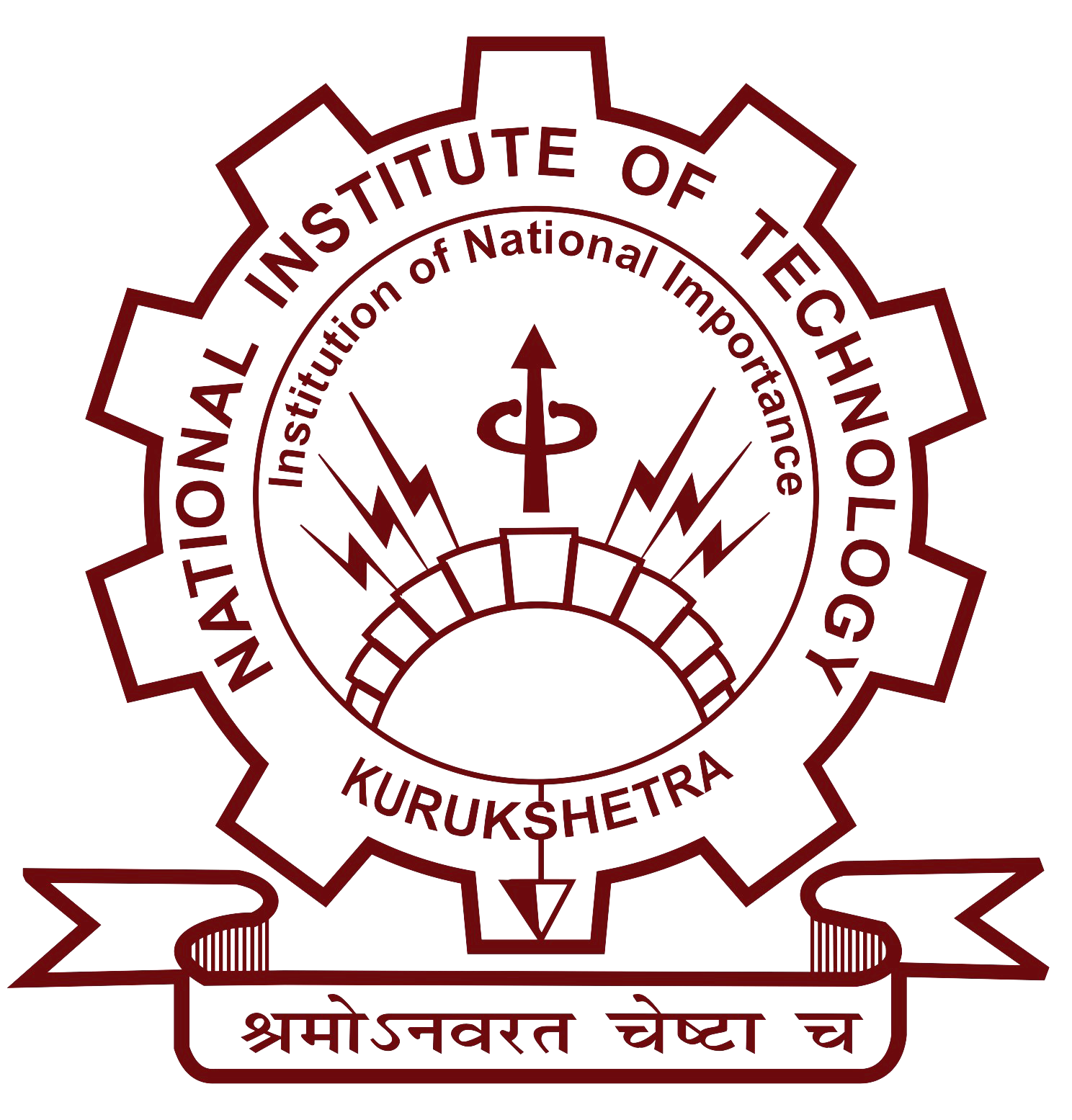
**DEPARTMENT OF COMPUTER APPLICATIONS**

National Institute of Technology Kurukshetra Haryana, India



**OOP’s(JAVA) PROJECT**

**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:**

**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.

1. **GamePanel():** constructor that use Some method to create and set the panel.
   * Random():creates object of random class mouse.
   * startGame(): this method to start game.
2. **startGame():** 
   * Call newSmallMouse()
   * Call newBigMouse()
   * running = true: the snake will travel.
   * 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): create objects of NormalMouse class or BigMouse class based on condition and draw small or big mouse and draw snake body.
   * 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. Make newSmallMouse.
   * 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() method. Make newBigMouse.
9. **checkCollision():** this method will check when case that makes game stop

* + When head touch any part of snake the game will stop.

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.**MyKeyAdapter Class:** this Inner class extends from awt.event.KeyAdapter to listen and active events is control direction of snake when we press the keyboard.

* + keyPressed(KeyEvent e): this method catch the action after press keyboard.
