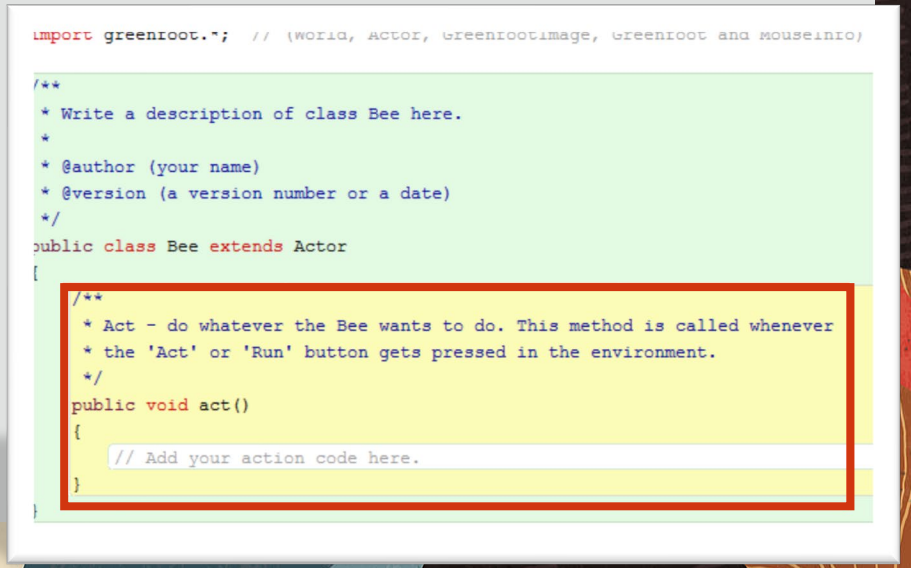

ORACLE
Academy

Creación de programas Java con Greenfoot

Lección 4

Código fuente y documentación



ORACLE Academy

Objetivos

- En esta lección se abordan los siguientes objetivos:
  - Demostrar los cambios en el código fuente para llamar a métodos mediante programación
  - Demostrar los cambios de código fuente para escribir una sentencia de toma de decisiones if
  - Describir un método para mostrar la orientación de objetos

Código fuente

- El código fuente es el plano o mapa que define cómo funcionan los objetos y programas
- Controla los objetos en su escenario para que se muevan e interactúen

Bee

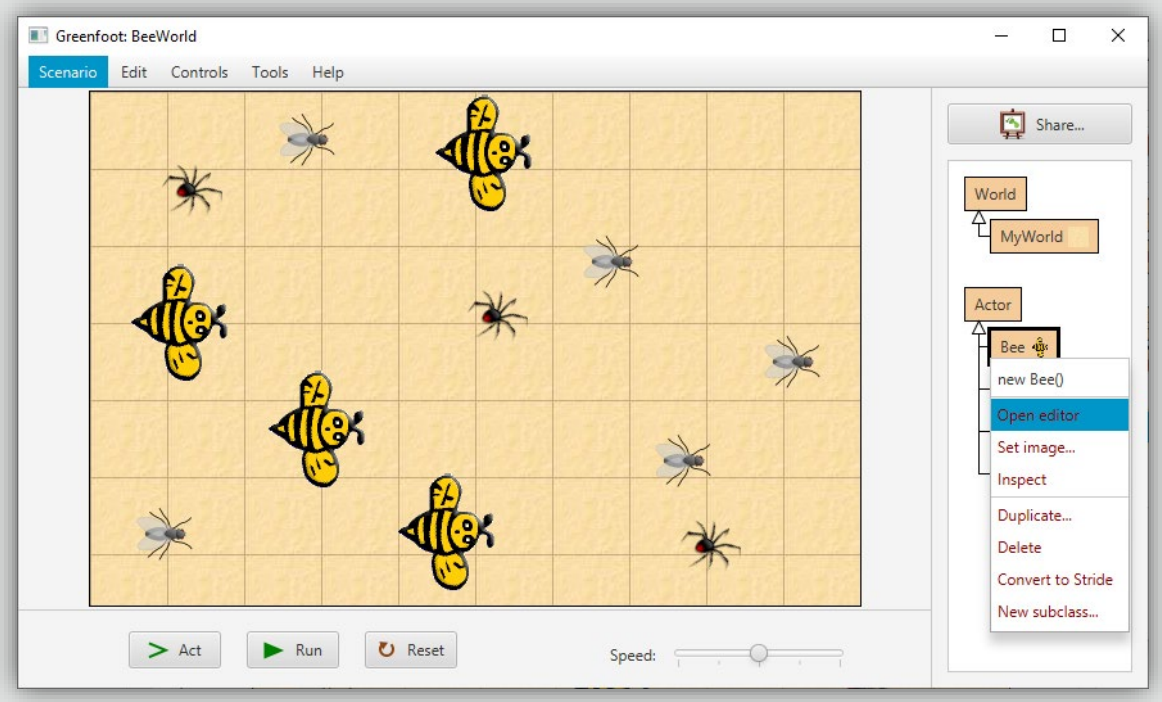
Compile | Undo | Cut | Copy | Paste | Find... | Close | Source Code

```
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.
    }
}
```

