VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELAGAVI 590018

Project Report on

Snakes and Ladder Game

By

Prajwal Vasantha Kumar (1BM24CS210) Niraj V(1BM24CS192)

Pranav Hebbar K(1BM24CS214) Niraj(1BM24CS191)

Under the Guidance of

Mrs. Monisha H M

Assistant Professor, Department of CSE

BMS College of Engineering

Work carried out at

Department of Computer Science and Engineering

BMS College of Engineering

(Autonomous college under VTU)

P.O. Box No.: 1908, Bull Temple Road, Bangalore-560 019

2025-2026

BMS COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the OOPS with JAVA project titled Snakes and Ladder game has been carried out by Prajwal Vasantha Kumar (1BM24CS210), Niraj V (1BM24CS192), Pranav Hebbar K (1BM24CS214), Niraj (1BM24CS191) during the academic year 2025-2026.

Signature of the guide
Mrs. Monisha H M
Assistant Professor,
Department of Computer Science and Engineering
BMS College of Engineering, Bangalore

BMS COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We, Prajwal Vasantha Kumar (1BM24CS210), Niraj V (1BM24CS192), Pranav Hebbar K (1BM24CS214), Niraj (1BM24CS191), students of 3rd Semester, B.E, Department of Computer Science and Engineering, BMS College of Engineering, Bangalore, hereby declare that, this project work entitled Snakes and Ladder Game has been carried out by us under the guidance of Mrs. Monisha H M, Assistant Professor, Department of CSE, BMS College of Engineering, Bangalore during the academic semester Sep-Dec 2025. We also declare that to the best of our knowledge and belief, the project reported here is not from part of any other report by any other students.

Signature of the Candidates

Prajwal Vasantha Kumar (1BM24CS210)

Niraj V (1BM24CS192)

Pranav Hebbar K (1BM24CS214)

Niraj (1BM24CS191)

Table of contents

Problem Statement

Develop a modular version of 'Snakes and Ladders' using Object-Oriented Programming (OOP) principles in Java. The project aims to solve the problem of hard-coded game logic by creating distinct classes for Player, Board, Entity (Snakes/Ladders), and Game Engine. The system should support a dynamic number of players and allow for easily configurable board layouts, demonstrating effect

Introduction

This project is a Java-based implementation of the classic board game "Snakes and Ladders."

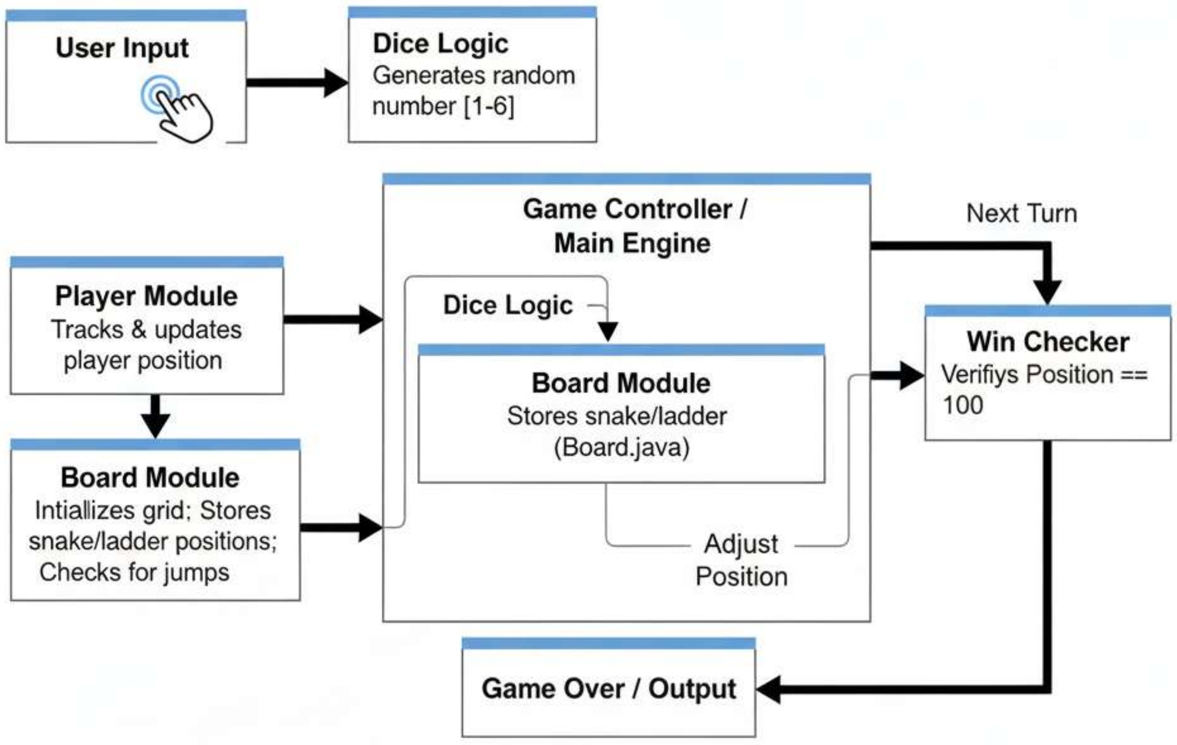
“Snakes and ladders “ is a board game for two or more players regarded today as a worldwide classic. The game originated in ancient India and was brought to United Kingdom in the 1890s.

It is played on a game board with numbered, gridded squares. A number of "ladders" and "snakes" are pictured on the board, each connecting two specific board squares. The object of the game is to navigate one's game piece, according to die rolls, from the start (bottom square) to the finish (top square), helped by climbing ladders but hindered by falling down snakes.

