
ORACLE
Academy

Creación de Programas Java con Greenfoot

Lección 7

Definición de métodos

```
private void catchFly() {
    if (isTouching(Fly.class)) {
        removeTouching(Fly.class);
    }
}
```

ORACLE
Academy

Objetivos

- En esta lección se abordan los siguientes objetivos:
  - Describir la colocación adecuada de métodos en una superclase o subclase
  - Simplificar la programación mediante la creación y llamada a métodos definidos
  - Gestionar colisiones

Colocación efectiva de métodos

- En ocasiones, se necesitan muchas líneas de código para programar un comportamiento
  - Por ejemplo, puede que desee programar una instancia para que se coma otros objetos o que se gire cuando llegue al fin del mundo
- Puede definir nuevos métodos para simplificar la lógica, ahorrar tiempo y utilizar menos líneas de código
  - Defina (codifique) un nuevo método para una acción debajo del método act
  - Llame (utilice) al nuevo método en el método act de un mundo o actor, o bien dentro de otro método
  - Defina (codifique) el método en la superclase si desea que sus subclases hereden de forma automática el método

Métodos definidos

- Los métodos definidos son los nuevos creados por el programador
- Estos métodos:
  - Se pueden ejecutar de inmediato o almacenarse para llamarlos posteriormente
  - No cambian el comportamiento de la clase cuando se almacenan
  - Separan el código en métodos más cortos, lo cual facilita la lectura

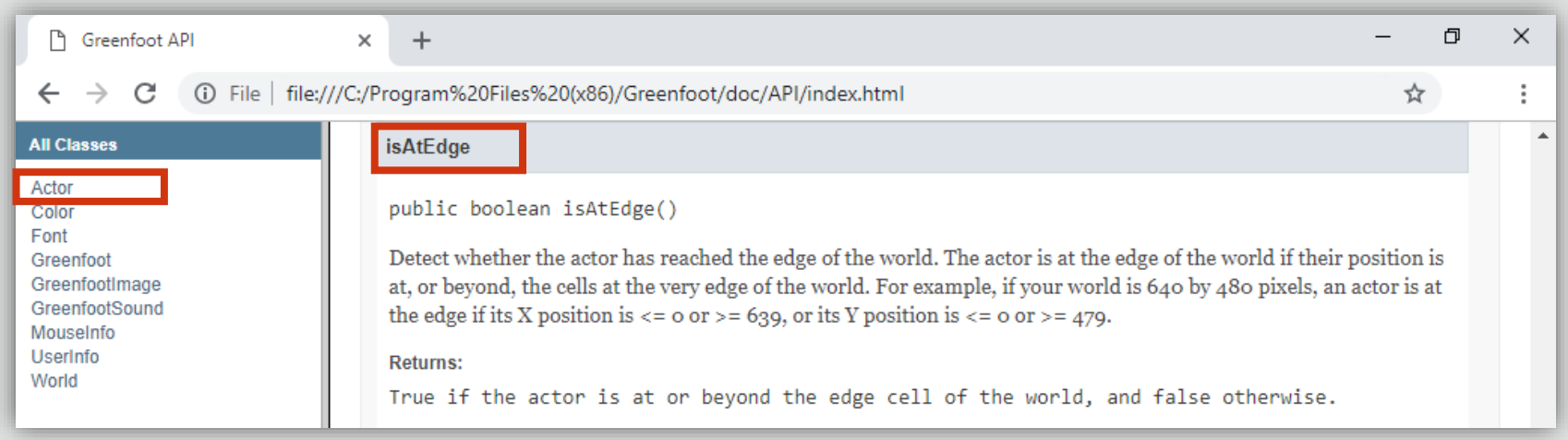
Los métodos definidos crean un nuevo método que no poseía aún una clase. Estos métodos se escriben en el código fuente de una clase debajo del método act.

