**DEPARTMENT OF COMPUTER APPLICATIONS**

National Institute of Technology Kurukshetra Haryana, India



**PROJECT TITLE :-** Snake Game

**Submitted By: Submitted to:**

Abhishek :- 523410016  **Dr. Kapil**

Ritik Kumar :- 523410010 **( Assistant Professor)**

Devendra :- 523110065

# OOP Game Report

**I.Game Rules :**

The player controls a dot, square, or object on a bordered plane. As it moves forward, it leaves a trail behind, resembling a moving snake. In some games, the end of the trail is in a fixed position, so the snake continually gets longer as it moves. In another common scheme, the snake has a specific length, so there is a moving tail a fixed number of units away from the head.

Eating food makes the snake grow. When the food is eaten it moves to another random position.

The snake will wrap around to the other side of the screen when it goes off the edge.

The game is over when the snake crashes into itself and the screen border.

**II.Diagram and Design Explanation:**

**Main package :-**

**Snake Class():** Contains main() method this is method that running first when run code. Here we create object of GameFrame class.

**GameFrame Class** **:** class that create Frame for game . This class will extends javax.swing.JFrame to create Frame and add Panel on that Frame.

**GamePanel Class :-** GamePanel class that include all method to control snake. This class will be implements java.awt.event.ActionListener to peform Action from keyboard and KeyListner.

1. **GamePanel():** constructor that use Some method to create and set the panel.
   * Random():creates object of random class mouse.
   * GameMenu(): create object of GameMenu class
   * MouseInput(): create object of MouseInput class
2. **startGame():** 
   * Call newSmallMouse()
   * Call newBigMouse()
   * Create object of Timer class that is used to update frame based on delay.
3. **paintComponent(Graphics g):** 
   * paintComponent(g): extends from awt.Graphics to draw components.
   * draw(g): call method draw(g).
4. **draw(Graphics g):** 
   * if(running): based on condition and draw small or big mouse and draw snake body.
   * if(!running&&!gameEnd) : render to game menu
   * else: call gameOver(g) this method will draw game over screen when snake stop .
5. **newSmallMouse():** this method that create next new Small Mouse according units size & coordinate XY.
6. **newBigMouse():** this method that create next new Big Mouse according units size & coordinate XY.
7. **moveSnake():** this method that help snake follow 4 directions(UP, DOWN, LEFT, RIGHT):
   * switch case(): every case will have different direction.
8. **checkMouse():** this method have tasks are check, count the number of eaten mouse and length of bodyParts
   * If X,Y coordinate of smallmouse the same position of head snake: this smallMouse will change to 1 unit of bodyParts, score will be plus +1.
   * newSmallMouse(): called from newSmallMouse() method for new coordinates of Small Mouse
   * If X,Y coordinate of Bigmouse the same position of head snake: this BigMouse will change to 1 unit of bodyParts, score will be plus +4 and newBigMouse.
   * newBigMouse(): called from newBigMouse() methodfor new coordinates of Big Mouse
9. **checkCollision():** this method will check when case that makes game stop

* + When head touch any part of snake the game will stop.
  + other hand we check for gamemode according to game mode

easy and medium snake can pass from border but in hard level snake cannot touch border.

1. **gameOver(Graphics g):** this method will show score that player
   * Some draw method of awt.Graphics.
2. **actionPerformed(ActionEvent e):** this is method that perform actions when snake eat mouse collision with body This is the method of awt.events.ActionListenerinterface
   * call moveSnake() method.
   * call checkMouse() method.
   * call checkcollision() method.

11**keyPressed(KeyEvent e):** this implements from awt.event.KeyListner to listen and active events is control direction of snake when we press the keyboard.

* + Switch case for 4 direction.

