Task Report

Name: Makula Pravallika

Email-id: makulapravallika02@gmail.com

Task Title: Snake Game in Java

Task Description:

The Snake Game is a very popular and fun game.Every time the snake eats the food,its length grows longer that makes the game more difficult.Snake can move in any direction for to get food but if snake crosses itself or come out of the frame then the game will be Over. It display score as how many times snake get food.

Steps Taken:

1. Create one java project.
2. Inside this Project I create two classes that are GameBoard.java and Snake.java
3. In GameBoard class I setup the game board
4. In Snake class I create some methods -move(),food(),collision(),draw(),paintComponent() and also I use two interfaces-ActionListener,KeyListener.
5. The draw() method performs following actions that are create snake,food in game board and displaying score and after eating food snake length will grow.
6. placefood()-It perform generate food in random locations in the game board.
7. collision()-It checks if the snake collides itself or not.
8. move()- It checks if snake will touch the game board body it display Game over if not then the snake eat food then place new food for it and after eat the food the snake length will grow otherwise not.
9. actionPerformed()-when we press the key in the keyboard it will rum move(),repaint() methods or it checks gameover condition also.
10. keyPressed()-It handle user inputs.

Challenges Faced:

1. I faced lot of errors in to increase snake length after eating food.
2. I faced errors in handle collisions.

Solutions Implemented:

1. GameBoard.java

```
import javax.swing.*;
public class GameBoard {
    public static void main(String[] args) {
        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);
        Snake ob=new Snake(boardwidth,boardheight);
        frame.add(ob);
        frame.pack();
        ob.requestFocus();
    }
}
```

2. Snake.java

```
3. import java.awt.*;
   import java.awt.event.*;
   import java.util.ArrayList;
   import java.util.Random;
   import javax.swing.*;
   public class Snake 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
       Timer gameloop;
       int velocityx,velocityy;
       boolean gameover=false;
       Snake(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<>();
           food=new Tile(10,10);
           random=new Random();
           placeFood();

           velocityx=0;
```

```
            velocityy=0;

            gameloop=new Timer(100,this);
            gameloop.start();

        }
        public void paintComponent(Graphics g){
            super.paintComponent(g);
            draw(g);
        }
        public void draw(Graphics g){
            //Grid
           /* 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,tiles
    ize,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,tiles
    ize,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 boolean collision(Tile tile1,Tile tile2){
            return tile1.x==tile2.x && tile1.y==tile2.y;
        }
        public void move(){
            //eat food
            if(collision(snakehead,food)){
                snakebody.add(new Tile(food.x,food.y));
                placeFood();
            }
            //snake body
```

```
            for(int i=snakebody.size()-1;i>=0;i--){
                Tile snakepart=snakebody.get(i);
                if(i==0){
                    snakepart.x=snakehead.x;
                    snakepart.y=snakehead.y;
                }else{
                    Tile prevsnakepart=snakebody.get(i-1);
                    snakepart.x=prevsnakepart.x;
                    snakepart.y=prevsnakepart.y;
                }
            }
            //snake head
            snakehead.x+=velocityx;
            snakehead.y+=velocityy;
            //game over condition
            for(int i=0;i<snakebody.size();i++){
                Tile snakepart=snakebody.get(i);
                //collide with the snake head
                if(collision(snakehead,snakepart)){
                    gameover=true;
                }
            }
            if(snakehead.x*tilesize<0 || snakehead.x*tilesize>boardwidth
    || snakehead.y*tilesize<0 || snakehead.y*tilesize>boardwidth){
                gameover=true;
            }
        }
        @Override
        public void actionPerformed(ActionEvent e) {
            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_RIGHT && velocityx!=-
    1) {
                velocityx=1;
                velocityy=0;
            } else if (e.getKeyCode()==KeyEvent.VK_LEFT && velocityx!=1)
    {
                velocityx=-1;
                velocityy=0;
            }
        }

        @Override
        public void keyReleased(KeyEvent e) {

        }
        @Override
```

```
        public void keyTyped(KeyEvent e) {

        }


    }
```

Learnings: For this Project I learn java swing concept from youtube,google.

Project Update : Completed.