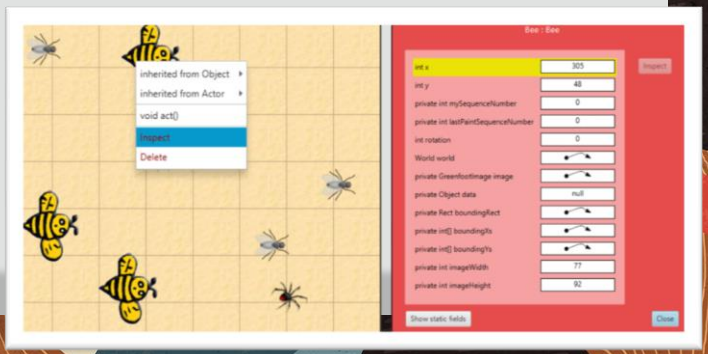
ORACLE

Academy

Java Fundamentals

3-2

Methods, Variables, and Parameters



ORACLE
Academy

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

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

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

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

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

When learning programming one of the most difficult tasks is to break down problems into simple steps for the computer to understand. Most people can carry out tasks without much problems, but asking them to detail to someone else how they did this in step by step notes can be more difficult that it sounds. This takes practice, but with experience it becomes easier.

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.

ORACLE Academy

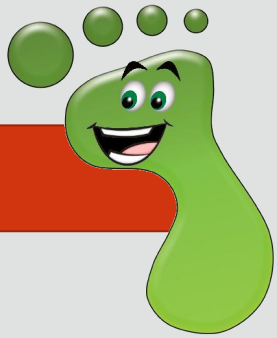
Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

An object can be anything that we wish to model. It can be something real like a car, or something in a game like a teleport (not real at time of writing!). We will learn about the benefits of modelling things as objects as we go through the course, but most of these ideas you will already be familiar with.

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.

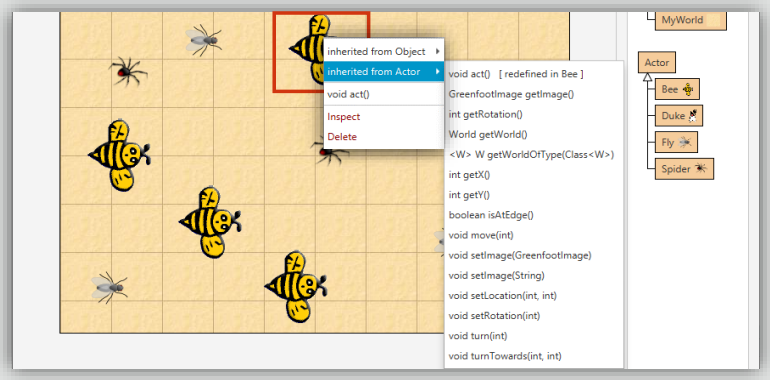


Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

When talking about the Alligator class we can say the Alligator class is a subclass of Actor. We can also say Actor is a superclass of Alligator. The Actor class will also have a superclass. So when talking about the Actor class it is a superclass to Alligator and a subclass to its parent. So a Class can be both a superclass and a subclass at the same time.

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

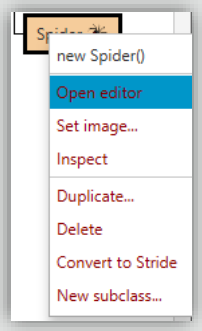


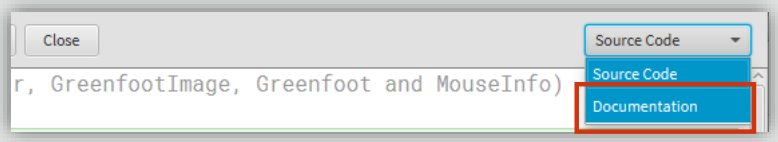
Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Notice that the first pop up menu shows all the methods that are defined in the Bee class. In this case just the act() method. The second drop down states that the act that was defined in the actor superclass has been redefined by the Bee subclass. We also see all of the methods that we have inherited from the Actor superclass.

