Developing and Testing an Application

What Will I Learn?

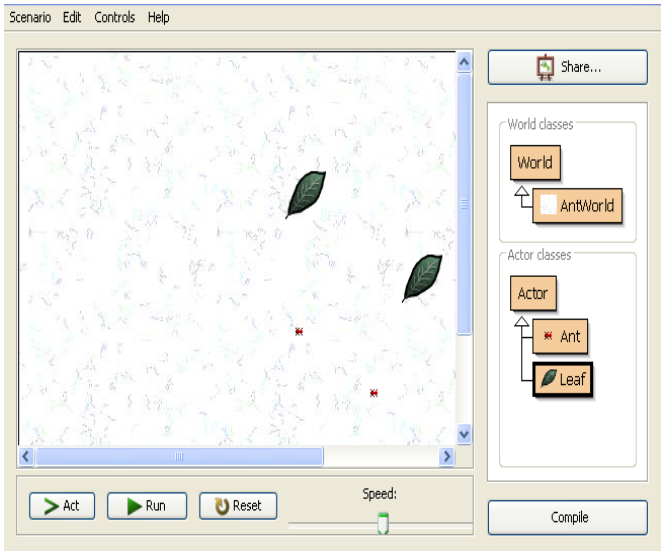
Objectives

- Demonstrate program testing strategies
- Recognize phases for developing a software application

Why Learn It?

Purpose

When developing games and other programs, the developer must test his/her program before sending it to production. Testing helps a programmer find the errors in a program or game before it is distributed to users. Testing is an important stage of game development.



Program Testing Strategies

Programmers will test a program many times during the course of its development to ensure it works properly.

Testing guidelines:

- Test your program frequently after each method, or a sequence of methods, are written.
- Compile the program to ensure it is error-free.
- If errors appear, correct them.
- Run the program to see how the method(s) make the object move.
- Continue to add methods and adjust as necessary.

Compilation and Debugging

Every character in source code counts. One missing or incorrect character could cause your program to fail.

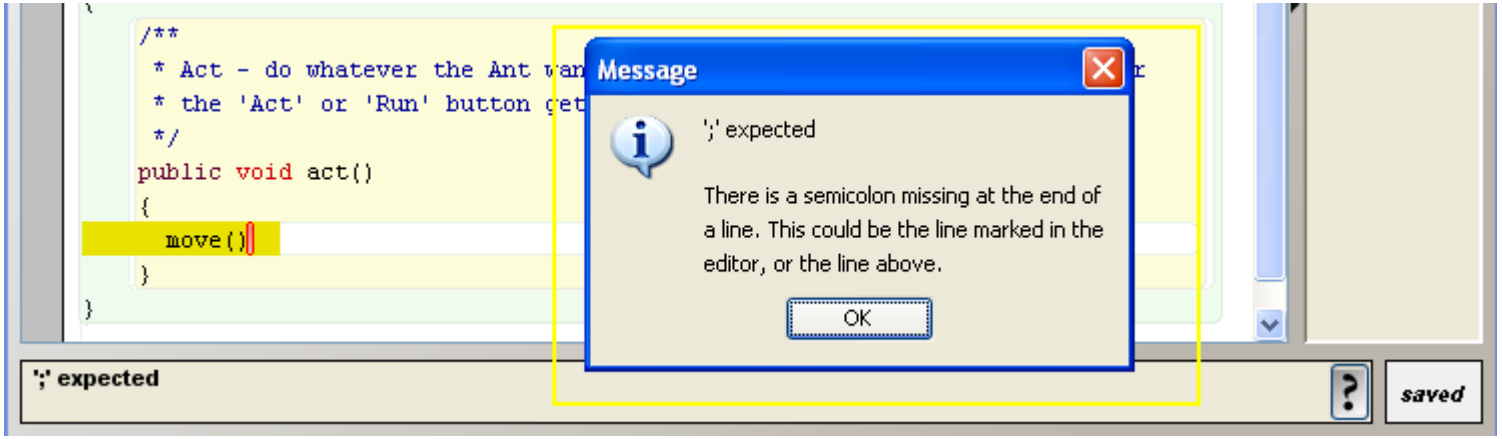
In Greenfoot, compilation highlights bugs and what is required to correct them. This will help you develop good programming techniques.