The game is a simple race based on sheer luck, and it is popular with young children. The historic version had its roots in morality lessons, on which a player's progression up the board represented a life journey complicated by virtues (ladders) and vices (snakes). The game is also sold under other names, such as the morality themed Chutes and Ladders, which was published by the Milton Bradley Company starting in 1943.

The game was popular in ancient India by the name Moksha Patam. It was also associated with traditional Hindu philosophy contrasting *karma* and kama, or destiny and desire. It emphasized destiny, as opposed to games such as pachisi, which focused on life as a mixture of skill (free will) and luck. The underlying ideals of the game inspired a version introduced in Victorian England in 1892. The game has also been interpreted and used as a tool for teaching the effects of good deeds versus bad.

Overview of the Project


User Input
Dice Logic
Generates random number [1-6]
Game Controller / Main Engine
Next Turn
Player Module
Tracks & updates player position
Dice Logic
Board Module
Stores snake/ladder (Board.java)
Win Checker
Verifiys Position == 100
Board Module
Intiallizes grid; Stores snake/ladder positions; Checks for jumps
Adjust Position
Game Over / Output

Tools Used and Their Purpose

Java 17 (JDK 17)

Purpose:
Java 17 is the core programming language used to write the application logic. It is a Long-Term Support (LTS) version, which ensures long-term stability, security updates, and improved performance. Java 17 also introduces modern language features such as enhanced switch expressions and overall performance optimizations.

Apache Maven

Purpose:
Apache Maven is a build automation and project management tool. It is responsible for:

- Managing project dependencies (defined in pom.xml)
- Compiling Java source code
- Packaging the application into a runnable build

Maven simplifies dependency handling and ensures consistent builds across different environments.

JavaFX (Version 21.0.1)

Purpose:
JavaFX is used to create the Graphical User Interface (GUI) for the application.

Key JavaFX Modules Used:

- javafx-controls:
  Provides UI components such as Button, Text, and other interactive controls.
- javafx-graphics:
  Handles rendering of the game board, snakes, ladders, player tokens, and animations such as dice rolls and token movement.
- javafx-fxml (dependency included):
  Supports separation of UI layout from application logic. Although the current implementation builds the UI programmatically, this module enables future scalability.

OOPS Concepts used

A. Encapsulation

Definition:
Encapsulation is the practice of bundling data (fields) and methods together within a class while restricting direct access to the internal data.

Implementation in the Project:

- Player Class:
  Fields such as name, id, currentPosition, and color are declared as private.
  Access to these fields is controlled through public getter and setter methods (e.g., getPosition() and setPosition()), ensuring data integrity.

- Board Class:
  Internal data structures such as the maps for snakes and ladders are kept private.
  Other classes interact with this data through controlled methods like getSnakeTail(int head), preventing unintended modifications.

B. Inheritance

Definition:
Inheritance allows a class to acquire properties and behaviors from another class, promoting code reuse.

Implementation in the Project:

- GameApp extends Application:
  The main application class inherits from the JavaFX Application class, gaining access to lifecycle methods required to launch and manage a windowed application.

- Dice extends VBox:
  The Dice class inherits from VBox, making it a self-contained UI component.
  This allows the dice to manage both its visual layout and rolling logic while being easily added to the scene graph.

C. Abstraction

Definition:
Abstraction hides complex implementation details and exposes only essential features to the user.

Implementation in the Project:

- Interface-Based Collections:
  Collections are declared using interfaces such as List<Tile> and Map<Integer, Integer> rather than concrete implementations like ArrayList or HashMap. This allows flexibility and loose coupling.

- Functional Abstraction in the Board Class:
  The Board class abstracts the complexity of drawing the grid, snakes, and ladders.
  The GameController simply calls methods like initializeSnakesAndLadders() without needing to understand the underlying mathematical calculations or drawing logic.

D. Polymorphism

Definition:
Polymorphism allows methods or objects to behave differently based on their implementation or context.

Implementation in the Project:

- Method Overriding:
  The start(Stage primaryStage) method in GameApp overrides the method from the Application class.
  When Application.launch() is called, JavaFX polymorphically executes the overridden start method.

- Event Handling with Lambdas:
  Lambda expressions such as
  rollButton.setOnAction(e -> { ... })
  treat blocks of behavior as objects, demonstrating functional polymorphism through event-driven programming.

