

[Final report]

[Software Project Lab-1]



Ludo game

*Final Report on*

***Ludo Game***

*Submitted to,*

SPL 1 Evaluation Committee 2022-2023

Bachelor of Science in Software Engineering

Institute of Information Technology

Noakhali Science and Technology University

*Submitted by,*

Makmudul Hasan Rabbi

[rabbi2516@student.nstu.edu.bd](mailto:rabbi2516@student.nstu.edu.bd)

*Supervised by,*

Dipanita Saha

[dipanita.iit@nstu.edu.bd](mailto:dipanita.iit@nstu.edu.bd)

Submission Date: 10 September 2023

*Final Report on* ***Ludo Game***

*Submitted By,*

Makmudul Hasan Rabbi

ASH2125008M

Year: 02 , Term: 01

Institute of Information Technology

[rabbi2516@student.nstu.edu.bd](mailto:rabbi2516@student.nstu.edu.bd)

*Approved by,*

Dipanita Saha

Assistant Professor,

Institute of Information Technology

Noakhali Science and Technology University

[dipanita.iit@nstu.edu.bd](mailto:dipanita.iit@nstu.edu.bd)

# Introduction

Ludo is a classic and popular board game that has been enjoyed by people of all ages for generations. It is a game of strategy, luck, and social interaction, often played by two to four players. It is simple to learn, yet it offers depth through its strategic decisions and unpredictable outcomes. Whether played casually for fun or in a more competitive setting, Ludo remains a beloved pastime enjoyed by people worldwide.

Objectives

* Move Tokens to the Home Column
* Capture Opponent's Tokens
* Block Opponents
* Use the Dice Effectively
* Strategic Token Placement
* Race to the Finish and win the Game

Features

* Multiplayer Gameplay
* Turn-Based System
* Dice Roll Simulation
* Competitive Gameplay
* Safety Zones
* Randomness and Luck

Tools and Resources

* Language: Java
* Tools: VS Code and Intellij

Deliverables

* Source Code
* Documents
* JAR File

Challenges

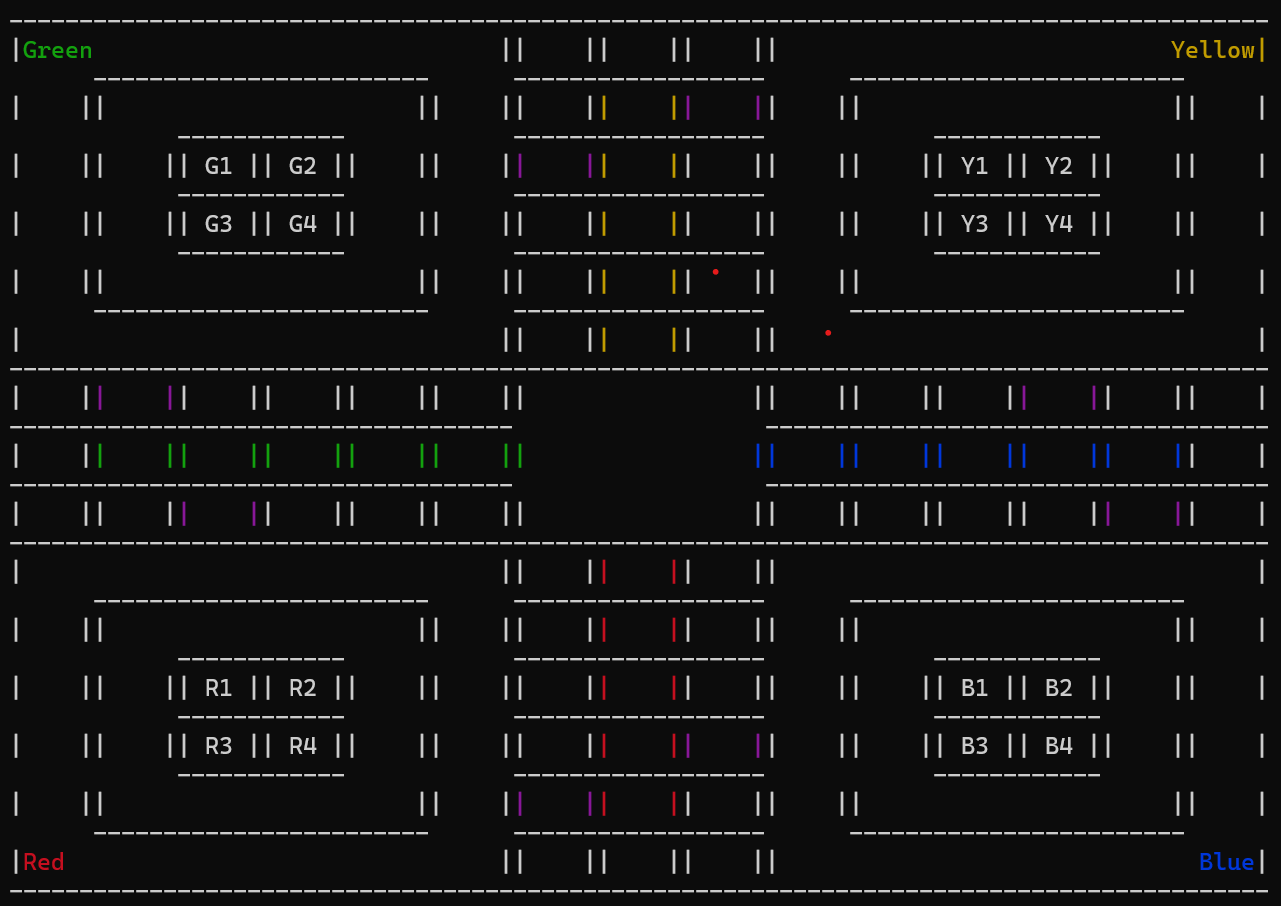
* Development Challenges
* No support of images and animations
* Multiplayer functionality
* Turn Management and Fairness
* Strategic Decision-Making

