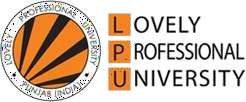
# Mini Project Ludo

Prathipati Venkatesh 12110145

# Submitted to

**Asst. Professor,**

# School of Computer Science and Engineering



LOVELY PROFESSIONAL UNIVERSITY PHAGWARA, PUNJAB

# Abstract

The Ludo game is a classic board game that can be implemented in Java programming language. In this game, each player has four tokens that they need to move from their starting point to the finish line by rolling a dice. The game can be played by two to four players, and the player who gets all their tokens to the finish line first wins.

The Java implementation of the Ludo game can include features such as a graphical user interface (GUI) with a game board and dice roll animation, token movement and collision detection, turn-based gameplay, and win/lose conditions. The game can also include additional features such as player scorekeeping, sound effects, and AI opponents.

To implement the Ludo game in Java, object-oriented programming principles can be used to create classes for the game board, tokens, dice, players, and game logic. The game can be developed using Java Swing for the GUI and Java's built-in random number generator for the dice roll. Proper exception handling and error checking can also be implemented to ensure the game runs smoothly and without errors.

Overall, the Java implementation of the Ludo game can be a fun and challenging project for Java developers looking to practice their programming skills.

# Existing System

**There are several existing implementations of the Ludo game in Java that can be used as a reference for developers looking to create their own version of the game. Here are a few examples:**

**LudoGame-Java: This is an open-source implementation of the Ludo game in Java that is available on GitHub. It includes a GUI with a game board, token movement, dice roll animation, and basic game logic.**

**Ludo Game in Java Swing: This is a tutorial on how to create a Ludo game using Java Swing for the GUI. The tutorial includes step-by-step instructions and code examples for creating the game board, tokens, and dice roll.**

**Ludo Game in Java: This is a tutorial on how to create a Ludo game in Java using object- oriented programming principles. The tutorial covers creating classes for the game board, tokens, and players, as well as implementing turn-based gameplay and win/lose conditions.**

**Ludo Game in Java using Socket Programming: This is a tutorial on how to create a multiplayer Ludo game in Java using socket programming. The tutorial includes instructions and code examples for setting up a server and client, as well as implementing game logic and communication between players.**

**These existing systems can provide valuable insights into the implementation of the Ludo game in Java and can be used as a starting point for developers looking to create their own version of the game.**

# Code

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

class Game {

// if Player 1

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

int Score1=0;

// it shows Score of player 1 int Score2=0;

// it shows Score of player 2

int turn ;

int player1 ; int player2 ;

// senitall loop

do {

System.out.println("\n Enter 1 to Continue game and\n 0 for End Game "); turn =input.nextInt();

System.out.println(" ");

if (turn == 1) {

player1 = random.nextInt(7); System.out.println(" Player 1 : Turn "+player1);

player2 = random.nextInt(7); System.out.println(" Player 2 : Turn "+player2);

// Player1 data >

if (player1==6)

{

Score1 += player1; System.out.println("This Six by Player 1 ");

System.out.println(" >");

player1 = random.nextInt(7);

System.out.println(" Player 1 : 2nd Turn "+player1); if(player1==6)

{

System.out.println("This @nd six by plyer 1");

System.out.println(" >"); Score1 += player1;

player1 = random.nextInt(7);

System.out.println(" Player 1 : 3nd Turn "+player1); if (player1==6)

{

Score1=0;

}

}

}

else {

Score1 += player1;

if (Score1 == 33 || Score1 == 66 || Score1 == 99) {

Score1 = 0;

System.out.println("oho! Snake Mouth Player 1 > ");

System.out.println("Now youR Score is Zero Try Again Good Luck \n

");

System.out.println(" >");

}

if (Score1 >= 100) {

System.out.println("Palayer 1 win the game");

System.out.println(" >"); break;

}

}

// Player2 data >

if(player2==6)

{

Score2 += player2;

System.out.println("This Six by Player 2 ");

System.out.println(" >"); player2 = random.nextInt(7);

