The Ultimate Trifecta: Tic Tac Toe, Ping Pong, and Number Guessing Combined

Name	Registration No	Roll NO
Bhadri Narayanan S	12110322	8
Neelotpal Chowdhury	12104606	38
Ambati Gopi Vikram	12104691	20

GitHub link:-

https://github.com/NeelotpalChowdhury529/multiplayergame

Table of content:-

Serial No	Content
1	Objectives
2	Introduction
3	Requirements
4	Module 1
5	Module 2
6	Module 3
7	Module 4
8	Conclusion
9	Learning Outcomes
10	References

Objectives of this project :-

1. To create an immersive and visually appealing game world that draws players in and encourages them to explore, compete, or cooperate with others.

2. To implement game mechanics and rules that are fun, challenging, and balanced, while also accommodating multiple players with different skill levels and preferences.

Introduction:-

We have created a multiplayer game using Java and JFrame which includes three games:-

1. Tic-Tac-Toe
2. Ping-Pong
3. Number-Guessing

Tic-Tac-Toe is a simple and popular game played on a 3x3 grid. The objective of the game is to get three of your marks (either X or O) in a row, either horizontally, vertically, or diagonally. It's a two-player game, with each player taking turns placing their mark on the board. The player who gets three in a row first, wins the game.To start the game, the first player (who is usually chosen randomly) places their mark (X or O) on an empty cell on the grid. Then the second player takes their turn, placing their mark on another empty cell. The game continues like this until one of the players has three of their marks in a row, or until all of the cells on the grid are filled up.If all the cells on the grid are filled up and neither player has won, then the game ends in a tie. Tic Tac Toe is a great game for teaching strategy and critical thinking skills, and it can be enjoyed by people of all ages.

Pong-Game is a classic and addictive game that has been enjoyed by generations! In this simple yet exciting game, players control paddles to hit a ball back and forth across the screen. The goal is to prevent the ball from passing your paddle while trying to score points by getting the ball past your opponent's paddle. PongGame is a perfect choice for anyone looking for a fun and challenging way to pass the time. So, get ready to test your reflexes and improve your skills in this timeless game!

Number-Guessing game is a fun and challenging game that will test your ability to guess a random number between 1 and 100. The objective of the game is to correctly guess the randomly generated number within the fewest attempts possible. With each guess, you will receive feedback indicating whether your guess is too high or too low, giving you clues to help you make your next guess. So, put your thinking cap on, sharpen your math skills, and let's see how good you are at guessing!

Requirements:-

1. JAVA
2. Java Swing
3. Java AWT
4. Java JFrame

Module 1:Main Frame:-

Screenshot:-

Code:-

```
import javax.swing.*;

import java.awt.*;

import java.awt.event.*;


class GameProject extends JFrame implements ActionListener {

   JButton ticTacToeButton;

   JButton pingpong;

   JButton numberguessing;

   JButton quitButton;


   public GameProject() {

      super("Game Project");
```

```
ticTacToeButton = new JButton("Tic-Tac-Toe");
pingpong = new JButton("Ping-Pong");
numberguessing =new JButton("Number-Guessing");

quitButton = new JButton("Quit");

Font buttonFont = new Font("Arial", Font.PLAIN, 80);
ticTacToeButton.setFont(buttonFont);
pingpong.setFont(buttonFont);
numberguessing.setFont(buttonFont);
quitButton.setFont(buttonFont);

ticTacToeButton.addActionListener(this);
pingpong.addActionListener(this);
numberguessing.addActionListener(this);
quitButton.addActionListener(this);

JPanel panel = new JPanel(new GridLayout(2, 2));
panel.add(ticTacToeButton);
panel.add(pingpong);
panel.add(numberguessing);
panel.add(quitButton);

getContentPane().add(panel);
```

```
    pack();
    setVisible(true);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}

public void actionPerformed(ActionEvent e) {
    if (e.getSource() == ticTacToeButton) {
        TicTacToe game = new TicTacToe();
    } else if (e.getSource() == pingpong) {
        PongGame game = new PongGame();
        GameFrame frame = new GameFrame();

    }
    else if (e.getSource()==numberguessing){
        NumberGuessingGame game=new NumberGuessingGame();

    }
     else if (e.getSource() == quitButton) {
        System.exit(0);
    }
}
    public static void main(String[] args) {
    GameProject project =new GameProject();
}
```

}

Explanation:-

This is a Java Swing GUI program that creates a menu of games that the user can select to play. The program has a main window that contains four buttons: Tic-Tac-Toe, Ping-Pong, Number-Guessing, and Quit.The code starts by importing the necessary Java Swing and AWT libraries using the import statements.

