



# ORACLE

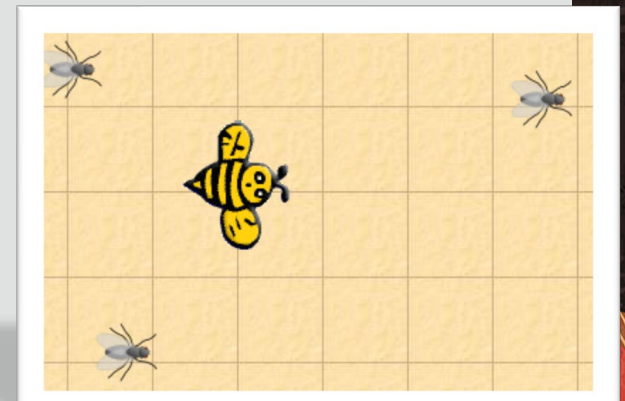
## Academy



# Crear programas de Java con Greenfoot

## Lección 6

### Aleatorización y constructores



```
if (Greenfoot.getRandomNumber(100) < 10)
{
    turn(Greenfoot.getRandomNumber(20));
}
```

# Objetivos

- En esta lección se abordan los siguientes objetivos:
  - Crear comportamientos aleatorios
  - Definir los operadores de comparación
  - Crear sentencias de control if-else
  - Crear una instancia de una clase
  - Reconocer y describir la notación de puntos



# Método getRandomNumber()

- El método getRandomNumber() es un método estático que devuelve un número aleatorio entre cero y un parámetro límite
- Este método se utiliza para eliminar la previsibilidad de un programa
- Firma del método

```
public static int getRandomNumber(int limit)
```

# Notación de puntos

- Las subclases nuevas que creamos no heredan el método `getRandomNumber()`
- Para recuperar este método de la clase de Greenfoot hay que usar la notación de puntos
- Ejemplo:

```
public void act()  
{  
    Greenfoot.getRandomNumber(20);  
}
```

Cuando quiera usar un método no heredado por la clase que está programando, especifique la clase u objeto que contiene el método antes del nombre del método, seguido de un punto y del nombre del método. Esta técnica se denomina notación de puntos.





# Formato de la notación de puntos

- El formato del código con notación de puntos incluye:
  - El nombre de la clase u objeto al que pertenece el método
  - Un punto
  - El nombre del método al que queremos llamar
  - Una lista de parámetros
  - Un punto y coma

```
className.methodName (parameters);  
objectName.methodName (parameters);
```



# Ejemplo de notación de puntos

- El método `getRandomNumber()` que se ve abajo:
  - Llama a un número aleatorio entre 0 y hasta 15, pero sin incluirlo
  - Devuelve un número aleatorio entre 0 y 14

```
public void act()  
{  
    Greenfoot.getRandomNumber(15);  
}
```

# La API de Greenfoot

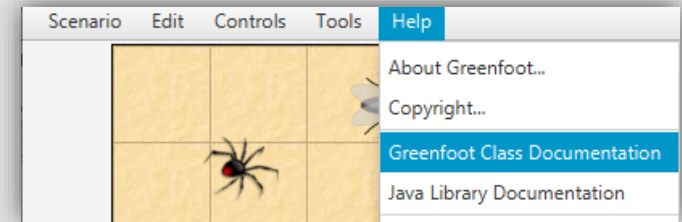
- Remite a la interfaz de programación de aplicaciones (API) de Greenfoot en busca de otros métodos de llamada usando la notación de puntos

La API de Greenfoot muestra todos los métodos y clases disponibles en Greenfoot.

