Title SNAKE GAME

Problem Statement:

Snake can move in a given direction and when it eats the food, the length of snake increases. When snake crosses itself, the game will over.

Objective:

In this game the player controls a snake. The objective is to eat as many apples as possible. Each time the snake eats an apple its body grows. The snake must avoid the walls and its own body.

Software Requirements:

Any kind of operating system Java IDE like: Eclipse.

Source Code:

SnakeGame.java:

```
public class SnakeGame {

	public static void main(String[] args) {

		new GameFrame();
	}
}
```

GameFrame.java:

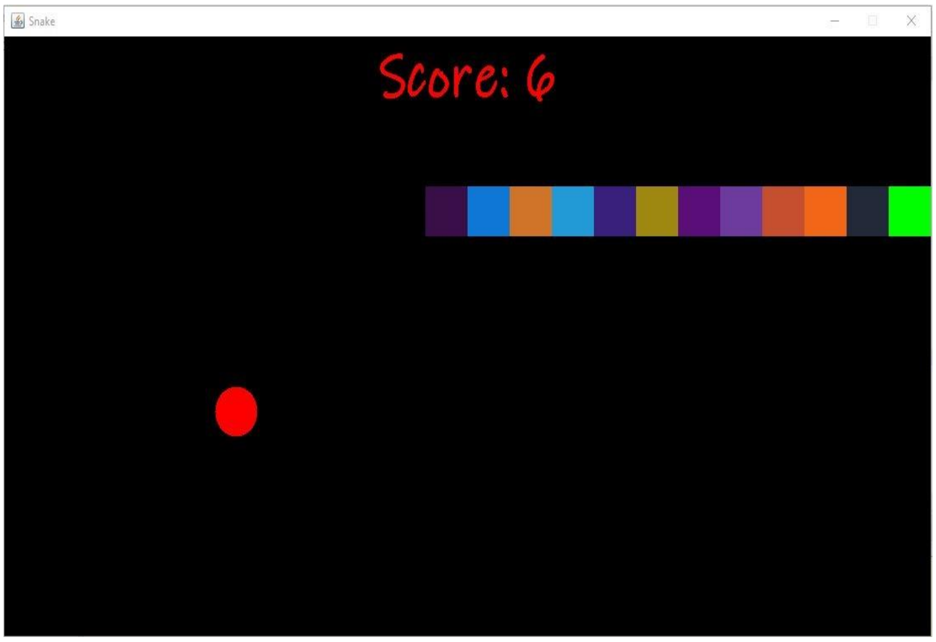
```
import javax.swing.JFrame;

public class GameFrame extends JFrame{

	GameFrame(){

		this.add(new GamePanel());
		this.setTitle("Snake");
		this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		this.setResizable(false);
		this.pack();
		this.setVisible(true);
		this.setLocationRelativeTo(null);

	}
}
```

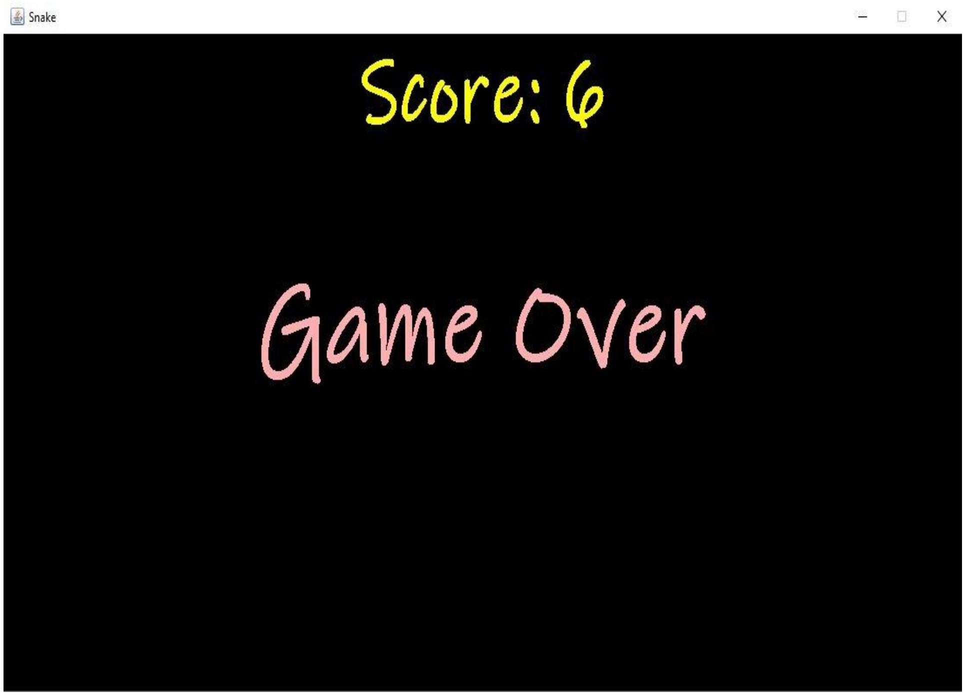
GamePanel.java:

