Instructions:

a. Strictly follow the OOP approach
b. Strictly follow the questions and answer them appropriately.
c. Comment your codes as much as you can
d. Upload **only your .java** files in a zip file at the course website, assignment section.
e. Prepare at most 10 mins demo to show the class what you have done.

Deadline:
20th March 2014 11:59pm.
Demo Date:
21st March 2014

Part I : Input and Output.

Problem 0.

Write a program that asks a user for a file name using the command line and prints the number of characters, words, and lines in that file. DO NOT GET INPUT USING THE SCANNER CLASS

Hints:
a. Class names : FileCounter and FileCounterTester
b. three instance variables
c. a constructor initilizing the above instance variables
c. getters and setters for those instance variables
d.

Compiling your program: javac *.java
Running your program: java FileCounterTester filetocount.txt

Problem 1:

Write a program that asks the user for a file name and counts the number of characters, words, and lines in that file.

Then the program asks for the name of the next file. When the user enters a file that doesn't exist, the program prints the total counts of characters, words, and lines in all processed files and exits.

Hints: Same as FileCounter as in Problem 0.
a. In you main method, read the input from the command line or using Scanner
b. In your main method keep invoking your custom counter method that you developed in FileCounter class as long as the new input file is valid.
c. Use try and catch to help you see if file exist or not.

Problem 2:
Write a program that concatenates the contents of several files into one file.

For example,

java CatFiles chapter1.txt chapter2.txt chapter3.txt book.txt

makes a file, book.txt , that contains the contents of the files chapter1.txt , chapter2.txt , and chapter3.txt. The output file is always the last file specified on the command line.

This is intuitive and straight forward.

Problem 3:

Write a program Find that searches all files specified on the command line and prints out all lines containing a reserved word. For example, if you call java

Find ring report.txt address.txt Homework.java
then the program might print
report.txt: has broken up an international ring of DVD bootleggers that
address.txt: Kris Kringle, North Pole
address.txt: Homer Simpson, Springfield
Homework.java: String filename;

The reserved word is always the first command line argument.

Problem 4:

Write a program that replaces each line of a file with its reverse. For example, if you run
java Reverse HelloPrinter.java

then the contents of HelloPrinter.java are changed to

```
retnirPolleH ssalc cilbup
{
        )sgra ][gnirtS(niam diov citats cilbup
        {
                wodniw elosnoc eht ni gniteerg a yalpsiD //
        ;)"!dlroW ,olleH"(nltnirp.tuo.metsyS
}
}
```

Of course, if you run Reverse twice on the same file, you get back the original file.

Problem 5:

Write a program SeparateBabyNames that reads in babynames.txt and produces two files boynames.txt and girlnames.txt , separating the data for the boys and girls. Study the structure of the file and you should see where boys and girls are.

Part II Exception Handling

Problem 6:

Write a BankAccount class to throw an IllegalArgumentException when the account is constructed with

a. negative balance
b. when a negative amount is deposited
c. when an amount that is not between 0 and the current balance is withdrawn.

Write a test program that causes all three exceptions to occur and that catches them all.

Problem 7:

Write a program that asks the user to input a set of floating-point values. When the user enters a value that is not a number, give the user a second chance to enter the value. After two chances, quit reading input. Add all correctly specified values and print the sum when the user is done entering data. Use exception handling to detect improper inputs.

Problem 8:

Write a program that reads in a set of coin descriptions from a file. The input file has the format

coinName1 coinValue1
coinName2 coinValue2
. . .

Add a method
void read(Scanner in) throws FileNotFoundException to the Coin class. Throw an exception if the current line is not properly formatted.

Then implement a method
static ArrayList<Coin> readFile(String filename) throws FileNotFoundException

In the main method, call readFile .

If an exception is thrown, give the user a chance to select another file. If you read all coins successfully, print the total value