Project 3: Rat in a MazeProblem

Introduction:

The MazeSolver class is designed to solve a maze using multithreading in Java. It extends the Thread class and implements methods to navigate through the maze matrix concurrently.

Project Description:

Our project the maze solver , is basically 2 projects in one ! The user gets to enter an integer specifying a maze's dimensions ,then the program generates a random maze ,where the beginning is the top left square and the ending is the bottom left square . Then a thread starts traversing in Depth first search algorithm to get to the bottom ,while keeping in mind that it can only move forward or downward, and when both options are open ,the program creates a new thread to go forward while the main thread continues downward ,a thread is terminated when it hits a dead block!

What we have actually done:

We managed to make the solver perfectly ,considering our project's name ,with a tweak to our generator ,instead of letting the user make the maze himself ,we made the program generate the maze automatically! Everything else was done exactly the same! With different thread colors and everything!

Team members roles: We periodically made meetings to discuss and work together ,so we generally worked all together but if we have to separate ourselves to roles then:

GUI: Mirna,Nada

Solver: Ismail,Mervat,Roaa

Documentation: Ismail

Code documentation:

static int[][] matrice: Represents the maze matrix.

int xPos, yPos: Current position coordinates in the maze.

int n, m: Dimensions of the maze matrix.

String label: A label for identifying threads.

static HashMap<Long, Integer> idColors: Associates thread IDs with colors.

static Mazet tt: Instance of Mazet for visualization.

Constructor:

MazeSolver(int[][] matrice, int[] start, String label)

Parameters:

matrice: The maze matrix.

start: Starting position coordinates.

label: Identification label for the thread.

Purpose:

Initializes a MazeSolver object with the provided maze, start position, and label.

Methods:

public void run():

Purpose:

Executed when a thread starts.

Navigates through the maze, updates positions, and handles visualization.

Functionality:

Gets thread ID, updates maze matrix, checks for goal achievement, and visualizes the maze.

Thread Interaction:

Pauses for 100ms between movements.

Repaints the visualization (tt.repaint()).

public int move():

Purpose:

Controls the movement of the thread in the maze.

Functionality:

Checks neighboring positions and moves accordingly.

Creates new threads in case of a fork in the path.

Returns status codes for movement continuation or termination.

public ArrayList<int[]> checkNeigbours(int xPos, int yPos):

Parameters:

xPos, yPos: Current position coordinates.

Purpose:

Identifies accessible neighboring positions.

Functionality:

Checks right and down neighbors for accessibility.

Returns an ArrayList of accessible neighboring positions.

Summary: The MazeSolver class employs multithreading to explore a maze concurrently. It keeps track of each thread's progress, updates the maze matrix, and utilizes visualization components for representation.

Java Code Documentation

Mazet Class:

Introduction: The Mazet class extends JPanel and is designed to visually represent a matrix-based maze. It creates a graphical representation of the maze based on the given matrix values.

Attributes:

private static HashMap<Integer, Color> idColors: HashMap associating integer IDs with colors for maze visualization.

private int[][] matrice: Instance variable representing the maze matrix.

3. Constructors:

public Mazet():

Purpose:

Default constructor initializing the panel with a black background.

Functionality:

Sets the background color of the panel to black.

public Mazet(int[][] matrice):

Parameters:

matrice: The maze matrix to be visualized.

Purpose:

Constructor initializing the panel with the provided maze matrix and a black background

Functionality:

Sets the provided maze matrix.

Sets the background color of the panel to black

4. Methods:

public void setMatrice(int[][] matrice):

Parameters:

matrice: The maze matrix to be set.

Purpose:

Sets the maze matrix for visualization.

public void paintComponent(Graphics g):

Parameters:

g: Graphics object for drawing.

Purpose:

Overrides the paintComponent method to draw the maze based on the matrix values.

Functionality:

Checks if theJava Code Documentation: Mazet Class

Introduction: The Mazet class extends JPanel and is designed to visually represent a matrix-based maze. It creates a graphical representation of the maze based on the given matrix values.