Editor de códigos

- El código fuente se gestiona en el editor de códigos
- Para ver el editor de códigos, haga clic en cualquier clase en el entorno y, a continuación, seleccione Open editor en el menú



Funciones del editor de códigos

- En el editor de códigos, puede:
  - Escribir código fuente para programar que actúen instancias de la clase
  - Modificar el código fuente para cambiar el comportamiento de una instancia
  - Revisar los métodos y propiedades heredados de la clase
  - Revisar los métodos creados específicamente para la clase por el programador que escribió el código fuente

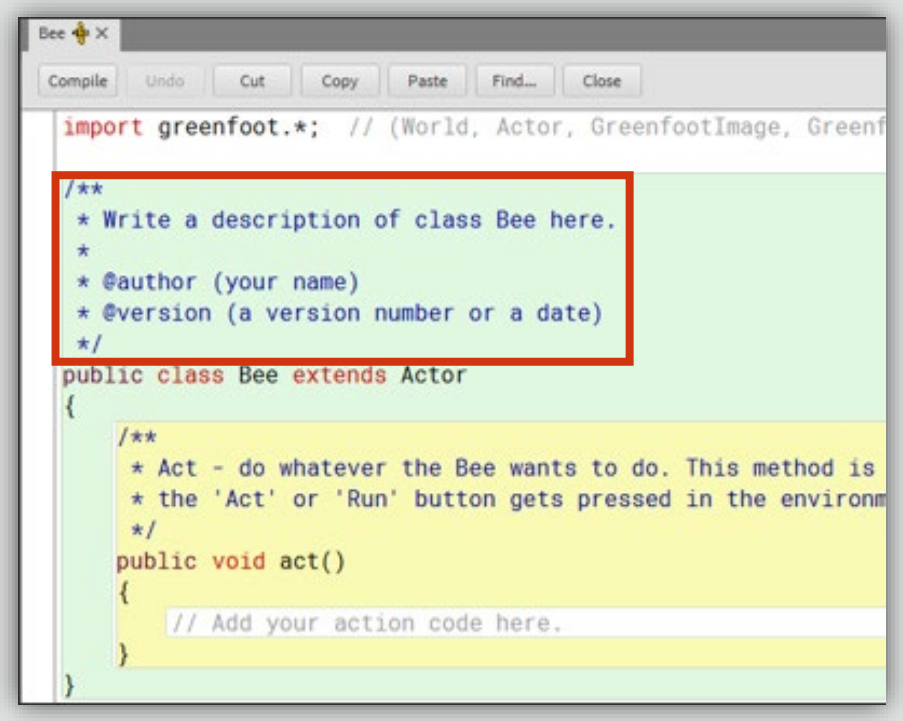
Componentes del código fuente

1	Descripción de clase
2	Método act()
3	Firma de método
4	Cuerpo de método
5	Comentarios
6	Documentación
7	Definición de clase

Descripción de clase

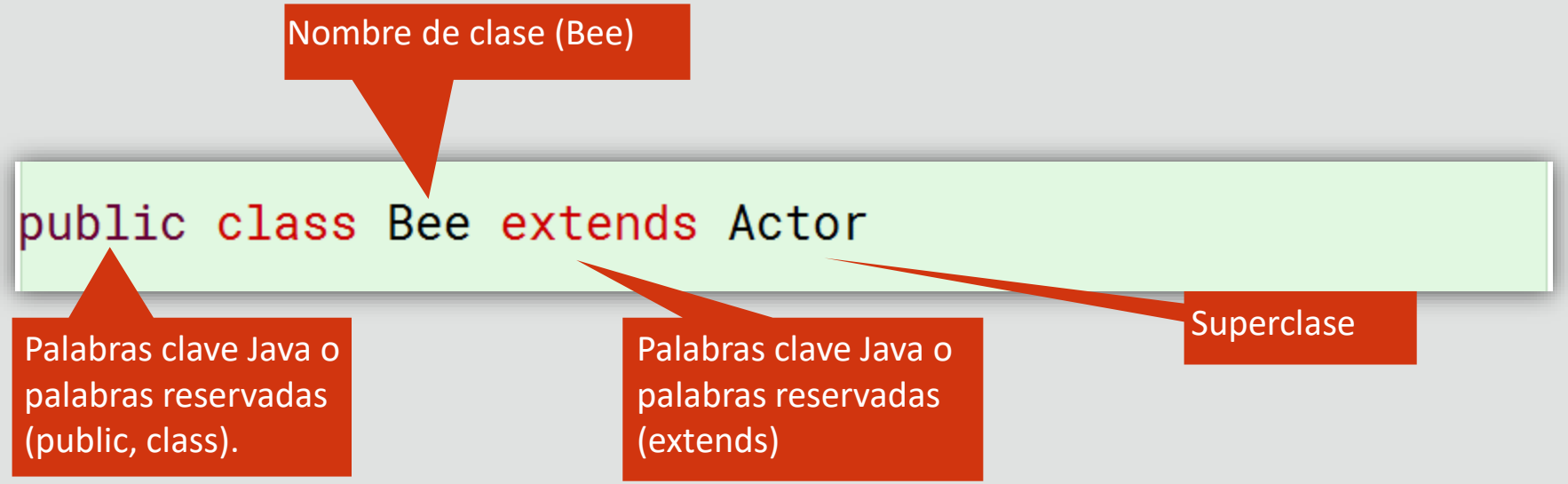
La descripción de clase es una serie de comentarios que se pueden modificar para describir la clase. Incluye:

