**PROGRAM COMMENTS**  
  
Programming exercises are key component of learning any programming language. For this course, there are four programming assignments that are arranged from the simplest case to more complex tasks as you progress through the course.

The textbook comes with a CD-ROM that includes Java. Students may install and use this version of Java for all course assignments. The CD-ROM also may include an integrated development environment (IDE). You do not need to use the IDE to complete these assignments, however, you are welcome to learn and use the IDE if you wish.

Downloading the latest version of Java from the Sun web site is a excellent idea, provided you have a fast Internet connection. Sun also provides a downloadable IDE called NetBeans. This IDE has a level of support from Sun that the IDE on the CD-ROM may not.

For your assignment to be graded, you must identify yourself by including Java comment block at the beginning of your program source code. This comment block must include your name, the date, the assignment number, and your student identification number. You also must provide a description and list of key variables and parameters used.

An example of the minimum Java comment block is:  
  
/\*\*  
\* Name: Who you are.  
\* Date: When you did the work.  
\* Assignment: Number  
\* Student Number: AAXXXXXXX  
\* Description: This program does a, b, and c.  
\*/  
  
You must name your Java class files (your source code) by concatenating the assignment number (A1 through A4) and your student identification number. For example, if your student number were AB7654321, the your Java class file would be named: A1AB7654321...A7AB7654321

If any assignment fails to compile cleanly, it must be reworked. It will be embarrassing if any of your programs were to fail to compile because the filename and class name did not agree.

In naming your variables, it is essential that you use an acceptable, consistent style and reasonable variable names. Variable names like x, y, and z are to be avoided. If your program needs to compute a total, then an appropriate variable for storing the value would be total or totValue or another meaningful identifier.

Using the defacto standard style for naming and indenting is a good idea, but not required.  What is required is that the style you use is consistent, easily readable, and structured.  If you don't have a personal style, follow the the style you see throughout the textbook.

In addition, when you declare an important variable, place an inline comment using //, followed by a brief explanation of the variable's use in the program. For example:

int idxR = 0; // Row Index  
  
float inpLen = 0.0; // Input Length

Finally, your .java file must be in plain text suitable for editing using Notepad. It must not have embedded tabs or graphic characters. If you edit your file using an operating system that does not included a carriage return and line feed at the end of each line (Linux), then you must state that in each submission so the instructor knows to convert it -- the instructor looks at every submission, line by line, in Notepad!

Programming Assignment 1  
  
Write a console program that prompts the user to input the length and width of a rectangle and then prints the rectangle's area and perimeter. Note that a "console program" is one that is runnable from the MS-DOS command prompt.

**Programming Assignment 2**  
  
Write a console program that:

1. Uses a while loop to perform the following steps:
   1. Prompt the user to input two integers: *firstNum* and *secondNum* where *secondNum* is at least 10 greater than *firstNum,* both numbers are positive integers, and *secondNum* is less than 1000.
   2. Verify that the user entered acceptable numbers, and if not, provide error feedback and prompt them again.
   3. Output all results to a file in the same directory as the program, placing an appropriate label between each section of output. Note that your program must be able to run repeatedly overwriting the file from the previous run.
   4. Output all odd numbers between firstNum and secondNum inclusive, one number per line.
   5. Output the sum of all numbers between firstNum and secondNum exclusive.
2. Uses a for loop to perform the following steps:
   1. Continue writing to the same file as before.
   2. Write a label as before.
   3. Output all numbers from *secondNum* to *firstNum* in a single line with commas separating the numbers.
3. Write the date and time as the last line in the file in the format yyyy-mm-dd hh:mm:ss.

Notes:

1. Inclusive means that the firstNum and secondNum are output if appropriate.
2. Exclusive means that the firstNum and secondNum are not output.
3. The instructor will attempt to crash your program by entering numbers you don't expect -- if you program thros an exception you will be marked down!
4. While this looks like a daunting task, it is actually very simple -- just think it through!

