2024 OOP Programming Class02 Team 13

Project #4

with Java

[Team 13]

[redacted]

2024 OOP Programming Class02 Team 13

Chapter_

01_ Project Introduction ;

02_ Rules of Game ;

03_ Used Libraries ;

04_ Major Function ;

05_ Execution Video ;

2024 OOP Programming Class02 Team 13

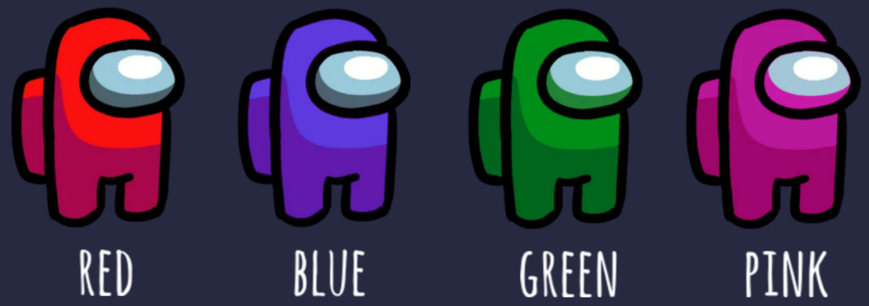
01_ Project Introduction

This project implements a 2D running game using Java Swing. The game features a mechanism where the player moves left and right to avoid obstacles and collect coins.

02_ Rules of Game

Before the game starts

1. Click GAME START.
2. Select your own character.



- If you click YES, the game will start.
- If you click NO, you can choose another character.

2024 | OOP Programming | Class02 | Team 13

02_ Rules of Game

During the game

- You have to move to the left and right to avoid obstacles called "Impostor"  and get coins 🪙.
- If you crash 3 times with the Impostor, the game will be over.

