Name:

CS 122 Lab 9

02/28/2017

Task 1—Placing Exception Handlers

File *ParseInts.java* contains a program that does the following (this is NOT the *ParseInt* class in the java library):

- Prompts for and reads in a line of input
- Uses a second Scanner to take the input line one token at a time and parses an integer from each token as it is extracted.
- Sums the integers.
- Prints the sum.

Save *ParseInts.java* to your directory and compile and run it. If you give it the input

10 20 30 40

it should print

The sum of the integers on the line is 100.

Try some other inputs as well. Now try a line that contains both integers and other values, e.g.,

We have 2 dogs and 1 cat.

You should get a NumberFormatException when it tries to call Integer.parseInt on "We", which is not an integer. One way around this is to put the loop that reads inside a try and catch the NumberFormatException but not do anything with it. This way if it's not an integer it doesn't cause an error; it goes to the exception handler, which does nothing. Do this as follows:

- Modify the program to add a try statement that encompasses the entire while loop. The try and opening { should go before the while, and the catch after the loop body. Catch a NumberFormatException and have an empty body for the catch.

- Compile and run the program and enter a line with mixed integers and other values. You should find that it stops summing at the first non-integer, so the line above will produce a sum of 0, and the line "1 fish 2 fish" will produce a sum of 1. This is because the entire loop is inside the try, so when an exception is thrown the loop is terminated. To make it continue, move the try and catch inside the loop. Now when an exception is thrown, the next statement is the next iteration of the loop, so the entire line is processed. The dogs-and-cats input should now give a sum of 3, as should the fish input.
- Now modify the body of the catch so that it prints a useful message (e.g., "Not a number"). Run the program again. How many times the message is printed out? This number is equal to _____. (fill the black with your answer)

Task 2 —Exceptions Aren't Always Errors

Background knowledge needed: All characters are fundamentally saved as integers.

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. 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.

1. Run CountLetters and enter a phrase, which contains only letters.
2. Run the program again with both letters and nonletters. Insert such as spaces, or other punctuation in between the letters. It should throw an ArrayIndexOutOfBoundsException, because a nonletter will generate an index that is not between 0 and 25.
3. Now let's handle the nonletters. 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 nonletter occurs, the handler will be empty. Modify this method to do this as follows:
    - Put the **body of the first for loop** (not entire for loop, why?) in a try.
    - Add a catch that catches the exception, but don't do anything with it.

   Compile and run your program.

4. 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.

5. In your print statement, replace the exception with the character that created the out of bounds index. Run the program again; much nicer!