- Una descripción de la función de la clase
- El nombre de la persona que creó el código
- La fecha en la que se modificó por última vez el código fuente

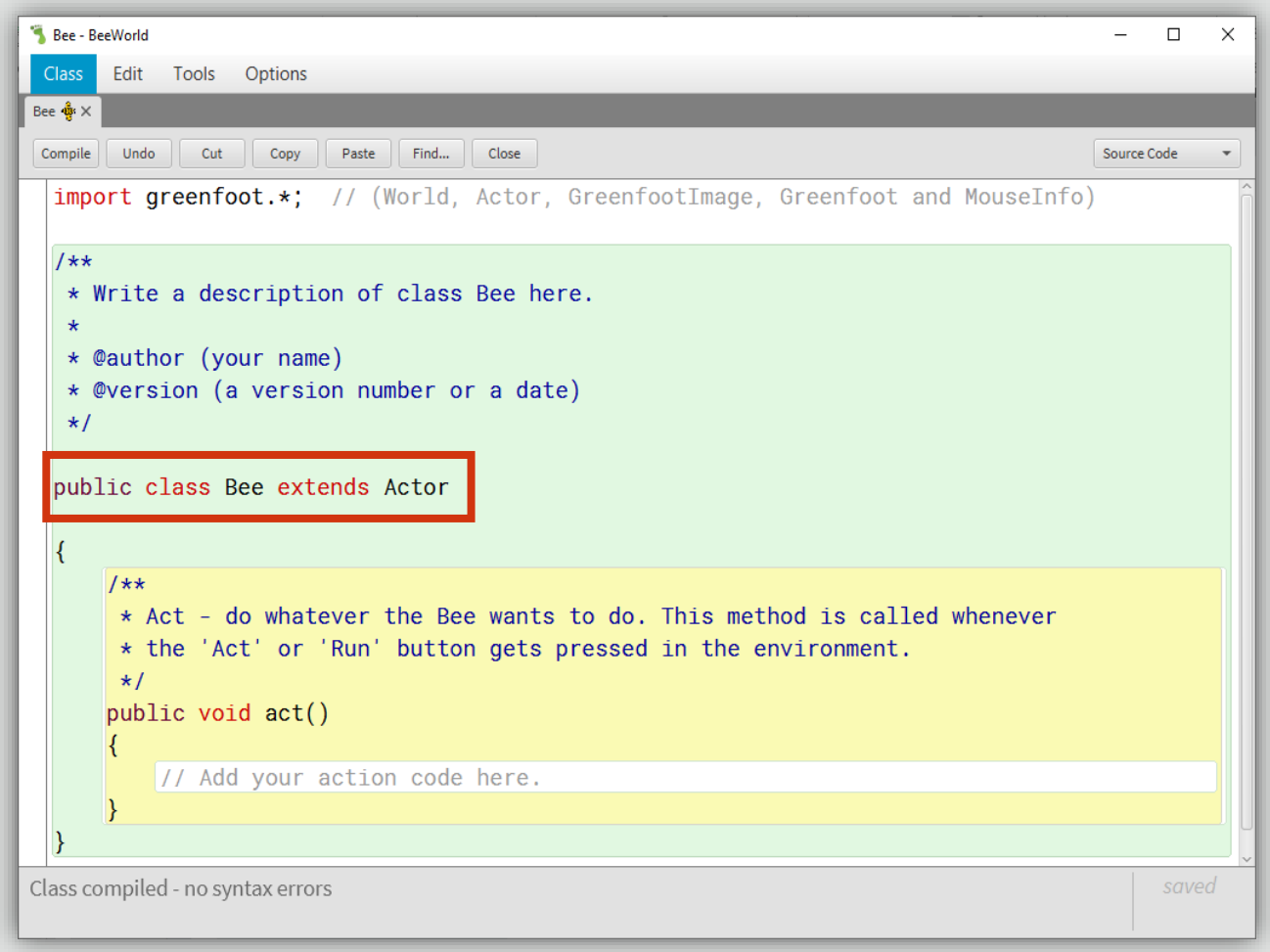


Componentes de la definición de clase

- La definición de clase incluye:
  - Palabras clave Java o palabras reservadas
  - El nombre de la clase tal como lo defina el programador
  - El nombre de la superclase que amplía la subclase