System.out.println(" Player 1 : 2nd Turn "+player2); if(player2==6)

{

Score2 += player2;

player2 = random.nextInt(7);

System.out.println("This Six by Player 2 ");

System.out.println(" >"); System.out.println(" Player 1 : 3nd Turn "+player2);

if (player2==6)

{

Score2=0;

}

}

}

else {

Score2 += player2;

if(Score2==33|| Score2==66|| Score2==99){

Score2=0;

System.out.println("oho! Snake Mouth Player 2 >");

System.out.println("Now youR Score is Zero Try Again Good Luck \n

");

System.out.println(" >");

}

if (Score2 >= 100) {

System.out.println("Palayer 2 win the game");

System.out.println(" >"); break;

}

System.out.println("SCORE OF Player 1 : "+Score1);

System.out.println("SCORE OF Player 2 : "+Score2);

}

// end of if

}

else {

break;

}

//end of Do

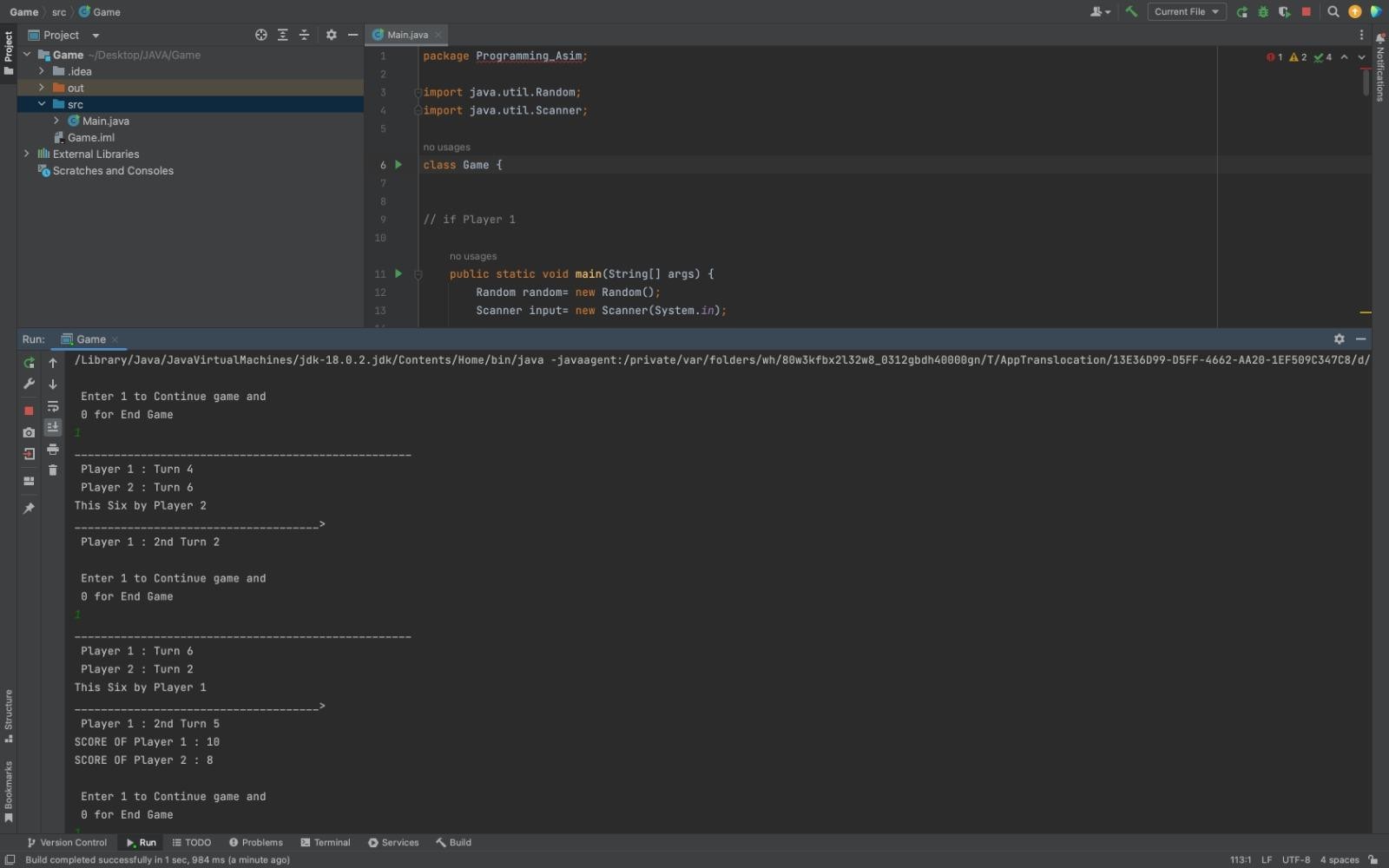
}

while (1==1);

}

}

# Snapshot



**Proposed System**

A proposed system for the Ludo game in Java could include the following features:

Graphical User Interface (GUI): The game can have a visually appealing GUI with a game board and dice roll animation. The board can be designed using Java Swing or JavaFX, with options for customizing the color and design of the board.

Token Movement: The game can have token movement functionality with collision detection. The tokens can move based on the number rolled on the dice, and the collision detection can ensure that only one token occupies a given space on the board at a time.

Multiplayer and Singleplayer Modes: The game can have both singleplayer and multiplayer modes. In the singleplayer mode, the player can play against an AI opponent, while in multiplayer mode, the player can play against other players online.

Turn-Based Gameplay: The game can have turn-based gameplay, where each player takes turns rolling the dice and moving their tokens. The game can have a timer to limit the duration of each turn.

Win/Lose Conditions: The game can have win/lose conditions where the player who gets all their tokens to the finish line first wins. The game can also have a tie-breaker mechanism in case of a tie.

Sound Effects: The game can have sound effects for dice rolling, token movement, and win/lose conditions to enhance the gaming experience.

Saving and Loading: The game can have functionality to save and load the game, allowing players to resume their game later.

