Comprehensive Programming Assignment: Phase 1
Phase 1 Due: Start of Class Tuesday, 18 April 2017

In this comprehensive programming assignment, you will combine the various components of the Java programming language that you learned this semester:

- Inheritance & Interfaces & Abstract classes (Ch 9)
- File Processing & Command Line Arguments & Exception Handling (Ch 7)
- Graphical User Interfaces (Ch 10-11)

The program will be based on observations of earthquakes. The earthquakes will be input using a text file in a comma-delimited format:

dateTimeZ, latitude, longitude, magnitude, id, place

The input file must be processed and validated.

- dateTimeZ is a String representing the dateTime of the earthquake.
  (String: 24 characters in the format yyyy-mm-ddT##:##:##.###Z)
- latitude must be a valid latitude value (decimal: -90 to 90)
- longitude must be a valid longitude value (decimal: -180 to 180)
- magnitude represents the magnitude of the earthquake (decimal: -1.0 to 10.0)
- id is the unique id assigned to the earthquake (String)
- place is the location of the earthquake (String)

Based on the values of the earthquake observations, an Earthquake will be created as either NonClassified or Classified. Please see the details of the UML diagram below and the following table, indicating the classification of the earthquake magnitude:

Classification	Minor	Light	Moderate	Strong	Major	Great
Magnitude	*3-3.9*	*4-4.9*	*5-5.9*	*6-6.9*	*7-7.9*	*8 or more*

There will be three filenames ***that will be specified as command line arguments***:

java EarthquakeTester earthquakescsv.txt invalidInput.txt sortedEarthquakes.txt

The first argument specifies the name of the input file, the second argument is the name of an output file *that includes the invalid input with a line number and error message,* and the last argument is an output file name for printing out storms in sorted order. The Earthquake class is to implement the Comparable interface such that eathquakes are sorted such that the highest magnitude earthquakes appear first. A toString method that displays the class name and all of its attributes, *including inherited ones* must be available on each class - see the UML diagram. (Use the approach specified in Special Topic 9.6 Inheritance and the toString Method.)

The EarthquakeTester.java file will process the input file, populating an ArrayList with valid earthquakes, sort the ArrayList, and write the sorted earthquakes to the output file using the toString of the Earthquake.

NOTE: Only programs that successfully compile and execute will be considered for assessment.

- ***Your program must adhere to the UML specification***
- ***Your javadoc comments must be correct and complete and successfully generate an HTML document without warnings.***

REMINDER: THIS IS AN INDIVIDUAL ASSIGNMENT!

	File (.class/.java)	Overview	Description
File Summary	Earthquake.java	Abstract superclass	implements Comparable; contains common earthquake attributes
	NonClassified.java	Subclass of Earthquake	
	Classified.java	Subclass of Earthquake	contains an additional magnitudeClass attribute
	EarthquakeTester.java	main	handles command-line arguments for io

On the due date for Phase 1, you will turn in the following:

1. A hard copy of your .java files (must have name as java comment).
2. A printout of the compilation and execution of your program.
3. An electronic copy of your .java files must be turned in through the assignment facility.

Incremental Development Suggestions:

1. Implement the inheritance hierarchy that illustrates the correctness of your implementation by creating a collection of earthquakes (make sure that you include all types) by hardcoding the values and displaying the content of that collection in sorted order.
2. Add the command line arguments for files and process the data per the specification.Make sure that the program first works on valid input and then extend to handle data validation.