Ejemplo de definición de clase



Método act()

- El método act() es la parte de la definición de clase que indica a los objetos los métodos que deben realizar cuando se hace clic en los controles de ejecución Act o Run en el entorno

```
/**
 * 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.
    }
}
```

Definición de clases

- La definición de clase define:
  - Las variables (o campos) que almacenan datos de forma persistente en una instancia
  - Los constructores que configuran inicialmente una instancia
  - Los métodos que proporcionan los comportamientos de una instancia
- Utilice un formato coherente cuando defina una clase
  - Por ejemplo, defina primero las variables, los constructores en segundo lugar y los métodos por último

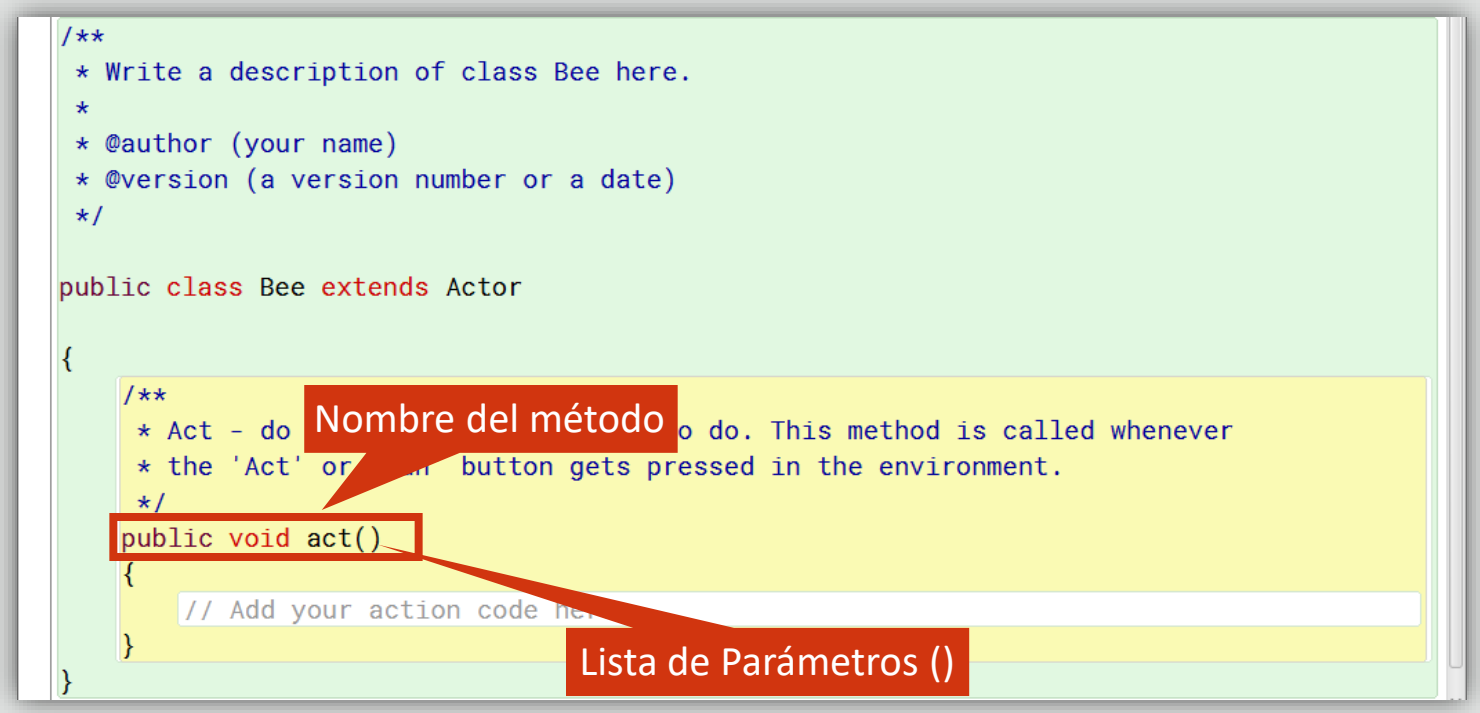
```
public class Duke extends Actor
{
    //Create Instance variables first

    //Create Constructors next

    //Create all methods next
}
```

Firma de método

- La firma de método describe lo que hace el método
- La firma contiene un nombre de método y una lista de parámetros