2024 OOP Programming Class02 Team 13

02_ Rules of Game

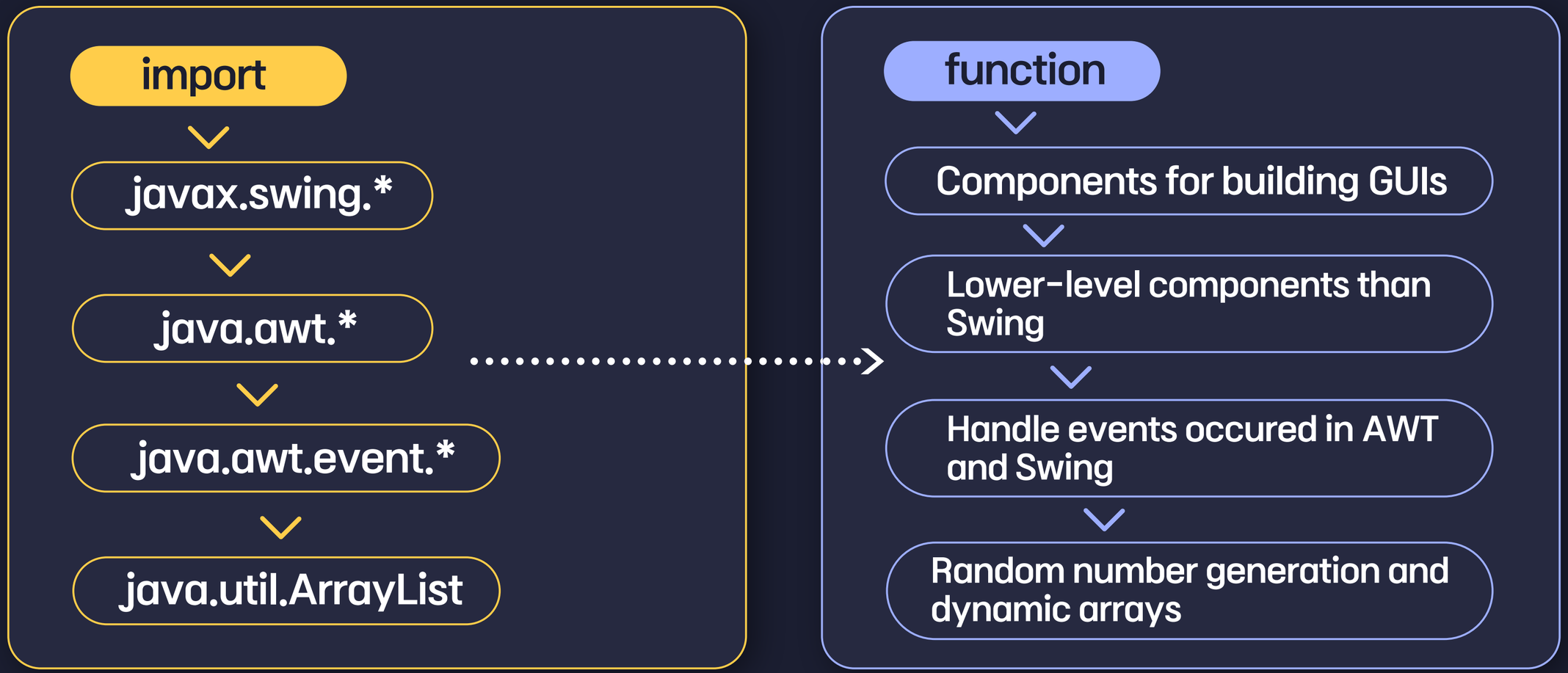
After the game ends

When GAME OVER ,

- If you want to restart the game, click RETRY .
- If you want to end the game, click EXIT .

2024 | OOP Programming | Class02 | Team 13

03_ Used Libraries



2024 | OOP Programming | Class02 | Team 13

04_ Major Fuctions _ Track

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

public class Track {  5 usages
    private final int width, height;  4 usages
    private final Image trackImage;  4 usages
    private final int x;  5 usages
    private int y;  4 usages

    public Track(int x, int y, int width, int height,String imagePath) {  3 usages
        this.x = x;
        this.y = y;
        this.width = width;
        this.height = height;
        this.trackImage = new ImageIcon(imagePath).getImage();
    }

    public void draw(Graphics g) {  1 usage
        g.drawImage(trackImage, x, y, width, height,  observer: null);
        g.drawImage(trackImage, x,  y: y - height, width, height,  observer: null);
        g.drawImage(trackImage, x,  y: y + height, width, height,  observer: null);
    }
    public int getX() { return x; }  8 usages
}
```

Track

- draw()
  : Draws the 3 tracks on the screen.

- getX()
  : Getter method that provides external access to the track's x-coordinate.

2024 OOP Programming Class02 Team 13

04_ Major Fuctions _ Player

```java
public class Player {  5 usages
    private int x, y;  6 usages
    private int width, height;  3 usages
    private int lives;  5 usages
    private int coins;  4 usages
    private int currentTrack;  8 usages
    private int currentFrame;  4 usages
    private int frameDelay;  3 usages

    private final int initialX = 200;  2 usages
    private final int initialY = 300;  2 usages
    private final int MAX_LIVES = 3;  2 usages
    private final int INITIAL_COIN = 0;  2 usages

    private Image[] runFrames;  9 usages

    private boolean isAnimating;  3 usages
```

```java
public Player() {  1 usage
    this.x = initialX;
    this.y = initialY;
    this.lives = MAX_LIVES;
    this.coins = INITIAL_COIN;

    this.currentTrack = 1;
    this.currentFrame = 0;
    this.frameDelay = 0;

    this.runFrames = new Image[0];

    this.isAnimating = false;
}
```

2024 OOP Programming Class02 Team 13

04_ Major Fuctions _ Player

Player

- setPlayerImage()
  : Loads the player's running animation frames.
- update()
  : Implements running animation by switching animation frames.
- moveLeft()
  : Player's movement to the left.