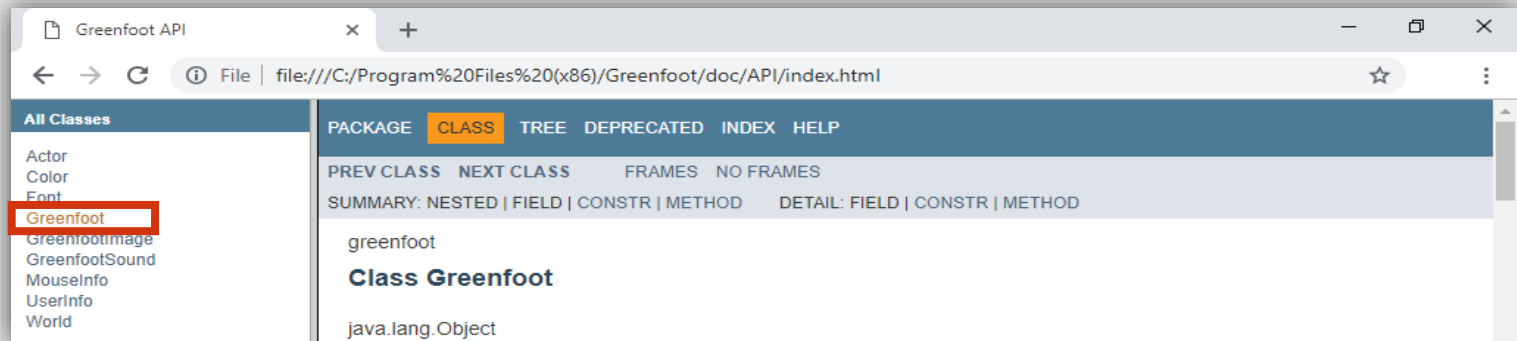


# Pasos para ver los métodos en la clase de Greenfoot

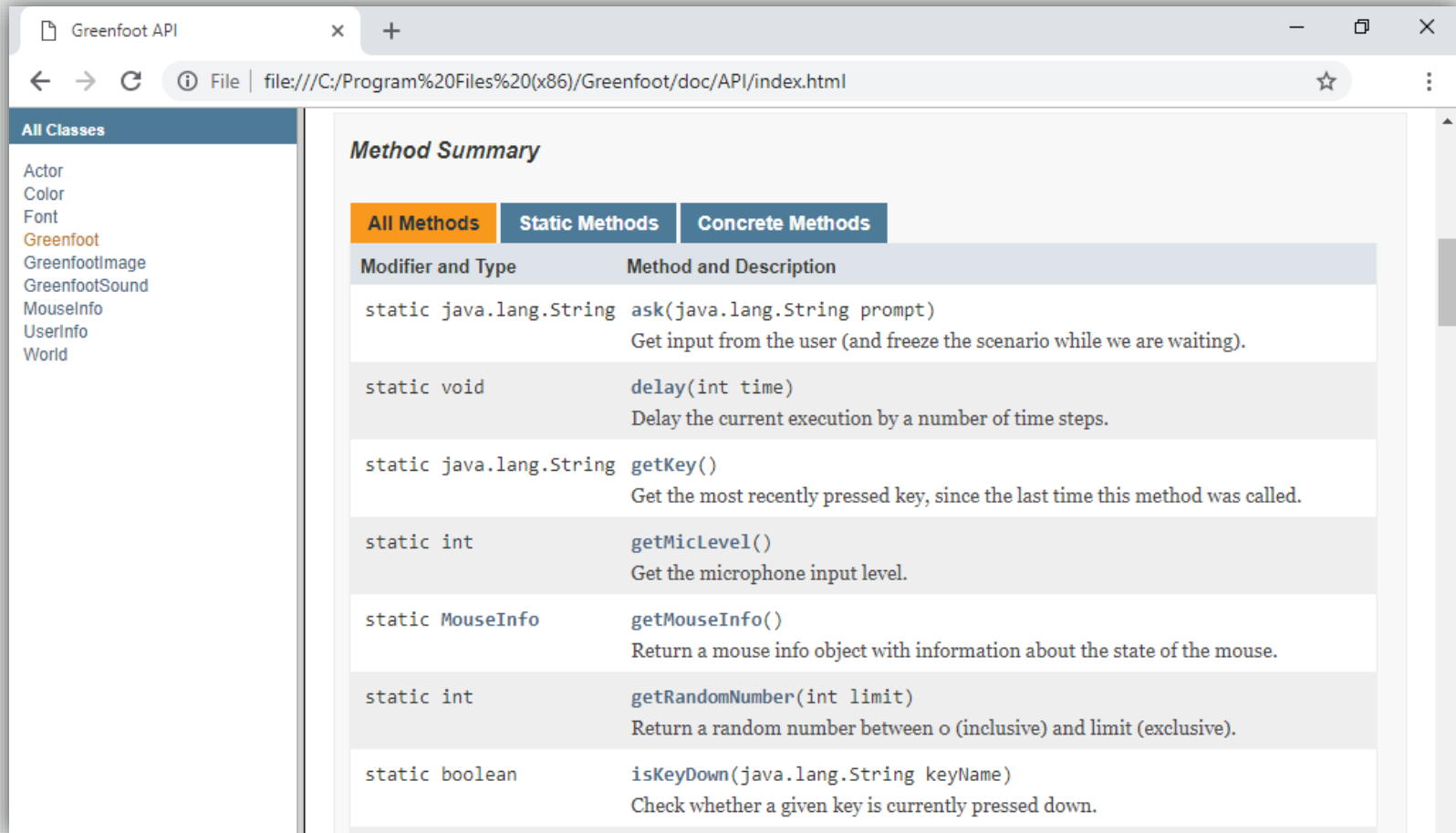
- En el entorno de Greenfoot, seleccione el menú Help
- Seleccione Greenfoot Class Documentation



- Haga clic en la clase de Greenfoot
- Revise las firmas y las descripciones del método



# Interfaz de la API de Greenfoot



The screenshot shows a web browser window displaying the Greenfoot API documentation. The browser's address bar shows the file path: `file:///C:/Program%20Files%20(x86)/Greenfoot/doc/API/index.html`. The page has a sidebar on the left titled "All Classes" listing various classes: Actor, Color, Font, Greenfoot, GreenfootImage, GreenfootSound, MouseInfo, UserInfo, and World. The main content area is titled "Method Summary" and features three tabs: "All Methods", "Static Methods", and "Concrete Methods". The "All Methods" tab is selected, showing a table of methods.