Giro en el fin del mundo

- Problema:
  - Las instancias se paran y no se pueden mover cuando llegan al fin del mundo
  - Las instancias se deben girar y moverse en la dirección contraria cuando llegan al fin del mundo
- Solución:
  - Cree una subclase de Actor que defina un método que pueda detectar si el objeto está en el fin del mundo y girarse como corresponda
  - Cree subclases de la subclase Actor que deban heredar el método
  - Llame (utilice) al nuevo método en las subclases que debería poder girar y moverse cuando llegue al fin del mundo

Prueba de si un actor está en el fin del mundo

- Greenfoot tiene un método en la clase Actor denominado isAtEdge()
- que devuelve true si Actor está en uno de los bordes
- Lo podemos utilizar para detectar y, a continuación, girar los actores en lugar de que sigan colocados sobre uno de los bordes



Prueba de la posición de un objeto en el mundo

- Para probar si un objeto está cerca del fin del mundo:
  - Utilice una sentencia **if** con el método booleano isAtEdge() para probar si la condición es true o false
  - Ejemplo: Podemos girar una instancia 180 grados si está en el fin del mundo

```
public void act()
{
    move(1);
    if(isAtEdge())
    {
        turn(180);
    }
}
```

Operadores lógicos

Se pueden agregar operadores lógicos a las sentencias if/else para combinar varias expresiones booleanas en una expresión booleana.

Operador lógico	Significado	Definition
Signo de exclamación (!)	NOT	Invierte el valor de una expresión booleana (si b es true, !b es false. Si b es false, !b es true).
Ampersand doble (&&)	AND	Combina dos valores booleanos, y devuelve un valor booleano que es true solo si sus dos operandos son true.
Dos líneas (II)	OR	Combina dos variables o expresiones booleanas y devuelve un resultado que es true si uno de los dos operandos o ambos son true.

Operadores lógicos: ejemplos

- Si hay un objeto Spider en la mitad superior derecha del escenario, muévalo hacia adelante:

```
public void act()
{
    if((getX() > getWorld().getWidth()/2) && (getY()< getWorld().getHeight()/2))
    {
        move(2);
    }//endif
}
```

- Si la posición horizontal de Actor es mayor que el centro del mundo horizontal, y la posición vertical de Actor es menor que el centro del mundo vertical...
  - getX() devuelve la posición horizontal de Actor
  - getWorld().getWidth() devuelve el tamaño horizontal de World
  - getY() devuelve la posición vertical de Actor
  - getWorld().getHeight() devuelve el tamaño vertical de World

Operadores lógicos: ejemplos

```
public void act()
{
    if((getX() > getWorld().getWidth()/2) && (getY()< getWorld().getHeight()/2))
    {
        move(2);
    }//endif
}
```

Sustituya && por || para probar si un objeto Spider está en la mitad superior O en la mitad derecha del escenario.

```
public void act()
{
    if((getX() > getWorld().getWidth()/2) || (getY()< getWorld().getHeight()/2))
    {
        move(2);
    }//endif
}
```

Operadores lógicos: ejemplos

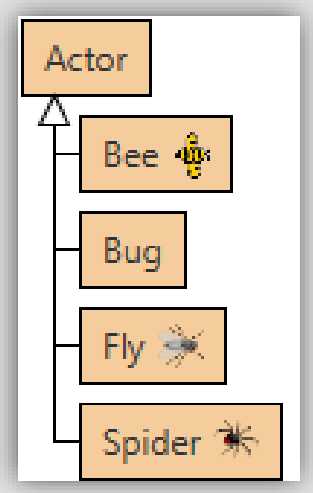
```
public void act()
{
    if((getX() > getWorld().getWidth()/2) || (getY()< getWorld().getHeight()/2))
    {
        move(2);
    }//endif
}
```

Utilice ! para mover un objeto Spider hacia delante si NO está en la mitad derecha del escenario.

```
public void act()
{
    if(!(getX() > getWorld().getWidth()/2) || (getY()< getWorld().getHeight()/2))
    {
        move(2);
    }//endif
}
```

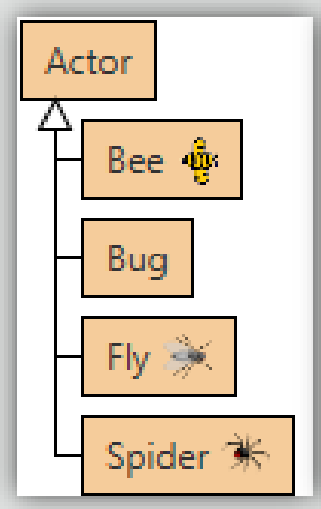
Creación de una superclase

- Para permitir que tanto Spider como Fly hereden el comportamiento de giro colocamos el código en una superclase
- El nombre de esta superclase debe ser representativo de las características comunes de las clases de las que heredará
- Crearemos una clase Bug que no tiene ninguna imagen y no tendrá instancias que actúen en el escenario, pero contendrá los métodos definidos que sus subclases heredarán
  - Haga clic con el botón derecho en Actor y seleccione New subclass
  - Asigne a la nueva clase el nombre "Bug"
  - No asigne ninguna imagen
  - Haga clic en OK

