SWEN30006 Project 1-Snakes and Ladders Report

Workshop 15, Team 10

Task 1

This task allows the game to use a preset number (in property file) of dice, the preset number is random, and it is fixed since the game started.

The key point for this task is the roll() method, which is in the NavigationPane class. Referring to the GRASP principles, the NavigationPane is the role of Controller, because it contains many necessary user inputs and outputs. But it also contains game operation method, this could cause Low Cohesion problem. The solution we applied is Pure Fabrication to reinforce the cohesion, which is to create a new class called DiceRoller which only responsible for rolling dice. Moreover, to achieve the goal which is rolling a die multiple times (pre-set number) and puppet moving after all rolling done instead of puppet moving after every rolling. And, the Die class should also be modified. In this case, every Die should be created by DiceRoller instead of NavigationPane. Therefore, Creator pattern is formed between Die and DiceRoller.

Thus, the basic operation logic for task1 is NagivationPane stores the number of dice from property file, when dice need to be rolled, NavigationPane will call DiceRoller to execute roll() method to create new Die object, after finish the rolling, return the total values to NavigationPane.

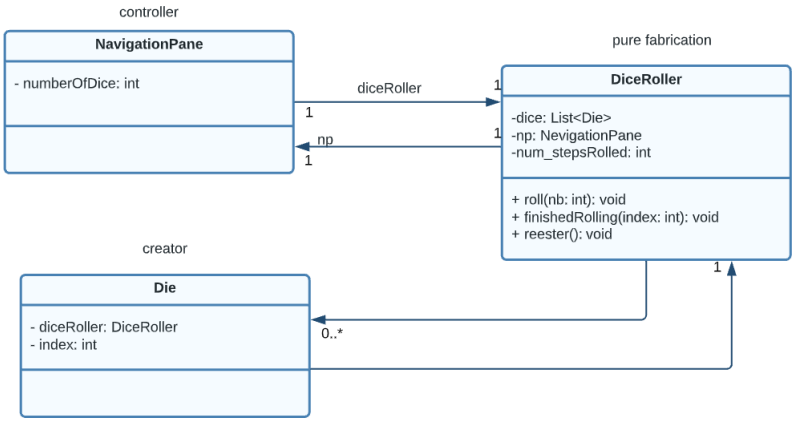

Figure 1: Partial Design Class Diagram for Task1

Task 2

This task allows players to not travel down a path symbol if they rolled the lowest possible value to land on the symbol square.

It is mainly about the player movement. Referring to the base code, there is a go() methodin the Puppet class, which is to set how many squares the player should move. So, the requirement is to determine whether a rolled is the lowest number, if it is and the player reaches to the downward path symbol, the player will not travel down. Based on Information Expert principle, we added a boolean attribute isLowest in Puppet class to check whether a rolled is the lowest possible number.

Task 3

For this task, first we need to check whether the player is moving back. So. we added a new attribute isBack: boolean and a setter setBack in Puppet class. Through NavigationPane class to check whether the player lands on the same square as their opponent after finished moving. If it does, set isBack to true, opponent moves one square backwards. Then, we modify moveToNextCell method in Puppet class to determine whether move forwards or backwards and counting number of steps need to move. Because the act() method depends on cellIndex to move the player, so invoke cellToLocation method from GamePane class to update the cellIndex. Also, we added getCurrentPuppetIndex() method in GamePane class. NavigationPane class invokes getCurrentPuppetIndex() would know which player just finished rolled and which square it is on now. If another player lands on the same square as the one just finished rolled, the setBack which in Puppet class set to be true, then activate go() method to move one square backwards.

In particular, we added an attribute isMovedOnCon in Puppet class, which is applied to avoid one player go backwards, if it collides another puppet through connection.

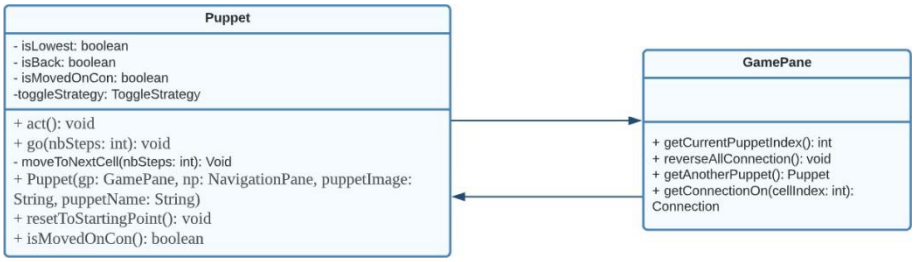

Figure 2: Partial Design Class Diagram for Task2&3

Task 4

For auto player, we applied Protected Variation principle, because the strategy would be changed in the future. So, we created an interface called ToggleStrategy which contain abstract method for template of Toggle strategy, and a new class called RealToggleStrategy to apply the ToggleStrategy interface. For future possible changes of strategy, only modify the RealToggleStrategy or adding new method to interface. For simulated player, they can apply the strategy to decide whether toggle.

If a player decides to toggle, all connections would be reversed. So, we added a new method called reverseConnection() in Connection class to switch start and end, a new method toggleButton to check whether toggle button is pressed. Also, we added a related method reverseAllConnection() in GamePane class, used to reverse all exist connections. When an auto player has decided to toggle,press the toggle button, the reverseAllConnection() method will be called.