Modifier and Type	Method and Description
static java.lang.String	<code>ask(java.lang.String prompt)</code> Get input from the user (and freeze the scenario while we are waiting).
static void	<code>delay(int time)</code> Delay the current execution by a number of time steps.
static java.lang.String	<code>getKey()</code> Get the most recently pressed key, since the last time this method was called.
static int	<code>getMicLevel()</code> Get the microphone input level.
static MouseInfo	<code>getMouseInfo()</code> Return a mouse info object with information about the state of the mouse.
static int	<code>getRandomNumber(int limit)</code> Return a random number between 0 (inclusive) and limit (exclusive).
static boolean	<code>isKeyDown(java.lang.String keyName)</code> Check whether a given key is currently pressed down.



# Operadores de comparación

- Los operadores de comparación se usan para comparar un valor aleatorio con otro valor en una sentencia de control
- El siguiente ejemplo determina si el número aleatorio es inferior a 20
- Si es así, entonces el objeto gira diez grados

```
public void act()
{
    if (Greenfoot.getRandomNumber(100) < 20)
    {
        turn(10);
    }
}
```

Los operadores de comparación son símbolos que comparan dos valores.



# Símbolos de los operadores de comparación

Símbolo	Description
<	Menor que
>	Mayor que
<=	Menor o igual que
>=	Mayor o igual que
==	Igual
!=	Diferente

# Usar un comportamiento aleatorio para resolver un problema con el juego

- Problema:

- Un objeto Fly debe moverse de forma aleatoria para que al objeto controlado con el teclado, Bee, le resulte más difícil capturarlo

- Solución:

- El objeto Fly debería girar un poco mientras se mueve
- Para programar esta solución, gire el objeto Fly un número aleatorio de grados, hasta 20, el 10 % del tiempo mientras se mueve

```
public void act()
{
    if (Greenfoot.getRandomNumber(100) < 10)
    {
        turn(Greenfoot.getRandomNumber(20));
    }
}
```

# Formato de un comportamiento aleatorio

- La siguiente sentencia de programación incluye:
  - Sentencia de control IF con el método getRandomNumber()
    - Parámetro límite de 100
    - Operador de comparación <
    - Número 10 para limitar el rango de valores para devolver 0-9
  - Método body con sentencia para indicar que el objeto debería virar hasta 20 grados si la condición se cumple

```
if (Greenfoot.getRandomNumber(100) < 10)
{
    turn(Greenfoot.getRandomNumber(20));
}
```



# Formato de un comportamiento aleatorio

- El problema del ejemplo anterior es que el movimiento del objeto Fly siempre es dextrógiro
- Esto se puede modificar cambiando el parámetro en el método turn, como se muestra más abajo
- Esto generará un número aleatorio entre -45 y +44
- No hay que olvidar que el movimiento generado por un valor negativo es levógiro

```
public void act()
{
    move(1);
    if (Greenfoot.getRandomNumber(100) < 10)
    {
        turn(Greenfoot.getRandomNumber(90)-45);
    }
}
```



