

Programming in Java

with Greenfoot

http://www.greenfoot.org

The screenshot shows a web browser window displaying the Greenfoot website at www.greenfoot.org. The page has a green header with the title "Greenfoot" and the subtitle "Teach & Learn Java Programming". Below the header is a large image of the Greenfoot JavaWorld interface, showing a rocket ship launching from a cloud. To the right of the interface, there are three main features: "Interactive" (with a description of visual tools and programming), "Real Programming" (with a description of Java as a popular language), and "Global" (with a description of user communities). A "find out more" button is located next to the Global section. Below the main image, a banner states "Educational software designed to make learning programming easy and fun". At the bottom of the page, there are two sections: "The Software" (with icons for Windows, Mac, and Linux) and "Enter Greenfoot site" (with a link to the book "Introduction to Programming with Greenfoot"). The book cover is for "Object-Oriented Programming in Java™ with Games and Simulations". A note below the book says it's the official book used by both teachers and students. The bottom right corner features the text "The Greenroom".

Greenfoot

www.greenfoot.org

Greenfoot

Teach & Learn Java Programming

Interactive

Visual tools coupled with programming, making learning to program easier

Real Programming

Learn and teach Java, one of the most popular languages in the world

Global

Discuss, share and interact with user communities, both for learners and teachers

find out more

Educational software designed to make learning programming easy and fun

The Software

Enter Greenfoot site

Download Greenfoot for free: runs on Windows, Mac and Linux

Introduction to Programming
with **Greenfoot**

Object-Oriented Programming in Java™ with Games and Simulations

'Introduction to Programming with Greenfoot' is the official book used by both teachers and students

The Greenroom

<http://www.greenfoot.org/doc>

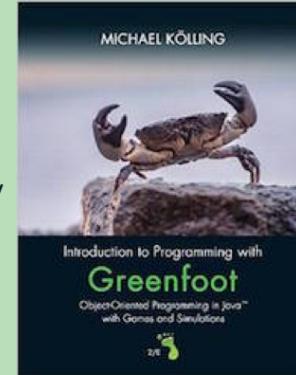
The book

'Introduction to Programming with Greenfoot' is the official book used by both teachers and students.

The second edition of the book is out now.

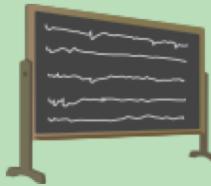
The new edition has several new chapters; new scenarios; end-of-chapter drill and practice sections added; more gradual introduction; improvement of presentation of key concepts; new Greenfoot features included; programming with Microsoft Kinect.

Please visit the book's web page for more information about this new edition.



Tutorials

Learn how to use Greenfoot and begin programming:



1. Interacting with Greenfoot
2. Movement and Key Control
3. Detecting and Removing Actors, and Making Methods
4. Saving the World, Making and Playing Sound
5. Adding a Randomly Moving Enemy
6. How to Access One Object From Another

Reference

The Greenfoot API is available online. You can also access it from within Greenfoot - use the "Greenfoot Class Documentation" menu item, from the Help menu.



Videos

The [Joy Of Code](#) is a thorough introduction to Greenfoot. A wide range of other short videos are also available.



German translations are available for some videos (thanks to Frajo Ligmann).



Greenfoot API

The Greenfoot API consists of five classes:

Actor	Actor methods are available to all actor subclasses.	MouseInfo	Provides information about the last mouse event.
World	World methods are available to the world.	GreenfootImage	For image presentation and manipulation.
Greenfoot	Used to communicate with the Greenfoot environment itself.		

Class World

World(int worldWidth, int worldHeight, int cellSize)	Construct a new world.
void act()	Act method for the world. Called once per act round.
void addObject(Actor object, int x, int y)	Add an Actor to the world.
GreenfootImage getBackground()	Return the world's background image.
int getCellSize()	Return the size of a cell (in pixels).
Color getColorAt(int x, int y)	Return the color at the center of the cell.
int getHeight()	Return the height of the world (in number of cells).
List getObjects(Class cls)	Get all the objects in the world.
List getObjectsAt(int x, int y, Class cls)	Return all objects at a given cell.
int getWidth()	Return the width of the world (in number of cells).
int numberOfObjects()	Get the number of actors currently in the world.
void removeObject(Actor object)	Remove an object from the world.
void removeObjects(Collection objects)	Remove a list of objects from the world.
void repaint()	Repaint the world.
void setActOrder(Class... classes)	Set the act order of objects in the world.
void setBackground(GreenfootImage image)	Set a background image for the world.
void setBackground(String filename)	Set a background image for the world from an image file.
void setPaintOrder(Class... classes)	Set the paint order of objects in the world.
void started()	Called by the Greenfoot system when execution has started.
void stopped()	Called by the Greenfoot system when execution has stopped.

