SubHunter Stinks!

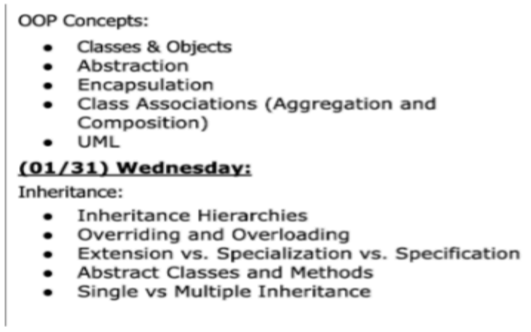
Goal: Clean up SubHunter with everything you know so far about OOP.

Can work in teams of 2.

Assignment #2 Part 1:

Enhance the SubHunter game by implementing Object-Oriented Programming (OOP) concepts.

Some OOP concepts we discussed include:



OOP Concepts:

- Classes & Objects
- Abstraction
- Encapsulation
- Class Associations (Aggregation and Composition)
- UML

(01/31) Wednesday:

Inheritance:

- Inheritance Hierarchies
- Overriding and Overloading
- Extension vs. Specialization vs. Specification
- Abstract Classes and Methods
- Single vs Multiple Inheritance

Some ideas for what to implement:

1. **Classes & Objects**
   - Implement classes to represent key game entities (e.g., submarines, player ship).
   - Utilize objects to instantiate and manage instances of these classes.
2. **Abstraction**
   - Apply abstraction to hide unnecessary details and expose only essential features.
   - Identify areas in the game code where abstraction can improve design.
3. **Encapsulation**
   - Enhance data organization and security by implementing encapsulation.
   - Protect sensitive data and ensure proper data access within the game code.
4. **Class Associations (Aggregation and Composition)**
   - Introduce associations between game entities using aggregation and composition.
   - Illustrate scenarios where these associations enhance the overall game structure.
5. **Inheritance**
   - Implement inheritance hierarchies to represent commonalities between game entities.
   - Demonstrate how inheritance can be used to model relationships within the game.

Assignment #2 Part 2:

Analyze the SubHunter code for common code smells.

Some smells we discussed include:

- Duplicated Code
- Long Methods
- Large Class
- Long Parameter List
- Divergent Change
- Shotgun Surgery
- Feature Envy
- Data Clumps
- Primitive Obsession
- Switch Statements
- Inappropriate Intimacy

Part 2: Refactoring Techniques:

Implement common refactoring techniques to address the identified code smells. This may include extracting methods, reducing code duplication, and improving the overall structure of the code.

A few we discussed include:

- Extracting Methods
- Extracting Classes and Interfaces
- Organizing Data
- Replacing Conditional Statements with Polymorphism
- Other relevant refactoring techniques based on the identified code smells.
- Review the PowerPoint slides for more...

Submission Details:

Write a one to two page report discussing the changes that you made and why you made those changes. Include as much detail as possible, however, do not just write filler, your assignment will be graded on quality, more so than quantity. If you are struggling to write a full page of high quality description, you might want to revisit the lectures.

Consider the following points in your report:

1. **OOP Implementation:**
   - Briefly outline the integration of OOP concepts (Classes & Objects, Abstraction, Encapsulation, Class Associations, Inheritance) in the SubHunter game.
2. **Code Smells Identification:**
   - Articulate specific issues associated with each code smell and their impact on code quality.
3. **Refactoring Choices and Techniques:**
   - Explain the rationale behind refactoring choices to address identified code smells.
4. **Integration of OOP Concepts with Refactoring:**
   - Briefly describe how OOP concepts from Part 1 seamlessly enhance the refactoring process in Part 2.
   - Highlight how these concepts contribute to improved code maintainability and readability.
5. **Challenges and Solutions:**
   - Summarize challenges faced during implementation and refactoring.
   - Provide succinct insights into the strategies used to overcome these challenges.

Also:

- A zip file containing your entire Android Studio Project

- Provide a screenshot of the SubHunter game running in the emulator to demonstrate successful implementation.

- SUBMIT YOUR REPORT AS A PDF FILE WITH YOUR NAME / NAMES