Simple Maze Game

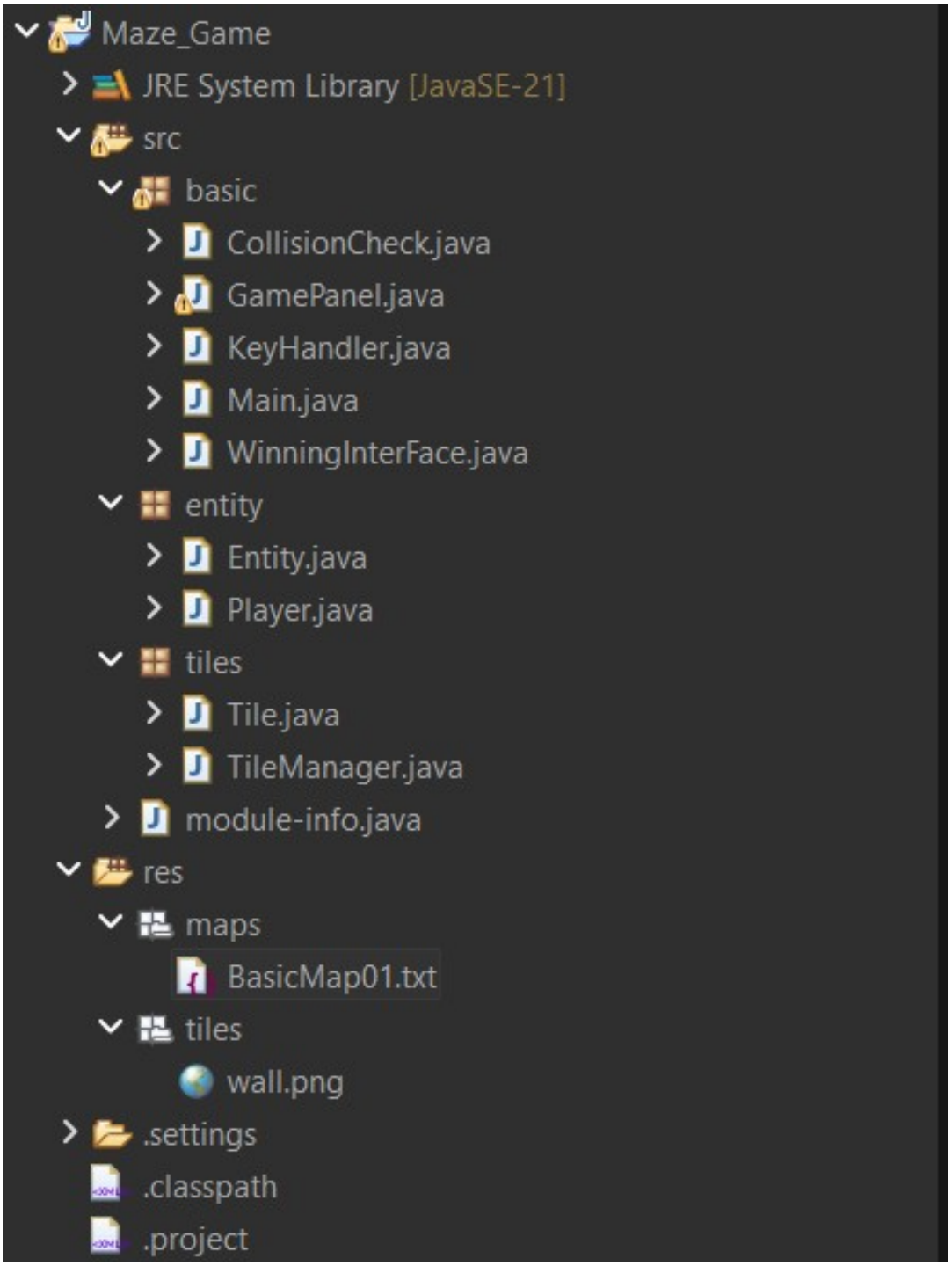
CST 124 - 2 -OBJECT ORIENTED PROGRAMMING

UWU/IIT/23/097
W.A.D.T.Udayanga.