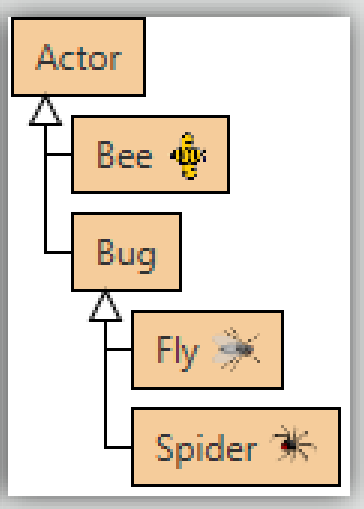


Creación de las subclases Bug

- Podríamos volver a crear nuestros objetos Spider y Fly haciendo clic con el botón derecho en Bug y seleccionando New subclass
- Pero, como los hemos creado con anteriormente, podemos modificar nuestro código fuente Spider y Fly para ampliar desde Bug en lugar de hacerlo desde Actor




```
public class Spider extends Bug
{
    /**
     * Act - do whatever the Spider wants to do. Thi
     * the 'Act' or 'Run' button gets pressed in the
     */

    public void act()
    {
        if(!(getX() > getWorld().getWidth()/2) || (g
        {
```



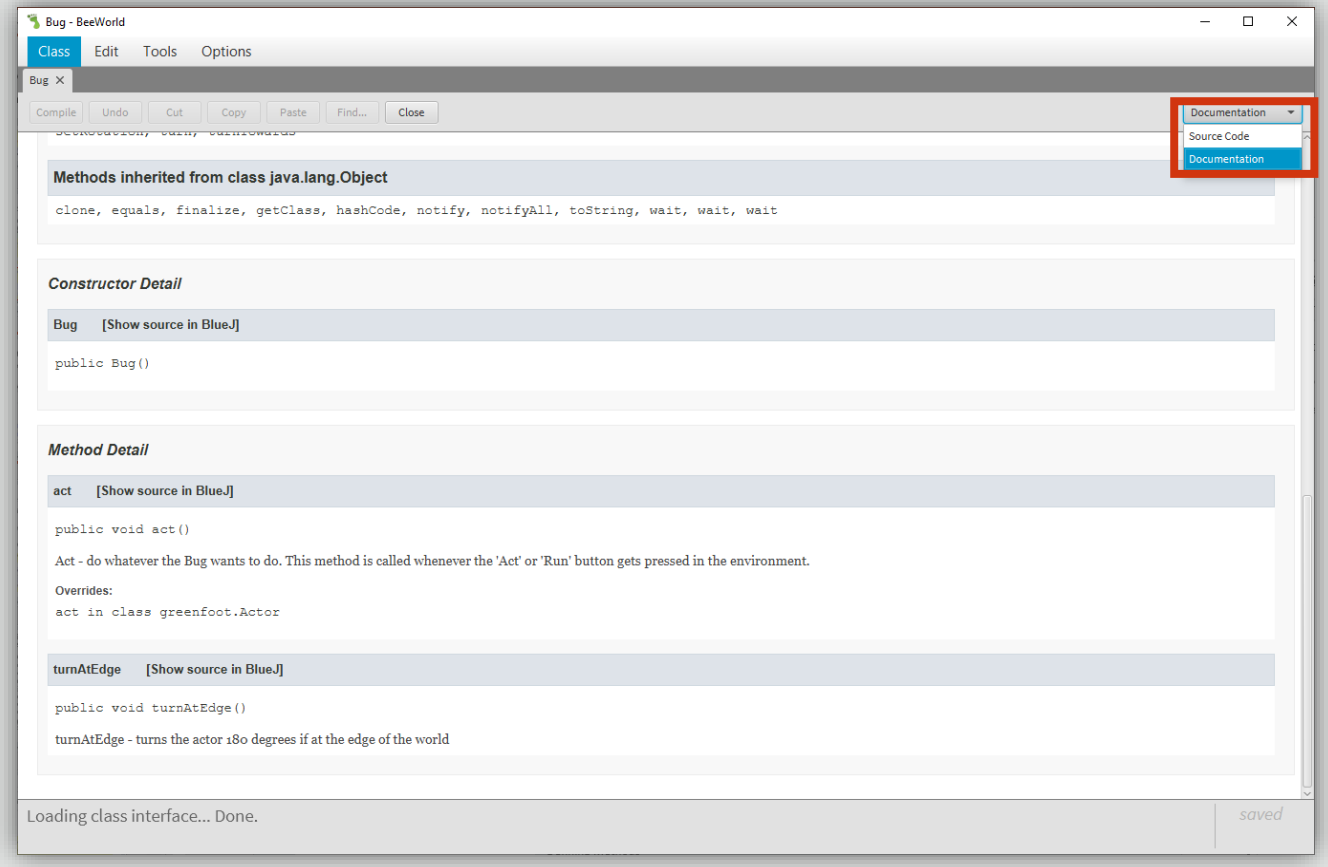
Definición del método turnAtEdge() en la superclase

