EECS 1021 – Lab B: Matrix

Dr. James Andrew Smith, PEng and Richard Robison

This lab takes place in the third week of school.

Summary: Practice loops, strings, methods, arrays and helpful methods that Java provides.

Pre-Lab: There are interactive "Lab B" activities on eClass that are due on the Sunday night before Lab B. Make sure to do them on time. View the two "walk-through" videos on Lab B, too. They'll help you.

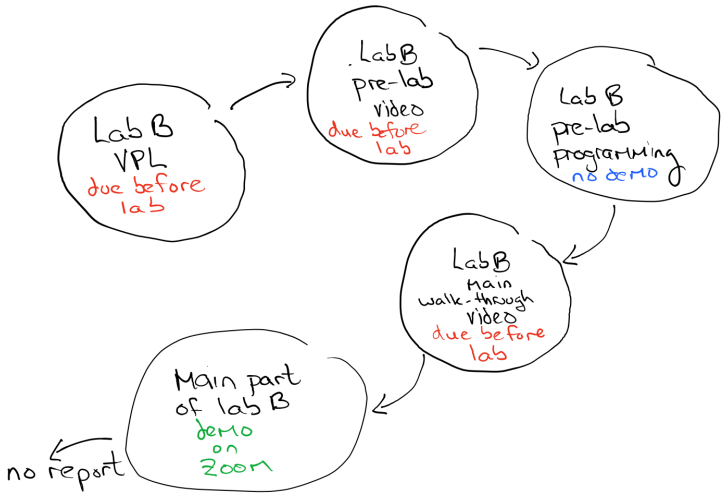


Figure 1 The work you need to do for Lab B. VPL is due on Sunday before your lab.

Intro

Matrices and arrays are key concepts in all branches of engineering. It's why you take courses like MATH 1025 ("Applied Linear Algebra"). It's also why we teach you how to use Matlab in 1st semester. Here we're going to show you that you can work with matrices in Java using objects like arrays.

Due dates.

- *Pre-Lab*: All of the interactive pre-lab activities are due on the Sunday night before your lab.
- *Lab Demo*: The lab demonstration is due on the day of your scheduled lab and must be conducted via a Zoom session with your lab teaching assistant.

Marking Guide:

- All interactive Pre-lab activities are graded out of 1 and count towards your "interactive" activity grade. Any other pre-lab activity is not graded.
- The lab demo is out of 1 and counts towards your lab grade. If you complete the lab it counts as 1 out of 1: 0.2 to parse strings of values, 0.2 to perform one of three math operations on two 2D arrays, 0.2 to display the contents of a 2D integer array and 0.2 to demonstrate all these tasks in a coherent fashion in one program.

Lab Background

This lab introduces several basic concepts of programming, such as using arrays, switch expressions, and I/O.

You will be creating a program that applies various operations to two matrices of numbers.

Arrays

An array is a collection of values of the same type. For example, `var x = new int[]{1, 2, 3};` creates an array of three integers.

Technically speaking, an array is a contiguous block of memory. Arrays can either be created with fixed values like the previous example, or an array can be created with a size and then filled later, like `var x = new int[3];`

To assign a value (`42` for example) to the element of the array (`a` for example) at some index `i`, you would write `a[i] = 42;`

Arrays can be multidimensional. For example, `var x = new int[2][3];` creates a two-dimensional array of three integers (aka a 2x3 matrix).

Note that unlike some modern OOP languages such as Kotlin or Swift, arrays are not objects. They are just a collection of values.

Switch Expressions

Switch Expressions are a way to write a switch statement in a more concise way. They are relatively new to Java, but exist in many other languages.

Suppose you have the following switch expression:

```
class Example {
   static void exampleSwitch() {
      var a = 1; // or any number
      var x = switch (a) {
         case 1 -> "one";
         case 2 -> "two";
         case 3 -> "three";
         default -> "other";
      };
      // x is now "one"
   }
}
```

In this case, the value of `x` is dependent on the value of `a`. That is, if `a == 1`, then `x = "one"`, and similarly for the other cases.