Class Actor

```
Actor()
void act()

protected void addedToWorld(World world)

GreenfootImage getImage()
protected List
getIntersectingObjects(Class cls)
protected List getNeighbours(int distance,
boolean diagonal, Class cls)
protected List getObjectsAtOffset(int dx,
int dy, Class cls)
protected List getObjectsInRange(int r,
Class cls)
protected Actor getOneIntersectingObject
(Class cls)
protected Actor getOneObjectAtOffset
(int dx, int dy, Class cls)
int getRotation()
World getWorld()
int getX()
int getY()
protected boolean intersects(Actor other)
void setImage(GreenfootImage image)
void setImage(String filename)
void setLocation(int x, int y)
void setRotation(int rotation)
```

Construct an Actor.

The act method is called by the Greenfoot framework to give objects a chance to perform some action.

This method is called by the Greenfoot system when the object has been inserted into the world.

Return the image used to represent this Actor.

Return all the objects that intersect this object.

Return the neighbours to this object within a given distance.

Return all objects that intersect the given location (relative to this object's location).

Return all objects within range 'r' around this object.

Return an object that intersects this object.

Return one object that is located at the specified cell (relative to this object's location).

Return the current rotation of the object.

Return the world that this object lives in.

Return the x-coordinate of the object's current location.

Return the y-coordinate of the object's current location.

Check whether this object intersects another given object.

Set the image for this object to the specified image.

Set an image for this object from an image file.

Assign a new location for this object.

Set the rotation of the object.

Class Greenfoot

Greenfoot()	Constructor.
static void delay(int time)	Delay execution by a number of time steps. The size of one time step is defined by the speed slider.
static String getKey()	Get the most recently pressed key since the last time this method was called.
static MouseInfo getMouseInfo()	Return a mouse info object with information about the state of the mouse.
static int getRandomNumber(int limit)	Return a random number between 0 (inclusive) and limit (exclusive).
static boolean isKeyDown(String keyName)	Check whether a given key is currently pressed down.
static boolean mouseClicked(Object obj)	<i>True</i> if the mouse has been clicked on the given object.
static boolean mouseDragEnded(Object obj)	<i>True</i> if a mouse drag has ended.
static boolean mouseDragged(Object obj)	<i>True</i> if the mouse has been dragged on the given object.
static boolean mouseMoved(Object obj)	<i>True</i> if the mouse has been moved on the given object.
static boolean mousePressed(Object obj)	<i>True</i> if the mouse has been pressed on the given object.
static void playSound(String soundFile)	Play sound from a file.
static void setSpeed(int speed)	Set the speed of the simulation execution.
static void start()	Run (or resume) the simulation.
static void stop()	Stop the simulation.

Class MouseInfo

<code>Actor getActor()</code>	Return the actor (if any) that the current mouse behaviour is related to.
<code>int getButton()</code>	The number of the pressed or clicked button (if any).
<code>int getClickCount()</code>	The number of mouse clicks of this mouse event.
<code>int getX()</code>	The current x position of the mouse cursor.
<code>int getY()</code>	The current y position of the mouse cursor.
<code>String toString()</code>	Return a string representation of this mouse event info.

Class GreenfootImage

<code>GreenfootImage(GreenfootImage image)</code>	Create a GreenfootImage from another GreenfootImage.
<code>GreenfootImage(int width, int height)</code>	Create an empty (transparent) image with the specified size.
<code>GreenfootImage(String filename)</code>	Create an image from an image file.
<code>void clear()</code>	Clear the image.
<code>void drawImage(GreenfootImage image, int x, int y)</code>	Draw the given image onto this image.
<code>void drawLine(int x1, int y1, int x2, int y2)</code>	Draw a line, using the current drawing color, between the points (x1, y1) and (x2, y2).
<code>void drawOval(int x, int y, int width, int height)</code>	Draw an oval bounded by the specified rectangle with the current drawing color.
<code>void drawPolygon(int[] xPoints, int[] yPoints, int nPoints)</code>	Draw a closed polygon defined by arrays of x and y coordinates.
<code>void drawRect(int x, int y, int width, int height)</code>	Draw the outline of the specified rectangle.
<code>void drawString(String string, int x, int y)</code>	Draw the text given by the specified string, using the current font and color.