  + Switch case for 4 direction.

**NormalMouse Class:** This class extend JPanel and implement abstract method of DrawMouse interface.

* + **NormalMouse(GamePanel game):** constructor set object of game panel.
  + **paintComponent(Graphics g):**
  + paintComponent(g): extends from awt.Graphics to draw components.
  + draw(g): call method draw(g).
  + **draw(Graphics g):** draw a small mouse.
  + **checkMouse():**this method have tasks are check, count the number of eaten mouse and length of bodyParts
  + If X,Y coordinate of small mouse the same position of head snake: this smallMouse will change to 1 unit of bodyParts, score will be plus +1 and newSmallMouse.
  + newSmallMouse(): called from newSmallMouse() method. Make newSmallMouse after sum number of them to bodyParts.

**BigMouse Class:** This class extend JPanel and implement abstract method of DrawMouse interface.

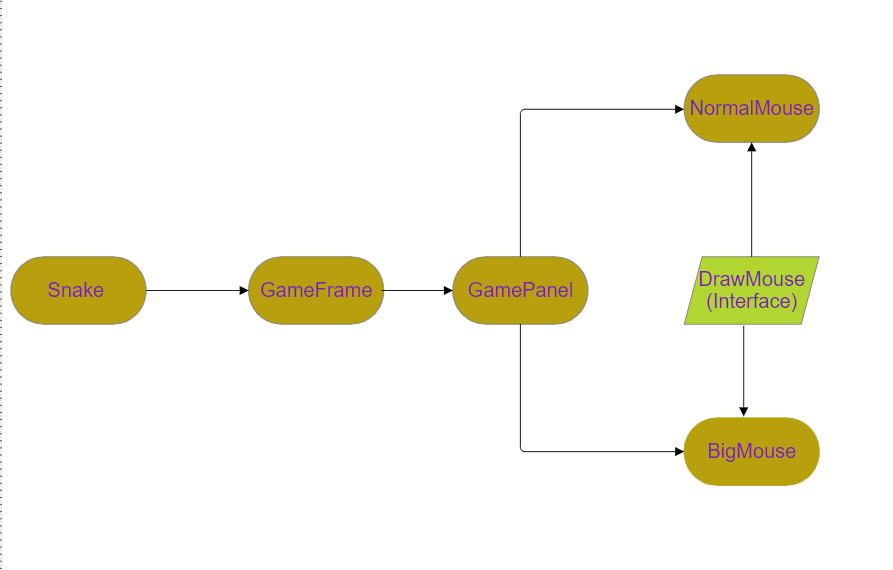
* + **BigMouse():** constructor set object of game panel.
  + **paintComponent(Graphics g):**
  + paintComponent(g): extends from awt.Graphics to draw components.
  + draw(g): call method draw(g).
  + **draw(Graphics g):** draw a small mouse.
  + **checkMouse():**this method have tasks are check, count the number of eaten mouse and length of bodyParts
  + 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() method. Make newBigMouse.

