

**Assignment 02**

Submitted to:

**Ahmed Abdal Shafi Rasel**

**Lecturer, Department of CSE**

Submitted by:

**MD Asaduzzaman Atik**

ID: **2023-1-60-130**

Course: **CSE110**

Section: **16**

Date: **March 08, 2024**

Table of Contents

[Assignment Overview 2](#_Toc160813824)

[Tasks Checklist 2](#_Toc160813825)

[Menu-Driven Approach 2](#_Toc160813826)

[Project Website 2](#_Toc160813827)

[Accessible Problems 3](#_Toc160813828)

[Author 3](#_Toc160813829)

[Instructor 3](#_Toc160813830)

[Tools and Technologies 3](#_Toc160813831)

[Disclaimer 3](#_Toc160813832)

[Task Solutions 4](#_Toc160813833)

[Entry program 4](#_Toc160813834)

[// App.java 4](#_Toc160813835)

[Task 01: **Palindrome Integer** 7](#_Toc160813836)

[// T01\_Palindrome.java 7](#_Toc160813837)

[Task 02: **Display Matrix of 0s and 1s** 9](#_Toc160813838)

[// T02\_Matrix.java 9](#_Toc160813839)

[Task 03: Check Password 10](#_Toc160813840)

[// T03\_Password.java 10](#_Toc160813841)

[Task 04: Count the Letters in a String 12](#_Toc160813842)

[// T04\_CountLetters.java 12](#_Toc160813843)

[Task 05: Occurrences of a Specified Character 13](#_Toc160813844)

[// T05\_CountChar.java 13](#_Toc160813845)

[Task 06: Stock Class 15](#_Toc160813846)

[// T06\_Stock.java 15](#_Toc160813847)

[Task 07: Use the GregorianCalendar Class 18](#_Toc160813848)

[// T07\_CalendarTest.java 18](#_Toc160813849)

[Task 08: **Stopwatch** 20](#_Toc160813850)

[// T08\_StopWatch.java 20](#_Toc160813851)

[Task 09: Algebra: 2 \* 2 Linear Equations 24](#_Toc160813852)

[// T09\_LinearEquation.java 24](#_Toc160813853)

[Task 10: The Location Class 28](#_Toc160813854)

[// T10\_Location.java 28](#_Toc160813855)

# Assignment Overview

The objective of this assignment is to enhance problem-solving skills related to object-oriented programming concepts. The tasks involve creating methods, method overloading, class and object creation, constructor usage, access modifiers, instance and static methods, and more.

## Tasks Checklist

* **✅ 1.** [**Palindrome Integer**](#_Task_01:_Palindrome)
* **✅ 2.** [**Display Matrix of 0s and 1s**](#_Task_02:_Display)
* **✅ 3.** [**Check Password**](#_Task_03:_Check)
* **✅ 4.** [**Count the Letters in a String**](#_Task_04:_Count)
* **✅ 5.** [**Occurrences of a Specified Character**](#_Task_05:_Occurrences)
* **✅ 6.** [**Stock Class**](#_Task_06:_Stock)
* **✅ 7.** [**Use the GregorianCalendar**](#_Task_07:_Use)
* **✅ 8.** [**Stopwatch**](#_Task_08:_Stopwatch)
* **✅ 9.** [**Algebra: 2 \* 2 Linear Equations**](#_Task_09:_Algebra:)
* **✅ 10.** [**The Location Class**](#_Task_10:_The)

## Menu-Driven Approach

This project utilizes a menu-driven approach for user interaction and task execution. The [**App class's main method**](#_//_App.java) serves as the **entry point** of the program and implements the following functionalities:

1. **Display a menu** listing all available tasks.
2. **Allow users to select a task** by entering the corresponding number.
3. **Execute the chosen task** based on the user's selection.
4. **Provide clear instructions and prompts** within each task.

This approach enhances the user experience and simplifies the interaction with the various functionalities implemented in the assignment.

## Project Website

This project's documentation and resources are readily available within [this GitHub repository](https://github.com/mrasadatik/academic.cse110.assignment02). Explore the various sections to access detailed information and solutions.