**Input Package :-**

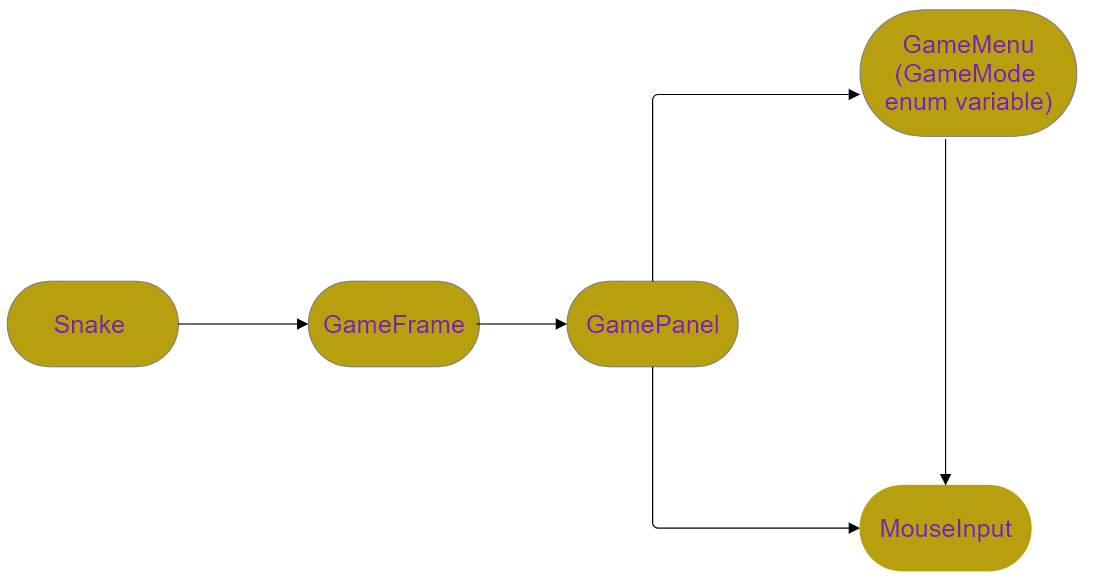
**MouseInput():-** this class extends GameMenu class and implements abstract method of MouseListner and MouseMotionListner .

* mousePressed(MouseEvent e): this method is used to take game menu from GameMenu and set level to according to level input call startGame() method   
  set running variable true to start game.

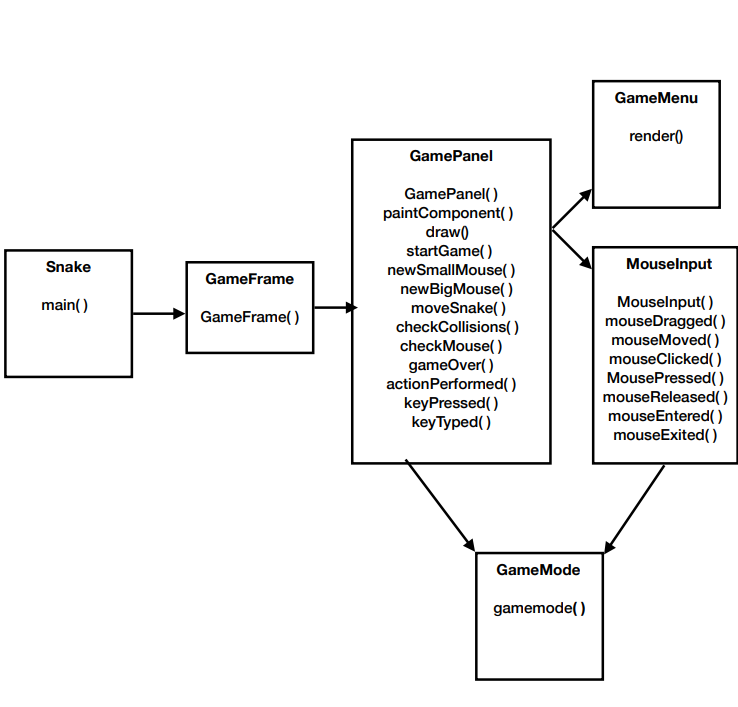
**State Package() :-** In this Package we create GameMenu class that is   
responsible for draw menu on frame . In this class we an enum data type  
for different game mode or game Level.

* GameMenu .
* enum data type GameMode.