Since the number of possible values is infinite (i.e., it's the set of all possible integers), a `default` case is required in case `a` is not one of the values in the switch expression.

So, if `a == 4`, then `x = "other"`.

Pre-Lab

There are four tasks you should do before starting the lab, all within the Prelab Java class.

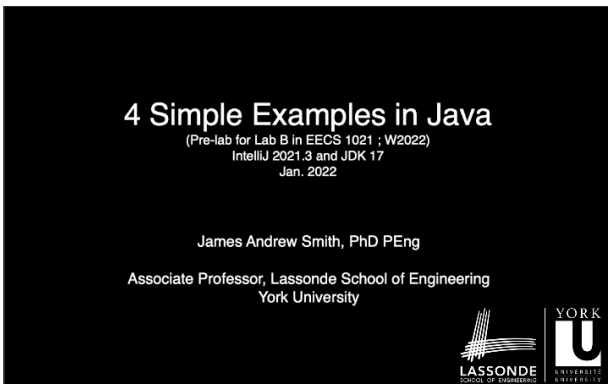


Figure 2 Walk-through of the pre-lab on YouTube: https://youtu.be/bAIPDbgz8GU

- **Task 1**: add elements together from two different arrays and print out the results.
  - Reference: Sharan Kishori: https://bit.ly/31Csrhk
    - Array: Chapter 15
    - length: pg. 125 in Ch 4
    - println: multiple examples
- **Task 2**: read number input from the user and sum up all those numbers, after converting from string to integer.
  - Reference: Sharan Kishori: https://bit.ly/31Csrhk
    - For loops: Chapter 5 of https://bit.ly/31Csrhk
    - parseInt(): pg 321 (Ch 8) of https://bit.ly/31Csrhk
  - Reference: external websites:
    - Scanner:
      - https://www.javatpoint.com/post/java-scanner-next-method
      - https://docs.oracle.com/javase/8/docs/api/java/util/Scanner.html
- **Task 3**: assign a numeric value to a variable based on a character value in another variable.
  - External websites:
    - *Modern* switch-case: https://www.wearedevelopers.com/magazine/modern-java-switch-expressions
  - Reference: Sharan Kishori: https://bit.ly/31Csrhk
    - Ch 5: *old* way of doing switch-case
    - Ch. 9 & pg 403: Exceptions (IllegalArgumentException())
- **Task 4**: fill up a matrix (array) with values using a loop.
  - Reference: Sharan Kishori: https://bit.ly/31Csrhk
    - For loops: Chapter 5
    - length: pg. 125 in Ch 4
    - println: multiple examples

```
package eecs1021;

import java.util.Scanner;

public class Prelab {
   public static void main(String[] args) {
      var array = new int[]{1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
      var array2 = new int[]{10, 9, 8, 7, 6, 5, 4, 3, 2, 1};

      // 1. Iterate through the arrays, summing their corresponding values together and then printing the result.
      //   The output should be "11 11 11 11 11 11 11 11 11 11"


      // 2. Read a line of input using `sc`. The line should be five space-separated integers. Then, print the sum of the integers.

      var sc = new Scanner(System.in);

      // 3. Using a switch expression, assign the value of a variable `x` based on these cases:
      //   - If `flag` is `A`, set `x` to `1`
      //   - If `flag` is `B`, set `x` to `2`
      //   - If `flag` is `C`, set `x` to `3`
      //   - otherwise, throw some Exception;
      //
      //  Then, print `x`

      char flag = 'A';

      // 4. Fill up the matrix `mat` with the following values using a for loop: `0, 2, 4, 6, 8`

      var mat = new int[5];

   }
}
```

Template for the pre-lab. Watch the video for a walk-through and try it yourself.