DEPARTMENT OF COMPUTER SCIENCE & INFORMATICS
UVA WELLASSA UNIVERSITY

Introduction
I created a simple Java game using Swing GUI. I wanted to make a fun and entertaining application, so I developed a simple maze game.

File Path



Java Code

-src Folder

basic Package

1. Main.Java

```java
package basic;

import javax.swing.JFrame;

public class Main
{

    public static void main(String[] args)
    {
        //basic GUI
        JFrame window = new JFrame();
        window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        window.setResizable(false);
        window.setTitle("New Adventure");

        GamePanel gamePanel = new GamePanel();

        window.add(gamePanel);
        window.pack();

        window.setLocationRelativeTo(null);
        window.setVisible(true);

        gamePanel.gameStart();

    }

}
```

Explain,
- basic GUI and Create some Object
- Game Thread Start here, inside the main method.

2. GamePanel.Java

```java
package basic;

import java.awt.Color;
import java.awt.Dimension;
import java.awt.Graphics;
import java.awt.Graphics2D;

import entity.*;
import tiles.TileManager;

import javax.swing.JPanel;

public class GamePanel extends JPanel implements Runnable
{
    //screen setting

    final int originalTileSize = 16;
    final int scale = 3;

    public final int TileSize = originalTileSize * scale;

    public final int maxColumn = 30;
    public final int maxRow = 15;

    final int screenWidth = maxColumn * TileSize;
    final int screenHeight = maxRow * TileSize;

    //Thread variable
    Thread gameThread;


    //key handler OBJ
     KeyHandler keyH = new KeyHandler();

     //player OBJ
     Player player = new Player(this,keyH);

     public CollisionCheck cCheck = new CollisionCheck(this);
```

```java
      //map OBJ
      TileManager tileManager = new TileManager(this);

      //create FPS
      int FPS = 60;

    public GamePanel()
    {
        this.setPreferredSize(new Dimension(screenWidth,screenHeight));
        this.setBackground(Color.black);
        this.setDoubleBuffered(true);

        //add keyH
        this.addKeyListener(keyH);
        this.setFocusable(true);
    }

    public void gameStart()
    {
        gameThread = new Thread(this);
        gameThread.start();
    }
```

```java
public void run()
{
    //FPS time cal

    double drawInterval = 1000000000/FPS;

    double nextDrawTime = System.nanoTime() + drawInterval;

    while(gameThread != null)
    {
        //System.out.println("Running");

        player.update();

        repaint();
        try
        {
            double remainTime = nextDrawTime - System.nanoTime();
            remainTime = remainTime/1000000;

            if(remainTime < 0)
                remainTime = 0;

            Thread.sleep((long)remainTime);


            nextDrawTime += drawInterval;
        }
        catch(Exception e)
        {

        }


    }
```

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

        Graphics2D g2 = (Graphics2D)g;

        tileManager.Draw(g2);

        player.Draw(g2);
    }
```

Explain,

- Screen setting Variable
- Thread Variable and Object

-Function

- **GameStart()**
  -Game Thread Start

- **Run()**
  -Inside the run method It create FPS time Calculation

- **PaintComponent()**
  -Inside the paint Component call the Tile Draw Function and Player Draw Function.

3.KeyHandler.Java

```java
package basic;

import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;

public class KeyHandler implements KeyListener
{

    public boolean up,down,left,right;

    //not use this now
    @Override
    public void keyTyped(KeyEvent e) {
        // TODO Auto-generated method stub

    }

    @Override
    public void keyPressed(KeyEvent e)
    {
        int code = e.getKeyCode();
        if(code == KeyEvent.VK_W)
            up = true;
        if(code == KeyEvent.VK_S)
            down = true;
        if(code == KeyEvent.VK_A)
            left = true;
        if(code == KeyEvent.VK_D)
            right = true;
    }

    @Override
    public void keyReleased(KeyEvent e)
    {
        int code = e.getKeyCode();
        if(code == KeyEvent.VK_W)
            up = false;
        if(code == KeyEvent.VK_S)
            down = false;
        if(code == KeyEvent.VK_A)
            left = false;
        if(code == KeyEvent.VK_D)
            right = false;
    }


}
```

Explain,

-Created a **KeyHandler** class and implemented the **KeyListener** interface.

-It is used to read user keyboard input

4.CollisionChecker.java

