AP Computer Science Java
Mr. Clausen
Programs 6A, 6B, 6C

Program 6A LastNameFirstNameP6A (Quadratic Formula: 50 points)
(Including complex or irrational roots)

Write a program that begins by explaining the purpose of this program to the user. Don't forget to display a quadratic equation in standard form in this portion of the program ($ax^2 + bx + c = 0$). The purpose of this program is to allow the user to enter the coefficients a, b, and c, and calculate the solutions to the quadratic equation using the Quadratic Formula. The solutions will either be 2 real roots, one real root that is a double root, or 2 complex roots.

DO NOT USE a break statement to escape from any type of loop in this class regardless of what the book says or models.

1) Type comments at the beginning of the program to display your name and other information just like those used for program 2A.

2) Import the package:
   import java.util.Scanner;

3) Declare the class name in the format LastNameFirstNameP6A. Don't forget that the filename needs to be the same when you save your program. This would be a good time to save your program if you haven't done so already, LastNameFirstNameP6A.java.

4) Declare the main method:
   public static void main(String [] args)

5) There are no constants necessary for this program.

6) Instantiate an object of the Scanner class in order to use input from the keyboard. Use the statement:
   Scanner reader = new Scanner(System.in);

7) Declare all of the variables necessary for this program. You will need variables of type double to store: a, b, c, discriminant, realRoot1, realRoot2, doubleRoot, realPart, and imaginaryPart. Later you will need a variable of type String in order to ask if the user wishes to run the program again. **If the user wishes to run the program again, typing "y" is sufficient, please don't make the user type "yes".** Make sure that you use descriptive variable names to create readable, self-documenting code. (It's OK to use a, b, and c as variable names in this case, since this is a well known formula, but add comments to your variable declarations to make sure we know which is the coefficient of the quadratic term, linear term, and constant in the formula.)

8) Type the following comment:
//-------------------------Display My Information-------------------------
Follow this comment with println statements to display your name and period output just like those used for program 2A.

9) Type the following comment:
//-----------------------Display Program Explanation-----------------------
Follow this comment with println statements to display an explanation of the program. Don't forget to display a quadratic equation in standard form in this portion of the program ($ax^2 + bx + c = 0$).

10) For the Input section, type the following comment:
//-----------------------------------Input-----------------------------------
Ask the user to enter the coefficients of the quadratic equation, the values of a, b, and c, each in a separate input statement. Immediately after entering "a", use a "primed while loop" to "error trap" the value of the coefficient "a". Make sure that "a" is not zero. Give the user a message stating that if the value of "a" was zero, you wouldn't have a quadratic equation, but a linear equation. Be sure to ask the user to enter another value for the coefficient "a". After this loop, include an assert statement to guarantee that the value of "a" is not equal to zero.

11) For the Calculations section, type the following comment:
//---------------------------------Calculations---------------------------------
First, calculate the value of the discriminant ($b^2 - 4ac$). Next, use **extended** IF ELSE statements to calculate the solutions depending upon whether the discriminant ($b^2 - 4ac$) is a positive number (2 real solutions), is zero (one real solution multiplicity 2), or a negative number (2 complex roots). Remember, the quadratic formula is:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

You will need to use the Math library for the absolute value (abs), power (pow), and square root (sqrt) methods. You do not need to import the Math package, it is accessible with using any import statements. For b^2 use the power method (pow) instead of multiplying "b" times itself. A summary of methods included in the Math class can be found in Appendix B, page B-5 in your textbook.

12) For the Output section, type the following comment:
//----------------------------------Output----------------------------------
For the output portion of your program, echo back the values of a, b, & c before displaying the solution(s). You will need extended if statements in this section of the program to determine which case of the solution to display. Tell the user whether there are 2 real roots, one real root multiplicity 2, or 2 complex roots in addition to displaying what the roots are. Remember, complex roots are binomials in the form a + bi. Also remember that complex roots occur in conjugate pairs. So if a + bi is one solution, then a - bi is a solution also. The "a" and "b" have nothing to do with the a, b, and c in the original quadratic equation. Remember also, the "a" in the complex root is a real number

(-1b)/(2a) and the "bi" is the imaginary number. You will have to be very careful when calculating this imaginary part and printing the "i" after it. Make this output easy to read and formatted nicely.

13) After you get this program working, add a "do…while" loop around the whole program asking the user if they would like to run the program again. If the answer is **not** "y" or "Y" then quit the program, otherwise run the program again. Since we are using a String variable to determine if the user wishes to run the program again, don't forget that we cannot check equality for strings with the = = symbol in our Boolean expression. You must use the "equals" method to test for equality in strings. ("do … while" is not part of the APCS Java Subset, but good to know. Information about "do … while" can be found in Appendix B, page B-4 in your textbook.)

