**Singleton Pattern**

The singleton pattern is one of the simplest design patterns. Sometimes we need to have only one instance of our class for example a single DB connection shared by multiple objects as creating a separate DB connection for every object may be costly. Similarly, there can be a single configuration manager or error manager in an application that handles all problems instead of creating multiple managers.

**Model View Controller**

The purpose of the model-view-controller pattern is to separate concerns of the programmer for easier manipulation of instructions of code and easier ability to remove and inject a new function or a function to replace another without much problems

**Factory Pattern**

The factory pattern has not been implemented in our code however the purpose of the factory pattern is to ensure that the user or client does not understand the production of the item produced, it is used to create a particular type of item at runtime and return to the abstract class type in which it was constructed: Example: If there are monsters in a game called Goblins and there are two types of them: Big Goblin and Small Goblin. The Big Goblin will have different characteristics than the small goblin and therefore needs its own information to store it. However, when the time comes for the goblin to appear on the screen, if produced with the factory pattern, the user or the hero or main character will not know what goblin will appear on the screen but all they know is that one will appear. However in the library a function can be used to determine if the goblin is big at all times, small at all times, if it is random which one is produced or if it produces one 80% of the time, etc. However we have found that in our code it will not be useful to construct the factory pattern since it produces randomness.

**Source Code Management and Code Generation**

IntelliJ IDEA application has been used to manage Code Generation and the use of Github and the Github Desktop Application has been used to maintain continuous integration as well as the management of source code on the different computers. JetBrains owns IntelliJ, the program does not belong to this group upon the production of our project or code.

References

Retrieved from <https://www.geeksforgeeks.org/singleton-design-pattern/>

Retrieved from <https://www.javacodegeeks.com/2018/05/a-guide-to-code-generation.html>

Retrieved from <https://www.tutorialspoint.com/design_pattern/mvc_pattern.htm>

Derek Banas, Retrieved from <https://www.youtube.com/watch?v=ub0DXaeV6hA>

Retrieved from <https://www.jetbrains.com/idea/download/>

Retrieved from <https://github.com/about>

Retrieved from <https://www.geeksforgeeks.org/singleton-design-pattern/>