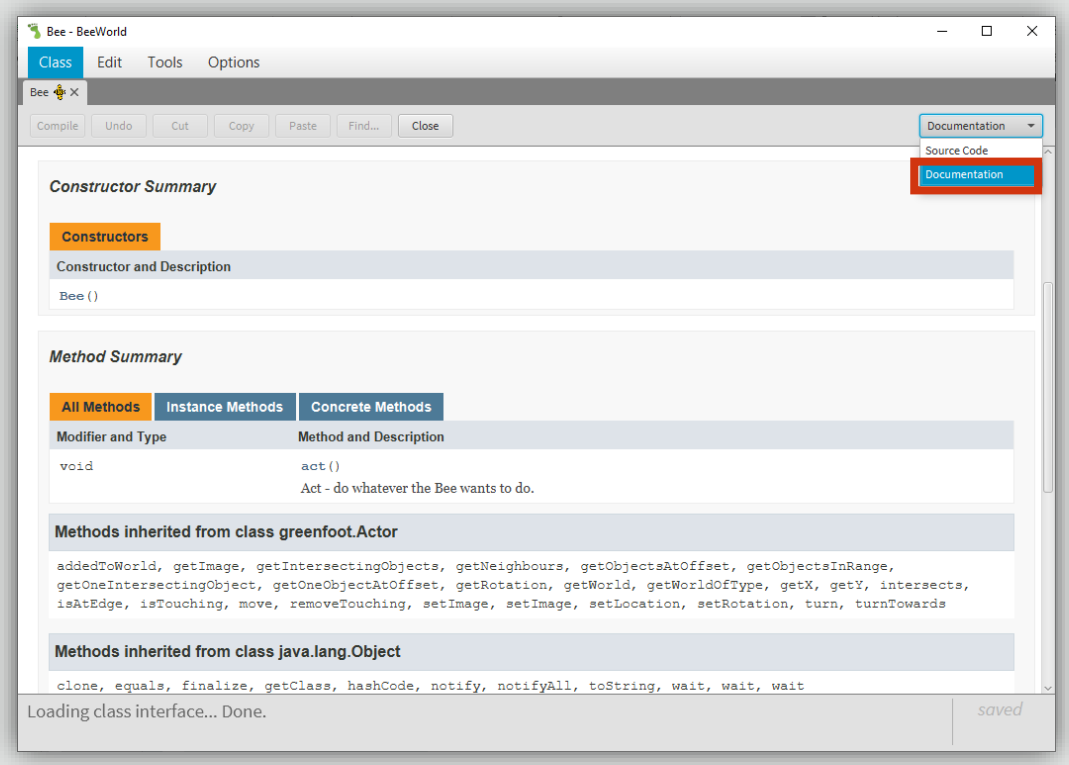
Comentarios

- Los comentarios describen lo que hace el código fuente
  - No afectan a la funcionalidad del programa
  - Empiezan por una barra inclinada y dos asteriscos /** o simplemente una doble barra inclinada
  - Finalice los comentarios /** con */
  - Escritos en color azul (en Greenfoot)

```
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.
    }
}
```

Documentación

- La documentación describe las propiedades de la clase
- Para verla, seleccione Documentation en el menú desplegable en la parte superior derecha del editor de códigos



Llamada a los métodos mediante programación

- Se debe llamar a métodos para que ordenen a las instancias que actúen en su juego
- Llame a los métodos mediante programación escribiéndolos en el cuerpo del método act() en el espacio entre los corchetes

```
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);
    }
}
```

