
ORACLE
Academy

Creación de programas Java con Greenfoot

Lección 9

Uso de imágenes y animaciones

ORACLE
Academy

Objetivos

- En esta lección se abordan los siguientes objetivos:
  - Crear un objeto usando un constructor
  - Escribir sentencias de programación para utilizar la palabra clave new
  - Definir el objetivo y la sintaxis de una variable
  - Reconocer la sintaxis para definir y probar las variables
  - Escribir sentencias de programación para alternar entre dos imágenes

Constructores

- Cuando se crea y se compila una nueva subclase World, Greenfoot ejecuta un constructor que crea una instancia de la misma para que se muestre en el escenario
- Los constructores configuran la instancia y establecen un estado inicial, como el tamaño y la resolución de la instancia
  - Los constructores no tienen un tipo de retorno
  - Su nombre, inmediatamente después de la palabra "public", es el mismo que la clase en la que se han definido

Los constructores son métodos especiales que se ejecutan automáticamente cuando se crea una nueva instancia de la clase.

Parámetros de constructor

- Los parámetros de un constructor permiten la transferencia de los valores iniciales de una instancia en el constructor
- Estos parámetros:
  - Solo están disponibles para la instancia creada por el constructor
  - Tienen un alcance restringido que se limita al momento en el que se declaró el constructor
  - Tienen una duración restringida que se limita a la ejecución única del constructor
  - Desaparecen una vez que el constructor ha finalizado la ejecución
  - Son variables válidas siempre que exista la instancia

Ejemplo de constructor

- Este constructor en la subclase World utiliza la palabra clave super() para transferir los valores de altura, anchura y resolución del mundo a la instancia

```
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);

    }
```

Ejemplo de parámetros

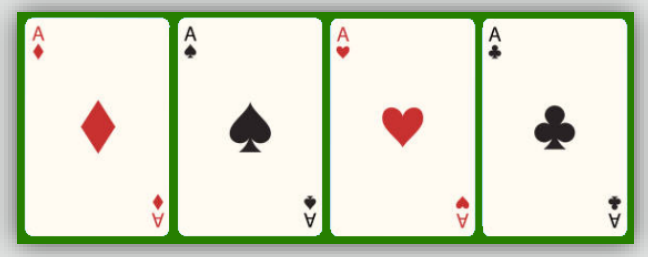
- Para cambiar el tamaño de un tablero de juego, modifique los argumentos en el parámetro del constructor
- Este ejemplo hace que el mundo sea cuadrado en lugar de rectangular, cambiando el límite de la coordenada x a 400

```
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);

    }
```

Crear automáticamente instancias Actor

- Escriba código en el constructor World para agregar automáticamente instancias de Actor al juego cuando se inicializa el escenario
- De este modo, no es necesario que el jugador tenga que agregar instancias manualmente antes de que comience el juego
- Por ejemplo, en un juego de correspondencias, las cartas se deben mostrar automáticamente en el escenario cuando se inicia la partida



Código para crear instancias automáticamente

- El código en el constructor World incluye los siguientes componentes:
  - Sentencia super() con el tamaño del mundo como argumentos
  - Método addObject() con los siguientes argumentos:
    - Palabra clave new, seguida del nombre de la clase, que indica al constructor que debe agregar una nueva instancia de esa clase
    - Las coordenadas X e Y donde se debe posicionar la nueva instancia en el mundo

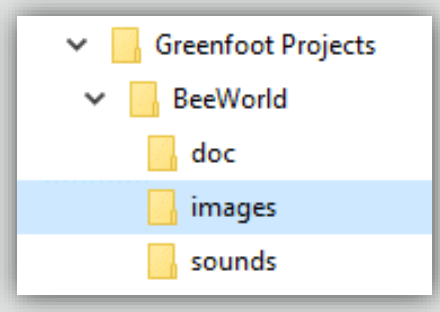
```
public BeeWorld()
{
    // Create a new world with 600x400 cells with a cell size of 1x1 pixels.
    super(600, 400, 1);
    addObject (new Bee(), 150, 100);
}
```