```java
public void setPlayerImage(String imagePath) {  4 usages
    this.runFrames = new Image[4];
    for (int i = 0; i < runFrames.length; i++) {
        runFrames[i] = new ImageIcon( filename: imagePath + "run" + (i + 1) + ".png").getImage();
    }
    this.width = runFrames[0].getWidth( observer: null);
    this.height = runFrames[0].getHeight( observer: null);
    this.isAnimating = true;
}

public void update() {  1 usage
    if (isAnimating) {
        frameDelay++;
        if (frameDelay % 5 == 0) {
            currentFrame = (currentFrame + 1) % runFrames.length;
        }
    }
}

public void moveLeft(boolean leftPressed) {  2 usages
    if(leftPressed){
        if (currentTrack == 1 || currentTrack == 2) {
            currentTrack--;
            x -=100;
        }
    }
}
```

2024 | OOP Programming | Class02 | Team 13

04_ Major Fuctions _ Player

Player

- moveRight()
  : Player's movement to the right.
- draw()
  : Draws the player's current animation frame on the screen.
- hitObstacle()
  : Decreases lives when the player collides with an obstacle.

```
public void moveRight(boolean rightPressed) {  2 usages
    if(rightPressed){
        if (currentTrack == 0 || currentTrack == 1) {
            currentTrack++;
            x += 100;
        }
    }
}

public void draw(Graphics g) {  1 usage
    if (runFrames.length > 0) {
        g.drawImage(runFrames[currentFrame], x, y, width, height,  observer: null);
    }
}

public void hitObstacle() {  1 usage
    if (lives > 0) {
        lives--;
    }
}
public void hitCoin() { coins++; }  1 usage


public int getLives() { return lives; }  2 usages
public int getCoins(){ return coins; }  2 usages

public void resetLives() { this.lives = MAX_LIVES; }  1 usage
public void resetCoins() { this.coins = INITIAL_COIN; }  1 usage
```

2024 | OOP Programming | Class02 | Team 13

04_ Major Fuctions _ Player

Player

- hitCoin()
  : Increases coins when the player collects a coin.
- getLives(), getCoins()
  : Provides external access to the current number of lives/coins.
- resetLives(), resetCoins()
  : Resets the number of lives/coins.

```
public void moveRight(boolean rightPressed) {  2 usages
    if(rightPressed){
        if (currentTrack == 0 || currentTrack == 1) {
            currentTrack++;
            x += 100;
        }
    }
}

public void draw(Graphics g) {  1 usage
    if (runFrames.length > 0) {
        g.drawImage(runFrames[currentFrame], x, y, width, height,  observer: null);
    }
}

public void hitObstacle() {  1 usage
    if (lives > 0) {
        lives--;
    }
}
public void hitCoin() { coins++; }  1 usage


public int getLives() { return lives; }  2 usages
public int getCoins(){ return coins; }  2 usages

public void resetLives() { this.lives = MAX_LIVES; }  1 usage
public void resetCoins() { this.coins = INITIAL_COIN; }  1 usage
```

2024 OOP Programming Class02 Team 13

04_ Major Fuctions _ Player

Player

- resetPosition()
  : Resets the player's position to the initial state.
- getBounds()
  : Defines the player's collision area for detecting collisions with obstacles and coins.

```
    public void resetPosition() {  1 usage
        this.x = initialX;
        this.y = initialY;
        this.currentTrack = 1;
    }

    public Rectangle getBounds() {  3 usages
        return new Rectangle(x, y, width, height);
    }
}
```

2024 | OOP Programming | Class02 | Team 13

04_ Major Fuctions _ Coin