To implement this proposed system, object-oriented programming principles can be used to create classes for the game board, tokens, dice, players, and game logic. The game can be developed using Java Swing or JavaFX for the GUI, and proper exception handling and error checking can be implemented to ensure the game runs smoothly and without errors. Overall, this proposed system can provide a fun and engaging gaming experience for players of all ages.

# Advantages of ludo game

Social Interaction: Ludo is a multiplayer game, which allows players to interact and socialize with each other. It provides an opportunity for friends and family to spend time together, bond, and have fun.

Strategic Thinking: Ludo involves strategic thinking and planning, as players must decide which tokens to move and which moves to make to gain an advantage over their opponents. It helps to develop analytical and problem-solving skills.

Improves Concentration: Playing Ludo requires focus and concentration, which helps to improve attention span and memory retention.

Enhances Mathematical Skills: Ludo involves rolling a dice, which helps to improve mathematical skills such as counting, addition, and probability.

Provides Entertainment: Ludo is a fun and entertaining game that can be enjoyed by players of all ages. It provides a break from the monotony of daily life and helps to relieve stress and anxiety.

Enhances Sportsmanship: Playing Ludo teaches good sportsmanship, as players learn to accept defeat graciously and appreciate the efforts of their opponents. It helps to develop positive character traits such as humility and respect.

Overall, the Ludo game provides several advantages that can contribute to personal growth and development. It helps to improve social interaction, strategic thinking, concentration, mathematical skills, and sportsmanship, while also providing entertainment and enjoyment.

# System Requirement of ludo game

The system requirements for a Ludo game in Java will depend on the complexity and features of the game. Here are the general requirements for a basic Ludo game:

Operating System: The game can run on any operating system that supports Java, such as Windows, macOS, and Linux.

Java Runtime Environment (JRE): The system must have the latest version of JRE installed to run the game.

Processor: The game can run on a processor with a clock speed of 1 GHz or higher. RAM: The system should have at least 512 MB of RAM to run the game smoothly.

