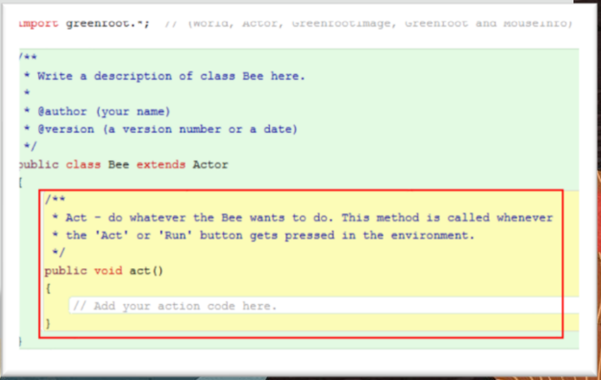
ORACLE
Academy

Java Fundamentals

3-3
Source Code and Documentation



```
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()
    {
        // Add your action code here.
    }
}
```

ORACLE
Academy

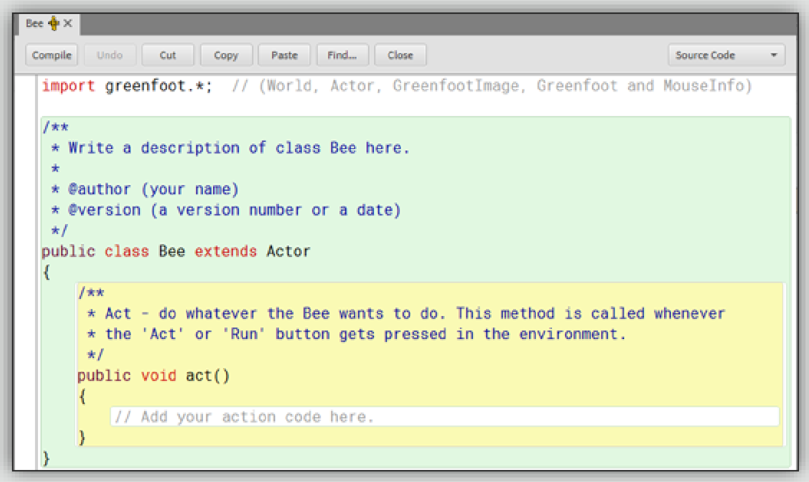
Objectives

- This lesson covers the following objectives:
  - Demonstrate source code changes to invoke methods programmatically
  - Demonstrate source code changes to write an if decision statement
  - Describe a method to display object orientation

ORACLE
Academy

Source Code

- Source code is the blueprint or map that defines how your objects and program function
- It commands the objects in your scenario to move and interact



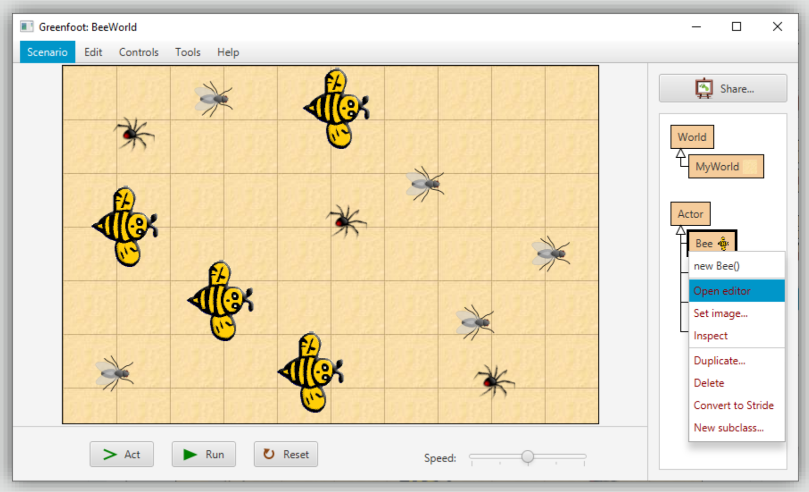
```
import greenfoot.*;  // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/**
 * 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()
    {
        // Add your action code here.
    }
}
```

ORACLE
Academy

After initial placement of your actors you will spend most of your time working with the source code.

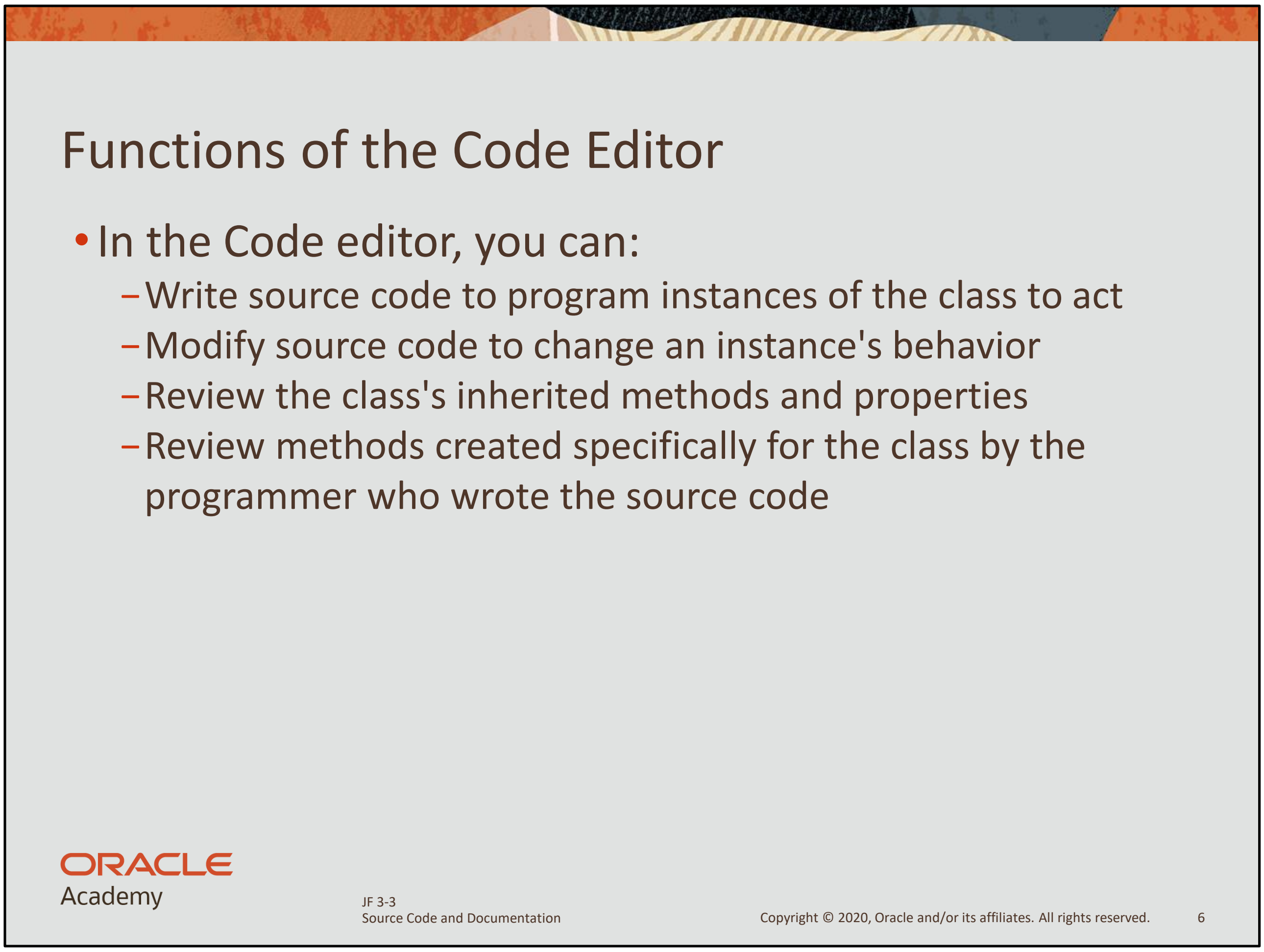
Code Editor

- Source code is managed in the Code editor
- To view the Code editor, right click on any class in the environment, then select Open editor from the menu



ORACLE Academy

You can also double click on the class and the default code editor will open in a new window.