Abra el editor de códigos de la clase Bug. Escriba el código para el método turnAtEdge(), bajo el método act().

```
public class Bug extends Actor
{
    /**
     * Act - do whatever the Bug 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.
    }

    /**
     * turnAtEdge - turns the actor 180 degrees if at the edge of the world
     */
    public void turnAtEdge()
    {
        if(isAtEdge())
        {
            turn(180);
        }//endif
    }
}
```

Documentación de la clase

En la documentación de la clase Bug se muestra el nuevo método después de su definición.



Llamada al método turnAtEdge() en la subclase

- Abra el editor de códigos para la subclase Fly de la lección anterior
- Agregue una llamada al método turnAtEdge() en el método act()
- Repita esta operación con la subclase Spider

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

Definición de los métodos at[left, right, top, bottom]Edge() en la clase Bee

- Para la clase Bee, escribiremos métodos que
  - determinarán el borde al que se aproxima Bee
  - moverán Bee al borde contrario (parece ajustarse)
- Por ejemplo, si Bee se mueve hacia la derecha y se acerca al borde derecho:
  - desaparecerá del borde derecho
  - volverá a aparecer en el borde izquierdo
  - seguirá volviendo a moverse hacia el borde derecho

- Para saber qué borde está tocando un actor, definiremos 4 métodos distintos, uno para cada lado

Definición del método atRightEdge() en la clase Bee

- Abra el editor de códigos de la clase Bee.
- Escriba el código para el método atRightEdge(), bajo el método act

```
/**
 * Test if we are close to the right edge of the world
 * Return true if the object is.
 */
private boolean atRightEdge()
{
    if(getX() > getWorld().getWidth() - 20)
        return true;
    else
        return false;
}//end method atRightEdge
```

Definición del método atBottomEdge() en la clase Bee

- Abra el editor de códigos de la clase Bee
- Escriba el código para el método atBottomEdge(), bajo el método act

```
}//end method act

/**
 * Test if we are close to the bottom edge of the world
 * Return true if the object is.
 */
private boolean atBottomEdge()
{
    if(getY() > getWorld().getHeight() - 20)
       return true;
    else
       return false;
}//end method atBottomEdge
```

Métodos llamados desde atRightEdge() y atBottomEdge()

- Entre los métodos utilizados en atRightEdge() se incluyen:
  - getX(): método Actor que devuelve la coordenada x de la ubicación actual del actor
  - getY(): método Actor que devuelve la coordenada y de la ubicación actual del actor
  - getWorld(): método Actor que devuelve el mundo en el que vive este actor
  - getHeight(): método de la clase World que devuelve la altura del mundo
  - getWidth(): método de la clase World que devuelve el ancho del mundo

Llamada a los métodos de la clase Bee

- Abra el editor de códigos de la clase Bee
- Cree una sentencia IF que llame al método atRightEdge() y atBottomEdge() como condición en act
- Si Bee está a la izquierda, volverá a aparecer a la derecha y viceversa

El desplazamiento siempre se realizará.

