Assignment 4 Lab 8

Author: Aashay Pawar
NUID: 002134382

Problem 1
Write a program that randomly fills in 0-9 into an n × m matrix, prints the matrix, and finds the rows and columns with the largest value. You can use different approaches to solve this problem. For example, you can use two-dimensional arrays to store randomly generated data and ArrayList for saving the indices.

Note that we provide "lab8_reference.pdf" document to review two-dimensional array. It is also on the Canvas. If you are not familiar with this topic, please study this document as well.

Expected results:

```
Enter the number of rows: 4
Enter the number of columns: 6

The array content is:
7 9 6 0 4 3
0 4 8 4 8 2
2 1 1 4 5 2
7 3 5 9 6 0

The index of the largest row: 3
The index of the largest column: 4
```

Problem Description: A program that generates an n × m matrix filled with random integers between 0 and 9. After generating the matrix, the program must print it and then find and display the indices of the row and column with the largest sum of elements.

Analysis:
1. User Input: The program will prompt the user to enter the number of rows (n) and columns (m) for the matrix.
2. Matrix Generation: Using the provided dimensions, a random matrix will be generated with integers between 0 and 9, stored in a 2D array using the `funGenerateRandomMatrix()` function.
3. Matrix Printing: The generated matrix will be printed using the `funPrintMatrix()` function.
4. Row and Column Sum Calculation: The program will calculate the sum of elements for each row and column using the `funSumRow()` and `funSumCol()` functions, respectively.
5. Largest Row and Column Identification: The program will identify the row and column with the largest sum of elements using the `funFindLargestRow()` and `funFindLargestCol()` functions, respectively.
6. Display Results: The program will display the index of the row with the largest sum and the index of the column with the largest sum.

The program is implemented using separate functions for each step, promoting modular and easy-to-understand code. It uses the `Random` class to generate random integers for the matrix elements and employs nested loops for matrix generation and sum calculation.

