**OBJECT ORIENTED PROGRAMMING**

**SEMESTER PROJECT**

**SNAKE 2D GAME**

**PREPARED BY:**

**ABDUL BASIT**

**193227**

**BSCS-6C**

**CODE:**

**MAIN CLASS:**

import java.awt.Color;

import javax.swing.JFrame;

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

/\*\*

\*

\* @author Abdul Basit

\*/

public class Main {

public static void main(String [] args){

//CREATING JAVA FRAME

JFrame obj=new JFrame();

//CREATING GAMEPLAY OBJECT

GamePlay gameplay=new GamePlay();

//SETTING BOARD

obj.setBounds(10,10,905,700);

obj.setBackground(Color.BLUE);

obj.setResizable(false);

obj.setVisible(true);

obj.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

//ADDING GAMEPLAY OBJECT IN THE JFRAME OBJECT

obj.add(gameplay);

}//MAIN ENDS

}//MAIN CLASS ENDS

**GAMEPLAY CLASS:**

import java.awt.Color;

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.ImageIcon;

import javax.swing.JPanel;

import javax.swing.Timer;

public class GamePlay extends JPanel implements KeyListener,ActionListener{

//VARIABLES FOR STTORING THE LENGTH OF SNAKE OR ITS POINTS

private int[] snakeXlength = new int [750];

private int[] snakeYlength = new int [750];

//DEFAULT MOVEMENTS AND THEIR VALUES

private boolean left=false;

private boolean right=false;

private boolean up=false;

private boolean down=false;

//MOUTH ICON FOR SNAKE

private ImageIcon leftmouth;

private ImageIcon rightmouth;

private ImageIcon upmouth;

private ImageIcon downmouth;

//FOOD POSITIONS IN X AND Y COORDINATES

private int [] foodXpos={25,50,75,100,125,150,175,200,225,250,275,300,325,350,375,400,425,450,475,500,525,550,575,600,625,650,675,700,725,750,775,800,825,850};

private int [] foodYpos={75,100,125,150,175,200,225,250,275,300,325,350,375,400,425,450,475,500,525,550,575,600,625};

//FOOD IMAGE

private ImageIcon foodimage;

//RANDOM NUMBER GENERATOR TO RANDOMIZE THE LOCATION OF FOOD

private Random random = new Random();

//GETTING FOOD X-Y COORDINATES RANDOMLY

private int xpos=random.nextInt(34);

private int ypos=random.nextInt(23);

//SCORES VARIABLE

private int score=0;

//LENGTH OF SNAKE AND ITS DEFAULT VALUE

private int lengthOfSnake=3;

//TIMER TO MOVE THE COORDINATES

private Timer timer;

//GIVING DELAY TO TIMER

private int delay =100;

//SNAKE IMAGE

private ImageIcon snakeimage;

//TITLE OF THE GAME

private ImageIcon titleImage;

//MOVES CALCULATER

private int moves =0;

//GAMEPLAY FUNCTION IN WHICH ALL THE FUNCTIONALITY OF GAME PERFORMS

public GamePlay()

{

//KEY LISTENER WILL DETECT THE KEY

addKeyListener(this);

setFocusable(true);

setFocusTraversalKeysEnabled(false);

//MAKING NEW TIMER OBJECT AND STARTING IT

timer = new Timer (delay,this);

timer.start();

}

//DRAWING THE GAME(THE POSITIONS AND MOVEMENTS)

public void paint(Graphics g)

{

//GIVING DEFAULT POSITION OF THE SNAKE

if(moves==0)

{

snakeXlength[2]=50;

snakeXlength[1]=75;

snakeXlength[0]=100;

snakeYlength[2]=100;

snakeYlength[1]=100;

snakeYlength[0]=100;

}

//DRAW TITLE IMAGE BORDER

g.setColor(Color.WHITE);

g.drawRect(24,10,851,55);