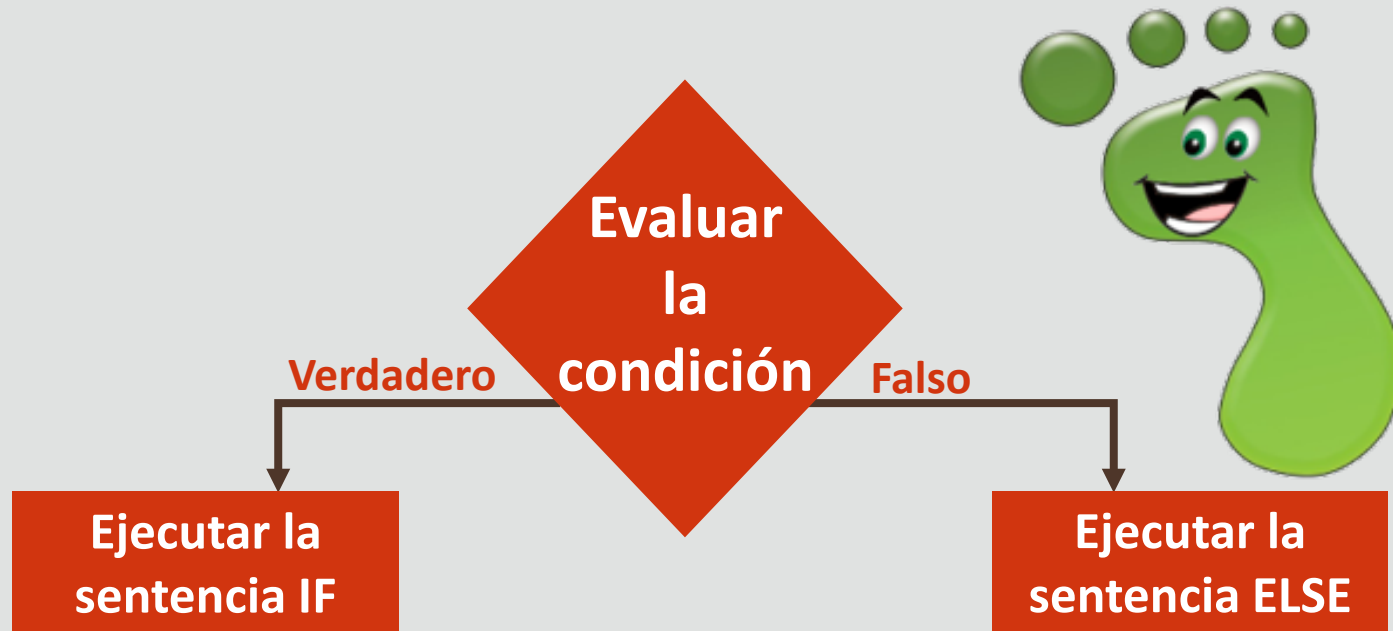
# Comportamiento condicional

- Las instancias pueden ser programadas para que se comporten de una forma determinada si no se cumple una condición usando una sentencia if-else
- Por ejemplo, si una instancia está programada para girar el 6 % del tiempo, ¿qué hace durante el 94 % restante?

Una sentencia if-else ejecuta su primer segmento de código si se cumple una condición, y el segundo segmento de código si una condición no se cumple, pero no ambas.



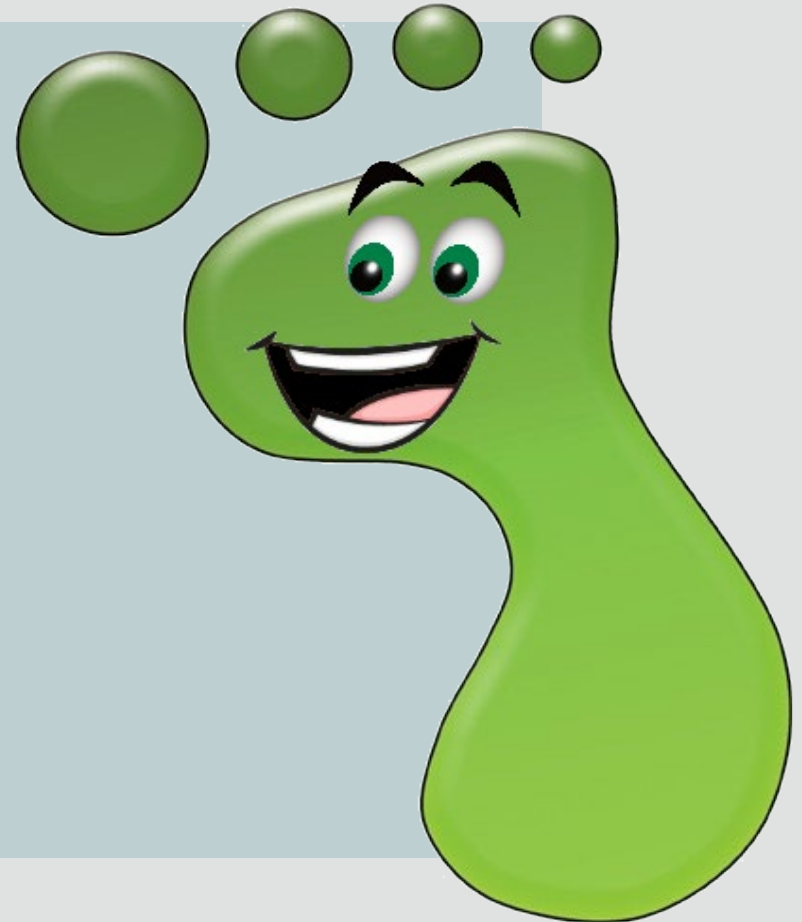
# Ejecución de la sentencia if-else





# Formato de una sentencia if-else

```
if (condition)
{
    statements;
}
else
{
    statements;
}
```



# Ejemplo de sentencia if-else

Si se selecciona un número aleatorio entre 0 y 6, vira 10 grados. De lo contrario, vira 5 grados.

```
public void act()
{
    move(1);
    if (Greenfoot.getRandomNumber(100) < 7)
    {
        turn(10);
    }
    else
    {
        turn(5);
    }
}
```