Code/Implementation

Chess Board (Board.java)

```
package com.snakeandladder;

import javafx.scene.Group;

import javafx.scene.paint.Color;

import javafx.scene.shape.Line;

import javafx.scene.shape.StrokeLineCap;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;


public class Board {

    private Group boardGroup;

    private List<Tile> tiles;

    private final int ROWS = 10;

    private final int COLS = 10;

    private double tileSize = 60;

    private double width;

    private double height;

    private Map<Integer, Integer> snakes;

    private Map<Integer, Integer> ladders;

    private Group slElementsGroup;


    public Board() {
        boardGroup = new Group();

        tiles = new ArrayList<>();

        slElementsGroup = new Group();

        initializeSnakesAndLadders();

        createGrid();

        boardGroup.getChildren().addAll(slElementsGroup);

    }


    public Group getBoardGroup() {

        return boardGroup;

    }


    private void createGrid() {

        boardGroup.getChildren().removeAll(tiles);

        tiles.clear();

        for (int i = 0; i < ROWS * COLS; i++) {

            Tile tile = new Tile(i + 1, tileSize);

            tiles.add(tile);

            boardGroup.getChildren().add(0,  tile);

        }

        drawBoard();

    }


    private void initializeSnakesAndLadders() {

        snakes = new HashMap<>();

        ladders = new HashMap<>();
    snakes.put(16, 6);

    snakes.put(47, 26);

    snakes.put(49, 11);

    snakes.put(56, 53);

    snakes.put(62, 19);

    snakes.put(64, 60);

    snakes.put(87, 24);

    snakes.put(93, 73);

    snakes.put(95, 75);

    snakes.put(98, 78);

    ladders.put(1, 38);

    ladders.put(4, 14);

    ladders.put(9, 31);

    ladders.put(21, 42);

    ladders.put(28, 84);

    ladders.put(36, 44);

    ladders.put(51, 67);

    ladders.put(71, 91);

    ladders.put(80, 100);

}


private void drawBoard() {

    for (int i = 0; i < tiles.size(); i++) {

        int number = i + 1;

        Point p = getCoordinatesForNumber(number);

        Tile t = tiles.get(i);
        t.setTranslateX(p.x);
        t.setTranslateY(p.y);
        t.updateSize(tileSize);
    }
    drawSnakesAndLadders();
}

private void drawSnakesAndLadders() {
    slElementsGroup.getChildren().clear();
    ladders.forEach((start, end) -> {
        Point p1 = getCenterMoveCoordinates(start);
        Point p2 = getCenterMoveCoordinates(end);
        double dx = p2.x - p1.x;
        double dy = p2.y - p1.y;
        double length = Math.sqrt(dx * dx + dy * dy);
        double nx = -dy / length;
        double ny = dx / length;
        double width = 10;
        Line leftRail = new Line(p1.x - nx * width, p1.y - ny * width, p2.x - nx * width, p2.y - ny * width);
        Line rightRail = new Line(p1.x + nx * width, p1.y + ny * width, p2.x + nx * width, p2.y + ny * width);
        leftRail.setStroke(Color.DARKGREEN);
        rightRail.setStroke(Color.DARKGREEN);
        leftRail.setStrokeWidth(3);
        rightRail.setStrokeWidth(3);
        slElementsGroup.getChildren().addAll(leftRail, rightRail);
        int steps = (int) (length / 20);
        for (int i = 0; i <= steps; i++) {
            double t = (double) i / steps;
            double cx = p1.x + dx * t;
            double cy = p1.y + dy * t;
            Line rung = new Line(cx - nx * width, cy - ny * width, cx + nx * width, cy + ny * width);
            rung.setStroke(Color.DARKGREEN);
            rung.setStrokeWidth(2);
            slElementsGroup.getChildren().add(rung);
        }
    });
    snakes.forEach((start, end) -> {
        Point p1 = getCenterMoveCoordinates(start);
        Point p2 = getCenterMoveCoordinates(end);
        javafx.scene.shape.QuadCurve curve = new javafx.scene.shape.QuadCurve();
        curve.setStartX(p1.x);
        curve.setStartY(p1.y);
        curve.setEndX(p2.x);
        curve.setEndY(p2.y);
        double midX = (p1.x + p2.x) / 2;
        double midY = (p1.y + p2.y) / 2;
        double dx = p2.x - p1.x;
        double dy = p2.y - p1.y;
        curve.setControlX(midX + dy * 0.3);
        curve.setControlY(midY - dx * 0.3);
        curve.setStroke(Color.RED);
        curve.setStrokeWidth(4);
```

```
            curve.setFill(null);
            curve.setStrokeLineCap(StrokeLineCap.ROUND);
            javafx.scene.shape.Circle head = new javafx.scene.shape.Circle(p1.x, p1.y, 6, Color.DARKRED);
            slElementsGroup.getChildren().addAll(curve, head);
        });
    }

    public static class Point {
        public double x, y;
        public Point(double x, double y) { this.x = x; this.y = y; }
    }

    private Point getCoordinatesForNumber(int number) {
        int row = (number - 1) / ROWS;
        int viewRow = (ROWS - 1) - row;
        int col = (number - 1) % COLS;
        if (row % 2 == 1) col = (COLS - 1) - col;
        return new Point(col * tileSize, viewRow * tileSize);
    }

    public Point getCenterMoveCoordinates(int number) {
        Point p = getCoordinatesForNumber(number);
        return new Point(p.x + tileSize / 2, p.y + tileSize / 2);
    }

    public double getTileSize() {
        return tileSize;
    }


    public void resizeBoard(double width, double height) {
        this.width = width;
        this.height = height;
        double minDim = Math.min(width, height);
        this.tileSize = (minDim * 0.9) / ROWS;
        double startX = (width - (tileSize * COLS)) / 2;
        double startY = (height - (tileSize * ROWS)) / 2;
        boardGroup.setTranslateX(startX);
        boardGroup.setTranslateY(startY);
        drawBoard();
    }


    public int getSnakeTail(int head) {
        return snakes.getOrDefault(head, -1);
    }


    public int getLadderTop(int bottom) {
        return ladders.getOrDefault(bottom, -1);
    }
}
```