//DRAW THE TITLE IMAGE

titleImage = new ImageIcon("snaketitle.jpg");

titleImage.paintIcon(this, g, 25, 11);

//DRAW BORDER FOR GAMEPLAY

g.setColor(Color.WHITE);

g.drawRect(24, 74, 851, 577);

//DRAW BACKGROUND FOR THE GAMEPLAY

g.setColor(Color.BLACK);

g.fillRect(25, 75, 850, 575);

//DRAW SCORES

g.setColor(Color.WHITE);

g.setFont(new Font ("arial",Font.PLAIN,14));

g.drawString("Scores: " +score,780,30);

//DRAW LENGTH

g.setColor(Color.WHITE);

g.setFont(new Font ("arial",Font.PLAIN,14));

g.drawString("Length: " +lengthOfSnake,780,50);

//IMPORTING THE FIRST IMAGE FOR THE RIGHT DIRECTION WHEN THE GAME STARTS

rightmouth=new ImageIcon("rightmouth.png");

rightmouth.paintIcon(this, g, snakeXlength[0], snakeYlength[0]);

//GIVING THE SNAKE THE APPROPRIATE IMAGE WHEN MOVING IN THE PARTICULAR DIRECTION

for(int a=0;a<lengthOfSnake;a++)

{

if(a==0 && right)

{

rightmouth=new ImageIcon("rightmouth.png");

rightmouth.paintIcon(this, g, snakeXlength[a], snakeYlength[a]);

}

if(a==0 && left)

{

leftmouth=new ImageIcon("leftmouth.png");

leftmouth.paintIcon(this, g, snakeXlength[a], snakeYlength[a]);

}

if(a==0 && down)

{

downmouth=new ImageIcon("downmouth.png");

downmouth.paintIcon(this, g, snakeXlength[a], snakeYlength[a]);

}

if(a==0 && up)

{

upmouth=new ImageIcon("upmouth.png");

upmouth.paintIcon(this, g, snakeXlength[a], snakeYlength[a]);

}

if(a!=0)

{

snakeimage=new ImageIcon("snakeimage.png");

snakeimage.paintIcon(this, g, snakeXlength[a], snakeYlength[a]);

}

}

//DRAWING FOOD ON THE BOARD AND GIVING THE RANDOM LOCATION

foodimage=new ImageIcon("enemy.png");

if((foodXpos[xpos]==snakeXlength[0] && foodYpos[ypos]==snakeYlength[0]))

{

score++;

lengthOfSnake++;

xpos=random.nextInt(34);

ypos=random.nextInt(23);

}

//PAINTING THE IMAGE

foodimage.paintIcon(this,g,foodXpos[xpos],foodYpos[ypos]);

//DRAWING THE PANEL WHEN THE SNAKE COLLIDES WITH ITSELF

for(int b=1;b<lengthOfSnake;b++)

{

if(snakeXlength[b]==snakeXlength[0] && snakeYlength[b]==snakeYlength[0])

{

right=false;

left=false;

up=false;

down=false;

g.setColor(Color.WHITE);

g.setFont(new Font("arial",Font.BOLD,50));

g.drawString("Game Over!!", 300, 300);

g.setFont(new Font("arial",Font.PLAIN,16));

g.drawString("PRESS SPACE FOR RESTART\n "

+ "&", 320, 350);

g.setFont(new Font("arial",Font.PLAIN,16));

g.drawString("ENTER TO CLOSE THE GAME", 320, 390);

}

}

//FUNCTION THAT ENSURES THAT NO MORE DRAWING

g.dispose();

}

//ABSTRACT METHOD IN ACTION LISTENER

@Override

public void actionPerformed(ActionEvent a) {

//STARTING TIMER AGAIN

timer.start();

//SETTING IF SNAKE GOES FROM RIGHT SIDE THEN CAME BACK FROM LEFT

if(right)

{

for(int r=lengthOfSnake-1;r>=0;r--)