```java
import java.awt.*;

public class Coin {  5 usages
    private int x, y;  4 usages
    private final int width, height;  3 usages
    private final Image coinImage;  3 usages

    public Coin(int x, int y, int width, int height, Image coinImage){  1 usage
        this.x = x;
        this.y = y;
        this.width = width;
        this.height = height;
        this.coinImage = coinImage;
    }

    public void draw(Graphics g) {  1 usage
        if (coinImage != null) {
            g.drawImage(coinImage, x, y, width, height,  observer: null);
        }
    }

    public boolean checkCollision(Player player) {  1 usage
        return player.getBounds().intersects(new Rectangle(x, y, width, height));
    }

    public int getY() { return y; }  2 usages

    public void setX(int x) { this.x = x; }  3 usages
    public void setY(int y) { this.y = y; }  4 usages
}
```

Coin

- draw()
  : Draws the coin's image on the screen.
- checkCollision()
  : Checks whether the player has collided with the coin and returns the collision status as a boolean value (true or false).

2024 | OOP Programming | Class02 | Team 13

04_ Major Fuctions _ Coin

```java
import java.awt.*;

public class Coin {  5 usages
    private int x, y;  4 usages
    private final int width, height;  3 usages
    private final Image coinImage;  3 usages

    public Coin(int x, int y, int width, int height, Image coinImage){  1 usage
        this.x = x;
        this.y = y;
        this.width = width;
        this.height = height;
        this.coinImage = coinImage;
    }

    public void draw(Graphics g) {  1 usage
        if (coinImage != null) {
            g.drawImage(coinImage, x, y, width, height,  observer: null);
        }
    }

    public boolean checkCollision(Player player) {  1 usage
        return player.getBounds().intersects(new Rectangle(x, y, width, height));
    }

    public int getY() { return y; }  2 usages

    public void setX(int x) { this.x = x; }  3 usages
    public void setY(int y) { this.y = y; }  4 usages
}
```

Coin

- getY()
  : Provides external access to the coin's current y-coordinate.

- setX(), setY()
  : Allows external setting of the coin's x-coordinate/y-coordinate.

2024 | OOP Programming | Class02 | Team 13

04_ Major Fuctions _ Obstacle

Obstacle

- draw()
  : Draws the obstacle's image on the screen.
- checkCollision()
  : Checks whether the player has collided with the obstacle and returns the collision status as a boolean value (true or false).

```java
import java.awt.*;

public class Obstacle {  5 usages
    private int x, y;  4 usages
    private final int width, height;  3 usages
    private final Image obstacleImage;  3 usages

    public Obstacle(int x, int y, int width, int height, Image obstacleImage) {  1 usage
        this.x = x;
        this.y = y;
        this.width = width;
        this.height = height;
        this.obstacleImage = obstacleImage;
    }

    public void draw(Graphics g) {  1 usage
        if (obstacleImage != null) {
            g.drawImage(obstacleImage, x, y, width, height,  observer: null);
        }
    }

    public boolean checkCollision(Player player) {  1 usage
        return player.getBounds().intersects(new Rectangle(x, y, width, height));
    }

    public int getY() { return y; }  2 usages

    public void setX(int x) { this.x = x; }  3 usages
    public void setY(int y) { this.y = y; }  4 usages
}
```

2024 | OOP Programming | Class02 | Team 13

04_ Major Fuctions _ Obstacle

Obstacle

getY()

: Provides external access to the obstacle's current y-coordinate.

- **setX, setY()**

: Allows external setting of the obstacle's x-coordinate/y-coordinate.

```java
import java.awt.*;

public class Obstacle {  5 usages
    private int x, y;  4 usages
    private final int width, height;  3 usages
    private final Image obstacleImage;  3 usages

    public Obstacle(int x, int y, int width, int height, Image obstacleImage) {  1 usage
        this.x = x;
        this.y = y;
        this.width = width;
        this.height = height;
        this.obstacleImage = obstacleImage;
    }

    public void draw(Graphics g) {  1 usage
        if (obstacleImage != null) {
            g.drawImage(obstacleImage, x, y, width, height,  observer: null);
        }
    }

    public boolean checkCollision(Player player) {  1 usage
        return player.getBounds().intersects(new Rectangle(x, y, width, height));
    }

    public int getY() { return y; }  2 usages

    public void setX(int x) { this.x = x; }  3 usages
    public void setY(int y) { this.y = y; }  4 usages
}
```

2024 | OOP Programming | Class02 | Team 13

04_ Major Fuctions _ RunningGame