# Automatizar la creación de instancias

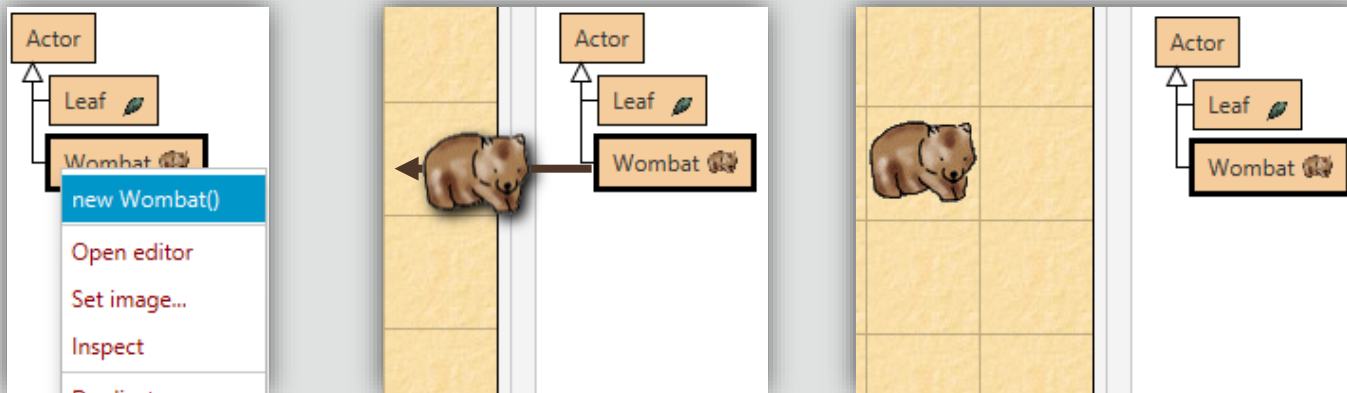
- Con la subclase World se pueden programar instancias Actor para que aparezcan automáticamente en el mundo cuando se inicie un escenario
- En Greenfoot, el comportamiento por defecto de las instancias es el siguiente:
  - La instancia de la subclase World se añade automáticamente al entorno después de la compilación o la inicialización del escenario
  - Las instancias de la subclase Actor debe añadirlas el jugador manualmente



# Creación automática de instancias en un escenario

- Problema:

- Cuando se inicia un escenario Greenfoot (como leaves-and-wombats), el jugador debe añadir las instancias manualmente para visualizar el juego



- Solución:

- Programar las instancias para que se añadan automáticamente al mundo cuando se inicia el escenario

# Código fuente de la clase “World”

- El nombre por defecto de la subclase “World” es “MyWorld”

```
public class MyWorld extends World
{
    /**
     * Constructor for objects of class MyWorld.
     *
     */
    public MyWorld()
    {
```

- Para identificar claramente el valor “World” actual, puede que sea útil cambiar el nombre
- Actualice todas las incidencias de “MyWorld” en el código fuente a “BeeWorld”

```
public class BeeWorld extends World
{
    /**
     * Constructor for objects of class BeeWorld.
     *
     */
    public BeeWorld()
    {
```



# Código fuente de la clase World

- Para entender cómo automatizar la creación de instancias Actor, es necesario conocer la estructura del código fuente de clase World
- El constructor World se usa para automatizar la creación de instancias Actor cuando se inicia el escenario

```
public class BeeWorld extends World
{
    /**
     * Constructor for objects of class BeeWorld.
     *
     */
    public BeeWorld()
    {
        // Create a new world with 600x400 cells with a cell size of 1x1 pixels.
        super(600, 400, 1);
    }
}
```

Cabecera de la clase

Comment

Constructor



# Constructores

- Constructores:
  - Definen el tamaño y la resolución de la instancia
  - No tienen tipo de retorno
  - Se llaman igual que la clase
  - Por ejemplo, un constructor del objeto World se llama World

Un constructor es un tipo especial de método que se ejecuta automáticamente cuando se crea una instancia nueva de una clase.



# Ejemplo de constructor World

- El constructor de este ejemplo construye la instancia de la superclase World como sigue:
  - Tamaño:  $x = 600$ ,  $y = 400$
  - Resolución: 1 píxel por celda
  - La palabra clave `super` en el cuerpo del constructor llama al constructor de la superclase World en cualquier instancia de la subclase BeeWorld

```
public BeeWorld()  
{  
    // Create a new world with 600x400 cells with a cell size of 1x1 pixels.  
    super(600, 400, 1);  
}
```

Size


Resolución



# Crear instancias Actor de forma automática

- Con el método `addObject()`, este constructor `World` añade un nuevo objeto `Bee` en unas coordenadas `X` e `Y` previamente especificadas

```
/**
 * Constructor for objects of class BeeWorld.
 *
 */
public BeeWorld()
{
    // Create a new world with 600x400 cells with a cell size of 1x1 pixels.
    super(600, 400, 1);
    addObject (new Bee(), 150, 100);
}
```



# Método addObject()

- El método addObject() es un método de la clase World que añade un objeto nuevo al mundo en unas coordenadas X e Y determinadas
- Consta de:
  - Palabra clave new que le indique a Greenfoot que cree un objeto nuevo de determinada clase
  - Parámetros del método
    - Objeto llamado de la clase Actor
    - Valor entero de la coordenada X
    - Valor entero de la coordenada Y
- Definición del método:

```
void addObject(Actor object, int x, int y)
```