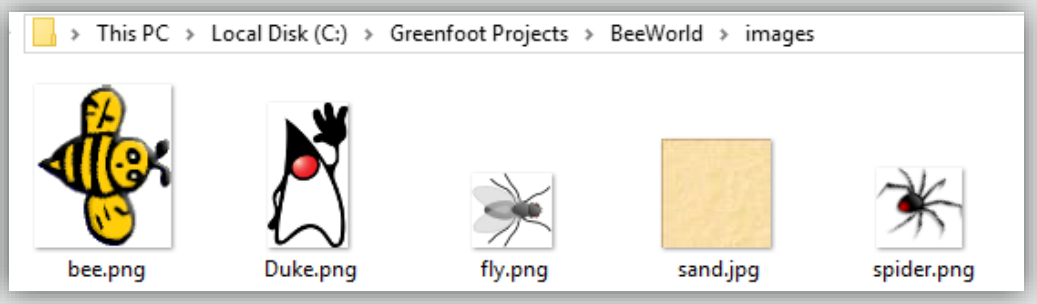
Instancias de actor de Greenfoot

- Alternar entre dos imágenes que tienen un aspecto ligeramente distinto proporciona a una instancia la sensación de movimiento
- Instancias de actor de Greenfoot:
  - Recibir y albergar una imagen de su clase
    - La imagen fue asignada a la clase cuando se creó la clase
  - Tienen la capacidad de albergar varias imágenes
  - Puede programarse para cambiar la imagen que muestran en cualquier momento

Clase GreenfootImage

- La clase GreenfootImage permite a los actores de Greenfoot mantener su imagen visible albergando un objeto de tipo GreenfootImage
- Esta clase se utiliza para ayudar a la clase a obtener y manipular distintos tipos de imágenes
- Las imágenes que utilizará esta clase deben existir previamente en la carpeta de imágenes del escenario





Constructor para obtener el nuevo objeto de imagen

- Cree un constructor que recupere un nuevo objeto de imagen a partir de un archivo al crear una instancia de una clase
- El siguiente constructor de ejemplo crea la nueva imagen y la adjunta a la clase Actor

```
/**
 * Bee - sets the initial values of the bee
 */
public Bee()
{
    setImage(new GreenfootImage("bee.png"));
}//end constructor
```

Palabra clave new

Nombre del archivo de imagen como argumentos en la lista de parámetros

Método setImage

Clase GreenfootImage

Asignación de una nueva imagen a una clase

- La siguiente sentencia crea el nuevo objeto de imagen a partir del archivo de imagen con nombre
- Cuando se inserta en el código fuente de la clase, el objeto de esta imagen está listo para que lo use la clase
- La sentencia se ejecuta del siguiente modo:
  - Se crea el objeto GreenfootImage en primer lugar
  - Se ejecuta la llamada al método setImage(), transfiriendo el objeto de imagen recién creado como un argumento a la lista de parámetros

```
public Bee()
{
    setImage(new GreenfootImage("bee.png"));
}//end constructor
```

Ejemplo de asignación de una imagen

- El método setImage() asigna la imagen en el archivo "bee.png" a la clase Actor
- Cada vez que se agrega una instancia de esta clase al escenario, muestra la imagen "bee.png"

```
/**
 * Bee - sets the initial values of the bee
 */
public Bee()
{
    setImage(new GreenfootImage("bee.png"));
}//end constructor
```

Crea la nueva imagen.

Nombre de archivo de imagen como argumento.

Permite que la clase Actor utilice el objeto de imagen. Espera la imagen como parámetro.

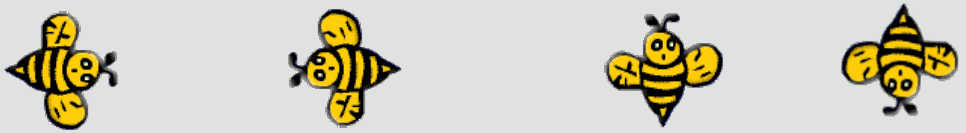
Imagen de la clase Greenfoot.