**DrawMouse Interface :-** have two method

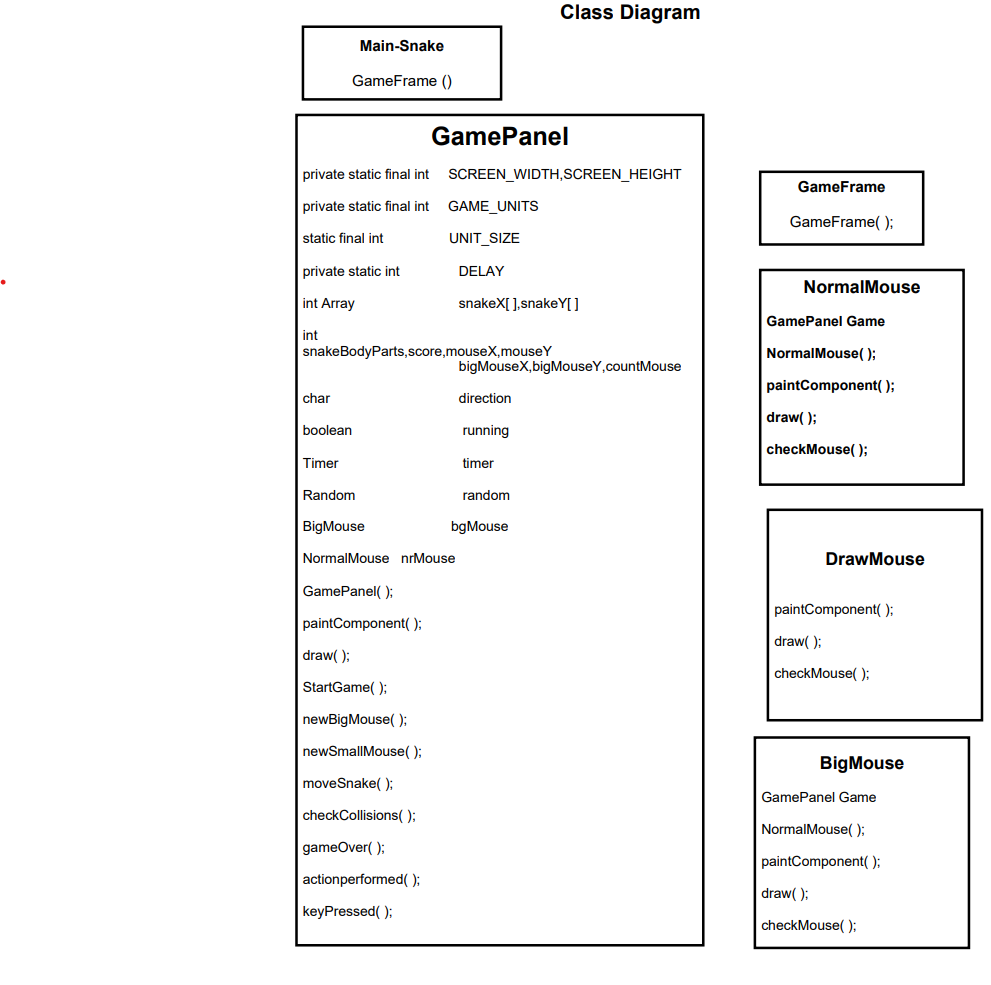
draw(Graphic g) : - use to draw mouse.

checkMouse () :- use to check snake head is on mouse coordinates or not.

