**Name:**

**Java Programming**

**Exception Handling**

**Lab Exercise 12/11/2019**

An exception is simply an error. Instead of saying an error occurred, we say that an

**exception is thrown**.

**Two types of exceptions:**

1. Checked
2. Unchecked

**Two choices for handling checked exceptions:**

1. Handle the exception with ***try*, *catch*, *finally***.
2. Put a *throws IOException* (or some other appropriate checked exception) tag on the method signature as in the following example:

public void readTheDisk( ) **throws IOException**

{

… code that uses a file reader…might encounter a corrupt file…

}

**Try, Catch, Finally Example:**

public void myMethod(double d)

{

try

{

String s = in.nextLine( ); //**might produce an IOException**

int x = Integer.parseInt(s); //**bad s might produce a NumberFormatException**

}

catch (IOException e)

{

System.out.println(“Input/output error ” + e);

}

catch (NumberFormatException e)

{

System.out.println(“Input was not a num ” + e);

}

finally

{

//optional but code will always execute

}

}

**Project… Keep Trying**

Create a new project called *ExceptionsProjects* that will contain two classes, *Tester* and

*FileInput*. Create the *FileInput* class by modifying your *BaseClass* project as

follows:

import java.io.\*; //necessary for File and IOException

import java.util.\*; //necessary for Scanner

public class **FileInput**

{

public static void readTheFile(String fileName) throws IOException

{

Scanner sf = new Scanner(new File(fileName));

int maxIndx = -1; //-1 so when we increment below, the first index is 0

String text[] = new String[100]; //declare more than we need

while(sf.hasNext( ))

{

maxIndx++;

text[maxIndx] = sf.nextLine( );

}

//maxIndx is now the highest index of text[], = -1 if no lines of text.

sf.close( ); //we opened file so we must close it

for (int j = 0; j <= maxIndx; j++)

{

System.out.println(text[j]);

}

} //end of FileInput

} //end of class

Now create a *Tester* class with a *main* method in which you repeatedly loop while inputting a file

name from the keyboard. Also, inside the loop call the *readTheFile* method of the *FileInput* class

and pass the file name input from the keyboard as a parameter. Set up a *try-catch* pair in *main* so

as to keep looping if *readTheFile* passes an *IOException* up the calling chain. If everything in

*readTheFile* completes successfully, then in *main* release from the loop and output “It worked.”

Provide for an escape from the loop by informing the user that he can enter the word “exit”. This

should provide a release from the loop and print “It did not work”.