A

Mini Project Report

On

‘LudoGame’

Presented by

Chandrashekhar Kalyan

SY_CSE [A]

Second Year Engineering

2024-2025

Guided By

Ms Nitu.L.Pariyal

(Department of Computer Science and Engineering)

Submitted to

MGM’s College of Engineering, Nanded

Under

Dr. Babasaheb Ambedkar Technological University, Lonere

Certificate

This is to certify that the report entitled

Ludo Game

Submitted By

Kalyan Chandrashekhar

in satisfactory manner as a partial fulfillment of

SY_CSE[A] in Second Year Engineering

To

MGM's College of Engineering, Nanded

Under

Dr. Babasaheb Ambedkar Technological University, Lonere

has been carried out under my guidance,

Ms.Nitu.L.Pariyal

Project Guided

Dr.A.M.Rajurkar HOD

Department of Computer Science and Engineering

Dr. Lathkar G. S.

Director

MGM's College of Engineering, Nanded

ACKNOWLEDGEMENT

We are greatly indebted to our seminar guided by Ms.Nitu.L.Pariyal for her able guidance throughout this work. It has been an altogether different experience to work with her and we would like to thank her for her help, suggestions and numerous discussions.
We gladly take this opportunity to thank Dr.A.M.Rajurkar. (Head of Computer Science & Engineering, MGM's College of Engineering, Nanded).

We are heartily thankful to Dr. Lathkar G. S. (Director, MGM's College of Engineering, Nanded) for providing facility during progress of Mini project; also for her kindly help, guidance and inspiration.

Last but not least we are also thankful to all those who help directly or in directly to develop this seminar and complete it successfully.

With Deep Reverence,

Kalyan Chandrashekhar [152]

[SY-CSE-A]

ABSTRACT

Ludo Game

Ludo is a classic board game for 2-4 players that involves strategy, luck, and competition. It originated from the ancient Indian game "Pachisi" and is played on a square board divided into four colored sections (red, blue, green, and yellow). Each player controls four tokens, which must be moved from the starting point, around the board, and into the player's home column. Movement is determined by rolling a single die, and the goal is to be the first player to get all four tokens to the center of the board. Tokens move clockwise around the track, and landing on an opponent's token sends it back to its starting point. Players must roll a six to move a token out of the starting area. The game requires a balance of offensive and defensive strategies, as players decide whether to advance their tokens or block opponents. Ludo's simple rules combined with elements of chance make it a popular and timeless game for all ages.

Kalyan Chandrashekhar [152]

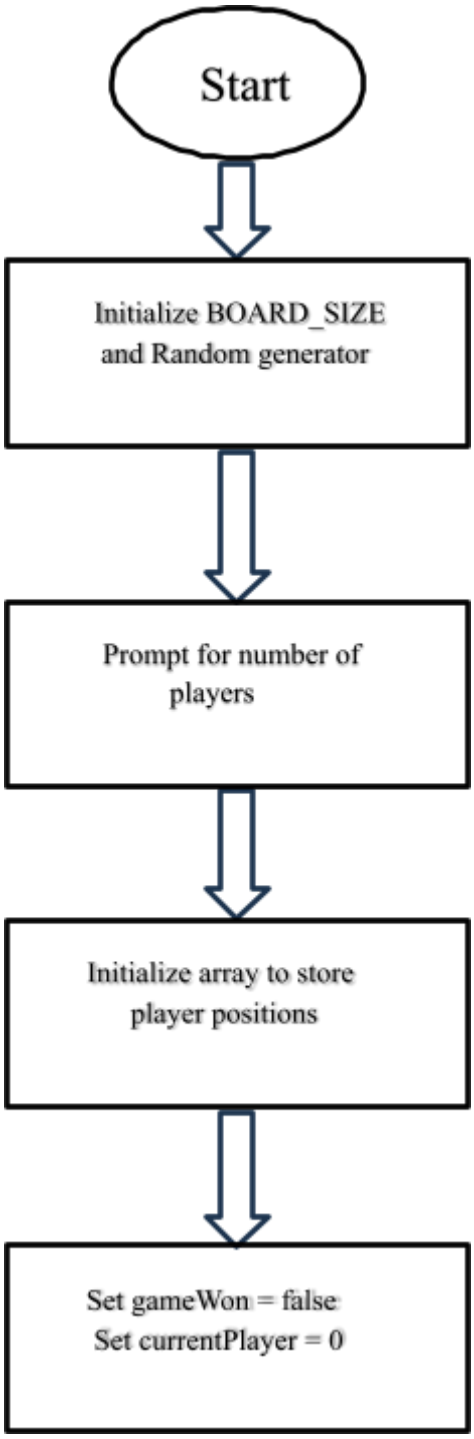
SY-A[CSE]

