Final Project: Revamping the Snake Game

Description:

In this assignment, you will be tasked with refactoring and redesigning the Snake game in Chapter 17 to improve its structure, readability, and maintainability. The goal is to separate different functions into their own classes and apply design patterns where appropriate. You will also need to enhance the gameplay by introducing new features and challenges. You can work in groups of up to 4 people. Use everything you've learned so far to upgrade Snake to the next level!

Tasks:

Task 1: Code Refactoring and Class Separation (Must create new classes / interfaces / etc.):
100 Points

- Review the existing Snake game codebase provided to you.
- Identify areas where code can be refactored to improve readability and reduce redundancy.
- Separate different functions into their own classes to increase modularity and maintainability.
- Refactor code with Refactoring techniques.

Task 2: Design Pattern Implementation (**Must choose 3):**
100 Points

- Identify opportunities to apply design patterns within the codebase.

Some ideas:

- Implement the Strategy Pattern to separate audio-related functionality into the `Audio` and `AudioContext` classes.
- Implement the Builder Pattern to create `Apple` objects with different properties based on their type (good or bad).
- Attempt to implement the Composite Pattern for the `Snake` class (Note: If you encounter issues, document them).

**** These are recommended design patterns; you may use any which are applicable. ****

Task 3: Gameplay Enhancements **(Must choose 6)**
200 Points

- Introduce randomness to assign scores to good apples and bad apples.

- Power-Ups: Introduce power-up items that the snake can collect. These power-ups can have various effects, such as increasing the snake's speed, making it invulnerable for a short time, or allowing it to eat bad apples without penalties.

- Obstacles: Add obstacles on the game board that the snake needs to avoid. Colliding with an obstacle could result in a penalty, such as a decrease in the snake's length or speed.

- High Scores and Leaderboards: Implement a high score system that keeps track of the best scores achieved by players. Display the current game score as well while the game is running.

- Sound Effects and Music: Enhance the audio experience with background music and different sound effects for various in-game actions.

- Progressive Difficulty: Make the game progressively harder as the player advances, with faster apples spawning, more obstacles, or additional challenges. This keeps the game interesting for longer periods.

- Game Over Screen: After a game over, show the player's final score and provide options to restart the game or return to the start screen.

- Pause and Resume: Include the option to pause the game and resume it later. This is a useful feature for players who may need to temporarily leave the game.

- **If you have other ideas, run them by me first, but be creative. These features are just suggestions.**

Your game needs to have a “wow” factor. This could include lots of extra features, how well designed (and fun) the game is, or just good old-fashioned polish—anything that will make your game really stand out!

Submission Details

If you encounter difficulties with any part of the assignment, document your efforts and any issues faced. The goal is not just to implement design patterns but also to learn from the challenges you encounter during the refactoring process. I do expect tangible improvements to the Snake Game that are more than just a few new classes. Try your best to improve this game according to the assignment's requirements.

Report:

Reports and Progress Tracking: We will have three progress reports, scheduled every two weeks throughout the assignment. These reports are essential for tracking your progress and assessing your work. They should include the following:

2. Take a screenshot of your game running in the emulator. Include the full U/I of the emulator, including the control panel that has the power button just to the right of the main emulator screen.

3. 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 step 1 of this part.

There is not one fixed or correct answer to this assignment. You will be graded on how much effort you put into thinking about this simple program in an object-oriented way. Put effort into documenting your classes with significant comments that describe your thought process. Putting effort here as you go will almost guarantee that your report will be easier to write. You should summarize your detailed comments in your report. See the Piazza response from a past assignment below for more detail on this.

This is an individual or partner assignment. You may work with your partner, or you may work alone. If you are working with a partner then only one of you needs to submit the report and source code.

SUBMIT YOUR REPORT AS A PDF FILE. SUBMIT ALL .JAVA FILES FOR THE PROJECT. DO NOT ARCHIVE THEM, SUBMIT THEM SEPARATELY. SUBMIT A SCREENSHOT OF THE GAME RUNNING IN THE SIMULATOR, AS DESCRIBED ABOVE.

Also, include these in your Report:

a. Names of everyone involved and the work they contributed to.

b. A detailed description of the changes you have made since the last report.

c. A list of Git logs showing commits, including commit messages, commit authors, and commit timestamps. This will help us understand the evolution of your project over time.

Submit these reports on time. Only one group member should submit the assignment. The reports will ensure that all group members are actively contributing to the project.