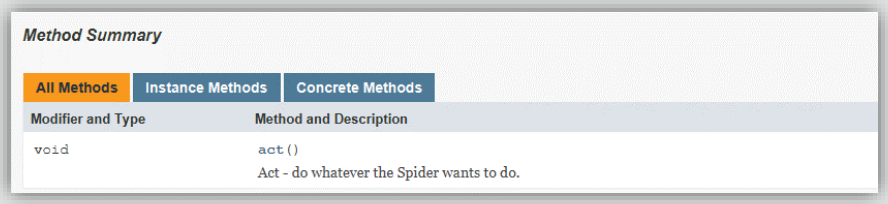
Steps to View Inherited Methods in the Code Editor

- 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

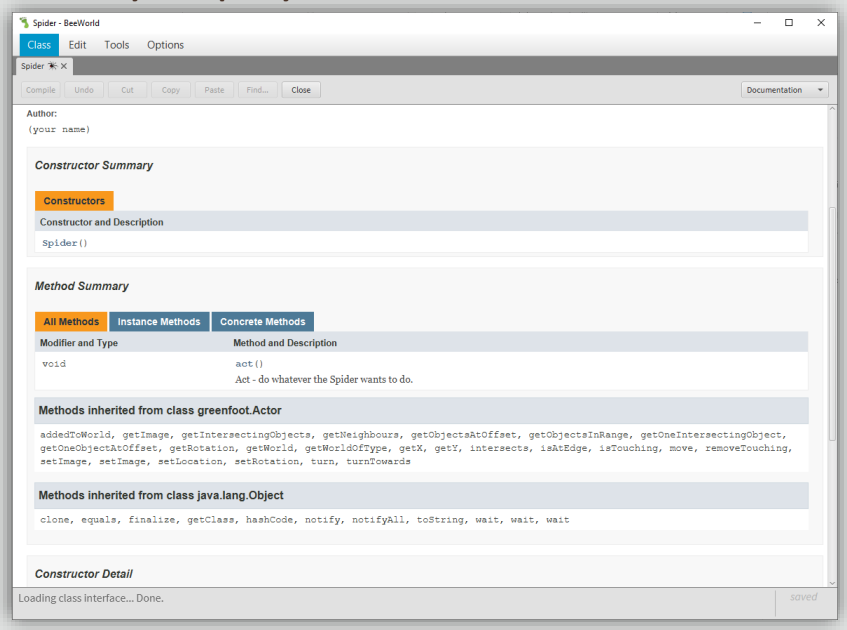


Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Before you select documentation notice in the class you have selected that we only have one method defined – act()

Method Summary

- The method summary displays the class's inherited methods



ORACLE Academy

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

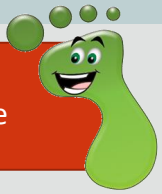
If you look at the source code page you can see there are comments throughout. Giving these more meaningful text will mean your documentation is more complete.

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.



ORACLE Academy

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

The example void move(3) instructs the actor to move 3 places in its current direction. The method does not return a value to the calling program line so we use the word void. Void means that no value is returned from the method.
In the second example we call the method getX(). This returns the current X co-ordinate of the actor which is a whole number. In Java we represent whole numbers as integers, so this method has a return type of int.

Method Signature

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

```
void move(int)
```

Method name

Parameter List ()

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Normally when we describe a method we also include its return type. Technically in Java the definition of method signature does not include the return type.

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

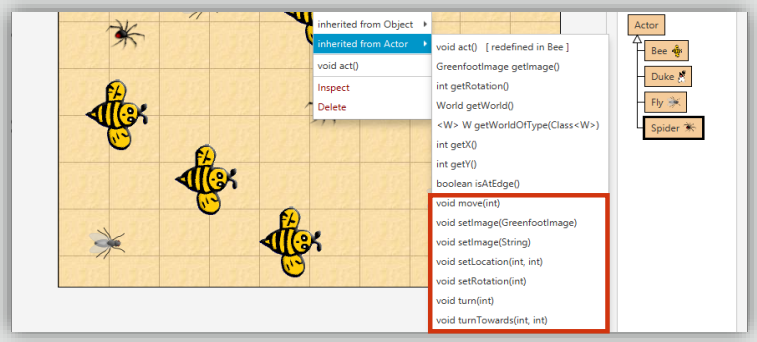
ORACLE Academy

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Normally when a method is a void we are instructing it to do something like move or turn. Where it is non void we are asking it for information like its current co-ordinates.

