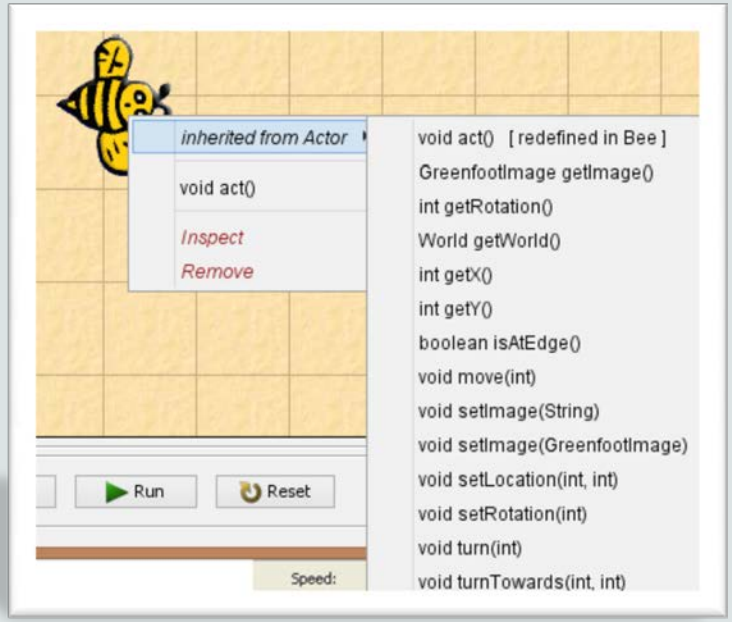

ORACLE
ACADEMY

Java Fundamentals

3-2
Methods, Variables, and Parameters



ORACLE ACADEMY

Objectives

This lesson covers the following objectives:

- Define parameters and how they are used in methods
- Understand inheritance
- Describe properties of an object
- Examine the purpose of a variable
- Discuss programming concepts and define terminology

Methods Example

- In order to complete a task, such as math homework, there are several subtasks:
  - Student completes the math homework.
  - Student goes to school.
  - Student submits the homework to their teacher.
- Because of learned experiences in school, combined with pre-programmed abilities (such as thinking), the student is capable of completing this task.

Methods

- In programming, each object has a set of operations (or tasks) it can perform.
- Programmers write a program to tell an object how and when to perform tasks, such as:
  - Command an object to perform an action.
  - Ask an object a question to learn more about what it does.

Methods are a set of operations or tasks that instances of a class can perform. When a method is invoked, it will perform the operation or task specified in the source code.

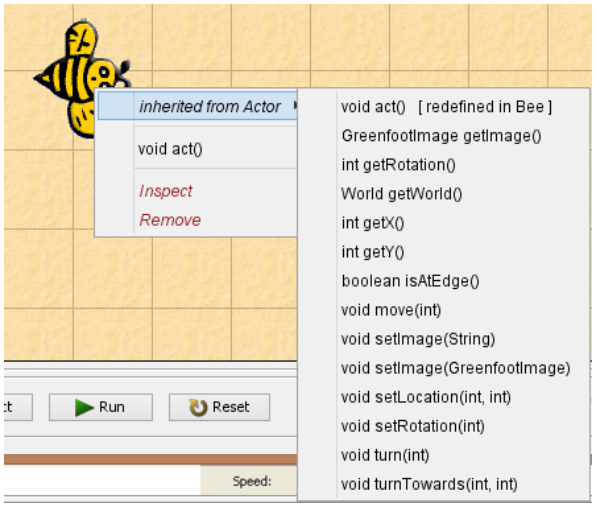
Inheritance

- Greenfoot objects inherit the methods and properties of their class and superclass.
- For example, an Alligator instance would inherit the methods of the Actor superclass and Alligator class.

Inheritance means that each subclass inherits its methods from its superclass.