When you run the program to test it, run it several times using the test data:

a=0 (just to make sure that your error trapping works)
a=1, b=0, and c= - 4
a=1, b= - 4 and c= 4
a=1, b= - 5 and c= 6 and
a= 1, b=1, and c=1

Don't forget to check your answers with paper and pencil to make sure they are correct. If your calculations are not correct, your grade will be reduced accordingly.

Use blank lines to separate each of the program sections listed in all of the steps above. Make sure that everything inside left and right curly brackets are indented three spaces from where the brackets are indented. Use good programming style as described in class. When you are finished with your program, have tested it thoroughly to make sure that your calculations are correct, and are sure that you don't need to make any changes, then save your program in the "T" network mapping, in the Program 6A folder. **Turn in the file named LastNameFirstNameP6A.java, I don't need the ".class" file for this program.**

Program 6B LastNameFirstNameP6B (Guessing Game: 25 points)

Write a program to practice using **public static** methods, while loops, and extended "if else" statements to play a guessing game where the user is given feedback whether their guess is too high or too low. Start by asking the user for the upper limit (the largest number used in the guessing game). Next, generate a random number between 1 and the upper limit using **Math.random()**, not by importing and using the random class like the textbook teaches (refer to the PowerPoint presentation for this). Next, the user guesses the mystery number while you are giving them feedback whether their guess is too high or too low. When the user guesses the correct number, tell them they are correct, and tell them how many guesses they used to guess the random number.

DO NOT USE a break statement to escape from any type of loop in this class regardless of what the book says or models.

1) Type comments at the beginning of the program to display your name and other information just like those used for program 2A.

2) Import the package:
   import java.util.Scanner;
   Do not import the Random class.

3) Declare the class name in the format LastNameFirstNameP6B. Don't forget that the filename needs to be the same when you save your program. This would be a good time to save your program if you haven't done so already, LastNameFirstNameP6B.java.

4) Declare the main method:
   public static void main(String [] args)

5) Declare the constant, LOW = 1.

6) "Stub" out the main method. Add your right curly bracket to "close" the method for now. We will come back and finish the main method after we write the other three methods for this program.

7) **Remember that all of the methods we use in this program need to be "public static" methods and need to be declared after the right curly bracket that ends the main method and before the right curly bracket that ends the class, LastNameFirstNameP6B.**

8) Write the void method displayMyInfo(). This void method should have println statements to display your name and period output just like those used for program 2A.

9) Write the void method checkGuess. Pass this method two integers: the user's guess and the random number that they are trying to guess. Give the user feedback whether their guess is too high or too low, or is correct using extended if statements.

10) Write the method getNumber which takes three parameters: reader, message, and upperLimit and returns the number that the user entered. Use a primed while loop to make sure that the user's guess is less than the upper limit and greater than zero.

11) Back in the main method, instantiate an object of the Scanner class in order to use input from the keyboard. Use the statement:
    Scanner reader = new Scanner(System.in);

12) Declare all of the variables necessary for this program. You will need variables of type **int** to store: upperLimit, numberOfGuesses, randomNumber, and userGuess. You will need a variable of type String named message to pass to the method that gets numbers

from the user. Make sure that you use descriptive variable names to create readable, self-documenting code.

13) Type the following comment:
//-------------------------Display My Information-------------------------
Follow this comment with a call to the method displayMyInfo();

14) For the Input section, type the following comment:
//-----------------------------------Input-----------------------------------
Ask the user to enter the upper limit. We are going to do this by calling a "method" and sending it the reader object, a string literal message, and a numeric literal of 1000000. Store the value returned by the method in the variable upperLimit.

15) For the Calculations section, type the following comment:
//------------------------Calculations and Output----------------------
In the main method, generate the random number between 1 and the upperLimit that the user chose using **Math.random()**. Use a while loop that repeats while the user's guess is not equal to the random number. Inside this loop ask for the user's guess by calling the same method that we used to get the upper limit. Send the reader object, a string literal message, and the variable upperLimit. Store the value returned by this method in the variable userGuess. Next, inside this loop call the method checkGuess. After the while loop is finished, tell the user how many guesses it took to guess the mystery number.

Use blank lines to separate each of the program sections listed in all of the steps above. Make sure that everything inside left and right curly brackets are indented three spaces from where the brackets are indented. Use good programming style as described in class. When you are finished with your program, have tested it thoroughly to make sure that everything is correct, and are sure that you don't need to make any changes, then save your program in the "T" network mapping, in the Program 6B folder. **Turn in the file named LastNameFirstNameP6B.java, I don't need the ".class" file for this program.**

Program 6C LastNameFirstNameP6C (Student Grades: 30 points)

This program will require two java files. You will create a class to work with Student Grades, and a client program to test your class file. To see a model for this type of program, double click on My Computer, then these folders, APCSFileJava, 3rd Edition Textbook Source Codes, Chapter05, Case Study, and open the files: Student.java and StudentApp.java.
DO NOT USE a break statement to escape from any type of loop in this class regardless of what the book says or models.

14) Begin with the class file (server file) Student Grades.