```java
package basic;

import entity.Entity;

public class CollisionCheck
{
    GamePanel gp;
    public CollisionCheck(GamePanel gp)
    {
        this.gp = gp;
    }

    public void checkCollisionOn(Entity entity)
    {
        int playerLeftX = entity.X+2;
        int playerRightX = entity.X + (gp.TileSize/2)-2;
        int playerTop = entity.Y +2 ;
        int playerBottom = entity.Y + (gp.TileSize/2)-2;

        int leftCol = playerLeftX / (gp.TileSize/2);
        int RightCol = playerRightX / (gp.TileSize/2);
        int TopRow = playerTop / (gp.TileSize/2);
        int BottomRow = playerBottom / (gp.TileSize/2);

        int num1 = 0, num2 = 0;

        switch(entity.direction)
        {
        case "up":
            TopRow = (playerTop - entity.speed) / (gp.TileSize/2);
            num1 = gp.tileManager.MapTileNum[TopRow][leftCol];
            num2 = gp.tileManager.MapTileNum[TopRow][RightCol];

            if(gp.tileManager.tile[num1].collision == true || gp.tileManager.tile[num2].collision == true) {
                entity.collision = true;
            }
            break;
```

```java
        case "down":
            BottomRow = (playerBottom + entity.speed) / (gp.TileSize/2);
            num1 = gp.tileManager.MapTileNum[BottomRow][leftCol];
            num2 = gp.tileManager.MapTileNum[BottomRow][RightCol];

            if(gp.tileManager.tile[num1].collision == true || gp.tileManager.tile[num2].collision == true) {
                entity.collision = true;
            }
            break;

        case "left":
            leftCol = ( playerLeftX - entity.speed)/ (gp.TileSize/2);

            num1 = gp.tileManager.MapTileNum[TopRow][leftCol];
            num2 = gp.tileManager.MapTileNum[BottomRow][leftCol];

            if(gp.tileManager.tile[num1].collision == true || gp.tileManager.tile[num2].collision == true)
            {
                entity.collision = true;
            }
            break;

        case "right":
            RightCol = ( playerRightX + entity.speed)/(gp.TileSize/2);

            num1 = gp.tileManager.MapTileNum[TopRow][RightCol];
            num2 = gp.tileManager.MapTileNum[BottomRow][RightCol];

            if(gp.tileManager.tile[num1].collision == true || gp.tileManager.tile[num2].collision == true)
            {
                entity.collision = true;
            }
            break;


        }
```

```
        System.out.println("num1: " + num1 + " collision: " + gp.tileManager.tile[num1].collision);
        System.out.println("num2: " + num2 + " collision: " + gp.tileManager.tile[num2].collision);
        System.out.println(entity.X);
        boolean won = false;

        if(won == false && (entity.X == 1368 && (num1 == 2 || num2 == 2 )))
        {
            won = true;
            gp.gameThread = null;
            new WinningInterFace();
        }


    }

}
```

Include,

- It checks for tile collisions and prevents the player from moving through blocked tiles, allowing movement only along the winning path.

-It also checks if the player has reached the target. When the player does, it calls the winning interface

5.WinningInterFace.java

```java
package basic;

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

public class WinningInterFace {

    public WinningInterFace()
    {

            JLabel label = new JLabel("* You Achieved Your Goal !!!", SwingConstants.CENTER);
            label.setFont(new Font("Segoe UI", Font.BOLD, 28));
            label.setForeground(new Color(255, 215, 0));
            label.setBorder(BorderFactory.createEmptyBorder(40, 10, 20, 10));


            JFrame frame = new JFrame(" You Win!");
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            frame.setSize(500, 300);
            frame.setResizable(false);
            frame.setLayout(new BorderLayout());
            frame.getContentPane().setBackground(new Color(30, 30, 30));
            frame.setLocationRelativeTo(null);

            frame.setVisible(true);
            frame.add(label, BorderLayout.CENTER);

        System.out.println("great Winning!!! ");
    }


}
```

Explain,

- Used basic Swing components to create a winning interface.
- Designed a new frame with labels and attractive text to display the winning message

entity Package

1.Entity.java

```java
package entity;

public class Entity
{
    public int X,Y;
    public int speed;
    public String direction;
    public boolean collision = false;

