(1) Every .java FILE must include a header comment with this format, as the first information in the file:

```
/* AUTHOR: Put your name here
   INSTRUCTOR: Dr. Sharad Sharma
   COURSE: CTEC 396-101: Java Programming
   SEMESTER: Fall 2007
   ASSIGNMENT NO: X
   PROJECT: Program number or topic
   DUE DATE: Date specified
   SUBMISSION DATE: when you handed it in
   SUMMARY
            What the code does and what the user should know to understand the program.
   INPUT
            Keyboard, file, arguments, etc.
  OUTPUT
            Tell what shows up on the screen, what is written to any files, etc.
  CLASS HIERARCHY
            Include the full hierarchy starting from the Object class for every class you define in the file.
            Object - Component - Container - Panel - Applet - YourAppletClassName
            Object - YourApplicationClassName
 ASSUMPTIONS
            What do you expect the user to know, to do, to NOT do, etc.
*/
```

(2) All code, comments, and output should fit on the page; i.e., they should not wrap around or be truncated. Long statements that would run off the page should be split in a logical manner and should be indented at least 3 spaces.

- Tabs are set every three spaces.
- Variable and method names are lowercase, with occasional upperCase characters in the middle.
- Class names start with an Uppercase letter
- Constant names are UPPERCASE, with an occasional UNDER_SCORE.
- Braces must line up vertically.
- Every method, except for main and overridden library methods, must have a comment.
- No continue or break is allowed.
- All non-final variables must be private.

(3) Each statement begins on a new line.

(4) Identifiers must be meaningful - variable, constant, method, class names
    Exceptions: (1) Names that Sun uses (String [] args, Graphics g, etc.)
        (2) Simple for loop index (i,j, etc.) declared in the loop
            for (int i = 0; i < maximum; i++)

(5) Comment all variables, constants, methods, classes, lines of code, etc. whose meaning would not be obvious to a beginning programmer.
Names like temp and button3 are not meaningful. (Often, a better name for a variable can be found in the comment. Then you don't need the comment any more!)

(6) Blank lines separate logical blocks of code.

(7) Comment logical blocks of code (like chapter titles in a book).

(8) Left braces { must appear on a new line right under the first character of the previous line. Right braces} must line up with the corresponding left brace and have a comment telling what is ending. Also opening and closing braces must line up vertically: For example:

```
class Hello
{
   public static void main (String [] args)
   {
      System.out.println("hello world");
   }  // main
}  // Hello
```

(9) Indent 3 spaces inside each pair of {} or each control statement.
(Use a fixed width font (Monaco, Courier, etc.) so the spaces show up.)

```
while (x < 8)
   x += 4;

while (y > 2)
{
   x++;
   y += x + z;
   z = y + x;
} // while
```

(10) Put a space before and after all binary operators.

(11) In classes, put the variables first, constructors second, other methods third, inner classes fourth, and methods with no code last.

(12) Do not define two variables on the same line:

```
int dimes = 0, nickels = 0; // Don't
```

Instead, use two separate definitions:

```
int dimes = 0; // OK
int nickels = 0;
```

(13) Do not use magic numbers! A magic number is a numeric constant embedded in code, without a constant definition. Any number except -1, 0, 1, and 2 is considered magic:

```
if (p.getX() < 300) // Don't
```

Use final variables instead:

```
final double WINDOW_WIDTH = 300;
. . .
if (p.getX() < WINDOW_WIDTH) // OK
```

(14) Use blank lines freely to separate parts of a method that are logically distinct. Use a blank space around every binary operator:

```
x1 = (-b - Math.sqrt(b * b - 4 * a * c)) / (2 * a);  // Good

x1=(-b-Math.sqrt(b*b-4*a*c))/(2*a);//Bad
```