# Palabra clave new

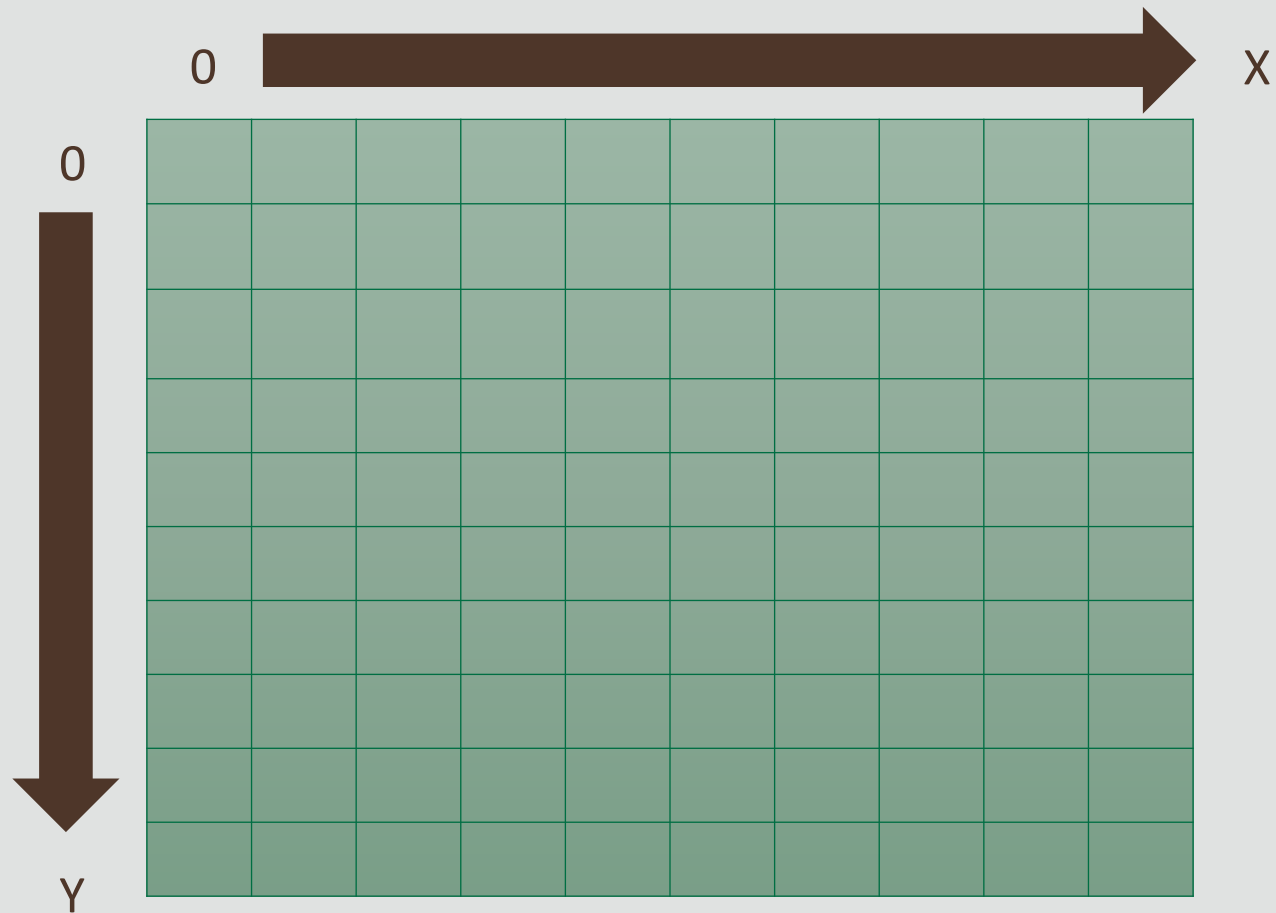
- La palabra clave new crea instancias nuevas de clases existentes
- Comienza con la palabra clave new, seguida del constructor al que se quiere llamar
  - La lista de parámetros transfiere argumentos (valores) al constructor necesarios para iniciar las variables de las instancias del objeto
  - El constructor por defecto tiene una lista de parámetros vacía y establece las variables de las instancias del objeto con sus valores por defecto



