Assignment 4 (7 marks)

Maze Runner - Game

Objectives

- Design an object-oriented model for a full game.
- Apply the OOP concepts of inheritance and polymorphism to your design.
- Apply different design patterns to your model.
- Get to know the concepts of game programming.

Game Description

This game is like the famous Pac-Man game, instead, here we will have a maze, and you will be the actor (maze runner), you should run through the maze to find your way out! But be careful not to hit a bomb, you can also destroy any obstacle you face by a weapon, the weapon will be fulfilled with limited ammo. While you are moving inside the maze, you should collect some gifts in your way. The end of the game is by reaching the gate of the maze, or if you died due to hitting multiple bombs.

Tasks

- The maze should be with minimum 30*30 sizes and it can represented as a 2D matrix or any other way you see fit, the maze has one starting gate and one ending gate.
  Suggested: Interface Cell; Player, Gift, Wall will implement it.

- **[40%]** Your program should have a GUI, which is simple. The idea is updating the Canvas with the maze each time you move your maze runner, this should be an easy task after your experiences in the previous projects.

- You should move the maze runner using the 4 arrow keys in the keyboard, and you should fire the weapon with the space bar. Or you can use the mouse instead.

- Walls, maze runner, bombs, and gifts are all **images** being rendered on the GUI of your program.

- You should develop the game with animated **sprites**; it is something like gif images.

- You should not support only one type of maze's walls; you should support other walls shapes (you should have a class Wall). You should support at least two walls (stone, trees). Note that each wall has its own dimensions which should be stored as static variables in each concrete shape. One can be destroyed and the other can't.

- You should not support only one type of bombs; you should support other bombs shapes (you should have a class Bomb). You should support at least two bombs. Note that each bomb has its own dimensions which should be stored as static variables in each concrete shape. They can be differed in damage, decrease health o score, …

- You should not support only one type of gifts; you should support other gifts shapes (you should have a class Gift). You should support at least two gifts. Note that each gift has its own dimensions which should be stored as static variables in each concrete shape. One grants health, the other grants bullets, and a third one that grants armor.

- If the maze runner hit a bomb, his health should be decreased depending on the bomb's type.

- The maze runner has a gun with only 6 bullets, he can use them to destroy wall "trees". If a bullet hits one type of the bombs or all the gifts, they should be destroyed, the other types won't be affected. Taking one type of the gifts should give him some bullets.

- If the maze runner collected a gift, the score should be increased depending on the type of the gift, you should make an equation to calculate the score with considering some variables: time, collected gifts, bombs, …etc.

- Using **MVC** AP means you should isolate the three main parts of the design. Model, View and Controller. You will be enforced to use Observer design pattern to handle MVC.

- **[60%]** You should use (at least) the following patterns in your design:
  - **Singleton** (Ex: Player, Playing Ground, …)
  - **Factory** (Ex: Gifts, Walls, …)
  - **Decorator** (Ex: Put the armor on the player, …)
  - **Observer** (Ex: Info panel observes player health and score, …)
  - **State** (Ex: Game start and Stop state, Bomb state, …)
  - **Memento** (Ex: Checkpoint on score, …)
  - Optional: Adapter (Ex: Key Controller, …)
  - Optional: and choose at least another one you feel it suits your design.

  This assignment mainly tackles the application of the design patterns you studied during the course, so you are supposed to give time for the design.

- You need to provide save and load feature in the game. Also, you need to be able to pause and continue the game.

- You need to show the time, player score, health, …etc. while the game is running.

- Use your imagination, it is a game after all. Try to add to the above points, develop a good story for it, and try to make it fun to play with different types of actions.

Bonus

- There will be monsters in the maze moving around with random movements.
- You should not support only one type of monsters; you should support other monsters' shapes (you should have a class Monster). You should support at least two monsters. Note that each monster has its own dimensions which should be stored as static variables in each concrete shape.
- If the maze runner hit one type of monsters, he will die, the other type could steal his bullets or anything else.
- If the maze runner shot a monster, he will gain additional score, and the monster will die.
- Be creative and use your imagination for the bonus part and any other feature you can add to the game.

Notes

- Check the codes given in the sections/Schoology as you might use some of them here.
- You should maintain a clean code for your program, like: variables, classes, functions naming, code indentation, function length, …etc.
    - You can check this book for more details Clean Code – Robert C. Martin
- It is better to use a git version control service like bitbucket to collaborate with your teammates. However, your repository must be private so no one can copy from it, which will consider an act of cheating for both parties!
- This assignment mainly **tackles the design issues**. Heavy load will be on the good design in addition to the required patterns.

Deliverables

- You should work in **groups of four**.
- You should develop this assignment in Java.
- You are free to use any GUI framework you like. However, you **MUST NOT** use any **game engine**.
- You should submit this assignment online on Schoology, you should include the following into **one compressed file (.zip or .rar)**, then rename it with your **id1_id2_id3_id4**:
    - Your **code** folder.
    - A **self-executable jar file (JRE 1.8)**: The program should be executable by simply double-clicking the icon, given that you have a running JRE. And make sure you use relative pathing for all the images, files, and libraries.
- A discussion date will be determined later.
- Your program **MUST NOT** crash under any circumstances, even against malicious users!
- Late submission is not permitted.
- [**Cheating Policy**] Delivering a copy will be severely penalized for both parties, so delivering nothing is so much better than delivering a copy.

Good luck 😊