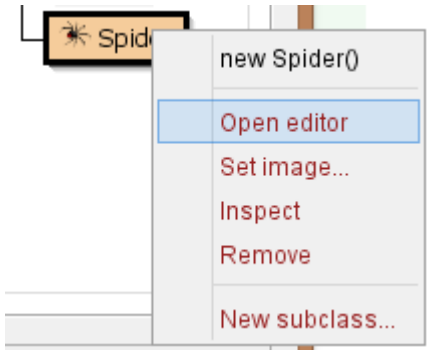
View Inherited Methods in Object Menu

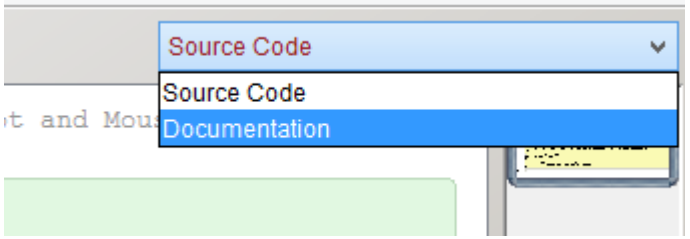
- The object menu displays all of the methods that the instance inherits from its class and superclass.
  - Right click on the instance to display the menu.
  - Inherited From Actor displays a list of the methods that the class inherits from the Actor superclass.



Steps to View Inherited Methods in the Code Editor

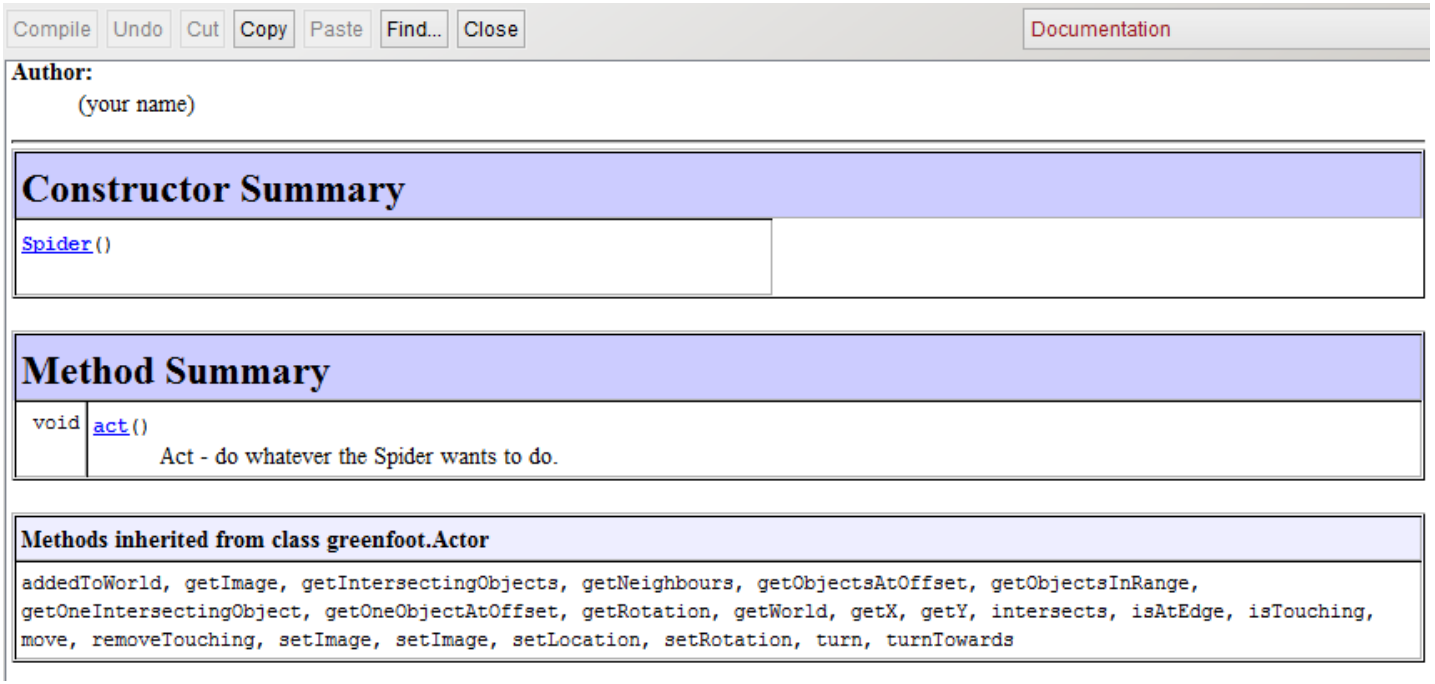
- Right click on a class (this example is the Spider).
- Click Open Editor.
- In the Code editor, select Documentation from the drop-down menu at the top right.
- Scroll down to the Method Summary.





Method Summary

- The method summary displays the class's inherited methods.