Implementation

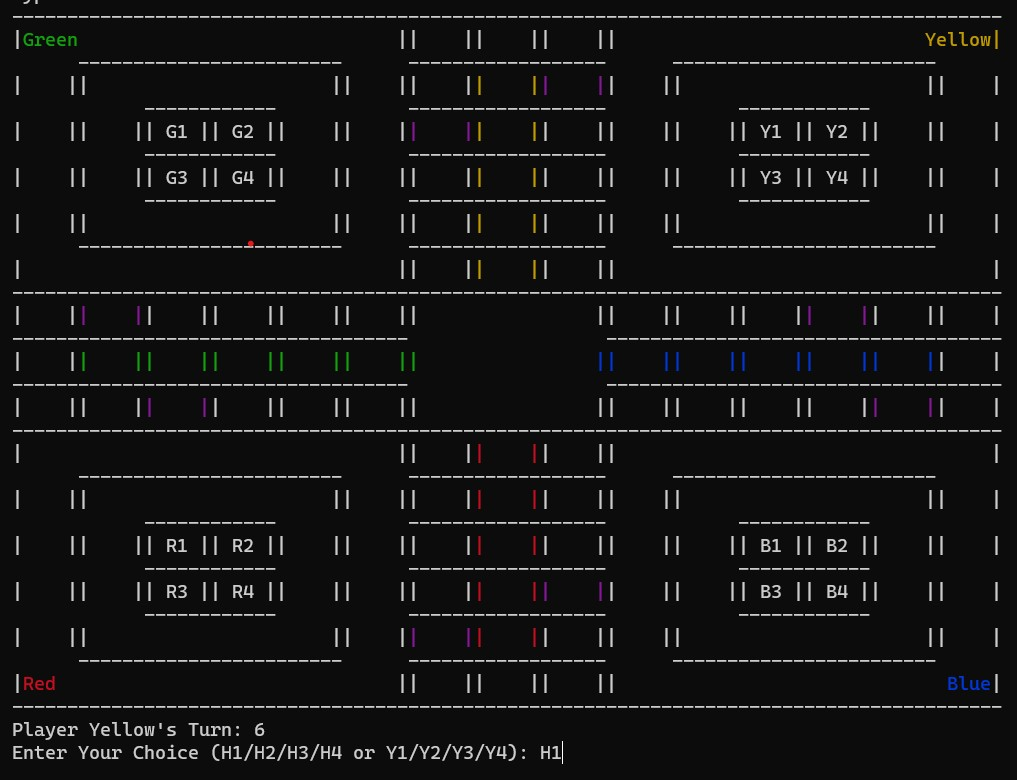
* Game Logic and Rules Implementation
* Board Representation and Data Structures
* Randomization and Dice Roll
* Player Interaction and Turns
* Game Progression and Winning Conditions

