CSC 210 PROGRAMMING STYLE/DOCUMENTATION GUIDELINES

1. PROGRAM HEADER

Each file must begin with a *program header*. This is a comment block that has the following format (fill in the parts in square brackets with the appropriate information):

```
/*************************************************************
 *
 * File: [name of file]
 * By: [name of author(s)]
 * Date: [date last revised]
 *
 * Description: [short description on what program does, what user
 *               enters, what result(s) are displayed]
 *
 *************************************************************/
```

Note that this illustrates another way to indicate a comment block in Java. (Earlier, we saw that all text after // in the same line make up a comment.) Similarly, all lines in between /* and */ are comments. Hence:

```
/*
   a comment
   more comments
   even more comments
   etc etc
*/
```

2. VARIABLE NAMES AND COMMENTS

Names of major variables should be *descriptive* of their role or usage in the program. For example, for a variable that contains the number of students in a class, numStudents is usually a better name than n.

Each major variable should have a comment describing its role in the program, when it is declared. For example:

```
   int numStudents; // number of students in class
```

3. USING WHITESPACES AND INDENTATION

Blank lines should be used to separate major blocks or sections of your program. This makes your program more readable.

Mark each level of nesting of statements with a separate level of indentation. You should indent with at least 2 spaces or a tab for each level; some prefer more spaces. Be consistent about how many spaces you use. Consider this code fragment:

```
      if (score >= 70) {   // process passing score
        System.out.println("Passed");
        pass = pass + 1;
      }
      else { // process failing score
        System.out.println("Failed");
        fail = fail + 1;
      } // end if (score >= 70)
```

Note that the statements for printing "Passed" and adding 1 to pass both belong in the block after if (score >= 70); hence, they are both indented a few spaces to the right of the if statement.

Similarly, the statements for printing "Failed" and adding 1 to fail both belong in the block after the else; they are both indented a few spaces to the right of the else.

A more complex fragment:

```
while (score >= 0) {
   if (score >= 70) {   // process passing score
      System.out.println("Passed");
      pass = pass + 1;
   }
   else { // process failing score
      System.out.println("Failed");
      fail = fail + 1;
   } // end if (score >= 70)

   // get next score
   System.out.println("Enter next score: ");
   score = input.nextInt();
} // end while (score >= 0)
```

The leftmost (outermost) level is the while loop. The next level is the body of the while loop, which consists of the if-else block. The level after that are the printf and increment statements within the if-block and the else-block.

Note that the close curly brace of a block should be lined up with the statement that begins that block. Note from the same example:

```
while (score >= 0) {

// some stuff omitted

} // end while (score >= 0)
```

There are two common options for placing the open curly brace. I usually put it on the same line as the statement it follows. It's also possible to place it on the following line by itself, indented at the same level as the statement it follows:

```
while (score >= 0)
{
   if (score >= 70)  // process passing score
   {
      System.out.println("Passed");
      pass = pass + 1;
   }
   else        // process failing score
   {
      System.out.println("Failed");
      fail = fail + 1;
   } // end if (score >= 90)

// get next score
   System.out.println( "Enter next score: ";
score = input.nextInt();

} // end while (score >= 0)
```

Either way is fine. Just be consistent.

Do not put more than one statement on the same line. Statements that are too long should be spread over multiple lines in a readable way. For example:

```
x = (- b + sqrt( pow(b, 2.0) – 4 * a * c) ) )
     / (2. * a);
```

4. INSERTING COMMENTS IN SOURCE CODE

Comments should be inserted to describe the tasks being undertaken. For example, in the if-else block:

```
if (score >= 70)  // process passing score
   {
      System.out.println("Passed");
      pass = pass + 1;
   }
else        // process failing score
   {
      System.out.println("Failed");
      fail = fail + 1;
   } // end if (score >= 90)
```

Or:

```
   // get next score
   System.out.println(“Enter next score: “);
score = input.nextInt();
```

Different programmers prefer different densities of comments. In general, there should be enough comments for a reader to follow the main actions in the program relatively easily.

Note also the use of comments to match close curly braces with open curly braces. This improves readability in large blocks. We see that the last closed curly brace is for the while loop, and the second to last close curly brace ends the if-else block:

```
while (score >= 0) {
   if (score >= 70) {   // process passing score


   }
   else { // process failing score


   } // end if (score >= 70)


} // end while (score >= 0)
```

5. NETBEANS SHORTCUTS

In NetBeans, please try this shortcut: **ALT + SHIFT + F**