(continued)

Class GreenfootImage (continued)

<code>void fill()</code>	Fill the entire image with the current drawing color.
<code>void fillOval(int x, int y, int width, int height)</code>	Fill an oval bounded by the specified rectangle with the current drawing color.
<code>void fillPolygon(int[] xPoints, int[] yPoints, int nPoints)</code>	Fill a closed polygon defined by arrays of x and y coordinates.
<code>void fillRect(int x, int y, int width, int height)</code>	Fill the specified rectangle.
<code>BufferedImage getAwtImage()</code>	Return the BufferedImage that backs this GreenfootImage.
<code>Color getColor()</code>	Return the current drawing color.
<code>Color getColorAt(int x, int y)</code>	Return the color at the given pixel.
<code>Font getFont()</code>	Get the current font.
<code>int getHeight()</code>	Return the height of the image.
<code>int getTransparency()</code>	Return the transparency of the image (range 0–255).
<code>int getWidth()</code>	Return the width of the image.
<code>void mirrorHorizontally()</code>	Mirror the image horizontally (flip around the x-axis).
<code>void mirrorVertically()</code>	Mirror the image vertically (flip around the y-axis).
<code>void rotate(int degrees)</code>	Rotate this image around the center.
<code>void scale(int width, int height)</code>	Scale this image to a new size.
<code>void setColor(Color color)</code>	Set the current drawing color.
<code>void setColorAt(int x, int y, Color color)</code>	Set the color at the given pixel to the given color.
<code>void setFont(Font f)</code>	Set the current font.
<code>void setTransparency(int t)</code>	Set the transparency of the image (range 0–255).
<code>String toString()</code>	Return a string representation of this image.