Method Components

- A method has several components that describe the operations or tasks it performs.
  - Return type: Specifies the type of data that the method returns.
  - Method name: Describes what the method does.
  - Parameter list: Information that goes into the method call.
- Examples of methods:

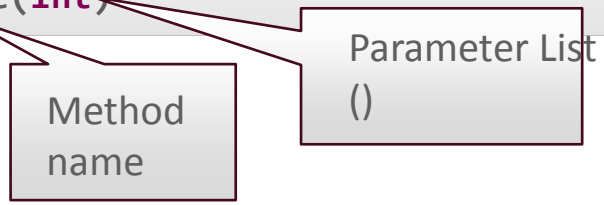
```
void move(3)
int getX()
```

A method call commands the instance to perform an operation or task in Greenfoot. Read the method to understand what operation or task is to be performed.

Method Signature

- The method signature describes the intentions of the method.
- It includes the following components:
  - Method name
  - Parameter list

```
void move(int)
```



Return Types

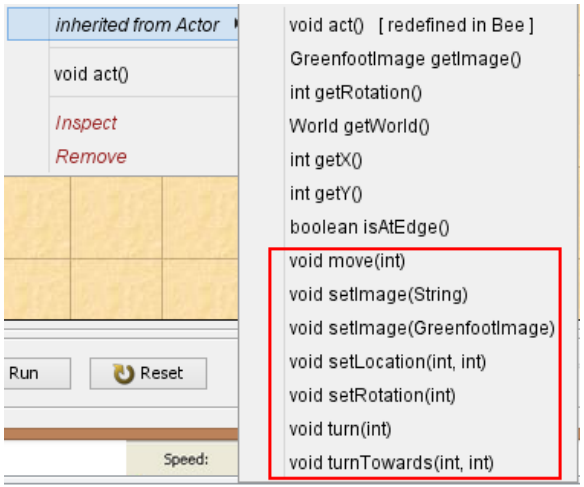
- The return type is the word at the beginning of the method that indicates what type of information a method call will return.
- There are two types of return types:
  - Void: No data return - Issues a command to the object.
  - Non-void: Returns Data - Asks the object a question.

```
void move(int)
```

Return type

Methods with Void Return Types

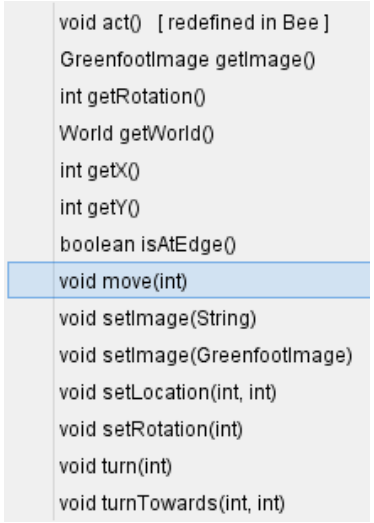
- Methods with void return types issue a command that carries out an action.
  - Includes the word "void".
  - Does not return information about the object.
  - Is used to make the object do something.



Invoking Methods with Void Return Types

- You will invoke methods with void return types:
  - To precisely position objects in your initial scenario (the starting point of the game).
  - To command objects to perform actions in your game.

Example of selecting
The move method

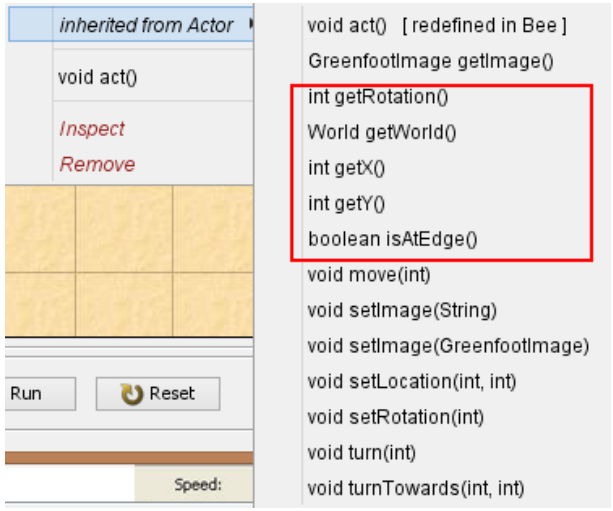


Ask Objects Questions

- As a programmer, you will ask objects questions by invoking methods with non-void return types to learn what an object can do, or has done in the past.
- For example, in school, teachers ask students questions to see if the students comprehend the material they learned in class that day.
- Students respond with answers to let the teachers know how much they learned.