**Code Flow :-**

****

**UML Diagram :-**

****

**Code : - Snake.java**

**package** Main;

**public** **class** Snake {

**public** **static** **void** main(String[] args) {

**new** GameFrame();// create object of GameFrame and calling constructor GameFrame

}

}

**GameFrame.Java**

**package** Main;

**import** javax.swing.JFrame;

**public** **class** GameFrame **extends** JFrame {

GamePanel gamePnl = **new** GamePanel();

GameFrame() {

**this**.add(gamePnl);

**this**.setTitle("Snake");

**this**.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);// default closer when we click on x sign on frame.

**this**.setResizable(**false**);

**this**.pack();

**this**.setVisible(**true**);// make frame visible.

**this**.setLocationRelativeTo(**null**); // window to appear on middle of our screen.

}

}

**GamePanel.java:**

**package** Main;

**import** java.awt.Color;

**import** java.awt.Dimension;

**import** java.awt.Font;

**import** java.awt.Graphics;

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

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

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

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

**import** java.util.Random;

**import** javax.swing.JPanel;

**import** javax.swing.Timer;

**import** Inputs.MouseInput;

**import** State.GameMode;

**import** State.GameMenu;

**public** **class** GamePanel **extends** JPanel **implements** ActionListener, KeyListener {

**private** **static** **final** **int** ***screenWidth*** = 600;

**private** **static** **final** **int** ***screenHeight*** = 600;

**static** **final** **int** ***unitSize*** = 25;// unit of each block on Panel;

**private** **static** **final** **int** ***gameUnits*** = (***screenWidth*** \* ***screenHeight***) / ***unitSize***;

**public** **static** **int** *delay* = 100;// this delay will use to update frame

**int** snakeX[] = **new** **int**[***gameUnits***]; // x coordinate of body of the snake

**int** snakeY[] = **new** **int**[***gameUnits***]; // y coordinate of body of the snake

**private** **int** snakeBodyParts = 4; // initial length of snake it increase when snake eat mouse

**private** **int** score = 0;

**private** **int** mouseX;

**private** **int** mouseY; // random position of small mouse

**private** **int** bigMouseX;

**private** **int** bigMouseY; // random position of big mouse

**private** **int** countMouse = 0;

**private** **char** direction = 'R'; // snake begin to flow in right direction!

**public** **boolean** running = **false**;// static because we access it in

**private** **boolean** gameEnd = **false**;

**private** Timer timer;

**private** Random random; // To generate random number

**private** GameMenu menu;// object of GameMenu class

**private** MouseInput mouseinputs; // object of MouseInput class

GamePanel() {

menu = **new** GameMenu();

random = **new** Random();

setPreferredSize(**new** Dimension(***screenWidth***, ***screenHeight***));

mouseinputs = **new** MouseInput(**this**);

addMouseListener(mouseinputs);// take input of mouse

addMouseMotionListener(mouseinputs);

setBackground(**new** Color(160, 32, 240)); // To set BGC

setFocusable(**true**); // By default , It is false we have set true

addKeyListener(**this**);// For keyBoardInput

}

**public** **void** paintComponent(Graphics g) {

**super**.paintComponent(g);

draw(g);

}

// Method for draw Snake.

**public** **void** draw(Graphics g) {

**if** (!running && !gameEnd) {

menu.render(g);

}

**else** **if** (running && !gameEnd) {

**if** (countMouse == 5) {

// draw a bigMouse

g.setColor(Color.***RED***);

g.fillOval(bigMouseX, bigMouseY, ***unitSize*** + 4, ***unitSize*** + 4);

}

// draw small mouse

g.setColor(Color.***PINK***);

g.fillOval(mouseX, mouseY, ***unitSize***, ***unitSize***);

**for** (**int** i = 1; i < snakeBodyParts; i++) {

g.setColor(**new** Color(255, 255, 0)); // color of snake body

g.fillOval(snakeX[i], snakeY[i], ***unitSize***, ***unitSize***); // For Shape of Snake

}

g.setColor(Color.***RED***);// color of snake head

g.fillOval(snakeX[0], snakeY[0], ***unitSize***, ***unitSize***);

g.setColor(Color.***WHITE***);

g.setFont(**new** Font("Times New Roman", Font.***BOLD***, 40));

g.drawString("Score: " + score, ***unitSize***, ***unitSize***);

} **else** {

gameOver(g);

}

}

**public** **void** startGame() {

newBigMouse();// generate coordinates of big mouse

newSmallMouse();// generate coordinates of small mouse

timer = **new** Timer(*delay*, **this**); // Manage speed WRT to time

timer.start();

}

**public** **void** newBigMouse() {

// generate coordinates of big mouse

bigMouseX = random.nextInt((**int**) (***screenWidth*** / ***unitSize***)) \* ***unitSize***;

bigMouseY = random.nextInt((**int**) (***screenHeight*** / ***unitSize***)) \* ***unitSize***;