Part 1 - Wombat



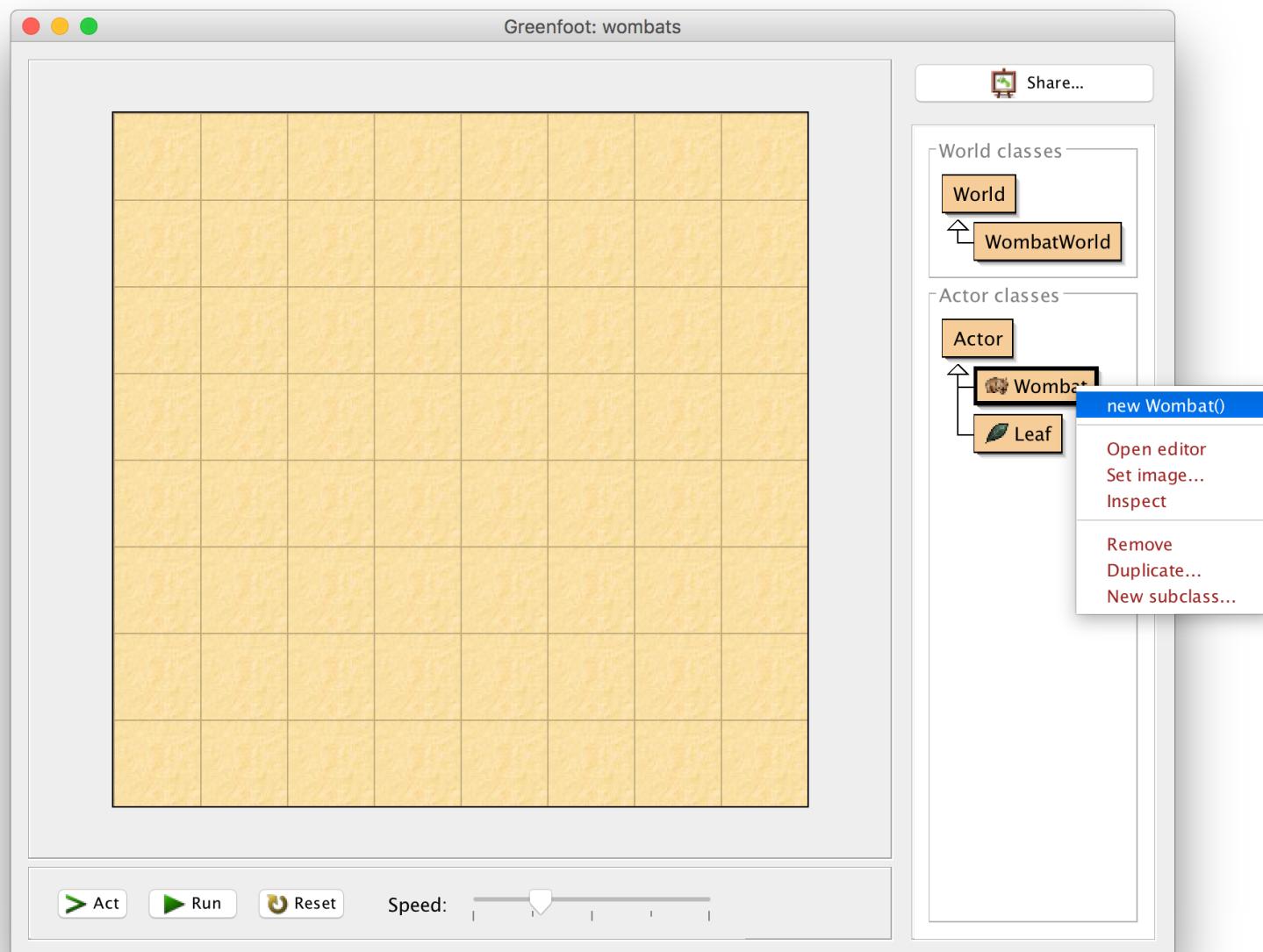
Greenfoot: wombats

The Greenfoot interface is shown, featuring a 10x10 grid world with a light brown