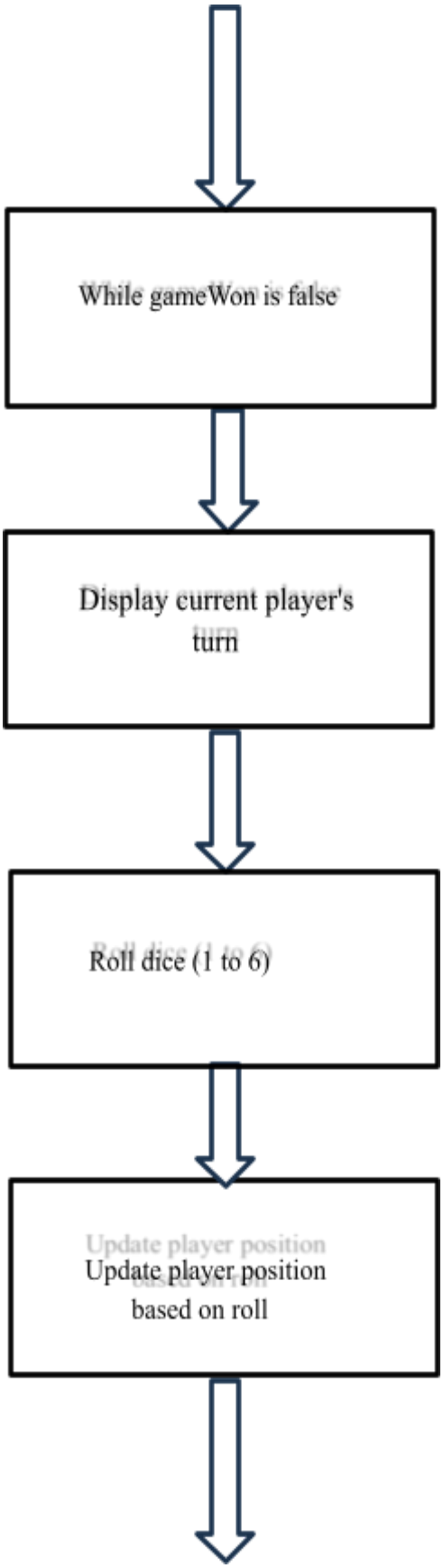
CONTENT

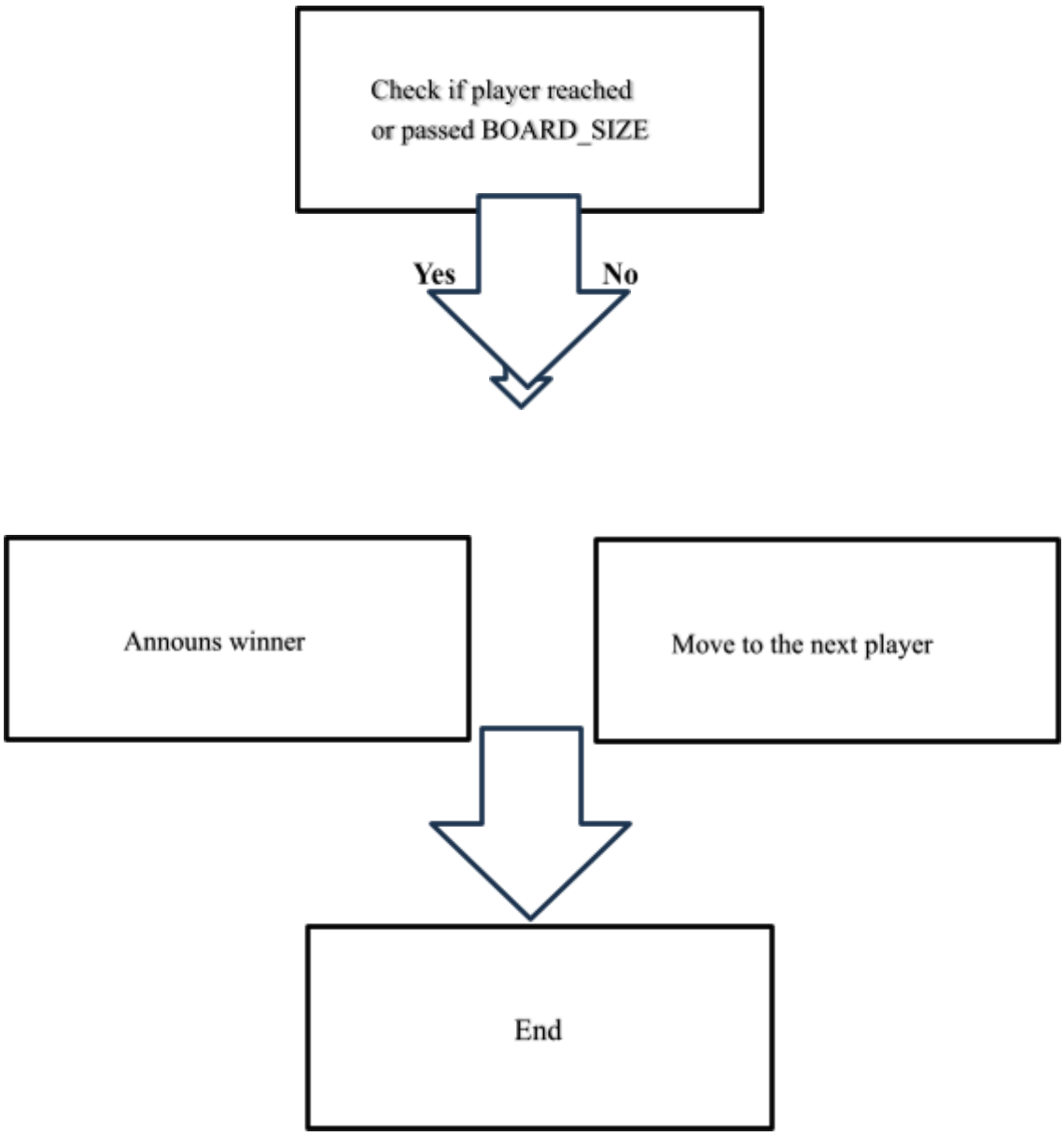
Sr.No.	TITLE	Page No.
	ACKNOWLEDGEMENT	I
	ABSTRACT	II
	CONTENT	III
1	**UML Diagram**	4
2	**Flowchart**	5
3	**Code Of LudoGame**	8
4	**OUTPUT**	9
5	**Explanation Of Code**	11
	CONCLUSIONS	14

1. UML Diagram

2. **Flowchart**


Start
Initialize BOARD_SIZE and Random generator
Prompt for number of players
Initialize array to store player positions
Set gameWon = false
Set currentPlayer = 0

