Java GUI Snake game

Asked 2 years, 3 months ago Modified 2 years, 3 months ago Viewed 955 times

5

I just got started coding in Java and I just made my first project. I was wondering if someone would review my code and show me the things I can improve in it. I am eager to learn and would appreciate any advice.

Here is the code:

```
package snake.app;

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

import javax.swing.JPanel;


public class Main {

    public static void main(String[] args) {
        new GameFrame();
    }
}

import javax.swing.JFrame;

public class GameFrame extends JFrame{
    public GameFrame() {
        this.add(new GamePanel());
        this.setTitle("Snake");
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setResizable(true);
        this.pack();
```



```
    static final int screenHeight = 600;
```

```
static final int unitSize = 25;
static final int gameUnits = (screenWidth * screenHeight) / unitSize;
static final int delay = 75;

private boolean running = true;

private Snake snake;
private Apple apple;

private Timer timer;
private Random random;


public GamePanel() {
    timer = new Timer(delay, this);
    timer.start();
    random = new Random();

    this.setPreferredSize(new Dimension(screenWidth, screenHeight));
    this.setBackground(Color.BLACK);
    this.setFocusable(true);
    this.addKeyListener(new Adapter());

    startGame();
}

public void startGame() {
    running = true;
    snake = new Snake();
    apple = new Apple();
}

@Override
public void actionPerformed(ActionEvent e) {
    if (running) {
        move();
        checkCollisions();
    }
    repaint();
}

public void move() {
    for (int i = snake.length; i > 0; i--) {
        snake.XPositions[i] = snake.XPositions[i - 1];
        snake.YPositions[i] = snake.YPositions[i - 1];
    }
```



```
}
```

```
    public void checkCollisions() {
        checkBodyCollision();
        checkWallCollision();
        checkAppleCollision();
    }

    private void checkBodyCollision() {
        for (int i = 0; i < snake.length; i++) {
            for (int j = 0; i < snake.length; i++) {
                if (snake.XPositions[i] == snake.XPositions[j] &&
snake.YPositions[i] == snake.YPositions[j]) {
                    if (i != j) {
                        running = false;
                    }
                }
            }
        }
    }
    private void checkWallCollision() {
        if (snake.XPositions[0] > screenWidth) {
            running = false;
        } else if (snake.XPositions[0] < 0) {
            running = false;
        } else if (snake.YPositions[0] > screenHeight) {
            running = false;
        } else if (snake.YPositions[0] < 0) {
            running = false;
        }
    }
    public void checkAppleCollision() {
        for (int i = 0; i < snake.length; i++) {
            if (snake.XPositions[i] == apple.XPosition && snake.YPositions[i] ==
apple.YPosition) {
                apple = new Apple();
                snake.length++;
            }
        }
    }

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

    public void draw(Graphics g) {
        drawGrid(g);
```



```
    }
```

```
    }
    private void drawApple(Graphics g) {
        g.setColor(Color.RED);
        g.fillRect(apple.XPosition, apple.YPosition, unitSize, unitSize);
    }

    public class Apple {
        public int XPosition;
        public int YPosition;

        public Apple() {
            XPosition = random.nextInt((int)(screenWidth / unitSize)) * unitSize;
            YPosition = random.nextInt((int)(screenHeight / unitSize)) *
unitSize;
        }
    }

    public class Snake {
        private final int startXPosition = (int) (gameUnits / 2);
        private final int startYPosition = (int) (gameUnits / 2);

        private final int[] XPositions = new int[gameUnits];
        private final int[] YPositions = new int[gameUnits];

        private int length = 10;
        private Direction direction;

        public Snake() {
            createRandomDirection();
            createBodyPartPositions();
        }

        private void createRandomDirection() {
            int i = 1;
            switch (i) {
                case 1:
                    direction = Direction.Right;
                    break;
                case 2:
                    direction = Direction.Left;
                    break;
                case 3:
                    direction = Direction.Up;
                    break;
                case 4:
                    direction = Direction.Down;
                    break;
```



```
YPositions[i] = ((startYPosition + i) * unitSize);
```

```
                }
            } else if (direction == Direction.Down) {
                for (int i = 0; i < this.length; i++) {
                    YPositions[i] = ((startYPosition - i) * unitSize);
                }
            }
        }
    }
    public class Adapter extends KeyAdapter {
        @Override
        public void keyPressed(KeyEvent e) {
            switch(e.getKeyCode()) {
                case KeyEvent.VK_RIGHT:
                    if (snake.direction != GamePanel.Direction.Left) {
                        snake.direction = GamePanel.Direction.Right;
                    }
                    break;

                case KeyEvent.VK_LEFT:
                    if (snake.direction != GamePanel.Direction.Right) {
                        snake.direction = GamePanel.Direction.Left;
                    }
                    break;

                case KeyEvent.VK_UP:
                    if (snake.direction != GamePanel.Direction.Down) {
                        snake.direction = GamePanel.Direction.Up;
                    }
                    break;

                case KeyEvent.VK_DOWN:
                    if (snake.direction != GamePanel.Direction.Up) {
                        snake.direction = GamePanel.Direction.Down;
                    }
                    break;
            }
        }
    }
}
```

java beginner snake-game

Share Improve this question Follow edited Mar 17, 2021 at 21:36 asked Mar 16, 2021 at 17:43

I'm aware that the time signal is not related to keypress timings (e.g. you might make a direction change but not actually move for up to 75ms). In principle I guess you can tap the arrow keys and not move the snake if you do it quick enough, is that right? Also the amount you move for e.g. a 200ms keypress is actually a bit random depending on how it interacts with the timer.

If your snake speed needed to be lower than 75ms the delays and discontinuities this might be noticeable in the gameplay.

I'd be edgy about a class called Adapter when even the subclass has more information in its name. Maybe DirectionAdapter?

Your use of the `running` value as a substitute for return values in various functions was a surprise compared to calling them and responding to the result. won't give you a lot of options for control flow.

Apples can appear under the snake and score instant points which seems a shame.

You have clearly structured and thought about your approach and it's a great start!

Share Improve this answer Follow

answered Mar 16, 2021 at 20:37

cefn
141 2

1

Beside all hints that you might gather from PMD, Findbugs and Checkstyle?

I would suggest to review the design. The inheritance of GamePanel is

> GamePanel extends JPanel implements ActionListener

whearat inheritance is meant to be used for something that "is" something, and fields are used to use for something that "has" something.

From design I would expect that the GamePanel *have* a ActionListener but I would not expect the

```
private void checkWallCollision() {
    if (snake.XPositions[0] > screenWidth) {
        running = false;
```

Since your class is non-final you allow inheritance and might be able to override basic behaviour. The you better use `setRunning(false);` to be able to override this basic state-information.

```
random = new Random();
```

Well you might get trouble by a Gambling Authority. You might better use:

```
random = new SecureRandom();
```

`GamePanel.move();` have code that should be part of the Snake-Object. Because currently the game moves the snake, this is unusual from the point-of-view of the mother nature. Mother nature would mention that snakes are able to (and usually) move themselves, sure, limited by the environmental situation.

Share Improve this answer Follow

answered Mar 20, 2021 at 19:52

Grim
510 3 13

1

Just one additional aspect not yet mentioned by the other answers so far.

Don't use wildcard imports as in

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

Share Improve this answer Follow

answered Mar 20, 2021 at 21:31

Ralf Kleberhoff

3,104 7 12