}

**public** **void** newSmallMouse() {

// generate coordinates of small mouse

mouseX = random.nextInt((**int**) (***screenWidth*** / ***unitSize***)) \* ***unitSize***;

mouseY = random.nextInt((**int**) (***screenHeight*** / ***unitSize***)) \* ***unitSize***;

}

**public** **void** moveSnake() {

**for** (**int** i = snakeBodyParts; i > 0; i--) {

snakeX[i] = snakeX[i - 1];

snakeY[i] = snakeY[i - 1];

}

**switch** (direction) {

**case** 'U':

snakeY[0] = snakeY[0] - ***unitSize***;

**break**;

**case** 'D':

snakeY[0] = snakeY[0] + ***unitSize***;

**break**;

**case** 'L':

snakeX[0] = snakeX[0] - ***unitSize***;

**break**;

**case** 'R':

snakeX[0] = snakeX[0] + ***unitSize***;

**break**;

}

}

**public** **void** checkCollisions() {

// head touches body of the snake!

**for** (**int** i = snakeBodyParts; i > 0; i--) {

**if** ((snakeX[0] == snakeX[i] && snakeY[0] == snakeY[i])) {

running = **false**;

gameEnd = **true**;

}

}

**if** (GameMode.*state* == GameMode.***EASY*** || GameMode.*state* == GameMode.***MEDIUM***) {

// if snake go inside of any one border then it will comes from opposite border

**if** (snakeX[0] < 0) {

// running = false;

snakeX[0] = ***screenWidth***;

}

**if** (snakeX[0] > ***screenWidth***) {

snakeX[0] = 0;

// running = false;

}

**if** (snakeY[0] < 0) {

snakeY[0] = ***screenWidth***;

// running = false;

}

**if** (snakeY[0] > ***screenHeight***) {

snakeY[0] = 0;

// running = false;

}

} **else** **if** (GameMode.*state* == GameMode.***HARD***) {

**if** (snakeX[0] < 0) {

running = **false**;

gameEnd = **true**;

}

**if** (snakeX[0] > ***screenWidth***) {

running = **false**;

gameEnd = **true**;

}

**if** (snakeY[0] < 0) {

running = **false**;

gameEnd = **true**;

}

**if** (snakeY[0] > ***screenHeight***) {

running = **false**;

gameEnd = **true**;

}

}

}

**private** **void** checkMouse() {

**if** ((snakeX[0] == mouseX) && (snakeY[0] == mouseY)) {

**if** (countMouse == 5)

countMouse = 0;

snakeBodyParts++;

score++;

countMouse++;

newSmallMouse();

} **else** **if** ((countMouse == 5) && (snakeX[0] == bigMouseX) && (snakeY[0] == bigMouseY)) {

countMouse = 0;

snakeBodyParts++;

score += 4;

newBigMouse();

}

}

**public** **void** gameOver(Graphics g) {

// score text

g.setColor(Color.***WHITE***);

g.setFont(**new** Font("Times New Roman", Font.***BOLD***, 40));

g.drawString("Score: " + score, ***unitSize***, ***unitSize***);

// game over text

g.setColor(Color.***WHITE***);

g.setFont(**new** Font("Times New Roman", Font.***BOLD***, 75));

g.drawString("Game Over", ***screenWidth*** / 5, ***screenHeight*** / 2);

}

**public** **void** actionPerformed(ActionEvent e) {

**if** (running && !gameEnd) {

moveSnake();// move snake or change direction if any direction key pressed

checkMouse();

checkCollisions(); // check snake head touch his body if touch then game over

}

repaint();

**if** (gameEnd) {

timer.stop();

}

}

**public** **void** keyPressed(KeyEvent e) {

**switch** (e.getKeyCode()) {

**case** KeyEvent.***VK\_LEFT***:

**if** (direction != 'R') {

direction = 'L';

}

**break**;

**case** KeyEvent.***VK\_RIGHT***:

**if** (direction != 'L') {

direction = 'R';

}

**break**;

**case** KeyEvent.***VK\_UP***:

**if** (direction != 'D') {

direction = 'U';

}

**break**;

**case** KeyEvent.***VK\_DOWN***:

**if** (direction != 'U') {

direction = 'D';

