**Name**

**Advanced Programming in Java**

**Lab Exercise 11/1/2022**

**Lesson 35 – Two Dimensional Arrays**

Consider the following *a* matrix for problems 1 - 11:



1. Write a single line of code that will create the above integer array. Call the array *a*.

2. Write a line of code that will printout the array element occupied by –101.

3. The above table can be described as:

(a) an Array

(b) a Matrix

(c) numbers that could be represented as subscripted variables

(d) a, b, and c

(e) none of these

4. Write a line of code that will print the number of rows in matrix *a*.

5. Write a line of code that will print the number of columns (in matrix a) in the row given by index 2.

6. What is printed by *System.out.println(a[1][3]);* ?

7. Show what the printout will look like:

for (int row = 0; row < a.length; row++)

{

for(int col = 0; col < a[row].length; col++)

{

System.out.print(a[row][col] + “\t”);

}

System.out.println(“ ”);

}

8. What is printed by the following?

Arrays.sort(a[0]);

System.out.println(Arrays.binarySearch(a[0],5));

9. What is printed by the following?

Arrays.sort(a[0]);

System.out.println( Arrays.binarySearch(a[0],0) );

10. Show what the matrix *a* would look like after the following code executes:

for (int row = 0; row < a.length; row++)

{

for(int col = 0; col < a[row].length; col++)

a[row][col] = row \* col;

}

11. Show what the matrix *a* would look like after the following code executes:

Arrays.fill(a[2], -156);

12. Must all two-dimensional arrays have the same number of columns in each row?

In the remaining problems, some of the code might not compile or will give a run-time

exception. If that’s the case then state, “Won’t compile” or “Run-time exception.”

13. What is printed by the following?

double d[][] = new double[8][25];

System.out.println(d[4][2]);

14. If *x* and *y* both represent two-dimensional *int* arrays and have identically the same

elements, what does the following print?

System.out.println( Arrays.equals(x,y) );

15. Is it possible to sort *z* (a two-dimensional array) with *Arrays.sort(z);* ?

16. Is it possible to use one of the *sort* methods of the *Arrays* class to sort a single row (index3) of the two-dimensional matrix *g*? If so, show the code that will accomplish this.

**Multiplying a Matrix by Another Matrix**

But to multiply a matrix **by another matrix** we need to do the "[dot product](https://www.mathsisfun.com/algebra/vectors-dot-product.html)" of rows and columns ... what does that mean? Let us see with an example:

To work out the answer for the **1st row** and **1st column**:



The "Dot Product" is where we **multiply matching members**, then sum up:

(1, 2, 3) • (7, 9, 11) = 1×7 + 2×9 + 3×11  
    = 58

We match the 1st members (1 and 7), multiply them, likewise for the 2nd members (2 and 9) and the 3rd members (3 and 11), and finally sum them up.



