TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI

KHOA ĐÀO TẠO QUỐC TẾ

BÁO CÁO ĐỀ TÀI JAVA

Môn học: Java

Đề tài: Game Snakes

Nhóm thực hiện: Nhóm 24

Thành viên: Nguyễn Huy Anh Duy - 882126008

Lớp: Công Nghệ Thông Tin Việt Anh 1

Giảng viên hướng dẫn: Vũ Huấn

Hà Nội, năm 2023

TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI
KHOA ĐÀO TẠO QUỐC TẾ

BÁO CÁO BÀI TẬP LỚN

ĐỀ TÀI JAVA

Game Snakes

Nhóm sinh viên thực hiện: Nhóm 24

Thành viên nhóm: Nguyễn Huy Anh Duy - 882126008

Lớp: Công Nghệ Thông Tin Việt Anh 1

Giảng viên hướng dẫn: Vũ Huấn

Hà Nội, tháng 4 năm 2023

MỤC LỤC

LỜI CẢM ƠN

Trước hết chúng em bày tỏ lòng biết ơn sâu sắc tới thầy giáo hướng dẫn Huấn – Giảng viên bộ môn Lập trình Java trong khoa Quốc Tế và ngành Công Nghệ Thông Tin đã trang bị cho chúng em những kiến thức cơ bản để có thể hoàn thành được bài tập lớn này.

Tuy nhiên trong quá trình nghiên cứu và làm bài tập lớn, do kiến thức chuyên ngành còn hạn chế nên nhóm chúng em còn thiếu sót khi tìm hiểu, đánh giá và trình bày về đề tài. Rất mong nhận được sự quan tâm góp ý của các thầy cô giảng viên bộ môn đề tài của nhóm chúng em được đầy đủ và hoàn chỉnh hơn.

Em xin trân trọng cảm ơn!

Hà Nội, tháng 04 năm 2023

Sinh Viên

Duy

Nguyễn Huy Anh Duy

1. Giới thiệu về Game

1.1 Lịch Sử

Trò chơi gốc (1970): Trò chơi Snakes ban đầu được phát triển bởi nhà phát triển game người Mỹ Gremlin Industries vào những năm 1970. Phiên bản đầu tiên của trò chơi này có tên là "Blockade" và nó cho phép hai người chơi điều khiển hai con rắn trên một màn hình.

Snakes trên các nền tảng khác: Trò chơi Snakes đã xuất hiện trên nhiều nền tảng khác nhau sau đó, bao gồm các phiên bản trên PC, máy chơi trò chơi cầm tay, và các nền tảng trò chơi điện tử khác.

- Theo thời gian thì nhiều người đã chuyển sang internet, game snakes thành game online và dễ dàng truy cập hơn. Do đó trò chơi vẫn là 1 trong số những trò chơi phổ biến nhất, và có thể dễ dàng tìm thấy trên google

1.2 Cách chơi

- Người chơi di chuyển con rắn sao cho con rắn trên màn hình di chuyển và ăn những quả táo màu đỏ.
- Khi con rắn ăn những quả táo màu đỏ sẽ được tính là 1 điểm nếu con rắn ăn táo.
- Khi người chơi mà di chuyển con rắn chạm vào viền của cửa sổ game thì sẽ kết thúc trò chơi và hiện số điểm mà người chơi đạt được

1.3 Các mẹo chơi

- Người chơi dung các nút mũi tên trên bàn phím để di chuyển con rắn đến chỗ quả táo
- Con rắn không thể đi xuyên qua chính cơ thể của rắn.