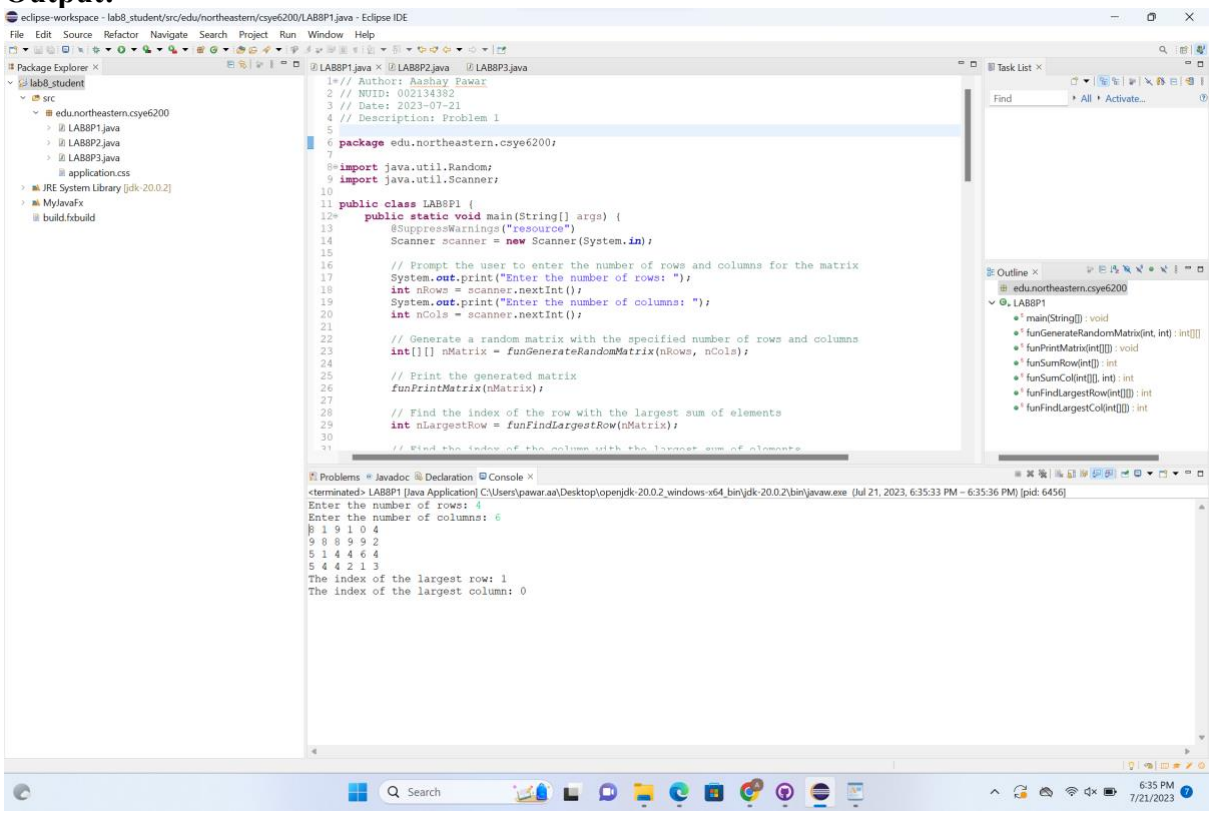
Code:

```
// Author: Aashay Pawar
// NUID: 002134382
// Date: 2023-07-21
// Description: Problem 1

package edu.northeastern.csye6200;

import java.util.Random;
import java.util.Scanner;

public class LAB8P1 {
    public static void main(String[] args) {
        @SuppressWarnings("resource")
        Scanner scanner = new Scanner(System.in);

        // Prompt the user to enter the number of rows and columns for the matrix
        System.out.print("Enter the number of rows: ");
        int nRows = scanner.nextInt();
        System.out.print("Enter the number of columns: ");
        int nCols = scanner.nextInt();

        // Generate a random matrix with the specified number of rows and columns
        int[][] nMatrix = funGenerateRandomMatrix(nRows, nCols);

        // Print the generated matrix
        funPrintMatrix(nMatrix);

        // Find the index of the row with the largest sum of elements
        int nLargestRow = funFindLargestRow(nMatrix);

        // Find the index of the column with the largest sum of elements
        int nLargestCol = funFindLargestCol(nMatrix);

        // Print the index of the row and column with the largest sums
        System.out.println("The index of the largest row: " + nLargestRow);
        System.out.println("The index of the largest column: " + nLargestCol);
    }

    // Generate a random matrix with the specified number of rows and columns
    public static int[][] funGenerateRandomMatrix(int rows, int cols) {
        int[][] nMatrix = new int[rows][cols];
        Random nRandom = new Random();

        // Fill the matrix with random numbers between 0 and 9
        for (int i = 0; i < rows; i++) {
```

```
            for (int j = 0; j < cols; j++) {
                nMatrix[i][j] = nRandom.nextInt(10); // Random number between 0 and 9
            }
        }

        return nMatrix;
    }

    // Print the given matrix
    public static void funPrintMatrix(int[][] matrix) {
        for (int[] row : matrix) {
            for (int n : row) {
                System.out.print(n + " ");
            }
            System.out.println();
        }
    }

    // Calculate the sum of elements in a given row
    public static int funSumRow(int row[]) {