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



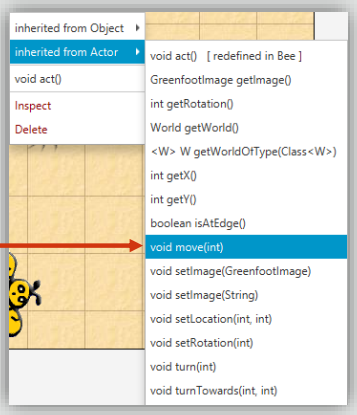
Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

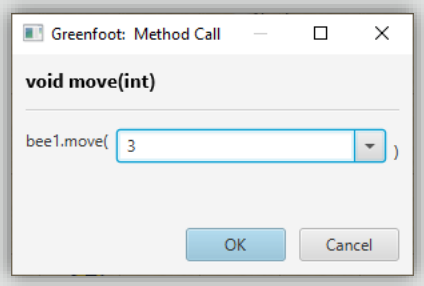
Choosing good names makes working with your objects and methods a lot easier. You could probably deduce what each method does just by looking at the method names in the screenshot. Have a go and see which ones you can guess the purpose of.

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





ORACLE Academy

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Test some of the other methods such as setLocation, setRotation and turn.

Ask Objects Questions

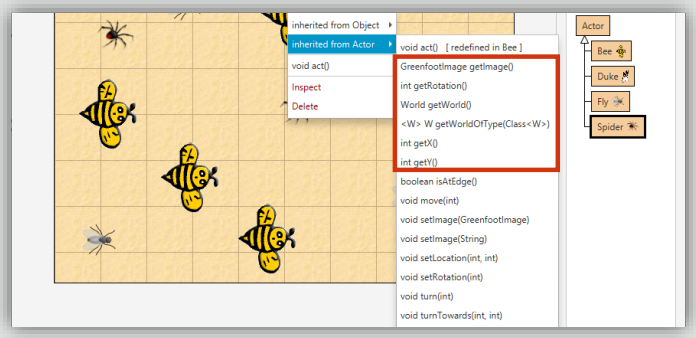
- As a programmer, you will ask objects questions by invoking methods with non-void return types to learn what an object can do, or has done in the past
- For example, in school, teachers ask students questions to see if the students comprehend the material they learned in class that day
- Students respond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

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

You can choose many objects in the real world and think of what questions you could ask it and how it would be returned to you. Example – a cooker. What methods does a cooker have of telling you data about its operations. Examples: How hot is the cooker? Is the cooker ready? How long something has been cooking? Is the cooker on?

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



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Later we will see what each of the return Data types are. In the example here we have an int (whole number), World (World instance) and boolean (true or false).

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?

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

The most common types that we will look at are integers, strings and booleans so it is worth becoming familiar with these.

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.

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Parameters are a mechanism that allow us to send values to the object via a method. We would use parameters to turn, move and position object instances in our world.