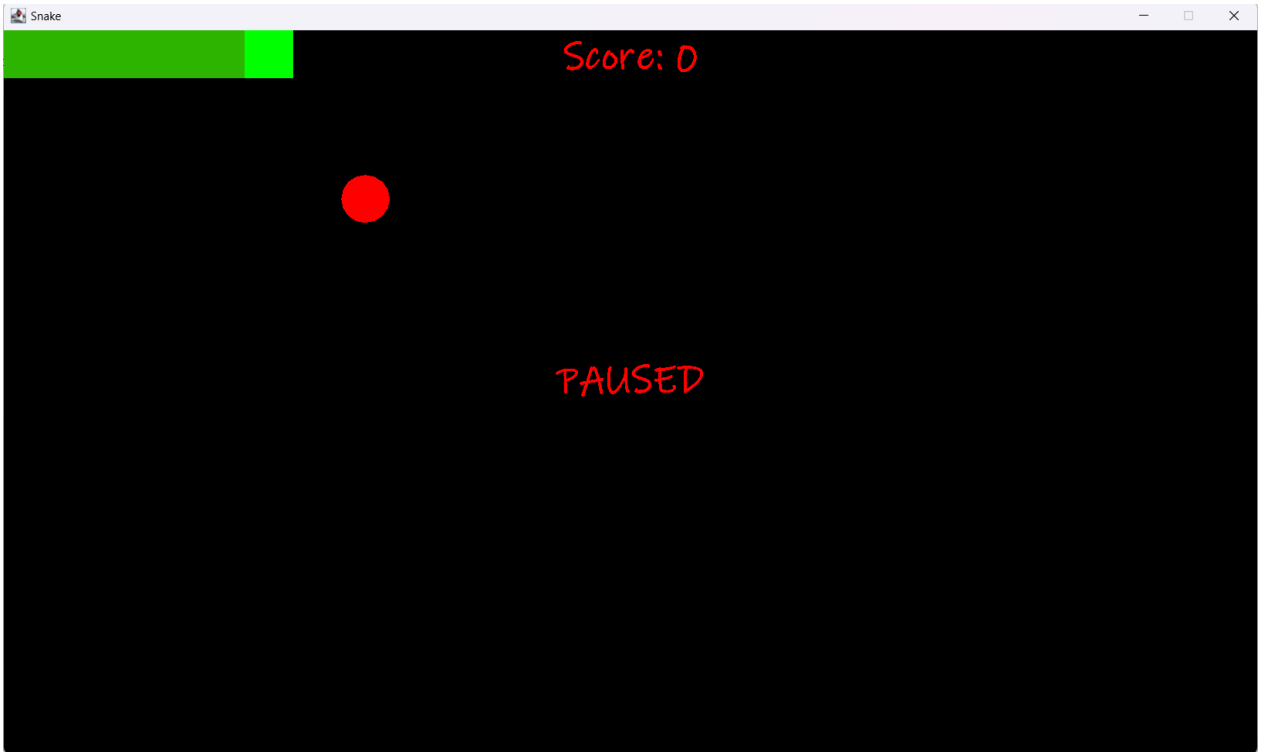
2. Thiết kế

2.1 Nội dung trò chơi:

- Người chơi di chuyển con rắn trên cửa sổ trò chơi để cho rắn ăn những quả táo màu đỏ

- Màn chơi kết thúc khi con rắn chạm vào viền của cửa sổ trò chơi hoặc là rắn chạm vào chính cơ thể của nó.

3. Giao Diện



4. Mô tả các Class

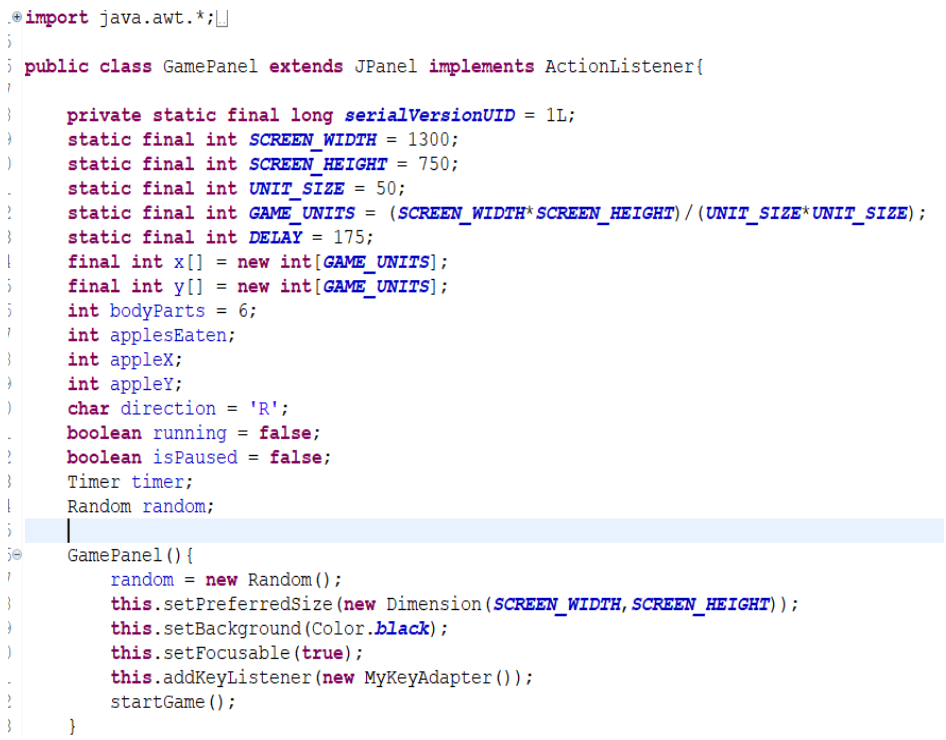
4.1 Lớp GameFrame:

- Mô tả: Lớp "GameFrame" được mô tả trong đoạn mã trên là một lớp con của lớp JFrame trong thư viện Swing của Java. Nó kế thừa các tính năng và phương thức của JFrame để tạo ra một cửa sổ trò chơi cho trò chơi "Snake".

```
import javax.swing.JFrame;

public class GameFrame extends JFrame{

    private static final long serialVersionUID = 1L;

    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);

    }
}
```

4.2 Lớp GamePanel:

- Mô tả: Nơi hiển thị hình ảnh chính của trò chơi
- Nội dung: Lớp "GamePanel" trong mã trên là một lớp con của lớp JPanel trong thư viện Swing của Java. Nó thực hiện giao diện ActionListener để xử lý sự kiện và cung cấp phương thức để vẽ trò chơi "Snake" trên bề mặt JPanel.



```java
import java.awt.*;

public class GamePanel extends JPanel implements ActionListener{

    private static final long serialVersionUID = 1L;
    static final int SCREEN_WIDTH = 1300;
    static final int SCREEN_HEIGHT = 750;
    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;
    boolean isPaused = 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();
    }
```

Hàm khởi tạo constructer-GamePanel

- Hàm "startGame" là một phương thức công khai (public method) mà không trả về giá trị (void). Hàm này được sử dụng để bắt đầu trò chơi.
  - Gọi phương thức "newApple()": Đây là một phương thức không được hiển thị trong đoạn mã bạn đã cung cấp. Tuy nhiên, từ tên phương thức, có thể suy ra rằng nó tạo ra một quả táo mới trong trò chơi. Có thể rằng nó đặt một vị trí ngẫu nhiên cho quả táo hoặc thực hiện các thao tác liên quan khác.

  - Gán giá trị true cho biến "running": Đây là một biến kiểu boolean (Boolean) mà có thể được sử dụng để kiểm tra xem trò chơi đang chạy hay không. Gán giá trị true cho biến này đánh dấu rằng trò chơi đang trong quá trình chạy.

  - Tạo một đối tượng Timer: Hàm này tạo ra một đối tượng Timer để xử lý việc gọi lại (callback). Đối tượng Timer được khởi tạo với hai đối số: DELAY và this. DELAY là một hằng số đại diện cho thời gian trễ (delay) giữa các lần gọi lại. Đối số thứ hai (this) là tham chiếu đến đối tượng hiện tại (đối tượng chứa phương thức "startGame"). Điều này có nghĩa là

phương thức "startGame" sẽ được gọi lại thông qua đối tượng Timer sau mỗi khoảng thời gian DELAY.

- Bắt đầu Timer: Gọi phương thức start() trên đối tượng Timer để bắt đầu quá trình gọi lại (callback) theo khoảng thời gian đã được định nghĩa bởi DELAY.

```
public void startGame() {
    newApple();
    running = true;
    timer = new Timer(DELAY,this);
    timer.start();
}
```

- Hàm "paintComponent" là một phương thức công khai (public method) không trả về giá trị (void). Phương thức này được gọi để vẽ các thành phần của giao diện người dùng trên một đối tượng đồ họa (Graphics) được truyền vào.
  - Gọi phương thức "super.paintComponent(g)": Đây là một lời gọi đến phương thức paintComponent() của lớp cha (thường là lớp JPanel) để đảm bảo rằng các thành phần của lớp cha cũng được vẽ chính xác. Điều này đặc biệt quan trọng khi vẽ các thành phần giao diện người dùng lên một JPanel hoặc một lớp con của JPanel.