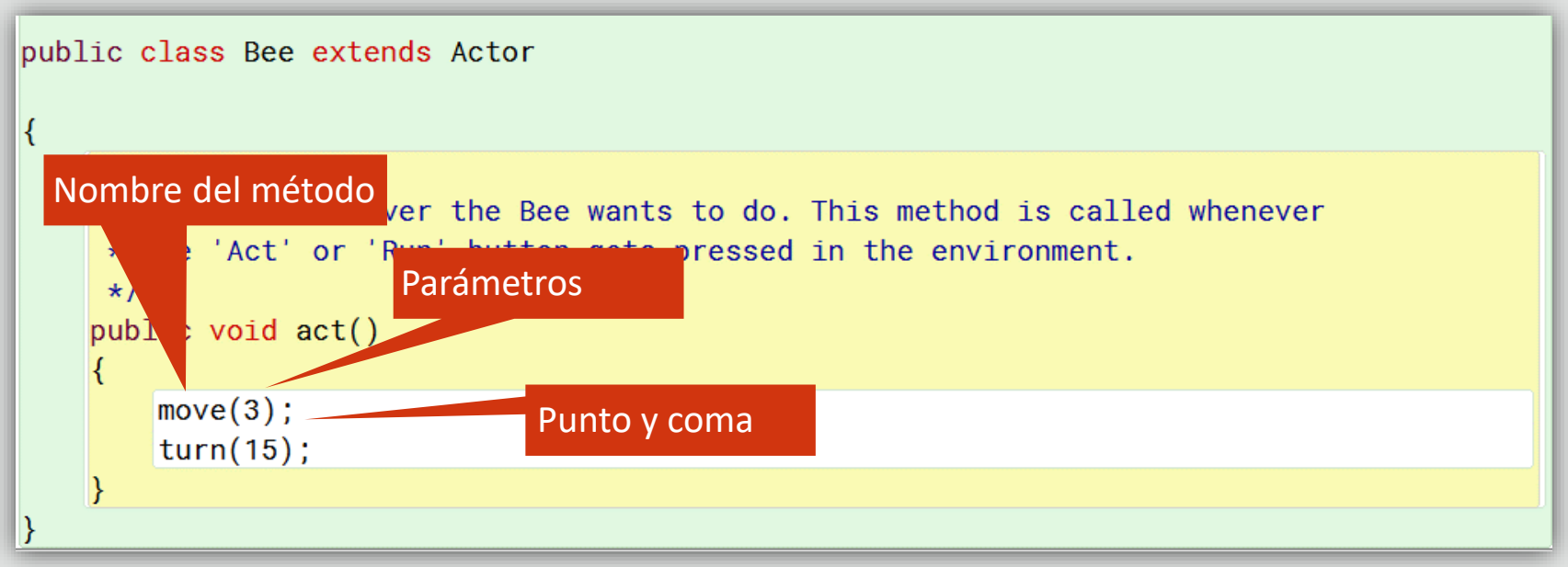
Componentes de la llamada al método

- Componentes de la llamada al método:
  - Tipo de retorno
    - Tipo de dato del valor de retorno
    - Los tipos de retorno void no necesitan variables ni devuelven datos
  - Nombre del método
  - Lista de parámetros para indicar el tipo de argumentos que llamar, si es necesario
  - Punto y coma para marcar el final de la llamada al método

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

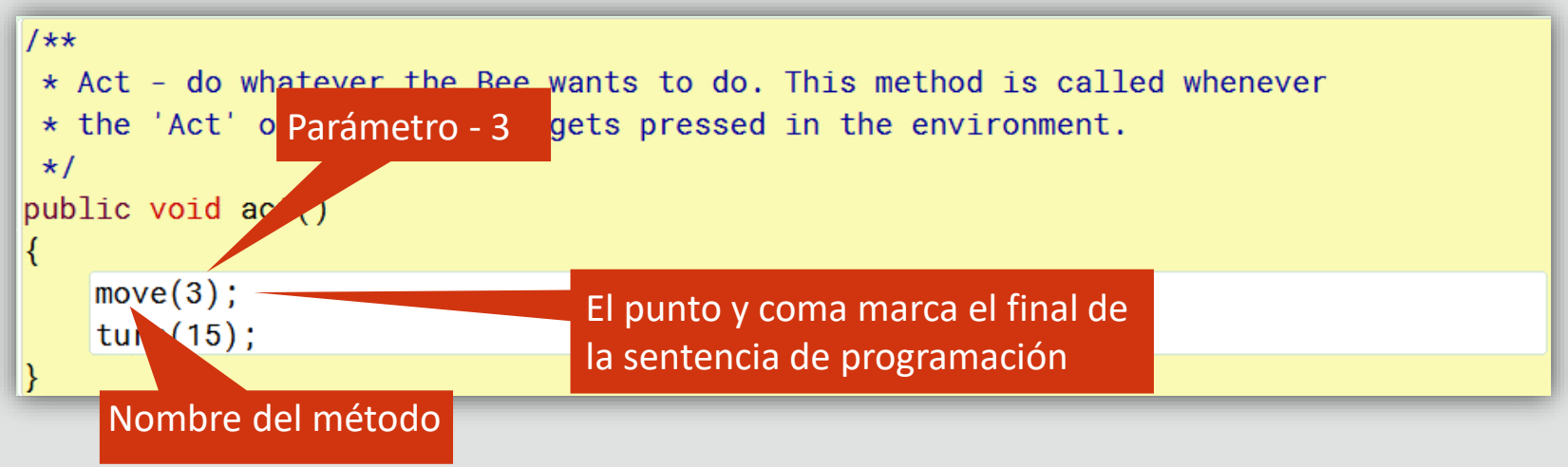
Ejemplo 1 de llamada a métodos

- Cada método se escribe en el espacio entre los corchetes



Ejemplo 2 de llamada a métodos

- La primera llamada de método se escribe en el cuerpo del método act() y termina con un punto y coma
- Cada llamada al método adicional se escribe directamente debajo, hasta que todos los métodos se introducen en el espacio entre los corchetes