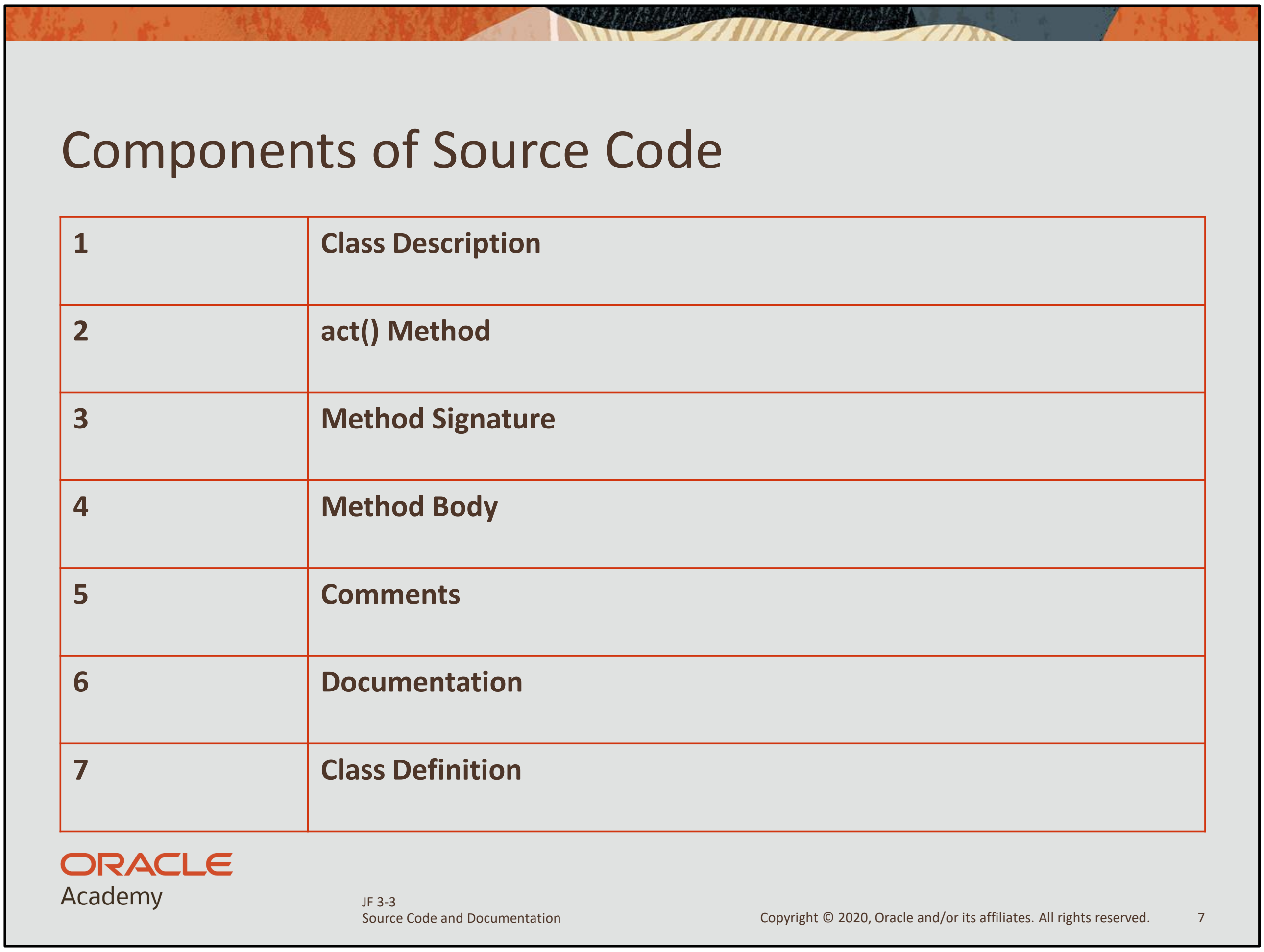


Functions of the Code Editor

- In the Code editor, you can:
  - Write source code to program instances of the class to act
  - Modify source code to change an instance's behavior
  - Review the class's inherited methods and properties
  - Review methods created specifically for the class by the programmer who wrote the source code

ORACLE
Academy

Don't worry if looking at the code looks quite complicated at the moment. This will become very clear in the near future.



Components of Source Code

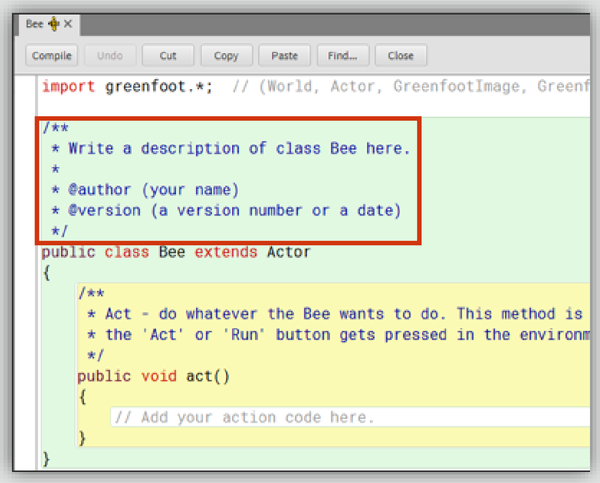
1	**Class Description**
2	**act() Method**
3	**Method Signature**
4	**Method Body**
5	**Comments**
6	**Documentation**
7	**Class Definition**

ORACLE
Academy

All of these except for the description, comments and documentation are required to make your program work. The description, comments and documentation make it easier for others and yourself to see what your code is trying to achieve. Never underestimate these sections and you will be grateful for the documentation supplied by the Greenfoot development team. You should get into the good habit of commenting and documenting your own code.

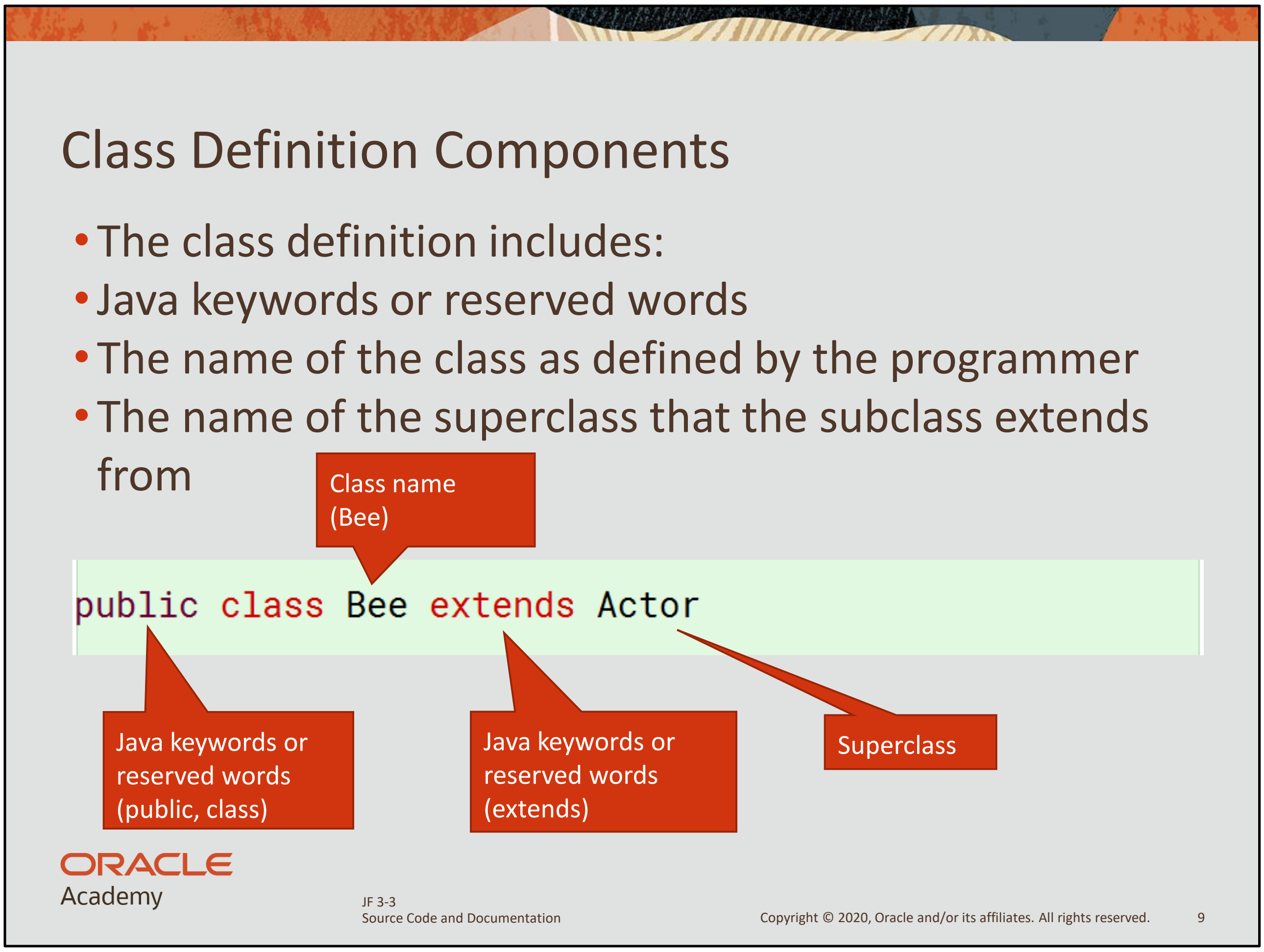
Class Description

- The class description is a set of comments that can be modified to describe the class
- This includes:
  - A description of what the class does
  - The name of the person who authored the code
  - The date the source code was last modified



```
import greenfoot.*;  // (World, Actor, GreenfootImage, Greenf

/**
 * 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
     * the 'Act' or 'Run' button gets pressed in the environm
     */
    public void act()
    {
        // Add your action code here.
    }
}
```

Earlier we looked at the auto generated documentation. Text from these comments is used to create more meaningful documentation.