## Accessible Problems

Detailed descriptions of the problems are available in the **"**[**assignment02-problems**](https://github.com/mrasadatik/academic.cse110.assignment02/blob/main/app/src/main/resources/assignment02-problems.pdf)**"** PDF.

## Author

* [Md Asaduzzaman Atik](https://www.github.com/mrasadatik) (Code)

## Instructor

* [Ahmed Abdal Shafi Rasel](http://fse.ewubd.edu/computer-science-engineering/faculty-view/ahmed.shafi), Lecturer, Department of CSE, [East West University](https://www.ewubd.edu)

## Tools and Technologies

* **IDE:** [Apcahe NetBeans IDE 21](https://netbeans.apache.org/front/main/index.html)
* **Build Tool:** [Gradle](https://gradle.org/)@8.6
* **Programming Language:** [Java](https://www.oracle.com/java/)@21
* **Documentation:** [Gemini](https://gemini.google.com), [ChatGPT](https://chat.openai.com)

## Disclaimer

**Please note:** This project currently does not have any automated or unit test implementation. While the program functions as intended, future development efforts might include the addition of testing frameworks for improved code reliability and maintainability.

# Task Solutions

## Entry program

### // App.java

**package** academic.cse110.assignment02;

**import** **java.util.Scanner**;

**import** **academic.cse110.assignment02.tasks.\***;

**public** **class** App {

**public** **static** **void** main(String[] args) {

**try** (Scanner cliInput = **new** Scanner(System.in)) {

**int** taskChoice;

System.out.println();

System.out.println("Assignment\t: 02");

System.out.println("\tSubmitted to\t: Ahmed Abdal Shafi Rasel (AASR), Lecturer, Department of CSE");

System.out.println();

System.out.println("Name\t\t: Md Asaduzzaman Atik");

System.out.println("Student ID\t: 2023-1-60-130");

System.out.println("Couse title\t: Object Oriented Programming");

System.out.println("Couse code\t: CSE110");

System.out.println("Section\t\t: 16");

System.out.println("Semester\t: Spring 2024");

**do** {

System.out.println();

System.out.println();

System.out.println("Choose a task by entering the corresponding number:");

System.out.println("\t1. Palindrome Integer");

System.out.println("\t2. Display Matrix of 0s and 1s");

System.out.println("\t3. Check Password");

System.out.println("\t4. Count the Letters in a String");

System.out.println("\t5. Occurrences of a Specified Character");

System.out.println("\t6. Stock Class");

System.out.println("\t7. Use the GregorianCalendar Class");

System.out.println("\t8. Stopwatch");

System.out.println("\t9. Algebra: 2 \* 2 Linear Equations");

System.out.println("\t10. The Location Class");

System.out.println();

System.out.println("\t\t0. Exit");

System.out.print("\nEnter your choice: ");

taskChoice = cliInput.nextInt();

System.out.println();

System.out.println();

**switch**(taskChoice) {

**case** 1 -> T01\_Palindrome.runPalindrome(cliInput);

**case** 2 -> T02\_Matrix.runPrintMatrix(cliInput);

**case** 3 -> T03\_Password.passwordValidator(cliInput);

**case** 4 -> T04\_CountLetters.runCountLetters(cliInput);

**case** 5 -> T05\_CountChar.runCountOccurrence(cliInput);

**case** 6 -> T06\_Stock.displayStockDetails();

**case** 7 -> T07\_CalendarTest.displayDate();

**case** 8 -> T08\_StopWatch.getExecutionTime();

**case** 9 -> T09\_LinearEquation.runLinearEquationCalculator(cliInput);

**case** 10 -> T10\_Location.runMaximalLocator(cliInput);

**case** 0 -> {

System.out.println("Exiting the program...");

**break**;

}

**default** -> System.out.println("Invalid choice. Please try again.");

}

} **while** (taskChoice != 0);

}

}

}

## Task 01: **Palindrome Integer**

Write the methods with the following headers:

**public** **static** **int** reverse(**int** number)

**public** **static** **boolean** isPalindrome(**int** number)

Use the reverse method to implement isPalindrome. A number is a palindrome if its reversal

is the same as itself. Write a test program that prompts the user to enter an integer and reports

whether the integer is a palindrome.