Examples of Method Parameters

- Integer (int):
  - Enter or display numerical values

```
private void updateLives(int change)
{
    lives +=change;
    updateImage(lives);
}//end of method
```

- Boolean:
  - Display true or false values

```
private void storeCurrentPosition(boolean collided)
{
    if(collided){
      prevX = getX();
      prevY = getY();
     }//endif
}//end of method store current position
```

- String :
  - Enter or display text values

```
private void play(String sound)
{//will play a sound file
  Greenfoot.playSound(sound);
}//end method Play
```

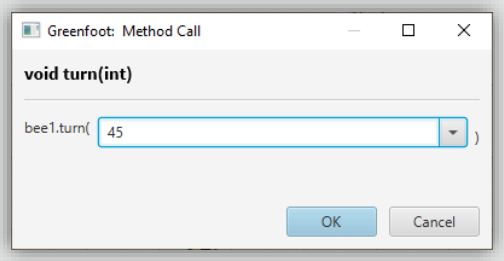
ORACLE Academy

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

The coordinate system within Greenfoot is based on whole numbers so most of the interaction with numbers will tend to follow this. You can make it more accurate by using fractional numbers, but our notes will stick with whole numbers.
Strings are for displaying alphanumeric characters.

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

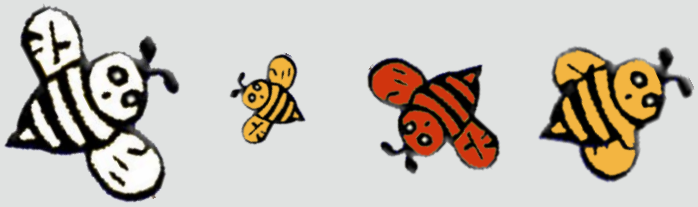


Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

In our screenshot we see that if we invoke the turn method within an instance of the Bee class it asks us to complete values for all of its parameters. In this example turn only expects an integer value, so it only prompts for one input. This is the degrees in clockwise that we wish it to rotate.

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

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

You can try and think of the properties that a human has. Then think of some instances such as yourself and state the variable values of each of the properties for this instance. Try this with another person. Each of you has the same properties, but may have different values to store in the variables.

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

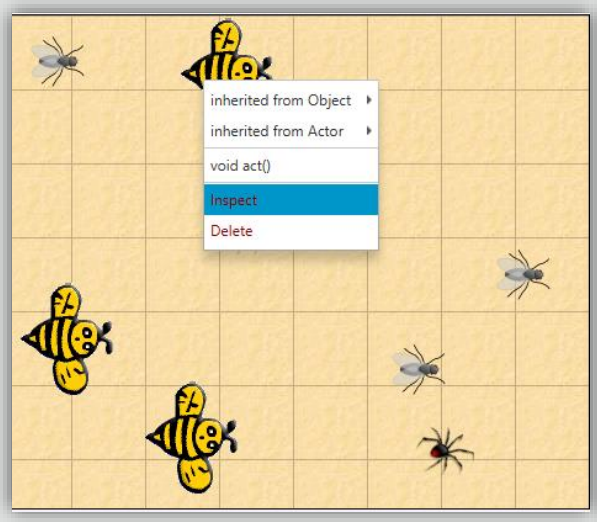
Instance variables are the memory that belong to the instance of the class. That memory can be saved and accessed later as long as the instance exists.

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Instance variables give the object state. An actor can be positioned at point (15, 20) facing 20 degrees and another actor of the same type can be located at (130,200) facing 180 degrees. They are both instances of the same class, but have different values associated with their variables or fields. In other words they have a different state. It's the values in the variables that set them apart.

View Instance Variables

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



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

This technique of looking at an actors instance variables is useful when you want to record its current values. Later we can use this in our code.

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

```
/**
 * Prepare the world for the start of the program.
 * That is: create the initial objects and add them to the world.
 */
private void prepare()
{
    Bee bee = new Bee();
    addObject(bee,298,184);
    Fly fly = new Fly();
    addObject(fly,466,289);
    Spider spider = new Spider();
    addObject(spider,79,79);
    fly.setLocation(462,287);
    bee.setLocation(305,48);
```