Methods with Non-void Return Types

- Methods with non-void return types ask the object a question.
  - The method signature does not include the word "void".
  - The method returns information about the object, but does not change or move it.



Examples of Non-Void Return Types

- Integer (displayed as int)
  - Refers to whole numbers
  - Asks the object: How many?
- Boolean
  - Returns a true or false value.
  - Types of questions it may ask an object:
    - Are you touching another object?
    - Are you at the edge of the world?

Method Parameters

- Parameters provide methods with additional data to make an object perform a task, when information is required to invoke the method.
- Parameters are defined by two components:
  - Parameter type
  - Parameter name

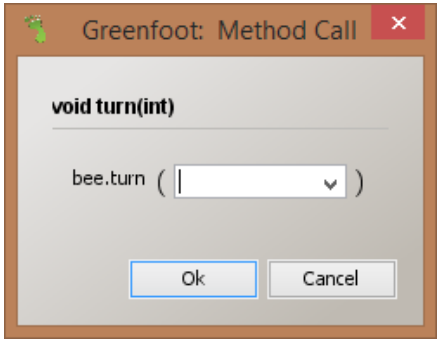
Parameters are used to command objects to move, or to tell objects what type of response is expected when we ask an object a question.

Examples of Method Parameters

- Integer (int): Enter or display numerical values.
- Boolean: Display true or false values.
- String : Enter or display text values.

Method Parameter Lists

- Method parameter lists indicate whether the method requires additional information to be invoked, and what type of information.
  - Parameter lists display as data within parentheses.
  - They typically have two states:
    - Empty: No data expected to invoke the method (getRotation() method).
    - Non-empty: Have data and expect one or more parameters to invoke the method
      - i.e. turn(int) method



Object Properties

- Object properties describe the instance's appearance and abilities, such as:
  - Size
  - Color
  - Range of movements
- Properties can be viewed and modified in the class's source code.
- Each time you create a new instance of an Actor such as Bee it has its own properties.

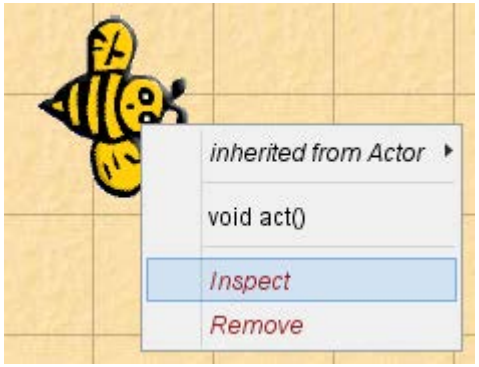
Variables

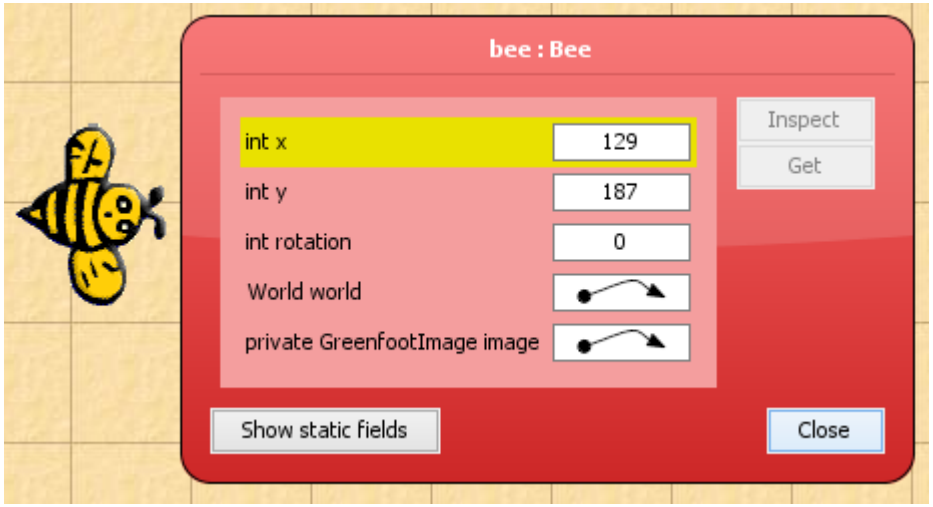
- A variable, or field, allows the instance to store information to use immediately or later.
- For example, object properties are variables that store information about the instance, such as its position in the world.

