SNAKES AND LADDERS

CS1701 – LEVEL 1 GROUP PROJECT LECTURES AND TUTORIALS

ASSIGNMENT 3 – SOFTWARE IMPLEMENTATION (CS1810)

NAME – MUNZER MAHMOOD

STUDENT ID – 2050179

GAME NUMBER – 2

VARIATION – STUDENT A

GROUP – 3 YELLOW (PROFESSOR GILBERT)

REQUIREMENT SPECIFICATION

LEGEND:

Specification MET ✓

Specification NOT MET ✗

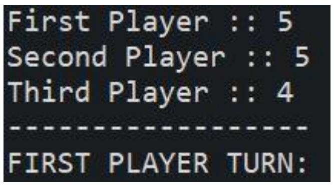
- The program will allow users to play Snakes and Ladders against 1-3 other players. ✓
- The board needs to have 100 grids/boxes for the players to play on. ✓
- All the players need different colours assigned to them. ✗
- All the players need to start from grid/box 1. ✓
- Each of the player will need to roll the dice when their turn comes. How many spaces they will move will be determined by the number that lands on the dice. For example, if one of their position is at 3 and they roll a 3, they will move 3 to grid/box 6. ✓
- Players will repeat their turn if they roll a 6 on their dice. ✗
- The program should have snakes and ladders with their positions assigned to different boxes. ✓
- The program should give the players access to the variations, Big Stick and Cookie. ✗
- If for example player is on box,98; the value 2 must be rolled in order to go to 100 and win. Starting from 94 to 99, the exact values to add up to 100 must be rolled to win. ✓
- The big stick should enable players to use it as an extra step on ladders by 10 squares but not for ladders that lead to the finishing square, the row ranging from 91-100 ✗
- The cookie should enable the players to avoid getting eaten by snakes. ✗
- The snakes' head and top of the ladder should always be on a higher number compared to its tail and bottom of ladder. ✓
- No Ladders starting on rows 91 -100. ✓
- The winner will be displayed. ✓

Due to time constraints and a few extenuating circumstances within my family, I did not find the time to carry out my program in Graphical User Interface as I had intended to by creating all the graphical templates beforehand (as seen on assignment 2 report). This caused me to fail in visioning how I was going to code in console, the functionalities such as players having different

colours assigned to them and my variation functionalities such as the cookie and big sticks to avoid being eaten by snakes and using the sticks for extra steps on the ladders.

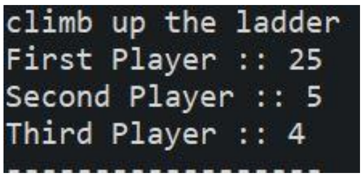
ADDITIONAL FUNCTIONALITIES

However, regardless of making it in console, I was still able to put in an additional few message functionalities as shown below.



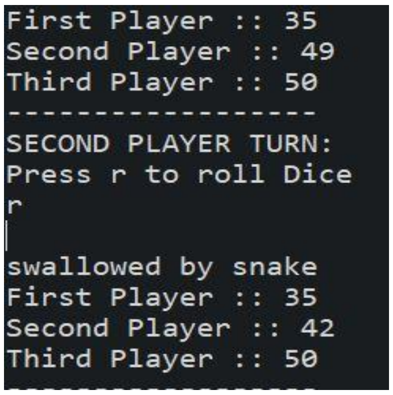
```
First Player :: 5
Second Player :: 5
Third Player :: 4
------------------
FIRST PLAYER TURN:
```

All position of players are displayed and also whose turn it is going to be next is also displayed in text. In this case, it is the FIRST PLAYER'S turn.



```
climb up the ladder
First Player :: 25
Second Player :: 5
Third Player :: 4
```

If players reach a box where the bottom of the ladder's coordinates have been placed, they climb up the ladder and the message is displayed.



```
First Player :: 35
Second Player :: 49
Third Player :: 50
------------------
SECOND PLAYER TURN:
Press r to roll Dice
r

swallowed by snake
First Player :: 35
Second Player :: 42
Third Player :: 50
```

As seen on the screenshot, previously second player was on box 49 and after it was the second player's turn and rolled the dice, the player landed on a snake and went down to box 42.