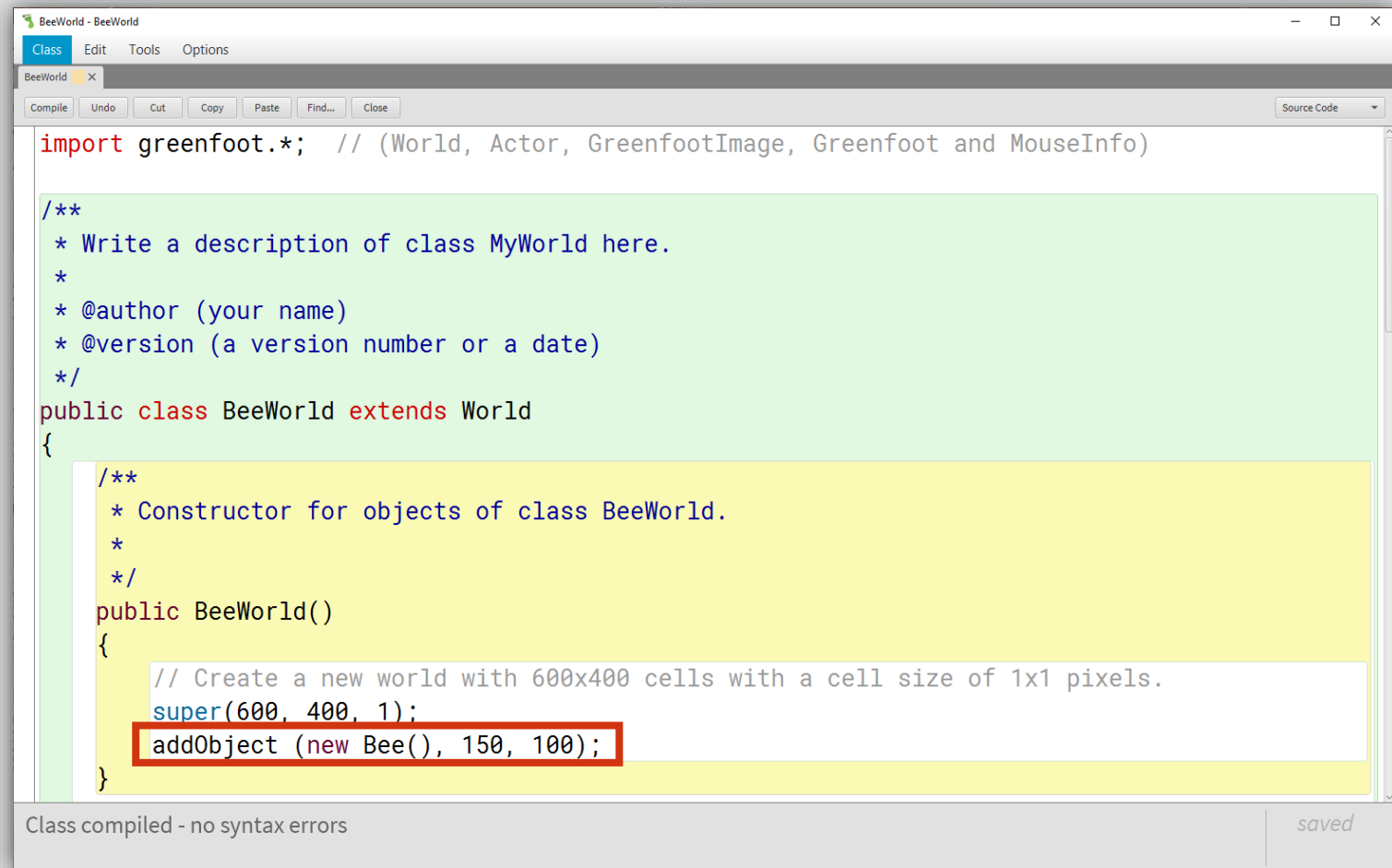
**new** ConstructorName()



# Sistema de coordenadas del objeto World de Greenfoot



# Añadir objetos mediante el ejemplo del constructor World



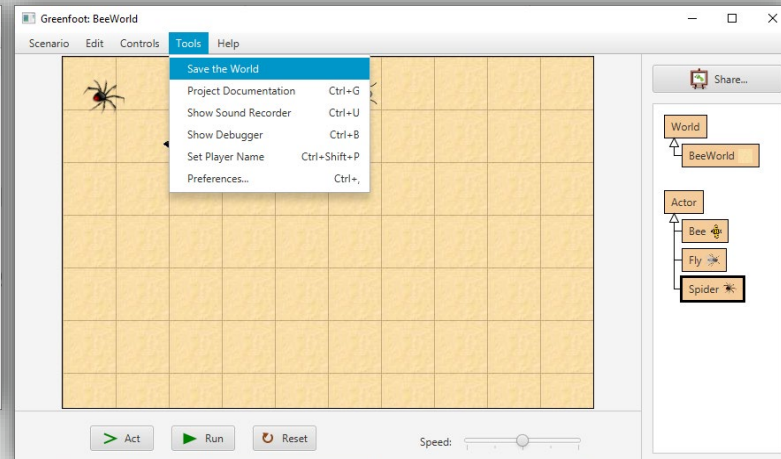
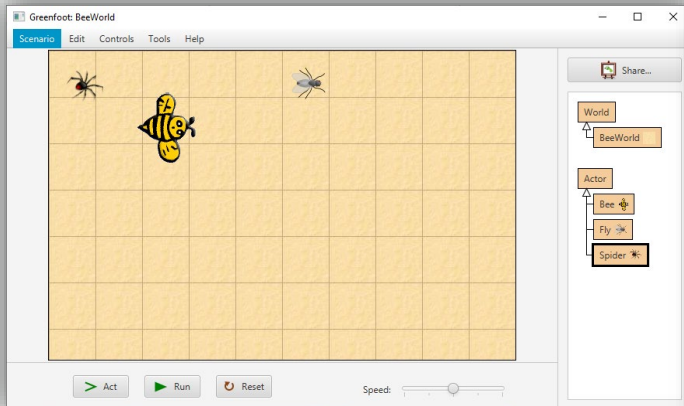
```
import greenfoot.*; // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/**
 * Write a description of class MyWorld here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class BeeWorld extends World
{
    /**
     * Constructor for objects of class BeeWorld.
     *
     */
    public BeeWorld()
    {
        // Create a new world with 600x400 cells with a cell size of 1x1 pixels.
        super(600, 400, 1);
        addObject (new Bee(), 150, 100);
    }
}
```