Instance variables are the memory that belong to the instance of the class. That memory can be saved and accessed later as long as the instance exists.

View Instance Variables

- Right click on an actor instance, then click Inspect to view the instance's variables in the object inspector.



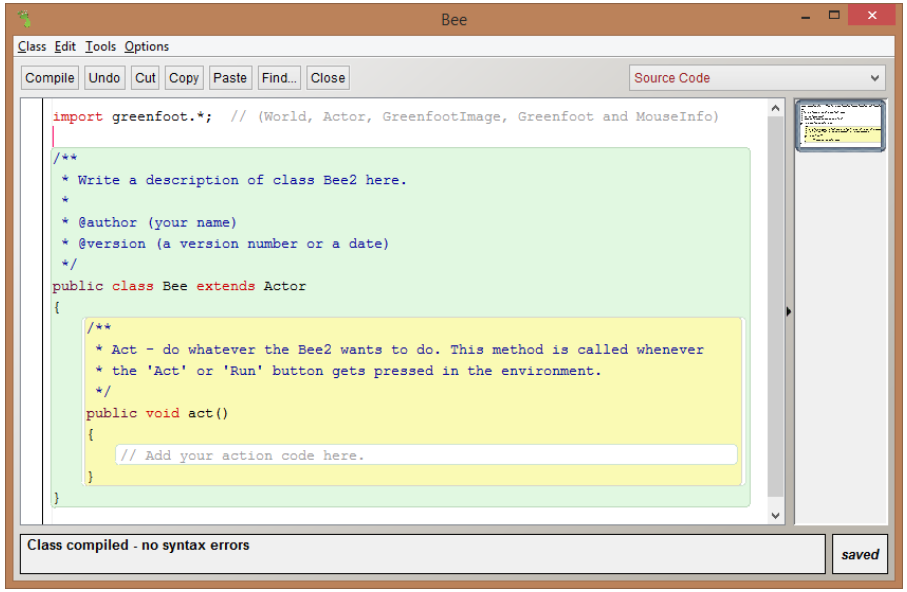


Programming Syntax

- The source code specifies all of the properties and characteristics of a class and its objects.
- Write source code (also known as syntax) in the class's Code editor to command objects in your scenario to act.

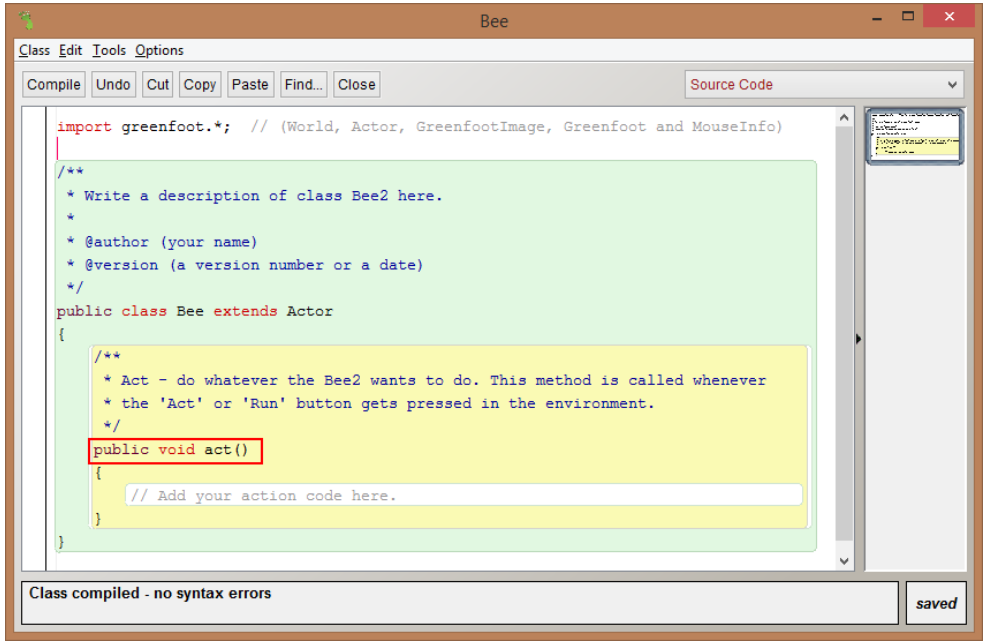
Display Class Source Code

- From the world, right-click on a class and select Open Editor to display the Code editor.
- The source code displayed defines what objects in the class can do.