**Code Flow :-**

****

**UML Diagram :-**



**Code : - Snake.java**

package Main;

public class Snake {

public static void main(String[] args) {

new GameFrame();// craeting object of GameFrame class and calling constructor GameFrame()

}

}

**GameFrame.Java**

package Main;

import javax.swing.JFrame;

public class GameFrame extends JFrame {

GameFrame() {

this.add(new GamePanel());// add panel on frame

this.setTitle("Snake");

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);// close frame or terminate game

this.setResizable(false);

this.pack();

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

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.FontMetrics;

import java.awt.Graphics;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.KeyAdapter;

import java.awt.event.KeyEvent;

import java.util.Random;

import javax.swing.JPanel;

import javax.swing.Timer;

public class GamePanel extends JPanel implements ActionListener {

private static final int SCREEN\_WIDTH = 600;

private static final int SCREEN\_HEIGHT = 600;

static final int UNIT\_SIZE = 25;

private static final int GAME\_UNITS = (SCREEN\_WIDTH \* SCREEN\_HEIGHT) / UNIT\_SIZE;

private static int DELAY = 100;// use to update frame

final int snakeX[] = new int[GAME\_UNITS]; // x cordinate of body of the snake

final int snakeY[] = new int[GAME\_UNITS]; // y coordinate of body of the snake

int snakeBodyParts = 4;// intial body size of snake is 4 it is increase when snake eat mouse

int score = 0;

int mouseX;

int mouseY; // random position of normal mouse

int bigMouseX;

int bigMouseY;// random position of big mouse

int countMouse = 0;// counting for its time to draw a big mouse or not

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

boolean running = false;

Timer timer;// object of Timer class use in gameloop

Random random; // To generate random number that use to find new cordinates of mouse

BigMouse bgMouse;// object of BigMouse class

NormalMouse nrMouse;// object of NormalMouse class

GamePanel() {

random = new Random();

bgMouse = new BigMouse(this);// creating object of BigMouse class and calling constructor BigMouse()

nrMouse = new NormalMouse(this);// creating object of NormalMouse class and calling constructor NormalMouse()

this.setPreferredSize(new Dimension(SCREEN\_WIDTH, SCREEN\_HEIGHT));

this.setBackground(Color.BLACK); // To set Background Color

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

this.addKeyListener(new MyKeyAdapter());// For keyBoardInput

startGame();

}

public void paintComponent(Graphics g) {

super.paintComponent(g);// call parent class (JPanel) Method

draw(g);

}

// Method for draw Snake and mouse.

public void draw(Graphics g) {

if (running) {

if (countMouse == 5) {

bgMouse.paintComponent(g);// draw big mouse

} else {

nrMouse.paintComponent(g);// draw normal mouse

}

for (int i = 0; i < snakeBodyParts; i++) {

g.setColor(Color.green);// color of snake body

g.fillOval(snakeX[i], snakeY[i], UNIT\_SIZE, UNIT\_SIZE); // For Shape of Snake

}

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

g.fillOval(snakeX[0], snakeY[0], UNIT\_SIZE, UNIT\_SIZE);

g.setColor(Color.WHITE);

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

FontMetrics metrics = getFontMetrics(g.getFont()); // provides dimensions of that font

g.drawString("Score: " + score, (SCREEN\_WIDTH - metrics.stringWidth("Score: " + score)) / 2,

SCREEN\_HEIGHT / 10);

} else {

gameOver(g);

}

}

public void startGame() {

newBigMouse();

newSmallMouse();

running = true; // To move snake , Starting of the Game

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

timer.start();

}

public void newBigMouse() {

bigMouseX = random.nextInt((int) (SCREEN\_WIDTH / UNIT\_SIZE)) \* UNIT\_SIZE;// generates new coordinates of big

// mouse

bigMouseY = random.nextInt((int) (SCREEN\_HEIGHT / UNIT\_SIZE)) \* UNIT\_SIZE;

}

public void newSmallMouse() {

mouseX = random.nextInt((int) (SCREEN\_WIDTH / UNIT\_SIZE)) \* UNIT\_SIZE;// generates new coordinates of normal

// mouse

mouseY = random.nextInt((int) (SCREEN\_HEIGHT / UNIT\_SIZE)) \* UNIT\_SIZE;

