Assignment 3 - Play Game

SENG301 Software Engineering II
Due date: 5pm on 31st May 2019
Drop dead date: 5pm on 7th June 2019 (no penalty)*

This assignment covers content taught in Moffat's part.

1 Prelude

You have been given a Game prototype. The Game is console based with only a small range of functionality being implemented. The player in this game visits many different rooms to find and collect as much treasure as possible. There is a monster (enemy) waiting to attack and take the player's life. When a player loses a life the game ends.

The module you have been given is responsible for setting up the game (number of rooms) and randomly taking the player through the rooms. The player has a choice of either searching for a treasure or leaving the room. If one of the rooms has an enemy waiting then the enemy will attack the player and the outcome of the fight (who wins) is randomly determined in the moment. If the player dies, the game ends at that point, but if the player wins he visits another room to search for treasure. This continues until a player either dies or has visited all the rooms.

Pania, a junior developer, has been asked to continue developing this game. She is not familiar with design principles. Your job is to help Pania along by explaining parts of the design or by making small modifications to the code. When you make any decisions, please report it in your explanations file providing your reasoning and justifications; this file is very important. Your justifications should include design principles you have learned in class.

2 What you should have

You should have:

- the `seng301-2019-assignment3.zip` file, which contains a small working Java program. The project is provided in such a way that it should be easy to import into IntelliJ, and
- this instruction file.

*There will be no penalty applied for submitting by the drop dead date

3 What you have to submit

- Please submit one file that contains:
  - the codebase containing your modifications (you do not need to submit any test files), and
  - your explanations file.
- Please make sure your name is part of the filename in the following format: `MyName-seng301-2019- assignment3.zip` (replacing *MyName* with your name).
- Please put a comment (e.g.#Task 1) in the code wherever you have made a change.

4 How will the marker mark this assignment?

The marker will:

1. look at the code you have changed and see if it is correct, appropriate, and adequate.
2. write statements in your `main()` method that check for correct operation of your program.
3. read your explanations' file and see if s/he is convinced by your design decisions.

5 Marker notes

- A significant portion of the marks are devoted to your explanations and decisions, so make sure these are in-depth.
- When documenting contracts or patterns, use the format shown in class.
- You should aim to make the simplest, smallest change possible.

6 Tasks

Before changing anything, make sure you can compile and run the program. Study the program to understand its design. Check out the given UML.

6.1 Task 1: Getting your hands dirty [5 marks]

In this task, you are to extend the play function, found in *AdventureGame* by adding *enter* a room option (you only have *leave* and *search* commands implemented at the moment). Make sure you change the least amount of code to achieve this. In particular, try to follow the open/close principle.

6.2 Task 2: Collections [5 marks]

In the current game set up, a player collects discovered treasure. The initial requirement is to collect a unique range of treasure; the possibility of having more than one treasure of the same kind was not taken care of. Change the code so the user can have more than one treasure of the same type while keeping the order of collected treasure.

Document your code changes in the assignment report and discuss your decision. If you also have to keep track of the rooms where the treasure was collected, what further changes (if any) would you need to make?

6.3 Task 3: OO Wisdom [10 marks]

Comment on the design of the existing code. How many flaws can you identify? Can you name them? For each criticism, explain how you would improve the code. Do not change any code. Assume the intent of the design is valid, even if you don't like the approach.

6.4 Task 4: Make the board unique [5 marks]

Currently, the board can be created multiple times throughout the duration of the game. This means that the current state of the game can be overridden at any point in time.

Change the implementation of the Board class so that its uniqueness of the board is preserved throughout the game. Document your changes as shown in class.

6.5 Task 5: Change Observer pattern implementation [25 marks]

Identify and document the Observer pattern implemented in the code. Java already has built-in facilities to implement the Observer pattern. We may increase the chances of errors by rolling our own Observer pattern.

We should use Java's Observer or Listeners. Modify the currently implemented Observer pattern using one of these two options provided by Java.

Document the pattern (as shown in class) and discuss your changes.

6.6 Task 6: Recognising patterns [30 marks]

Identify and document three GoF design patterns that you learned in class, already contained in this design (apart from Observer pattern that you have already worked on).

When documenting each pattern, provide a table that shows how every feature in the standard pattern (class names, attribute names, method names, and relationships) corresponds to a feature (or features) in this design. Few marks will be given for naming patterns. The majority of marks will be allocated for correctly describing how the pattern is applied. If a GoF concept is omitted in this application of the pattern, say so in the table. If a feature has no name, describe it, e.g. "the relationship between class A and class B." Do not discuss the intent of the patterns; just show where they are used. Do not document more than three patterns; if you do, only the first three will be marked.

Do not change any code.

6.7 Task 7: Design pattern implementation [20 marks]

Based on the existing design, MixedRoom uses functionality of two types of rooms. If in the future we need to expand this by introducing other types of rooms with some of them being more complex (being made up of two or more existing types) would Composite pattern add value to our design? To add it to the existing code what changes would you need to make? If this is possible and you think it would be useful, implement and document the changes and discuss the design principle(s) introduced or violated (if any).

If you think that implementing Composite pattern is not a good idea document your reasons and suggest an alternative way of managing your design in this case.