Side effects

A side effect occurs when evaluation of an expression changes a value. For example, a side effect occurs if evaluation of the following loop condition, which is a call on function f, changes some variable:

```
while (f(...)) {...}
```

Generally, side effects should be avoided. This means, for example, that an assignment statement like

```
b= ...;
```

should change no variable other than b.

A side-effect that changes something else could be disastrous, unless it is very well documented and the programmer is aware of what is happening. In fact, the theory of proving programs correct tells us that formal proofs of correctness when side effects may occur are *far* more complex than when they may not.

Standard paradigms

One does see some standard paradigms that use side effects. For example, below is a loop that reads and prints lines of a BufferedReader br; the loop condition has the side effect of assigning to variable *line*.

```
String line;
while ((line= br.readLine()) != null) {
   System.out.println("line is " + line);
}
```

However, we would never write the loop like this, preferring to write it as follows, in a way that has no side effects, even though it is longer that the one above.

```
String line= br.readLine();
// invariant: all lines read in, except line, have been printed.
while (line != null) {
   System.out.println("line is " + line);

   line= br.readLine();
}
```

Benevolent side effects

There is one class of side effect that is OK. A *benevolent* side effect is one that changes the state but not in a way that the user cannot observe. A benevolent side effect can be used to make a program more efficient. Here is an example.

> Consider implementing a class that maintains a set of questions and answers. When someone asks a question, the corresponding answer is given. The questions and answers are implemented in an array, and one has to search the list in linear fashion to find a question.
>
> It is better if more frequently asked questions are at the front of the list, so they are more efficiently found. This can be achieved by the following: Whenever a question is asked, move it up one position in the array.
>
> This optimization changes the *representation* of the set of questions and answers, but not the set itself; the user cannot notice the change, except perhaps in the speed with which questions are answered. This is a benevolent side effect.