}

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] - UNIT\_SIZE;

break;

case 'D':

snakeY[0] = snakeY[0] + UNIT\_SIZE;

break;

case 'L':

snakeX[0] = snakeX[0] - UNIT\_SIZE;

break;

case 'R':

snakeX[0] = snakeX[0] + UNIT\_SIZE;

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;

}

}

if (snakeX[0] < 0) {

// running = false;//if touch the border then out

snakeX[0] = SCREEN\_WIDTH;// snake comes from opposite direction

}

if (snakeX[0] > SCREEN\_WIDTH) {

snakeX[0] = 0;

// running = false;

}

if (snakeY[0] < 0) {

snakeY[0] = SCREEN\_WIDTH;

// running = false;

}

if (snakeY[0] > SCREEN\_HEIGHT) {

snakeY[0] = 0;

// running = false;

}

if (!running) {

timer.stop(); // stops sending action to the event listener

}

}

public void gameOver(Graphics g) {

// score text

g.setColor(Color.WHITE);

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

FontMetrics metrics = getFontMetrics(g.getFont());

g.drawString("Score: " + score, (SCREEN\_WIDTH - metrics.stringWidth("Score: " + score)) / 2,

SCREEN\_HEIGHT / 10);

// game over text

g.setColor(Color.WHITE);

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

g.drawString("Game Over", SCREEN\_WIDTH / 5, SCREEN\_HEIGHT / 2);

}

public void actionPerformed(ActionEvent e) {

if (running) {

moveSnake();// makes if there any changes in direction

if (countMouse == 5) {

bgMouse.checkMouse();// check for score if eat big mouse score increase by 4

} else {

nrMouse.checkMouse();// check for score if eat normal mouse score increase by 1

}

checkCollisions();

}

repaint();

}

public class MyKeyAdapter extends KeyAdapter {

// MyKeyAdapter to handle keyBoard Input

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;

}

}

}

}

**NormalMouse.java:**

package Main;

import java.awt.Color;

import java.awt.Graphics; import javax.swing.JPanel;

public class NormalMouse extends JPanel implements DrawMouse {

GamePanel game;

NormalMouse( GamePanel game){ this.game = game;

}

public void paintComponent(Graphics g) {

super.paintComponent(g); draw(g);

}

public void draw(Graphics g) { // draw mouse

g.setColor(Color.PINK);

g.fillOval(game.mouseX, game.mouseY, game.UNIT\_SIZE, game.UNIT\_SIZE);

}

public void checkMouse() {

if ((game.snakeX[0] == game.mouseX) &&

(game.snakeY[0] == game.mouseY)) {

game.snakeBodyParts++;

game.score++;

game.countMouse++;

game.newSmallMouse();

}

}

}

**BigMouse.java:**

package Main;

import java.awt.Color; import java.awt.Graphics; import javax.swing.JPanel;

public class BigMouse extends JPanel implements DrawMouse { GamePanel game;

BigMouse( GamePanel game){

this.game = game;

}

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

draw(g);

}

public void draw(Graphics g) { // draw mouse g.setColor(Color.PINK);

g.fillOval(game.mouseX, game.mouseY, game.UNIT\_SIZE, game.UNIT\_SIZE);

g.setColor(Color.RED);

g.fillOval(game.bigMouseX, game.bigMouseY, game.UNIT\_SIZE+4, game.UNIT\_SIZE+4);

}

public void checkMouse() {

if ((game.snakeX[0] == game.mouseX) &&

(game.snakeY[0]== game.mouseY)) {

if(game.countMouse==5) game.countMouse=0;

game.snakeBodyParts++; game.score++;

game.countMouse++;

game.newSmallMouse();

}

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

game.countMouse=0; game.snakeBodyParts++; game.score+=4;

game.newBigMouse();

}

}

}

**DrawMouse.java:** package Main;

import java.awt.Graphics;

public interface DrawMouse {

abstract public void draw(Graphics g); abstract public void checkMouse();

}

**Output:**