### // T01\_Palindrome.java

**package** academic.cse110.assignment02.tasks;

**import** **java.util.Scanner**;

*/\*\**

*\**

*\* @author mrasadatik*

*\*/*

**public** **class** T01\_Palindrome {

**public** **static** **int** reverse(**int** number) {

**int** reversed = 0;

**while** (number != 0) {

**int** digit = number % 10;

reversed = reversed \* 10 + digit;

number /= 10;

}

**return** reversed;

}

**public** **static** **boolean** isPalindrome(**int** number) {

**return** number == reverse(number);

}

**public** **static** **void** runPalindrome(Scanner scanner) {

System.out.print("Enter an integer: ");

**int** num = scanner.nextInt();

**if** (isPalindrome(num)) {

System.out.println(num + " is a palindrome.");

} **else** {

System.out.println(num + " is not a palindrome.");

}

}

}

## Task 02: **Display Matrix of 0s and 1s**

Write a method that displays an n-by-n matrix using the following header:

**public** **static** **void** printMatrix(**int** n)

Each element is 0 or 1, which is generated randomly. Write a test program that prompts the user to enter n and displays an n-by-n matrix.

### // T02\_Matrix.java

**package** academic.cse110.assignment02.tasks;

**import** **java.util.Scanner**;

*/\*\**

*\**

*\* @author mrasadatik*

*\*/*

**public** **class** T02\_Matrix {

**public** **static** **void** printMatrix(**int** n) {

**for** (**int** i = 0; i < n; i++) {

**for** (**int** j = 0; j < n; j++) {

System.out.print((**int**) (Math.random() \* 2) + "\t");

}

System.out.println();

}

}

**public** **static** **void** runPrintMatrix(Scanner scanner) {

System.out.print("Enter n: ");

**int** n = scanner.nextInt();

printMatrix(n);

}

}

## Task 03: Check Password

Some websites impose certain rules for passwords. Write a method that checks whether a string is a valid password. Suppose the password rules are as follows:

* A password must have at least eight characters.
* A password consists of only letters and digits.
* A password must contain at least two digits.

Write a program that prompts the user to enter a password and displays Valid Password if the rules are followed or Invalid Password otherwise.

### // T03\_Password.java

**package** academic.cse110.assignment02.tasks;

**import** **java.util.Scanner**;

*/\*\**

*\**

*\* @author mrasadatik*

*\*/*

**public** **class** T03\_Password {

**public** **static** **void** passwordValidator(Scanner scanner) {

scanner.nextLine();

**boolean** lengthRequirementFulfilled = **true**;

**boolean** lettersAndDigitsRequirementFulfilled = **true**;

System.out.print("Enter password: ");

String password = scanner.nextLine();

**if** (password.length() < 8) {

lengthRequirementFulfilled = **false**;

}

**int** digitCount = 0;

**for** (**char** c : password.toCharArray()) {

**if** (!Character.isLetterOrDigit(c)) {

lettersAndDigitsRequirementFulfilled = **false**;

}

**if** (Character.isDigit(c)) {

digitCount++;

}

}

**if** (lengthRequirementFulfilled && lettersAndDigitsRequirementFulfilled && (digitCount >= 2)) {

System.out.println("Valid Password");

} **else** {

System.out.println("Invalid Password");

}

}

}

## Task 04: Count the Letters in a String

Write a method that counts the number of letters in a string using the following header:

**public** **static** **int** countLetters(String s)

Write a test program that prompts the user to enter a string and displays the number of letters in the string.

### // T04\_CountLetters.java

**package** academic.cse110.assignment02.tasks;

**import** **java.util.Scanner**;

*/\*\**

*\**

*\* @author mrasadatik*

*\*/*

**public** **class** T04\_CountLetters {

**public** **static** **int** countLetters(String s) {

**int** count = 0;

**for** (**char** c : s.toCharArray()) {

**if** (Character.isLetter(c)) {

count++;

}

}

**return** count;

}

**public** **static** **void** runCountLetters(Scanner scanner) {

scanner.nextLine();

System.out.print("Enter a string: ");

String s = scanner.nextLine();

**int** letterCount = countLetters(s);

System.out.println("The number of letters in the string is: " + letterCount);

}

}

## Task 05: Occurrences of a Specified Character

Write a method that finds the number of occurrences of a specified character in a string using the following header:

**public** **static** **int** count(String str, **char** a)

For example, count("Welcome", 'e') returns 2. Write a test program that prompts the user to enter a string followed by a character and displays the number of occurrences of the character in the string.

