

Coding & Debugging

Version of 001224

Write small simple programs to test constructs you haven't used before or about which you feel shaky. It's easier to locate your misunderstanding in a short program than in a long one.

When you're debugging a program, pay attention only to the first error. The others may simply be side-effects of that first error.

When you're writing a program, keep the specification in front of you. How else can you be sure that you're solving the correct problem?

Don't attempt to write out all the code at once and then try to debug it.

Always build on existing code which compiles and runs.

Grow programs incrementally. If you extend a working program and it doesn't work anymore, then you can localize the error.

Keep working checkpoint programs as you move along. If you seriously wound your code, you can retreat to a prior version which worked.

Implement a large program in well-defined steps. Examine the problem and determine how you can implement the required capabilities in an incremental manner.

Learn how to document errors:

- they must be repeatable;
- the code should be short;
- collect hardcopy of the program, the compile, and the test run.

Human memory is fallible, and computers are unforgiving, so no one will believe you if you don't have documentation. Experts can't help you if you come in with a folktale — "I tried that, but it gave me an error." You must be able to replicate the error.

If you encounter an error that you can't understand, then keep the code which generated the error and find out what was wrong. You may produce a working program by trying another approach, but if you don't know why you got the error, then there's a gap in your knowledge of Java.