**CSC103 Spring 2011**

**Sara Wexler**

**Assignment #1**

**This assignment is to be done by a group of two persons**

Do programming Project 8 &9 ( all in one class) chapter 2 in the book.

Create the class as described in the book.

* Name your class **Quadratic**
* Your class should have **ONLY** the following instance variables:

**double coefA, double coefB, double coefC**

* Your class should have the following methods:
* A **constructor**, and a **default constructor**
* Methods to retrieve the coefficients (called **getCoefA**, **getCoefB**, **getCoefC**)
* A Method to evaluate the quadratic expression of a particular value of x ( called **evalExpression**)
* The **sum** method (as described in the book)
* The **scale** method (as described in the book)
* A method to retrieve the number of real roots ( called **getRootNum**)
* Methods to retrieve the value of the roots **getRootOne**, **getRootTwo**
* An **equals** method
* A **clone** method
* You may have other helper methods that help you to solve the problem.
* Write another class called **QuadTest**
* This class will have **ONLY** **three objects of the class quadratic** (for the first quadratic, the second quadratic, and the quadratic which is the sum of the two.)
* QuadTest has at least the following methods ( they should not be static methods) :
* A method call **intro** will give a description what the program is doing
* A method called **calculations** , this method will to do the required calculation, and to create a clone to the first quadratic, and check if they are aliases, then check if the two quadratic have the same coefficients
* A method called **parse**, which parse the line of input

This method will get the a string of eight numbers per line: three numbers for the three coefficients, one number for the value of x, the value of the number to scale the quadratic expression, and the next three numbers are the coefficients for the second quadratic.

* A method called **output,** to create the output (not a static method) .

**pay attention:**

If the quadratic has just one root, display **only** one root

If the quadratic has infinite number of roots, display: there are infinity number of roots ( do not output: number of roots is 3).

The number of roots = 3 only helps you in your program to determine that there are infinity number of roots

* Create another class called **Lab1** that contain *main*
* The *main* method will read from an input file one line at a time, each line will contain eight numbers. The main method will continue the process until there is no more data to process.

The main method will call **run** method in the **QuadTest** class with the input line to be processed.

* Create a clear output.

* Test your program with the following data:( the data will check for no real roots, every value of X is a real root, one root, two roots )
* Name the file as **inputCoef.txt**

2 8 6 3 2.0 1 1 1

0 0 0 1 4.0 2 2 2

0 0 5 2 5 0 1 2

4 10 7

0 2 6 3 2 1 0 2

2 4 9 1 5 1 2 0

3 6 3 5 2 5 5 5

The first three numbers are for the three coefficients, the next numbers are for the value of x, the value of the number to scale the quadratic expression ( a double value), and the next three numbers are the coefficients for the second quadratic.

If there are not eight tokens per line, Output the line and give an appropriate message like: **Not enough data**, and skip to the next line

Your program should also:

* **Write your names** at each file of your programs and the date
* Write program Description after the Date
* Write **Clear comments**. The comments should be created in a style that will create the javadoc documents
* **Create a javadoc** document
* **Handle errors** in reading (example: not enough coefficients where read)

**Create the following output:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Test # 1

-------------

The first quadratic is:

2\*x\*x +8\*x + 6

The values of the first quadratic expression with x = 3 is : 48

The first quadratic after scaling with R = **2** :

4\*x\*x + 16\*x + 12

Number of roots: 2

Value of the roots:

Value of root1: -3

Value of root2: -1

The Second quadratic is:

1\*x\*x +1\*x + 1

The quadratic which is sum of first and second quadratics:

3\*x\*x +9\*x + 7

A clone of the first quadratic is :

2\*x\*x +8\*x + 6

The first quadratic and the clone are not aliases, but are equal to each other

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Test # 2

**Due Date**: at the beginning of class on  **2/18 (week 4)**

**You will need to demonstrate to me during lab that the project is working properly (it is part of your grade).**

Use this guideline for checking your program over before handing it in:

* Hard copy of **source code** **stapled together**
* **Test cases** stapled together
* You name in a comment at the top of each file
* Due Date in a comment **under** your name
* Program Description after the Date
* Proper indentation throughout your code
* Descriptive identifiers
* Constants defined as needed
* Comments:
* comments at the beginning of program explaining the purpose of the program
* comments in the program to clarify code
* generating javadoc
* if you give poor comments or not enough you will loose 20%
* Properly Labeled CD or Jump drive
* CD or Jump drive contains **Quadratic.java**, **Quadratic.class**, **QuadtTest.java, QuadTest.class**, **lab1.java**, **lab1.class**, **input.txt**, **Output.txt** ( Name of output file)
* File names are as given above
* Formatted Output - Correct spelling and capitalization is consistent. Create an output file: **Output.txt**
* Correct Output

**Hand In:**

In a **closeable** large envelop (no folders) with your **name** clearly stated outside and **assignment #**

* A **labeled disk** with the source code ( the programs with the. java file ) and the .class files.
* A **hard copy** of the source code programs stapled together.
* A **hard Copy** of the **test cases** stapled together with your name marked on top
* You will be required to **show** me that the programs are working

**Grading criteria**

1. - if the project is working good, get correct output, and there are sufficient comments
2. - if the project is working good, but there are some problems with the

code, or some of the output is incorrect

or the comments are not sufficient

2 - if the code compile but does not run

1 - if the does not compile