### // T05\_CountChar.java

**package** academic.cse110.assignment02.tasks;

**import** **java.util.Scanner**;

*/\*\**

*\**

*\* @author mrasadatik*

*\*/*

**public** **class** T05\_CountChar {

**public** **static** **int** count(String str, **char** a) {

**int** count = 0;

**for** (**char** ch : str.toCharArray()) {

**if** (ch == a) {

count++;

}

}

**return** count;

}

**public** **static** **void** runCountOccurrence(Scanner scanner) {

scanner.nextLine();

System.out.print("Enter a string: ");

String str = scanner.nextLine();

System.out.print("Enter a character to count: ");

**char** a = scanner.next().charAt(0);

**int** charCount = count(str, a);

System.out.println("The character '" + a + "' occurs " + charCount + " times in the string.");

}

}

## Task 06: Stock Class

Design a class named Stock that contains:

* A string data field named symbol for the stock’s symbol.
* A string data field named name for the stock’s name.
* A double data field named previousClosingPrice that stores the stock price for the previous day.
* A double data field named currentPrice that stores the stock price for the current time.
* A constructor that creates a stock with the specified symbol and name.
* A method named getChangePercent() that returns the percentage changed from previousClosingPrice to currentPrice.

Draw the UML diagram for the class and then implement the class. Write a test program that creates a Stock object with the stock symbol ORCL, the name Oracle Corporation, and the previous closing price of 34.5. Set a new current price to 34.35 and display the price-change percentage.

### // T06\_Stock.java

*/\**

*UML (Unified Modeling Language) Diagram for this program:*

*+----------------------------------------+*

*| Stock |*

*+----------------------------------------+*

*| - symbol: String |*

*| - name: String |*

*| - previousClosingPrice: double |*

*| - currentPrice: double |*

*+----------------------------------------+*

*| + Stock(symbol: String, name: String) |*

*| + getPreviousClosingPrice(): double |*

*| + setPreviousClosingPrice(double) |*

*| + getCurrentPrice(): double |*

*| + setCurrentPrice(double) |*

*| + getSymbol(): String |*

*| + getName(): String |*

*| + getChangePercent(): double |*

*+----------------------------------------+*

*+----------------------------------------+*

*| T06\_Stock |*

*+----------------------------------------+*

*| + runStockDetails() |*

*+----------------------------------------+*

*\*/*

**package** academic.cse110.assignment02.tasks;

**class** Stock {

@SuppressWarnings("FieldMayBeFinal")

**private** String symbol;

@SuppressWarnings("FieldMayBeFinal")

**private** String name;

**private** **double** previousClosingPrice;

**private** **double** currentPrice;

**public** Stock(String symbol, String name) {

**this**.symbol = symbol;

**this**.name = name;

}

**public** **double** getPreviousClosingPrice() {

**return** previousClosingPrice;

}

**public** **void** setPreviousClosingPrice(**double** previousClosingPrice) {

**this**.previousClosingPrice = previousClosingPrice;

}

**public** **double** getCurrentPrice() {

**return** currentPrice;

}

**public** **void** setCurrentPrice(**double** currentPrice) {

**this**.currentPrice = currentPrice;

}

**public** String getSymbol() {

**return** symbol;

}

**public** String getName() {

**return** name;

}

**public** **double** getChangePercent() {

**return** ((currentPrice - previousClosingPrice) / previousClosingPrice) \* 100;

}

}

*/\*\**

*\**

*\* @author mrasadatik*

*\*/*

**public** **class** T06\_Stock {

**public** **static** **void** displayStockDetails() {

Stock oracleStock = **new** Stock("ORCL", "Oracle Corporation");

oracleStock.setPreviousClosingPrice(34.5);

oracleStock.setCurrentPrice(34.35);

System.out.println("Stock Symbol: " + oracleStock.getSymbol());

System.out.println("Stock Name: " + oracleStock.getName());

System.out.println("Previous Closing Price: $" + oracleStock.getPreviousClosingPrice());