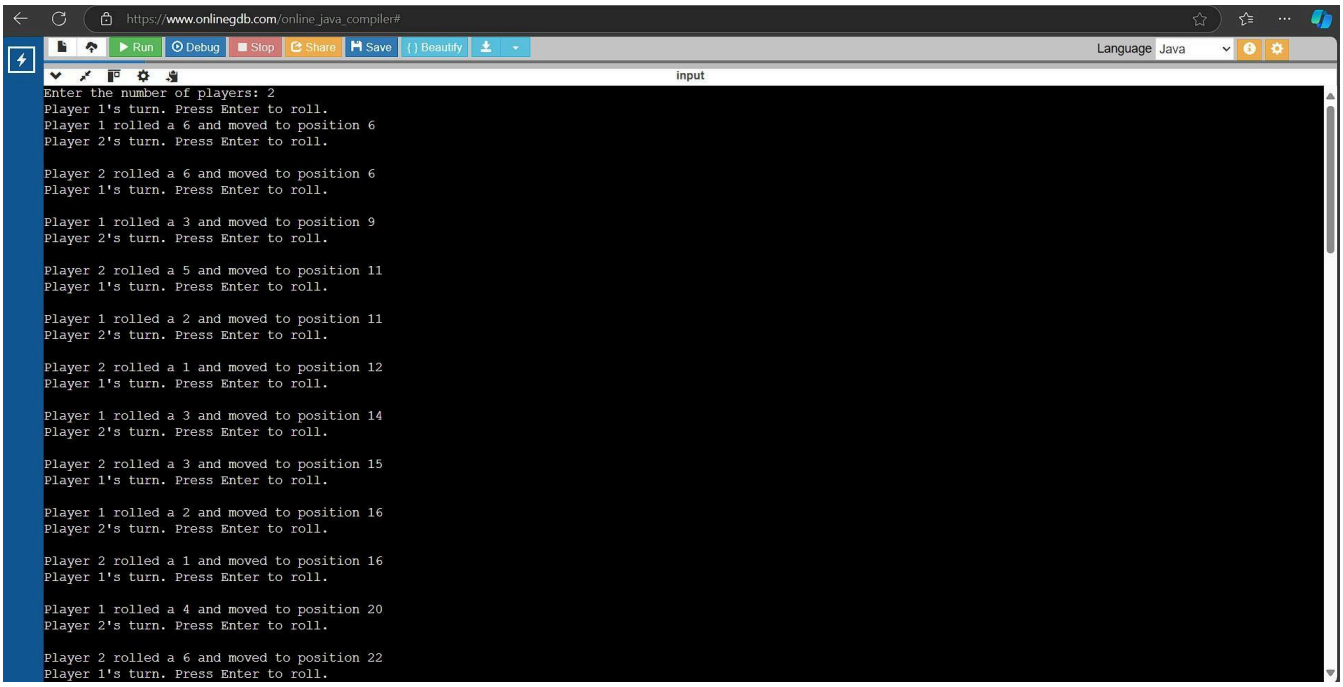

While gameWon is false
Display current player's turn
Roll dice (1 to 6)
Update player position based on roll