  - Gọi phương thức "draw(g)": Đây là một phương thức không được hiển thị trong đoạn mã bạn đã cung cấp. Tuy nhiên, từ tên phương thức, có thể suy ra rằng nó được sử dụng để vẽ các thành phần của giao diện người dùng lên đối tượng đồ họa (Graphics) được truyền vào.

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

- Hàm "draw" là một phương thức không được công khai (public method) không trả về giá trị (void). Phương thức này được sử

dụng để vẽ các thành phần của trò chơi lên đối tượng đồ họa (Graphics) được truyền vào.

- Kiểm tra biến "running": Nếu biến "running" có giá trị true, tức là trò chơi đang chạy, thì tiếp tục vẽ các thành phần của trò chơi. Nếu không, chuyển sang bước kế tiếp để vẽ màn hình kết thúc trò chơi.

- Vẽ quả táo (apple): Đặt màu đỏ (Color.red) cho đối tượng đồ họa (Graphics) và sử dụng phương thức fillOval() để vẽ một hình tròn (quả táo) tại vị trí (appleX, appleY) với kích thước là UNIT_SIZE.

- Vẽ phần thân con rắn (bodyParts): Sử dụng một vòng lặp for để vẽ các phần thân của con rắn. Nếu chỉ số i của vòng lặp là 0, tức là đầu của con rắn, thì đặt màu xanh lá cây (Color.green) cho đối tượng đồ họa (Graphics) và sử dụng phương thức fillRect() để vẽ một hình chữ nhật tại vị trí (x[i], y[i]) với kích thước là UNIT_SIZE. Nếu chỉ số i khác 0, đặt màu xanh lá cây tùy chỉnh (new Color(45,180,0)) và vẽ một hình chữ nhật tương tự.

- Vẽ điểm số (applesEaten): Đặt màu đỏ (Color.red) cho đối tượng đồ họa (Graphics) và đặt font chữ (Font) có kiểu in đậm (Font.BOLD) và kích thước là 40. Sử dụng phương thức getFontMetrics() để tính toán các thông số về font chữ và sử dụng phương thức drawString() để vẽ chuỗi "Score: " + applesEaten tại vị trí trung tâm của màn hình.

- Nếu biến "running" có giá trị false, tức là trò chơi đã kết thúc, gọi phương thức "gameOver(g)" để vẽ màn hình kết thúc trò chơi.

- Kiểm tra biến "isPaused": Nếu biến "isPaused" có giá trị true, tức là trò chơi đang tạm dừng, đặt màu đỏ (Color.red) cho đối tượng đồ họa (Graphics) và đặt font chữ (Font) có kiểu in đậm (Font.BOLD) và kích thước là 40. Sử dụng phương thức

getFontMetrics() để tính toán các thông số về font chữ và sử dụng phương thức drawString() để vẽ chuỗi "PAUSED" tại vị trí trung tâm của màn hình.

```
public void draw(Graphics g) {

    if(running) {
        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.fillRect(x[i], y[i], UNIT_SIZE, UNIT_SIZE);
            }
        }
        g.setColor(Color.red);
        g.setFont( new Font("Ink Free",Font.BOLD, 40));
        FontMetrics metrics = getFontMetrics(g.getFont());
        g.drawString("Score: "+applesEaten, (SCREEN_WIDTH - metrics.stringWidth("Score: "+applesEaten))/2, g.getFont().getSize());
    }
    else {
        gameOver(g);
    }
    if (isPaused) {
        g.setColor(Color.red);
        g.setFont( new Font("Ink Free",Font.BOLD, 40));
        FontMetrics metrics = getFontMetrics(g.getFont());
        g.drawString("PAUSED", (SCREEN_WIDTH - metrics.stringWidth("PAUSED"))/2, SCREEN_HEIGHT/2);
    }

}
```