```java
public class RunningGame extends JPanel implements ActionListener, KeyListener {  2 usages

    private final Image startBackground;  3 usages
    private final Image characterBackground;  3 usages
    private final Image heartImage;  2 usages
    private final Image coinImage;  3 usages
    private final Image obstacleImage;  2 usages
    private Image confirm;  5 usages
    private Image gameOver;  3 usages

    private JButton startButton;  4 usages
    private JButton yesButton, noButton;  7 usages
    private JButton retryButton, exitButton;  3 usages
    private JButton player1Button, player2Button, player3Button, player4Button;  4 usages

    private final Timer timer;  2 usages
    private final Player player;  22 usages
    private final  ArrayList<Track> tracks;  21 usages
    private final ArrayList<Obstacle> obstacles;  5 usages
    private final ArrayList<Coin> coins;  5 usages

    private int backgroundY1, backgroundY2;  4 usages
    private final int scrollSpeed = 5;  4 usages
    private final int BASE_HEIGHT = 600;  10 usages

    private boolean isGameStarted = false;  5 usages
    private boolean isPlayerSelected = false;  5 usages
    private boolean isGameOver = false;  4 usages
    private boolean leftPressed = false;  4 usages
    private boolean rightPressed = false;  4 usages
```

RunningGame

- This class serves as the main game panel, rendering and updating all game elements.
- Extends JPanel to enhance its functionality as a custom panel and implements ActionListener and KeyListener interfaces to handle action events and keyboard events.

2024 | OOP Programming | Class02 | Team 13

04_ Major Fuctions _ RunningGame

```java
public RunningGame() {  1 usage
    setLayout(null);

    startBackground = new ImageIcon( filename: "image/start_2.png").getImage();
    characterBackground = new ImageIcon( filename: "image/background.png").getImage();
    heartImage = new ImageIcon( filename: "image/life.png").getImage();
    coinImage = new ImageIcon( filename: "image/coin.png").getImage();
    obstacleImage = new ImageIcon( filename: "image/impostor.png").getImage();

    startButton();

    player = new Player();
    tracks = new ArrayList<>();
    obstacles = new ArrayList<>();
    coins = new ArrayList<>();

    tracks.add(new Track( x: 100,  y: 0,  width: 50, BASE_HEIGHT,  imagePath: "image/track.png"));
    tracks.add(new Track( x: 200,  y: 0,  width: 50, BASE_HEIGHT,  imagePath: "image/track.png"));
    tracks.add(new Track( x: 300,  y: 0,  width: 50, BASE_HEIGHT,  imagePath: "image/track.png"));

    addObstacle( y: -200);
    addObstacle( y: -400);
    addObstacle(  y: 0);

    addCoin( y: -100);
    addCoin( y: -300);
    addCoin( y: -500);

    backgroundY1 = 0;
    backgroundY2 = -BASE_HEIGHT;
    timer = new Timer( delay: 30,  listener: this);
    timer.start();

    setFocusable(true);
    addKeyListener( l: this);
}
```

RunningGame

- Layout and Image Loading
- Game Object Initialization
- Adding Tracks, Obstacles, and Coins
- Background , Timer, Input Handling Setup

2024 | OOP Programming | Class02 | Team 13

04_ Major Fuctions _ RunningGame

```java
private JButton createButton(String imagePath, int x, int y, int width, int height) {  9 usages
    JButton button = new JButton(new ImageIcon(new ImageIcon(imagePath).
            getImage().getScaledInstance(width, height, Image.SCALE_SMOOTH)));
    button.setBounds(x, y, width, height);
    button.setBorderPainted(false);
    button.setContentAreaFilled(false);
    button.setFocusPainted(false);
    button.setOpaque(false);
    return button;
}

private void startButton() {  2 usages
    startButton = createButton( imagePath: "image/start button(title).png",  x: 160,  y: 275,  width: 175,  height: 45);

    add(startButton);

    startButton.addActionListener( ActionEvent e -> {
        isGameStarted = true;
        remove(startButton);
        showPlayerSelection();
        startGame();
        repaint();
    });