Main Lab

Specification

In the first part of the program, read the data from the console input:

1. The program should print "Enter the first matrix values: " and then read the following line from the input. The input should be a string of numbers separated by spaces. The amount of numbers should be the number of values in the matrix (in this case, DIMENSION * DIMENSION)
2. Repeat Step 1 for the second matrix values
3. The program should print "Enter the desired operation: " and then read the following line from the input. The input should be one of: *, +, or -.
4. The resulting matrix from applying the operation should then be printed in the following format:

```
[a0, a1, a2, ..., aN]
[b0, b1, b2, ..., bN]
...
[c0, c1, c2, ..., cN]
```

The whole thing is visualized on the right, in a flowchart.[1]

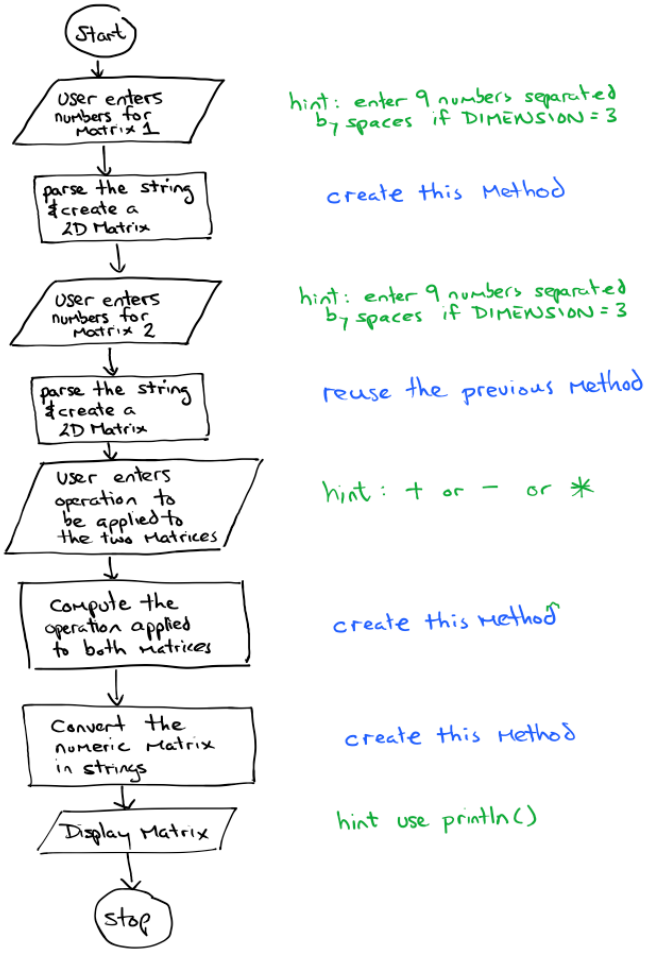


Figure 3 Flowchart view of the program.

Procedure for the main part of the lab

In the `main` method,

1. Create a new `Scanner` object
2. Print the prompt as per the specification and read the inputs

Next, define a function with the signature `private static int[][] parseMatrix(String s, int rowCount, int colCount)`

1. This function should split the input string `s` by " ";
2. For each element in the split string, convert it to an integer and store it in the corresponding element of the matrix

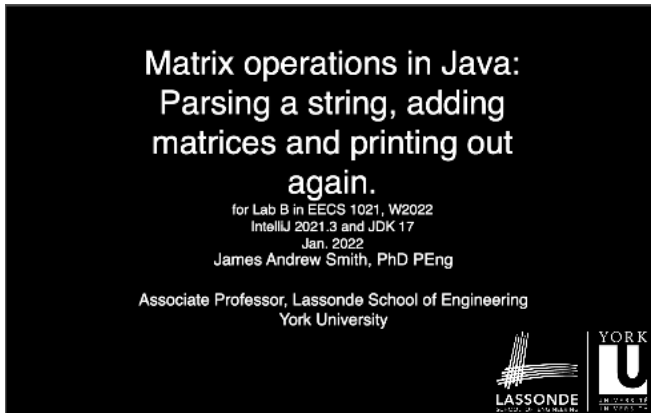


Figure 4 Walk-through of important topics for the main part of the lab: https://youtu.be/XPgW4x8Ep20

[1] Flowcharting: see pgs 10 - 12 in JAVA Programming for Engineers: https://bit.ly/3na8Ts6