}

**break**;

}

}

**public** **void** keyTyped(KeyEvent e) {

// **TODO** Auto-generated method stub

}

**public** **void** keyReleased(KeyEvent e) {

// **TODO** Auto-generated method stub

}

}

**State Package**

**GameMenu.java**

**package** State;

**import** java.awt.Graphics;

**import** java.awt.\*;

**public** **class** GameMenu {

**protected** **int** height = 50;// height of buttons.

**protected** **int** width = 150;// width of buttons.

**protected** Rectangle EASY = **new** Rectangle(200, 200, width, height);

**protected** Rectangle MEDIUM = **new** Rectangle(200, 270, width, height);

**protected** Rectangle HARD = **new** Rectangle(200, 340, width, height);

**protected** Rectangle quitButton = **new** Rectangle(200, 450, width, height);

**public** **void** render(Graphics g) {

Graphics2D g2d = (Graphics2D) g; // object of Graphics2D class that used to draw rectangle.

Font font = **new** Font("Bradley Hand ITC", Font.***BOLD***, 28);

g.setFont(font);

g.setColor(Color.***WHITE***);

g.drawString("CHOOSE GAME LEVEL ", 30, 150);

Font lvlFont = **new** Font("Comic Sans MS", Font.***BOLD***, 30);

g.setFont(lvlFont);

g.setColor(Color.***YELLOW***);

g.drawString("EASY", EASY.x + EASY.x / 10, EASY.y + EASY.y / 5);

g.drawString("MEDIUM", MEDIUM.x + MEDIUM.x / 15, MEDIUM.y + MEDIUM.y / 10 + 10);

g.drawString("HARD", HARD.x + HARD.x / 10, HARD.y + HARD.y / 10);

g2d.draw(EASY);

g2d.draw(MEDIUM);

g2d.draw(HARD);

Font Qfont = **new** Font("Comic Sans MS", Font.***BOLD***, 30);

g.setColor(Color.***black***);

g.setFont(Qfont);

g.drawString("Quit", quitButton.x + quitButton.x / 5, quitButton.y + quitButton.y / 10);

g2d.draw(quitButton);