Motivos por los que las instancias albergan varias imágenes

- Es posible que desee que una instancia albergue y acceda a varias imágenes:
  - Para que parezca que cambian de color
  - Para que parezca que cambian de un tipo de objeto a otro
  - Por ejemplo, cambie un conejo por una tortuga como por arte de magia
  - Para que parezca que se mueven:
    - Para caminar: Cambie de un objeto con la pierna izquierda extendida, a un objeto con la pierna derecha extendida
    - Para dar vuelta una carta: Cambie una carta en blanco por una carta que no esté en blanco
    - Para volar: Cambie unas alas extendidas por unas alas plegadas

Acceso a varias imágenes

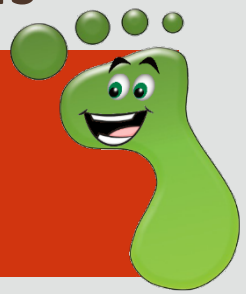
- Por ejemplo, una instancia podría acceder a dos imágenes, cada una con una posición ligeramente distinta de las alas, por lo que la instancia agita las alas mientras se mueve
- Para lograr este movimiento:
  - Cree dos imágenes de la instancia, cada una con una posición de las alas ligeramente distinta
  - Guarde las dos imágenes en la instancia, de modo que pueda acceder a ellas varias veces mientras el objeto se mueve
  - Codifique la clase para alternar entre las dos imágenes que se muestran

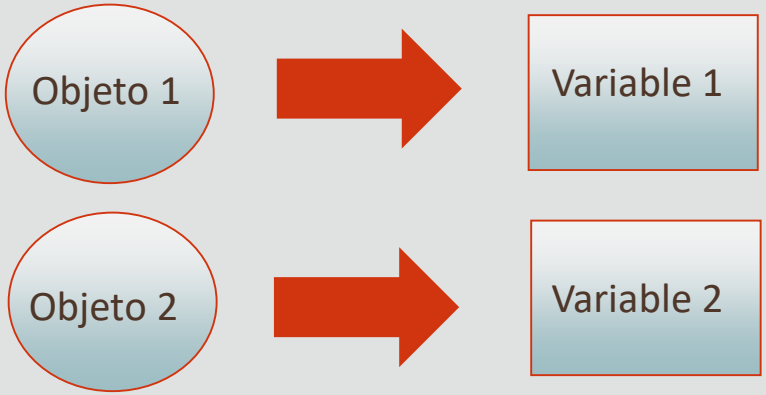


Variables

- Utilice variables de clase para guardar los dos objetos de imagen
- Esto permite a la clase acceder fácilmente a las variables para utilizarlas dentro de las instancias

Se ha declarado una variable en una clase. Se utiliza para almacenar información para su uso posterior o para transferir información. Puede almacenar objetos o valores.





Formato de la variable

- El formato de la variable incluye:
  - Tipo de datos: Tipo de datos que se almacenan en la variable
  - Nombre de variable: Una descripción de la finalidad de la variable para que se pueda hacer referencia a la misma más adelante

```
private variable-type variable-name;
```

- En este ejemplo, el nombre de la variable es image1 y el tipo de variable es GreenfootImage

```
private GreenfootImage image1;
```

Declaración de variables

- Declare variables antes de los constructores y los métodos
- El formato para declarar una variable incluye:
  - Palabra clave private para indicar que la variable solo está disponible dentro de la clase Actor
  - Clase a la que pertenece la imagen
  - Marcador de posición para la variable en el que se almacenará la imagen

```
public class Bee extends Actor

{
    private GreenfootImage image1;
    private GreenfootImage image2;

    /**
     * Bee - sets the initial values of the bee
```

Variables

Sentencias Assignment

- Es necesaria una asignación para almacenar objetos en una variable
- Cuando se asigna un objeto a una variable, la variable contiene una referencia a ese objeto
- Una sentencia assignment:
  - Almacena el objeto o valor en una variable
  - Se escribe con un símbolo igual
- Formato:

```
variable = expression;
```