}
```

RunningGame

- createButton()
  : Helps to easily create buttons with custom images.

- startButton()
  : Initializes the game start button and adds it to the screen.

2024 | OOP Programming | Class02 | Team 13

04_ Major Fuctions _ RunningGame

```
private void showPlayerSelection() {  1 usage
    setLayout(null);

    player1Button = createButton( imagePath: "image/red/ch.png",  x: 25,  y: 190,  width: 100,  height: 150);
    player2Button = createButton( imagePath: "image/blue/ch.png",  x: 140,  y: 190,  width: 100,  height: 150);
    player3Button = createButton( imagePath: "image/green/ch.png",  x: 260,  y: 190,  width: 100,  height: 150);
    player4Button = createButton( imagePath: "image/pink/ch.png",  x: 375,  y: 190,  width: 100,  height: 150);

    add(player1Button);
    add(player2Button);
    add(player3Button);
    add(player4Button);

    player1Button.addActionListener( ActionEvent e -> {
        player.setPlayerImage("image/red/");
        showConfirmButton();
    });

    player2Button.addActionListener( ActionEvent e -> {
        player.setPlayerImage("image/blue/");
        showConfirmButton();
    });

    player3Button.addActionListener( ActionEvent e -> {
        player.setPlayerImage("image/green/");
        showConfirmButton();
    });

    player4Button.addActionListener( ActionEvent e -> {
        player.setPlayerImage("image/pink/");
        showConfirmButton();
    });

    repaint();
}
```

RunningGame

- showPlayerSelection()
  : Displays the player character selection screen.

2024 | OOP Programming | Class02 | Team 13

04_ Major Fuctions _ RunningGame

```
private void showConfirmButton() {  4 usages
    confirm = new ImageIcon( filename: "image/select character_2.png").getImage();

    yesButton = createButton( imagePath: "image/yes.png",  x: 140,  y: 130,  width: 29,  height: 28);
    noButton = createButton( imagePath: "image/no.png",  x: 320,  y: 130,  width: 27,  height: 25);

    add(yesButton);
    add(noButton);

    yesButton.addActionListener( ActionEvent e -> {
        isPlayerSelected = true;

        remove(player1Button);
        remove(player2Button);
        remove(player3Button);
        remove(player4Button);
        startGame();
        repaint();
    });

    noButton.addActionListener( ActionEvent e -> {
        isPlayerSelected = false;
        confirm = null;
        yesButton.setVisible(false);
        noButton.setVisible(false);
        repaint();
    });

    yesButton.setVisible(true);
    yesButton.setEnabled(true);
    noButton.setVisible(true);
    noButton.setEnabled(true);
}
```

RunningGame

- showConfirmButton()
  : Displays confirmation buttons (Yes or No) after selecting a player character.

2024 | OOP Programming | Class02 | Team 13

04_ Major Fuctions _ RunningGame

```
private void showGameOver() {  1 usage
    setLayout(null);

    gameOver = new ImageIcon( filename: "image/game over.png").getImage();

    retryButton = createButton( imagePath: "image/retry.png",  x: 120,  y: 230,  width: 73,  height: 39);
    exitButton = createButton( imagePath: "image/exit.png",  x: 320,  y: 230,  width: 56,  height: 39);

    add(retryButton);
    add(exitButton);

    retryButton.addActionListener( ActionEvent e -> resetGame());
    exitButton.addActionListener( ActionEvent e -> System.exit( status: 0));

    revalidate();
    repaint();
}

private void startGame() {  2 usages
    if (isGameStarted && isPlayerSelected) {
        removeAll();
        revalidate();
        repaint();
    }
}
```

RunningGame

- showGameOver()
  : Displays the game over screen and provides options to restart or exit.

- startGame()
  : Removes previous UI components and prepares the game screen before starting the game.

2024 OOP Programming Class02 Team 13

04_ Major Fuctions _ RunningGame