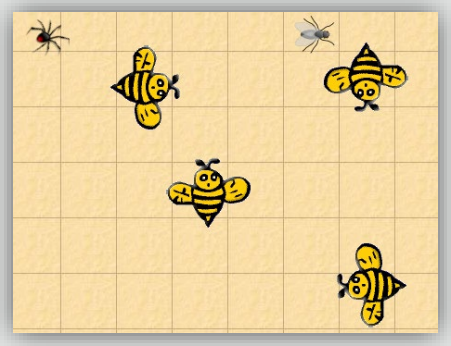
Si Bee está cerca del borde derecho, la coordenada x se definirá en 6. La posición Y actual se mantiene con el método getY().

Si Bee está cerca del borde inferior, la coordenada Y se definirá en 6. La posición X actual se mantiene con el método getX().

```
public void act()
{
    move(1);
    if(atRightEdge()){
        setLocation(6, getY());
    }else
        if(atBottomEdge()){
            setLocation(getX(), 6);
        }//endif
    //endif
}//end method act
```

Llamada a los métodos de la clase

- Abra el editor de códigos de la clase myWorld
- Si es necesario,
  - agregue cuatro objetos Bee al mundo
  - seleccione Controls -> Save the World



- En el método prepare(), agregue lo siguiente:
  - bee2.turn(90);
  - bee3.turn(-90);
  - bee4.turn(180);

```
private void prepare()
{
    Fly fly = new Fly();
    addObject(fly,334,42);
    Spider spider = new Spider();
    addObject(spider,46,48);
    Bee bee = new Bee();
    addObject(bee, 150, 100);
    Bee bee2 = new Bee();
    addObject(bee2,388,87);
    Bee bee3 = new Bee();
    addObject(bee3,220,214);
    Bee bee4 = new Bee();
    addObject(bee4,396,312);
    bee2.turn(90);
    bee3.turn(-90);
    bee4.turn(180);
```

Llamada al método en la clase

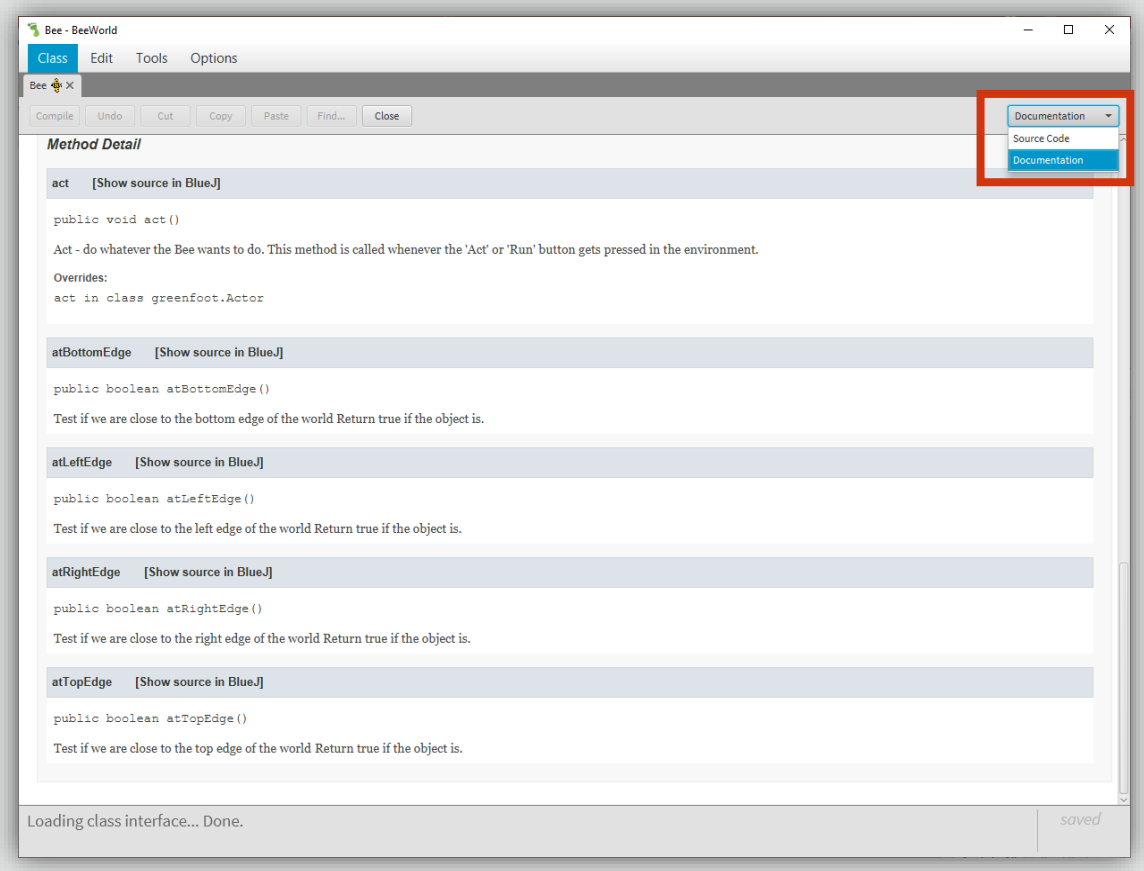
- Complete la sentencia IF para atLeftEdge() y atTopEdge()
- Pruebe el programa para asegurarse de que las instancias de Bee se separan de los bordes del mundo en dirección contraria

```
}//end method act

/**
 * Test if we are close to the top edge of the world
 * Return true if the object is.
 */
private boolean atTopEdge()
{
    if(getY() < 6)
       return true;
    else
       return false;
}//end method atTopEdge

/**
 * Test if we are close to the left edge of the world
 * Return true if the object is.
 */
private boolean atLeftEdge()
{
    if(getX() < 6)
       return true;
    else
       return false;
}//end method atLeftEdge
```

```
/**
 * 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);
    if(atRightEdge()){
        setLocation(6, getY());
    }
    else if(atBottomEdge()){
            setLocation(getX(), 6);
    }
    else if(atLeftEdge()){
            setLocation(getWorld().getWidth()-20, getY());
    }
    else if(atTopEdge()){
            setLocation(getX(), getWorld().getHeight()-20);
    }//endif
}//end method act
```