System.out.println("Current Price: $" + oracleStock.getCurrentPrice());

System.out.println("Price Change Percentage: " + oracleStock.getChangePercent() + "%");

}

}

## Task 07: Use the GregorianCalendar Class

Java API has the GregorianCalendar class in the **java.util** package, which you can use to obtain the year, month, and day of a date. The no-arg constructor constructs an instance for the current date, and the methods get(GregorianCalendar.YEAR), get(GregorianCalendar.MONTH) and get(GregorianCalendar.DAY\_OF\_MONTH) return the year, month, and day. Write a program to perform two tasks:

* Display the current year, month, and day.
* The GregorianCalendar class has the setTimeInMillis(long), which can be used to set a specified elapsed time since January 1, 1970. Set the value to 1234567898765L and display the year, month, and day.

### // T07\_CalendarTest.java

**package** academic.cse110.assignment02.tasks;

**import** **java.util.GregorianCalendar**;

*/\*\**

*\**

*\* @author mrasadatik*

*\*/*

**public** **class** T07\_CalendarTest {

**private** **static** **void** displayCurrentDate() {

GregorianCalendar currentDate = **new** GregorianCalendar();

System.out.println("Current Date (elapsed time January 1, 1970 + " + currentDate.getTimeInMillis() + "millis):");

System.out.println("Year: " + currentDate.get(GregorianCalendar.YEAR));

System.out.println("Month: " + currentDate.get(GregorianCalendar.MONTH) + 1);

System.out.println("Day: " + currentDate.get(GregorianCalendar.DAY\_OF\_MONTH));

}

**private** **static** **void** setElapsedTimeAndDisplay(**long** elapsedTime) {

GregorianCalendar specifiedDate = **new** GregorianCalendar();

specifiedDate.setTimeInMillis(elapsedTime);

System.out.println("Specified Date (elapsed time January 1, 1970 + " + elapsedTime + "millis):");

System.out.println("Year: " + specifiedDate.get(GregorianCalendar.YEAR));

System.out.println("Month: " + specifiedDate.get(GregorianCalendar.MONTH) + 1);

System.out.println("Day: " + specifiedDate.get(GregorianCalendar.DAY\_OF\_MONTH));

}

**public** **static** **void** displayDate() {

displayCurrentDate();

System.out.println();

setElapsedTimeAndDisplay(1234567898765L);

}

}

## Task 08: **Stopwatch**

Design a class named StopWatch. The class contains:

* Private data fields startTime and endTime with getter methods.
* A no-arg constructor that initializes startTime with the current time.
* A method named start() that resets the startTime to the current time.
* A method named stop() that sets the endTime to the current time.
* A method named getElapsedTime() that returns the elapsed time for the stopwatch in milliseconds.

Draw the UML diagram for the class and then implement the class. Write a test program that measures the execution time of sorting 100,000 numbers using selection sort.

### // T08\_StopWatch.java

*/\**

*UML (Unified Modeling Language) Diagram for this program:*

*+-------------------------------------+*

*| StopWatch |*

*+-------------------------------------+*

*| - startTime: long |*

*| - endTime: long |*

*+-------------------------------------+*

*| + StopWatch() |*

*| + getStartTime(): long |*

*| + getEndTime(): long |*

*| + start() |*

*| + stop() |*

*| + getElapsedTime(): long |*

*+-------------------------------------+*

*+-------------------------------------+*

*| T08\_StopWatch |*

*+-------------------------------------+*

*| + selectionSort(int[] arr) |*

*| + getExecutionTime() |*

*+-------------------------------------+*

*\*/*

**package** academic.cse110.assignment02.tasks;

**class** StopWatch {

**private** **long** startTime;

**private** **long** endTime;

@SuppressWarnings("OverridableMethodCallInConstructor")

**public** StopWatch() {

start();

}

**public** **long** getStartTime() {

**return** startTime;

}

**public** **long** getEndTime() {

**return** endTime;

}

**public** **void** start() {

startTime = System.currentTimeMillis();

}

**public** **void** stop() {

endTime = System.currentTimeMillis();