- Hàm "newApple" là một phương thức không được công khai (public method) không trả về giá trị (void). Phương thức này được sử dụng để tạo một quả táo mới trong trò chơi.
- Tạo vị trí ngẫu nhiên cho quả táo: Sử dụng đối tượng "random" (có thể là một đối tượng thuộc lớp Random) để tạo hai số ngẫu nhiên. Số ngẫu nhiên đầu tiên (appleX) được tính toán bằng cách lấy một số ngẫu nhiên trong khoảng từ 0 đến (SCREEN_WIDTH/UNIT_SIZE - 1), sau đó nhân với UNIT_SIZE để có được vị trí x của quả táo. Tương tự, số ngẫu nhiên thứ hai (appleY) được tính toán bằng cách lấy một số ngẫu

nhiên trong khoảng từ 0 đến (SCREEN_HEIGHT/UNIT_SIZE - 1), sau đó nhân với UNIT_SIZE để có được vị trí y của quả táo.

```
public void newApple(){
    appleX = random.nextInt((int)(SCREEN_WIDTH/UNIT_SIZE))*UNIT_SIZE;
    appleY = random.nextInt((int)(SCREEN_HEIGHT/UNIT_SIZE))*UNIT_SIZE;
}
```

- Hàm "move" là một phương thức không được công khai (public method) không trả về giá trị (void). Phương thức này được sử dụng để di chuyển con rắn trong trò chơi.
  - Di chuyển các phần thân của con rắn: Sử dụng một vòng lặp for để duyệt qua các phần thân của con rắn, bắt đầu từ phần thân cuối cùng và di chuyển đến phần thân đầu tiên. Điều này được thực hiện bằng cách gán vị trí của phần thân (x[i], y[i]) bằng vị trí của phần thân trước đó (x[i-1], y[i-1]). Tức là, phần thân thứ i sẽ di chuyển đến vị trí của phần thân thứ i-1.

  - Di chuyển đầu con rắn dựa trên hướng (direction): Sử dụng một câu lệnh switch-case để kiểm tra giá trị của biến "direction" và thực hiện di chuyển tương ứng. Nếu "direction" là 'U', tức là di chuyển lên, thì giảm giá trị y của phần đầu con rắn đi UNIT_SIZE. Nếu "direction" là 'D', tức là di chuyển xuống, thì tăng giá trị y của phần đầu con rắn lên UNIT_SIZE. Nếu "direction" là 'L', tức là di chuyển sang trái, thì giảm giá trị x của phần đầu con rắn đi UNIT_SIZE. Nếu

"direction" là 'R', tức là di chuyển sang phải, thì tăng giá trị x của phần đầu con rắn lên 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;
    }

}
```

- Hàm "checkApple" là một phương thức không được công khai (public method) không trả về giá trị (void). Phương thức này được sử dụng để kiểm tra xem đầu của con rắn có ăn được quả táo hay không trong trò chơi.
  - Kiểm tra xem đầu con rắn có trùng vị trí với quả táo hay không: So sánh tọa độ x[0] của đầu con rắn với tọa độ x của quả táo (appleX) và tọa độ y[0] của đầu con rắn với tọa độ y của quả táo (appleY). Nếu cả hai tọa độ này trùng khớp, tức là

đầu con rắn đang nằm trên quả táo, thì tiến hành thực hiện các bước sau.

- Tăng số phần thân của con rắn (bodyParts): Tăng giá trị của biến bodyParts lên 1. Điều này có nghĩa là con rắn sẽ có thêm một phần thân sau khi ăn được quả táo.

- Tăng số quả táo đã ăn (applesEaten): Tăng giá trị của biến applesEaten lên 1. Điều này đếm số lượng quả táo đã được ăn trong trò chơi.

- Tạo một quả táo mới (newApple): Gọi phương thức newApple() để tạo một vị trí ngẫu nhiên cho quả táo mới, sau khi con rắn đã ăn quả táo trước đó.
- Hàm updatePlayerPos: Cập nhập vị trí của player và sự tác động của player với bóng