The main class is called GameProject, which extends the JFrame class and implements the ActionListener interface. The JFrame class is used to create the main window for the program.The program creates four buttons using the JButton class.

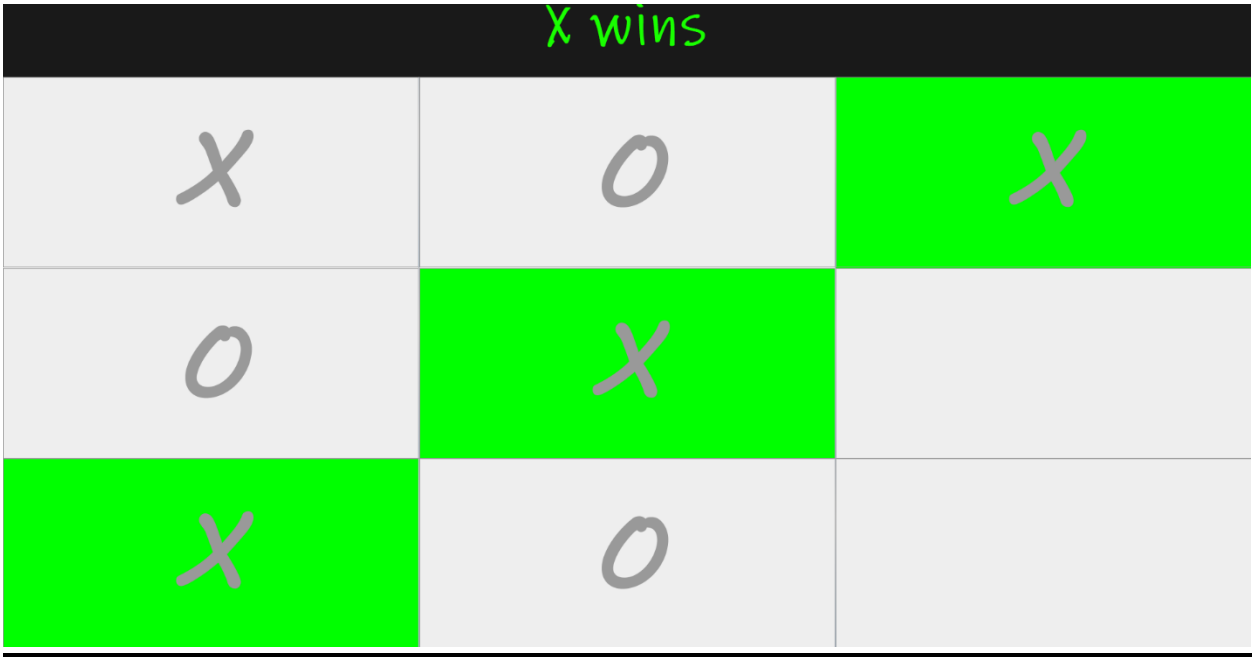
Each button is given a label that is displayed on the button.The program sets up an event listener for each button using the addActionListener() method.

The this keyword is used to refer to the current instance of the GameProject class, which implements the ActionListener interface.The program then creates a JPanel object with a 2x2 grid layout and adds the four buttons to the panel.

The this keyword is used to refer to the current instance of the GameProjectclass, which implements the ActionListener interface.The program then creates a JPanel object with a 2x2 grid layout and adds the four buttons to the panel.