3. Return the matrix

Then, define a function with the signature `private static int[][] computeMatrixExpression(int[][] a, int[][] b, String op)`

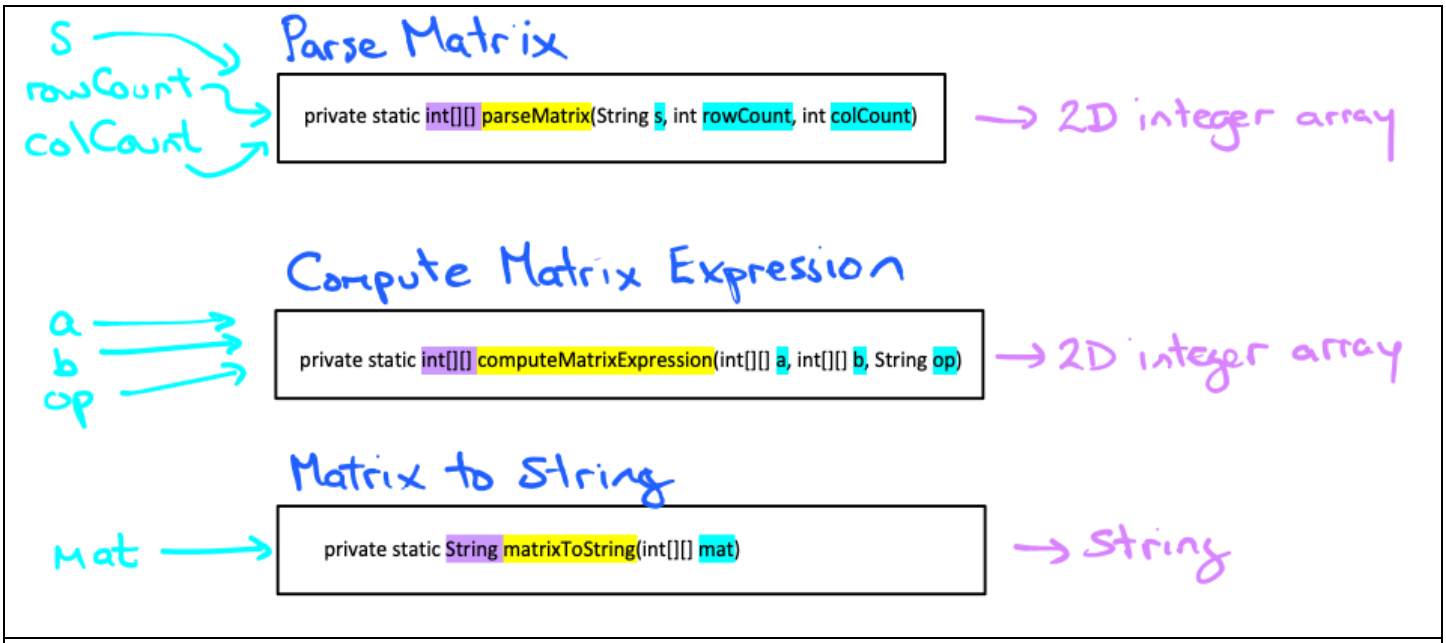
1. Iterate over all elements in the matrices
2. For each element, use a switch expression to assign the result of the operation to the corresponding element of the result matrix
3. For the `default` case, throw an `IllegalArgumentException`

Then, define a function with the signature `private static String matrixToString(int[][] mat)`

1. Create an instance of `StringJoiner` using `\n` as the delimiter
2. Iterate over all rows in the matrix, and use `Arrays#toString` and add the string to the `StringJoiner`
3. Return the string from the `StringJoiner`

Use all of these functions in the `main` method:

1. After having received all the input, call the `parseMatrix` function to parse the matrices.
2. Then, use the `computeMatrixExpression` function and store the result.
3. Finally, convert the matrix to a string and print it.



