| Experiment No. 12 |
| --- |
| Course Project based on the content of the syllabus. |
| Date of Performance: |
| Date of Submission: |

## Code

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

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.KeyEvent; import

java.awt.event.KeyListener; import

java.util.ArrayList; import

java.util.Random;


public class SnakeGameGUI extends JPanel implements ActionListener, KeyListener

{

private static final int CELL_SIZE = 30;

private static final int BOARD_WIDTH = 25;

private static final int BOARD_HEIGHT = 25;
```

```
private static final int DELAY = 400;         private static final int INITIAL_SNAKE_LENGTH =
3;


   private ArrayList<Point> snake;
private Point food;         private
char[][] board;   private int direction;
            private int score;
private boolean gameStarted;
            private JButton startButton;
            private JButton upButton;
            private JButton leftButton;
            private JButton downButton;
            private JButton rightButton;


   public SnakeGameGUI() {        snake = new ArrayList<>();
initializeBoard();        initializeSnake();                food =
generateFood();             direction = 1;     score = 0;
gameStarted = false;      startButton = new JButton("Start");
startButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
startGame();
            }
            });
  upButton = new JButton("Up");
upButton.addActionListener(new ActionListener() {         public
void actionPerformed(ActionEvent e) { setDirection(0); // Up
```

```
        }
        });

        leftButton = new JButton("Left");
leftButton.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
setDirection(3); // Left
        }
        });

        downButton = new JButton("Down");
downButton.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
setDirection(2); // Down
        }
        });

        rightButton = new JButton("Right");
rightButton.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
setDirection(1); // Right
        }
        });
```

```
   this.add(startButton);
this.add(upButton);        this.add(leftButton);
this.add(downButton);
this.add(rightButton);


            Timer timer = new Timer(DELAY, this);
       timer.start();


            setPreferredSize(new Dimension(BOARD_WIDTH * CELL_SIZE, BOARD_HEIGHT *
CELL_SIZE));
            setFocusable(true);
addKeyListener(this);
            }


            public void keyTyped(KeyEvent e) {
            }
   public void keyPressed(KeyEvent e) {
            char key = e.getKeyChar();          if
(gameStarted) {              switch (key) {
            case 'w':

                        setDirection(0); // Up
            break;


            case 'a':
```

```
                    setDirection(3); //
Left                break;              case
's':
                    setDirection(2); //
Down                break;              case
'd':                setDirection(1); // Right
          break;              case '\n':
          startGame();                  break;
          }
          }
          }

          public void keyReleased(KeyEvent e) {
          }
          private void initializeBoard() {
          board = new char[BOARD_HEIGHT][BOARD_WIDTH];
          for (int i = 0; i < BOARD_HEIGHT; i++) {
          for (int j = 0; j < BOARD_WIDTH; j++) {
board[i][j] = 0;

          }
}
          }

   private void initializeSnake() {        for (int i = 0; i < INITIAL_SNAKE_LENGTH; i++)
{         snake.add(new Point(BOARD_WIDTH / 2 - i, BOARD_HEIGHT / 2));
          }
```

```
            }


    private Point generateFood() {
Random random = new Random();
            int x, y;
            do {
            x = random.nextInt(BOARD_WIDTH);
y = random.nextInt(BOARD_HEIGHT);
            } while (board[y][x] != 0 || snake.contains(new Point(x, y)));


        return new Point(x, y);
            }


    protected void paintComponent(Graphics g) {
            super.paintComponent(g);
drawBoard(g);

            drawFood(g);
drawSnake(g);




}


    private void drawBoard(Graphics g) {              for
(int y = 0; y < BOARD_HEIGHT; y++) {                  for
(int x = 0; x < BOARD_WIDTH; x++) {
              g.setColor(Color.WHITE);
```

```
            g.fillRect(x * CELL_SIZE, y * CELL_SIZE, CELL_SIZE, CELL_SIZE);
            g.setColor(Color.BLACK);
            g.drawRect(x * CELL_SIZE, y * CELL_SIZE, CELL_SIZE, CELL_SIZE);
          }
          }
          }
          private void drawFood(Graphics g) {
          g.setColor(Color.RED);
          int x = food.x * CELL_SIZE;
          int y = food.y * CELL_SIZE;
     g.fillRect(x, y, CELL_SIZE, CELL_SIZE);
    g.setColor(Color.WHITE);
          g.setFont(new Font("Arial", Font.PLAIN, 12));      String pointsString =
Integer.toString(score);       int pointsStringWidth =
g.getFontMetrics().stringWidth(pointsString);

     g.drawString(pointsString, x + CELL_SIZE - pointsStringWidth - 2, y +
CELL_SIZE - 2);



}


          private void drawSnake(Graphics g) {
          g.setColor(Color.GREEN);
for (Point point : snake) {          int x =
point.x * CELL_SIZE;                int y =
point.y * CELL_SIZE;
```

```
            g.fillRect(x, y, CELL_SIZE, CELL_SIZE);
              }
              }

    public void actionPerformed(ActionEvent e) {
              if (gameStarted) {          moveSnake();
              checkCollision();           repaint();
              }
              }
    private void startGame() {
gameStarted = true;
              }
              private void moveSnake() {
        Point head = snake.get(0);

              Point newHead = new Point(head.x, head.y);

              switch (direction) {
              case 0: // Up
              newHead.y--;
              break;
case 1: // Right
newHead.x++;
break;              case 2: //
Down
newHead.y++;
```

```
break;              case 3: //
Left
newHead.x--;
break;
            }
            if (newHead.equals(food)) {
            food = generateFood();
score++;            } else {
snake.remove(snake.size() - 1);
            }
  snake.add(0, newHead);
            }

            private void checkCollision() {
            Point head = snake.get(0);

            if (head.x < 0 || head.x >= BOARD_WIDTH || head.y < 0 || head.y >=
BOARD_HEIGHT) {
            gameOver();
            return;
            }           for (int i = 1; i < snake.size();
i++) {              if
(head.equals(snake.get(i))) {
gameOver();                 return;
            }
            }
            }
```

```
        private void gameOver() {
    JOptionPane.showMessageDialog(this, "Game Over. Final Score: " + score);
    System.exit(0);
        }
  private void setDirection(int newDirection) {     if
(Math.abs(newDirection - direction) != 2) {
direction = newDirection;

        }
        }

        private class Point {
        int x, y;

        Point(int x, int y) {
        this.x = x;
this.y = y;
        }
        }
        public static void main(String[] args) {
    JFrame frame = new JFrame("Snake Game");
        SnakeGameGUI snakeGameGUI = new SnakeGameGUI();
frame.add(snakeGameGUI);        frame.pack();
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(BOARD_WIDTH * CELL_SIZE, BOARD_HEIGHT * CELL_SIZE);
        frame.setLocationRelativeTo(null);
frame.setVisible(true);
```

```
        }

    }
```