Module 2:Tic-Tac-Toe game:-

Screenshot:-



Code:-

```
import java.awt.*;

import java.awt.event.*;

import java.util.*;

import javax.swing.*;


public class TicTacToe implements ActionListener{


   Random random = new Random();

   JFrame frame = new JFrame();

   JPanel title_panel = new JPanel();

   JPanel button_panel = new JPanel();
```

```
JLabel textfield = new JLabel();
JButton[] buttons = new JButton[9];
boolean player1_turn;

TicTacToe(){

   frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   frame.setSize(800,800);
   frame.getContentPane().setBackground(new Color(50,50,50));
   frame.setLayout(new BorderLayout());
   frame.setVisible(true);

   textfield.setBackground(new Color(25,25,25));
   textfield.setForeground(new Color(25,255,0));
   textfield.setFont(new Font("Ink Free",Font.BOLD,75));
   textfield.setHorizontalAlignment(JLabel.CENTER);
   textfield.setText("Tic-Tac-Toe");
   textfield.setOpaque(true);

   title_panel.setLayout(new BorderLayout());
   title_panel.setBounds(0,0,800,100);

   button_panel.setLayout(new GridLayout(3,3));
   button_panel.setBackground(new Color(150,150,150));
```

```
    for(int i=0;i<9;i++) {
        buttons[i] = new JButton();
        button_panel.add(buttons[i]);
        buttons[i].setFont(new Font("MV Boli",Font.BOLD,120));
        buttons[i].setFocusable(false);
        buttons[i].addActionListener(this);
    }

    title_panel.add(textfield);
    frame.add(title_panel,BorderLayout.NORTH);
    frame.add(button_panel);

    firstTurn();
}

@Override
public void actionPerformed(ActionEvent e) {

    for(int i=0;i<9;i++) {
        if(e.getSource()==buttons[i]) {
            if(player1_turn) {
                if(buttons[i].getText()=="") {
                    buttons[i].setForeground(new Color(255,0,0));
```

```
                buttons[i].setText("X");
                player1_turn=false;
                textfield.setText("O turn");
                check();
            }
        }
        else {
            if(buttons[i].getText()=="") {
                buttons[i].setForeground(new Color(0,0,255));
                buttons[i].setText("O");
                player1_turn=true;
                textfield.setText("X turn");
                check();
            }
        }
    }
}

public void firstTurn() {

    try {
        Thread.sleep(2000);
    } catch (InterruptedException e) {
```

```
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

        if(random.nextInt(2)==0) {
            player1_turn=true;
            textfield.setText("X turn");
        }
        else {
            player1_turn=false;
            textfield.setText("O turn");
        }
    }

    public void check() {
        //check X win conditions
        if(
                (buttons[0].getText()=="X") &&
                        (buttons[1].getText()=="X") &&
                        (buttons[2].getText()=="X")
        ) {
            xWins(0,1,2);
        }
        if(
```

```
        (buttons[3].getText()=="X") &&
                (buttons[4].getText()=="X") &&
                (buttons[5].getText()=="X")
    ) {
       xWins(3,4,5);
    }
    if(
        (buttons[6].getText()=="X") &&
                (buttons[7].getText()=="X") &&
                (buttons[8].getText()=="X")
    ) {
       xWins(6,7,8);
    }
    if(
        (buttons[0].getText()=="X") &&
                (buttons[3].getText()=="X") &&
                (buttons[6].getText()=="X")
    ) {
       xWins(0,3,6);
    }
    if(
        (buttons[1].getText()=="X") &&
                (buttons[4].getText()=="X") &&
                (buttons[7].getText()=="X")
```

```
) {
   xWins(1,4,7);
}
if(
      (buttons[2].getText()=="X") &&
            (buttons[5].getText()=="X") &&
            (buttons[8].getText()=="X")
) {
   xWins(2,5,8);
}
if(
      (buttons[0].getText()=="X") &&
            (buttons[4].getText()=="X") &&
            (buttons[8].getText()=="X")
) {
   xWins(0,4,8);
}
if(
      (buttons[2].getText()=="X") &&
            (buttons[4].getText()=="X") &&
            (buttons[6].getText()=="X")
) {
   xWins(2,4,6);
}
```

```
//check O win conditions
if(
      (buttons[0].getText()=="O") &&
            (buttons[1].getText()=="O") &&
            (buttons[2].getText()=="O")
) {
   oWins(0,1,2);
}
if(
      (buttons[3].getText()=="O") &&
            (buttons[4].getText()=="O") &&
            (buttons[5].getText()=="O")
) {
   oWins(3,4,5);
}
if(
      (buttons[6].getText()=="O") &&
            (buttons[7].getText()=="O") &&
            (buttons[8].getText()=="O")
) {
   oWins(6,7,8);
}
if(
      (buttons[0].getText()=="O") &&
```

```
            (buttons[3].getText()=="O") &&
            (buttons[6].getText()=="O")
) {
   oWins(0,3,6);
}
if(
      (buttons[1].getText()=="O") &&
            (buttons[4].getText()=="O") &&
            (buttons[7].getText()=="O")
) {
   oWins(1,4,7);
}
if(
      (buttons[2].getText()=="O") &&
            (buttons[5].getText()=="O") &&
            (buttons[8].getText()=="O")
) {
   oWins(2,5,8);
}
if(
      (buttons[0].getText()=="O") &&
            (buttons[4].getText()=="O") &&
            (buttons[8].getText()=="O")
) {
```

```
        oWins(0,4,8);
    }
    if(
          (buttons[2].getText()=="O") &&
                (buttons[4].getText()=="O") &&
                (buttons[6].getText()=="O")
    ) {
        oWins(2,4,6);
    }
}

public void xWins(int a,int b,int c) {
    buttons[a].setBackground(Color.GREEN);
    buttons[b].setBackground(Color.GREEN);
    buttons[c].setBackground(Color.GREEN);

    for(int i=0;i<9;i++) {
        buttons[i].setEnabled(false);
    }
    textfield.setText("X wins");
}
public void oWins(int a,int b,int c) {
    buttons[a].setBackground(Color.GREEN);
    buttons[b].setBackground(Color.GREEN);
```

```
        buttons[c].setBackground(Color.GREEN);


        for(int i=0;i<9;i++) {
            buttons[i].setEnabled(false);
        }
        textfield.setText("O wins");
    }
}
public class Main1 {


    public static void main(String[] args) {


        TicTacToe tictactoe = new TicTacToe();
    }
}
```