```
private void resetGame() {  1 usage
    isGameOver = false;
    isGameStarted = false;
    isPlayerSelected = false;

    player.resetLives();
    player.resetPosition();
    player.resetCoins();

    int obstacley = -200;
    for (Obstacle obstacle : obstacles) {
        int trackIndex = (int) (Math.random() * tracks.size());
        obstacle.setX(tracks.get(trackIndex).getX());
        obstacle.setY(obstacley);
        obstacley -= 100;
    }

    int coiny = -300;
    for (Coin coin : coins) {
        int trackIndex = (int) (Math.random() * tracks.size());
        coin.setX(tracks.get(trackIndex).getX());
        coin.setY(coiny);
        coiny -= 200;
    }

    removeAll();
    revalidate();
    repaint();

    startButton();
}
```

RunningGame

- **resetGame()**
  : Resets and reinitializes the game state when restarting the game after game over.

2024 | OOP Programming | Class02 | Team 13

04_ Major Fuctions _ RunningGame

```
private void addObstacle(int y) {  3 usages
    int trackIndex = (int) (Math.random() * tracks.size());
    int trackX = tracks.get(trackIndex).getX();

    obstacles.add(new Obstacle(trackX, y,  width: 50,  height: 50, obstacleImage));
}

private void addCoin(int y) {  3 usages
    int trackIndex = (int) (Math.random() * tracks.size());
    int trackX = tracks.get(trackIndex).getX();

    coins.add(new Coin(trackX, y,  width: 50,  height: 50, coinImage));
}
```

RunningGame

- addObstacle(), addCoin()

  : Randomly selects a track index and adds a new obstacle and a new coin at the specified Y-coordinate.

2024 | OOP Programming | Class02 | Team 13

04_ Major Fuctions _ RunningGame

```java
@Override
protected void paintComponent(Graphics g) {
    super.paintComponent(g);

    if (!isGameStarted) {
        if (startBackground != null) {
            g.drawImage(startBackground, x: 0, y: 0, getWidth(), getHeight(), observer: null);
        }
    } else if (isGameOver) {
        g.drawImage(characterBackground, x: 0, y: 0, getWidth(), getHeight(), observer: null);
        if (gameOver != null) {
            g.drawImage(gameOver, x: 100, y: 100, width: 290, height: 76, observer: this);
        }
        g.setColor(Color.WHITE);
        g.setFont(new Font( name: "Arial", Font.BOLD, size: 20));
        g.drawString( str: "Your Score : " + player.getCoins() + " !", x: 170, y: 210);
    } else {
        g.drawImage(characterBackground, x: 0, y: 0, getWidth(), getHeight(), observer: null);

        if (confirm != null) {
            g.drawImage(confirm, x: 65, y: 30, width: 363, height: 76, observer: this);
        }
        if (isPlayerSelected) {
            confirm = null;
            yesButton.setVisible(false);
            noButton.setVisible(false);

            for(Track track : tracks){
                track.draw(g);
            }
            for (Obstacle obstacle : obstacles) {
                obstacle.draw(g);
            }
            for (Obstacle obstacle : obstacles) {
                obstacle.draw(g);
            }
            for (Coin coin : coins) {
                coin.draw(g);
            }

            for (int i = 0; i < player.getLives(); i++) {
                g.drawImage(heartImage, x: 10 + (i * 35), y: 10, width: 30, height: 30, observer: null);
            }

            g.setColor(Color.YELLOW);
            g.setFont(new Font( name: "Arial", Font.BOLD, size: 18));
            g.drawImage(coinImage, x: 10, y: 40, width: 30, height: 30, observer: null);
            g.drawString( str: "       X " + player.getCoins(), x: 10, y: 60);
            player.draw(g);
        }
    }
}
```

RunningGame

- Renders various graphic elements according to the current state of the game.

2024 | OOP Programming | Class02 | Team 13

04_ Major Fuctions _ RunningGame