textured background. On the right side, the class hierarchy is displayed:

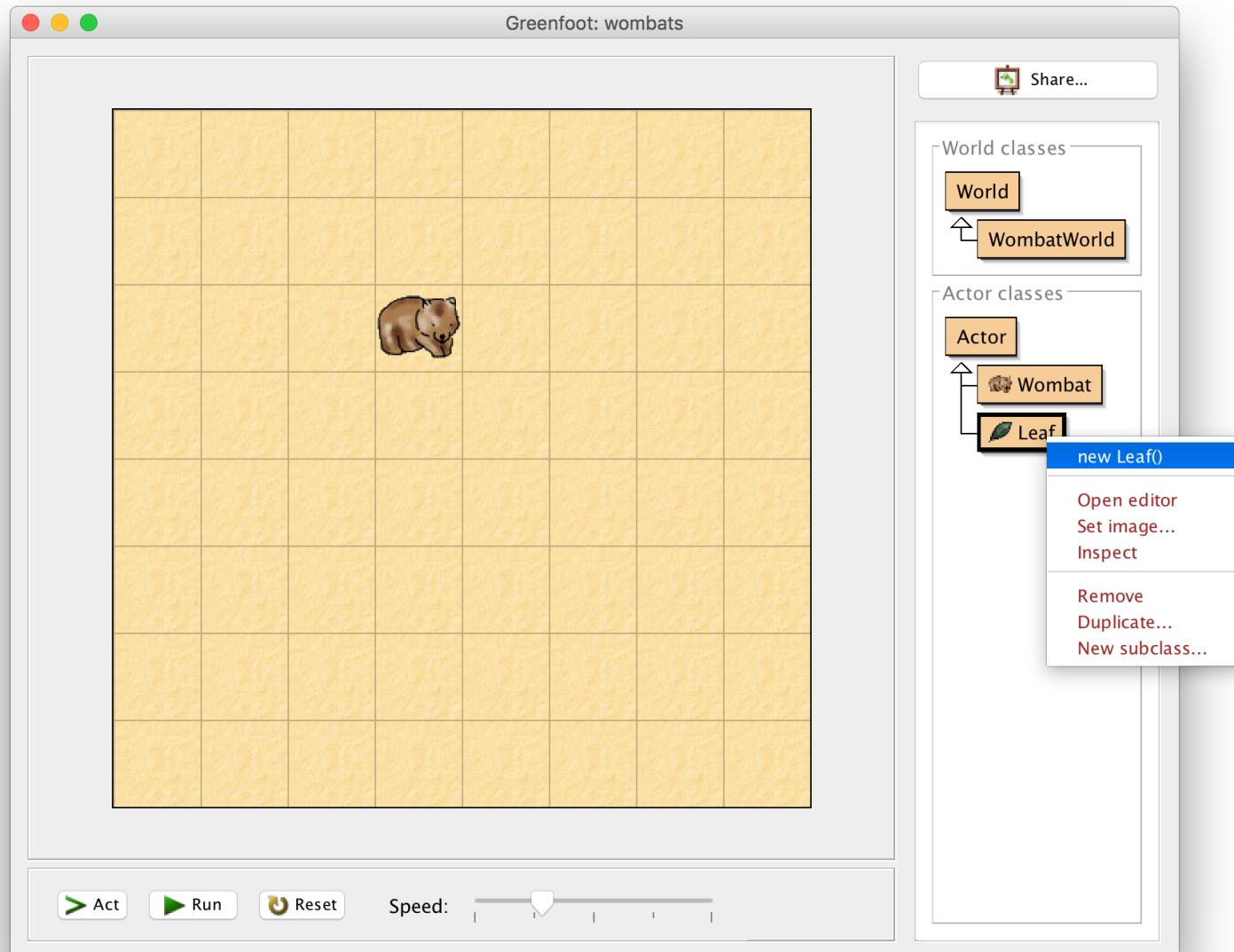
- World
 - WombatWorld
- Actor
 - Wombat
 - Leaf

