SWEN30006 Report – Project 1 Tetris Madness

Part 1: Analysis of the current design

Firstly, the current design lacks low coupling, high cohesion, and polymorphism. In each block class, there are fifteen same methods. The common properties of block shapes are not encapsulated to a parent class. Thus, when adding new block shapes, those common properties should be implemented in the new shape class again. The current design employs a number of switch statements to generate random blocks, making it challenging to introduce additional block shapes for increased difficulty levels. The same problem also occurred in moving blocks. When moving blocks, it uses a combination of switch and if-else statements for each shape of blocks, which is redundant. In addition, in the medium and madness difficulty level, there are new requirements compared to the simple level. The current design has low cohesion, making it hard to implement new features such as change of speed and disable of rotate function.

Part 2: Proposed new design of simple tetris

As mentioned in Part 1, the current design lacks polymorphism. Therefore, based on polymorphism, a new parent class “Block” should be implemented to encapsulate all common methods and attributes of blocks except the setLocation method which is unique to each shape. All other shapes should extend this class. For three new shapes in medium and madness levels, they can be implemented easily by extending the Block class as well. In the current analysis, the Tetris class is responsible for creating random blocks. However, to achieve low coupling and high cohesion and based on creator and pure fabrication, a new class RandomBlock should be implemented and responsible for creating new blocks. Compared to the simple tetris, medium and madness have more features. Therefore, to achieve high cohesion, low coupling and polymorphism, the same features for all three difficulty levels can be preserved in the AbstractTetris class. The features for simple tetris can be encapsulated to a new class TetrisEasy and extend the AbstractTetris class. For the other two difficulty levels, they can be implemented easily by extending the Tetris class.

Part 3: Proposed design of extended version

In the extended version, three more shapes should be added. Based on polymorphism, they can be implemented by extending Block class and adding their

own setLocation logic. In addition to new shapes, in medium level, the speed will get 20% faster and in madness level, the speed will be random. As mentioned in Part 2, a class called AbstractTetris including all the same features for three levels has been created based on polymorphism. Thus, medium and madness level can just be implemented by designing two new classes TetrisMedian and TetrisHard and extending AbstractTetris class. For speed features, they can override the getSimulationTime() method in AbstractTetris class to change the speed. For game statistics, it could be implemented in AbstractTetris class, but based on protected variations and low coupling, a new class GameStat is designed to be responsible for all game statistics recording.