Initial Part of The Game

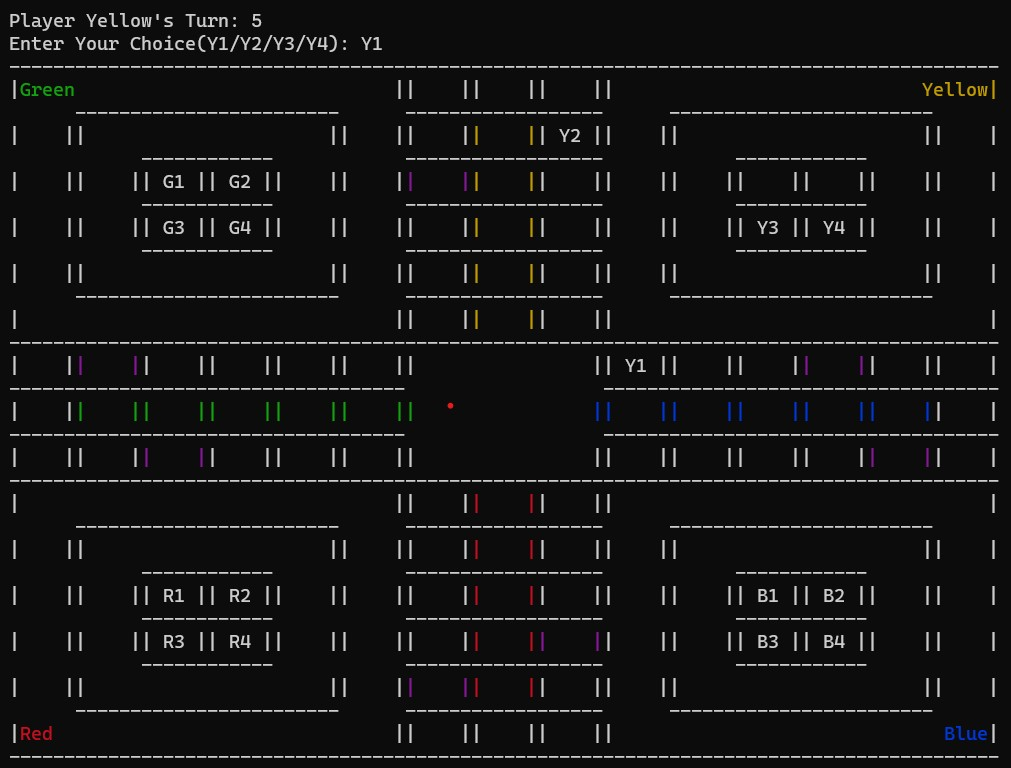
* When this game started, The Ludo Board be like following this picture.



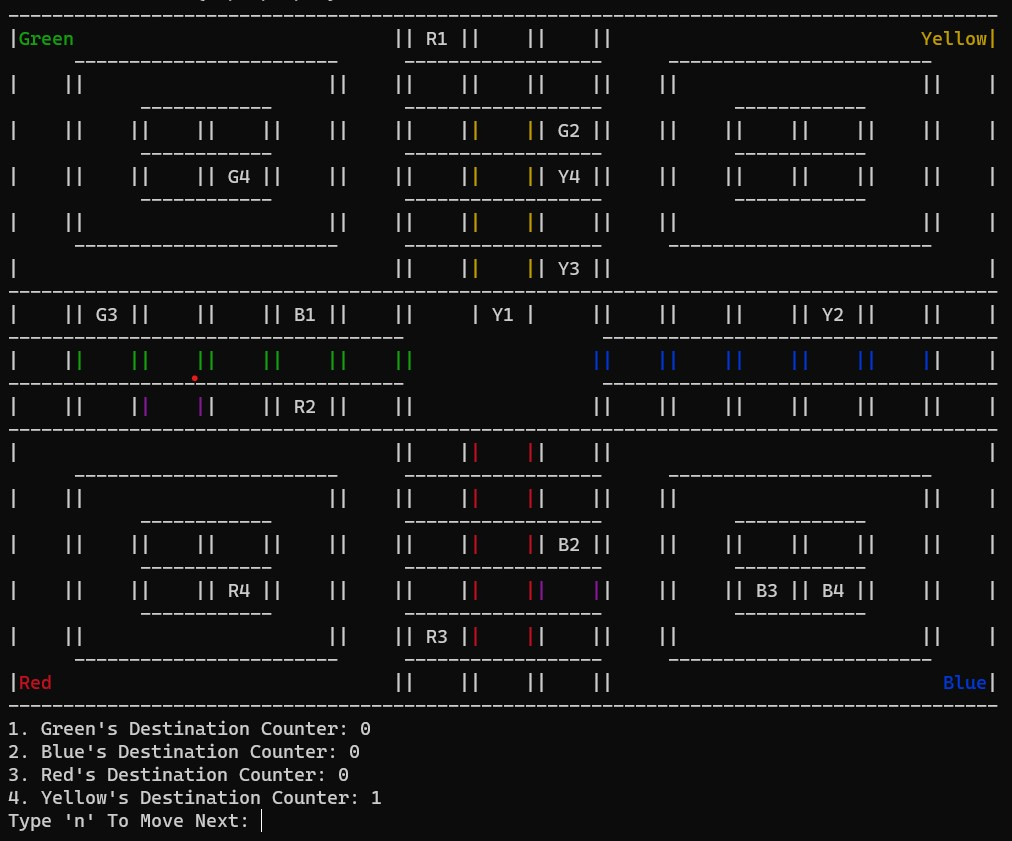
* When dice rolled for six, Then we have two option, such as the token is in home position or not. If the token is in the home position, then move for outside of the home position. One the other hand, if token is outside of the home position, then move it.



