# *Programming III (420-B31-HR)*

# *Lab 3 –Error Handling and Exceptions; Efficiency*

Date assigned & due: Tuesday, September 6, 2016

**\* Reminder that class starts at 9 am today, not 8 am. This is a short lab since we will be having a lecture first, at 9 am.**

**Objectives:**

1. Review:
   1. Throwing and catching exceptions
   2. Creating new exceptions
2. Learn to determine the Big-O notation for an algorithm

**Reference:** Exceptions notes.

Efficiency notes.

**To be handed in:**

1. The Lab 3 Moodle quiz
2. Your ***username*\_B31\_L03\_Exceptions\_Efficiency** folder should be zipped and uploaded to **Moodle**. Show your lab to the teacher before you leave to get it marked.

# Review of Java’s Exception Handling

**Objectives:** Review throwing and catching exceptions and creating your own exception.

**To Start:**

1. Copy the **B31\_L03\_Exceptions\_Efficiency** folder from the **Moodle page** for the course. Rename it to ***username*\_B31\_L03\_Exceptions\_Efficiency**.
2. Start **Eclipse** and use your **420-B31\Labs** folder as your workspace.
3. Create a new **Java Project** called ***username*\_B31\_L03\_Exceptions\_Efficiency**. (Use the **JavaSE-1.8** execution environment JRE.)

**To Do:**

## Open the **MiscFunctions** class. The method **f()** takes a double precision argument x and returns . The **ExceptionTester** class reads in a double precision value, calls **f()** for that value and prints the result. Run **ExceptionTester** and complete the following table:

|  |  |
| --- | --- |
| **x** | **Value returned from f(x)** |
| 1.2 | 0.8532291603080615 |
| 1 | 0.0 |
| 0.9 | NaN |

**Explanation**: Since the square root of a negative number is not a real number, x must not be less than 1.

## Modify the method **f()** to throw an **ArithmeticException** if x is less than 1. Display an error message that displays the value of x that was used and states that it must not be less than 1. Use the data from the table above to test your exception. Your output for the last test should be similar to:

Exception in thread main

java.lang.ArithmeticException: ERROR: Parameter to f(x) must not be less than 1. 0.9 is invalid.

at me\_b31\_l03\_exceptions\_and\_testing.MiscFunctions.f(MiscFunctions.java:19)

at me\_b31\_l03\_exceptions\_and\_testing.ExceptionTester.main(ExceptionTester.java:13)

Process exited with exit code 1.

## The method **checkedSorted()** in the **MiscFunctions** class verifies that a String array is sorted. The **arrayTester** class contains code to test it. Run **arrayTester**.

Now instead of displaying a message if the array is not sorted, we want to raise a user defined exception.

## Create a subclass of the **RuntimeException** to create a new exception called **UnSortedArrayException**. It should be similar to the **ShapeException** class in the notes.

## Modify the **checkSorted()** method so that it raises the **UnSortedArrayException** instead of displaying a message. Test your change.

## Now we want to validate the **numElements** parameter. Throw an **IllegalArgumentException** if **numElements** < 0, is equal 0 or is greater than the length of the array. The message should indicate which of these situations has occurred and should include the value of **numElements**.

## Test your changes using the **arrayTester** class.

## Add a try/catch block to **checkSorted()** to print the message “*The array has not been initialized*” if **testArray** has not been initialized.

## Add code to **arrayTester** to test your changes.

# Program Efficiency

**Objectives:** Learn how to calculate the relative run-time for an algorithm and to determine the corresponding big-O notation.

**To Do:**

## What does the following algorithm do? Write your answer in the appropriate location in ***username*\_B31\_L03\_Answers.docx**

public static int doSomething( int array[], int n ) {

int k = 0;

for ( int i = 1; i < n; i++ )

if ( array[i] > array[k] )

k = i;

int temp = array[0];

array[0] = array[k];

array[k] = temp;

k = 1;

for ( int i = 2; i < n; i++ )

if ( array[i] > array[k] )

k = i;

return array[k];

}

## What is the time complexity (Big-O notation) of the algorithm in question 2? Write your answer in the appropriate location in ***username*\_B31\_L03\_Answers.docx**

# Moodle Quiz

1. Log on to **Moodle**, go to the **Programming III** course page and complete the **Lab 3 Review Quiz by September 9**.

**Marking Scheme**

|  |  |  |
| --- | --- | --- |
|  | **Mark** | **Out of** |
| **Part A: Exceptions** |  |  |
| MiscFunctions Class |  | 17 |
| UnSortedArrayException Class |  | 4 |
| **Part B: Efficiency** |  |  |
| Question 1 & 2 |  | 3 |
| **Total** |  | **24** |