Explanation:-

The GUI is built using the Java Swing library.The program starts by importing the necessary libraries: java.awt.*, java.awt.event.*, java.util.*, and javax.swing.*. Then, it defines a class called TicTacToe that implements the ActionListener interface.In the TicTacToe class, several instance variables are defined: random, frame, title_panel, button_panel, textfield, buttons, and player1_turn.random is an instance of the Random class and is used to randomly choose which player goes first. frame is an instance of the JFrame class and is the main window of the GUI. title_panel and button_panel are instances of

the JPanel class and are used to organize the GUI components. textfield is an instance of the JLabel class and is used to display the game status. buttons is an array of nine instances of the JButton class and represents the Tic Tac Toe board. player1_turn is a boolean variable that keeps track of which player's turn it is.

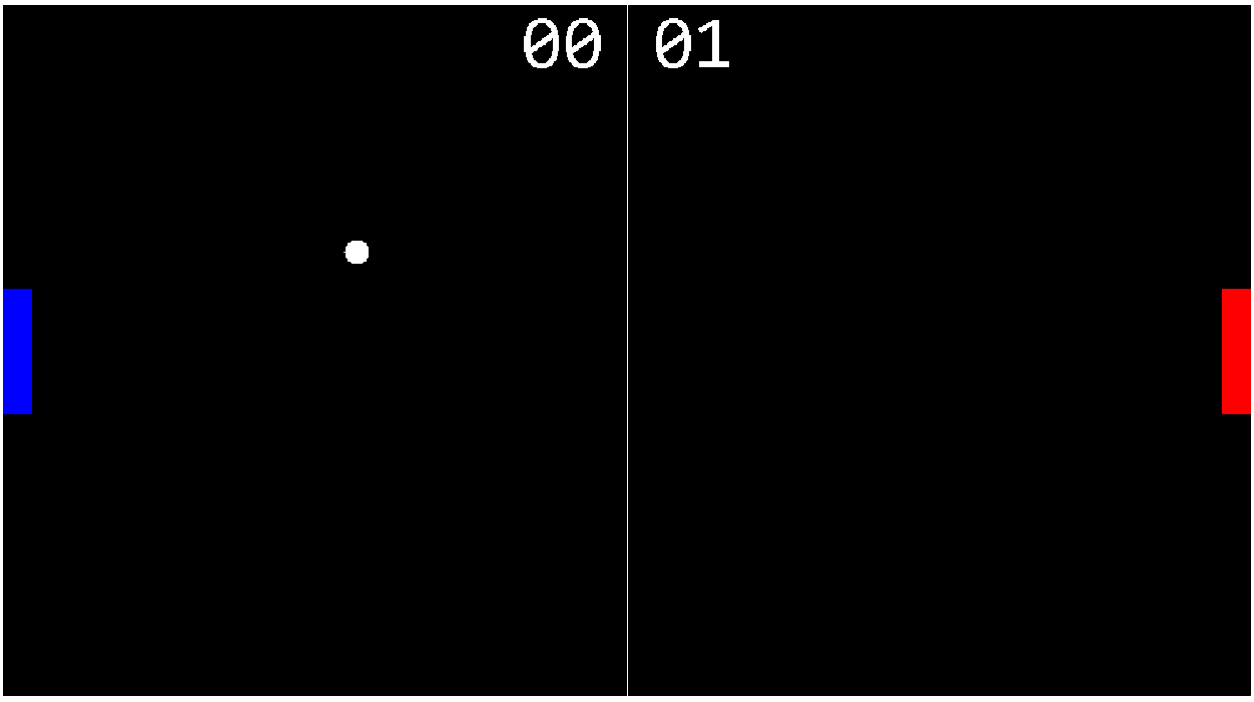
The constructor of the TicTacToe class initializes the GUI components. It sets the size, background color, and layout of the frame. It sets the background color, font, and text of the textfield. It sets the layout and background color of the button_panel. It creates nine JButton objects and adds them to the button_panel. It sets the font and action listener for each JButton. Finally, it adds the title_panel and button_panel to the frame and calls the firstTurn method to randomly choose which player goes first.The firstTurn method uses the random instance to randomly choose which player goes first. If the result is 0, then player1_turn is set to true and the textfield displays "X turn". Otherwise, player1_turn is set to false and the textfield displays "O turn".The actionPerformed method is called whenever a JButton is clicked. It checks which button was clicked and sets the text and color of the button according to the current player's turn. It then updates the player1_turn variable and the textfield to reflect the next player's turn. Finally, it calls the check method to check if the game has ended.

