**COMP 2150 – Fall 2019**

**Homework 5: Exception Error Handling (56 points)**

Number of People: Individual. Feel free to ask me for help, or visit the Computer Science Learning Center (<http://www.memphis.edu/cs/current_students/cslc.php>), or visit the Fri/Sat 9 – 10 am supplemental class @DH103

Due: (Oct 6, 2019 ) (SUNDAY) 11:59 pm

Submission: Zip all your **Java source files** (the one with .java extension) (you can zip the entire project folder if using an IDE) into a single zip file and upload it to the proper folder in the eLearn dropbox at <https://elearn.memphis.edu>.

Coding Style: (use the Format tool in Eclipse) . Use standard Java naming conventions for **variableAndMethodNames**, **ClassNames**, **CONSTANT\_NAMES**. You must have comments for all your codes. Comments should focus on what and why and NOT how.

Graders:

Naeem Khoshnevis ([**nkhshnvs@memphis.edu**](mailto:nkhshnvs@memphis.edu)) (Section 001)

Utsav Shrestha ([**shrstha3@memphis.edu**](mailto:shrstha3@memphis.edu) ) (Section 002)

Questions about grading? Please contact your grader first.

Grader: Rong Qi, [rqi@memphis.edu](mailto:rqi@memphis.edu). Questions about grading? Please contact her first!

**Objectives: : Exception Error handling (Try Catch Throw, and finally)**

1. (16 points) This exercise practices using try-catch block to capture a situation when an object reference is initialized to **null.** This exercise also demonstrates the use of multiple try-catch blocks, importance of catch blocks orders, and error messages from the default Exception handling. This program compiles without error as it without changes.

Write your explanation on the source code with comments.

Steps

1. Download the program Assign5\_7TryCatch.java from eLearn and run it. It should crash with a message.
2. Click on the error message (last line) should bring you to the i.tostring() evil.
   * 1. Explain what happen.

**We tried to use a method on the “i” object which was set to null. This threw a ‘java.lang.NullPointerException” because the object was null.**

1. Comment out the “evil line” (line 15)
   * 1. Explain what happen.

**Commenting out the first call to `SOP(i.toString())` allowed the JRE to move to the next line, which is wrapping the ` SEP (i.toString())` method in a `try-catch` block. The try catch block is setup to catch the ‘java.lang.NullPointerException` and print the stack trace.**

1. Comment out the e.printStackTrace() line 30 in the “NullPointerException” catch ; and uncomment it back and observe the difference.
   * 1. Explain what happen.

**The exception was caught still when line 30 was commented out, but the stack trace was not printed (the stack trace meaning what line or series of methods triggered the exception).**

1. Add i = 100 at line 24 (inside the 2nd try block), and run it,

Change i = null, and run it

* + 1. Explain what happen

**When `i` is defined to be an Integer of `100` the `i` variable is no longer null, and the `SEP(i.toString())` method works perfectly (outputs `100`). When the `i` variable is defined as `null` the ` SEP (i.toString())` method throws a `java.lang.NullPointerException` because, once again, you cannot call a method on a `null` variable.**

1. Add another catch block before finally (~ line 32) after the “NullPointerException” catch

catch (Exception x) {

**System.*err*.**println**(" I am Exception !!! ");**

**x.**printStackTrace**();**

}

1. Run it, explain what happen;

**Nothing happens beyond the first two try-catch blocks outputting their respective messages and stack traces. This is because the exception is already handled by the last try-catch block.**

1. Change the order of the two catches (put NullPointerException) after the catch (Exception x) that you entered in step.

Explain what happen.

**The additional catch block now handles the exception. This is because `java.lang.NullPointerException` is a subclass of the `java.lang.Exception` class, thus it being handled by the more “general” try-catch block.**

1. Put them back in a right order. Now add new line

int y = 9/0; in front of i=100; run it

i. Explain what happen

**The additional try-catch block handles the exception thrown by defining `y` to be a result of a division by zero. The exception thrown is `java.lang.ArithmeticException` which is a subclass of `java.lang.Exception`**

1. (10 points) A CheckAge.java program is provided for this question. This program crashes and throws an exception message if you enter a negative value. For this question, define your own exception class so that the program will throw your custom exception to make the error message more user-friendly.
2. (15 marks) Write a program that creates an integer array with 50 random values, prompts the user to enter the index of an element in the array between 0 and 49, then displays the corresponding element value. If the specified index is out of bounds, display an error message (e.g. “Out of Bounds”) and ask the user to enter another index. Use a while loop that will keep prompting the user until a valid input is received. To handle invalid inputs, write two versions of your program: one that uses exception handling, and one that uses defensive programming. Assume a user will always enter numbers. Verify that the displayed element number is indeed from the index number entered

**Sample run:**

**list[50] of random int generated**

14 37 19 98 31 1 68 11 18 90

36 14 37 63 68 54 64 51 80 75

66 85 30 65 97 66 87 47 52 63

9 22 68 70 11 32 23 69 37 12

62 62 76 89 33 0 43 87 30 27

**Enter an index: 55**

**Out of Bound. Try again:50**

**Out of Bound. Try again:5**

**The element is 1**

1. (15 pts) “An InputMismatchException is thrown by a Scanner to indicate that the token retrieved does not match the pattern for the expected type, or that the token is out of range for the expected type”. For example, if you use the nextDouble to read a token that is not of the type double.

Write a program that prompts the user to enter a mathematical formula (e.g., 4.3 + 5.1) and then displays the result. Your program should prompt the user to try again if the input has an invalid number or invalid operator (i.e., valid operators are: +, -, /, and \*). Assume the user enters spaces between the operands and the operator. **Read the numbers using the Scanner’s nextDouble method, and the operator using the next method.**

Hints:

 Use a while loop that will keep prompting the user for input as long as it is invalid.

 Use a try-catch statement to check for the numbers’ validity.

 Use a selection statement (e.g., if, switch, etc.) to check for the operator’s validity.

**Sample run**

Enter a simple mathematical formula: 9.1 plus 4.2

Invalid Operator. Try again.

9.1x + 4.2

Invalid number format. Try again.

9.1 + 4.2

Result: 13.3