Documentación de la clase

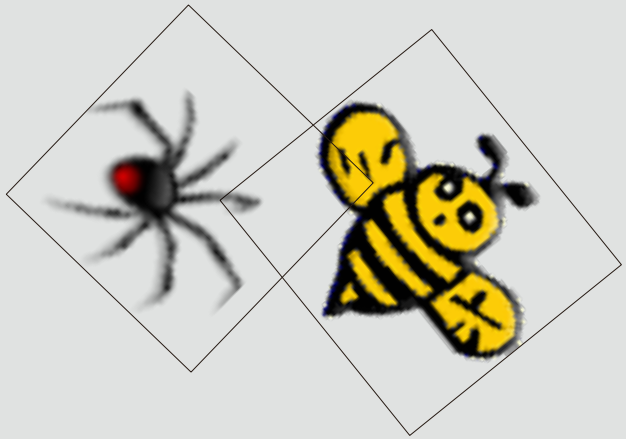
En la documentación de la clase Bee se muestra el nuevo método después de su definición.



ORACLE Academy

Colisiones

- La mayoría de los proyectos de juegos tendrán que detectar cuando dos actores entran en contacto, a lo que se denomina colisión
- En GreenFoot hay varias formas de detectar esto
- Algunas de estas son:
  - isTouching()
  - getOneIntersectingObject(Class)
  - getOneObjectAtOffset(Class)
  - getIntersectingObjects(Class)
  - getNeighbours(distance,diagonal)
  - getObjectsAtOffset(dx,dy,Class)



Colisiones

Método	¿Cuándo utilizarlo?
isTouching()	Cuando se desea detectar una colisión con un objeto
getOneIntersectingObject()	Cuando se desea devolver una referencia al objeto con el que ha colisionado. Utilice este método para realizar una acción en el objeto de la colisión.
getOneObjectAtOffset()	Igual que getOneIntersectingObject(), excepto por el hecho de que puede cambiar el lugar donde se detectará la colisión en relación con el objeto actual. Por lo que podría haber detectado la colisión antes de que ocurra. Por ejemplo, para impedir que un actor ande hacia una pared.