The check method checks if either player has won the game. It does this by checking each possible win condition: three in a row horizontally, vertically, or diagonally. If a win condition is met, it calls either the xWins or oWins method to display the winner and end the game.

Overall, this program provides a simple implementation of the classic Tic Tac Toe game using a graphical user interface.

Module 3:Ping-Pong game:-

Screenshot:-



Code:-

```
import java.awt.*;

import java.awt.event.*;

import java.util.*;

import javax.swing.*;


public class GamePanel extends JPanel implements Runnable{


    static final int GAME_WIDTH = 1000;

    static final int GAME_HEIGHT = (int)(GAME_WIDTH * (0.5555));

    static final Dimension SCREEN_SIZE = new Dimension(GAME_WIDTH,GAME_HEIGHT);

    static final int BALL_DIAMETER = 20;
```

```
static final int PADDLE_WIDTH = 25;
static final int PADDLE_HEIGHT = 100;
Thread gameThread;
Image image;
Graphics graphics;
Random random;
Paddle paddle1;
Paddle paddle2;
Ball ball;
Score score;

GamePanel(){
   newPaddles();
   newBall();
   score = new Score(GAME_WIDTH,GAME_HEIGHT);
   this.setFocusable(true);
   this.addKeyListener(new AL());
   this.setPreferredSize(SCREEN_SIZE);

   gameThread = new Thread(this);
   gameThread.start();
}

public void newBall() {
```

```
        random = new Random();
        ball = new Ball((GAME_WIDTH/2)-(BALL_DIAMETER/2),random.nextInt(GAME_HEIGHT-BALL_DIAMETER),BALL_DIAMETER,BALL_DIAMETER);
    }
    public void newPaddles() {
        paddle1 = new Paddle(0,(GAME_HEIGHT/2)-(PADDLE_HEIGHT/2),PADDLE_WIDTH,PADDLE_HEIGHT,1);
        paddle2 = new Paddle(GAME_WIDTH-PADDLE_WIDTH,(GAME_HEIGHT/2)-(PADDLE_HEIGHT/2),PADDLE_WIDTH,PADDLE_HEIGHT,2);
    }
    public void paint(Graphics g) {
        image = createImage(getWidth(),getHeight());
        graphics = image.getGraphics();
        draw(graphics);
        g.drawImage(image,0,0,this);
    }
    public void draw(Graphics g) {
        paddle1.draw(g);
        paddle2.draw(g);
        ball.draw(g);
        score.draw(g);
        Toolkit.getDefaultToolkit().sync(); // I forgot to add this line of code in the video, it helps with the animation

    }
    public void move() {
```

```
        paddle1.move();
        paddle2.move();
        ball.move();
    }
    public void checkCollision() {

        //bounce ball off top & bottom window edges
        if(ball.y <=0) {
            ball.setYDirection(-ball.yVelocity);
        }
        if(ball.y >= GAME_HEIGHT-BALL_DIAMETER) {
            ball.setYDirection(-ball.yVelocity);
        }
        //bounce ball off paddles
        if(ball.intersects(paddle1)) {
            ball.xVelocity = Math.abs(ball.xVelocity);
            ball.xVelocity++; //optional for more difficulty
            if(ball.yVelocity>0)
                ball.yVelocity++; //optional for more difficulty
            else
                ball.yVelocity--;
            ball.setXDirection(ball.xVelocity);
            ball.setYDirection(ball.yVelocity);
        }
```

```
if(ball.intersects(paddle2)) {
    ball.xVelocity = Math.abs(ball.xVelocity);
    ball.xVelocity++; //optional for more difficulty
    if(ball.yVelocity>0)
        ball.yVelocity++; //optional for more difficulty
    else
        ball.yVelocity--;
    ball.setXDirection(-ball.xVelocity);
    ball.setYDirection(ball.yVelocity);
}
//stops paddles at window edges
if(paddle1.y<=0)
    paddle1.y=0;
if(paddle1.y >= (GAME_HEIGHT-PADDLE_HEIGHT))
    paddle1.y = GAME_HEIGHT-PADDLE_HEIGHT;
if(paddle2.y<=0)
    paddle2.y=0;
if(paddle2.y >= (GAME_HEIGHT-PADDLE_HEIGHT))
    paddle2.y = GAME_HEIGHT-PADDLE_HEIGHT;
//give a player 1 point and creates new paddles & ball
if(ball.x <=0) {
    score.player2++;
    newPaddles();
    newBall();
```

```
            System.out.println("Player 2: "+score.player2);
        }
        if(ball.x >= GAME_WIDTH-BALL_DIAMETER) {
            score.player1++;
            newPaddles();
            newBall();
            System.out.println("Player 1: "+score.player1);
        }
    }
    public void run() {
        //game loop
        long lastTime = System.nanoTime();
        double amountOfTicks =60.0;
        double ns = 1000000000 / amountOfTicks;
        double delta = 0;
        while(true) {
            long now = System.nanoTime();
            delta += (now -lastTime)/ns;
            lastTime = now;
            if(delta >=1) {