Class Definition Components

- The class definition includes:
- Java keywords or reserved words
- The name of the class as defined by the programmer
- The name of the superclass that the subclass extends from

```
public class Bee extends Actor
```

Class name (Bee)

Java keywords or reserved words (public, class)

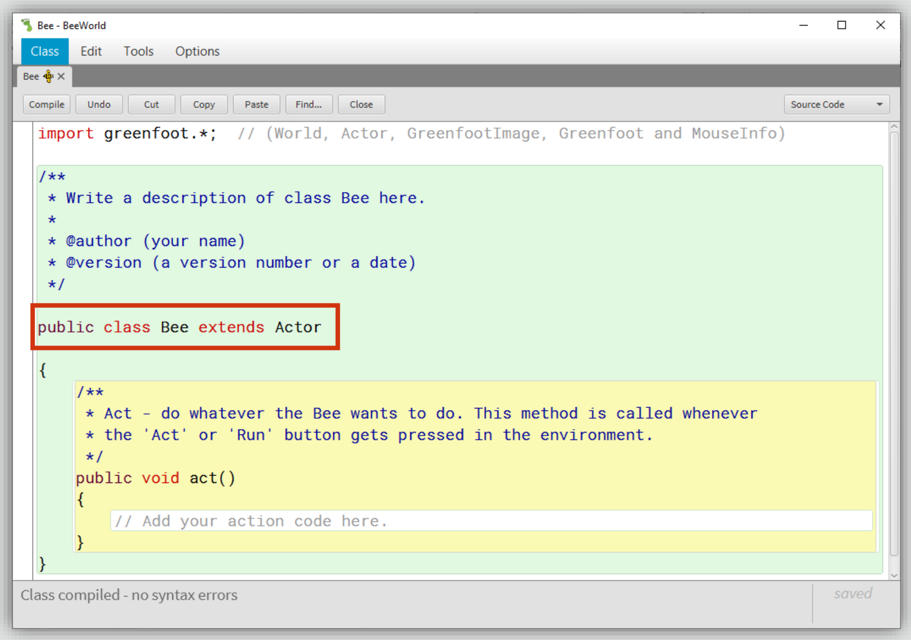
Java keywords or reserved words (extends)

Superclass

ORACLE Academy

Public tells java how accessible the object should be. Public is the least restrictive and private the most restrictive. We look at this in more detail later.
Class tells java that we are about to define our own class.
Extends tells Java that we wish to base it on the already defined class called Actor. This means that we are creating a subclass of Actor called Bee.

Class Definition Example



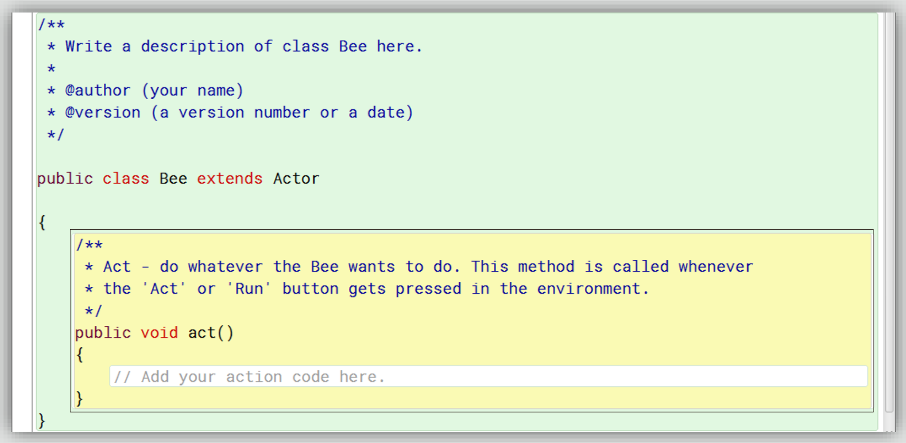
```
import greenfoot.*;  // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/**
 * 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()
    {
        // Add your action code here.
    }
}
```

Class compiled - no syntax errors

Remember that it is common to name the class starting with a capital. So in the example above we have used Bee rather than bee.

act() Method

- The act() method is the part of the class definition that tells objects which methods to perform when the Act or Run execution controls are clicked in the environment



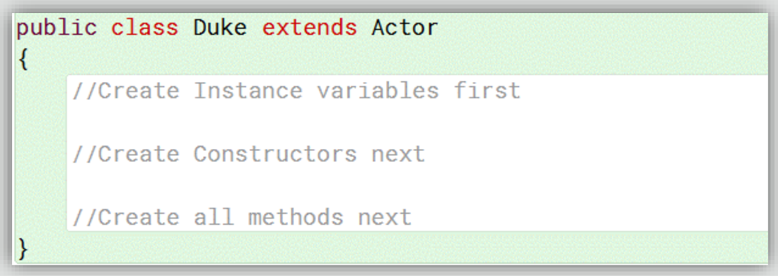
```
/**
 * Write a description of class Bee here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */

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()
    {
        // Add your action code here.
    }
}
```

ORACLE
Academy

The red highlighted area also includes the comments. These comments are also used in the auto generated documentation. Remember early that we stated that the act() method continually loops. That is it runs the code, then runs it again and so on.

Defining Classes

- The class definition defines:
  - Variables (or fields) that store data persistently within an instance
  - Constructors that initially set up an instance
  - Methods that provide the behaviors for an instance
- Use a consistent format when you define a class
  - For example, define variables first, constructors second, and methods third



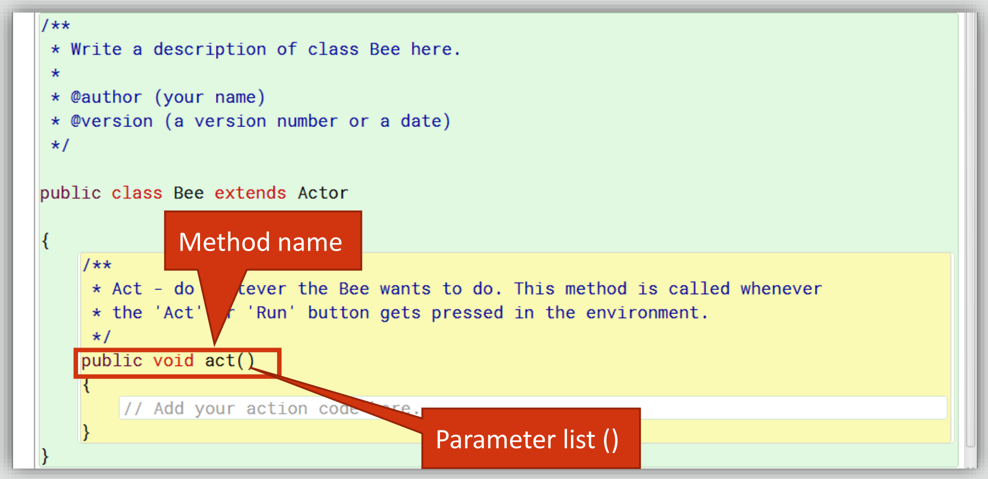
```java
public class Duke extends Actor
{
	//Create Instance variables first

	//Create Constructors next

	//Create all methods next
}
```

ORACLE
Academy

Methods fall into 3 varieties that we will explore later on. They either return information from the fields. This is usually the value or some calculation on the value. Or they set the value stored in the fields either by passing in a value, or again via some calculation. Or they provide some functionality.

Method Signature

- The method signature describes what the method does
- The signature contains a method name and parameter list

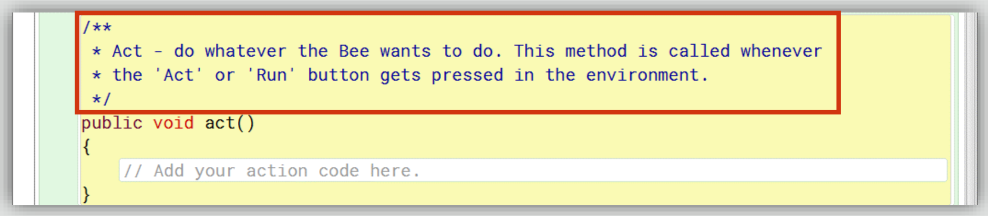


```
/**
 * Write a description of class Bee here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */

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()
    {
        // Add your action code here.
    }
}
```

Like a class the method definition also has accessibility. These are the same values that are used to define the class accessibility with public being the most accessible.
The act method does not return anything, so the void term is used to denote this. If you fail to add void or a return type then java will report this as a syntax error.

Comments

- Comments describe what the source code does
- Do not impact the functionality of the program
- Start with a forward slash and two asterisks /** or simply a double forward slash
- End /** comments with */
- Written in blue font (in Greenfoot)



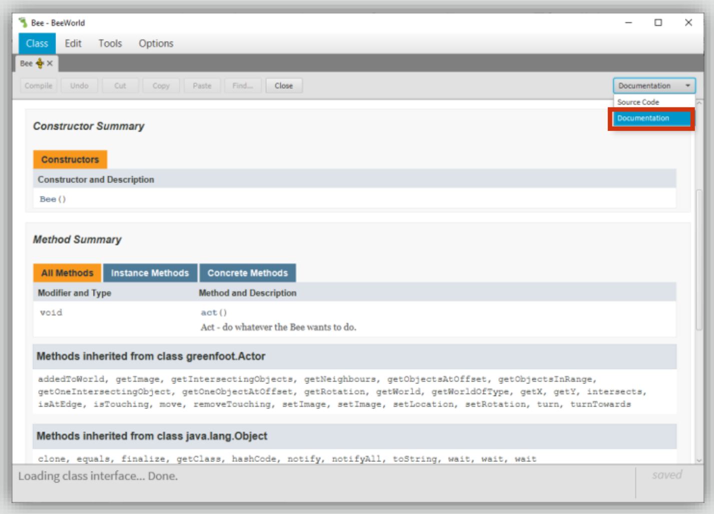
```
/**
 * Act - do whatever the Bee wants to do. This method is called whenever
 * the 'Act' or 'Run' button gets pressed in the environment.
 */
public void act()
{
    // Add your action code here.
}
```

ORACLE
Academy

Java ignores comments. They are used for us to better understand the code and its meaning. Remember you may be working with code that was written by someone else. You will be grateful if they have added comments.