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

public class GamePanel extends JPanel implements ActionListener{

	static final int SCREEN_WIDTH = 1100;
	static final int SCREEN_HEIGHT = 600;
	static final int UNIT_SIZE = 50;
	static final int GAME_UNITS =
(SCREEN_WIDTH*SCREEN_HEIGHT)/(UNIT_SIZE*UNIT_SIZE);
	static final int DELAY = 175;
	final int x[] = new int[GAME_UNITS];
	final int y[] = new int[GAME_UNITS];
	int bodyParts = 6;
	int applesEaten;
	int appleX;
	int appleY;
	char direction = 'R';
	boolean running = false;
	Timer timer;
	Random random;

	GamePanel(){
		random = new Random();
		this.setPreferredSize(new
Dimension(SCREEN_WIDTH,SCREEN_HEIGHT));
		this.setBackground(Color.black);
		this.setFocusable(true);
		this.addKeyListener(new MyKeyAdapter());
		startGame();
	}
	public void startGame() {
		newApple();
		running = true;
		timer = new Timer(DELAY,this);
		timer.start();
	}
	public void paintComponent(Graphics g) {
		super.paintComponent(g);
		draw(g);
	}
	public void draw(Graphics g) {

		if(running) {
			/*
			for(int i=0;i<SCREEN_HEIGHT/UNIT_SIZE;i++) {
				g.drawLine(i*UNIT_SIZE,		0,		i*UNIT_SIZE,
SCREEN_HEIGHT);
				g.drawLine(0,		i*UNIT_SIZE,		SCREEN_WIDTH,
i*UNIT_SIZE);
			}
			*/
			g.setColor(Color.red);
			g.fillOval(appleX, appleY, UNIT_SIZE, UNIT_SIZE);

			for(int i = 0; i< bodyParts;i++) {
				if(i == 0) {
					g.setColor(Color.green);
					g.fillRect(x[i], y[i], UNIT_SIZE, UNIT_SIZE);
				}
				else {
					g.setColor(new Color(45,180,0));
					g.setColor(new
Color(random.nextInt(255),random.nextInt(255),random.nextInt(255)));
					g.fillRect(x[i], y[i], UNIT_SIZE, UNIT_SIZE);
				}
			}
			g.setColor(Color.red);
			g.setFont( new Font("Ink Free",Font.BOLD, 60));
			FontMetrics metrics = getFontMetrics(g.getFont());
			g.drawString("Score:	"+applesEaten,	(SCREEN_WIDTH	-
metrics.stringWidth("Score: "+applesEaten))/2, g.getFont().getSize());
		}
		else {
			gameOver(g);
		}

	}
	public void newApple(){
		appleX								=
random.nextInt((int)(SCREEN_WIDTH/UNIT_SIZE))*UNIT_SIZE;
		appleY								=
random.nextInt((int)(SCREEN_HEIGHT/UNIT_SIZE))*UNIT_SIZE;
	}
	public void move(){
		for(int i = bodyParts;i>0;i--) {
			x[i] = x[i-1];
			y[i] = y[i-1];
		}

		switch(direction) {
		case 'U':
			y[0] = y[0] - UNIT_SIZE;
			break;
		case 'D':
			y[0] = y[0] + UNIT_SIZE;
			break;
		case 'L':
			x[0] = x[0] - UNIT_SIZE;
			break;
		case 'R':
			x[0] = x[0] + UNIT_SIZE;
			break;
		}

	}
	public void checkApple() {
		if((x[0] == appleX) && (y[0] == appleY)) {
			bodyParts++;
			applesEaten++;
			newApple();
		}
	}
	public void checkCollisions() {
		//checks if head collides with body
		for(int i = bodyParts;i>0;i--) {
			if((x[0] == x[i])&& (y[0] == y[i])) {
				running = false;
			}
		}
		//check if head touches left border
		if(x[0] < 0) {
			running = false;
		}
		//check if head touches right border
		if(x[0] > SCREEN_WIDTH) {
			running = false;
		}
		//check if head touches top border
		if(y[0] < 0) {
			running = false;
		}
		//check if head touches bottom border
		if(y[0] > SCREEN_HEIGHT) {
			running = false;
		}

		if(!running) {
			timer.stop();
		}
	}
	public void gameOver(Graphics g) {
		//Score
		g.setColor(Color.yellow);
		g.setFont( new Font("Ink Free",Font.BOLD, 80));
		FontMetrics metrics1 = getFontMetrics(g.getFont());
		g.drawString("Score: "+applesEaten, (SCREEN_WIDTH - 
metrics1.stringWidth("Score: "+applesEaten))/2, g.getFont().getSize());
		//Game Over text
		g.setColor(Color.pink);
		g.setFont( new Font("Ink Free",Font.BOLD, 100));
		FontMetrics metrics2 = getFontMetrics(g.getFont());
		g.drawString("Game Over", (SCREEN_WIDTH - 
metrics2.stringWidth("Game Over"))/2, SCREEN_HEIGHT/2);
	}
	@Override
	public void actionPerformed(ActionEvent e) {

		if(running) {
			move();
			checkApple();
			checkCollisions();
		}
		repaint();
	}

	public class MyKeyAdapter extends KeyAdapter{
		@Override
		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;
                                }
                        }
                }
}
```

Screenshot (Output):





Conclusion:

This case study presents much of the development of a program to play a snake game, similar to that found on certain old mobile phones.