A JavaFX app utilising Design Patterns Assignment

@00470183

Christopher Kearsley-Brown

Table of Contents

[Introduction 1](#_Toc100568828)

[Explanation of Software Design 1](#_Toc100568829)

[Application of Design Patterns 2](#_Toc100568830)

[Conclusion 2](#_Toc100568831)

[References 2](#_Toc100568832)

# Introduction

This is a report that will detail the design process and the code of a interactive game. The game is designed to teach an aspect of science to primary school teacher in a fun manner. The software must contain multiple design patterns in the functionality of the game.

# Explanation of Software Design

In the design phase, the initial objective was to have an idea of a game. The idea that was created was that the player is incontrol of a character who enters a labyrinth to retrieve an item and escape. Once they retrieved this item, they would have to escape. Though this would not be an easy tasks as the labyrinth would contain traps and patroling enemies. If the player hits a trap, a question will appear where they have a time limit to answer a question correctly. If they answer correctly, they can continue. However, if they answer incorrectly or if the timer runs out, a game over will display. Also, if they come into contact with a patroling enemy, the game will also be over. When this happens, the players can restart the game. If the player beats the game, they will be congrated and have the option to start again.

Next was to relate this concept to science. The immune system became the perfect for this. The game’s synopsis became a virus has invaded the body has to infect the body’s cells. Once all the cells has been infected, the virus must escape to infect cells from other area of the body. White blood cells will be used as the patrolling enemies and hidden mucous membrane will be the traps. Once, a membrane has been walked through, it will reveal itself, however if it is entered again, the player will be attacked again.

These description of the game where then used to create user stories. These user stories are as follows:

* As a player, I want to move, so I can traverse the maze.
* As a player, I want to collect/infect the cells, so I can complete the game objectives.
* As a player, I want to escape the labyrinth when I’ve completed all objectives, to complete the game.
* As a player I want to be able to restart the game so I can try again.
* As a player, I want to be able to answer a question to escape the traps.
* As a player, I want to be able to evade the enemies of the game so I am not defeated.
* As a player, I want to be congratulated so that I know I have beaten the game.

These were used to set objectives when creating the software so that the user stories can be met.

It is time to design the software. The first UML diagram that I created was a class diagram of the software which can be seen below.

The class diagram was constructed using the user stories and the description of the software. The description gave information about the objects to create and what their methods would be. Therefore classes such as Player, Item, Enemy and Trap was created and contained methods and attributes that related to them.

Through looking at the existing classes within the class diagram and how they were going to function decided which design patterns should be implemented within the software. The class diagram was than extended to show the design patterns working within the software. The first design pattern that was used was Prototype. “This pattern allows an object to create customised object to create customised objects by cloning an instance of a prototype”. The main benefit the prototype is that this design patterns allows copying existing objects without making the code dependent on their classes. This is important as often fields may be private and therefore not visible outside the object(Refactoring Guru). The game is gonna contain multiple instances of the enemies and traps. These objects will exaxtly the same and therefore this design pattern will be the perfect use for it.

The next design pattern that will be implented is singleton. The Singleton pattern only allows the creation of one instance of class while providing a global access point to the instance. As well of ensuring only one object of a class exists, it provides security to it’s variables. The player class is perfect for the use of this design as there will only be on instance of the player and because it is central to the software, it will be interacting with the majority of the other elements of the software.

The observer pattern will also be implented within this software. This pattern “allows objects to register dependencies between objects, so that an object will notify those objects that are dependent on it when its state changes”. As this application is a game, this software is automatically going to implement this pattern to work efficiently. This is because the game will contain an EventHandler that will observe interactions from the user and then make changes and notify objects from these interactions.

Finally, the composite pattern will be implemented. This pattern “compose objects into tree structures to represent part-whole hierarchies. Composite lets clients individuals and compositions of objects uniformly”. This is most relevant for the map of the game. The map will be make of different shapes but will act as one when the player or enemies interact with it.

# Application of Design Patterns

The

# Key Elements of the Code

The

# Conclusion

The

# References

* <https://refactoring.guru/design-patterns/prototype>
* https://refactoring.guru/design-patterns/singleton