(1, 2, 3) • (8, 10, 12) = 1×8 + 2×10 + 3×12  
    = 64

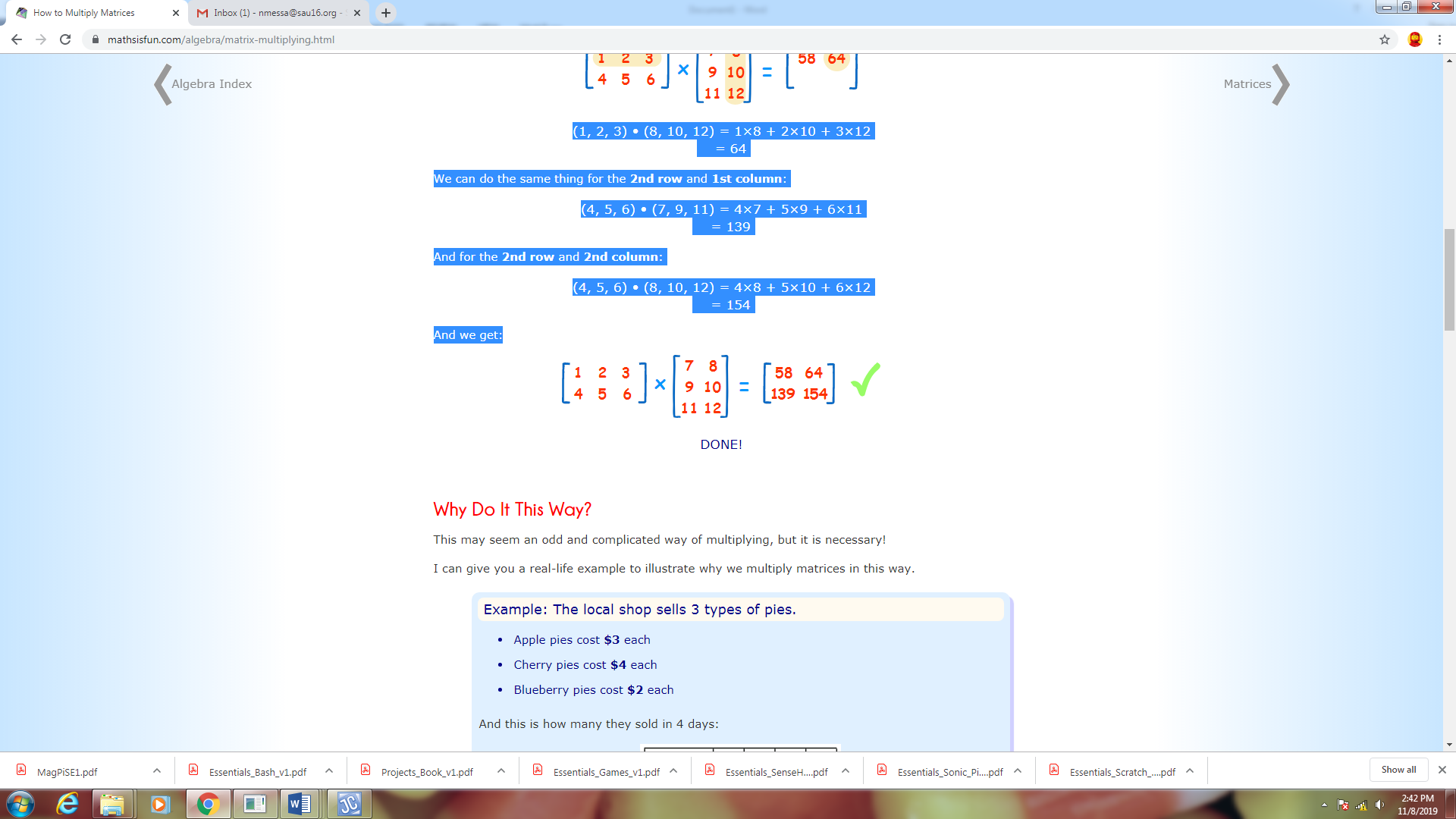
We can do the same thing for the **2nd row** and **1st column**:

(4, 5, 6) • (7, 9, 11) = 4×7 + 5×9 + 6×11  
    = 139

And for the **2nd row** and **2nd column**:

(4, 5, 6) • (8, 10, 12) = 4×8 + 5×10 + 6×12  
    = 154

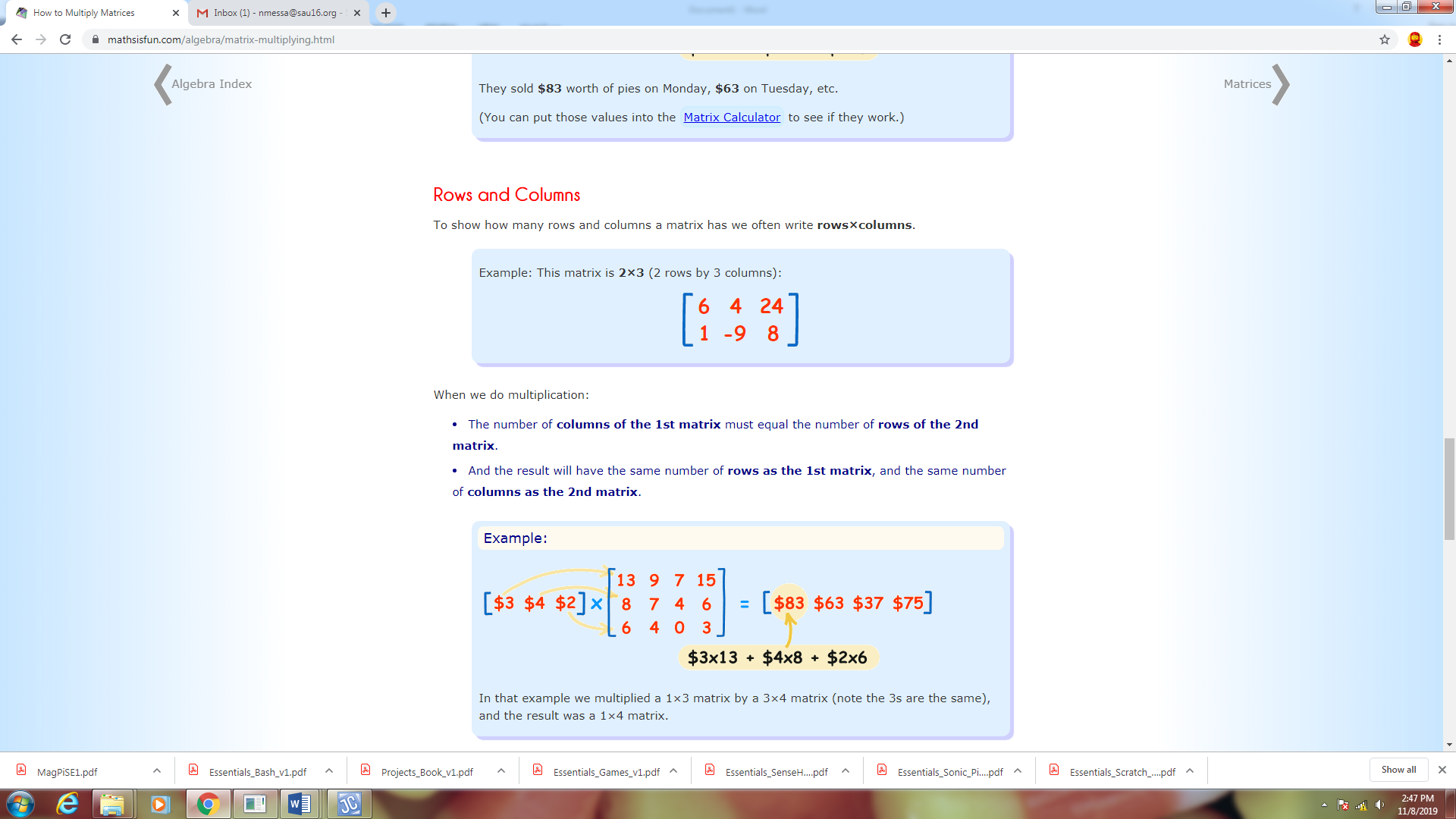
And we get:



Rows and Columns

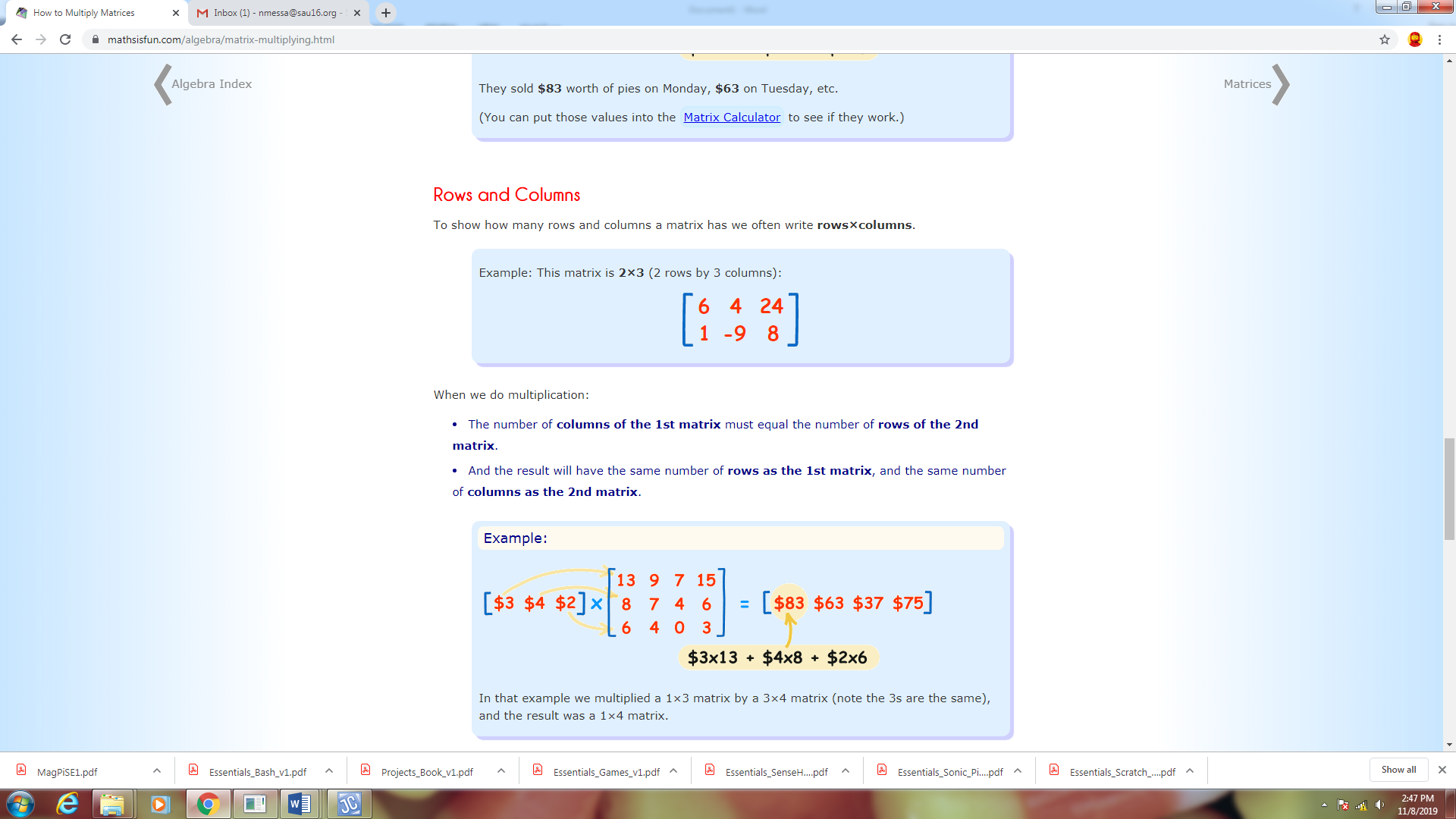
To show how many rows and columns a matrix has we often write **rows × columns**.

