REPORT

Details

Name: Ayush Maheshwari

Email: ayushm2020@gmail.com

Task Title

Snake Game Java Project Development

Task Description

Develop a simple Snake game in Java using Swing for the graphical interface. The game should include a snake that moves on a grid, consumes food, grows longer, and terminates upon collision with itself or the game borders.

Steps Taken

1. **Project Initialization:**

- Created the main class App.java to set up the game window using JFrame.
- Defined the board dimensions and set window properties such as size, title, and visibility.

Code:

```
// App.java

import javax.swing.*;

public class App {
    public static void main(String[] args) throws Exception {
        int boardWidth = 600;
        int boardHeight = boardWidth;

        JFrame frame = new JFrame("Snake");
        frame.setVisible(true);
        frame.setSize(boardWidth, boardHeight);
        frame.setLocationRelativeTo(null);
        frame.setResizable(false);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        SnakeGame snakeGame = new SnakeGame(boardWidth, boardHeight);
        frame.add(snakeGame);
        frame.pack();
        snakeGame.requestFocus();
    }
}
```

2. **SnakeGame Class Implementation:**

- Developed the SnakeGame class responsible for handling the game logic, drawing components, and managing user input.
- Utilized a nested Tile class to represent the snake body parts and food.
- Integrated a timer for continuous game updates.

Code:

```
// SnakeGame.java

import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;
import java.util.Random;
import javax.swing.*;

public class SnakeGame extends JPanel implements ActionListener, KeyListener {
    private class Tile {
        int x;
        int y;

        Tile(int x, int y) {
            this.x = x;
            this.y = y;
        }
    }

    int boardWidth;
    int boardHeight;
    int tileSize = 25;

    // snake
    Tile snakeHead;
    ArrayList<Tile> snakeBody;

    // food
    Tile food;
    Random random;

    // game logic
    int velocityX;
    int velocityY;
    Timer gameLoop;

    boolean gameOver = false;

    SnakeGame(int boardWidth, int boardHeight) {
        this.boardWidth = boardWidth;
        this.boardHeight = boardHeight;
        setPreferredSize(new Dimension(this.boardWidth, this.boardHeight));
        setBackground(Color.black);
        addKeyListener(this);
        setFocusable(true);

        snakeHead = new Tile(5, 5);
        snakeBody = new ArrayList<Tile>();

        food = new Tile(10, 10);
        random = new Random();
        placeFood();

        velocityX = 1;
        velocityY = 0;

        // game timer
        gameLoop = new Timer(100, this); // how long it takes to start timer, milliseconds
gone between frames
        gameLoop.start();
    }

    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        draw(g);
    }

    public void draw(Graphics g) {
        // Grid Lines
        for (int i = 0; i < boardWidth / tileSize; i++) {
            // (x1, y1, x2, y2)
            g.drawLine(i * tileSize, 0, i * tileSize, boardHeight);
            g.drawLine(0, i * tileSize, boardWidth, i * tileSize);
        }

        // Food
        g.setColor(Color.red);
        g.fill3DRect(food.x * tileSize, food.y * tileSize, tileSize, tileSize, true);
        // Snake Head
        g.setColor(Color.green);
        g.fill3DRect(snakeHead.x * tileSize, snakeHead.y * tileSize, tileSize, tileSize,
true);

        // Snake Body
        for (int i = 0; i < snakeBody.size(); i++) {
            Tile snakePart = snakeBody.get(i);
            g.fill3DRect(snakePart.x * tileSize, snakePart.y * tileSize, tileSize, tileSize,
true);
        }

        // Score
        g.setFont(new Font("Arial", Font.PLAIN, 16));
        if (gameOver) {
            g.setColor(Color.red);
            g.drawString("Game Over: " + String.valueOf(snakeBody.size()), tileSize - 16,
tileSize);
        } else {
            g.drawString("Score: " + String.valueOf(snakeBody.size()), tileSize - 16,
tileSize);
        }
    }

    public void placeFood() {
        food.x = random.nextInt(boardWidth / tileSize);
        food.y = random.nextInt(boardHeight / tileSize);
    }

    public void move() {
        // eat food
        if (collision(snakeHead, food)) {
            snakeBody.add(new Tile(food.x, food.y));
            placeFood();
        }

        // move snake body
        for (int i = snakeBody.size() - 1; i >= 0; i--) {
            Tile snakePart = snakeBody.get(i);
            if (i == 0) { // right before the head
                snakePart.x = snakeHead.x;
                snakePart.y = snakeHead.y;
            } else {
                Tile prevSnakePart = snakeBody.get(i - 1);
                snakePart.x = prevSnakePart.x;
                snakePart.y = prevSnakePart.y;
            }
        }
        // move snake head
        snakeHead.x += velocityX;
        snakeHead.y += velocityY;

        // game over conditions
        for (int i = 0; i < snakeBody.size(); i++) {
            Tile snakePart = snakeBody.get(i);

            // collide with snake head
            if (collision(snakeHead, snakePart)) {
                gameOver = true;
            }
        }

        if (snakeHead.x * tileSize < 0 || snakeHead.x * tileSize > boardWidth || // passed
left border or right border
                snakeHead.y * tileSize < 0 || snakeHead.y * tileSize > boardHeight) { //
passed top border or bottom
                                                                                      // border
            gameOver = true;
        }
    }

    public boolean collision(Tile tile1, Tile tile2) {
        return tile1.x == tile2.x && tile1.y == tile2.y;
    }

    @Override
    public void actionPerformed(ActionEvent e) { // called every x milliseconds by
gameLoop timer
        move();
        repaint();
        if (gameOver) {
            gameLoop.stop();
        }
    }

    @Override
    public void keyPressed(KeyEvent e) {
        if (e.getKeyCode() == KeyEvent.VK_UP && velocityY != 1) {
            velocityX = 0;
            velocityY = -1;
        } else if (e.getKeyCode() == KeyEvent.VK_DOWN && velocityY != -1) {
            velocityX = 0;
            velocityY = 1;
        } else if (e.getKeyCode() == KeyEvent.VK_LEFT && velocityX != 1) {
            velocityX = -1;
            velocityY = 0;
        } else if (e.getKeyCode() == KeyEvent.VK_RIGHT && velocityX != -1) {
            velocityX = 1;
            velocityY = 0;
        }
    }

    @Override
    public void keyTyped(KeyEvent e) {
    }

    @Override
    public void keyReleased(KeyEvent e) {
    }
}
```

