

Lab #6: Exception Handling

Laboratory Exercises for Monday, October 28

(Source: Bloss, A and Ingram, N.J., Lab Manual Java Software Solutions, Pearson, 2009)

The following laboratory exercises should be completed by the end of today's laboratory session. Upon completion of each exercise, you are expected to have the lab instructor or one of the lab assistants check the correctness of your program by demonstrating its execution. Make sure you have a printout of the source code ready before demonstrating your program. At that time, the lab instructor (or one of the slab assistants) will assign a grade for the completed program. If you have not completed both exercises by the end of the laboratory session, you still have to make sure you submit a printout of any partially working program to get partial credit for your efforts. **Failure to do so will result in a grade of ZERO for the lab exercise.**

Exercise #1: Exceptions Aren't Always Errors

File *CountLetters.java* contains a program that reads a word from the user and prints the number of occurrences of each letter in the word. Save it to your directory and study it, then compile and run it to see how it works. In reading the code, note that the word is converted to all upper case first, then each letter is translated to a number in the range 0..25 (by subtracting 'A') for use as an index. No test is done to ensure that the characters are in fact letters.

1. Run *CountLetters* and enter a phrase, that is, more than one word with spaces or other punctuation in between. It should throw an *ArrayIndexOutOfBoundsException*, because a non-letter will generate an index that is not between 0 and 25. It might be desirable to allow non-letter characters, but not count them. Of course, you could explicitly test the value of the character to see if it is between 'A' and 'Z'. However, an alternative is to go ahead and use the translated character as an index, and catch an *ArrayIndexOutOfBoundsException* if it occurs. Since you don't want to do anything when a non-letter occurs, the handler will be empty. Modify this method to do this as follows:
 - Put the body of the first for loop in a try.
 - Add a catch that catches the exception, but don't do anything with it.Compile and run your program.
2. Now modify the body of the catch so that it prints a useful message (e.g., "Not a letter") followed by the exception. Compile and run the program. Although it's useful to print the exception for debugging, when you're trying to smoothly handle a condition that you don't consider erroneous you often don't want to. In your print statement, replace the exception with the character that created the out of bounds index. Run the program again; much nicer!
3. Print a copy of your source code for the *CountLetters* class and demo your program to the instructor or the lab assistant.

Exercise #2: Throwing Exceptions

File *Factorials.java* contains a program that calls the *factorial* method of the *MathUtils* class to compute the factorials of integers entered by the user. Save these files to your directory and study the code in both, then compile and run *Factorials* to see how it works. Try several positive integers, then try a negative number. You should find that it works for small positive integers (values < 17), but that it returns a large negative value for larger integers and that it always returns 1 for negative integers.

1. Returning 1 as the factorial of any negative integer is not correct—mathematically, the factorial function is not defined for negative integers. To correct this, you could modify your *factorial* method to check if the argument is negative, but then what? The method must return a value, and even if it prints an error message, whatever value is returned could be misconstrued. Instead it should throw an exception indicating that something went wrong so it could not complete its calculation. You could define your own exception class, but there is already an exception appropriate for this situation -- *IllegalArgumentException*, which extends *RuntimeException*. Modify your program as follows:
 - Modify the header of the *factorial()* method to indicate that *factorial* can throw an *IllegalArgumentException*.
 - Modify the body of *factorial* to check the value of the argument and, if it is negative, throw an *IllegalArgumentException*. Note that what you pass to *throw* is actually an instance of the *IllegalArgumentException* class, and that the constructor takes a *String* parameter. Use this parameter to be specific about what the problem is.
 - Compile and run your *Factorials* program after making these changes. Now when you enter a negative number an exception will be thrown, terminating the program. The program ends because the exception is not caught, so it is thrown by the main method, causing a runtime error.
 - Modify the main method in your *Factorials* class to catch the exception thrown by *factorial* and print an appropriate message, but then continue with the loop. Think carefully about where you will need to put the *try* and *catch*.
2. Returning a negative number for values over 16 also is not correct. The problem is arithmetic overflow—the factorial is bigger than can be represented by an int. This can also be thought of as an *IllegalArgumentException*—this *factorial* method is only defined for arguments up to 16. Modify your code in *factorial* to check for an argument over 16 as well as for a negative argument. You should throw an *IllegalArgumentException* in either case, but pass different messages to the constructor so that the problem is clear.
3. Print a copy of your source code for both the *Factorials* and *MathUtils* classes and demo your program to the instructor or the lab assistant.