Example: This matrix is 2×3 (2 rows by 3 columns):



When we do multiplication:

* The number of columns of the 1st matrix must equal the number of rows of the 2nd matrix.
* And the result will have the same number of rows as the 1st matrix, and the same number of columns as the 2nd matrix.



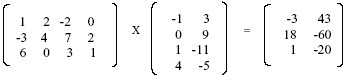
*In General:*

To multiply an **m×n** matrix by an **n×p** matrix, the **n**’s must be the same  
and the result is an **m×p** matrix.

matrix multiply rows cols

**Project… Matrix Multiplication**

The primary objective in this project will be to write code that will multiply the following two matrices and produce the indicated answer.



If you are not familiar with the intricacies of matrix multiplication, Appendix AA is supplied to provide a brief overview of the subject. In fact, the above example is used in that appendix.

Create a project called *MatrixStuff* that consists of two classes. These two classes will be called *Tester* and *MatrixMult* and will meet the specifications listed below.

**The *Tester* class (*main* method):**

1. Hard code the *int a[ ][ ]* array so as to be comprised of the 3 X 4 matrix on the left in the example above.

2. Hard code the *int b[ ][ ]* array so as to be comprised of the 4 X 2 middle matrix in the

example above.

3. Call a *static* method of the *MatrixMult* class called *mult* in which we pass the *a* and *b*

arrays (matrices) as arguments and receive back an integer array as the product matrix.

4. Print the product matrix.

5. The output of *main* should appear as follows:

-3 43

18 -60

1 -20

**The *MatrixMult* class:**

1. No constructor.

2. Create a single *static* method called *mult* that receives two *int* arrays (matrices) as

parameters that are to be multiplied in the order in which they are received.

3. The *mult* method is to return an array that is the product matrix of the two parameter

arrays it receives.

4. The code in the *mult* method is to determine the dimensions of the matrices that it

receives and set up a “product” array (matrix) to be returned with the appropriate

dimensions.

5. The code in *mult* should be general so as to adapt to any two matrices to be multiplied;

however, for the sake of simplicity, you may assume that the matrices received as

parameters **are** always compatible with each other for multiplication.

6. The code in the *mult* method will multiply the two incoming matrices so as to correctly

produce each element of the product matrix.

**Project… Matrix Multiplication with File Input**

Modify the previous project so that the matrices to be input are input from a text file. The “rules” of the text file are:

The start of a new matrix will be indicated with “matrix”. The start of a new row will be

indicated with “row” followed by the numbers assigned to each column of that row.

Call your data file *MatrixData.txt*. It should have the following content:

matrix

row

1

2

-2

0

row

-3

4

7

2

row

6

0

3

1

matrix

row

-1

3

row

0

9

row

1

-11

row

4

-5

Output should be identical to that in the previous project (since the two matrices to be multiplied are the same as before).