}
```

2.Player.java

```java
package entity;
import java.awt.*;
import basic.*;

public class Player extends Entity
{

    GamePanel gp;
    KeyHandler keyH;

    public Player(GamePanel gp,KeyHandler keyH)
    {
        this.gp = gp;
        this.keyH = keyH;
        setdefaultValue();
    }
    public void setdefaultValue()
    {
        X = 48;
        Y = 96;
        speed = 3;
        direction = "up";
    }
```

```java
  public void update()
  {
      if(keyH.up == true || keyH.down == true || keyH.left == true || keyH.right == true)
      {
          if(keyH.up == true)
          {
              direction = "up";
          }
          else if(keyH.down == true)
          {
              direction = "down";
          }
          else if(keyH.left == true)
          {
              direction = "left";

          }
          else if(keyH.right == true)
          {
              direction = "right";
          }

          collision = false;
          gp.cCheck.checkCollisionOn(this);

          if(collision == false)
          {
              if(direction.equals("up"))
              {
                  Y -= speed;
              }
              else if(direction.equals("down"))
              {
                  Y += speed;
              }
              else if(direction.equals("left"))
              {
                  X -= speed;
              }
              else if(direction.equals("right"))
              {
                  X += speed;
              }
          }

      }

  }
```

```
    public void Draw(Graphics2D g2)
    {
        g2.setColor(Color.white);
        g2.fillRect(X, Y, gp.TileSize/2, gp.TileSize/2);

        /*g2.setColor(Color.yellow);
        g2.fillRect(1345, 650, 48, 48);*/
        g2.dispose();


    }


}
```

Explain,

- Entity Class Create Some Variable
- Create class player and Extends Entity Class

-**What Player Class Do**,
- Checks the direction the user wants to move and moves the player in that direction at a specific speed.
- Draws the updated player location on the screen.

Function

- setdefaultValue()
  - Set The player Default location and Speed

- update()
  - Checks the direction the user wants to move using conditions and detects collisions along that path. If the path is clear, the player moves; if the area is solid, the player cannot move through it.

- Draw(Graphics2D g2)
  - Draws the updated player location on the screen.

-tiles Package

1.Tile.java

```java
package tiles;

import java.awt.image.BufferedImage;

public class Tile
{

    public BufferedImage image;
    public boolean collision = false;

}
```

2.TileManager.Java

```java
package tiles;
import java.awt.*;
import java.io.*;
import javax.imageio.ImageIO;
import basic.GamePanel;

public class TileManager
{


    GamePanel gp;
    public Tile[] tile;
    public int MapTileNum[][];

    int newMaxRow;
    int newMaxColumn;

    public TileManager(GamePanel gp)
    {
        this.gp = gp;
        tile = new Tile[10];

        newMaxRow = gp.maxRow*2;
        newMaxColumn = gp.maxColumn*2;

        MapTileNum = new int [newMaxRow][newMaxColumn];

        getImage();
        mapLoad();
    }

    public void getImage()
    {
        try
        {
            tile[0] = new Tile();
            tile[0].image = ImageIO.read(getClass().getResource("/tiles/wall.png"));
            tile[0].collision = true;

            tile[1] = new Tile();
            tile[1].collision = false;

            tile[2] = new Tile();
            tile[2].collision = true;


        }
        catch(IOException e)
        {
            e.printStackTrace();
        }

    }

    //*****************************************************************************
```

```java
    //*************************************************************************************
    public void mapLoad()
    {
        try
        {
            InputStream text01 = getClass().getResourceAsStream("/maps/BasicMap01.txt");
            BufferedReader br = new BufferedReader(new InputStreamReader(text01));

            for(int i = 0 ; i < newMaxRow ; i++)
            {
                String textLine = br.readLine();
                String[] number = textLine.split(" ");

                for(int j = 0; j < newMaxColumn ; j++)
                {
                    int num = Integer.parseInt(number[j]);
                    MapTileNum[i][j] = num;
                }
            }

            //test text file
            for(int i = 0 ; i < newMaxRow; i++)
            {
                for(int j = 0; j < newMaxColumn ; j++)
                {
                    System.out.print(MapTileNum[i][j]);
                }
                System.out.println();
            }

