# COMP 1020
# Lab 5

## MATERIAL COVERED

- Exceptions
- File reading and writing

Notes:

- The three exercises are independent – you can do them in any order.
- Only one of the three exercises is required.
- The Bronze and Silver exercises are similar – both will read in an entire text file and write it out again in modified form. About 75% of the code is the same in the two exercises. If you do both of them, use a copy of your Bronze program as a starting point for your Silver program.


## Adding line numbers to a file

1. Begin with the supplied file **`TemplateLab5Bronze.java`**.
2. Complete the main method so that it will read in the supplied file **`testLab5Bronze.txt`** and write out a modified version of this file as **`outputLab5Bronze.txt`**. (These two names are predefined for you as global constants.) The output file should be the same as the input file, except that every line should have a line number added to the front, in the format shown below.

   The supplied file **`testLab5Bronze.txt`** contains:

```
This is the top line

The previous one is blank.
Short one.
This is the last one.
```

   Your program should create an output file **`outputLab5Bronze.txt`** which contains:

```
1: This is the top line
2:
3: The previous one is blank.
4: Short one.
5: This is the last one.
```


## Adding indentation to a Java file

1. Begin with the supplied file **`TemplateLab5Silver.java`**.
2. Complete the main method so that it will read in the supplied file **`Lab5SilverTest.java`** which contains a Java program, but with randomized indentation. Your program should fix the indentation as described below, and write out a new version of this file named **`Lab5SilverOutput.java`**. (These names are predefined for you as global constants.) To fix the indentation, do the following things:

a. Remove any existing blanks from the front of each line. The built-in **`String`** method with the signature **`String trim()`** can be used to do this easily. It will remove all leading and trailing white space from any **`String`**. For example,

   **`"  test one  ".trim()`** will give the result **`"test one"`**

b. Add blanks to the front of each line to indent it properly. The amount of indentation should be 0 for the first line. Note the supplied **`String blanks(int n)`** method which will give you a **`String`** containing **`n`** blanks.

c. If any line contains a **`"{"`** character, then increase the indentation for all following lines (but not this one) by an extra **`INDENT_AMOUNT`** blanks. This is a predefined constant. Note that the **`String`** method **`boolean contains(String)`** will tell you whether or not one **`String`** is contained inside another. For example, **`"Test".contains("e")`** will return **`true`**.

d. If any line contains a **`"}"`** character, then decrease the indentation for this line, and all following lines, by **`INDENT_AMOUNT`**.

e. This makes the assumptions that **`{ }`** braces are the only reason to ever indent a line in a Java program (which is not true), that no line will ever contain more than one brace character (also not necessarily true), and that the braces are properly matched up. But it will do a reasonable job of any file for which these assumptions are true.

For example, if the input file contained:

```
     while(x){
doThis();
   if(y){
              doThat();
}//if
                        }//while
```

Then the output file should contain:

```
while(x){
     doThis();
     if(y){
          doThat();
     }//if
}//while
```


# Exceptional Exceptions

In this exercise, you will use Exceptions to complete a program, including throwing an Exception from the catch block that is handling a previous exception. Begin with the supplied file **`TemplateLab5Gold.java`**. The static variables **`intArray`** and **`numElements`** form a standard partially-full array of integers. The **`insertIntoArray`** method adds a new integer to this list. It does no error checking and will throw an **`ArrayIndexOutOfBoundsException`** when the array is full. Do not modify this method in any way. The **`expandArray`** method will double the capacity of **`intArray`**. Add the following exceptions and exception handling to this program:

1. Modify the **`addElement`** method (which, at the moment, simply calls the **`insertIntoArray`** method) so that it will catch the **`ArrayIndexOutOfBoundsException`** thrown by **`insertIntoArray`**. When it does, it should print **`"Expanding the array"`**, call **`expandArray`** and then call **`insertIntoArray`** again, now that there is more room. Now everything should work. The main program will add the numbers 1-20 with no problem.

2. But it has been decided that the array should never be allowed to become any larger than **`MAX_SIZE`** (predefined for you as 16). Modify the **`expandArray`** method so that it will not expand the array any more after it reaches this size. Instead, it should throw a **`RuntimeException`** containing the message **`"Array at maximum capacity"`**.

3. You will now have to catch the **`RuntimeException`** in the **`addElement`** method. Note that the place that the exception will occur is inside the catch block from part 1. The **`addElement`** method should respond by printing out **`"Error in addElement: "`** followed by the message contained in the Exception. It should then throw a **`RuntimeException`** itself, containing the message **`"Element xxx not added."`**.

4. Catch this **`RuntimeException`** in the main program, where it should print the message **`"Error in main: "`** followed by the message in the exception.

5. When working correctly, the output produced by the program should be:

```
element 1 added
element 2 added
Expanding the array
element 3 added
element 4 added
Expanding the array
element 5 added
element 6 added
element 7 added
element 8 added
Expanding the array
element 9 added
element 10 added
element 11 added
element 12 added
element 13 added
element 14 added
element 15 added
element 16 added
Expanding the array
Error in addElement: Array at maximum capacity
Error in main: Element 17 not added.
```