Your main method should use these three methods. Inputs and outputs are shown, too. You'll need to create them.

```
package eecs1021;

import java.util.Arrays;
import java.util.Objects;
import java.util.Scanner;
import java.util.StringJoiner;

public class Main {
   private static final int DIMENSION = 3;

   /**
    * Parses a matrix from a string.
    * @param s The string to parse. It must consist only of numbers separated by spaces. The amount of numbers should be
equal to {@code rowCount * columnCount}.
    * @param rowCount the number of rows in the matrix.
    * @param colCount the number of columns in the matrix.
    * @return The matrix represented by the string.
    */
   private static int[][] parseMatrix(String s, int rowCount, int colCount) {
      return null;
   }

   /**
    * Returns the matrix which is the result of the operation {@code op} on the matrices {@code a} and {@code b}.
    * @param a The first matrix.
    * @param b The second matrix.
    * @param op The operation to perform. Must be one of {@code +}, {@code -}, {@code *}.
    * @return The matrix which is the result of the operation.
    */
   private static int[][] computeMatrixExpression(int[][] a, int[][] b, String op) {
      return null;
   }

   /**
    * Converts a matrix to a string. For a given matrix {@code [[1, 2, 3], [4, 5, 6], [7, 8, 9]]}, the string representation will
be {@code "[1 2 3]\n[4 5 6]\n[7 8 9]"}.
    *
    * @param mat The matrix to convert.
    * @return The string representation of the matrix.
    */
   private static String matrixToString(int[][] mat) {
      return null;
   }

   public static void main(String[] args) {

   }
}
```

Template for the main part of the lab

Appendix

Pre-lab Template (Fill it in)

```
package eecs1021;

import java.util.Scanner;

public class Prelab {
   public static void main(String[] args) {
      var array = new int[]{1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
      var array2 = new int[]{10, 9, 8, 7, 6, 5, 4, 3, 2, 1};

      // 1. Iterate through the arrays, summing their corresponding values together and then printing the result.
      //    The output should be "11 11 11 11 11 11 11 11 11 11"


      // 2. Read a line of input using `sc`. The line should be five space-separated integers. Then, print the sum of the integers.

      var sc = new Scanner(System.in);

      // 3. Using a switch expression, assign the value of a variable `x` based on these cases:
      //    - If `flag` is `A`, set `x` to `1`
      //    - If `flag` is `B`, set `x` to `2`
      //    - If `flag` is `C`, set `x` to `3`
      //    - otherwise, throw some Exception;
      //
      //   Then, print `x`

      char flag = 'A';

      // 4. Fill up the matrix `mat` with the following values using a for loop: `0, 2, 4, 6, 8`

      var mat = new int[5];

   }
}
```

Lab Template (Fill it in)

```
package eecs1021;

import java.util.Arrays;
import java.util.Objects;
import java.util.Scanner;
import java.util.StringJoiner;

public class Main {
    private static final int DIMENSION = 3;

    /**
     * Parses a matrix from a string.
     * @param s The string to parse. It must consist only of numbers separated by spaces.
     * The amount of numbers should be equal to {@code rowCount * columnCount}.
     * @param rowCount the number of rows in the matrix.
     * @param colCount the number of columns in the matrix.
     * @return The matrix represented by the string.
     */
    private static int[][] parseMatrix(String s, int rowCount, int colCount) {
        return null;
    }

    /**
     * Returns the matrix which is the result of the operation {@code op} on the
     * matrices {@code a} and {@code b}.
     * @param a The first matrix.
     * @param b The second matrix.
     * @param op The operation to perform. Must be one of {@code +}, {@code -}, {@code *}.
     * @return The matrix which is the result of the operation.
     */
    private static int[][] computeMatrixExpression(int[][] a, int[][] b, String op) {
        return null;
    }

    /**
     * Converts a matrix to a string. For a given matrix
     * {@code [[1, 2, 3], [4, 5, 6], [7, 8, 9]]}, the string representation will be
     * {@code "[1 2 3]\n[4 5 6]\n[7 8 9]"}.
     *
     * @param mat The matrix to convert.
     * @return The string representation of the matrix.
     */
    private static String matrixToString(int[][] mat) {
        return null;
    }

    public static void main(String[] args) {

    }
}
```