}

}

**GameMode.java**

**package State;**

**public enum GameMode {**

***MENU*, *EASY*, *MEDIUM*, *HARD*;// Constants for Game Modes.**

**public static GameMode *state* = GameMode.*MENU*;// Initially Game Mode is MENU.**

**}**

**Input Package**

**MouseInput.java**

**package Inputs;**

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

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

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

**import Main.GamePanel;**

**import State.GameMenu;**

**import State.GameMode;**

**public class MouseInput extends GameMenu implements MouseListener, MouseMotionListener {**

**private GamePanel gamePnl;**

**public MouseInput(GamePanel gamePnl) {**

**this.gamePnl = gamePnl;**

**}**

**@Override**

**public void mouseDragged(MouseEvent e) {**

**// TODO Auto-generated method stub**

**}**

**@Override**

**public void mouseMoved(MouseEvent e) {**

**// TODO Auto-generated method stub**

**}**

**@Override**

**public void mouseClicked(MouseEvent e) {**

**// TODO Auto-generated method stub**

**}**

**@Override**

**public void mousePressed(MouseEvent e) {**

**int xPos = e.getX();**

**int yPos = e.getY();**

**// System.out.println("Easy");**

**// For PlayButton**

**if (xPos >= 200 && xPos <= 200 + 150) {// x position for all button is same but y is different so according to**

**// different y different gameMode set**

**if (yPos >= 200 && yPos <= 240) {**

**GamePanel.*delay* = 100;**

**gamePnl.running = true;// set running true mean game start now**

**System.*out*.println("Easy");**

**GameMode.*state* = GameMode.*EASY*;**

**gamePnl.startGame();// starting of game**

**} else if (yPos >= 270 && yPos <= 300) {**

**System.*out*.println("MEDIUM");**

**GamePanel.*delay* = 75;**

**gamePnl.running = true;**

**GameMode.*state* = GameMode.*MEDIUM*;**

**gamePnl.startGame();**

**} else if (yPos >= 340 && yPos <= 370) {**

**System.*out*.println("HARD");**

**GamePanel.*delay* = 75;**

**gamePnl.running = true;**

**GameMode.*state* = GameMode.*HARD*;**

**gamePnl.startGame();**

**} else if (yPos >= 450 && yPos <= 500) {**

**System.*exit*(0);**

**}**

**}**

**}**

**@Override**

**public void mouseReleased(MouseEvent e) {**

**// TODO Auto-generated method stub**

**}**

**@Override**

**public void mouseEntered(MouseEvent e) {**

**// TODO Auto-generated method stub**

**}**

**@Override**

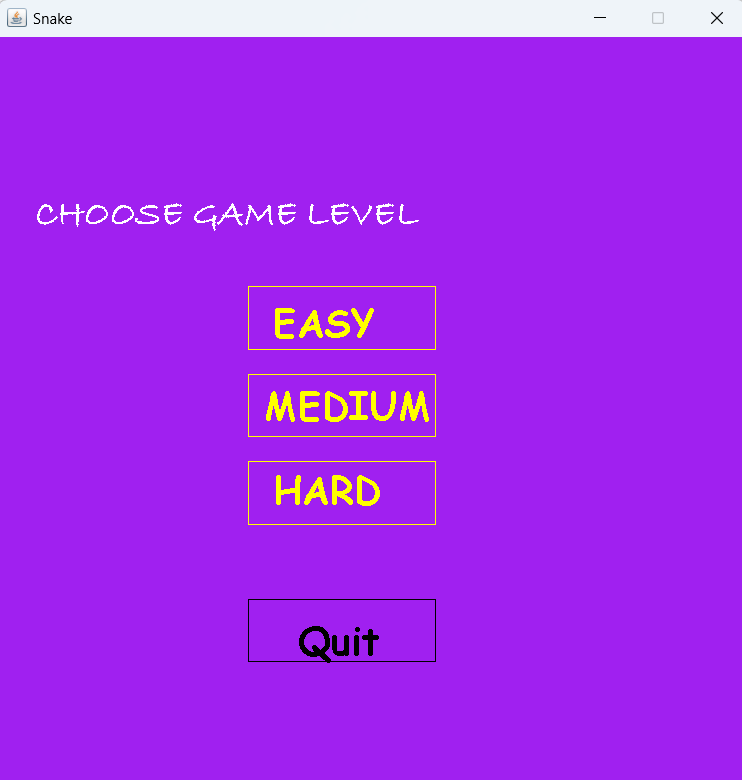
**public void mouseExited(MouseEvent e) {**

**// TODO Auto-generated method stub**

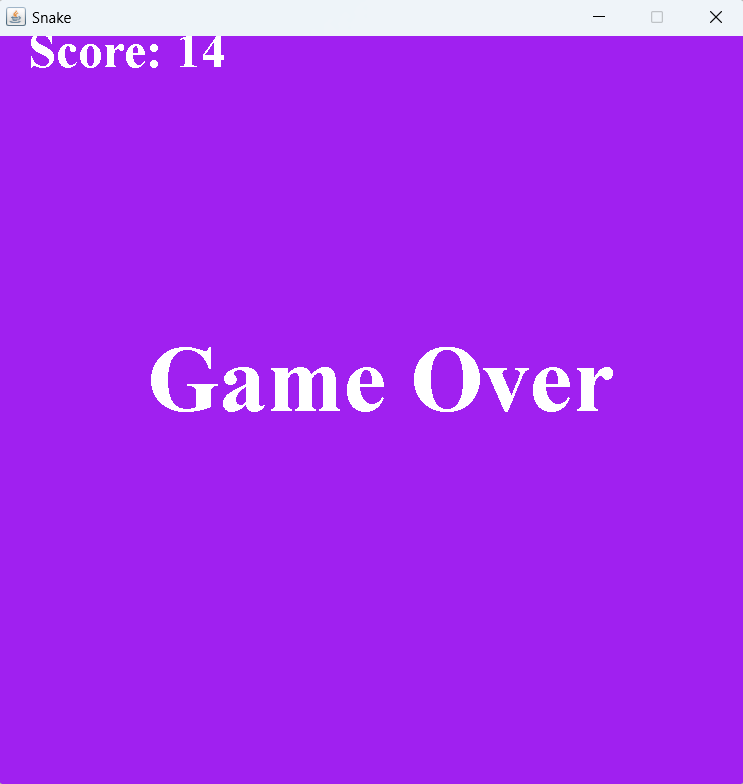
**}**

**}**

**Output:**

****

****

****