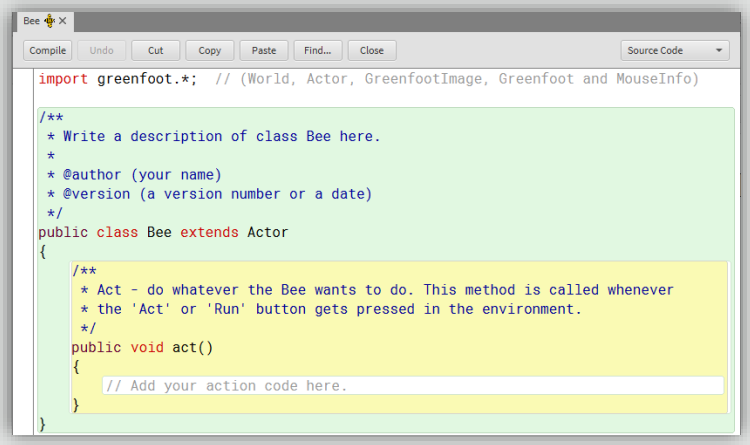
ORACLE Academy

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Remember syntax has to be exact for Java to compile it. As pointed out earlier, just because code compiles and is syntactically correct does not mean it is bug free.

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



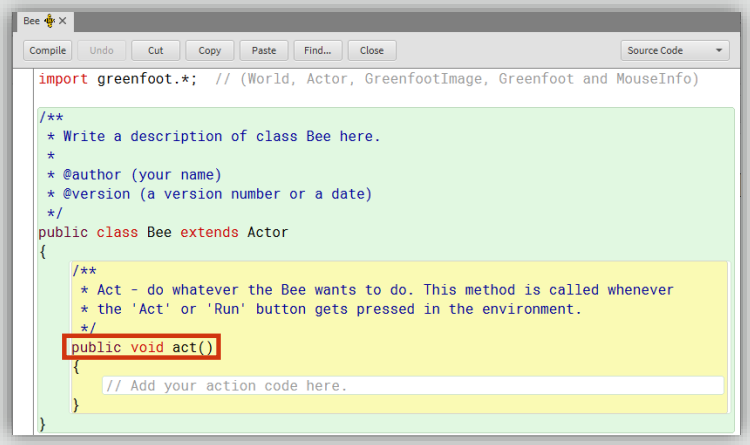
ORACLE Academy

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Note: If you see documentation, when you open your code window, remember to change the drop down at the top right to Source Code.

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

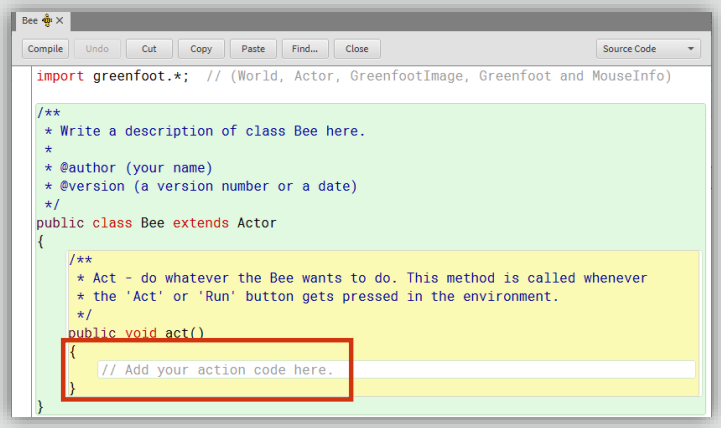


Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

It is important to note here the word repeatedly. The code in act() will loop. i.e. it will run all the code, then go back to the start of act and run it again and again and again.

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



Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

For every bracket "("or "{" (curly bracket) you add, you must have a corresponding closing bracket. This is one of the biggest syntax errors you will receive early on. Leaving too many open brackets will usually return the error "reached end of file while parsing" at the end of the code.

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

```
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);
    }
}
```

Class compiled - no syntax errors saved

ORACLE Academy

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Add
move(1);
turn(15);
to the act method of one of your actors. Create an instance of this actor in your world and select run.

