JC2002_Assignment

1. Introduction

In this ASSIGNMENT, I wrote a mini-game based on template code that creates a 2D grid map in which there exists one player and three monsters with all 100 health points located in each of the four corners. Each turn the player can move up, down, left and right according to the command, meanwhile the monsters will move randomly, if the player tries to move to a position occupied by a monster, the player is attacking the monster, and the monster has a 50% probability of losing 50 health points, on the contrary, if the monster tries to move to a position occupied by the player, the monster is attacking the player, and the player has a 30% probability of losing 20 health points. When a character's health points reach 0, that character dies, and the game is lost if the player dies first, or won if the monsters all die first.

There are two parts to this assignment, the first is the Character section, which names some variables and how to change them. The second part is the Game part, which mainly calls the variables and methods in the Character part, realizing the creation of the map and the internal logic of the game.

2. Classes

GameCharacter class

In this class, variables are declared for the name, health value and position of the column, and different methods are used to set them and return their values. In addition, two abstract methods, hurtCharacter() and successfulDefense(), are created for subclasses to override in order to implement attacks and reduce health points.

Monster class

In this class, there are three main methods implemented. First, there is an override of the hurtCharacter() method, which determines the success of a Monster attack and cuts the player's health points by calling the successfulDefense() method, and the successfulDefense() method. The latter makes a probabilistic judgment of the success of an attack by using a random number. Second, because Monster's movement is random, a method decideMove() is created to probabilistically determine the random direction of Monster's movement by random number generation.

Player class

In this class, similar to the Monster class, the methods hurtCharacter() and successfulDefense() are overridden to reduce the Monster's health points accordingly, based on the probability given to the success of the attack and the amount of blood deducted from the success of the attack.

Map class

In this class, in order to meet the requirements of the mission, a 2D array is initialized, a 2D map is created based on the length and width of the inputs, as well as the variables row and column from the GameCharacter class are used to store the indexes of the array, whereby each character's position on the map is described. One of the private methods, initialiseArray(), uses a for loop to initialize each point of the map to "." , and initializes Player with "*" and Monster with "%" in each of the four corners of the map. The public method printLayout() prints the current map to the console via a nested for loop.

Game class

In this class, the iteration of rounds is implemented, complete with the actions of the different characters in each round as well as determining whether the game is won or lost. I have divided a round into two parts in this class. The first part is the character movement part, which consists of two types of character movement, first is the Monster part, which consists of calling the method moveCharacter() to realize the movement of Monster. The second part is the Player part, which first determines whether the health points are greater than 0, and if they are greater than 0, then it calls the function to move the Player in the same way. The second part of the class is the Win or Fail part of the game. First, we use the Boolean value livingMonster to check if there are still alive Monsters in order to determine whether we have won or not, and second, we determine the Player's health points, and if they are equal to 0, we determine that it is a failure. At the end of the class, the health points of each character are also printed to show the health points more visually.

GameLogic class

In this class, the main implementation of the character's position of the movement and their position on the map icon to change and attack the implementation of the logic, including four methods, respectively, moveUp (), moveDown (), moveLeft (), moveDown (), these methods in the implementation of the logic is more or less the same, to one of the example. In the first step, I divide the situation into two categories, based on whether the moving target is within the map index range, and in the second step, I divide the situation where the moving target is within the map index range into four categories, based on the fact that the original location of the moving target is "." ," *","%" or " x", and in the third step, I categorize each case of a different target into two categories, based on whether the moving character is a Player or a Monster accordingly, all possible cases are covered, we can write different code to realize the character's movement or attack according to different situations, for example, if Player wants to move to the position where there is a dead Monster, it returns " Character already dead". Figure 1 shows the logical structure of the GameLogic class using the moveUp() method as an example.