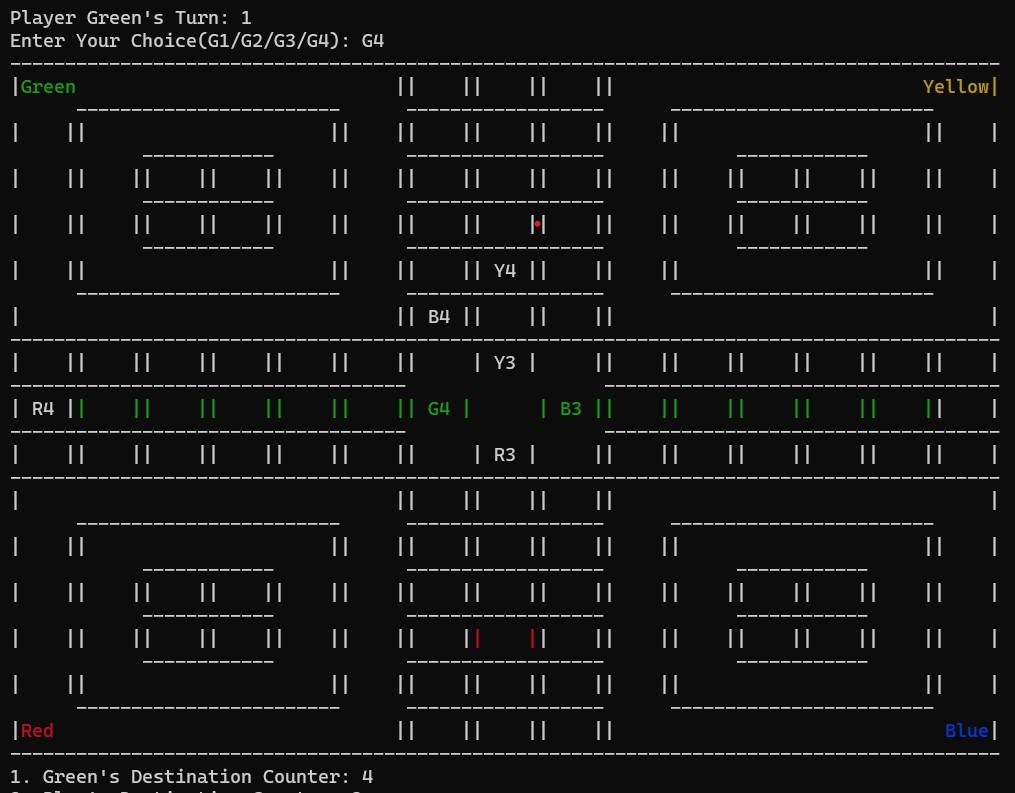
* In this board, Yellow token has rolled 2 six and five. In this moment, Y2 token is present in the initial block and Y1 token has rolled into his position that has length of 5 block.