Componentes de la sentencia assignment

- Una sentencia assignment incluye:
  - Variable: Nombre de la variable para almacenar un objeto o valor
  - El símbolo igual, que es el símbolo de asignación
  - Expresión:
    - Nombre del objeto o valor que se va a asignar
    - Una instrucción de que el objeto o valor es nuevo
    - La clase a la que pertenece la imagen
- Ejemplo:

```
public Bee()
{
    image1 = new GreenfootImage("bee.png");
    image2 = new GreenfootImage("bee2.png");
 }//end constructor
```

Inicialización de imágenes o valores

- La inicialización es el proceso de establecimiento de la instancia y sus valores iniciales
- Cuando la clase crea instancias nuevas, cada instancia contiene una referencia a las imágenes o valores que contienen las variables
- Instrucciones:
  - La firma no incluye un tipo de retorno
  - El nombre del constructor es el mismo que el nombre de la clase
  - El constructor se ejecuta automáticamente para transferir la imagen o valor en la instancia cuando se crea una instancia de la clase

Ejemplo de constructores de actor

- El siguiente constructor de actor indica a Greenfoot que cree automáticamente una nueva instancia Bee y que inicialice, o asigne, dos variables a la instancia
- La última línea del constructor, setImage(image1), indica que la primera variable se debe mostrar cuando se agrega la instancia en el escenario

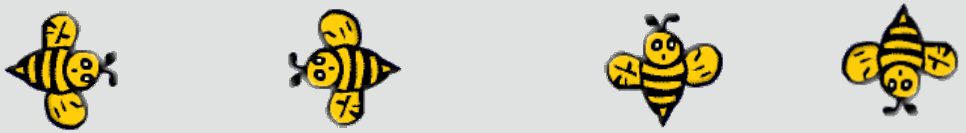
```
private GreenfootImage image1;
private GreenfootImage image2;

/**
 * Bee - sets the initial values of the bee
 */
public Bee()
{
    image1 = new GreenfootImage("bee.png");
    image2 = new GreenfootImage("bee2.png");
    setImage(image1);
 }//end constructor
```

ORACLE Academy

Probar valores de variables

- Una vez que la clase ha inicializado las dos variables con las imágenes, programe la instancia para que cambie automáticamente la imagen que muestra mientras se mueve
- Debido a que estas imágenes alternan con cada movimiento, hacen que la instancia parezca más animada
- Es posible programar el cambio entre imágenes sin tener que escribir muchas líneas de código que asocia cada imagen a cada movimiento



Escribir las acciones en pseudocódigo

- Identificar las acciones que se programan escribiéndolas en pseudocódigo
- El pseudocódigo expresa las tareas u operaciones para que las instancias utilicen una mezcla de lenguaje Java y palabras en inglés
- Esto permite entender mejor el comportamiento de las instancias antes de escribir el código real

Ejemplo de pseudocódigo

- Image1 se muestra cuando se crea la instancia
- Cuando la abeja realiza el siguiente movimiento, se debe mostrar image2 y viceversa
- Esto se expresa como una sentencia if-else

```
if (current image displayed is image1) then
      use image2 now
else
      use image1 now
```

Operador '=='

- Las sentencias de programación que indican a la instancia que alterne entre imágenes contienen:
  - Sentencia if-else
  - Operador '==' (dos signos igual)
- Operador '==':
  - Se utiliza en una sentencia if para probar si dos valores son iguales
  - Compara un valor con otro
  - Devuelve un resultado booleano (true o false)
- Recuerde que '=' es el símbolo de asignación, no el símbolo para probar si dos valores son iguales

Componentes de la sentencia if-else

- Componentes de la sentencia if-else:
  - El método getImage recibe actual de la instancia
  - El operador '==' comprueba que el valor que ha mostrado la instancia es igual a image1
    - si es igual, se muestra image2
    - De lo contrario, se muestra image1