3. **Drawing and Rendering:**

- Implemented the paintComponent and draw methods to draw grid lines, snake, food, and the score on the game window.

  Code:

```
// SnakeGame.java (continued)

public void paintComponent(Graphics g) {
    super.paintComponent(g);
    draw(g);
}

public void draw(Graphics g) {
    // Grid Lines
    for (int i = 0; i < boardWidth / tileSize; i++) {
        g.drawLine(i * tileSize, 0, i * tileSize, boardHeight);
        g.drawLine(0, i * tileSize, boardWidth, i * tileSize);
    }

    // Food
    g.setColor(Color.red);
    g.fill3DRect(food.x * tileSize, food.y * tileSize, tileSize, tileSize, true);
    // Snake Head
    g.setColor(Color.green);
    g.fill3DRect(snakeHead.x * tileSize, snakeHead.y * tileSize, tileSize, tileSize, true);

    // Snake Body
    for (int i = 0; i < snakeBody.size(); i++) {
        Tile snakePart = snakeBody.get(i);
        g.fill3DRect(snakePart.x * tileSize, snakePart.y * tileSize, tileSize, tileSize, true);
    }

    // Score
    g.setFont(new Font("Arial", Font.PLAIN, 16));
    if (gameOver) {
        g.setColor(Color.red);
        g.drawString("Game Over: " + String.valueOf(snakeBody.size()), tileSize - 16,
tileSize);
    } else {
        g.drawString("Score: " + String.valueOf(snakeBody.size()), tileSize - 16,
tileSize);
    }
}
```

4. **Game Logic:**

- Implemented methods for moving the snake, handling collisions, updating the score, and determining game over conditions.
- Incorporated logic for the snake to consume food and grow in length.

Code:

```
// SnakeGame.java (continued)

public void placeFood() {
    food.x = random.nextInt(boardWidth / tileSize);
    food.y = random.nextInt(boardHeight / tileSize);
}

public void move() {
    // eat food
    if (collision(snakeHead, food)) {
        snakeBody.add(new Tile(food.x, food.y));
        placeFood();
    }

    // move snake body
```

```
        for (int i = snakeBody.size() - 1; i >= 0; i--) {
            Tile snakePart = snakeBody.get(i);
            if (i == 0) { // right before the head
                snakePart.x = snakeHead.x;
                snakePart.y = snakeHead.y;
            } else {
                Tile prevSnakePart = snakeBody.get(i - 1);
                snakePart.x = prevSnakePart.x;
                snakePart.y = prevSnakePart.y;
            }
        }
        // move snake head
        snakeHead.x += velocityX;
        snakeHead.y += velocityY;

        // game over conditions
        for (int i = 0; i < snakeBody.size(); i++) {
            Tile snakePart = snakeBody.get(i);

            // collide with snake head
            if (collision(snakeHead, snakePart)) {
                gameOver = true;
            }
        }

        if (snakeHead.x * tileSize < 0 || snakeHead.x * tileSize > boardWidth ||
            snakeHead.y * tileSize < 0 || snakeHead.y * tileSize > boardHeight) {
            gameOver = true;
        }
    }

    public boolean collision(Tile tile1, Tile tile2) {
        return tile1.x == tile2.x && tile1.y == tile2.y;
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        move();
        repaint();
        if (gameOver) {
            gameLoop.stop();
        }
    }
```

5. **User Input Handling:**

- Responded to keyboard input using the KeyListener interface to control the snake's direction.

