Concept

Testing is the activity of finding out whether a piece of code (a method, class, or program) produces the intended behavior.

program is correct cannot, in general, be automated. It is so hard, in fact, that most software that is commercially sold is known to contain a significant number of bugs.

Thus, it is essential for a competent software engineer to learn how to deal with correctness, and how to reduce the number of errors in a class.

In this chapter, we shall discuss a variety of activities that are related to improving the correctness of a program. These include testing, debugging, and writing for maintainability.

Testing is an activity that is concerned with finding out whether a segment of code contains any errors. Testing well is not easy, and there is much to think about when testing a program.

Concept

Debugging is the attempt to pinpoint and fix the source of an error.

Debugging comes after testing. If tests have shown that an error is present, we use debugging techniques to find out exactly where the error is and how to fix it. There can be a significant amount of work between knowing that an error exists, finding the cause, and fixing it.

Writing for maintainability is maybe the most fundamental topic. It is about trying to write code in such a way that errors are avoided in the first place and, if they still slip in, that they can be found as easily as possible. Code style and commenting are part of it, as are the code quality principles that we have discussed in the previous chapter. Ideally, code should be easy to understand so that the original programmer avoids introducing errors and a maintenance programmer can easily find possible errors.

In practice, this is not always simple. But there are big differences between few and many errors, and also between the effort it takes to debug well-written code and not-so-well-written code.

9.2 Testing and debugging

Testing and debugging are crucial skills in software development. You will often need to check your programs for errors, then locate the source of those errors when they occur. In addition, you might also be responsible for testing other people's programs or modifying them. In the latter case, the debugging task is closely related to the process of understanding someone else's code, and there is a lot of overlap in the techniques you might use to do both. In the sections that follow, we shall investigate the following testing and debugging techniques:

- manual unit testing within BlueJ
- test automation
- manual walkthroughs
- print statements
- debuggers

We shall look at the first two testing techniques in the context of some classes that you might have written for yourself, and the remaining debugging techniques in the context of understanding someone else's source code.