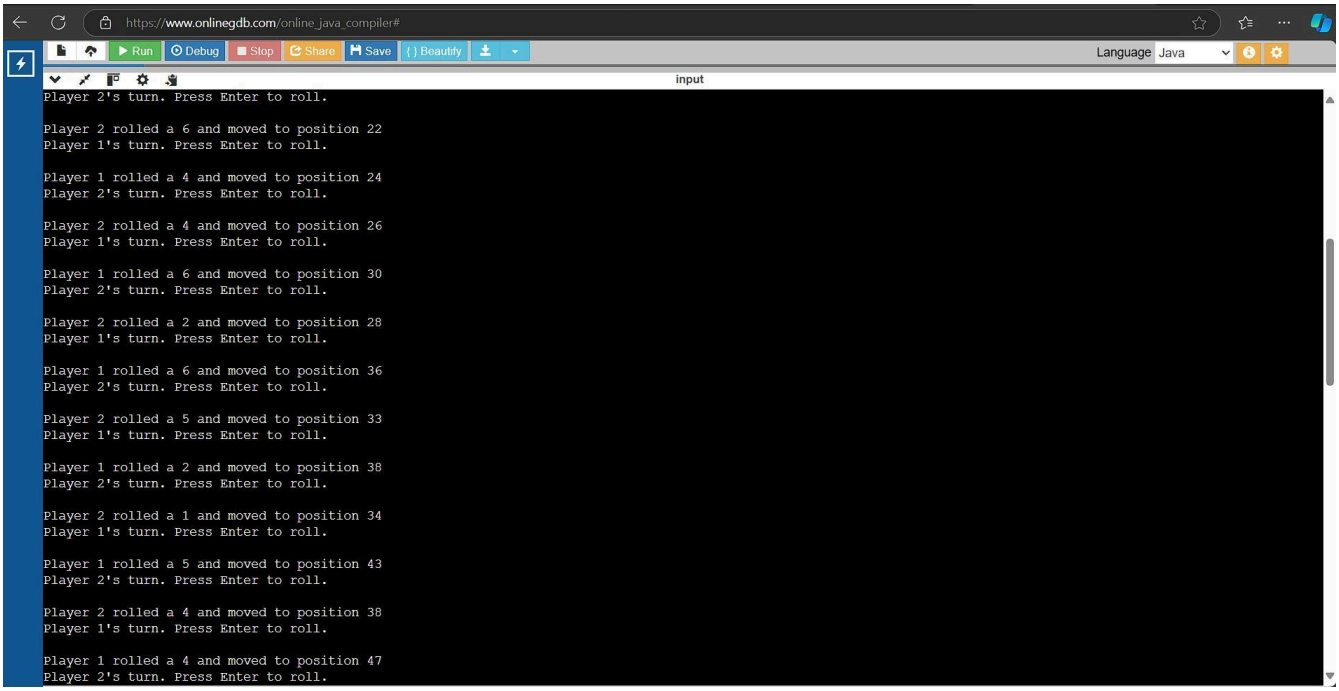

Check if player reached
or passed BOARD_SIZE
Yes
No
Announs winner
Move to the next player
End

