Alec Lund

Programming II: 6:15pm

1/28/20

50pts total

The purpose of this lab is to practice exception handling in Java.

\*Please don't forget to use proper code style:  include a program header and comment your code.\*

Download the file Lab2.java from D2L. You will also eventually need to create a plain-text file called input.txt. This file should initially contain a line of any five (reasonable) numbers, separated by spaces. Don’t create it yet (wait until after you answer Question 1).

**What to submit:**  (1) lastname\_Lab2.java, updated accordingly to instructions below; (2) This file, with your answers to the questions appearing below filled in.

**1)** What happens when you try to run Lab2 without input.txt being present in the same directory?

An Unhandled Exception Error occurs for the exception “FileNotFound”, causing the program to crash.

**2) C**reate input.txt in the same directory as your .java file, include a line of five numbers separated by spaces. Now try running Lab2 again. What happens?

 The same Unhandled Exception error occurs because the FileNotFound exception handle is not in place.

**3)** Deal with the problems you encountered in (1) and (2) using your knowledge of Java’s exception-handling system. After implementing your exception-handling code, run your program and describe what happens.

 The numbers written in the text file get printed out as doubles. The program runs fine with the FileNotFound exception handle implemented.

If needed, add more exception-handling code, clear error messages, or whatever else you find necessary to handle I/O problems “gracefully” (without crashing or exiting the program in a way that makes it difficult to work with for the user).

**4)** Edit input.txt by adding some non-numeric data to the file. For example, add “abc” to the middle of your data line. What happens when you run the program on this file?  An input mismatch exception occurs from the scanner and the application exits.

Add whatever exception-handling code you find most useful to handle the problem(s) introduced by this question.

**5)** There exists a class called Exception, and it is the generic ancestor of all “problem-specific” exception classes. This means that it is syntactically legitimate for you to write all your catch statements to catch this generic Exception. You would never have to worry about specific exception types… and life would be wonderful! … or would it?

a) Why might you not want to always catch generic Exception?

If you have multiple possible exceptions, it makes the exception ambiguous and can lead to future problems when attempting to debug and find the source of the problem.

b) When might it be useful to catch generic Exception?

When not knowing all the possible exceptions it might be good to add a general underlying exception catcher just to make sure the program doesn’t exit and mess up the program flow.