}

**public** **long** getElapsedTime() {

**return** endTime - startTime;

}

}

*/\*\**

*\**

*\* @author mrasadatik*

*\*/*

**public** **class** T08\_StopWatch {

**private** **static** **final** **int** MAX\_NUM\_IN\_A\_ROW = 10;

**private** **static** **void** selectionSort(**int**[] arr) {

**for** (**int** i = 0; i < arr.length - 1; i++) {

**int** minIndex = i;

**for** (**int** j = i + 1; j < arr.length; j++) {

**if** (arr[j] < arr[minIndex]) {

minIndex = j;

}

}

**int** temp = arr[minIndex];

arr[minIndex] = arr[i];

arr[i] = temp;

}

}

**public** **static** **void** getExecutionTime() {

StopWatch stopWatch = **new** StopWatch();

System.out.println("Stopwatch stared...");

System.out.println("Generating an array of 100000 random integers...");

**int**[] numbers = **new** **int**[100000];

**for** (**int** i = 0; i < numbers.length; i++) {

numbers[i] = (**int**) (Math.random() \* 100000);

}

System.out.println("Done.");

System.out.println("Generated 100000 random integers are:");

**for** (**int** i = 0; i < 25; i++) {

System.out.print(numbers[i] + "\t");

**if** ((i + 1) % MAX\_NUM\_IN\_A\_ROW == 0) {

System.out.println();

}

}

System.out.println("... ... ... ... ... ... ... ... ... ...");

System.out.println("Starting selection sort...");

selectionSort(numbers);

System.out.println("Done.");

System.out.println("Sorted generated 100000 random integers are:");

**for** (**int** i = 0; i < 25; i++) {

System.out.print(numbers[i] + "\t");

**if** ((i + 1) % MAX\_NUM\_IN\_A\_ROW == 0) {

System.out.println();

}

}

System.out.println("... ... ... ... ... ... ... ... ... ...");

stopWatch.stop();

System.out.println("Stopwatch stopped.");

System.out.println("Elapsed Time: " + stopWatch.getElapsedTime() + " milliseconds since " + stopWatch.getStartTime() + "milliseconds.");

}

}

## Task 09: Algebra: 2 \* 2 Linear Equations

Design a class named LinearEquation for a 2 \* 2 system of linear equations:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |

The class contains:

* Private data fields a, b, c, d, e, and f.
* A constructor with the arguments for a, b, c, d, e, and f.
* Six getter methods for a, b, c, d, e, and f.
* A method named isSolvable() that returns true if ad - bc is not 0.
* Methods getX() and getY() that return the solution for the equation.

Draw the UML diagram for the class and then implement the class. Write a test program that prompts the user to enter a, b, c, d, e, and f and displays the result. If ad - bc is 0, report that “The equation has no solution.”.

### // T09\_LinearEquation.java

*/\**

*UML (Unified Modeling Language) Diagram for this program:*

*+---------------------------------------------------+*

*| LinearEquation |*

*+---------------------------------------------------+*

*| - a: double |*

*| - b: double |*

*| - c: double |*

*| - d: double |*

*| - e: double |*

*| - f: double |*

*+---------------------------------------------------+*

*| + LinearEquation(a: double, |*

*| b: double, |*

*| c: double, |*

*| d: double, |*

*| e: double, |*

*| f: double) |*

*| + getA(): double |*

*| + getB(): double |*

*| + getC(): double |*

*| + getD(): double |*

*| + getE(): double |*

*| + getF(): double |*

*| + isSolvable(): boolean |*

*| + getX(): double |*

*| + getY(): double |*

*+---------------------------------------------------+*

*+---------------------------------------------------+*

*| T09\_LinearEquation |*

*+---------------------------------------------------+*

*| + runLinearEquationCalculator(scanner: Scanner) |*

*+---------------------------------------------------+*

*\*/*

**package** academic.cse110.assignment02.tasks;

**import** **java.util.Scanner**;

**class** LinearEquation {

@SuppressWarnings("FieldMayBeFinal")