        int s = 0;
        for (int n : row) {
            s += n;
        }
        return s;
    }

    // Calculate the sum of elements in a given column of the matrix
    public static int funSumCol(int matrix[][], int col) {
        int s = 0;
        for (int i = 0; i < matrix.length; i++) {
            s += matrix[i][col];
        }
        return s;
    }

    // Find the index of the row with the largest sum of elements
    public static int funFindLargestRow(int[][] matrix) {
        int maxS = Integer.MIN_VALUE;
        int maxR = 0;

        // Iterate through each row and calculate its sum, keeping track of the row index with the
largest sum
        for (int i = 0; i < matrix.length; i++) {
            int rowSum = funSumRow(matrix[i]);
            if (rowSum > maxS) {
                maxS = rowSum;
                maxR = i;
            }
        }
```

```
        return maxR;
    }

    // Find the index of the column with the largest sum of elements
    public static int funFindLargestCol(int[][] matrix) {
        int maxS = Integer.MIN_VALUE;
        int maxC = 0;

        // Iterate through each column and calculate its sum, keeping track of the column index
with the largest sum
        for (int i = 0; i < matrix[0].length; i++) {
            int columnSum = funSumCol(matrix, i);
            if (columnSum > maxS) {
                maxS = columnSum;
                maxC = i;
            }
        }

        return maxC;
    }
}
```

Output:



Problem 2
Write a program that uses a bar chart to display the percentages of the overall grade represented by project, exams, assignments, and the attendance, as shown in Figure 1. Suppose that project takes 35 percent and is displayed in blue, exams take 30 percent and are displayed in green, assignments take 30 percent and are displayed in red, and the attendance takes 5 percent and is displayed in orange. Please use the JavaFX Rectangle class to display the bars.

Problem Description: A Java program that displays a bar chart representing the percentages of the overall grade contributed by four components: project, exams, assignments, and attendance. The chart will use different colors to represent each component and display their respective percentages.

