Statement-comment

Just as the sentences of an essay are grouped in paragraphs, so the sequence of statements of the body of a method should be grouped into logical units. Often, the clarity of the program is enhanced by preceding a logical unit by a comment that explains what it does. This comment serves as the specification for the logical unit; it should say precisely what the logical unit does.

The comment for such a logical unit is called a statement-comment. It should be written as a command to do something. Here is an example.

```java
// Truthify x >= y by swapping x and y if needed.
if (x < y) {
    int tmp = x;
    x = y;
    y = tmp;
}
```

A statement-comment should explain what the group of statements does, not how it does it. Thus, it serves the same purpose as the specification of a method: it allows one to skip the reading of the statements of the logical unit and just read the comment. With suitable statement-comments in the body of a method, one can read the method at several "levels of abstraction", which helps one scan a program quickly to find a section of current interest, much like one scans section and subsection headings in an article or book. But this purpose is served only if statement-comments are precise.

Statement comments must be complete. The comment

```java
// Test for valid input
```

is inadequate. What happens if the input is valid? What if it isn't — is an error message written or is some flag set? Without this information, one must read the statements for which this statement-comment is a specification, and the whole purpose of the statement comment is lost.

Use of blank lines

Place a blank line after the implementation of each statement-comment. Consider this example:

```java
// Eliminate whitespace from the beginning and end of t
while (t.length() != 0 && isWhitespace(t.charAt(0))) {
    t = t.substring(1);
}
// If t is empty, print an error message and return
if (t.length() == 0) {
    ...  
    return;
}

if (containsCapitals(t)) {
    ...  
}
// Store the French translation of t in tFrench
...
```

This sequence consists of 4 statements:

1. eliminate the whitespace ...
2. print a message and return if t is empty,
3. do something if t contains capitals, and
4. store the French translation.

Three of these are statement-comments, and one can easily see, because of the blank lines, where their implementations end.