```
public void checkApple() {
    if((x[0] == appleX) && (y[0] == appleY)) {
        bodyParts++;
        applesEaten++;
        newApple();
    }
}
```

- Hàm "checkCollisions" là một phương thức không được công khai (public method) không trả về giá trị (void). Phương thức này được sử dụng để kiểm tra va chạm trong trò chơi.
  - Kiểm tra va chạm giữa đầu con rắn và phần thân: Sử dụng một vòng lặp for để duyệt qua các phần thân của con rắn, bắt đầu từ phần thân thứ 1 đến phần thân cuối cùng. Kiểm tra xem tọa độ của đầu con rắn (x[0], y[0]) có trùng với tọa độ của phần thân (x[i], y[i]) hay không. Nếu có va chạm, tức là đầu

con rắn chạm vào phần thân, thì đặt biến "running" thành false để kết thúc trò chơi.

- Kiểm tra va chạm với biên trái: Kiểm tra xem tọa độ x của đầu con rắn (x[0]) có nhỏ hơn 0 hay không. Nếu đầu con rắn chạm biên trái, thì đặt biến "running" thành false.

- Kiểm tra va chạm với biên phải: Kiểm tra xem tọa độ x của đầu con rắn (x[0]) có lớn hơn giá trị SCREEN_WIDTH (chiều rộng màn hình) hay không. Nếu đầu con rắn chạm biên phải, thì đặt biến "running" thành false.

- Kiểm tra va chạm với biên trên: Kiểm tra xem tọa độ y của đầu con rắn (y[0]) có nhỏ hơn 0 hay không. Nếu đầu con rắn chạm biên trên, thì đặt biến "running" thành false.

- Kiểm tra va chạm với biên dưới: Kiểm tra xem tọa độ y của đầu con rắn (y[0]) có lớn hơn giá trị SCREEN_HEIGHT (chiều cao màn hình) hay không. Nếu đầu con rắn chạm biên dưới, thì đặt biến "running" thành false.

- Dừng bộ đếm thời gian (timer): Nếu biến "running" là false, tức là đã xảy ra va chạm hoặc kết thúc trò chơi, thì gọi

phương thức stop() của đối tượng Timer để dừng bộ đếm thời gian.

```
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();
    }
}
```

- Hàm "gameOver" là một phương thức không được công khai (public method) không trả về giá trị (void). Phương thức này được sử dụng để hiển thị màn hình kết thúc trò chơi khi trò chơi kết thúc.
- Hiển thị điểm số (Score): Đặt màu vẽ (Color) là màu đỏ (Color.red). Đặt font chữ và kích thước chữ (Font) là "Ink Free", đậm (Font.BOLD), và có kích thước 40. Sử dụng đối tượng FontMetrics để lấy thông tin về kích thước của chuỗi văn bản. Vẽ chuỗi "Score: " kết hợp với số quả táo đã ăn (applesEaten) bằng cách sử dụng phương thức drawString() của đối tượng Graphics. Chuỗi được vẽ ở giữa màn hình theo chiều ngang, và vị trí theo chiều dọc là kích thước của font chữ.
-

- Hiển thị văn bản "Game Over": Đặt màu vẽ (Color) là màu đỏ (Color.red). Đặt font chữ và kích thước chữ (Font) là "Ink Free", đậm (Font.BOLD), và có kích thước 75. Sử dụng đối tượng FontMetrics để lấy thông tin về kích thước của chuỗi văn bản. Vẽ chuỗi "Game Over" ở giữa màn hình theo cả chiều ngang và chiều dọc, sử dụng phương thức drawString() của đối tượng Graphics.

```
public void gameOver(Graphics g) {
    //Score
    g.setColor(Color.red);
    g.setFont( new Font("Ink Free",Font.BOLD, 40));
    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.red);
    g.setFont( new Font("Ink Free",Font.BOLD, 75));
    FontMetrics metrics2 = getFontMetrics(g.getFont());
    g.drawString("Game Over", (SCREEN_WIDTH - metrics2.stringWidth("Game Over"))/2, SCREEN_HEIGHT/2);
}
```

- Hàm "actionPerformed" là một phương thức không được công khai (public method) không trả về giá trị (void). Phương thức này được gọi khi có sự kiện (ActionEvent) xảy ra.
  - Kiểm tra trạng thái của trò chơi: Kiểm tra biến "running" để đảm bảo rằng trò chơi đang chạy và biến "isPaused" để kiểm tra xem trò chơi có bị tạm dừng hay không.
  -
  - Di chuyển con rắn: Nếu trò chơi đang chạy và không bị tạm dừng, gọi phương thức "move()" để di chuyển con rắn.
  -
  - Kiểm tra va chạm với quả táo: Nếu trò chơi đang chạy và không bị tạm dừng, gọi phương thức "checkApple()" để kiểm tra xem con rắn có ăn được quả táo hay không.
  -
  - Kiểm tra va chạm: Nếu trò chơi đang chạy và không bị tạm dừng, gọi phương thức "checkCollisions()" để kiểm tra va chạm giữa con rắn và các đối tượng khác (phần thân, biên màn hình).
  -

- Vẽ lại màn hình: Gọi phương thức "repaint()" để vẽ lại màn hình, hiển thị các thay đổi mới nhất trong trạng thái của trò chơi.