Documentation

- Documentation describes the properties of the class
- To view, select Documentation from the drop-down menu at the top right of the Code editor



Constructor Summary

Constructors

Constructor and Description
`Bee ()`

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
`void`	`act ()` Act - do whatever the Bee wants to do.

Methods inherited from class greenfoot.Actor

```
addedToWorld, getImage, getIntersectingObjects, getNeighbours, getObjectsAtOffset, getObjectsInRange,
getOneIntersectingObject, getOneObjectAtOffset, getRotation, getWorld, getWorldOfType, getX, getY, intersects,
isAtEdge, isTouching, move, removeTouching, setImage, setImage, setLocation, setRotation, turn, turnTowards
```

Methods inherited from class java.lang.Object

```
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait
```

Loading class interface... Done.

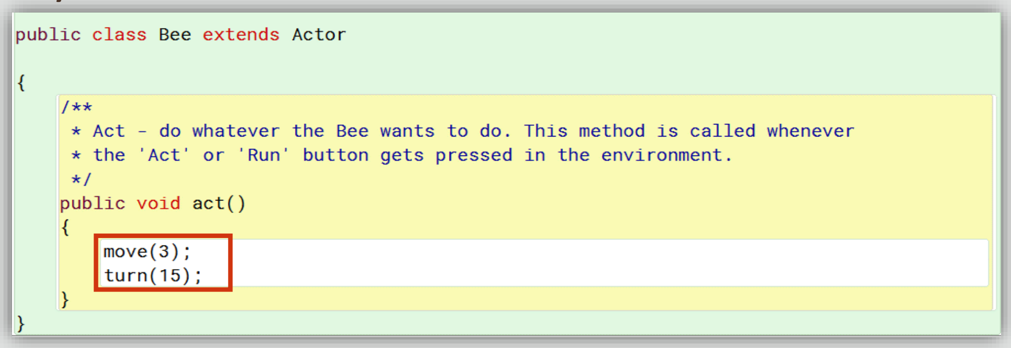
saved

ORACLE Academy

Notice that if you change the author name, the version number or the comments before the act then these now appear in the documentation.

Invoke Methods Programmatically

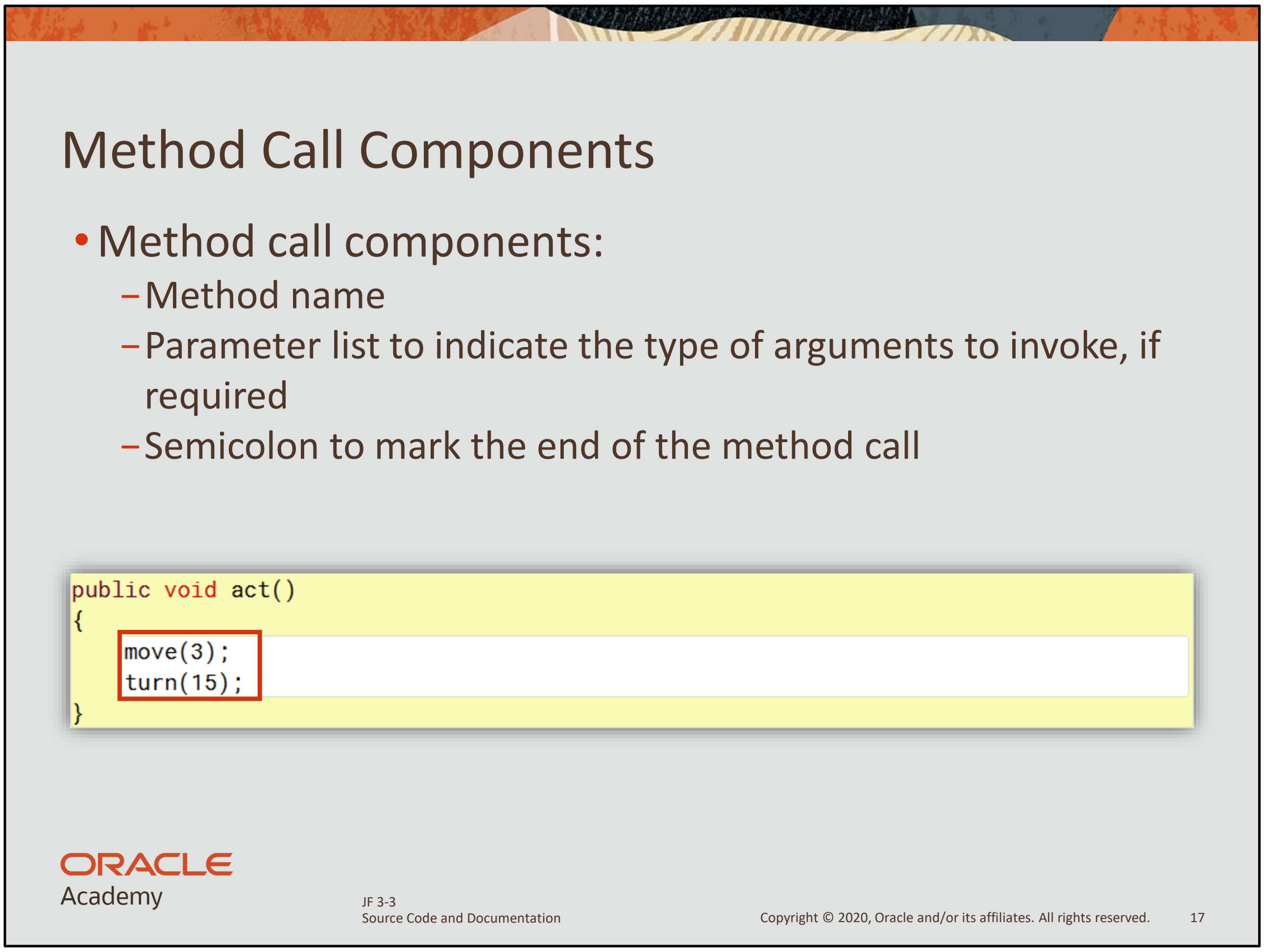
- Methods must be invoked to command instances to act in your game
- Invoke methods programmatically by writing them in the body of the act() method in the space between the curly brackets



```
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(3);
        turn(15);
    }
}
```

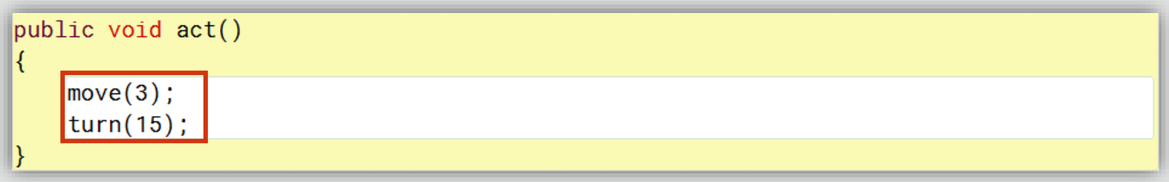
ORACLE Academy

These methods will be called sequentially within the act() method.



Method Call Components

- Method call components:
  - Method name
  - Parameter list to indicate the type of arguments to invoke, if required
  - Semicolon to mark the end of the method call



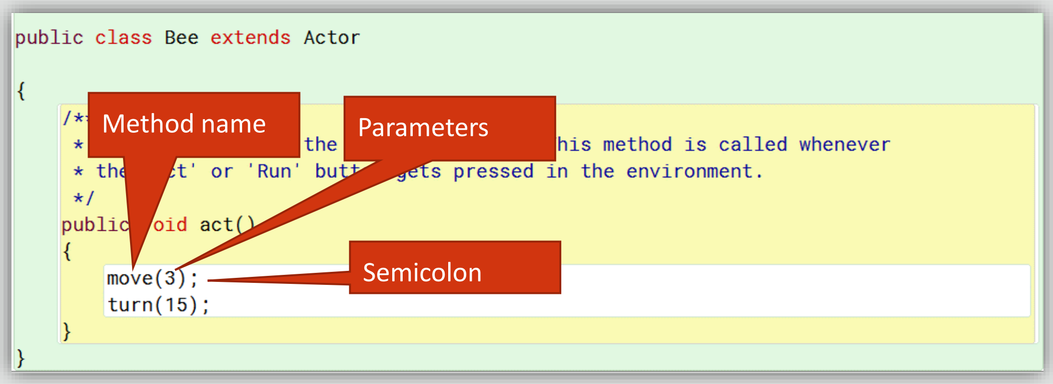
```
public void act()
{
    move(3);
    turn(15);
}
```

ORACLE Academy

If you are unsure of what a method does, returns or requires then look at its or its superclass's documentation.