        }
        catch(Exception e)
        {
            e.printStackTrace();
        }

    }
```

```java
    public void Draw(Graphics2D g2)
    {
        int tileX = 0;
        int tileY = 0;
        int tileImageNum = 0;
        int newTileSize = gp.TileSize/2;

        for(int i = 0 ; i <newMaxRow; i++ )
        {
            tileX = 0;
            for(int j = 0; j < newMaxColumn ; j++)
            {
                tileImageNum = MapTileNum[i][j];
                if(tileImageNum == 1)
                {
                    tileX += newTileSize;
                    continue;

                }

                if(tileImageNum == 0)
                {
                    g2.drawImage(tile[tileImageNum].image, tileX, tileY, newTileSize, newTileSize, null);
                }

                if(tileImageNum == 2)
                {
                    g2.setColor(Color.white);
                    g2.fillRect(tileX, tileY, gp.TileSize/2, gp.TileSize/2);
                }

                tileX += newTileSize;
            }
            tileY += newTileSize;
        }

    }
}
```

Explain,

-The Tile package and its files are used to manage how tiles are drawn on the screen and to define which tiles have collision properties.

Function

- getImage()
  -It is used to read and load images for the tile set .

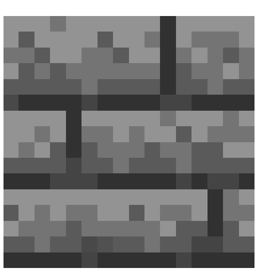
- MapLoad()
  -Reads the **BasicMap01.txt** file and retrieves its values. The program then uses those values to decide what to draw on the screen.

- Draw(Graphics2D g2)
  -Uses those values to draw the tiles on the screen.

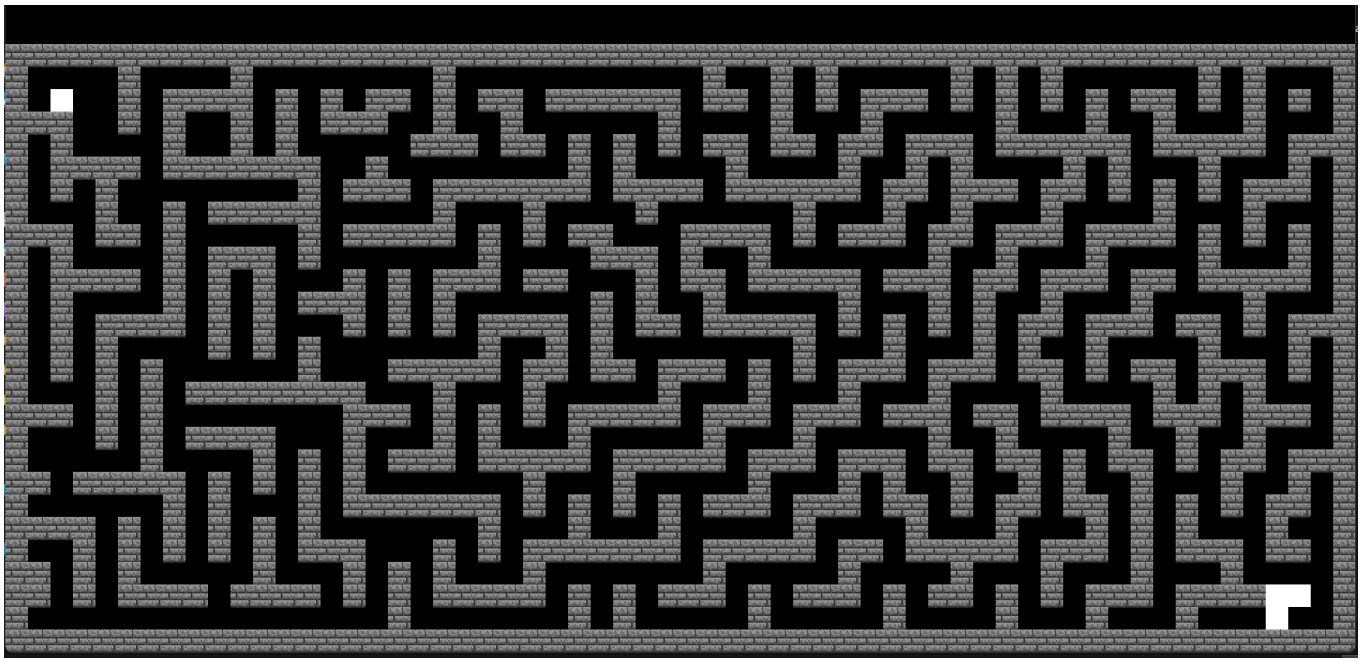
-res Folder
-maps Package
BasicMap01.txt file

