concept of **Composition over Inheritance** in Object-Oriented Programming (OOP), aiming to compare these two approaches for achieving code reuse and flexibility in software design. The purpose is to evaluate how composition and inheritance impact the quality of object-oriented programs, focusing on aspects such as maintainability, flexibility, and reusability. This initial plan outlines the research question, the methodology to be used, and the steps to conduct the research, as required for the High Distinction Task 9.3.

Research Question

The primary research question is: How does the use of composition compared to inheritance affect the flexibility, maintainability, and reusability of object-oriented programs? This question seeks to uncover the practical implications of choosing composition or inheritance when designing software, providing insights into their effectiveness in creating robust and adaptable systems.

Sub-Questions

To guide the investigation, the following sub-questions will be explored:

- What are the key implementation and design differences between composition and inheritance?
- How do composition and inheritance influence class coupling and cohesion in a program?
- In which specific scenarios does composition outperform inheritance, and vice versa?
- How do real-world examples demonstrate the practical benefits and limitations of each approach?

Research Method

The research will employ a combination of theoretical analysis and practical experimentation to compare composition and inheritance. The methodology is designed to be systematic and repeatable, ensuring reliable data for analysis. The steps are as follows:

- **Literature Review**: Examine academic resources, books, and documentation on composition and inheritance in OOP, with a focus on C#. This will involve studying design patterns (e.g., Strategy for composition, Template Method for inheritance) and gathering qualitative data on their advantages and disadvantages.

- **Implementation of Test Programs**: Create two C# programs solving the same problem—one using inheritance and the other using composition. For instance, a vehicle management system could be implemented, with behaviors like movement or fuel consumption defined differently in each version. Both programs will have equivalent functionality for a fair comparison.
- **Data Collection**: Gather metrics from both implementations, including:
    - Code complexity (e.g., lines of code, number of classes).
    - Maintainability (e.g., ease of adding new features, like a new vehicle type).
    - Flexibility (e.g., ability to dynamically change behaviors).
    - Coupling and cohesion (e.g., class dependencies and modularity).
      Metrics will be collected using tools like Visual Studio's code metrics or manual analysis.
- **Analysis**: Compare the metrics to evaluate how each approach performs in terms of OOP principles like encapsulation and polymorphism. Identify scenarios where one approach is more effective based on the data.
- **Documentation**: Compile the findings into a research report with sections for abstract, introduction, method, results, discussion, and conclusion. The report will include tables and graphs to visualize the data and highlight key differences.

Next Steps

The next steps involve discussing this plan with lecturer to confirm its suitability. Following approval, the literature review will begin, followed by the design and implementation of the test programs. Data collection and analysis will then be conducted, culminating in the preparation of the research report.