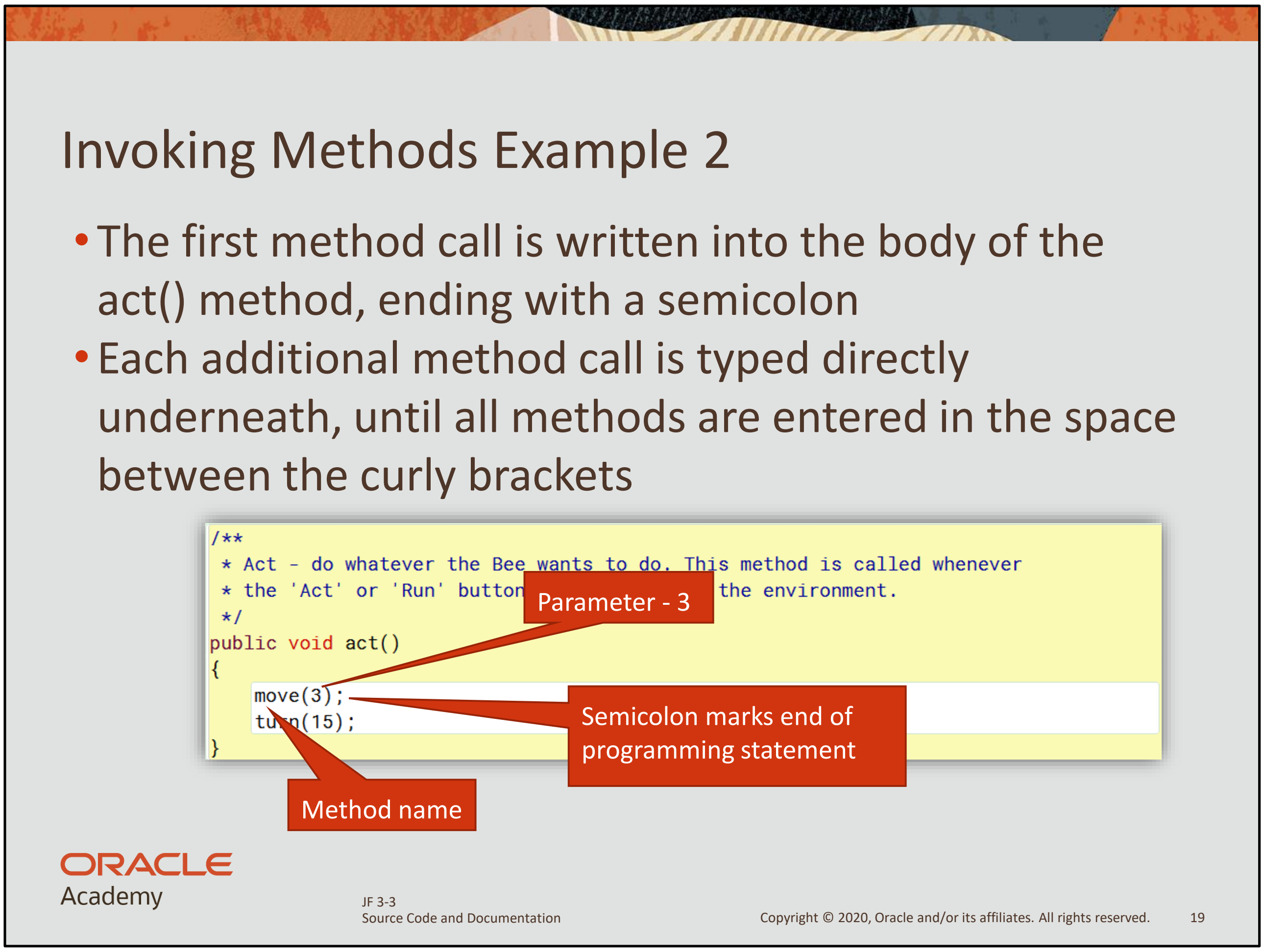
Invoking Methods Example 1

- Each method is written in the space between the curly brackets



```
public class Bee extends Actor
{
    /**
     *                the                  his method is called whenever
     * the    ct' or 'Run' butt    gets pressed in the environment.
     */
    public  oid act()
    {
        move(3);
        turn(15);
    }
}
```

Notice that the code after the curly brackets is indented. This is good coding practice and makes it more readable.



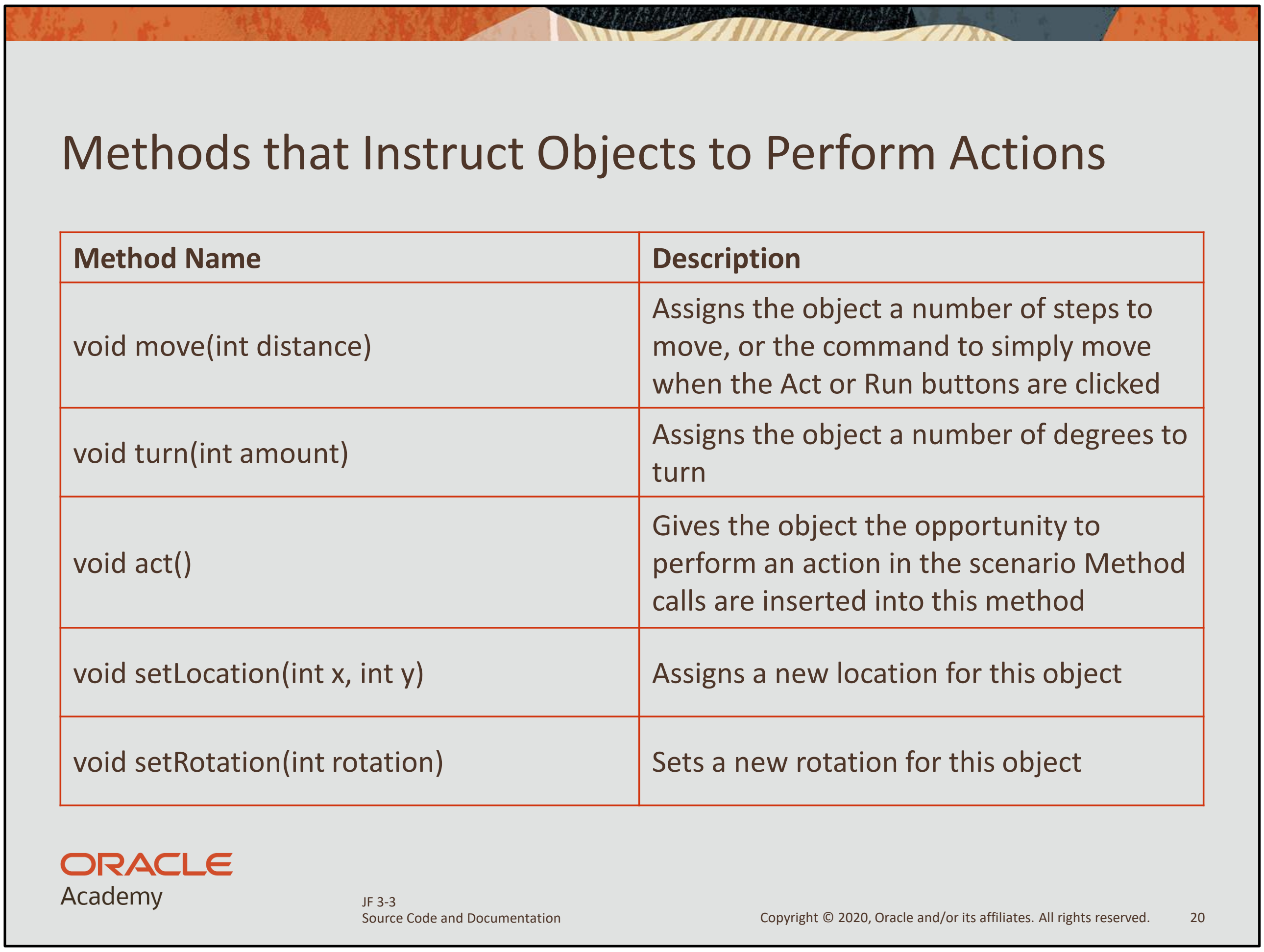
Invoking Methods Example 2

- The first method call is written into the body of the act() method, ending with a semicolon
- Each additional method call is typed directly underneath, until all methods are entered in the space between the curly brackets

```
/**
 * 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(3);
    turn(15);
}
```

ORACLE Academy

As we are putting this in the act() method they will constantly be called sequentially.