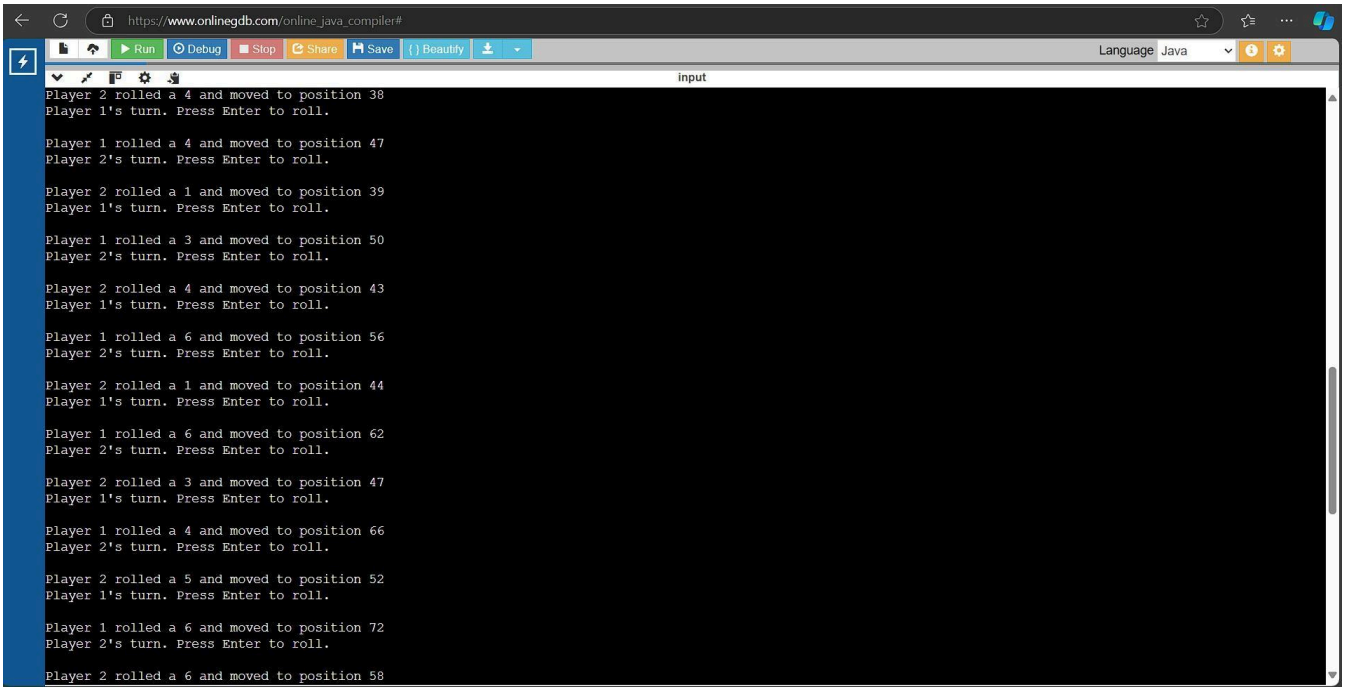
3. Code of LudoGame

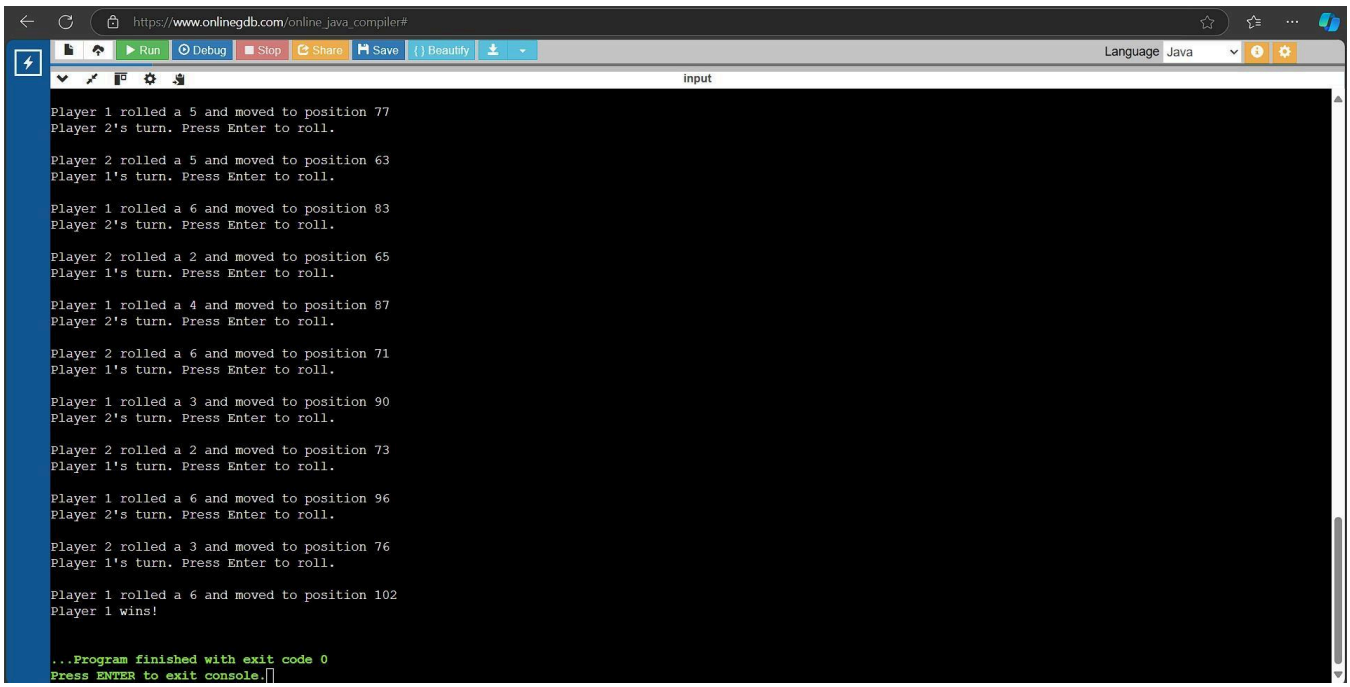
```
import java.util.Random;
import java.util.Scanner;

public class LudoGame {
   private static final int BOARD_SIZE = 100; // Board size
   private static Random random = new Random();

   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);

      System.out.print("Enter the number of players: ");
      int numPlayers = scanner.nextInt();
      int[] positions = new int[numPlayers]; // Array to store player positions

      boolean gameWon = false;
      int currentPlayer = 0;

      while (!gameWon) {
         System.out.println("Player " + (currentPlayer + 1) + "'s turn. Press Enter to roll.");
         scanner.nextLine(); // Wait for Enter key press

         int roll = random.nextInt(6) + 1; // Roll between 1 and 6
         positions[currentPlayer] += roll; // Update player position

         System.out.println("Player " + (currentPlayer + 1) + " rolled a " + roll + " and moved to position "
+ positions[currentPlayer]);

         if (positions[currentPlayer] >= BOARD_SIZE) {
            System.out.println("Player " + (currentPlayer + 1) + " wins!");
            gameWon = true;
         } else {
            currentPlayer = (currentPlayer + 1) % numPlayers; // Move to the next player
         }
      }

      scanner.close();
   }
}
```