Bugs are errors in the syntax of a computer program. To debug a program, the programmer compiles the program and reads any error messages that Greenfoot provides. Then, the programmer corrects errors in the syntax and re-compiles the program.

Steps to Debug Your Program

Follow these steps to debug your program:

1. Click Compile to compile the code you have written.
2. If there are no errors, the message “Class compiled – no syntax errors” displays.
3. If there are errors, the incorrect syntax is highlighted and a message attempts to explain the error.
4. Click the question mark icon to display additional information about the error.



Keys to Recognizing Java Syntax Errors

Keys to recognizing Java syntax errors are:

- Locate the beginning and end of a method.
- Ensure all beginning { and ending } braces exist.
- Ensure all open (and closed) parentheses exist.
- Ensure all lines of code end with a semi-colon.
- Review the spelling of a class name and ensure all proper capitalization is in place.
- Ensure all string quotes are double “ not single ‘.
- Examine all comment lines to ensure proper syntax is used for both single and multiple comment lines.
- Review all dot notation (i.e., System.out.println).
- Ensure all characters that look similar are correct (the number 1 versus the letter l).

Phases to Develop an Application

Developing a game in Greenfoot follows the same steps as developing a software application.

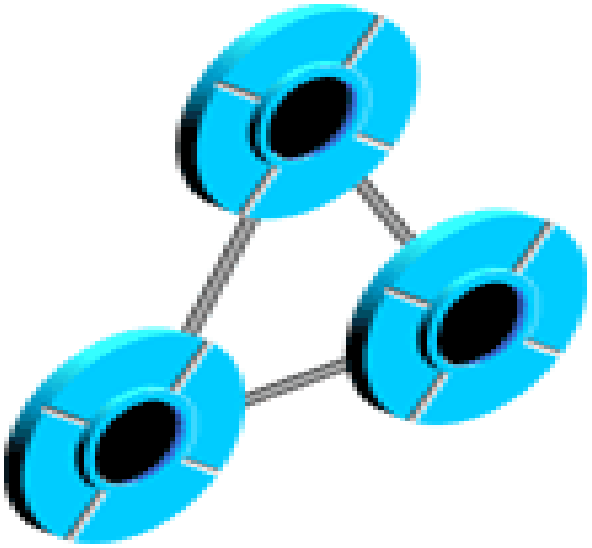
Follow these steps to develop a software application:

1. Analyze the problem to solve or task to perform.
2. Design the solution, which is a game in Greenfoot.
3. Develop the game in Greenfoot.
4. Test the game to ensure it works and meets the requirements of your analysis and design.

Analysis Phase

In the analysis phase, determine what problem the game will solve, or the task it will perform, using object oriented analysis.

In object oriented analysis, Java technology programmers analyze a problem and then create objects to build a system, or more specifically, to solve the problem.



Analysis Tasks

The analysis phase includes the following tasks:

1. Identify a problem to solve.
2. Write a brief statement of scope that states the type of solution (game) that will solve the problem.
3. Gather the target audience's requirements. These are the people who most likely will play your game.
4. Identify and describe objects in the game.
    1. Physical objects (e.g., Car, person, tree)
    2. Conceptual (“non-physical”) objects (e.g., timer that counts down time remaining)
    3. Attributes of all objects, such as color, size, name, and shape
    4. Operations that the objects perform (e.g.,move, turn, eat)

Analysis Example

Analysis Item	Description
Problem domain	I want to create a timed game to teach students how to count.
Game player's (customer) requirements	Easy for 7-8 year olds to play. User must have a keyboard and mouse and be able to count to 10.
Objects	1 object named “Duke”, the Java mascot; 10 objects for Duke to eat, and a background world that is a light color.
Objects' Operations	Duke: Move, turn, and eat code Code: Sit still and be eaten when Duke lands on them Timer: Count down from 10-1 seconds, then end the game Background: do nothing