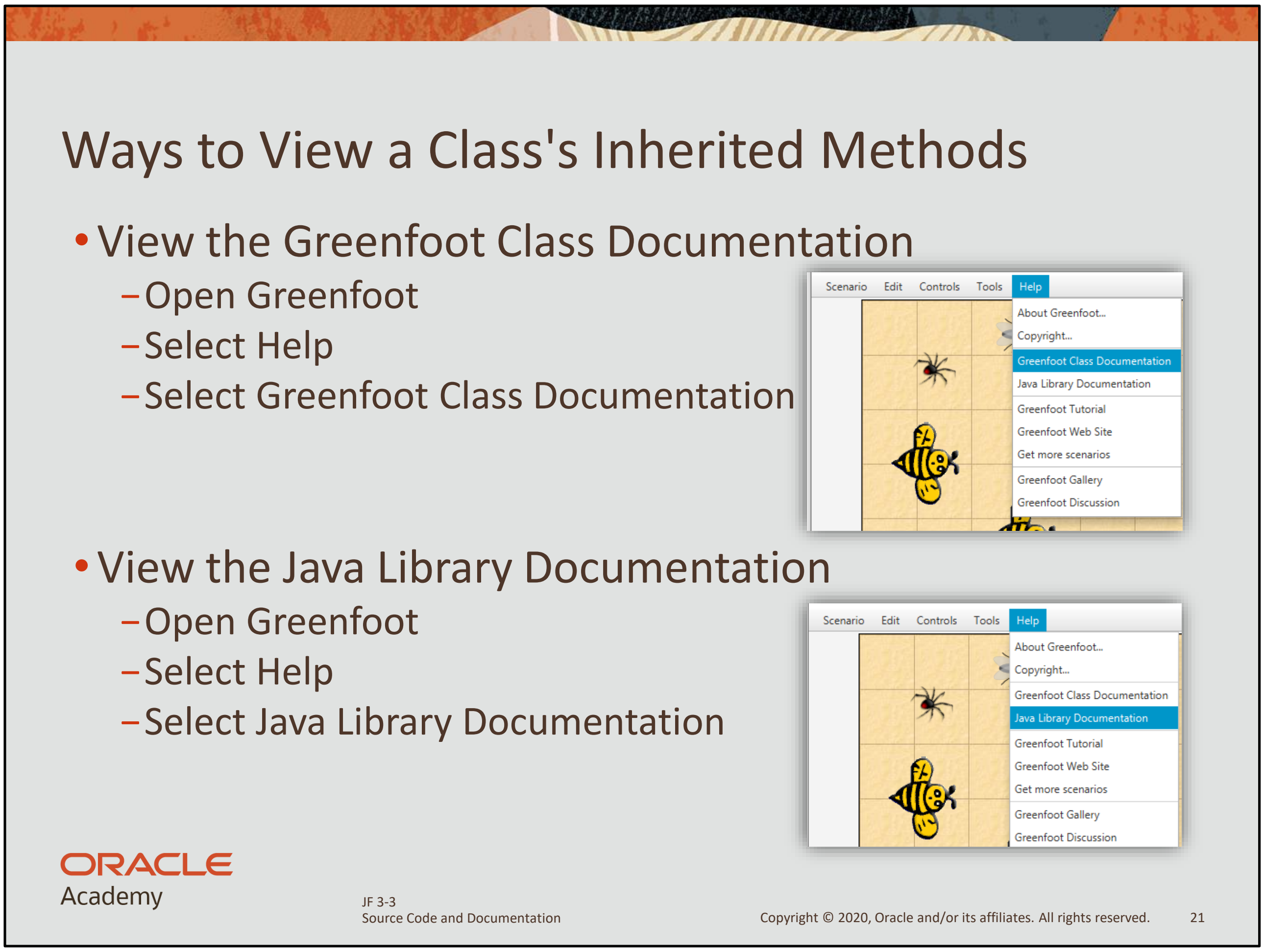


Methods that Instruct Objects to Perform Actions

Method Name	Description
void move(int distance)	Assigns the object a number of steps to move, or the command to simply move when the Act or Run buttons are clicked
void turn(int amount)	Assigns the object a number of degrees to turn
void act()	Gives the object the opportunity to perform an action in the scenario Method calls are inserted into this method
void setLocation(int x, int y)	Assigns a new location for this object
void setRotation(int rotation)	Sets a new rotation for this object

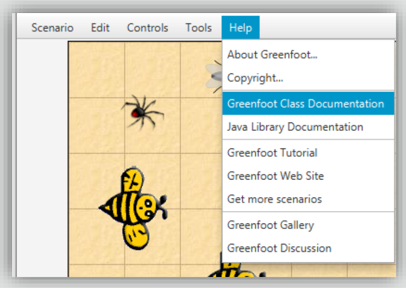
ORACLE
Academy

All of the methods shown can be tested by right clicking on an Actor instance and selecting the method. If it requires values for the parameters then these will be requested

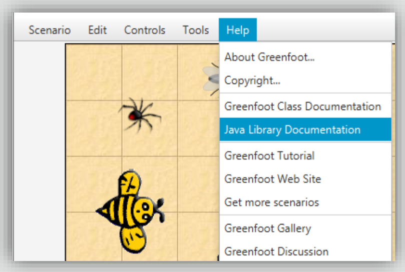


Ways to View a Class's Inherited Methods

- View the Greenfoot Class Documentation
  - Open Greenfoot
  - Select Help
  - Select Greenfoot Class Documentation



- View the Java Library Documentation
  - Open Greenfoot
  - Select Help
  - Select Java Library Documentation



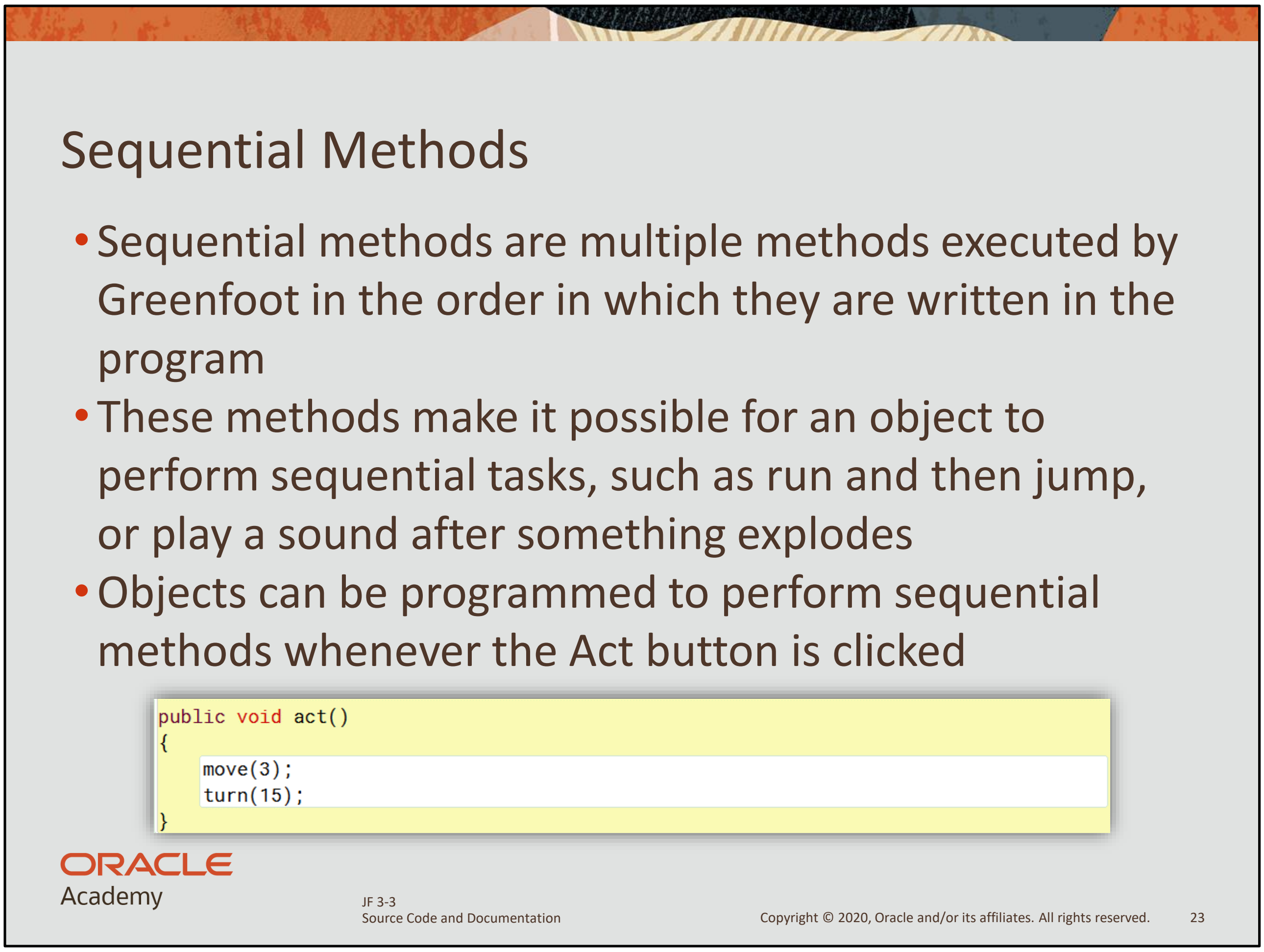
Inherited methods are the methods that have been defined in the superclass, or higher.

Sequential Tasks

- A single task, such as going to school, requires multiple sub-tasks:
  - Wake up
  - Take a shower
  - Brush your teeth
  - Get dressed…

- Within a sub-task, there could be more sub-tasks (walking to school requires the left leg and right legs to move forward, in order)

Breaking up a problem into smaller problems is a well known technique for allowing us to understand a scenario better. This same technique works well in programming where we break down a problem into smaller and smaller chunks until we feel we have gone far enough to fully understand that smaller problem.



Sequential Methods

- Sequential methods are multiple methods executed by Greenfoot in the order in which they are written in the program
- These methods make it possible for an object to perform sequential tasks, such as run and then jump, or play a sound after something explodes
- Objects can be programmed to perform sequential methods whenever the Act button is clicked

```
public void act()
{
    move(3);
    turn(15);
}
```