{

snakeYlength[r+1]=snakeYlength[r];

}

for(int r=lengthOfSnake;r>=0;r--)

{

if(r==0)

{

snakeXlength[r]=snakeXlength[r]+25;

}

else

{

snakeXlength[r]=snakeXlength[r-1];

}

if(snakeXlength[r]>850)

{

snakeXlength[r]=25;

}

}

//CALLING REPAINT FOR DRAWING THE SNAKE AGAIN WITH NEW COORDINATES

repaint();

}

//SETTING IF SNAKE GOES FROM LEFT SIDE THEN CAME BACK FROM RIGHT

if(left)

{

for(int r=lengthOfSnake-1;r>=0;r--)

{

snakeYlength[r+1]=snakeYlength[r];

}

for(int r=lengthOfSnake;r>=0;r--)

{

if(r==0)

{

snakeXlength[r]=snakeXlength[r]-25;

}

else

{

snakeXlength[r]=snakeXlength[r-1];

}

if(snakeXlength[r]<25)

{

snakeXlength[r]=850;

}

}

//CALLING REPAINT FOR DRAWING THE SNAKE AGAIN WITH NEW COORDINATES

repaint();

}

//SETTING IF SNAKE GOES FROM UP SIDE THEN CAME BACK FROM DOWN

if(up){for(int r=lengthOfSnake-1;r>=0;r--)

{

snakeXlength[r+1]=snakeXlength[r];

}

for(int r=lengthOfSnake;r>=0;r--)

{

if(r==0)

{

snakeYlength[r]=snakeYlength[r]-25;

}

else

{

snakeYlength[r]=snakeYlength[r-1];

}

if(snakeYlength[r]<75)

{

snakeYlength[r]=625;

}

}

//CALLING REPAINT FOR DRAWING THE SNAKE AGAIN WITH NEW COORDINATES

repaint();

}

//SETTING IF SNAKE GOES FROM DOWN SIDE THEN CAME BACK FROM UP

if(down)

{

for(int r=lengthOfSnake-1;r>=0;r--)

{

snakeXlength[r+1]=snakeXlength[r];

}

for(int r=lengthOfSnake;r>=0;r--)

{

if(r==0)

{

snakeYlength[r]=snakeYlength[r]+25;

}

else

{

snakeYlength[r]=snakeYlength[r-1];

}

if(snakeYlength[r]>625)

{

snakeYlength[r]=75;

}

}

//CALLING REPAINT FOR DRAWING THE SNAKE AGAIN WITH NEW COORDINATES

repaint();

}

}

//WHEN DIFFERENT KEYS ARE PRESSED THEN WHICH FUNCTIONALITY SHOULD HAPPEN IS IN THIS FUNCTION

@Override

public void keyPressed(KeyEvent a) {

//WHEN WANT TO CLOSE THE GAME WE HAVE TO PRESSED ENTER KEY

if(a.getKeyCode()==KeyEvent.VK\_ENTER)

{

System.exit(1);

}

//WHEN WE WANT TO RESTART THE GAME THEN THE SPACE KEY IS PRESSED

for(int b=1;b<lengthOfSnake;b++)

{

if(snakeXlength[b]==snakeXlength[0] && snakeYlength[b]==snakeYlength[0] && a.getKeyCode()==KeyEvent.VK\_SPACE)

{

moves=0;

score=0;

lengthOfSnake=3;

repaint();

}

}

//WHEN WANT TO MOVE THE SNAKE TOWARDS RIGHT DIRECTION

if(a.getKeyCode() == KeyEvent.VK\_RIGHT)

{

moves++;

right=true;

if(!left)

{

right=true;

}

else

{

right=false;

left=true;

}

up=false;

down=false;

}

//WHEN WANT TO MOVE THE SNAKE TO THE LEFT DIRECTION

if(a.getKeyCode() == KeyEvent.VK\_LEFT)