Dice (Dice.java)

```
package com.snakeandladder;
import javafx.animation.RotateTransition;
import javafx.scene.control.Button;

import javafx.scene.layout.VBox;

import javafx.scene.layout.StackPane;

import javafx.scene.paint.Color;

import javafx.scene.shape.Rectangle;

import javafx.scene.text.Font;

import javafx.scene.text.Text;

import javafx.util.Duration;

import java.util.Random;

import java.util.function.Consumer;


public class Dice extends VBox {

    private StackPane diceGraphics;

    private Rectangle diceFace;

    private Text diceValueText;

    private Button rollButton;

    private Random random;

    private boolean isRolling = false;


    public Dice(Consumer<Integer> onRollCallback) {

        this.random = new Random();

        this.setSpacing(10);

        this.setAlignment(javafx.geometry.Pos.CENTER);

        diceGraphics = new StackPane();

        diceFace = new Rectangle(60, 60, Color.WHITE);

        diceFace.setArcHeight(15);
        diceFace.setArcWidth(15);

        diceFace.setStroke(Color.BLACK);

        diceFace.setStrokeWidth(2);

        diceValueText = new Text("1");

        diceValueText.setFont(Font.font(30));

        diceGraphics.getChildren().addAll(diceFace, diceValueText);

        rollButton = new Button("ROLL");

        rollButton.setStyle("-fx-font-size: 16px; -fx-base: #4a90e2; -fx-text-fill: white;");

        rollButton.setOnAction(e -> {

            if (!isRolling) {

                performRoll(onRollCallback);

            }

        });

        getChildren().addAll(diceGraphics,  rollButton);

    }


    public void setRollingDisable(boolean disable) {

        rollButton.setDisable(disable);

    }


    private void performRoll(Consumer<Integer> callback) {

        isRolling = true;

        setRollingDisable(true);

        RotateTransition rt = new RotateTransition(Duration.millis(500), diceGraphics);

        rt.setByAngle(360);

        rt.setCycleCount(2);
```

```
        rt.setAutoReverse(true);
        rt.setOnFinished(e -> {
            int rolledNumber = random.nextInt(6) + 1;
            diceValueText.setText(String.valueOf(rolledNumber));
            isRolling = false;
            callback.accept(rolledNumber);
        });
        rt.play();
    }
}
```

Game App (GameApp.java)

```
package com.snakeandladder;
import javafx.application.Application;
import javafx.stage.Stage;

public class GameApp extends Application {
    @Override
    public void start(Stage primaryStage) {
        GameController controller = new GameController();
        controller.initialize(primaryStage);
    }

    public static void main(String[] args) {
        launch(args);
    }
}
```

Game Controller (GameController.java)

```
package com.snakeandladder;

import javafx.application.Platform;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.Alert;

import javafx.scene.control.Button;

import javafx.scene.control.Label;

import javafx.scene.control.TextInputDialog;

import javafx.scene.layout.BorderPane;

import javafx.scene.layout.VBox;

import javafx.scene.paint.Color;

import javafx.scene.text.Font;

import javafx.scene.text.FontWeight;

import javafx.scene.text.Text;

import javafx.stage.Stage;

import java.util.ArrayList;

import java.util.List;

import java.util.Optional;


public class GameController {

    private Stage primaryStage;

    private BorderPane rootLayout;

    private Board gameBoard;

    private Dice dice;

    private List<Player> players;
    private int currentPlayerIndex = 0;

    private boolean gameRunning = false;

    private VBox sidePanel;

    private Label statusLabel;

    private Label turnLabel;


    public void initialize(Stage stage) {

        this.primaryStage = stage;

        this.rootLayout = new BorderPane();

        this.players = new ArrayList<>();

        gameBoard = new Board();

        rootLayout.setCenter(gameBoard.getBoardGroup());

        createSidePanel();

        rootLayout.setRight(sidePanel);

        Scene scene = new Scene(rootLayout, 1000, 700);

        primaryStage.setTitle("Snake and Ladder - JavaFX");

        primaryStage.setScene(scene);

        primaryStage.setResizable(true);

        primaryStage.show();

        setupResizeListeners(scene);

        setupGame();

    }

    private void createSidePanel() {

        sidePanel = new VBox(20);

        sidePanel.setPrefWidth(250);

        sidePanel.setStyle("-fx-background-color:  #f4f4f4;  -fx-padding:  20;");
        sidePanel.setAlignment(Pos.CENTER);

        Label title = new Label("Snake & Ladder");

        title.setFont(Font.font("Arial", FontWeight.BOLD, 24));

        turnLabel = new Label("Waiting for start...");

        turnLabel.setFont(Font.font("Arial", FontWeight.BOLD, 18));

        turnLabel.setTextFill(Color.DARKBLUE);

        statusLabel = new Label("Welcome!");

        statusLabel.setWrapText(true);

        statusLabel.setFont(Font.font("Arial", 14));

        dice = new Dice(this::handleRoll);

        dice.setRollingDisable(true);

        sidePanel.getChildren().addAll(title, turnLabel, dice, statusLabel);

    }

    private void setupGame() {

        TextInputDialog dialog = new TextInputDialog("2");

        dialog.setTitle("Game Setup");