15) Type comments at the beginning of the program to display your name and other information just like those used for program 2A. **Don't forget to change the information from program 2A to include the new information for this program.**

16) There are no packages to import for this file.

17) Declare the class name in the format:
public class **LastNameFirstNameStudentGrades**.
Don't forget that the filename needs to be the same when you save your program. This would be a good time to save your program if you haven't done so already, **LastNameFirstNameStudentGrades.java**.

18) The class file does not have a main method

19) There are no constants for the class file.

20) You do not instantiate an object of the Scanner class or of any other classes in the Student Grades.

21) Declare all of the instance variables necessary for this class. You will need **integers** to store the numericGrade and numberOfScores, A **String** variable for the student's name. Don't forget to declare these variables as **private**. Make sure that you use descriptive variable names to create readable, self-documenting code.

22) Type the following comment:
//-------------------------Constructors-------------------------
Don't forget that constructors are **public** methods, do NOT have a return type, and **MUST have the same name as the name of the class**. You will need two constructors. A default constructor that initializes all of the integers to zero and all of the strings to the empty string, a parameterized constructor that allows the client to instantiate and initialize the name to whatever they wish while initializing the integers to zeros.

23) Type the following comment:
//---------------------------------Mutators---------------------------------
Don't forget that Mutator methods usually are **void** methods and therefore do NOT use a return statement. They will also need at least one parameter in order to change the values of the private variables. In this section, you will create the mutator methods setName, and addNumericGrade. The method addNumericGrade will need to use the accumulator algorithm to keep a running total of the grades sent to it and increment the number of grades.

24) Type the following comment:
//---------------------------------Accessors---------------------------------
Remember that Accessor methods usually do not have parameters, and only "access" the private variables. Therefore, they do need a return type and must include at least one return statement in order to return the value of the private variables to the client program.

In this section, you will create the accessor methods, getName, getAverage, and getLetterGrade. Also include the accessor method toString, which will display the name, numeric average, and the letter grade.
The method, getAverage should use a local variable for the average (type **double**), check to see if the number of scores is zero. If the number of scores is zero, the method should return -1 (negative one), otherwise, it should calculate and return the average (don't forget to type cast integer variables to double). The method getLetterGrade should use local variables for letterGrade and average and call getAverage, and use extended if statements to assign letter grades (cutoff scores are 90 for an A, 80 for a B, 70 for a C, 60 for a D, less than 60 is an F). The final else in this method should assign the string "No grade" to the variable letterGrade.

25) If you haven't done so already, compile this file, and debug all errors so that your class file is ready for the client program.

Use blank lines to separate each of the program sections listed in all of the steps above. Now it's time to create the client program, **LastNameFirstNameP6C.java**.

26) Type comments at the beginning of the program to display your name and other information just like those used for program 2A. **Don't forget to change the information from program 2A to include the new information for this program.**

27) Import the package:
import java.util.Scanner;

28) Declare the class name in the format **LastNameFirstNameP6C**. Don't forget that the filename needs to be the same when you save your program. This would be a good time to save your program if you haven't done so already, **LastNameFirstNameP6C.java**.

29) Declare the main method:
public static void main(String [] args)

30) Create a constant named SENTINAL and assign it a value of -99.

31) Instantiate an object of the Scanner class in order to use input from the keyboard. In addition, instantiate one object of the **StudentGrades** class using the default constructor.

32) Declare all of the variables necessary for this program. You will need an **integer** for score, and a string for name. Make sure that you use descriptive variable names to create readable, self-documenting code.

33) Type the following comment:
//--------------------------Display My Information--------------------------
Write the void method displayMyInfo(). This void method should have println statements to display your name and period output just like those used for program 2A. **Don't forget to change the information from program 2A to include the new**

information for this program. Remember that all of the methods we use in this client program need to be "public static" methods and need to be declared after the right curly bracket that ends the main method and before the right curly bracket that ends the class.

34) For the Input section, type the following comment:
    //----------Input----------
    Ask the user to enter the name, use the mutator method to change the name object according to what the user entered. In order to practice calling our accessor methods, after you ask the user for the name, echo this value back on the monitor screen using the accessor, getName.

35) For the Calculations section, type the following comment:
    //----------Calculations----------
    Create a "while loop" that repeats while the user does not enter the SENTINAL -99. Inside the loop ask the user to enter a score or the SENTINAL to quit. If the score is not the SENTINAL call the method addNumericGrade.

36) For the Output section, type the following comment:
    //----------Output----------
    For the output portion of your program, print the student object which will invoke the toString method and display the student's name, average, and letter grade (if there is one).

Use blank lines to separate each of the program sections listed in all of the steps above. When you are finished with your program, have tested it thoroughly to make sure that your calculations are correct, and are sure that you don't need to make any changes, then save your program in the "T" network mapping, in the Program 6C folder.

After you are finished, you will need to turn in these data files:

1) **LastNameFirstName StudentGrades.java**
2) **LastNameFirstNameP6C.java**