  Code:

```
// SnakeGame.java (continued)

@Override
public void keyPressed(KeyEvent e) {
   if (e.getKeyCode() == KeyEvent.VK_UP && velocityY != 1) {
      velocityX = 0;
      velocityY = -1;
   } else if (e.getKeyCode() == KeyEvent.VK_DOWN && velocityY != -1) {
      velocityX = 0;
      velocityY = 1;
   } else if (e.getKeyCode() == KeyEvent.VK_LEFT && velocityX != 1) {
      velocityX = -1;
      velocityY = 0;
   } else if (e.getKeyCode() == KeyEvent.VK_RIGHT && velocityX != -1) {
      velocityX = 1;
      velocityY = 0;
   }
}

@Override
public void keyTyped(KeyEvent e) {
}

@Override
public void keyReleased(KeyEvent e) {
}
```

Challenges Faced

- Ensuring smooth movement and rendering of the snake and food on the grid.
- Managing the game loop and updating the display at regular intervals.
- Implementing collision detection for both self-collision and border collision.

Solutions Implemented

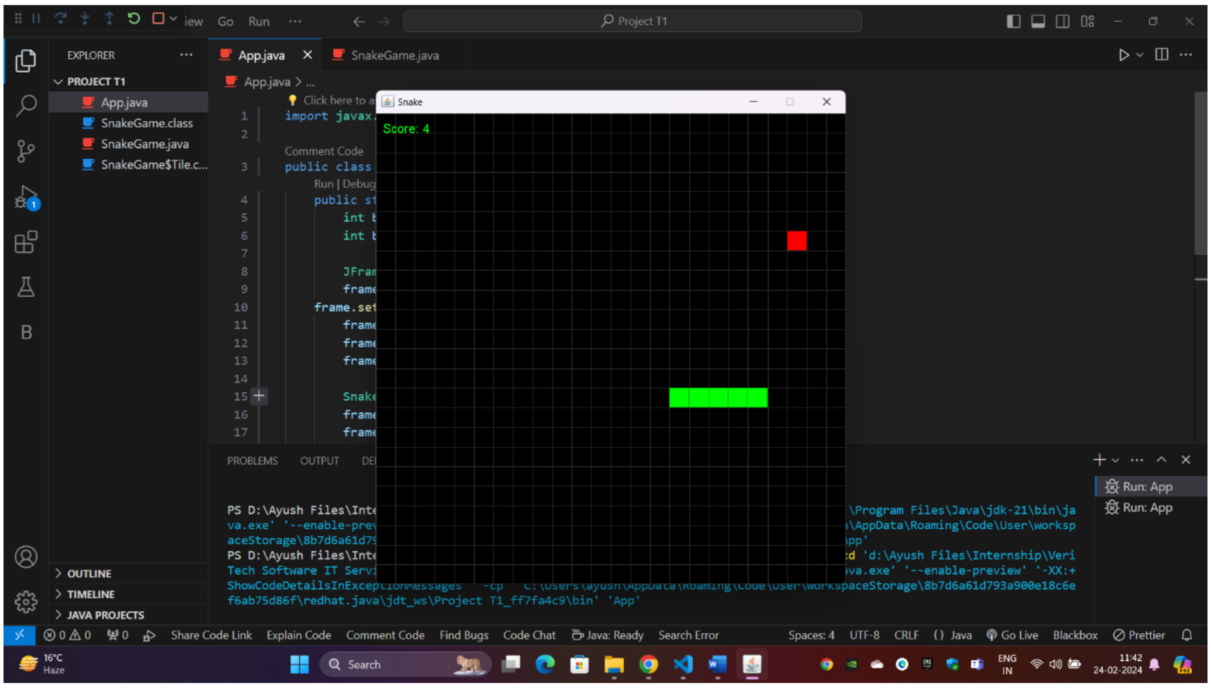
- Utilized the Swing library for graphical rendering and event handling.
- Incorporated a timer-based game loop to update the game state at regular intervals.
- Implemented collision detection logic to handle various scenarios.

Learnings

- Gain proficiency in using Java Swing for graphical user interface development.
- Understand the implementation of game loops for continuous gameplay updates.
- Enhance skills in managing user input and responding to keyboard events.

Project Update:

The Snake game project has been successfully implemented and tested. The game includes features such as snake movement, food consumption, and proper collision handling. The code has been organized into separate classes for better maintainability and readability. The project can be further extended with additional features, such as score tracking, levels, or enhanced graphics.



```
import javax.

public class
    public st
        int t
        int t

        JFram
        frame
    frame.set
        frame
        frame
        frame

        Snake
        frame
        frame
```

```
PS D:\Ayush Files\Inte                                    \Program Files\Java\jdk-21\bin\ja
va.exe' '--enable-prev                                    \AppData\Roaming\Code\User\worksp
aceStorage\8b7d6a61d79                                    pp'
PS D:\Ayush Files\Inte                                    d 'd:\Ayush Files\Internship\Veri
Tech Software IT Servi                                    va.exe' '--enable-preview' '-XX:+
ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\ayush\AppData\Roaming\Code\User\workspaceStorage\8b7d6a61d793a900e18c6e
f6ab75d86f\redhat.java\jdt_ws\Project T1_ff7fa4c9\bin' 'App'
```