**private** **double** a, b, c, d, e, f;

**public** LinearEquation(**double** a, **double** b, **double** c, **double** d, **double** e, **double** f) {

**this**.a = a;

**this**.b = b;

**this**.c = c;

**this**.d = d;

**this**.e = e;

**this**.f = f;

}

**public** **double** getA() {

**return** a;

}

**public** **double** getB() {

**return** b;

}

**public** **double** getC() {

**return** c;

}

**public** **double** getD() {

**return** d;

}

**public** **double** getE() {

**return** e;

}

**public** **double** getF() {

**return** f;

}

**public** **boolean** isSolvable() {

**return** (a \* d) - (b \* c) != 0;

}

**public** **double** getX() {

**if** (isSolvable()) {

**return** (e \* d - b \* f) / (a \* d - b \* c);

} **else** {

**return** Double.NaN;

}

}

**public** **double** getY() {

**if** (isSolvable()) {

**return** (a \* f - e \* c) / (a \* d - b \* c);

} **else** {

**return** Double.NaN;

}

}

}

*/\*\**

*\**

*\* @author mrasadatik*

*\*/*

**public** **class** T09\_LinearEquation {

**public** **static** **void** runLinearEquationCalculator(Scanner scanner) {

System.out.print("Enter a: ");

**double** a = scanner.nextDouble();

System.out.print("Enter b: ");

**double** b = scanner.nextDouble();

System.out.print("Enter c: ");

**double** c = scanner.nextDouble();

System.out.print("Enter d: ");

**double** d = scanner.nextDouble();

System.out.print("Enter e: ");

**double** e = scanner.nextDouble();

System.out.print("Enter f: ");

**double** f = scanner.nextDouble();

LinearEquation equation = **new** LinearEquation(a, b, c, d, e, f);

**if** (equation.isSolvable()) {

System.out.println("Solution:");

System.out.printf("x = %.2f\n", equation.getX());

System.out.printf("y = %.2f\n", equation.getY());

} **else** {

System.out.println("The equation has no solution.");

}

}

}

## Task 10: The Location Class

Design a class named Location for locating a maximal value and its location in a two-dimensional array. The class contains public data fields row, column, and maxValue that store the maximal value and its indices in a two dimensional array with row and column as int types and maxValue as a double type.

Write the following method that returns the location of the largest element in a two dimensional array: public static Location locateLargest(double[][] a).

The return value is an instance of Location. Write a test program that prompts the user to enter a two-dimensional array and displays the location of the largest element in the array.

### // T10\_Location.java

**package** academic.cse110.assignment02.tasks;

**import** **java.util.Scanner**;

**class** Location {

**private** **int** row;

**private** **int** column;

**private** **double** maxValue;

**public** Location(**int** row, **int** column, **double** maxValue) {

**this**.row = row;

**this**.column = column;

**this**.maxValue = maxValue;

}

**public** **int** getRow() {

**return** row;

}

**public** **int** getColumn() {

**return** column;

}

**public** **double** getMaxValue() {

**return** maxValue;

}

**public** **void** setRow(**int** row) {

**this**.row = row;

}

**public** **void** setColumn(**int** column) {

**this**.column = column;

}

**public** **void** setMaxValue(**double** maxValue) {

**this**.maxValue = maxValue;

}

**public** **static** Location locateLargest(**double**[][] a) {

**int** rows = a.length;

**int** columns = a[0].length;

Location location = **new** Location(0, 0, a[0][0]);

**for** (**int** i = 0; i < rows; i++) {

**for** (**int** j = 0; j < columns; j++) {

**if** (a[i][j] > location.getMaxValue()) {

location.setRow(i);

location.setColumn(j);

location.setMaxValue(a[i][j]);

}

}

}

**return** location;

}

}

*/\*\**

*\**

*\* @author mrasadatik*

*\*/*

**public** **class** T10\_Location {

**public** **static** **void** runMaximalLocator(Scanner scanner) {

System.out.print("Enter the number of rows and columns in the array: ");

**int** rows = scanner.nextInt();

**int** columns = scanner.nextInt();

System.out.println("Enter the array:");

**double**[][] array = **new** **double**[rows][columns];

**for** (**int** i = 0; i < rows; i++) {

**for** (**int** j = 0; j < columns; j++) {

array[i][j] = scanner.nextDouble();

}

}

Location location = Location.locateLargest(array);

System.out.printf("The location of the largest element is %.2f at (%d, %d)\n", location.getMaxValue(), location.getRow(), location.getColumn());

}

}

**THE END**