                move();
                checkCollision();
                repaint();
                delta--;
```

```
            }
        }
    }
    public class AL extends KeyAdapter{
        public void keyPressed(KeyEvent e) {
            paddle1.keyPressed(e);
            paddle2.keyPressed(e);
        }
        public void keyReleased(KeyEvent e) {
            paddle1.keyReleased(e);
            paddle2.keyReleased(e);
        }
    }
}

import java.awt.*;
import javax.swing.*;

public class GameFrame extends JFrame{

    GamePanel panel;

    GameFrame(){
        panel = new GamePanel();
```

```
        this.add(panel);
        this.setTitle("Pong Game");
        this.setResizable(false);
        this.setBackground(Color.black);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.pack();
        this.setVisible(true);
        this.setLocationRelativeTo(null);
    }
}

import java.awt.*;
import java.util.*;

public class Ball extends Rectangle{

    Random random;
    int xVelocity;
    int yVelocity;
    int initialSpeed = 2;

    Ball(int x, int y, int width, int height){
        super(x,y,width,height);
        random = new Random();
```

```
    int randomXDirection = random.nextInt(2);
    if(randomXDirection == 0)
        randomXDirection--;
    setXDirection(randomXDirection*initialSpeed);

    int randomYDirection = random.nextInt(2);
    if(randomYDirection == 0)
        randomYDirection--;
    setYDirection(randomYDirection*initialSpeed);

}

public void setXDirection(int randomXDirection) {
    xVelocity = randomXDirection;
}
public void setYDirection(int randomYDirection) {
    yVelocity = randomYDirection;
}
public void move() {
    x += xVelocity;
    y += yVelocity;
}
public void draw(Graphics g) {
    g.setColor(Color.white);
```

```
        g.fillOval(x, y, height, width);
    }
}

import java.awt.*;
import java.awt.event.*;

public class Paddle extends Rectangle{

    int id;
    int yVelocity;
    int speed = 10;

    Paddle(int x, int y, int PADDLE_WIDTH, int PADDLE_HEIGHT, int id){
        super(x,y,PADDLE_WIDTH,PADDLE_HEIGHT);
        this.id=id;
    }

    public void keyPressed(KeyEvent e) {
        switch(id) {
            case 1:
                if(e.getKeyCode()==KeyEvent.VK_W) {
                    setYDirection(-speed);
                }
```

```
            if(e.getKeyCode()==KeyEvent.VK_S) {
               setYDirection(speed);
            }
            break;
         case 2:
            if(e.getKeyCode()==KeyEvent.VK_UP) {
               setYDirection(-speed);
            }
            if(e.getKeyCode()==KeyEvent.VK_DOWN) {
               setYDirection(speed);
            }
            break;
      }
   }
   public void keyReleased(KeyEvent e) {
      switch(id) {
         case 1:
            if(e.getKeyCode()==KeyEvent.VK_W) {
               setYDirection(0);
            }
            if(e.getKeyCode()==KeyEvent.VK_S) {
               setYDirection(0);
            }
            break;
```

```
            case 2:
               if(e.getKeyCode()==KeyEvent.VK_UP) {
                  setYDirection(0);
               }
               if(e.getKeyCode()==KeyEvent.VK_DOWN) {
                  setYDirection(0);
               }
               break;
         }
      }
      public void setYDirection(int yDirection) {
         yVelocity = yDirection;
      }
      public void move() {
         y= y + yVelocity;
      }
      public void draw(Graphics g) {
         if(id==1)
            g.setColor(Color.blue);
         else
            g.setColor(Color.red);
         g.fillRect(x, y, width, height);
      }
   }
```

```
import java.awt.*;


public class Score extends Rectangle{


   static int GAME_WIDTH;
   static int GAME_HEIGHT;
   int player1;
   int player2;


   Score(int GAME_WIDTH, int GAME_HEIGHT){
      Score.GAME_WIDTH = GAME_WIDTH;
      Score.GAME_HEIGHT = GAME_HEIGHT;
   }
   public void draw(Graphics g) {
      g.setColor(Color.white);
      g.setFont(new Font("Consolas",Font.PLAIN,60));


      g.drawLine(GAME_WIDTH/2, 0, GAME_WIDTH/2, GAME_HEIGHT);


      g.drawString(String.valueOf(player1/10)+String.valueOf(player1%10), (GAME_WIDTH/2)-
85, 50);
      g.drawString(String.valueOf(player2/10)+String.valueOf(player2%10),
(GAME_WIDTH/2)+20, 50);
   }
}
```

```
public class PongGame {


   public static void main(String[] args) {


      GameFrame frame = new GameFrame();


   }
}
```