        dialog.setHeaderText("Welcome to Snake & Ladder");

        dialog.setContentText("Enter number of players (2-6):");

        Optional<String> result = dialog.showAndWait();

        if (result.isPresent()) {

            try {

                int count = Integer.parseInt(result.get());

                if (count < 2) count = 2;

                if (count > 6) count = 6;

                initializePlayers(count);

            } catch (NumberFormatException e) {
                initializePlayers(2);
            }
        } else {
            initializePlayers(2);
        }
    }

    private void initializePlayers(int count) {
        players.clear();
        Color[] availableColors = {Color.RED, Color.BLUE, Color.GREEN, Color.ORANGE, Color.PURPLE, Color.CYAN};
        for (int i = 0; i < count; i++) {
            Player p = new Player("Player " + (i + 1), i + 1, availableColors[i % availableColors.length]);
            players.add(p);
            gameBoard.getBoardGroup().getChildren().add(p.getToken());
            placePlayerAt(p, 1);
        }
        gameRunning = true;
        currentPlayerIndex = 0;
        dice.setRollingDisable(false);
        updateTurnUI();
    }

    private void placePlayerAt(Player p, int position) {
        p.setPosition(position);
        var point = gameBoard.getCenterMoveCoordinates(position);
        double offset = (p.getToken().getRadius() * 0.5) * (players.indexOf(p) % 3);
        p.placeAt(point.x + offset, point.y + offset);
    }

    private void handleRoll(int rolledValue) {
        if (!gameRunning) return;
        Player currentPlayer = players.get(currentPlayerIndex);
        statusLabel.setText(currentPlayer.getName() + " rolled a " + rolledValue);
        movePlayer(currentPlayer, rolledValue);
    }

    private void movePlayer(Player player, int steps) {
        int currentPos = player.getPosition();
        int targetPos = currentPos + steps;
        if (targetPos > 100) {
            statusLabel.setText(player.getName() + " needs exact roll to win!");
            switchTurn();
            return;
        }
        var point = gameBoard.getCenterMoveCoordinates(targetPos);
        double offset = (player.getToken().getRadius() * 0.5) * (players.indexOf(player) % 3);
        player.animateMove(point.x + offset, point.y + offset, () -> {
            player.setPosition(targetPos);
            checkTileEvents(player);
        });
    }
```

```
private void checkTileEvents(Player player) {
    int pos = player.getPosition();
    int snakeTail = gameBoard.getSnakeTail(pos);
    if (snakeTail != -1) {
        statusLabel.setText("Oh no! " + player.getName() + " bitten by a snake!");
        animateSpecialMove(player, snakeTail);
        return;
    }
    int ladderTop = gameBoard.getLadderTop(pos);
    if (ladderTop != -1) {
        statusLabel.setText("Yay! " + player.getName() + " climbed a ladder!");
        animateSpecialMove(player, ladderTop);
        return;
    }
    if (pos == 100) {
        gameRunning = false;
        statusLabel.setText("WINNER: " + player.getName());
        showVictoryDialog(player);
        return;
    }
    switchTurn();
}

private void animateSpecialMove(Player player, int targetPos) {
    var point = gameBoard.getCenterMoveCoordinates(targetPos);
    double offset = (player.getToken().getRadius() * 0.5) * (players.indexOf(player) % 3);
    var pause = new javafx.animation.PauseTransition(javafx.util.Duration.millis(500));
    pause.setOnFinished(e -> {
        player.animateMove(point.x + offset, point.y + offset, () -> {
            player.setPosition(targetPos);
            if (targetPos == 100) {
                gameRunning = false;
                statusLabel.setText("WINNER: " + player.getName());
                showVictoryDialog(player);
            } else {
                switchTurn();
            }
        });
    });
    pause.play();
}

private void switchTurn() {
    if (!gameRunning) return;
    currentPlayerIndex = (currentPlayerIndex + 1) % players.size();
    updateTurnUI();
    dice.setRollingDisable(false);
}

private void updateTurnUI() {
    Player p = players.get(currentPlayerIndex);
    turnLabel.setText("Turn: " + p.getName());
    turnLabel.setTextFill(p.getColor());
}

private void showVictoryDialog(Player winner) {
    Alert alert = new Alert(Alert.AlertType.INFORMATION);
    alert.setTitle("Victory!");
    alert.setHeaderText("We have a winner!");
    alert.setContentText(winner.getName() + " has won the game!");
    alert.show();
     dice.setRollingDisable(true);
}

private void setupResizeListeners(Scene scene) {
    scene.widthProperty().addListener((obs, oldVal, newVal) -> {
        if (gameBoard != null) {
            gameBoard.resizeBoard(newVal.doubleValue() - 250, scene.getHeight());
            refreshPlayerPositions();
        }
    });
    scene.heightProperty().addListener((obs, oldVal, newVal) -> {
        if (gameBoard != null) {
            gameBoard.resizeBoard(scene.getWidth() - 250, newVal.doubleValue());
            refreshPlayerPositions();
        }
    });
}
```

```
    private void refreshPlayerPositions() {
        for (Player p : players) {
            placePlayerAt(p, p.getPosition());
        }
    }
}
```