ORACLE
Academy

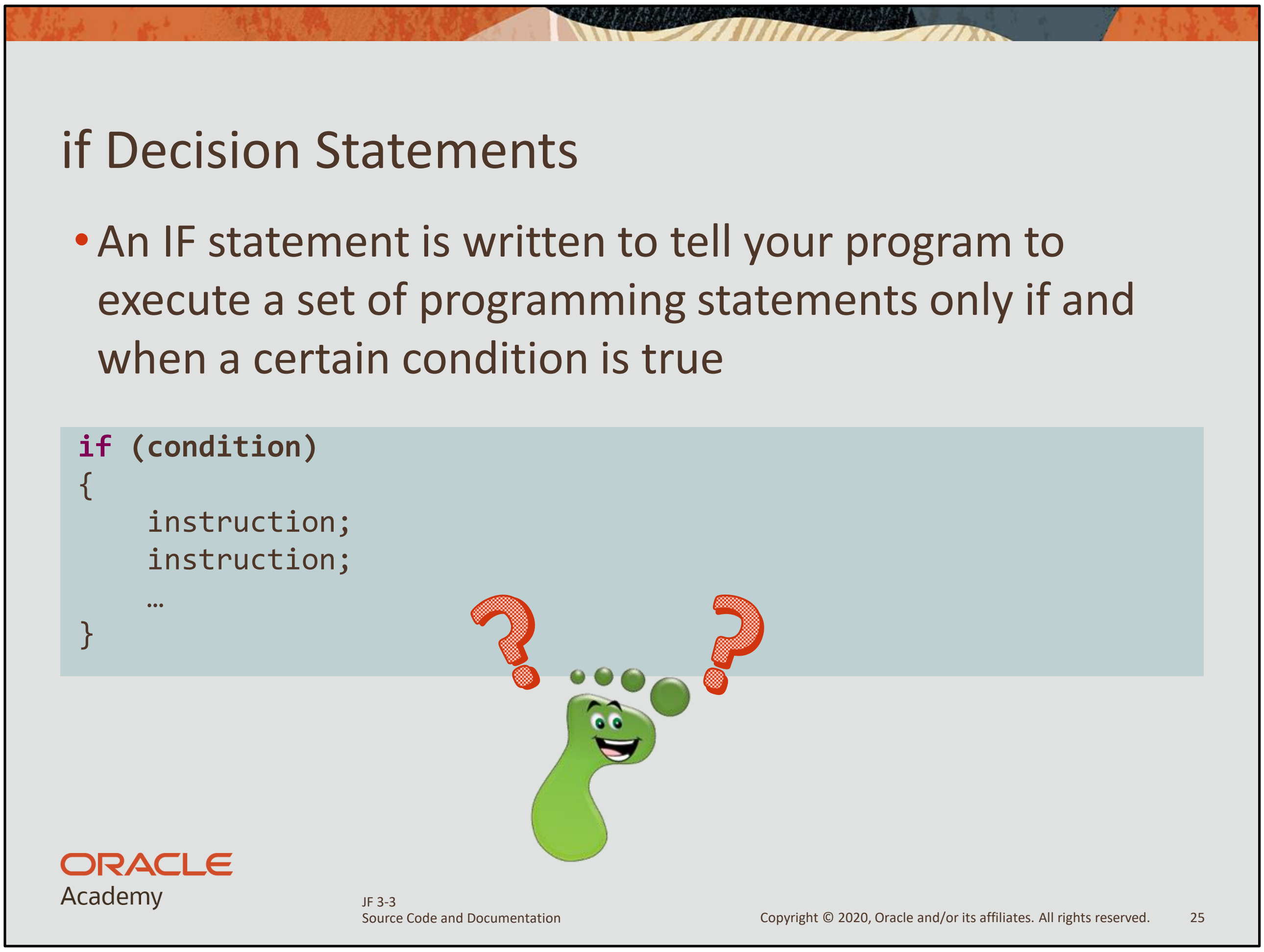
The order in which we carry out tasks may be important depending on what we are trying to achieve. If we have 2 tasks –
Put on left sock
Put on right sock
The order of these will not change the overall outcome – putting on both socks.
If we have the tasks
Put on left sock
Put on left shoe
The order of these is important. Unless we want a sock over our shoe…

if-then Relationships

- Many things around us have a cause and effect relationship, or "if-then" relationship
- If your cell phone rings, then you answer it
- If it doesn't ring, then you do not answer it
- If a flower starts to wilt, then you give it water
- If the flower looks healthy, then you do not give it water

ORACLE Academy

You may have seen IF statements before in other languages or even with software packages such as spreadsheets.



if Decision Statements

- An IF statement is written to tell your program to execute a set of programming statements only if and when a certain condition is true

```
if (condition)
{
    instruction;
    instruction;
    …
}
```

ORACLE Academy

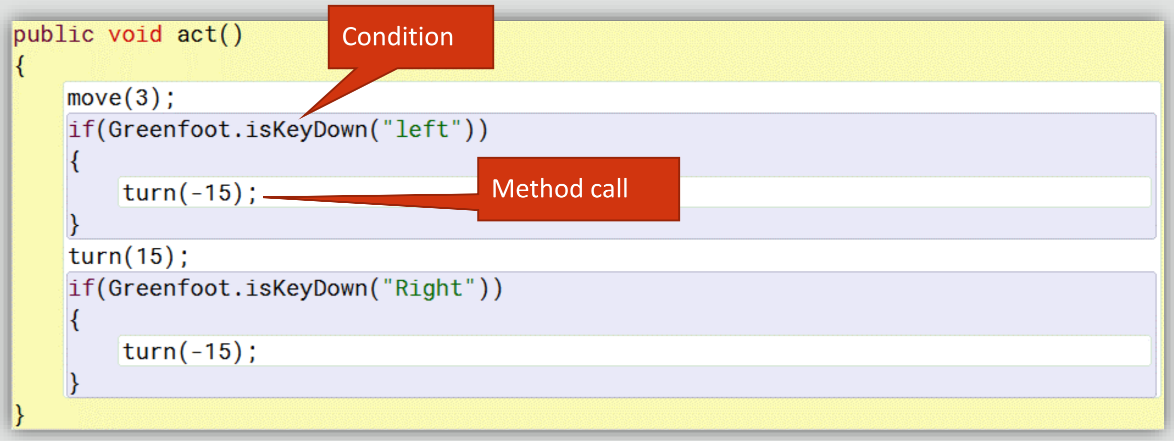
JF 3-3
Source Code and Documentation

25

The if statement can control whether or not a set of instructions should run. You will make great use of the if statement as it allows code to react to certain events in your scenario. i.e. 2 objects colliding, a user clicking a key or a mouse selecting an actor are just a few examples where we may wish to choose if certain sections of code should run.

if Decision Statement Components

- The if statement contains a condition, which is a true or false expression, and one or more method calls that are executed if the condition is met



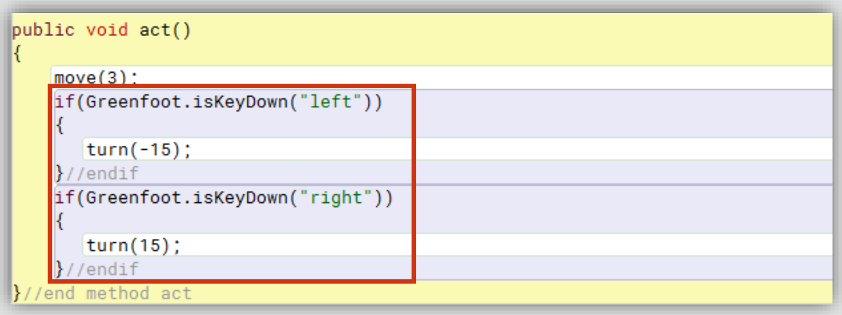
```
public void act()
{
    move(3);
    if(Greenfoot.isKeyDown("left"))
    {
        turn(-15);
    }
    turn(15);
    if(Greenfoot.isKeyDown("Right"))
    {
        turn(-15);
    }
}
```

ORACLE Academy

The condition isKeyDown("left") tests for the left arrow key.
Adding a negative turn value for the parameter in turn means the actor will turn in a counter clockwise direction.

if Decision Statement Example

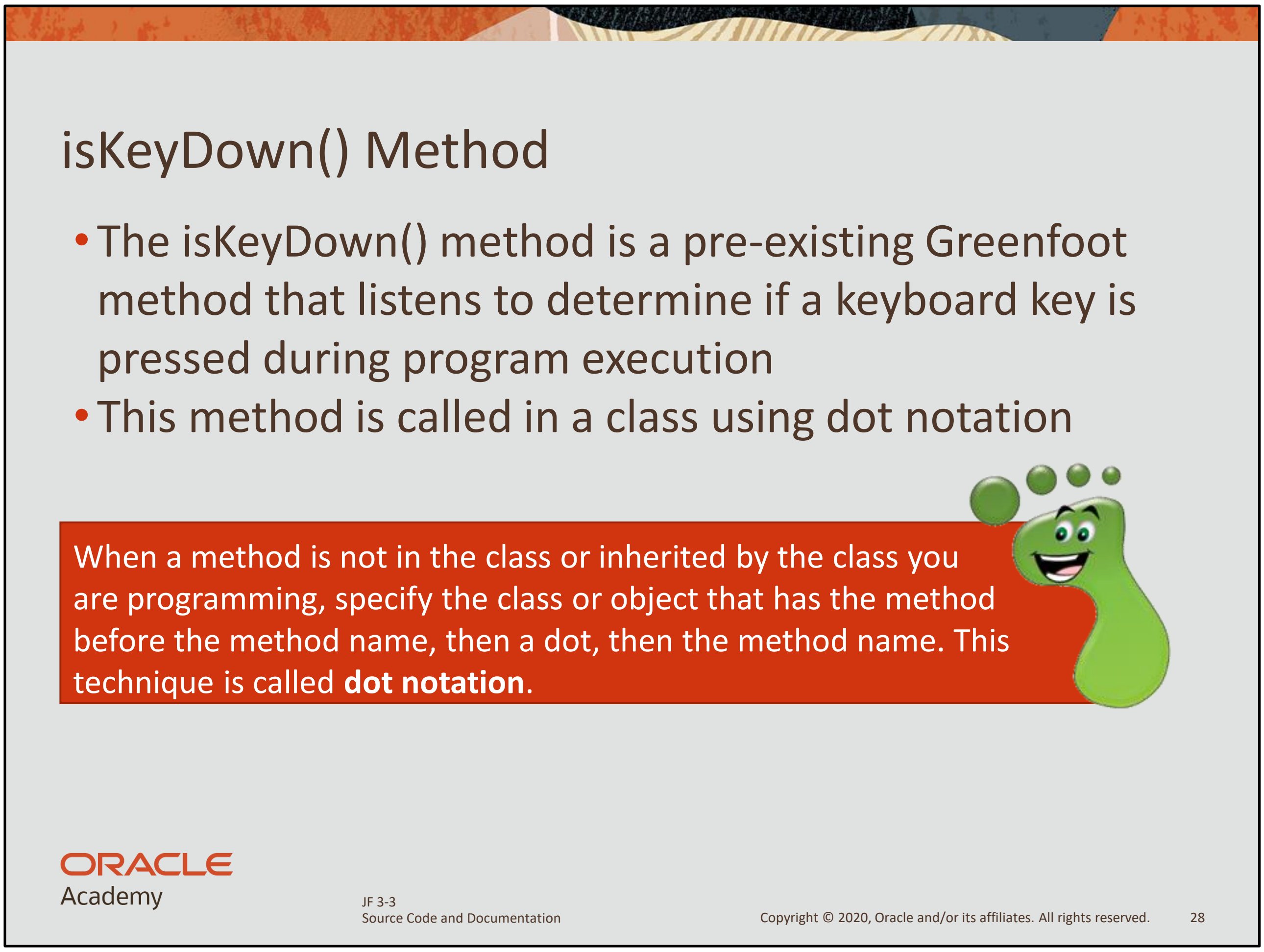
- In the following example:
  - The left and right arrow keys on the keyboard make the object turn left and right
  - If the condition is false, the method calls defined in the IF statement are not executed
  - The move() method is executed regardless of the IF statement



```
public void act()
{
    move(3);
    if(Greenfoot.isKeyDown("left"))
    {
       turn(-15);
    }//endif
    if(Greenfoot.isKeyDown("right"))
    {
       turn(15);
    }//endif
}//end method act
```