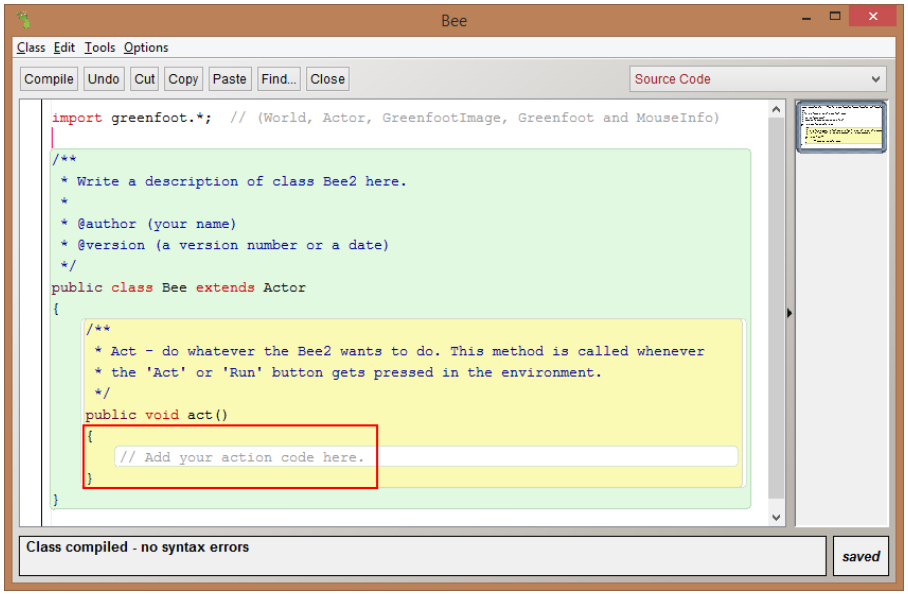
Act Method

- Whenever the Act or Run execution controls are clicked in the environment, the object will repeatedly do what is programmed in the act method.



Body of Act Method

- The curly brackets and content within them are the body of the method.
- Here you can write code to instruct instances of the class to act when the Act or Run buttons are clicked.



Act Method Example

- Call the move and turn methods in the Act method to make instances of the class move and turn.
- Methods must be written correctly with no typos, missing characters, or incorrect capitalization, or the source code wont compile.

```
import greenfoot.*;  // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/**
 * Write a description of class Bee here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Bee extends Actor
{
    /**
     * Act - do whatever the Bee wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public void act()
    {
        move(1);
        turn(15);
    }
}
```

Invoke Methods in Act Method

- To invoke methods in the Act method, write them in sequence as follows:
  - Name of the method in lowercase characters.
  - Parentheses, with parameter list if required.
  - Semicolon, to end the statement.

```
public class Bee extends Actor
{
    /**
     * Act - do whatever the Bee wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public void act()
    {
        move(1);
        turn(15);
    }
}
```

Debugging Process in Greenfoot

- Incorrectly written or missing characters in your source code will trigger error messages.
- When the Compile button is clicked, the compiler checks for syntax errors in the source code.
- If an error is found, an error message is displayed and must be corrected by the programmer before the program will work.
  - Greenfoot provides these error messages so its easier to correct mistakes and learn from them.

Debugging is the process of finding and removing bugs—or errors—in a computer program.