Player (Player.java)

```
package com.snakeandladder;
import javafx.animation.TranslateTransition;
import javafx.scene.paint.Color;
import javafx.scene.shape.Circle;
import javafx.util.Duration;


public class Player {
    private String name;
    private int id;
    private int currentPosition;
    private Circle token;
    private Color color;


    public Player(String name, int id, Color color) {
        this.name = name;
        this.id = id;
        this.color = color;
        this.currentPosition = 1;
    this.token = new Circle(15, color);

    this.token.setStroke(Color.BLACK);

    this.token.setStrokeWidth(2);

}


public Circle getToken() {

    return token;

}


public int getPosition() {

    return currentPosition;

}


public void setPosition(int position) {

    this.currentPosition = position;

}


public String getName() {

    return name;

}


public Color getColor() {

    return color;

}


public void animateMove(double x, double y, Runnable onFinished) {
        TranslateTransition tt = new TranslateTransition(Duration.millis(300), token);
        tt.setToX(x);
        tt.setToY(y);
        tt.setOnFinished(e -> {
            if (onFinished != null) onFinished.run();
        });
        tt.play();
    }

    public void placeAt(double x, double y) {
        token.setTranslateX(x);
        token.setTranslateY(y);
    }
}
```

Tile (Tile.java)

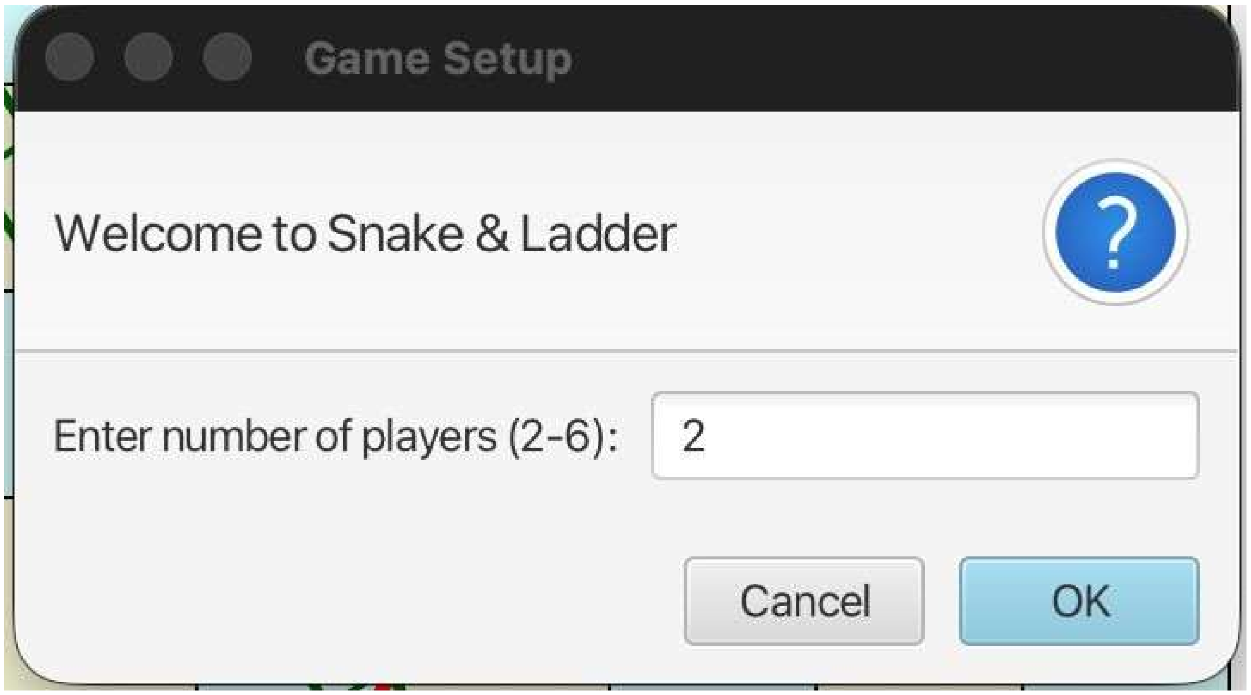
```
package com.snakeandladder;
import javafx.scene.layout.StackPane;
import javafx.scene.paint.Color;
import javafx.scene.shape.Rectangle;
import javafx.scene.text.Text;
import javafx.scene.text.Font;
import javafx.scene.text.FontWeight;

public class Tile extends StackPane {
    private Rectangle border;
    private Text text;
    private int number;


    public Tile(int number, double size) {
        this.number = number;
        border = new Rectangle(size, size);
        Color color = (number % 2 == 0) ? Color.LIGHTYELLOW : Color.LIGHTCYAN;
        border.setFill(color);
        border.setStroke(Color.BLACK);
        text = new Text(String.valueOf(number));
        text.setFont(Font.font("Arial", FontWeight.BOLD, 14));
        getChildren().addAll(border, text);
    }


    public void updateSize(double newSize) {
        border.setWidth(newSize);
        border.setHeight(newSize);
        text.setFont(Font.font("Arial", FontWeight.BOLD, newSize * 0.3));
    }
}
```

pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchemainstance”
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.snakeandladder</groupId>
  <artifactId>snakeandladder</artifactId>
  <version>1.0-SNAPSHOT</version>
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>17</maven.compiler.source>
    <maven.compiler.target>17</maven.compiler.target>
    <javafx.version>21.0.1</javafx.version>
  </properties>
  <dependencies>
    <dependency>
      <groupId>org.openjfx</groupId>
      <artifactId>javafx-controls</artifactId>
      <version>${javafx.version}</version>
    </dependency>
    <dependency>
      <groupId>org.openjfx</groupId>
      <artifactId>javafx-fxml</artifactId>
      <version>${javafx.version}</version>
    </dependency>
    <dependency>
      <groupId>org.openjfx</groupId>
```

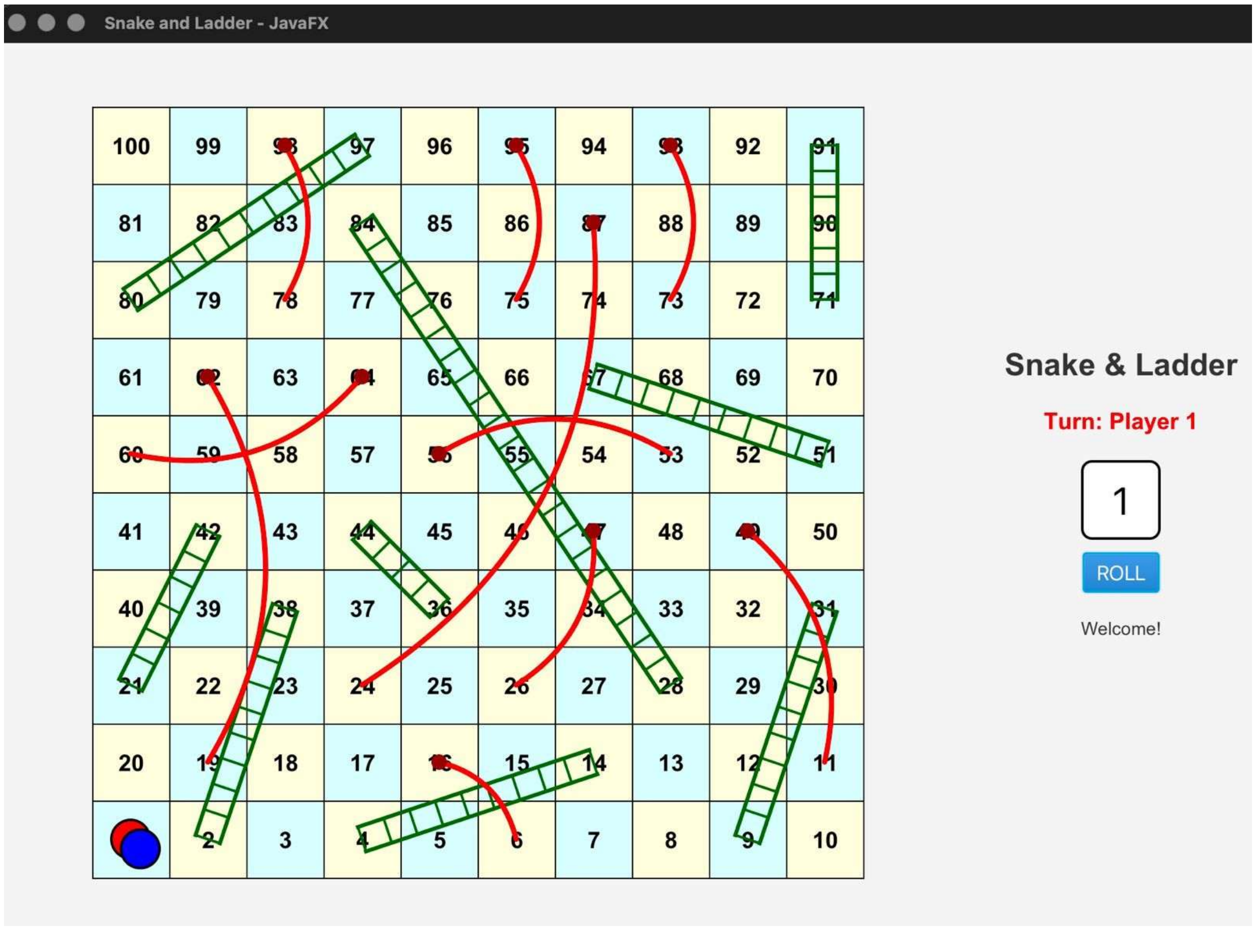
```
        <artifactId>javafx-graphics</artifactId>
        <version>${javafx.version}</version>
      </dependency>
    </dependencies>
    <build>
      <plugins>
        <plugin>
          <groupId>org.apache.maven.plugins</groupId>
          <artifactId>maven-compiler-plugin</artifactId>
          <version>3.8.1</version>
          <configuration>
            <source>17</source>
            <target>17</target>
          </configuration>
        </plugin>
        <plugin>
          <groupId>org.openjfx</groupId>