{

moves++;

left=true;

if(!right)

{

left=true;

}

else

{

left=false;

right=true;

}

up=false;

down=false;

}

//WHEN WANTS TO MOVE THE SNAKE TOWARDS THE UP DIRECTION

if(a.getKeyCode() == KeyEvent.VK\_UP)

{

moves++;

up=true;

if(!down)

{

up=true;

}

else

{

up=false;

down=true;

}

left=false;

right=false;

}

//WHEN WANTS TO MOVE THE SNAKE TO THE DOWN DIRECTION

if(a.getKeyCode() == KeyEvent.VK\_DOWN)

{

moves++;

down=true;

if(!up)

{

down=true;

}

else

{

down=false;

up=true;

}

left=false;

right=false;

}

}

//THE ABSTRACT METHOD THAT HAVE TO BE IMPLEMENTED

@Override

public void keyReleased(KeyEvent a) {

}

//THE ABSTRACT METHOD THAT HAVE TO BE IMPLEMENTED

@Override

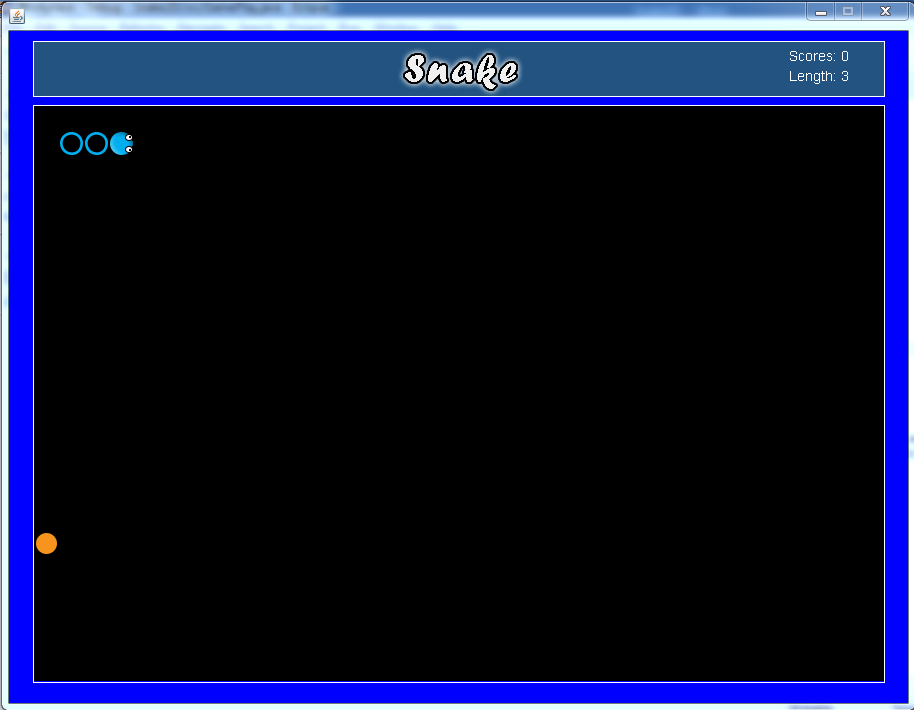
public void keyTyped(KeyEvent a) {

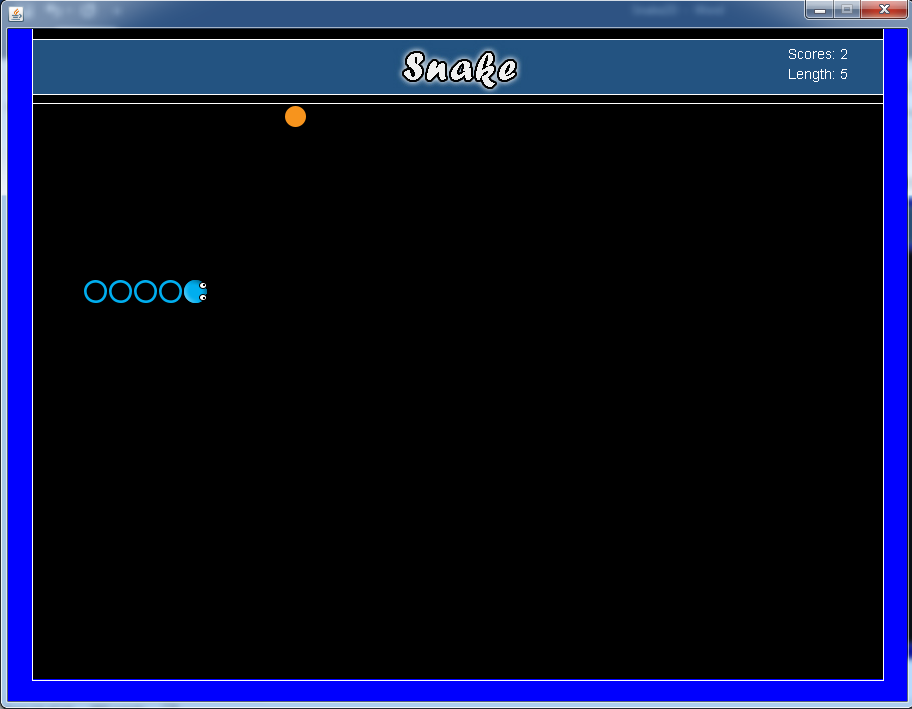
}

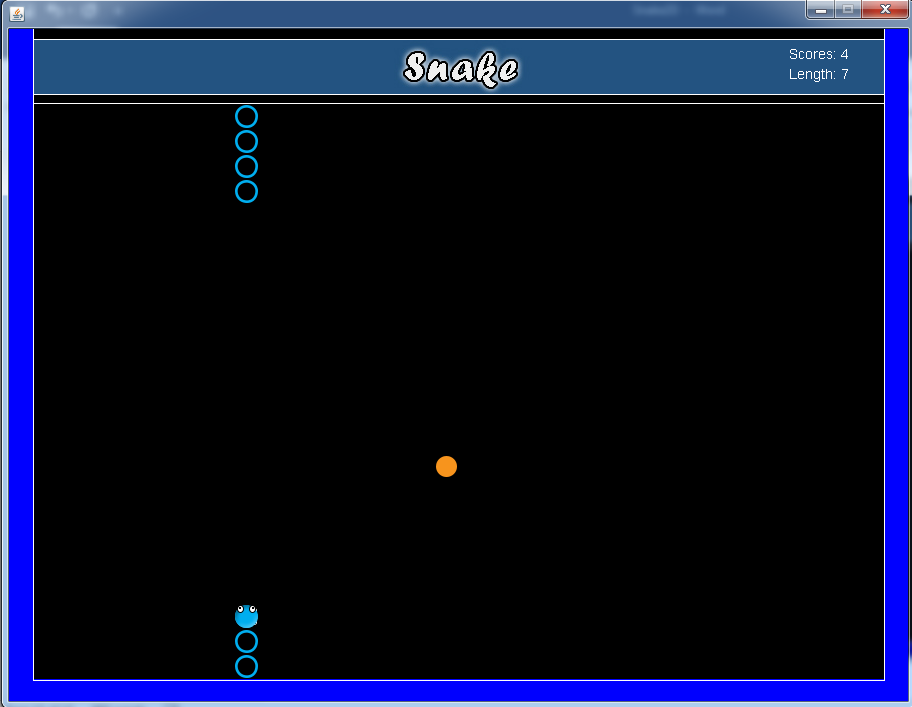
}//ENDING THE GAMEPLAY CLASS

**OUTPUT:**

**When the game just started**



**When we start moving snake on board**

**When snake enter from one side and out from other**

**When we collided with our own body**



**When we restart the game after collision**