```
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 0 1 1 1 1 1 0 1 0 1 0 1 1 1 1 1 1 0 1 0 1 1 1 0
0 1 1 1 1 0 1 0 0 0 0 1 0 1 0 1 0 0 1 0 1 0 0 1 0 0 0 0 0 0 1 0 0 1 0 1 0 1 0 0 0 1 0 1 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1 0
0 0 0 1 1 0 1 0 1 1 0 1 0 1 0 0 0 1 1 0 1 1 0 1 1 1 1 1 1 0 1 1 1 1 0 1 1 1 0 1 1 1 1 1 0 1 1 1 0 1 1 0 1 1 1 0 1 1 1 0
0 1 0 1 1 1 1 0 1 1 0 1 0 1 1 1 1 1 0 0 0 1 0 0 1 0 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0
0 1 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 0 1 1 1 1 0 1 1 0 1 0 1 1 1 1 0 1 0 1 1 1 0 1 1 1 0 1 0
0 1 0 1 0 1 1 1 1 1 1 1 1 0 1 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0 1 0 1 0 1 0 1 0 0 0 1 0
0 1 1 1 0 1 1 0 1 0 0 0 0 0 1 1 1 1 1 0 1 1 1 0 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 0 1 1 0 1 1 1 0 1 1 1 1 0 1 1 1 0 1 1 1 0
0 0 0 1 0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 1 0 1 0 1 0 0 1 1 1 0 0 0 0 1 0 1 0 0 0 1 0 0 1 0 0 0 1 0 0 0 0 1 0 1 0 1 0 1 0
0 1 0 1 1 1 1 0 1 0 0 0 1 0 1 1 1 1 1 1 1 0 1 1 1 1 0 0 0 1 0 1 1 0 1 1 1 1 1 1 1 0 1 1 0 1 1 1 0 1 1 1 1 0 1 1 1 0 1 0
0 1 0 0 0 0 1 0 1 0 1 0 1 1 1 0 1 0 1 0 0 0 1 0 0 1 1 1 0 1 0 0 1 0 0 0 0 0 1 0 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 0 1 0
0 1 0 1 1 1 1 0 1 0 1 0 1 0 0 0 1 0 1 0 1 1 1 1 1 1 0 1 0 1 1 0 1 1 1 1 1 0 1 1 1 0 1 0 1 1 0 1 1 1 0 1 1 1 1 0 1 1 1 0
0 1 0 1 0 0 0 0 1 0 1 0 1 1 1 0 1 0 1 0 1 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 1 0 1 0 1 0 1 0 1 0 0 1 0 0 0 1 0 0 1 0 1 0 0 0
0 1 0 1 0 1 1 1 1 1 0 1 0 1 0 1 1 1 1 1 1 1 0 1 1 0 1 0 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 0 1 0 1 1 0 1 1 1 1 0 1 1 1 0 1 0
0 1 0 1 0 1 0 1 1 1 1 1 1 0 1 1 1 0 0 0 0 0 1 0 0 1 0 0 1 0 0 0 1 0 1 0 1 0 0 0 1 0 0 0 1 0 0 1 0 1 0 0 1 0 0 0 1 0 1 0
0 1 1 1 0 1 0 1 0 0 0 0 0 0 0 0 1 1 1 0 1 1 1 0 1 1 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 1 0 1 1 1 1 0 1 0 1 0 1 1 1 0
0 0 0 1 0 1 0 1 1 1 1 1 1 1 1 0 0 0 1 0 1 0 1 0 1 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 0 1 0 0 0 1 0
0 1 1 1 0 1 0 1 0 0 0 0 1 1 1 0 1 1 1 0 1 0 1 1 1 0 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 0 1 1 0 1 1 1 1 0 1 1 0 1 1 0 1 1 1 0
0 1 1 1 1 1 0 1 1 1 1 0 1 0 1 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 1 0 0 1 0 1 0 0 1 0 1 0 0 1 0 0 0
0 0 1 0 0 0 0 0 1 0 1 0 1 0 1 0 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 0 1 1 1 0 1 0 1 1 0 1 1 0 1 0 1 1 0 1 1 1 0 1 1 0 1 0
0 1 1 1 1 1 1 0 1 0 1 1 1 0 1 0 0 0 0 0 0 0 1 0 1 0 1 0 1 0 1 0 0 0 1 0 0 0 1 0 0 1 0 1 0 0 1 0 0 1 0 1 0 1 0 1 0 0 1 0
0 0 0 0 1 0 1 0 1 0 1 0 1 0 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 1 1 0 1 1 1 1 0 1 1 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 1 0
0 1 1 0 1 0 1 0 1 0 1 0 1 0 0 0 1 1 1 0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 1 0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 1 0 0 1 0
0 0 1 0 1 0 1 1 1 1 1 0 1 1 1 0 1 0 1 0 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 1 0
0 0 1 0 1 0 0 0 0 1 0 0 0 0 1 0 1 0 1 0 0 0 0 0 1 0 1 0 1 0 0 0 1 0 1 0 0 0 1 0 1 0 0 1 0 1 0 1 0 0 0 1 0 0 0 0 2 2 1 0
0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 2 1 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