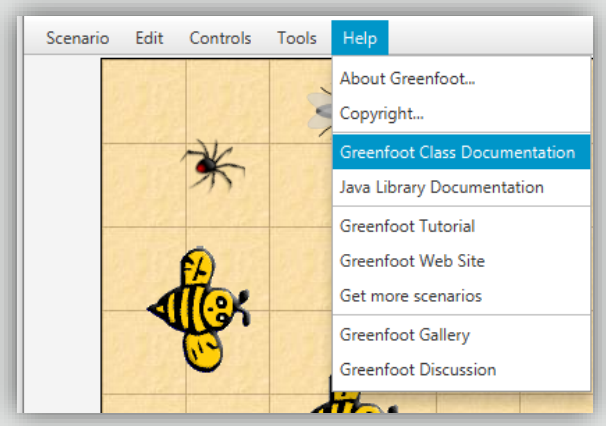


Métodos que indican a los objetos que realicen acciones

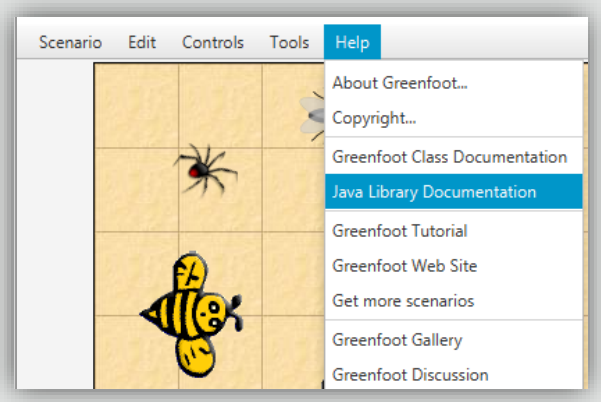
Nombre del método	Description
void move(int distance)	Asigna al objeto una serie de pasos para mover, o la orden de simplemente moverse cuando se hace clic en los botones Act o Run.
void turn(int amount)	Asigna al objeto una serie de grados para girar.
void act()	Da al objeto la oportunidad para realizar una acción en el escenario. Las llamadas a métodos se insertan en este método.
void setLocation(int x, int y)	Asigna una nueva ubicación a este objeto.
void setRotation(int rotation)	Establece una nueva rotación para este objeto.

Formas de ver los métodos heredados de una clase

- Consulte la documentación de clase de Greenfoot
  - Abra Greenfoot
  - Seleccione Help
  - Seleccione Greenfoot Class Documentation



- Vea la documentación de la biblioteca de Java
  - Abra Greenfoot
  - Seleccione Help
  - Seleccione Java Library Documentation



Tareas secuenciales

- Una única tarea, como ir a la escuela, necesita varias subtareas:
  - Despertarse
  - Ducharse
  - Cepillarse los dientes
  - Vestirse...

- Dentro de una subtarea, podría haber más subtareas (el recorrido a la escuela necesita que las piernas izquierda y derecha se muevan hacia adelante, en orden)

Métodos secuenciales

- Los métodos secuenciales son varios métodos que Greenfoot ejecuta en el orden en el que se escriben en el programa
- Estos métodos permiten que un objeto realice tareas secuenciales, como correr y luego saltar, o reproducir un sonido después de que algo explota
- Los objetos pueden programarse para realizar métodos secuenciales siempre que se haga clic en el botón Act

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

Relaciones if-then

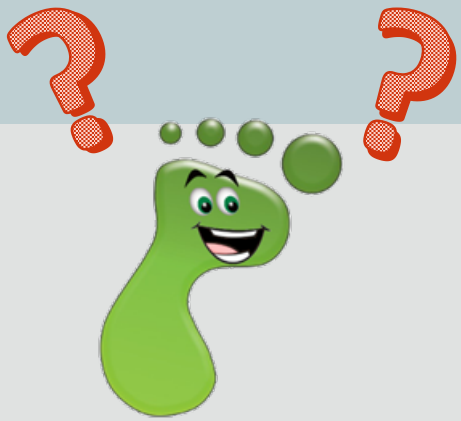
Muchas cosas que nos rodean tienen una relación causa y efecto, o relación "if-then".

- Si el teléfono móvil suena, responde
- Si no es así, no responde
- Si una flor empieza a marchitarse, la riega
- Si la flor parece sana, no la riega

Sentencias de toma de decisiones if

- Una sentencia IF se escribe para indicar al programa que ejecute un conjunto de sentencias de programación solo cuando una determinada condición sea verdadera

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



ORACLE Academy

Componentes de la sentencia de toma de decisiones if

- La sentencia if contiene una condición, que es una expresión verdadera o falsa, y una o más llamadas a métodos que se ejecutan si se cumple la condición

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

Condición

Llamadas a métodos

Ejemplo de sentencia de toma de decisiones if

- En el siguiente ejemplo:
  - Las teclas de flecha izquierda y derecha del teclado hacen que el objeto gire a la izquierda y a la derecha
  - Si la condición es false, las llamadas de métodos definidas en la sentencia IF no se ejecutan
  - El método move se ejecuta independientemente de la sentencia IF

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

Método isKeyDown

- El método isKeyDown es un método de Greenfoot preexistente que escucha para determinar si se pulsa la tecla de un teclado durante la ejecución del programa
- Este método se llama en una clase con una notación de puntos

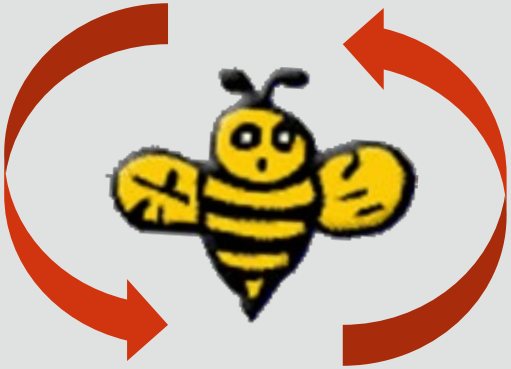
Cuando un método no está en la clase o lo hereda la clase que va a programar, especifique la clase u objeto que tiene el método antes del nombre del método, después, un punto, y luego, el nombre del método. Esta técnica se denomina notación de puntos.