```java
@Override
public void actionPerformed(ActionEvent e) {
    if (!isGameStarted || isGameOver) return;
    backgroundY1 += scrollSpeed;
    backgroundY2 += scrollSpeed;

    if (backgroundY1 >= BASE_HEIGHT) {
        backgroundY1 = -BASE_HEIGHT;
    }
    if (backgroundY2 >= BASE_HEIGHT) {
        backgroundY2 = -BASE_HEIGHT;
    }

    for (Obstacle obstacle : obstacles) {
        obstacle.setY(obstacle.getY() + scrollSpeed);

        if (obstacle.getY() > BASE_HEIGHT) {
            int trackIndex = (int) (Math.random() * tracks.size());
            obstacle.setY(-200);
            obstacle.setX(tracks.get(trackIndex).getX());
        }

        if (obstacle.checkCollision(player)) {
            player.hitObstacle();

            int trackIndex = (int) (Math.random() * tracks.size());
            obstacle.setY(-200);
            obstacle.setX(tracks.get(trackIndex).getX());

            if (player.getLives() <= 0) {
                isGameOver = true;
                showGameOver();
            }
        }
    }
```

```java
    for (Coin coin : coins) {
        coin.setY(coin.getY() + scrollSpeed);

        if (coin.getY() > BASE_HEIGHT) {
            int trackIndex = (int) (Math.random() * tracks.size());
            coin.setY(-200);
            coin.setX(tracks.get(trackIndex).getX());
        }

        if(coin.checkCollision(player)){
            player.hitCoin();
            int trackIndex = (int) (Math.random() * tracks.size());
            coin.setY(-200);
            coin.setX(tracks.get(trackIndex).getX());
        }
    }

    player.update();
    repaint();
}
```

RunningGame

- Handles the main operations of the game loop to update the game state.

2024 OOP Programming Class02 Team 13

04_ Major Fuctions _ RunningGame

```java
@Override
public void keyPressed(KeyEvent e) {
    int key = e.getKeyCode();

    if (key == KeyEvent.VK_LEFT) {
        leftPressed = true;
        player.moveLeft(leftPressed);
    } else if (key == KeyEvent.VK_RIGHT) {
        rightPressed = true;
        player.moveRight(rightPressed);
    }
}

@Override
public void keyReleased(KeyEvent e) {
    int key = e.getKeyCode();

    if (key == KeyEvent.VK_LEFT) {
        leftPressed = false;
        player.moveLeft(leftPressed);
    } else if (key == KeyEvent.VK_RIGHT) {
        rightPressed = false;
        player.moveRight(rightPressed);
    }
}

@Override
public void keyTyped(KeyEvent e) {}
}
```

RunningGame

- keyPressed(), keyReleased()
  : Starts/Stops the player's movement when the left or right key on the keyboard is pressed/released.
- keyTyped()
  : Although it is a required method of the KeyListener interface, it is not used in the current implementation.

2024 OOP Programming Class02 Team 13

04_ Major Fuctions _ Main

```java
public class Main {
    public static void main(String[] args) {
        javax.swing.JFrame frame = new javax.swing.JFrame( title: "Among us run ");
        RunningGame gamePanel = new RunningGame();

        frame.add(gamePanel);
        frame.setSize( width: 500, height: 400);
        frame.setDefaultCloseOperation(javax.swing.JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

Main

- main()

  : Creates a simple GUI application using Java Swing and runs the game by adding a custom panel called RunningGame to the frame.

2024 OOP Programming Class02 Team 13

04_ Execution Video

2024 OOP Programming Class02 Team 13

Thank You