ORACLE Academy

Remember that the act() method is continually looping. So the methods are continually being called sequentially within its brackets.



isKeyDown() Method

- The isKeyDown() method is a pre-existing Greenfoot method that listens to determine if a keyboard key is pressed during program execution
- This method is called in a class using dot notation

When a method is not in the class or inherited by the class you are programming, specify the class or object that has the method before the method name, then a dot, then the method name. This technique is called **dot notation**.

ORACLE
Academy

JF 3-3
Source Code and Documentation

28

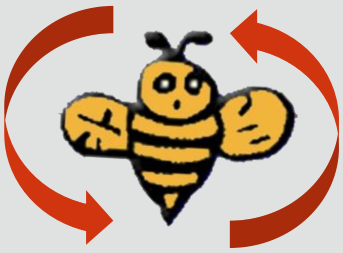
The isKeyDown() method is often used to give the user control of an actor via the keyboard.

Object Orientation in the Real World

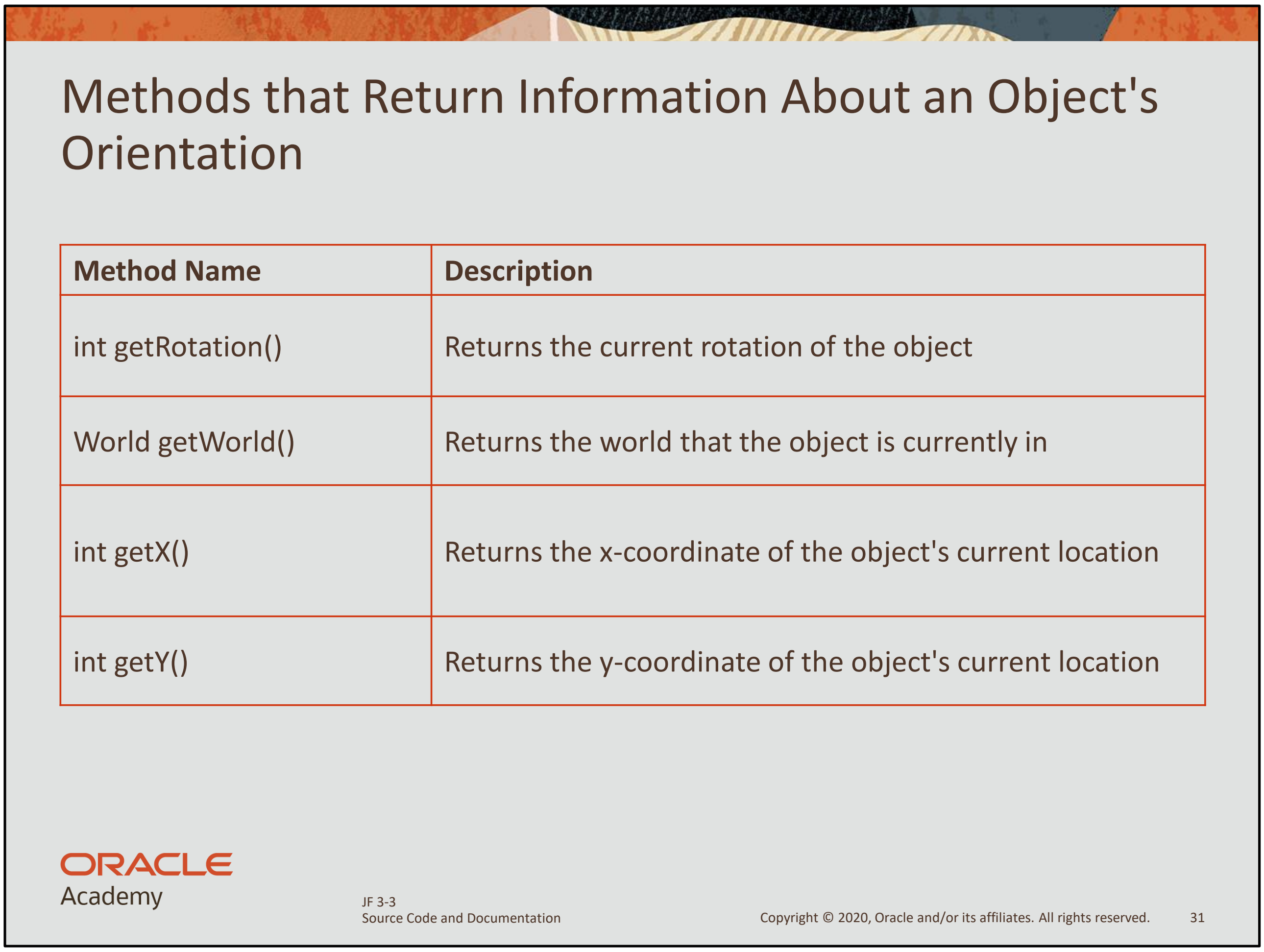
- As we move about the world we live in, it's important for us to know our orientation, or sense of direction
- When you drive a car, you always need to know if your car is in the correct lane of the road
- When a plane flies through the air, it needs to know where it's located relative to other planes, so a collision doesn't occur
- When you enter your location on a map in a cell phone, you receive coordinates that tell you where you are, and the address

Display an Object's Orientation

- Methods can tell us how an object is positioned in the world, relative to itself and other objects
- You can invoke a method:
  - with a specific data type, such as boolean, to ask the object a question about its orientation
  - in the environment to learn how the object is oriented in the scenario



An objects orientation in the world and its comparison to other objects are the building blocks of the software we will write in Greenfoot.



Methods that Return Information About an Object's Orientation

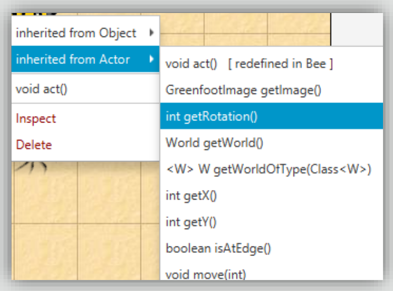
Method Name	Description
int getRotation()	Returns the current rotation of the object
World getWorld()	Returns the world that the object is currently in
int getX()	Returns the x-coordinate of the object's current location
int getY()	Returns the y-coordinate of the object's current location

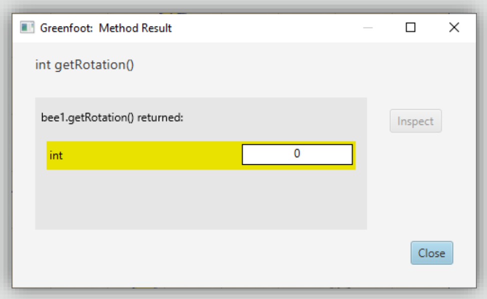
ORACLE
Academy

360 degrees is a full rotation. Be careful as 0 degrees is the angle pointing to the right of the screen. So down is 90, left is 180 and up is 270 degrees.

Steps to Invoke a Method that Displays an Object's Orientation

1. Right click on the instance in the world
2. Select Inherited from Actor to view its methods
3. Invoke (select) a method with a specific data type to ask the object a question about its orientation
4. The method result will display
5. Note the value returned, then click Close





int getRotation()

bee1.getRotation() returned:

int 0

ORACLE Academy

In Greenfoot the default world size is 600 across (x) and 400 down (y).
So in a world of this size the top left is the position (0,0) and the bottom right is the position(600,400)

Terminology

- Key terms used in this lesson included:
  - Class description
  - Comments
  - if decision statements
  - Invoking a method
  - Sequential methods

Summary

- In this lesson, you should have learned how to:
  - Demonstrate source code changes to invoke methods programmatically
  - Demonstrate source code changes to write an if decision statement
  - Describe a method to display object orientation

ORACLE
Academy

ORACLE

Academy