```
public void actionPerformed(ActionEvent e) {

    if(running && !isPaused) {
        move();
        checkApple();
        checkCollisions();
    }
    repaint();
}
```

- Hàm "MyKeyAdapter" là một lớp con của lớp "KeyAdapter" và được sử dụng để xử lý sự kiện nhấn phím trong trò chơi.

- Phương thức "keyPressed(KeyEvent e)" được ghi đè (override) từ lớp cha "KeyAdapter" để xử lý sự kiện khi một phím được nhấn.

- Kiểm tra phím được nhấn: Sử dụng câu lệnh switch để kiểm tra mã của phím được nhấn (keyCode) từ đối tượng KeyEvent.

- Xử lý các phím mũi tên (LEFT, RIGHT, UP, DOWN): Kiểm tra mã phím và thay đổi hướng di chuyển của con rắn dựa trên hướng di chuyển hiện tại và trạng thái tạm dừng. Nếu hướng di chuyển mới không trái ngược với hướng di chuyển hiện tại và trò chơi không bị tạm dừng, thì hướng di chuyển mới được thiết lập.

- Xử lý phím SPACE: Kiểm tra mã phím và thực hiện hành động tương ứng. Nếu trò chơi đang chạy, thì việc nhấn phím SPACE sẽ làm tạm dừng hoặc tiếp tục trò chơi bằng cách đảo ngược giá trị của biến "isPaused". Nếu trò chơi đã kết thúc, thì việc nhấn phím SPACE sẽ bắt đầu trò chơi mới bằng cách gọi phương thức "startGame()".

```
public class MyKeyAdapter extends KeyAdapter{
    @Override
    public void keyPressed(KeyEvent e) {
        switch(e.getKeyCode()) {
        case KeyEvent.VK_LEFT:
            if (direction != 'R' && !isPaused) {
                direction = 'L';
            }
            break;
        case KeyEvent.VK_RIGHT:
            if (direction != 'L' && !isPaused) {
                direction = 'R';
            }
            break;
        case KeyEvent.VK_UP:
            if (direction != 'D' && !isPaused) {
                direction = 'U';
            }
            break;
        case KeyEvent.VK_DOWN:
            if (direction != 'U' && !isPaused) {
                direction = 'D';
            }
            break;
        case KeyEvent.VK_SPACE:
            if (running) {
                isPaused = !isPaused;
            }
            else {
                startGame();
            }
            break;
    }
    }
}
```

4.3 Lớp SnakeGame:

- Mô tả: Lớp "SnakeGame" là một lớp công cộng (public class) và chứa phương thức "main". Đây là lớp chính của chương trình trò chơi rắn.
- Phương thức "main" được sử dụng để khởi chạy chương trình trò chơi. Trong phương thức này, tạo một đối tượng của lớp "GameFrame" và gán nó cho biến mới. Điều này sẽ tạo ra một cửa sổ trò chơi và bắt đầu trò chơi.

```
public class SnakeGame {

    public static void main(String[] args) {

        new GameFrame();
    }
}
```

4.4 Folder src:

- Nội Dung: Thư mục nơi chứa dữ liệu chính của trò chơi

5. Kết Quả-Demo

- CHƠI: Khi bắt đầu trò chơi thì con rắn sẽ ở vị trí ngẫu nhiên và quả táo cũng sẽ ở vị trí ngẫu nhiên trên cửa sổ trò chơi. Người chơi phải di chuyển con rắn sao cho con rắn phải ăn được quả táo.
- Score: hiển thị điểm mà người chơi cho rắn ăn được táo(mỗi quả táo ăn được sẽ được tính là 1 điểm)