# Developing & Testing Application

Testing is an important aspect of developing software. You are constantly testing your program as you write source code, compile and run. Having a clear testing strategy can greatly add to the quality of your software. Some aspects of your code will be tested by you, but other aspects will be tested by others. Having other users, especially the ones that the software is aimed at, testing your program, will help you remove errors and increase functionality of your software. Bugs are errors in the code of a computer program. To debug a program, the programmer reads any error messages that Greenfoot provides. Then, the programmer corrects those errors in the syntax. Testing will then move onto the logic in the code. Remember that successfully compiling software does not mean that it is bug free. It only means that the syntax is correct.

Using good code indentation will greatly improve the readability of your code. This makes locating errors like those listed above a lot easier and less time consuming. The auto layout will indent code in-between brackets. This demonstrates good program layout techniques to make your code more readable. You could write all your code on one line and Greenfoot wouldn't mind, but trying to find errors in your code becomes very difficult. Also simply trying to read how the code works becomes a huge onerous task.

Planning your game before you start coding will save you lots of time. Some simple games will require very little planning, but as the game complexity increases then so does the need for proper planning techniques. The identification of the objects required in software will help you determine the number of subclasses required under the Actor class. Although we will typically have one level of classes under Actor, in larger programs we may have multiple levels where we have Actor > subclass where classes share common fields and methods. Collecting of the information required will better help you plan a solution.Defining the actions of an object will give you the basis of the methods and fields required in your classes.

Testing can be planned before any coding has started. This has the benefit of having the programmers think about what is going to be tested as they start to code a solution. A good design allows you to think how all of your objects are going to act and interact. It is easy when writing code that does not follow a design to get caught up with only the current problem and not the bigger picture. This can lead to poorly coded solutions.

A textual storyboard is complete when you could give it to any programmer and they would produce very similar results to others. If they all created completely different solutions, then it was the storyboard that was incomplete. You can test your storyboard by giving it to three people and then have them explain back to you how the game works. If there are big differences in their explanations, then your storyboard requires additional information. Testing the program in small stages allows you to pinpoint errors easier as you have a better idea on where they probably reside. If you wrote your whole program before testing it would take a lot more work to find where these errors might be located.