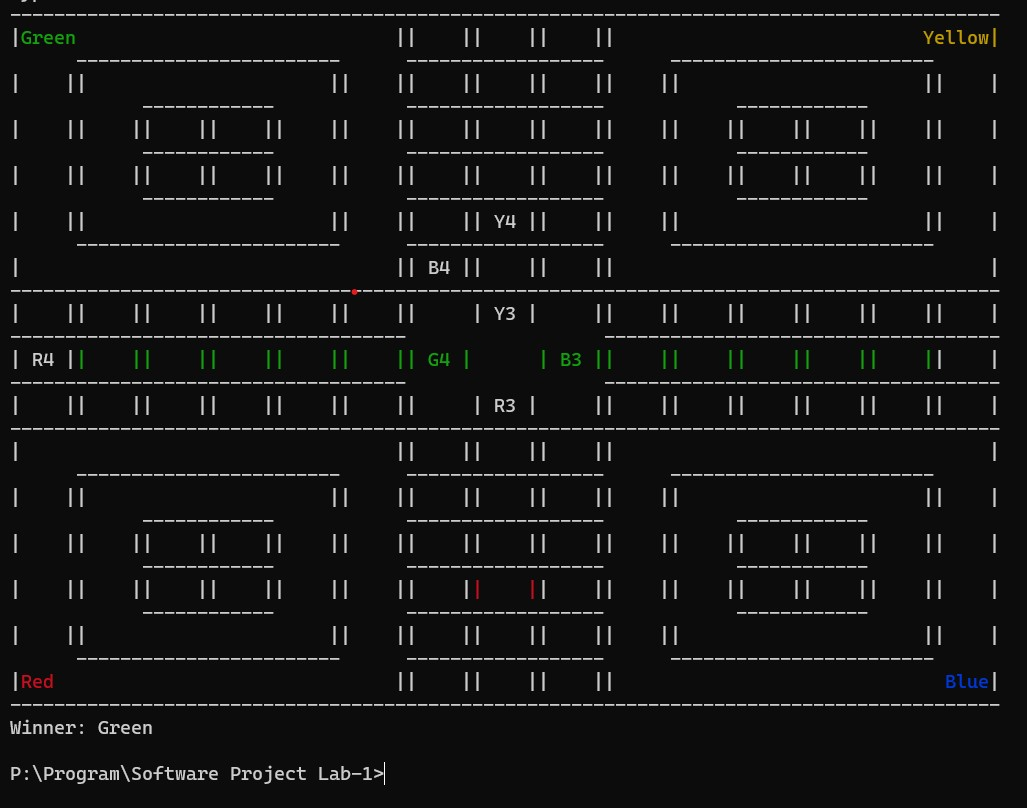
* Continue this game. When a token is enrolled for his last destination, then count 1 for this token. In this pic, Y1 token is enrolled for its last destination, so Yellow destination counter is 1.



* In this picture, Green token has enrolled all the token for its destination and Green destination counter is 4. When destination counter is 4, then the game is over and this token which has the destination counter is 4, then the player is winner.



* In this picture, Green token has destination counter 4. In this moment, the winner is Green.



Source Code Documentation

Here are the 4 class description in Ludo game:

|  |  |
| --- | --- |
| LoduGame | This class **LudoGame** encapsulates the functionality needed to play a simplified version of the Ludo board game. It handles player interactions, token movements, and game state management, providing a basic Ludo gaming experience within the console. |
| LudoBoard | The **LudoBoard** class appears to be a crucial part of implementing the Ludo game. It provides methods for initializing, updating, and displaying the game board, which is essential for the gameplay experience. The class uses a combination of 2D arrays to represent the game board's structure and tokens' positions, making it easier to manage and visualize the game state. |
| Player | The **Player** class serves as a container for organizing player-related data. It allows you to group a player's name and their tokens together for easier management and tracking during the game. |
| Token | These methods appear to be part of the logic for moving and managing tokens in your Ludo game. To use this class effectively, you would typically create instances of the **Token** class for each token in the game and call these methods as players take their turns and interact with the game board. |

Explanation of all methods in these classes:

**1.Player class:**

* **Player** class represents a player in the Ludo game.
  + **Player()**: Constructor method that initializes a player with a given name.

**2. Token class:**

* **Token** class represents individual tokens used by players in the Ludo game.
  + **isOutOfHome()**: Checks if a token is out of its home.
  + **takeFromHome()**: Moves a token from home to the game board.
  + **moveToken()**: Moves a token on the game board based on dice roll and handles interactions with other tokens.

**3. LudoBoard class:**

* + **LudoBoard** class represents the game board in the Ludo game.
  + **initializeHome()**: Initializes the home positions of tokens on the board.
  + **removeToken()**: Removes a token from a specific position on the board.
  + **printBoard()**: Prints the game board.
  + **drawBorder()**: Draws the border of the game board.

**4. LudoGame class:**

* + **LudoGame** class represents the main Ludo game.
  + **rollDice()**: Simulates rolling a dice and returns the result.
  + **playGame()**: Manages the main game loop, including player turns and interactions.
  + **main()**: The main method to start the Ludo game.

"Play to win, but learn to lose gracefully."