CS 249: Assignment 7

Exception Handling and Text I/O

Theory Questions (14%)

1. (2%) Write the line of Java code to throw a new ArithmeticException with the message ”Division by zero”.

2. (2%) Given the code below, what will print out?

```
try {
    throw new ArithmeticException();
}
catch(RuntimeException ex) {
    System.out.println("RUNTIME");
}
catch(Exception ex) {
    System.out.println("EXCEPTION");
}
```

3. (2%) Given the code below, what will print out?

```
try {
    throw new IOException();
}
catch(RuntimeException ex) {
    System.out.println("RUNTIME");
}
catch(Exception ex) {
    System.out.println("EXCEPTION");
}
```

4. (2%) If an exception is an **unchecked** exception, what classes does it inherit from?

5. (2%) Write a method named ”myMethod” that takes no parameters, is public, returns nothing, and **declares an IOException**. The body of this method should be empty. You also do NOT need to define the containing class.

6. (2%) Given the code below, what will print out?

```
try {
      System.out.println("HI");
      return;
}
catch(RuntimeException ex) {
      System.out.println("RUNTIME");
}
catch(Exception ex) {
      System.out.println("EXCEPTION");
}
finally {
            System.out.println("HOW ARE YOU?");
}
```

7. (2%) Creating an instance of a File class will create the corresponding file.

   (a) True

   (b) False

Programming Assignments (86%)

Ensure you are enforcing encapsulation!!!

For this assignment, use a SEPARATE Java file for THE TEST PROGRAM of each requirement (not sub-requirements)! Name these Java files "Assign7_N.java", where N is the requirement number.

Note that you will also create other classes; these should be named "ClassName.java", where ClassName is the name of the public class inside the .java file.

#	*Questions*	
1	**Implement the following classes: Calculator, InvalidExpressionException, and Assign7_1.**	
	InvalidExpressionException extends Exception	
	Has constructor that takes String message and Exception e; call super(message, e) in this constructor.	
	The Calculator class contains one static method: public static double eval(String expr) throws InvalidExpressionException	
	This method only can deal with POSITIVE real numbers.	
	This method only handles 2-number expressions (e.g., 5 + 78).	
	This method only handles the following operators: +, -	
	Do NOT assume there are spaces in the expression!	
	All of the code that follows should be in a try block.	
	Find the first index of any of the above operators.	
	Take two operands and convert them to doubles using Double.parseDouble().	
	Perform the operation requested (addition or subtraction).	
	Catch Exception, but rethrow as an InvalidExpressionException. **DO NOT FORGET TO PASS IN THE OLD EXCEPTION TO InvalidExpressionException's CONSTRUCTOR**.	
	Assign7_1 only has a main() method.	

	All the code that follows should be in a try block.	
	Ask user to enter an expression.	
	Read in line as expression.	
	Use Calculator.eval() to get the answer.	
	Print the answer.	
	Catch InvalidExpressionException and print exception message AND stack trace.	
	INPUT: 3 + 5 Answer is 8.0	
	INPUT: 5.0 + 8.2 Answer is -3.1999999999999993	
	INPUT: 3x + 5 Invalid expression InvalidExpressionException: Invalid expression (Rest of stack trace)	
	INPUT: 6 / 7 Invalid expression InvalidExpressionException: Invalid expression (Rest of stack trace)	
2	**Implement the following classes: WordCountData, WordCounter and Assign7_2.**	
	WordCountData	
	Contains three PRIVATE ints: charCnt, wordCnt, lineCnt	
	Constructor that takes number of characters, number of words, and number of lines; sets data fields accordingly.	
	Include getter methods for all three fields.	
	WordCounter has one method: public static WordCountData count(String path) throws Exception	
	If the String path contains ”http”, then create a Scanner from a URL. Otherwise, create a Scanner from a File.	

	Read through the data and count up the number of characters, the number of words, and the number of lines. Words are separated by whitespace (so don't worry about punctuation.)	
	Close Scanner object.	
	Put this data into a WordCountData object and return it.	
	Assign7_2 only has a main() method.	
	All the code that follows should be in a try block.	
	Ask the user to enter a file path or a URL. Read in next line as *path*.	
	Call WordCounter.count() and get the WordCountData object.	
	Print out the number of characters, the number of words, and the number of lines.	
	Create a PrintWriter that writes to the file "output.txt".	
	Write the *path* string on the first line.	
	Write the number of characters, the number of words, and the number of lines to the file.	
	Close the file.	
	Catch Exception and print "OH NO!" AND the stack trace.	
	INPUT: http://cs.armstrong.edu/liang/data/Lincoln.txt OUTPUT: Number of characters: 1469 Number of words: 277 Number of lines: 16	
	INPUT: gameData.txt OUTPUT: Number of characters: 385 Number of words: 42 Number of lines: 29	

Submission

You will submit the following items as a *.tar or *.zip file:

- A plaintext, Word doc, or PDF with your answers to any theory questions
- Your .java file(s)

Submit this tar/zip file on Blackboard under the appropriate assignment.

Do NOT submit:

- Your .class file(s)
- Your project files

Grading

Below is a list of SOME of the grading penalties:

- Submitting ONLY .class files and NOT .java files
- Sloppy or poor coding style
- Bad coding design principles
- Code that does not compile
- Code that crashes, does not run, or takes a VERY long time to complete
- Using code from ANY source other than the course materials
- Collaboration on code of ANY kind; this is an INDIVIDUAL PROJECT
- Sharing code with other people in this class or using code from this or any other related class
- Output that is incorrect
- Output that is NOT generated by the proper algorithms
- Algorithms/implementations that are incorrect
- Submitting improper files
- Failing to submit ALL required files