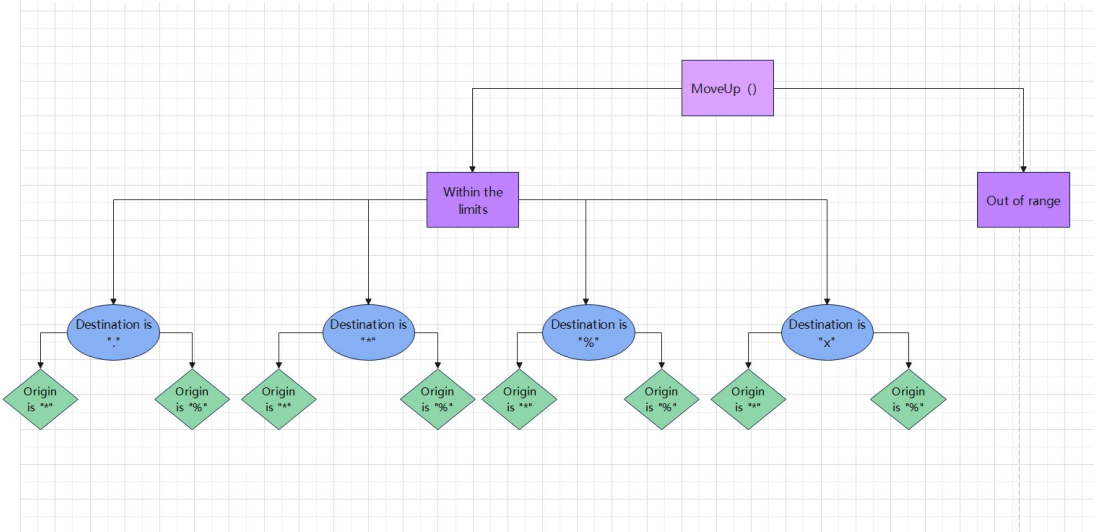

Figure 1

RunGame class

This class implements the logic to create a new object Game, get the height and width

in the main method and determine if they are correct, and determine the end of the game. There are two variables, the first is the gameOver variable, which takes user input from nextRound() and determines whether the game is over or not. The second is the integer variable round, which is used to update and record the number of rounds. Finally, I called the printLayout() method, which prints the current map to help the player visualize the current progress of the game.

3. Testing

We used the Junit framework to test our code. JUnit is an open source unit testing framework for the Java programming language. It allows developers to write and run automated tests to verify that various parts of the code work as expected. For example, in Task 1.2's test for SayName(), the checkSayName() method is used to validate it. This test verifies that the sayName() method accurately returns the name associated with the GameCharacter object.

4. Challenges

About logic

Throughout the assignment, I think the part that caused me the most trouble was the GameLogic class, where I mainly added new code to implement the new task, however, I encountered logic issues during implementation that kept me from passing the test because I didn't fully understand the entire task first. In the end, my solution was to discuss each of the original characters to be moved and the character to move the target to categorize them, as I said in the introduction to the GameLogic class, in order to enhance the logic of the code, avoid errors, and improve its extensibility.

About test

Throughout the assignment's test, there are some loopholes, for example, in the last test of assignment 5.5, the Player's health point is set to 0 first, but in the subsequent setting of the three Monsters' health points to 0, the Player's health points are not reset first, which makes it possible that if the code for the Player's turn code first, then the game will keep failing to win, resulting in failure to pass the test, after finding this problem, I wrote the Player's turn and the announcement of the Player's death in my code separately, and wrote the report of the announcement of the Player's death after the victory report, which finally solved this problem.

Bugs

During the game, because Player and Monster can't walk to Monster's corpse, there will be a Monster blocked in the corner by two corpses and can't walk out, and Player can't attack it, we can solve this problem by appropriately adjusting the map size and allowing Player and Monster to walk to the corpses. method to solve this problem.

Ideas for improvement

In this game, since I discussed every current position character and target position character situation in detail in GameLogic class, there are some situations that we don't have in the game, such as the situation where the current position is Player and the target position is also Player, since we only have one player, so my vision is that we can try to expand this game into a two-player or multiplayer mini-game, where you kill Monsters through teamwork.