Orientación de objetos en el mundo real

- A medida que avanzamos sobre el mundo en que vivimos, es importante que conozcamos nuestra orientación o sentido de la dirección
  - Al conducir el vehículo, siempre se necesita saber si el coche está en el carril correcto de la carretera
  - Cuando un avión vuela, tiene que saber dónde está ubicado en relación con otros aviones, de forma que no se produzca una colisión
  - Al introducir la ubicación en un mapa en un teléfono móvil, recibe las coordenadas que le indicarán dónde se encuentra, y la dirección

Mostrar la orientación de un objeto

- Los métodos pueden decirnos cómo se coloca un objeto en el mundo, con respecto a sí mismo y a otros objetos
- Puede llamar a un método:
  - Con un tipo de dato específico, como booleano, que pregunta al objeto sobre su orientación
  - En el entorno para saber cómo el objeto está orientado en el escenario

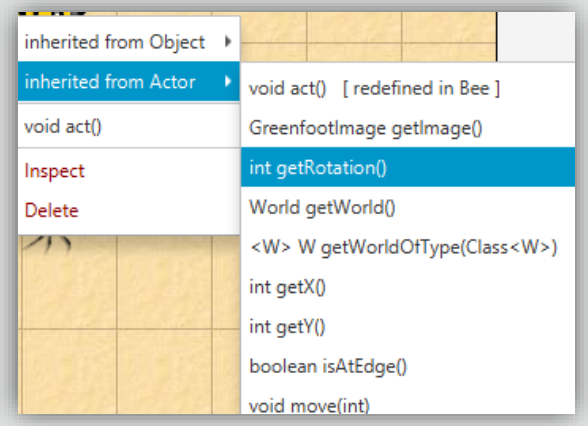


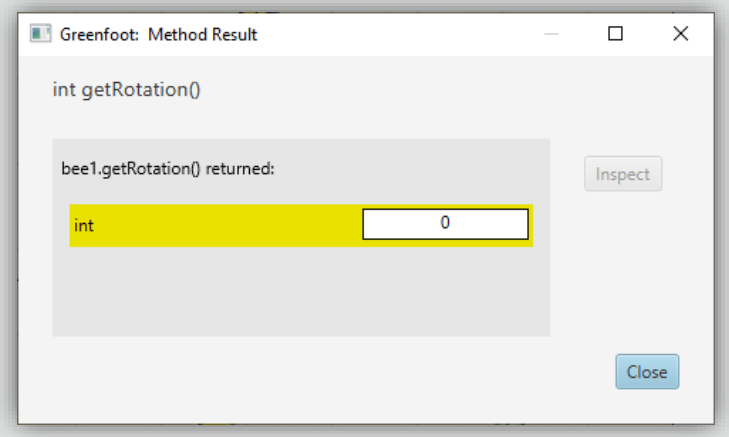
Métodos que devuelven información sobre la orientación de un objeto

Nombre del método	Description
int getRotation()	Devuelve el giro actual del objeto.
World getWorld()	Devuelve el mundo en el que está el objeto actualmente.
int getX()	Devuelve la coordenada x de la ubicación actual del objeto.
int getY()	Devuelve la coordenada y de la ubicación actual del objeto.

Pasos para llamar a un método que muestra la orientación de un objeto.

- Haga clic con el botón secundario en la instancia en el mundo
- Seleccione Inherited en Actor para ver sus métodos
- Llame (seleccione) a un método con un tipo de dato concreto para preguntar al objeto su orientación
- Se mostrará el resultado del método
- Anote el valor devuelto y, a continuación, haga clic en Close





Terminología

- Entre los términos clave utilizados en esta lección se incluyen:
  - Descripción de clase
  - Comentarios
  - Sentencias de toma de decisiones if
  - Llamada a un método
  - Análisis orientado a objetos
  - Métodos secuenciales

Inténtelo

Examinar métodos

Vaya a la API de Greenfoot y vea los métodos disponibles para las distintas clases de Greenfoot. La API de Greenfoot se encuentra en: http://www.greenfoot.org/files/javadoc/

Localice tres métodos en la clase Actor que le permitan saber más acerca de la orientación de un objeto.

Vaya a la clase World de la API de Greenfoot. ¿Qué método le indica el número de objetos en un escenario?

Vaya a la clase GreenfootImage en la API de Greenfoot. ¿Qué método indica la altura de una imagen?

Summary

- En esta lección, debe haber aprendido lo siguiente:
  - Demostrar los cambios en el código fuente para llamar a métodos mediante programación
  - Demostrar los cambios de código fuente para escribir una sentencia de toma de decisiones if
  - Describir un método para mostrar la orientación de objetos


ORACLE
Academy