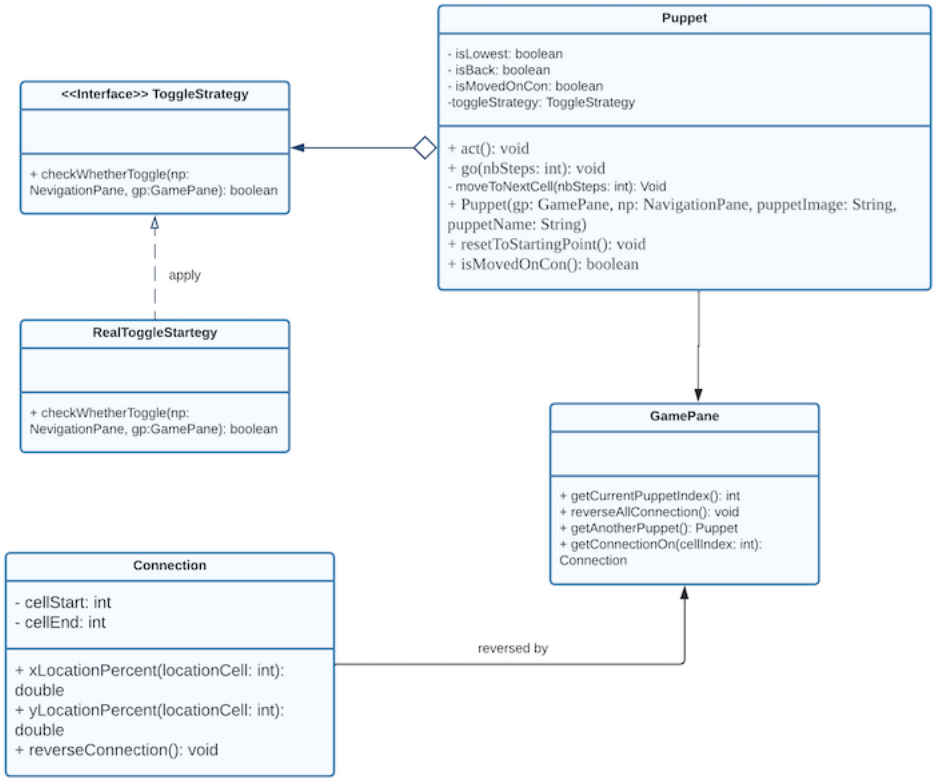

Figure 3: Partial Design Class Diagram for Task4

Task 5

For this task, refer to Pure Fabrication principle, need to create a new class called InformationCount to only record basic statistics for each puppet, with attributes playerIndex: String, HashMap<integer, integer>, travelDown: int and travelUp: int, and methods for recording the numbers a player rolled and how many times a player rolled a specific number, keeping track of player's movements (forwards or backwards), and howmany squares travelled for a movement. It will print out information for each player each turn. If additional statistics will be added in the future, only need to add new attributes or methods in the InformationCount class.

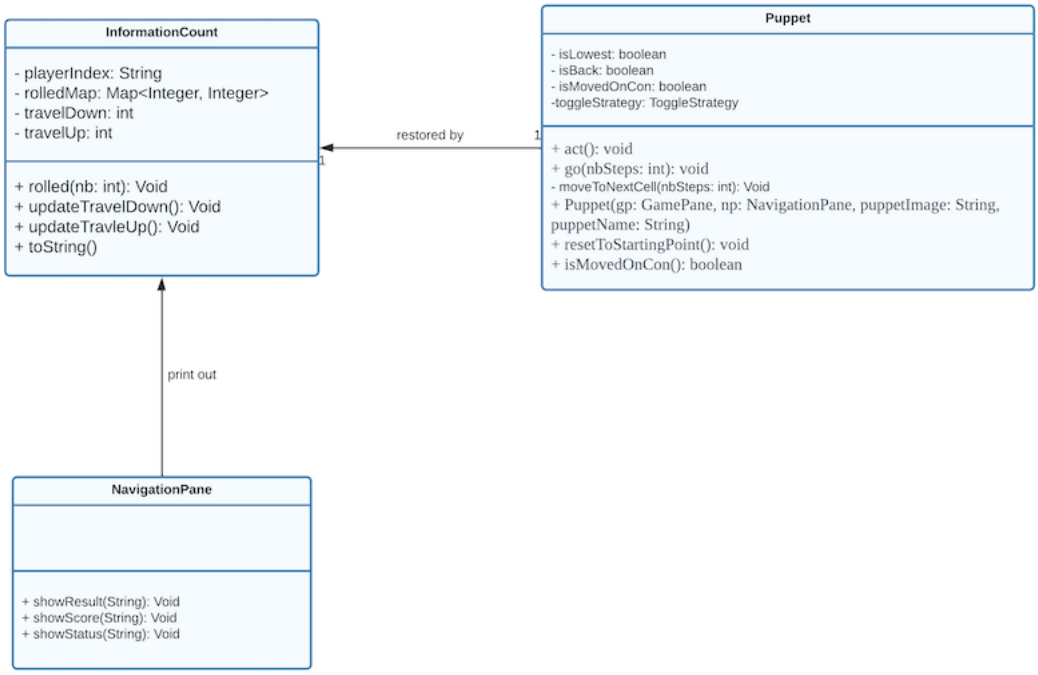

Figure 4: Partial Design Class Diagram for Task5