Analysis Pre- and Post-conditions

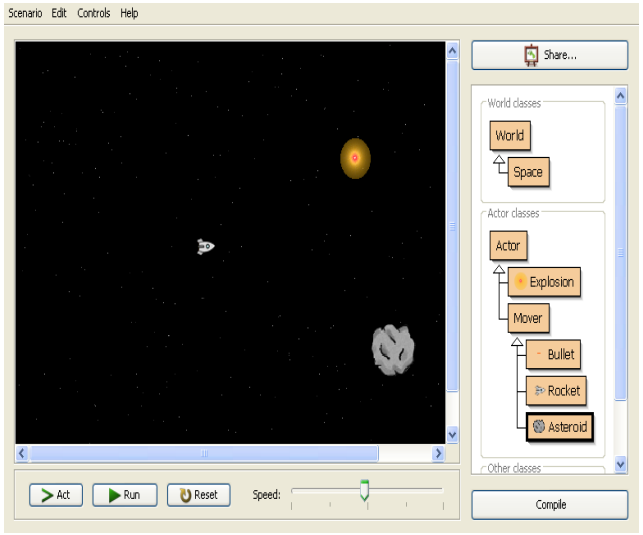
In the analysis phase capture information to support testing for:

1. Pre- and post game conditions.
   For example, variable initialized values versus final values after program execution.
2. Anticipated run times and comparison run rates given a set of conditions.
   For example run rates can vary based on computer memory size variance.
3. Expected results for statement execution counts.
   For example, a loop counter of 3 will produce 3 new variables.
4. Numerical representations and limitations.
   For example, the maximum value for an integer.

Design Phase

In Greenfoot, the solution you design will be in the form of a game that your target audience can play.

You will design your solution in a textual storyboard that maps out the algorithms—or methods—that objects will perform when the player uses keyboard commands or mouse clicks.



Textual Storyboard Example

This textual storyboard describes a simple game where Duke, the Java mascot, has 10 seconds to eat as many pieces of Java code as possible.

When Run button is clicked, Duke can move

Player uses arrow keys on keyboard to control Duke's movements

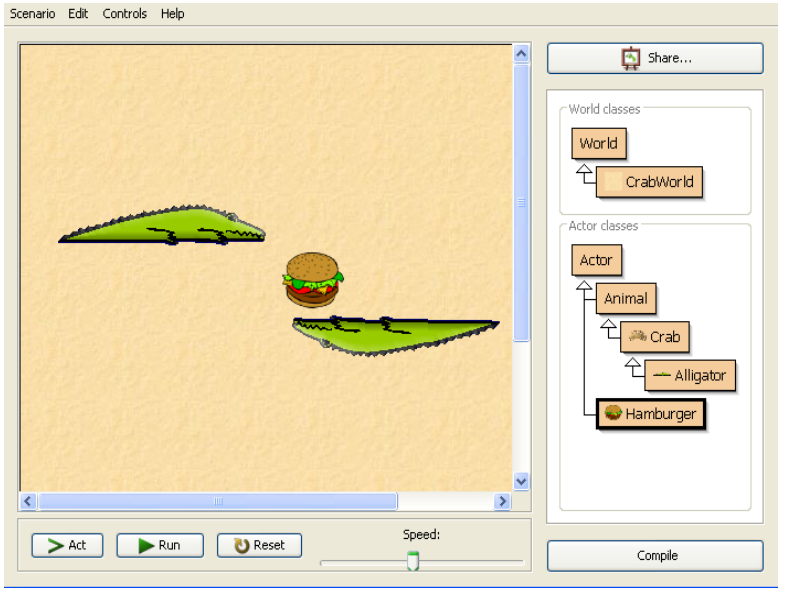
When Duke is in the same square as Code, Duke eats Code

After 10 seconds, the game ends and displays how many pieces of Code Duke ate

Development Phase

After you finalize your storyboard, develop your game in Greenfoot.

Refer to your storyboard to determine the methods you need to program.



Testing Phase

After you write a section of program code, compile the code and test the program by clicking the Run button in the environment.

- Watch how the objects move, and revise the source code as necessary.
- For quality assurance purposes:
  - Have other people test your game and give you feedback.
  - Seek people who fit the target audience for your game.
- Write test plans that:
  - Examine pre- and post conditions
  - Compare run time rates and execution counts
  - Test numerical representations and limitations

Terminology

Key terms used in this lesson included:

Bugs

Documentation

Summary

In this lesson, you learned how to:

- Demonstrate program testing strategies
- Recognize phases for developing a software application

Practice

The exercises for this lesson cover the following topics:

- Understanding documentation and testing
- Concept summary and terminology review