Ejemplo de sentencia if-else

- La siguiente sentencia if-else se escribe en el método act para que la instancia alterne la visualización de dos imágenes mientras avanza

```
/**
 * 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()
{
    if(getImage()==image1)
        setImage(image2);
    else
        setImage(image1);
    //endif
    handleMovement();
    catchFly();
    turnAtEdge();
}//end method act
```

Ejemplo de sentencia if-else

- También alejaremos el código de animación de su propio método para mantenerlo más limpio

```
public void act()
{
    animateBee();
    handleMovement();
    catchFly();
    turnAtEdge();
}//end method act

private void animateBee(){
    if(getImage()==image1)
       setImage(image2);
    else
       setImage(image1);
    //endif
}//end method animateBee
```

Terminología

- Entre los términos clave utilizados en esta lección se incluyen:
  - Constructor
  - Variable definida
  - Pseudocódigo

Inténtelo.

Defina una imagen con el nombre de archivo de la imagen.

Esta actividad necesita que comience con el archivo de proyecto que se ha guardado en el tema anterior FrogFly_L8T11.

FrogFly_L8T11.zip

Instrucciones:

Haga clic con el botón derecho sobre la clase Frog y seleccione set Image

Desde la lista de imágenes del escenario, seleccione la imagen frog. Luego, desde las herramientas de edición que están debajo de la lista de imágenes, seleccione Edit.

Desde el programa de dibujo o diseño gráfico de su equipo, rote o voltee la imagen para tener un total de cuatro imágenes, una para cada dirección hacia donde debe mirar la rana (izquierda, derecha, arriba y abajo).

Guarde las cuatro imágenes en la carpeta images dentro del escenario de Greenfoot Asígnele un nombre a cada una usando la dirección hacia la que mira.

Inténtelo.

Defina una imagen con el nombre de archivo de la imagen.

Instrucciones (continuación):

Abra el editor de códigos de la clase Frog. Mediante el método setLocation(), codifique la clase Frog para que se mueva hacia la izquierda, derecha, arriba o abajo al presionar las teclas de flecha.

Agregue una sentencia de programación para el cuerpo de cada sentencia if de forma que indique la imagen que debe mostrar cuando se presiona esa tecla. Por ejemplo, la sentencia de programación que mueve la rana hacia la derecha deberá mostrar la imagen de la rana mirando hacia a la derecha.

Compile el código. Ejecute el escenario y pruebe cómo funciona el código pulsando las teclas. Verifique que la imagen de la rana cambie según la dirección hacia la que debe mirar.

Guarde el escenario con el nombre FrogFly_L9T12.

Inténtelo.

Defina una imagen con el nombre de archivo de la imagen.

Esta actividad necesita que comience con el archivo de proyecto que se ha guardado en el tema anterior FrogFly_L9T12.

FrogFly_L9T12.zip

Instrucciones:

Abra el editor de códigos de la clase Frog.

Elimine el código que ahora define la imagen utilizando el nombre de archivo de la imagen. Sustitúyalo por un código que defina la imagen utilizando un objeto GreenfootImage.

Compile el código. Ejecute el escenario y verifique que las imágenes cambien cuando se presionan las teclas de flecha.

Guarde el escenario con el nombre FrogFly_L9T13.

Inténtelo.

Defina una imagen con el nombre de archivo de la imagen.

Esta actividad necesita que comience con el archivo de proyecto que se ha guardado en el tema anterior FrogFly_L9T13.

FrogFly_L9T13.zip

Instrucciones:

Abra el editor de códigos de la clase Frog.

Declare cuatro variables del tipo GreenfootImage con los nombres image1, image2, image3 e image4, donde se almacenarán las cuatro imágenes de ranas.

Compile el código.

Guarde el escenario con el nombre FrogFly_L9T14.

Summary

- En esta lección, debe haber aprendido lo siguiente:
  - Crear un objeto usando un constructor
  - Escribir sentencias de programación para utilizar la palabra clave new
  - Definir el objetivo y la sintaxis de una variable
  - Reconocer la sintaxis para definir y probar las variables
  - Escribir sentencias de programación para alternar entre dos imágenes


ORACLE
Academy