4. OUTPUT


https://www.onlinegdb.com/online_java_compiler#
Run
Debug
Stop
Share
Save
{ } Beautify
Language
Java
input
Enter the number of players: 2
Player 1's turn. Press Enter to roll.
Player 1 rolled a 6 and moved to position 6
Player 2's turn. Press Enter to roll.
Player 2 rolled a 6 and moved to position 6
Player 1's turn. Press Enter to roll.
Player 1 rolled a 3 and moved to position 9
Player 2's turn. Press Enter to roll.
Player 2 rolled a 5 and moved to position 11
Player 1's turn. Press Enter to roll.
Player 1 rolled a 2 and moved to position 11
Player 2's turn. Press Enter to roll.
Player 2 rolled a 1 and moved to position 12
Player 1's turn. Press Enter to roll.
Player 1 rolled a 3 and moved to position 14
Player 2's turn. Press Enter to roll.
Player 2 rolled a 3 and moved to position 15
Player 1's turn. Press Enter to roll.
Player 1 rolled a 2 and moved to position 16
Player 2's turn. Press Enter to roll.
Player 2 rolled a 1 and moved to position 16
Player 1's turn. Press Enter to roll.
Player 1 rolled a 4 and moved to position 20
Player 2's turn. Press Enter to roll.
Player 2 rolled a 6 and moved to position 22
Player 1's turn. Press Enter to roll.


https://www.onlinegdb.com/online_java_compiler#
Run
Debug
Stop
Share
Save
{ } Beautify
Language
Java
input
Player 2's turn. Press Enter to roll.
Player 2 rolled a 6 and moved to position 22
Player 1's turn. Press Enter to roll.
Player 1 rolled a 4 and moved to position 24
Player 2's turn. Press Enter to roll.
Player 2 rolled a 4 and moved to position 26
Player 1's turn. Press Enter to roll.
Player 1 rolled a 6 and moved to position 30
Player 2's turn. Press Enter to roll.
Player 2 rolled a 2 and moved to position 28
Player 1's turn. Press Enter to roll.
Player 1 rolled a 6 and moved to position 36
Player 2's turn. Press Enter to roll.
Player 2 rolled a 5 and moved to position 33
Player 1's turn. Press Enter to roll.
Player 1 rolled a 2 and moved to position 38
Player 2's turn. Press Enter to roll.
Player 2 rolled a 1 and moved to position 34
Player 1's turn. Press Enter to roll.
Player 1 rolled a 5 and moved to position 43
Player 2's turn. Press Enter to roll.
Player 2 rolled a 4 and moved to position 38
Player 1's turn. Press Enter to roll.
Player 1 rolled a 4 and moved to position 47
Player 2's turn. Press Enter to roll.


https://www.onlinegdb.com/online_java_compiler#
Run
Debug
Stop
Share
Save
{ } Beautify
Language
Java
input
Player 2 rolled a 4 and moved to position 38
Player 1's turn. Press Enter to roll.
Player 1 rolled a 4 and moved to position 47
Player 2's turn. Press Enter to roll.
Player 2 rolled a 1 and moved to position 39
Player 1's turn. Press Enter to roll.
Player 1 rolled a 3 and moved to position 50
Player 2's turn. Press Enter to roll.
Player 2 rolled a 4 and moved to position 43
Player 1's turn. Press Enter to roll.
Player 1 rolled a 6 and moved to position 56
Player 2's turn. Press Enter to roll.
Player 2 rolled a 1 and moved to position 44
Player 1's turn. Press Enter to roll.
Player 1 rolled a 6 and moved to position 62
Player 2's turn. Press Enter to roll.
Player 2 rolled a 3 and moved to position 47
Player 1's turn. Press Enter to roll.
Player 1 rolled a 4 and moved to position 66
Player 2's turn. Press Enter to roll.
Player 2 rolled a 5 and moved to position 52
Player 1's turn. Press Enter to roll.
Player 1 rolled a 6 and moved to position 72
Player 2's turn. Press Enter to roll.
Player 2 rolled a 6 and moved to position 58