CONSISTENCY AND CHANGES BETWEEN ALGORITHM AND USER INTERFACE DESIGN

The assignment two report had promised a graphical user interface for the game however I was not able to implement it graphically and did it in console instead due to time limitations.

The repeat turn feature for when dice roll = 6 was also missing in the program due to lack of skill as I couldn't implement it during the multiple method changes I had to make to change player turns more efficiently.

Neither could I implement the message what number the player has rolled on the dice in the program.

TESTING

Project Game: Snakes and Ladders Variation A

Test Case ID: **TC_27F9**

Test Title: **Snakes & Ladders Testing**

Test Priority: **High**

Test Designed By: **Munzer Mahmood (2050179)**

Test Executed By: **Munzer Mahmood(2050179)**

Date of Test: **11/03/2021**

Starting Time of Test: 18:45

Ending Time of Test: 19:27

INPUT OR ACTION	EXPECTED RESULTS	REAL RESULTS	PASS OR FAIL	REASON IF FAILED
Start off Game with First player's turn	The user should see "First Player's turn" after the beginning of the game	Same as expected, "First Player's turn"	**PASS**	**N/A**
Input 'r'	The users should see message asking to input the	Inputting r caused dice to run for current player's turn	**PASS**	N/A

	value r to roll dice			
While program running, dice rolled value landed 6	The current player get his/her turn repeated	The turn does not get repeated and moves on to the next player	**FAIL**	The repeat turn is a Boolean variable which can either be true or false for the players but wasn’t added within the event actions performed of each player.
Starting the program to see if it starts players at grid/box 1	The players should start off with their positions at grid/box 1	They do but however it does not display it as players roll a 5 on dice at the start and move up straight away whereas if they were starting at 0, they would have to go to 1 first and then 5	**50/50**	Values for player positions are displayed null in the beginning yet players still jump to different positions straight away after rolling dice which wouldn’t be possible if they started off from nothing/0.
Snakes’ head and top of ladder	The users should find the position of snakes’ head in a higher value than its tail, same with the top of the ladder with a higher value than its bottom	When landed on a snake, players are seen to move down to a position with a smaller number than the position they were at before. They are seen to move up to a higher position when at the bottom of a ladder	**PASS**	
Testing to see if rows 94-99 require the exact values that add up to	The user should see that if in for example position 97, he/she must	Player remains in same position(97) regardless of it being the player’s turn,	**PASS**	

100 for players to win	get a 3 on dice to move to 100	as the player did not roll a 3 and hence the turn moves on to the next player		
Testing to see if the board has 100 boxes for the players to move about	The program should have 100 grids in total where players acquire different positions	The winner is displayed as soon as the player who won reaches box value 100 meaning there are 100 grids/boxes	**PASS**	