The grading breakdown is as follows:
- Project: 35% (Displayed in blue)
- Exams: 30% (Displayed in green)
- Assignments: 30% (Displayed in red)
- Attendance: 5% (Displayed in orange)

The program will use JavaFX, specifically the `Rectangle` class, to create the visual representation of the bar chart.

Analysis:
1. JavaFX Application: The program utilizes JavaFX to create a graphical user interface for displaying the bar chart.
2. Pane and Scene: The application will create a `Pane` to hold the visual elements, and a `Scene` will be set with the specified dimensions to contain the `Pane`.
3. Bar Chart Design: The bar chart will be designed with four rectangles, one for each component (project, exams, assignments, and attendance). The rectangles will be positioned accordingly to represent their respective percentages within the chart.
4. Colors and Labels: The rectangles representing each component will be filled with the designated colors (blue, green, red, and orange). Text labels will be added to each rectangle to display the name of the component and its corresponding percentage.
5. Window Display: The chart will be displayed in a separate window (stage) with a specific width and height, defined by the `Scene`.

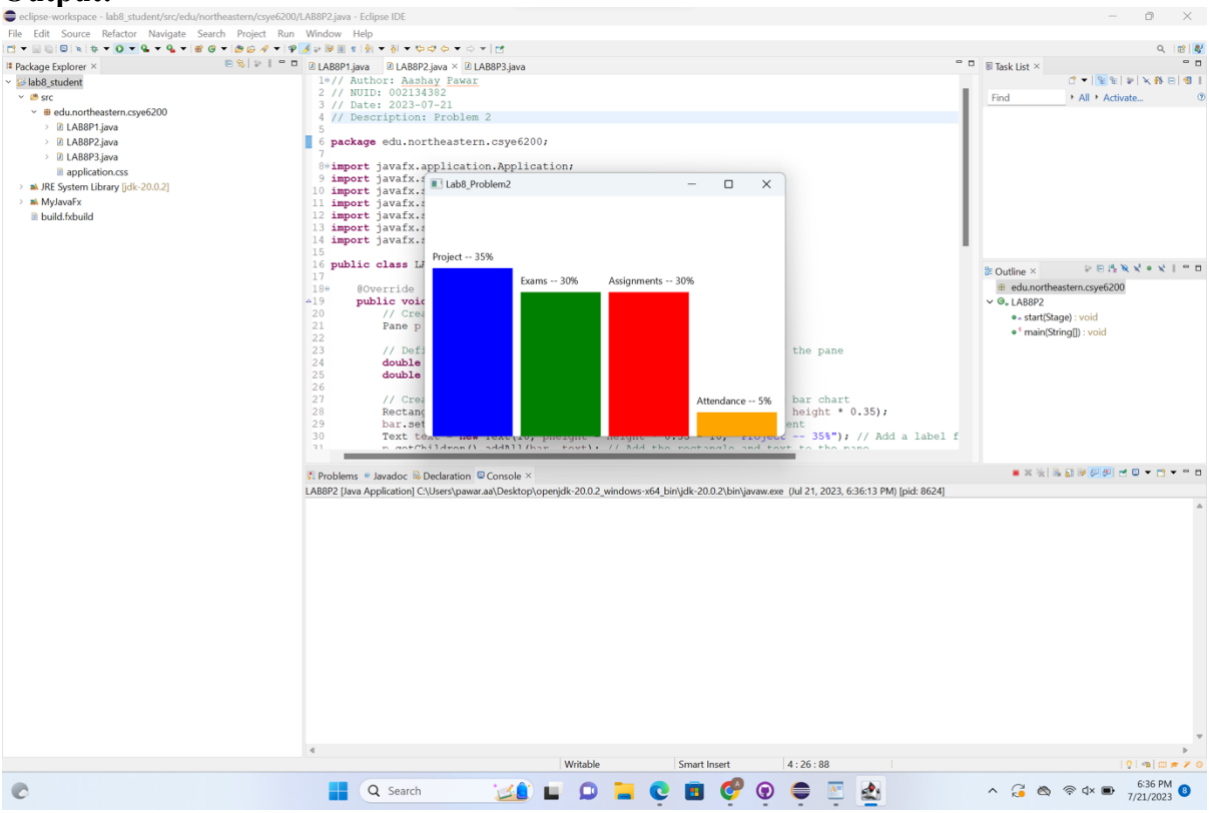
Solution:

```
// Author: Aashay Pawar
// NUID: 002134382
// Date: 2023-07-21
// Description: Problem 2

package edu.northeastern.csye6200;

import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.scene.shape.Rectangle;
import javafx.scene.text.Text;
```

```
import javafx.stage.Stage;

public class LAB8P2 extends Application {

	@Override
	public void start(Stage primaryStage) throws Exception {
		// Create a new Pane to hold the visual elements
		Pane p = new Pane();

		// Define the overall height of the bar chart and the height of the pane
		double height = 600;
		double pHeight = 300;

		// Create a rectangle representing the project component of the bar chart
		Rectangle bar = new Rectangle(10, pHeight - height * 0.35, 100, height *
0.35);
		bar.setFill(Color.BLUE); // Set the color of the project component
		Text text = new Text(10, pHeight - height * 0.35 - 10, "Project -- 35%"); //
Add a label for the project component
		p.getChildren().addAll(bar, text); // Add the rectangle and text to the pane

		// Create a rectangle representing the exams component of the bar chart
		bar = new Rectangle(10 + 110, pHeight - height * 0.3, 100, height * 0.3);
		bar.setFill(Color.GREEN); // Set the color of the exams component
		text = new Text(10 + 110, pHeight - height * 0.3 - 10, "Exams -- 30%"); // Add
a label for the exams component
		p.getChildren().addAll(bar, text); // Add the rectangle and text to the pane

		// Create a rectangle representing the assignments component of the bar chart
		bar = new Rectangle(10 + 220, pHeight - height * 0.3, 100, height * 0.3);
		bar.setFill(Color.RED); // Set the color of the assignments component
		text = new Text(10 + 220, pHeight - height * 0.3 - 10, "Assignments -- 30%");
// Add a label for the assignments component
		p.getChildren().addAll(bar, text); // Add the rectangle and text to the pane

		// Create a rectangle representing the attendance component of the bar chart
		bar = new Rectangle(10 + 330, pHeight - height * 0.05, 100, height * 0.05);
		bar.setFill(Color.ORANGE); // Set the color of the attendance component
		text = new Text(10 + 330, pHeight - height * 0.05 - 10, "Attendance -- 5%"); //
Add a label for the attendance component
		p.getChildren().addAll(bar, text); // Add the rectangle and text to the pane

		// Create a scene with the pane and set its dimensions
		Scene scene = new Scene(p, 450, pHeight);

		// Set the title of the stage (window)
		primaryStage.setTitle("Lab8_Problem2");

		// Set the scene for the stage and display it
		primaryStage.setScene(scene);
```

```
                primaryStage.show();
        }

        public static void main(String[] args) {
                // Launch the JavaFX application
                launch(args);
        }
}
```