https://www.onlinegdb.com/online_java_compiler#
Run
Debug
Stop
Share
Save
{ } Beautify
Language
Java
input
Player 1 rolled a 5 and moved to position 77
Player 2's turn. Press Enter to roll.
Player 2 rolled a 5 and moved to position 63
Player 1's turn. Press Enter to roll.
Player 1 rolled a 6 and moved to position 83
Player 2's turn. Press Enter to roll.
Player 2 rolled a 2 and moved to position 65
Player 1's turn. Press Enter to roll.
Player 1 rolled a 4 and moved to position 87
Player 2's turn. Press Enter to roll.
Player 2 rolled a 6 and moved to position 71
Player 1's turn. Press Enter to roll.
Player 1 rolled a 3 and moved to position 90
Player 2's turn. Press Enter to roll.
Player 2 rolled a 2 and moved to position 73
Player 1's turn. Press Enter to roll.
Player 1 rolled a 6 and moved to position 96
Player 2's turn. Press Enter to roll.
Player 2 rolled a 3 and moved to position 76
Player 1's turn. Press Enter to roll.
Player 1 rolled a 6 and moved to position 102
Player 1 wins!
...Program finished with exit code 0
Press ENTER to exit console.

5.Explanation of code

This Java program is a simple simulation of a Ludo-like game where players take turns rolling a die and moving along a board. The game continues until one player reaches or exceeds the final board position, which is set at 100. Here's a detailed breakdown:

1. Class Declaration and

Constants public class LudoGame

{

```
private static final int BOARD_SIZE = 100; // Board size
private static Random random = new Random();
```

LudoGame: The main class representing the game.
BOARD_SIZE: A constant that defines the size of the board (the finish line is at position 100).
random: An instance of Random, used to simulate the die roll for each player's turn.

2. Main Method

```
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
```

This is the entry point of the program, where the game starts.
scanner: A Scanner object to get user input.

3. Setting Up Players

```
System.out.print("Enter the number of players: ");
int numPlayers = scanner.nextInt();
int[] positions = new int[numPlayers]; // Array to store player positions
```

The program prompts the user to input the number of players.
numPlayers: Stores the number of players.
positions: An array where each index represents a player's position on the board. All positions start at 0.

4. Game Loop

```
boolean gameWon = false;
int currentPlayer = 0;

while (!gameWon) {
```

gameWon: A boolean flag indicating if someone has won the game. Initially set to false.
currentPlayer: Tracks which player's turn it is. Initialized to 0 (first player).
The while loop continues until gameWon is set to true.

5. Player's Turn

```
System.out.println("Player " + (currentPlayer + 1) + "'s turn. Press Enter to roll.");
scanner.nextLine(); // Wait for Enter key press

int roll = random.nextInt(6) + 1; // Roll between 1 and 6
positions[currentPlayer] += roll; // Update player position

System.out.println("Player " + (currentPlayer + 1) + " rolled a " + roll + " and moved to position " +
positions[currentPlayer]);
```

The current player is prompted to press Enter to "roll" the die.
roll: Simulates a die roll using random.nextInt(6) + 1, which produces a number between 1 and 6.
positions[currentPlayer] += roll: Adds the roll result to the player's current position.
The program outputs the roll and the updated position of the player.

6. Checking for a Win

```
if (positions[currentPlayer] >= BOARD_SIZE) {
    System.out.println("Player " + (currentPlayer + 1) + " wins!");
    gameWon = true;
} else {
    currentPlayer = (currentPlayer + 1) % numPlayers; // Move to the next player
}
```

If the player's position is >= BOARD_SIZE, they have won. A message is displayed, and gameWon is set to true, which ends the loop.
If the player hasn't won, currentPlayer = (currentPlayer + 1) % numPlayers moves to the next player. The % numPlayers ensures that if the last player completes their turn, it wraps back to the first player.

7. Ending the

Game

```
scanner.close();
```

Finally, the Scanner is closed to free up resources.

CONCLUSIONS

Ludo is a simple yet engaging game that blends strategy, luck, and competition, making it enjoyable for players of all ages. Its straightforward rules make it easy to learn, while the element of chance with dice rolls adds unpredictability and excitement. The game encourages strategic thinking, as players must decide when to advance, attack, or defend. Despite its roots in ancient Indian gaming traditions, Ludo remains relevant and widely played today, demonstrating its timeless appeal. It serves as both a fun pastime and a social activity, promoting friendly competition and interaction among players.