-tiles Package
-It is a Tile png image



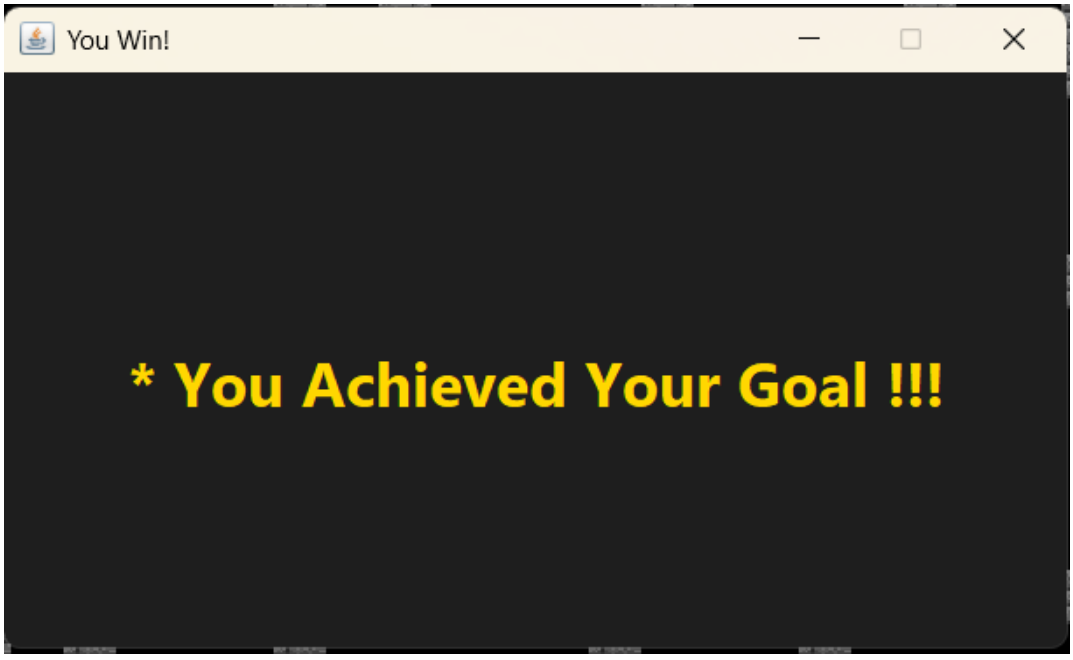
Application Output



- The application shows a simple maze game where the user must fill the missing part of the large white square using the **W**, **A**, **S**, and **D** keys to move.

When the user completes the task, the winning interface appears, and the program exits.

Winning Display



Conclusion

The program is a simple maze game created using the Java programming language and basic Swing GUI components. It is not a challenging game but represents a basic-level application. The goal is to find the correct path and connect the missing piece of the square