          <artifactId>javafx-maven-plugin</artifactId>
          <version>0.0.8</version>
          <configuration>
            <mainClass>com.snakeandladder.GameApp</mainClass>
          </configuration>
        </plugin>
      </plugins>
    </build>
</project>
```

Results



Welcome to Snake & Ladder

Enter number of players (2-6): 2

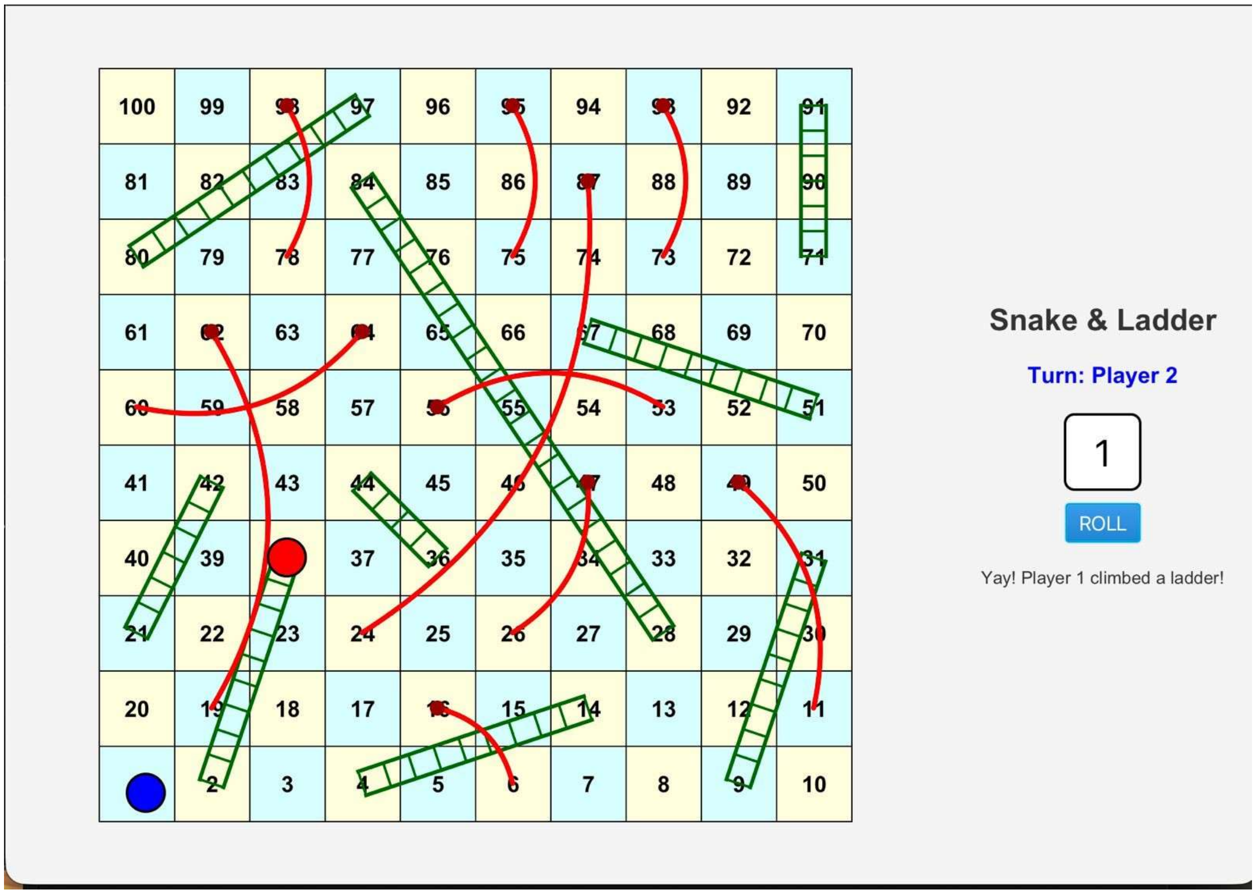


Snake & Ladder

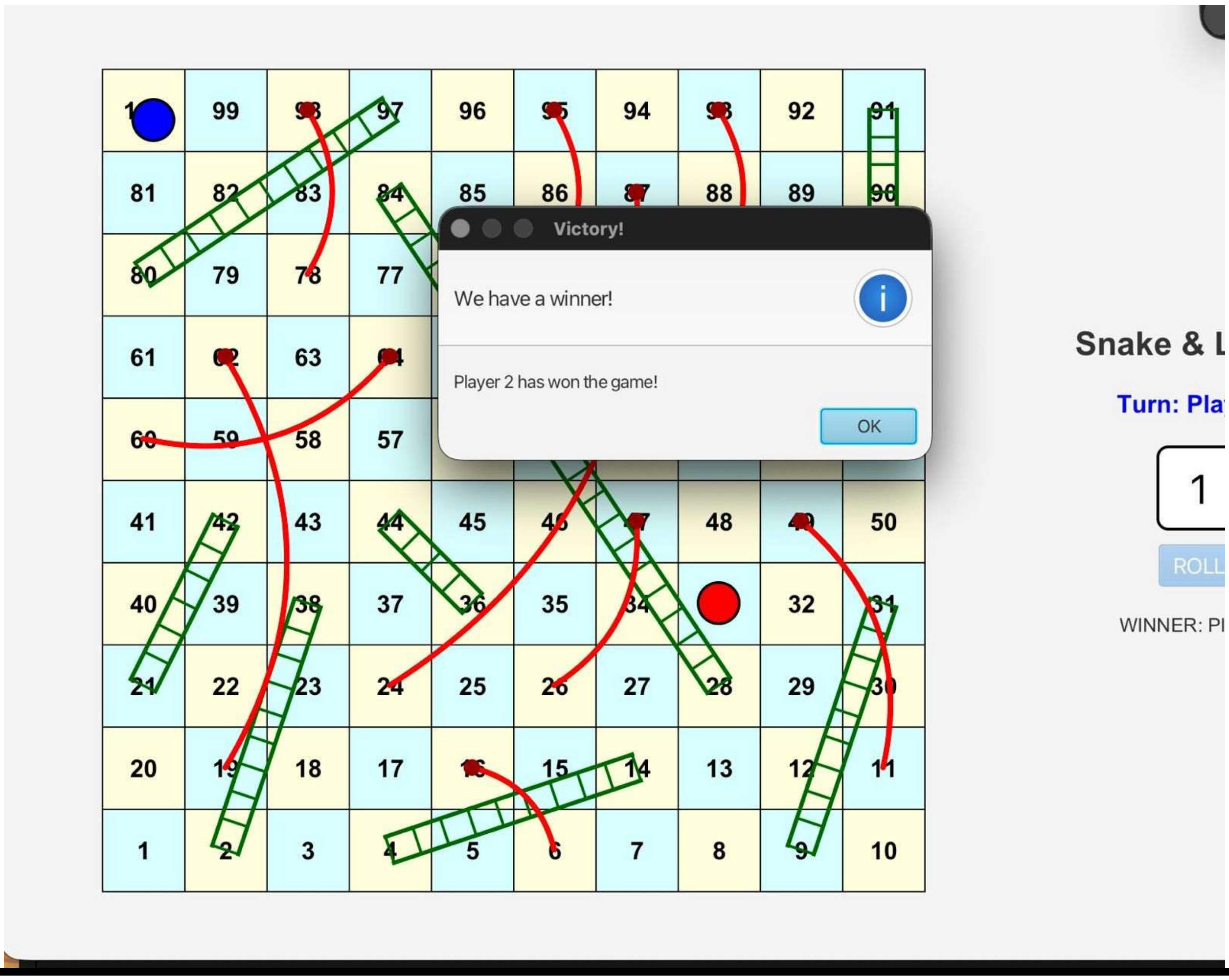
Turn: Player 1

1

Welcome!



Yay! Player 1 climbed a ladder!



We have a winner!

Player 2 has won the game!

WINNER: Pl

References

1. Oracle Corporation, Java SE Documentation.

Available at: https://docs.oracle.com/javase/

(Used for understanding Java syntax, OOP concepts, and standard libraries)

2. Herbert Schildt, Java: The Complete Reference, McGraw-Hill Education.

(Referred for Java programming fundamentals and object-oriented concepts)

3. GeeksforGeeks, Java Programming Tutorials.

Available at: https://www.geeksforgeeks.org/java/

(Referred for logic building, arrays, control structures, and game implementation ideas)

4. Baeldung, Java Tutorials.

Available at: https://www.baeldung.com/java-tutorial

(Referred for clean coding practices and Java concepts)

5. Wikipedia, Snakes and Ladders.

Available at: https://en.wikipedia.org/wiki/Snakes_and_Ladders

(Referred for understanding game rules and mechanics)