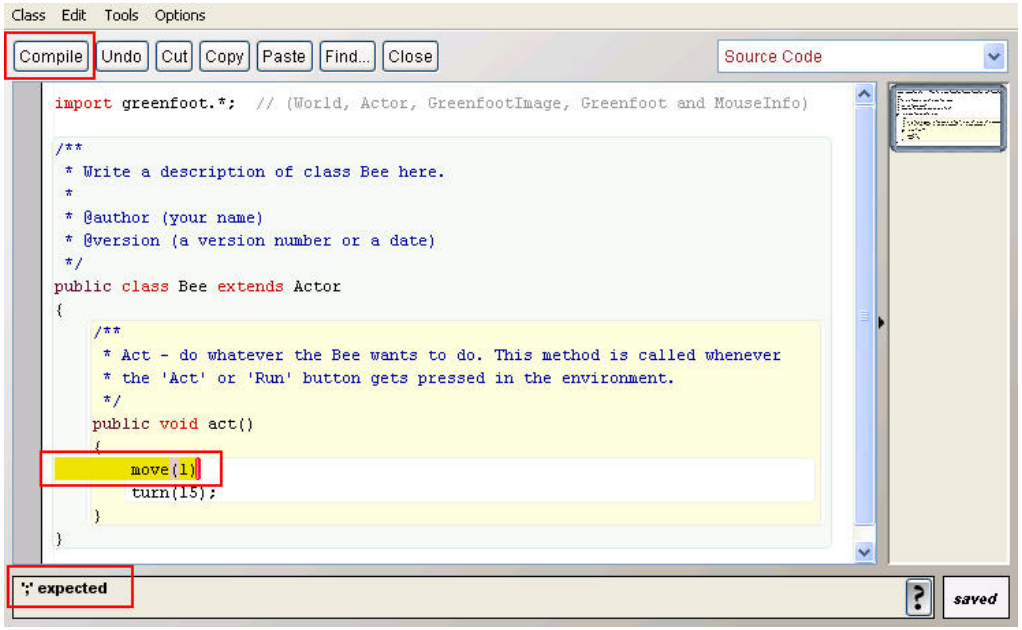
Syntax Error Example

- The move method is missing a semicolon.

```
public class Bee extends Actor
{
    /**
     * Act - do whatever the Bee wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public void act()
    {
        move(1)
        turn(15);
    }
}
```

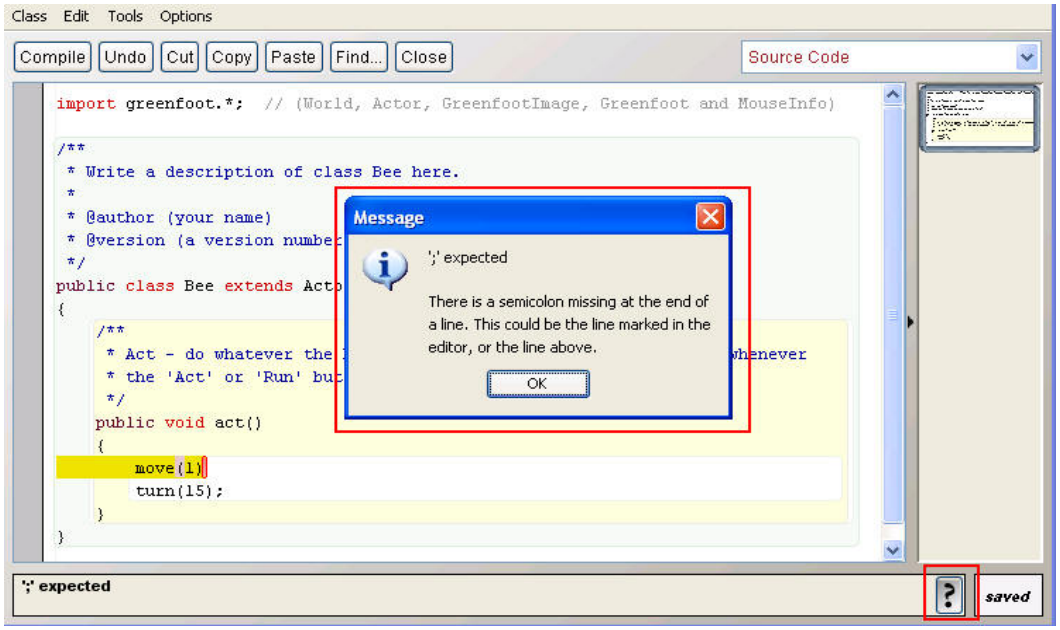
Error Message Example

- After the Compile button is clicked, an error message appears at the bottom of the screen and the incorrect code is highlighted.



Error Explanations

- Click the question mark (?) to display a more detailed error message that attempts to explain the error.
- Not all error messages will be easy to understand.



Terminology

Key terms used in this lesson included:

- Debug
- Inheritance
- Instance variable
- Method
- Method call
- Parameter
- Return type
- Method signature
- Variable

Summary

In this lesson, you should have learned how to:

- Define parameters and how they are used in methods
- Understand inheritance
- Describe properties of an object
- Examine the purpose of a variable
- Discuss programming concepts and define terminology


ORACLE
ACADEMY