Class compiled - no syntax errors

# Save the World

- Greenfoot ofrece otro método para configurar la ubicación inicial de los objetos Actor: usando "Save the World"
- Podemos colocar nuestros actores en una ubicación inicial en todo el mundo de forma manual
- Después, seleccionamos Tools -> Save the World



# Save the World

- Esto crea un método nuevo en el mundo llamado `prepare()` y también una llamada para esto en el constructor
- El método `prepare()` creará instancias del actor y las añadirá al mundo, en las posiciones donde usted las ubicó
- Esto es útil si va a poner muchos objetos

```
public BeeWorld()  
{  
    // Create a new world with 600x400  
    super(600, 400, 1);  
    addObject (new Bee(), 150, 100);  
    prepare();  
}
```

```
/**  
 * Prepare the world for the start of the program.  
 * That is: create the initial objects and add them to the world.  
 */  
private void prepare()  
{  
    Fly fly = new Fly();  
    addObject(fly, 334, 42);  
    Spider spider = new Spider();  
    addObject(spider, 46, 48);  
}
```

# Terminología

- Términos clave utilizados en esta lección:
  - Operadores de comparación
  - Constructor
  - Notación de puntos
  - Palabra clave new

# Inténtelo

## Crear comportamientos aleatorios

Esta actividad requiere empezar con el archivo del proyecto guardado en el tema anterior, FrogFly\_L3T3.

**FrogFly\_L3T3.zip**

### Instrucciones:

Abra el editor de código de la clase Fly. En lugar de que el objeto Fly vire siempre un número determinado de grados, debería virar un número aleatorio entre 0 y un parámetro límite de 20. Programe este comportamiento con el método act(), sustituyendo el método turn que ya hay.

Abra el editor de código de la clase Fly de nuevo. Cambie el parámetro límite del método de 20 a 15. Compile el código.

Guarde el escenario como FrogFly\_L6T4, ejecute el escenario y observe el cambio en el movimiento de las moscas.

# Inténtelo

## Programar la sentencia de la decisión IF

Esta actividad requiere empezar con el archivo del proyecto guardado en el tema anterior, FrogFly\_L6T4.

**FrogFly\_L6T4.zip**

### Instrucciones:

Escriba una sentencia IF-ELSE para programar que las hormigas se muevan 3 y sigan hasta 20 grados el 10 % del tiempo, o para que se muevan 5 y viren hasta 10 grados.

Compile y ejecute el escenario.

Guarde el escenario como FrogFly\_L6T5.

# Inténtelo

## Programar la sentencia de la decisión IF

Esta actividad requiere empezar con el archivo del proyecto guardado en el tema anterior, FrogFly\_L6T5.

**FrogFly\_L6T5.zip**

### Instrucciones:

Modifique el constructor World para cambiar el tamaño del mundo a 700 x 500, con un tamaño de celda de 1 píxel.

Compile el escenario. Observe que el tamaño del mundo ha cambiado.

Guarde el escenario como FrogFly\_L6T6.

# Inténtelo

## Programar la sentencia de la decisión IF

Esta actividad requiere empezar con el archivo del proyecto guardado en el tema anterior, FrogFly\_L6T6.

**FrogFly\_L6T6.zip**

### Instrucciones:

Añada las siguientes instancias al objeto World: 5 Rock, 1 Frog, 4 Ant, y 5 Fly.

Haga clic con el botón derecho en el mundo y, a continuación, seleccione Save the World.

Compile el escenario. Observe que las instancias se añaden automáticamente al mundo.

Guarde el escenario como FrogFly\_L6T7.

# Summary

- En esta lección, debe haber aprendido lo siguiente:
  - Crear comportamientos aleatorios
  - Definir los operadores de comparación
  - Crear sentencias de control if-else
  - Crear una instancia de una clase
  - Reconocer y describir la notación de puntos
  - Función Save the World



ORACLE  
Academy