Attributes:

private static HashMap<Integer, Color> idColors: HashMap associating integer IDs with colors for maze visualization.

private int[][] matrice: Instance variable representing the maze matrix.

3. Constructors:

public Mazet():

Purpose:

Default constructor initializing the panel with a black background.

Functionality:

Sets the background color of the panel to black.

public Mazet(int[][] matrice):

Parameters:

matrice: The maze matrix to be visualized

.

Purpose:

Constructor initializing the panel with the provided maze matrix and a black background.

Functionality:

Sets the provided maze matrix.

Sets the background color of the panel to black.

4. Methods:

public void setMatrice(int[][] matrice):

Parameters:

matrice: The maze matrix to be set.

Purpose:

Sets the maze matrix for visualization.

public void paintComponent(Graphics g):

Parameters:

g: Graphics object for drawing.

Purpose:

Overrides the paintComponent method to draw the maze based on the matrix values. Java Code Documentation: Mazet Class

1. Introduction:

The Mazet class extends JPanel and is designed to visually represent a matrix-based maze. It creates a graphical representation of the maze based on the given matrix values.

2. Attributes:

private static HashMap<Integer, Color> idColors: HashMap associating integer IDs with colors for maze visualization.

private int[][] matrice: Instance variable representing the maze matrix.

3. Constructors:

public Mazet():

Purpose:

Default constructor initializing the panel with a black background.

Functionality:

Sets the background color of the panel to black.

public Mazet(int[][] matrice):

Parameters:

matrice: The maze matrix to be visualized.

Purpose:

Constructor initializing the panel with the provided maze matrix and a black background.

Functionality:

Sets the provided maze matrix.

Sets the background color of the panel to black.

4. Methods:

public void setMatrice(int[][] matrice):

Parameters:

matrice: The maze matrix to be set.

Purpose:

Sets the maze matrix for visualization.

public void paintComponent(Graphics g):

Parameters:

g: Graphics object for drawing.

Purpose:

Overrides the paintComponent method to draw the maze based on the matrix values.

Functionality:

Checks if the matrix is not null.

Iterates through the matrix and draws squares (representing cells) based on their values:

0: Black squares representing walls.

1: White squares representing pathways.

-22: Gray squares representing failed paths.

2: Yellow square representing the exit.

Other values: Colored squares based on a hue-saturation-brightness model.

Colors are assigned based on the modulo of the color ID.

5. Summary:

The Mazet class serves as a panel for visualizing the maze matrix. It customizes the appearance of each cell based on its value, providing a visual representation of the maze's structure and navigation.

java Code Documentation main Class:

1. Introduction:

The main class serves as the entry point for the maze solver application. It generates a graphical user interface (GUI) allowing users to create a random maze, visualize the maze solving process, and exit the application.

2. Attributes:

public static int[] exit: Array representing the exit coordinates.

public static boolean goalAchieved: Flag indicating whether the maze goal is achieved.

int[][] matrice: Instance variable representing the maze matrix.

private JFrame frame: Instance of JFrame for the application window.

3. Constructors:

public main(int[][] matrice):

Parameters:

matrice: The maze matrix used in initialization.

Purpose:

Constructor initializing the maze matrix.

Functionality:

Sets the provided maze matrix.

public static void setStatic():

Purpose:

Sets the goalAchieved flag to true when the maze goal is achieved.

4. Methods:

public void createAndShowGUI():

Purpose:

Creates and displays the GUI components.

Functionality:

Initializes the application window (JFrame).

Adds components (labels, text fields, buttons) to the panel.

Handles button actions for generating the maze and exiting the application.

private void placeComponents(JPanel panel):

Parameters:

panel: Panel where components are placed.

Purpose:

Places GUI components (labels, text fields, buttons) on the panel.

Functionality:

Sets layout to null for precise component placement.

Creates and positions labels, text fields, and buttons within the panel.

Defines button action listeners for maze generation and application exit.

public static void main(String[] args):

Purpose:

Entry point of the application.

Functionality:

Creates an instance of the main class and invokes the GUI creation within the event dispatch thread.

private static int[][] createRandomMaze(int size):

Parameters:

size: Size of the maze (N x N).