Método definido para eliminar objetos

- Puede escribir código en su juego para que un objeto depredador pueda comerse a sus objetos presa
- Cree un método definido en la clase Bee denominado catchfly() para que podamos eliminar las moscas que captura la abeja
- Para crear este método definido vamos a utilizar la detección de colisión más simple: isTouching()
- Este método detecta la colisión de un objeto Bee con un objeto Fly y, a continuación, lo elimina

```
private void catchFly(){
    if(isTouching(Fly.class)){
        removeTouching(Fly.class);
    }//endif
}
```

Definición del método catchfly(): alternativa

También podríamos haber utilizado getOneIntersectingObject() y accedido a una referencia al actor antes de borrarlo.

```
}//end method act

/**
 * catchFly2 - if the Bee touches a fly the fly is removed
 */
private void catchFly2(){
    Actor fly = getOneIntersectingObject(Fly.class);
    if(fly != null){
        getWorld().removeObject(fly);
    }//endif
}

/**
 * catchFly - if the Bee touches a fly the fly is removed
```

Llamada a catchfly() en el método act()

- Llame al nuevo método catchfly() en el método act() del objeto Bee
- Tenga cuidado y agregue el código fuera de la sentencia if, ya que siempre queremos que la abeja intente capturar una mosca
- Ejecute el escenario para probar el código

```
public void act()
{
    move(1);
    if(atRightEdge()){
        setLocation(6, getY());
    }
    else if(atBottomEdge()){
            setLocation(getX(), 6);
    }
    else if(atLeftEdge()){
            setLocation(getWorld().getWidth()-20, getY());
    }
    else if(atTopEdge()){
            setLocation(getX(), getWorld().getHeight()-20);
    }//endif
    catchFly();
}//end method act
```

Llamada a catchfly() en el método act()

- Cambios adicionales recomendados para el método act():
  - volver a colocar la sentencia move() del método act() en un nuevo método, denominado handleMovement()
  - volver a colocar las sentencias if del método act() que compruebe si el objeto Bee está en el borde mediante la adición de un método turnAtEdge() que prueba si Bee está en el borde

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

```
private void handleMovement(){
    move(1);
}//end method handleMovement
```

```
private void turnAtEdge(){
    if(atRightEdge()){
        setLocation(6, getY());
    }
    else if(atBottomEdge()){
            setLocation(getX(), 6);
    }
    else if(atLeftEdge()){
            setLocation(getWorld().getWidth()-20, getY());
    }
    else if(atTopEdge()){
            setLocation(getX(), getWorld().getHeight()-20);
    }//endif
}//end method turnAtEdge
```

Terminología

- Entre los términos clave utilizados en esta lección se incluyen:
  - Métodos definidos
  - Colisiones

Inténtelo.

Llamar al método atWorldEdge()

Para llevar a cabo esta actividad debe comenzar con el archivo de proyecto que se guardó en el tema anterior FrogFly_L6T7.

FrogFly_L6T7.zip

Instrucciones:

Abra el editor de códigos de la clase Ant.

Llame al método isAtEdge() desde el método act().

Abra el editor de códigos de la clase Fly.

Llame al método isAtEdge() desde el método act() para la clase Fly.

Compile el escenario. Ejecute el escenario y pruebe su funcionamiento. Asegúrese de que tanto las hormigas como las moscas giran 180 grados cuando llegan al extremo del mundo. Guarde el escenario como FrogFly_L7T8 .

Inténtelo.

Definir un método para detectar y eliminar objetos que se entrecrucen

Para llevar a cabo esta actividad debe comenzar con el archivo de proyecto que se guardó en el tema anterior FrogFly_L7T8.

FrogFly_L7T8.zip

Instrucciones:

Abra el editor de códigos de la clase Frog y declare un método eatFly() que le permitirá detectar y eliminar los objetos Fly que se entrecrucen.

Llame al método eatFly() desde el método act().

Compile el código. Ejecute el escenario para comprobar que funciona.

Mueva la rana por el mundo e intente colisionar con una mosca. Compruebe que la mosca se elimina del mundo si colisiona con la rana.

Guarde el escenario como FrogFly_L7T9.

Summary

- En esta lección, debe haber aprendido lo siguiente:
  - Describir la colocación adecuada de métodos en una superclase o subclase
  - Simplificar la programación mediante la creación y llamada a métodos definidos


ORACLE
Academy