SOURCE CODE

```
import java.util.HashMap; //implements an associative array to compute index 
positions and maps keys to values
import java.util.Map;
import java.util.Random;
import java.util.Scanner;

public class SL2{

	public static void main(String[] args) {
		SnakeNLadder s = new             ();
		s.          ();

	}

}

class SnakeNLadder
{

	final static int FINISHLINE = 100;


	static Map<Integer,Integer> snake = new        <Integer,Integer>();
	static Map<Integer,Integer> ladder = new        <Integer,Integer>();

	{
		snake.put(99,54);
		snake.put(70,55); // since hashmaps map keys to values. the first 
coordinate(a key)
		snake.put(52,42); //  is the head of the snake which leads to the 
(value) tail of the snake
		snake.put(25,2);
```

```
             snake.put(95,72);

             ladder.put(6,25);
             ladder.put(11,40); // similarly with ladders, stores the starting of
a ladder as key and when
             ladder.put(60,85); // index value in array is reached, the players
move up to the top of the
             ladder.put(46,90); // ladder(higher value)
             ladder.put(17,69);
       }



      public int rollDice()
       {
             int n = 0;
             Random r = new       ();
             n=r.        (7);
             return (n==0?1:n);
       }
      public void playerTurn(int players){ // method to display message i-e which
player is going to roll
             String s = "";
             if(players==1){
                    s="FIRST";    // for player1 display FIRST
             }
             else if(players==2){
                    s="SECOND"; // for player2 diplay SECOND
             }
             else if(players==3){
                    s="THIRD"; // for player3 display THIRD
             }
             System.out.       (s+" PLAYER TURN: "); // followed by PLAYER TURN:
       }
      public void startGame()
       {
             int player1 =0, player2=0, player3=0; // three players
             boolean isP1 = true, isP2 = false, isP3 = false; // booleans to know
which player is rolling
             // at first Player1 will roll therefore it's true others are false
             Scanner s = new        (System.in);
             String str;
             int diceValue =0;
             do
             {
                    if(isP1){ // if P1 is rolling if isP1 is true then display the
text
                                    (1);
                    }
                    else if(isP2){
                                    (2); // if P2 is rolling if isP2 is true then
display the text
                    }
                    else if(isP3){
                                    (3);     // if P3 is rolling if isP3 is true
then display the text
                    }
                    // then print roll msg and roll the dice
                    System.out.       ("Press r to roll Dice");
                    str = s.    ();
                    diceValue =         ();


                    // if it was P1's turn then
                    if(isP1)
                    {
                          isP1 = false; // turn is over therefore make it false
                          isP2 = true; // after p1 always comes the turn of
player2 therefore making it true
                          player1 =                     (player1,diceValue);
                          System.out.       ("First Player :: " + player1);
                          System.out.       ("Second Player :: " + player2);
                          System.out.       ("Third Player :: " + player3);
                          System.out.       ("------------------");
                          if(     (player1))
                          {
                                System.out.       ("First player wins");
                                return;
                          }

                    }

                    else if(isP2)


                    {
                          isP2 = false; // turn is over therefore making it false
and moving to next player's turn
                          isP3 = true; // it's 3rd player's turn, after player2
comes player3
                          player2 =                     (player2,diceValue);
                          System.out.       ("First Player :: " + player1);
                          System.out.       ("Second Player :: " + player2);
                          System.out.       ("Third Player :: " + player3);
                          System.out.       ("------------------");
                          if(     (player2))
                          {
                                System.out.       ("Second player wins");
                                return;
                          }
                    }
                    else if(isP3)
                    {
                          isP3 = false; // turn is over, all three players
rolled, now again loop back firstPlayer will start rolling
                          isP1 = true;// therefore making it true
                          player3 =                     (player3,diceValue);
                          System.out.       ("First Player :: " + player1);
                          System.out.       ("Second Player :: " + player2);
                          System.out.       ("Third Player :: " + player3);
                          System.out.       ("------------------");
                          if(     (player3))
                          {
                                System.out.       ("Third player wins");
                                return;
                          }
```

```
            }



        }while("r".       (str));
    }


    public int calculatePlayerValue(int player, int diceValue)
    {
        player = player + diceValue;

        if(player > FINISHLINE)
        {
            player = player - diceValue; // if the value of the player is
more than the finishline(100)
            return player;              // then return function is used to
display the winner
        }

        if(snake.get(player)!=null)
        {
            System.out.       ("\nswallowed by snake");
            player= snake.get(player);
        }

        if(ladder.get(player)!=null)
        {
            System.out.       ("climb up the ladder");
            player= ladder.get(player);
        }
        return player;
    }

    public boolean isWin(int player)
    {
        return FINISHLINE == player;
    }

}
```