# TEAM NAME: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Recorder:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Facilitator for today: Other Team Members Present (please print):

Task 1: Review the lab from Wednesday. List all of the different exceptions that you found. Be prepared to share with the class and share how you would prevent the Exception from occurring.

Task 2: 1. For the following method (from the Array2D lab), identify as many of the Exceptions that might occur and describe why that Exception could be thrown. Be prepared to share with the class.

 public static double getRowTotal(double [][] matrix, int row)
 {
 double answer = 0;

 if ( matrix.length > row )
 {
 for ( int i = 0; i < matrix[row].length; i++ )
 answer += matrix[row][i];
 }
 else
 {
 System.out.println("Row index not acceptable");
 answer = Double.MIN\_VALUE;
 }
 return answer;
 }

Task 2: 2. Rewrite the method to prevent all of those Exceptions.

Task 3: Exploring Exceptions. After the brief lecture and demo of Exception handling, answer each of the following questions, using the Exception document for reference.

1. All Exceptions share the methods that you see in the Exception API. In the diagram, all of the Exception classes that follow the Exception class will also share all of those methods. We will see how to implement such a hierarchy of classes after I/O. So, how many methods are there that Exceptions can use. Hint: Look at the code snippet for an example of one of those methods. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
2. The try surrounds the code that we will conditionally execute. If an exception is raised, execution interrupts at the point of failure and proceeds with the catch block. What is inside of the catch block “parameter”?
3. So, what is ioe in the code snippet example?
4. Looking at the anatomy of an Exception message (on the same sheet as the diagram of the Exception hierarchy), what do you think the method printStackTrace does?

Task 4: Rewrite the original method as a series of try/catch blocks to catch and “handle” the various exceptions that you identified in Task 2: 1.