Explanation:-

GamePanel:

This class extends JPanel and implements Runnable.It creates the game panel, sets up the dimensions of the game screen and the ball, paddle, and score sizes.It initializes the game thread and starts it, adds a KeyListener to listen for paddle movements and sets the focusable property of the panel to true.It also contains methods to draw the game objects, move them, check for collisions, and create new paddles and balls when a player scores.Finally, it overrides the paint() method to draw the game graphics on the panel.

GameFrame:

This class extends JFrame.It creates the game frame, adds a GamePanel object to it, sets its properties, and makes it visible.

Ball:

This class extends Rectangle.It creates a ball object, sets its position and size, and initializes its x and y velocities.It also contains methods to

move the ball, set its x and y velocities, and check if it intersects with another object.

Paddle:

This class extends Rectangle.It creates a paddle object, sets its position, size, and player number, and initializes its y velocity.It also contains methods to move the paddle up or down, set its y velocity, and check if it intersects with another object.

Score:

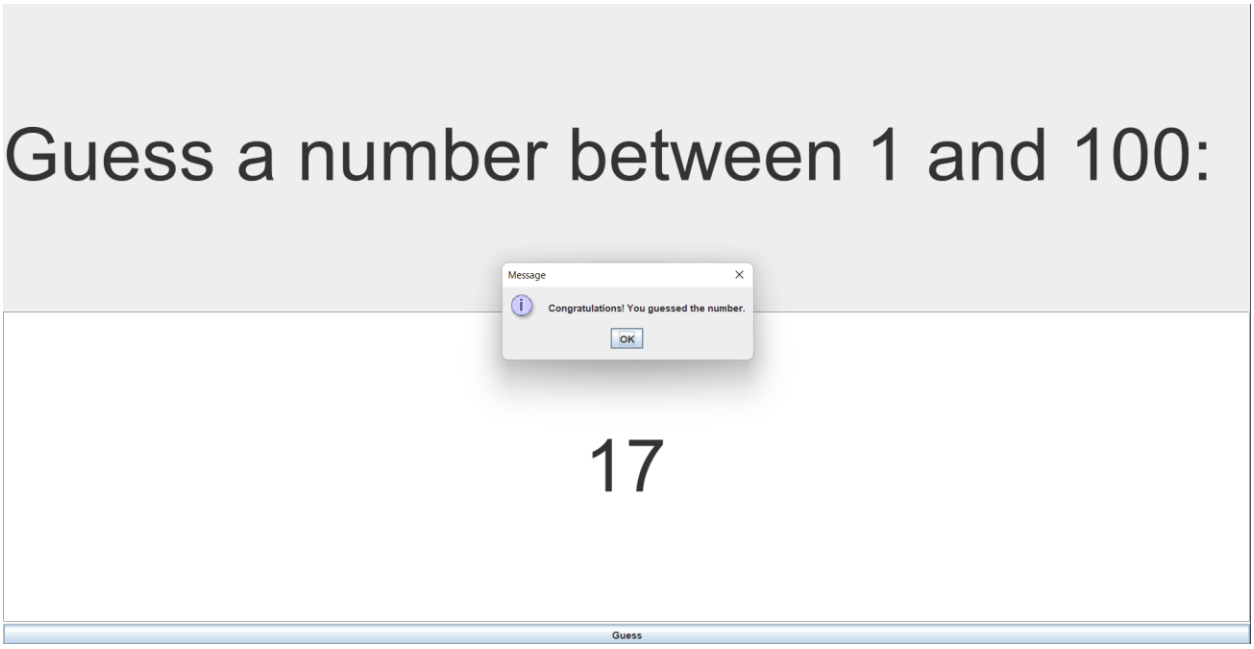
This class creates a Score object.It sets the player scores and the font, size, and color of the score text.It also contains a draw() method to draw the scores on the game panel.