Graphics Card: The system should have a graphics card that supports OpenGL 2.0 or higher to provide smooth animations and visuals.

Display: The game requires a minimum display resolution of 1024x768 to ensure the game elements are displayed correctly.

Input Devices: The game requires input devices such as a keyboard and mouse to control the gameplay.

Note that these requirements may vary depending on the specific implementation of the Ludo game and the additional features included. Developers should optimize the game to run efficiently on different systems and ensure that it provides a smooth and enjoyable gaming experience for all users.

# Conclusion

In conclusion, a Ludo game implemented in Java can be a fun and entertaining way to pass the time. It provides several advantages such as social interaction, strategic thinking, concentration, mathematical skills, and sportsmanship. The proposed system can include features such as a graphical user interface, token movement, multiplayer and singleplayer modes, turn-based gameplay, win/lose conditions, sound effects, and saving and loading functionality. To implement the system, object-oriented programming principles can be used, and Java Swing or JavaFX can be used for the graphical user interface. Overall, the Ludo game provides an enjoyable gaming experience for players of all ages, while also promoting personal growth and development.

a Ludo game developed in Java can provide a fun and engaging gaming experience for players of all ages. With its strategic gameplay, multiplayer options, and simple rules, Ludo is a game that can be enjoyed by anyone. The proposed system for the Ludo game in Java includes features such as a graphical user interface, token movement, turn-based gameplay, win/lose conditions, sound effects, and saving/loading functionality. By using object-oriented programming principles and proper exception handling, developers can create a game that runs smoothly and without errors. Additionally, playing the Ludo game has several advantages, including improving social interaction, strategic thinking, concentration, mathematical skills, and sportsmanship, while also providing entertainment and enjoyment. Overall, a well-designed Ludo game in Java can be an excellent addition to any game library and provide hours of fun for players.

**Future Enhancement**

There are several potential enhancements that could be made to the Ludo game in the future, including:

Online Multiplayer: Adding online multiplayer functionality to the game could allow players from all over the world to compete against each other, increasing the game's replayability and entertainment value.

Artificial Intelligence (AI) Opponents: Including AI opponents with different levels of difficulty could provide a challenge for players who prefer to play alone.

Customizable Boards and Tokens: Allowing players to customize the game boards and tokens could provide a more personalized experience and make the game more visually appealing.

Power-ups and Special Moves: Adding power-ups and special moves to the game could increase the strategic depth of the gameplay and provide more opportunities for players to gain an advantage over their opponents.

Quests and Challenges: Including quests and challenges within the game could provide players with additional goals and objectives to complete, increasing the game's longevity and replayability.

Mobile App: Developing a mobile app version of the game could allow players to access the game on the go and increase the game's accessibility.

Multi-Language Support: Adding multi-language support to the game could make it accessible to players who speak different languages, increasing the game's audience and popularity.

These enhancements could make the Ludo game more engaging and enjoyable for players and increase its appeal to a wider audience.

# References

1. **"Killer Game Programming in Java" by Andrew Davison. This book covers various aspects of game development in Java, including 2D and 3D graphics, animation, sound, input, networking, and AI. It includes code examples and practical advice for building games, making it a useful resource for developing a Ludo game.**
2. **"Java Game Programming for Beginners" by David Brackeen. This book provides an introduction to game development in Java, covering topics such as graphics, animation, input, sound, and game design. It includes code examples and step-by-step tutorials that can help beginners get started with game development.**
3. **"Beginning Java Game Development with LibGDX" by Lee Stemkoski. This book covers game development using the LibGDX framework, which is a popular open-source game development platform for Java. It covers topics such as graphics, animation, input, sound, and physics, and includes code examples and exercises that can help developers build their own games.**
4. **"Game Development with Java" by James Wallace and Doug Lombardi. This book covers game development in Java, including graphics, animation, input, sound, and networking. It includes case studies and code examples that can help developers understand the process of building a game from start to finish.**