Purpose:

Generates a random maze matrix.

Functionality:

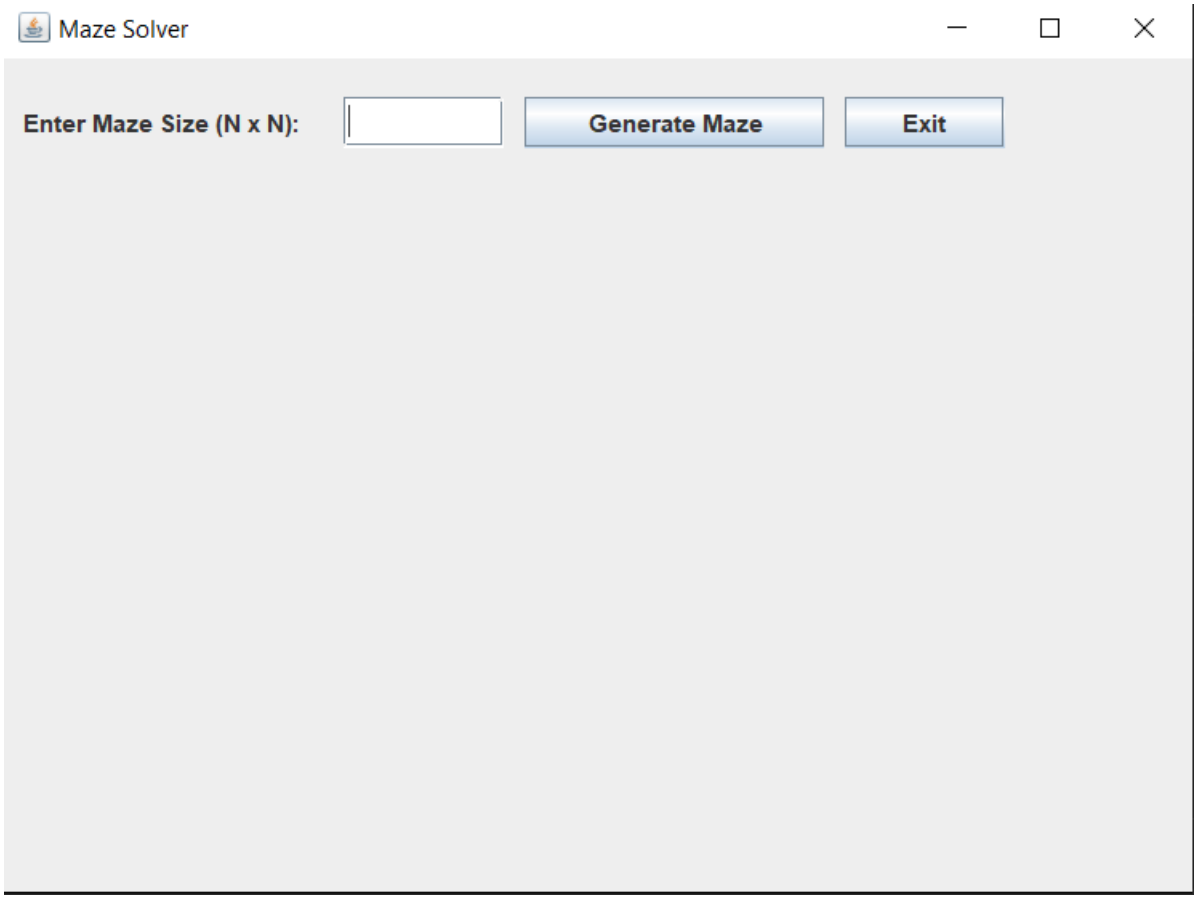
Initializes a maze with random values (70% cells, 30% walls).

Marks the exit block with a value of 2 within the maze.

5. Summary: The main class initializes the application, provides a GUI for maze generation, visualizes the maze-solving process, and enables users to exit the application.

Graphical user interface:

Input:



The user can enter any value for the maze size(N*N)

Output:

When the maze input=8*8