PongGame:

This class contains the main() method that creates a GameFrame object and runs the Pong game.

Module 4:Number-Guessing game:-

Screenshot:-



Code:-

```
import javax.swing.*;

import java.awt.*;

import java.awt.event.*;


public class NumberGuessingGame extends JFrame implements ActionListener {

    private JLabel label;

    private JTextField textField;

    private JButton button;

    private int randomNumber;


    public NumberGuessingGame() {
```

```
    setTitle("Number Guessing Game");

    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    label = new JLabel("Guess a number between 1 and 100:");
    label.setFont(new Font("Arial", Font.PLAIN, 90));
    textField = new JTextField(10);
    textField.setFont(new Font("Arial", Font.PLAIN, 90));
    textField.setHorizontalAlignment(JTextField.CENTER);
    button = new JButton("Guess");
    button.addActionListener(this);

    randomNumber = (int) (Math.random() * 100 + 1);

    JPanel panel = new JPanel(new GridLayout(2, 1));
    panel.add(label);
    panel.add(textField);

    getContentPane().add(panel, BorderLayout.CENTER);
    getContentPane().add(button, BorderLayout.SOUTH);

    pack();
    setVisible(true);
}
```

```
    public void actionPerformed(ActionEvent e) {
        try {
            int guess = Integer.parseInt(textField.getText());

            if (guess == randomNumber) {
                JOptionPane.showMessageDialog(null, "Congratulations! You guessed the number.");
                dispose();
            } else if (guess < randomNumber) {
                JOptionPane.showMessageDialog(null, "Too low. Try again.");
            } else {
                JOptionPane.showMessageDialog(null, "Too high. Try again.");
            }
        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(null, "Invalid input. Please enter a number between 1
and 100.");
        }
    }

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

Explanation:-

This is a Java code for a simple number guessing game. It uses Swing and AWT classes to create a graphical user interface (GUI) for the game.The class NumberGuessingGame extends the JFrame class and implements the ActionListener interface. This means that it can listen for events, such as button clicks, and respond to them.The class has four private instance variables: label, textField, button, and randomNumber. The label variable is a JLabel object that displays the instructions for the game. The textField variable is a JTextField object that allows the user to enter their guess. The button variable is a JButton object that the user clicks to submit their guess. The randomNumber variable stores a random number between 1 and 100 that the user must guess.

The constructor method initializes the GUI components, sets the title of the window, and sets the default close operation. It creates a JPanel with a GridLayout and adds the label and textField components to it. It then adds the panel and the button to the content pane of the frame.The actionPerformed method is called when the user clicks the button. It first parses the text in the textField to an integer. If the user input is not a valid number, it displays an error message. If the input is a valid number, it compares it to the randomNumber. If the input is equal to randomNumber, it displays a message telling the user they guessed correctly and closes the window. If the input is less than randomNumber, it displays a message telling the user their guess was too low. If the input is greater than randomNumber, it displays a message telling the user their guess was too high.The main method

simply creates a new NumberGuessingGame object, which starts the game.

Conclusion :-

By undertaking this projects in Java, we are able to strengthen our theoretical knowledge of the programming language and get a better understanding of its practical uses.

Learning Outcome of the project:

1. Practice: The more you practice, the better you become. By working on a Java project, we got ample opportunities to practice coding in Java and improve our skills.
2. Applying Concepts: Java has several advanced concepts such as OOP, multithreading, and exception handling. Working on a project, we got a chance to apply these concepts in a practical setting and understand them better.
3. Learning New Libraries and Frameworks: Java has several libraries and frameworks that can make our coding easier and efficient. Working on a project helped us to learn new libraries and frameworks and understand how they work.
4. Debugging Skills: Debugging is an essential part of software development. By working on a Java project, we encountered several errors and bugs, and we debugged them. This helped us to improve our debugging skills and become a better developer.
5. Teamwork: Many Java projects require collaboration and teamwork. Working on a Java project helped us to improve our communication, collaboration, and teamwork skills, which are essential in the software development industry.

References:-

1. Geeks For Geeks
2. Javatoint
3. W3Schools
4. YouTube

5. Google