Output:



Problem 3
Write a program that displays a 10-by-10 square matrix, as shown in Figure 2. Each element in the matrix is either 0 or 1, randomly generated. Display each number centered in a text field. Use TextField's setText method to set value 0 or 1 as a string

Problem Description:
You are required to create a Java program that displays a 10-by-10 square matrix of randomly generated 0s and 1s. Each element in the matrix will be represented by a centered `TextField` displaying the value as a string.

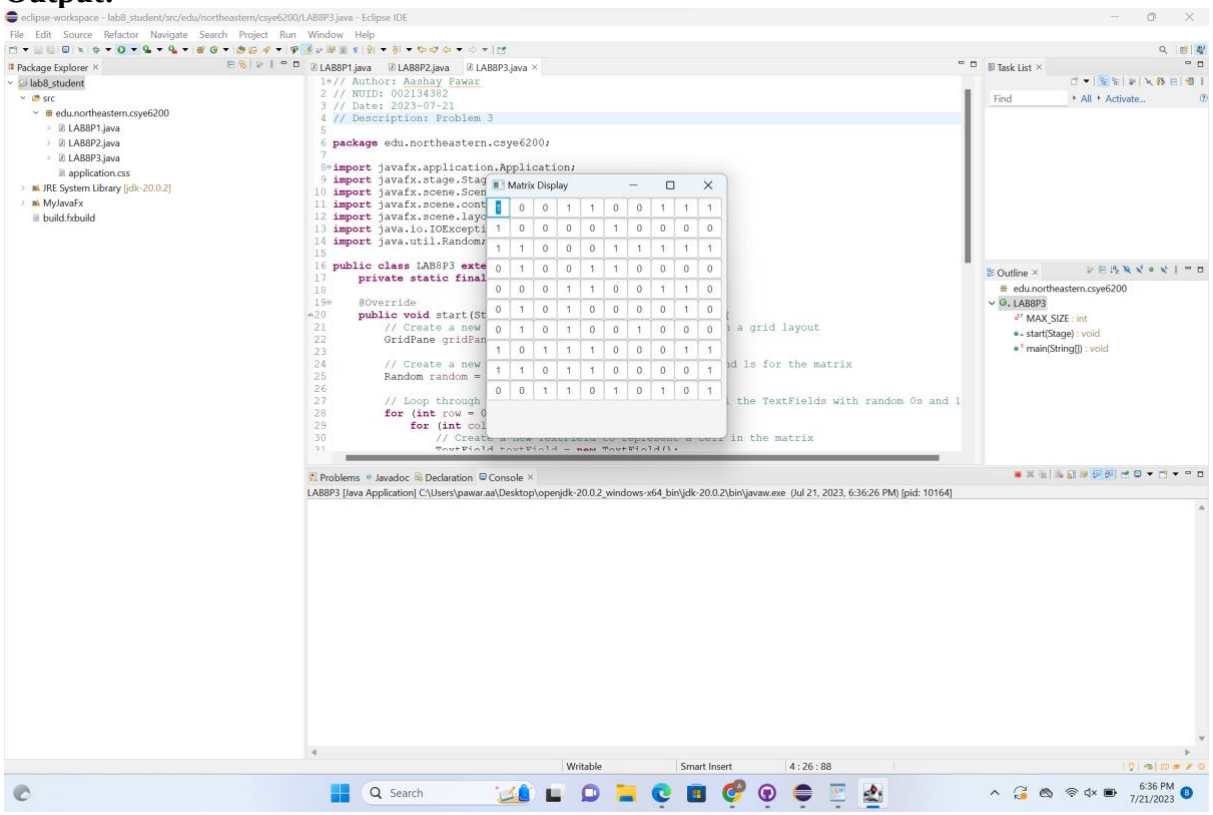
Analysis:
1. JavaFX Application: The program utilizes JavaFX to create a graphical user interface for displaying the matrix.
2. GridPane: The `GridPane` layout is used to organize the `TextFields` in a grid format, representing the 10-by-10 square matrix.
3. Random Matrix Generation: A nested loop iterates through each row and column of the matrix, generating random 0s and 1s using `Random.nextInt(2)`. Each value is then set as the text in the corresponding `TextField`.
4. Centered Text: The text in each `TextField` is centered using `textField.setAlignment(javafx.geometry.Pos.CENTER);`.
5. Scene and Stage: A `Scene` is created with the `GridPane`, and its dimensions are set to 300x300. The matrix is displayed in a separate window (stage) titled "Matrix Display".

Solution:

```
// Author: Aashay Pawar
// NUID: 002134382
// Date: 2023-07-21
// Description: Problem 3

package edu.northeastern.csye6200;

import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.control.TextField;
import javafx.scene.layout.GridPane;
import java.io.IOException;
import java.util.Random;

public class LAB8P3 extends Application {
	private static final int MAX_SIZE = 10;

	@Override
	public void start(Stage primaryStage) throws IOException {
		// Create a new GridPane to organize the TextFields in a grid layout
		GridPane gridPane = new GridPane();

		// Create a new Random object to generate random 0s and 1s for the matrix
		Random random = new Random();
```

```
		// Loop through each row and column to create and fill the TextFields with random 0s and 1s
		for (int row = 0; row < MAX_SIZE; row++) {
			for (int col = 0; col < MAX_SIZE; col++) {
				// Create a new TextField to represent a cell in the matrix
				TextField textField = new TextField();

				// Generate a random 0 or 1 and set it as the text in the TextField
				int val = random.nextInt(2);
				textField.setText(Integer.toString(val));

				// Center-align the text in the TextField
				textField.setAlignment(javafx.geometry.Pos.CENTER);

				// Add the TextField to the GridPane at the specified row and column
				gridPane.add(textField, col, row);
			}
		}

		// Create a new Scene with the GridPane and set its dimensions
		Scene scene = new Scene(gridPane, 300, 300);

		// Set the title of the stage (window)
		primaryStage.setTitle("Matrix Display");

		// Set the Scene for the stage and display it
		primaryStage.setScene(scene);
		primaryStage.show();
	}

	public static void main(String[] args) {
		// Launch the JavaFX application
		launch(args);
	}
}
```

Output:


Matrix Display
1 0 0 1 1 0 0 1 1 1
1 0 0 0 0 1 0 0 0 0
1 1 0 0 0 1 1 1 1 1
0 1 0 0 1 1 0 0 0 0
0 0 0 1 1 0 0 1 1 0
0 1 0 1 0 0 0 0 1 0
0 1 0 1 0 0 1 0 0 0
1 0 1 1 1 0 0 0 1 1
1 1 0 1 1 0 0 0 0 1
0 0 1 1 0 1 0 1 0 1