**Programming Assignment 3**

Write a program that consists of three classes.  The first class will be the actual program.

The second class will simply convert a string to lower case.

The third class will have three methods:  
  
public static String trimmed(String str)  
  
and  
  
public static String trimmed(String str, int len)  
  
and  
  
public static String squeeze(String str)  
  
The 1st trimmed method will return str without leading or trailing whitespace and will return an empty string if str is null.  
  
The 2nd trimmed method will return the first len characters after trimming the string. The 2nd method must make use of the 1st method.  
  
The squeeze method will replace all sequences of 2 spaces with 1 space.  
  
The program will read a file containing the data below (cut and paste it into notepad and save it), and will display each line:

1. as entered but trimmed and squeezed.
2. as entered but trimmed, squeezed, and shortened to 10 characters
3. as entered but trimmed, squeezed, converted to lower case, and shortened to 20 characters.

Data (copy after this line down to (not including) the line that says end data, you will have 5 lines):  
     This is a test, this is only a test!  
  
This    test    is    a    test   of    your   Java   programming   skills!  
xyz  
  ABCDEFGHIJKLMONPQRSTUVWXYZ1234567890-=\_+                         !  
end data  
  
Grading Notes:

1. You can do this program however you like (console, GUI, AWT, Swing, Applet, etc.).
2. If you use an inner classes for the second and third classes, you will receive up to 5 extra points, and if the third class is anonymous, you will receive up to five additional points.
3. If you make the program graphical, you will receive up to 10 extra points (total won't exceed 100).
4. You will lose points for poor formatting, poor commenting, not following instructions, program that don't work, etc.
5. In any case, your total cannot exceed 100 points.
6. Since this is the 3rd assignment, you are expected to get this right before you turn it in.  It will be graded much more rigorously than the earlier assignments and in most cases, you won't get a second chance!  If something is confusing in the instructions, ask before submitting!

**Programming Assignment 4**  
  
Write a console program that repeatedly prompts the user to enter data until they type *done* (any case, Upper, Lower, or Mixed). As they enter the data, assign it to a two-dimension array where the first dimension contains exactly what they type and the second dimension contains the first non-whitespace character of what they type, in lowercase. If they enter a blank line, it is acceptable to skip that line. When they type *done*, do these steps:

1. display: As Entered
2. for each entry in the array, display the index of the first dimension, the second dimension's value, and the first dimension's value on a single line separated by : (colon).
3. display: Bubble Sorted
4. using a bubble sort, for each entry in the array, display the original index of the first dimension, the second dimension's value, and the first dimension's value on a single line separated by : (colon) sorted by the second dimension.
5. display: Selection Sorted
6. using a selection sort, for each entry in the array, display the original index of the first dimension, the second dimension's value, and the first dimension's value on a single line separated by : (colon) sorted by the first dimension.

Grading Notes:

1. Please read the instructions carefully! Using the wrong sort or dimension will hurt your grade!
2. The instructor will attempt to crash your program -- an exception will cost you 10 points -- don't let exceptions happen!
3. If you ensure that duplicate values are sorted with the lower original array index first, you will get 5 extra points (final score not over 100).
4. You can use the sort methods from the book, but you will get 5 extra points if you do them as inner classes and FULLY comment them (final score not over 100).
5. Since this is the 4th assignment, you are expected to get this right before you turn it in. It will be graded rigorously and in most cases, you won't get a second chance! If something is confusing in the instructions, ask before submitting! If the instructor does give you a second chance, 10 points will be deducted.

Example:

*User types:*   
apple  
Apple  
Zone  
apple  
done

*You display:*As entered  
0:a:apple  
1:a:Apple  
2:z:Zone  
3:a:apple  
Bubble Sorted  
0:a:apple  
1:a:Apple  
3:a:apple  
2:z:Zone  
Selection Sorted  
1:a:Apple  
0:a:apple  
3:a:apple  
2:z:Zone