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);
    }
}
```

ORACLE Academy

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Forgetting to add a semi colon at the end of commands is one of the most common syntax errors. Java will normally detect this missing semi-colon error correctly and point you to it.

Debugging Process in Greenfoot

- Incorrectly written or missing characters in your source code will trigger error messages
- 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

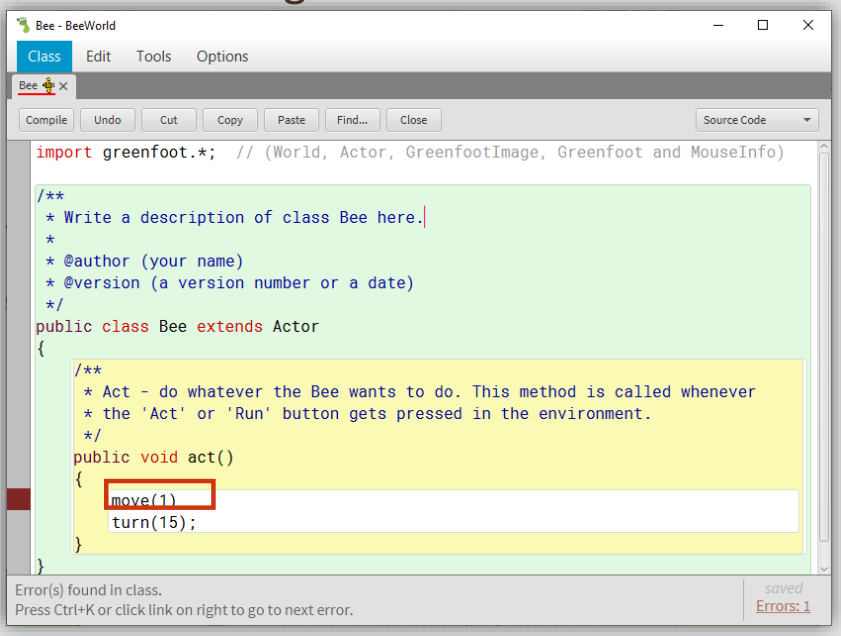
Debugging is the process of finding and removing bugs—or errors—in a computer program.

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

The more you program the more aware you will become of the messages that are generated via syntax errors.

Syntax Error Example

- The move() method is missing a semicolon



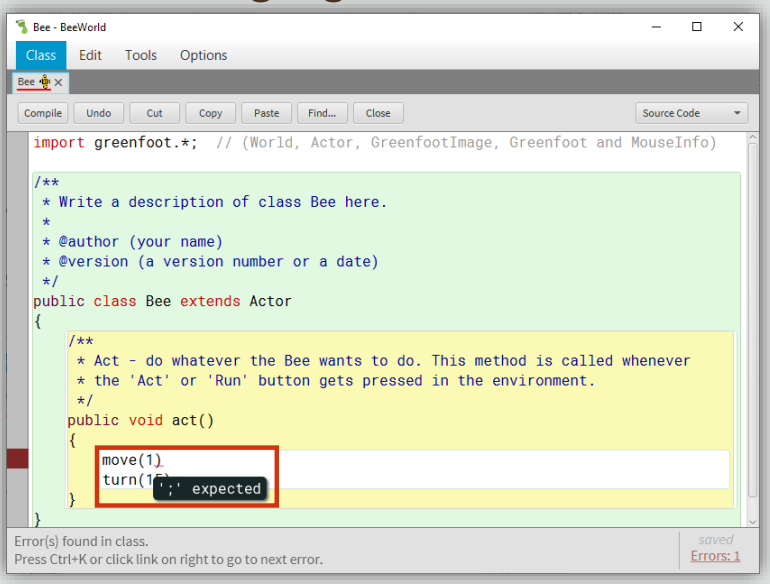
ORACLE Academy

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

This would return the error message ";" expected while highlighting the move(1) code line.

Error Message Example

- An error message appears at the bottom of the screen and the incorrect code is highlighted



ORACLE Academy

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

Terminology

- Key terms used in this lesson included:
  - Debug
  - Inheritance
  - Instance variable
  - Method
  - Method call
  - Parameter
  - Return type
  - Method signature
  - Variable

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

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

Copyright © 2020, Oracle and/or its affiliates. All rights reserved.

ORACLE
Academy