At the bottom, there are buttons for > Act, Run, Reset, and Speed controls.

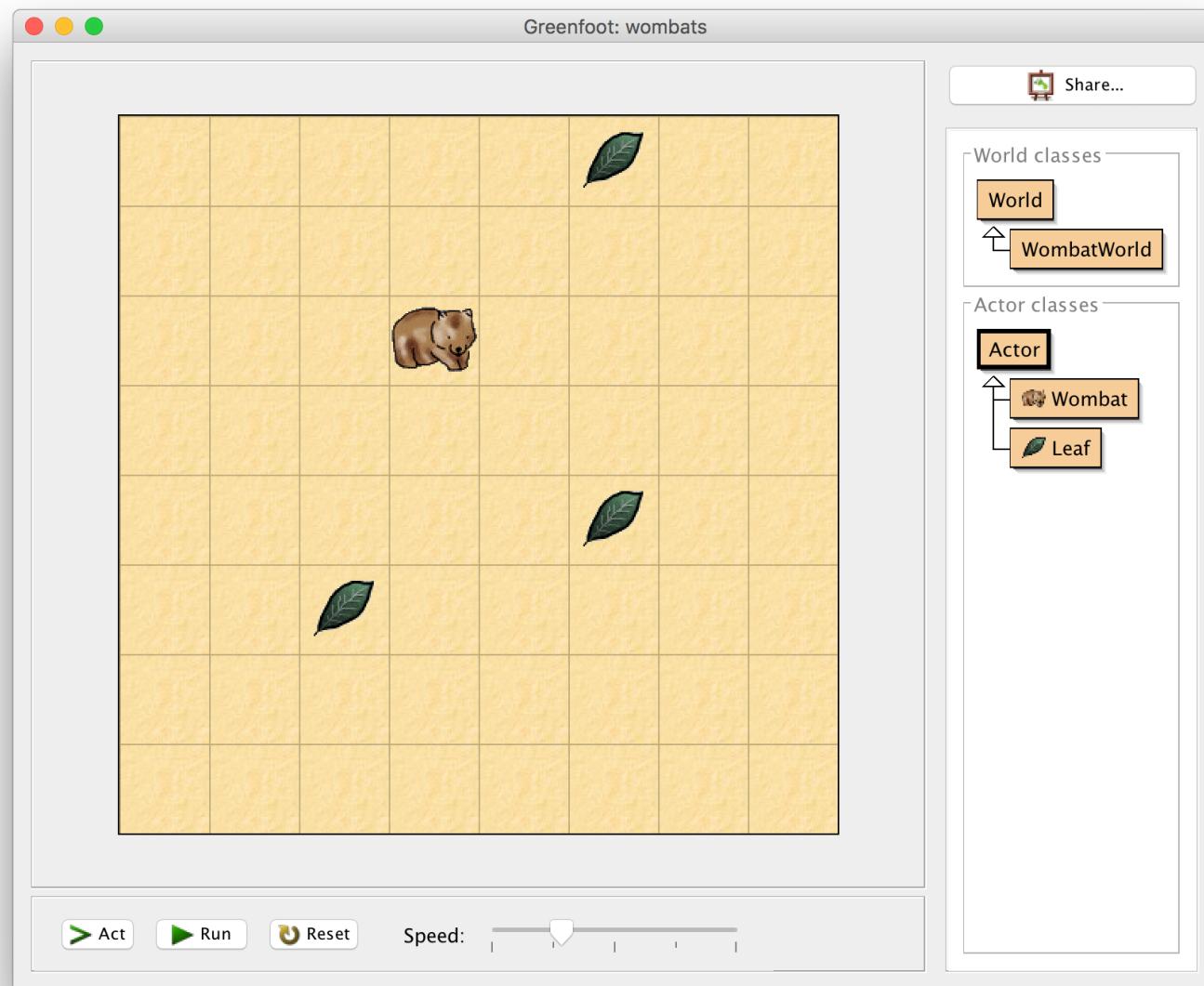
Create a Wombat



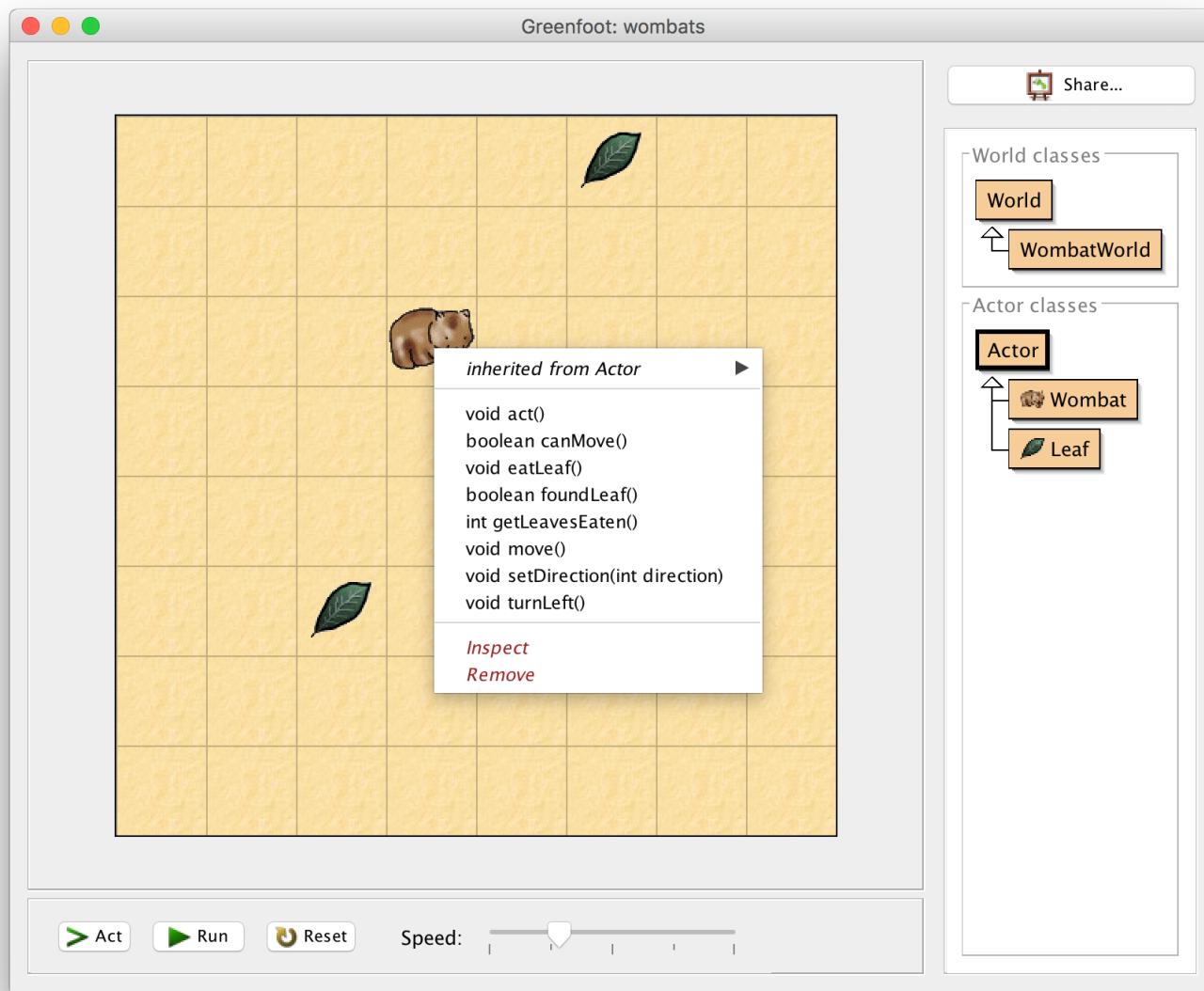
Create one or more Leaves



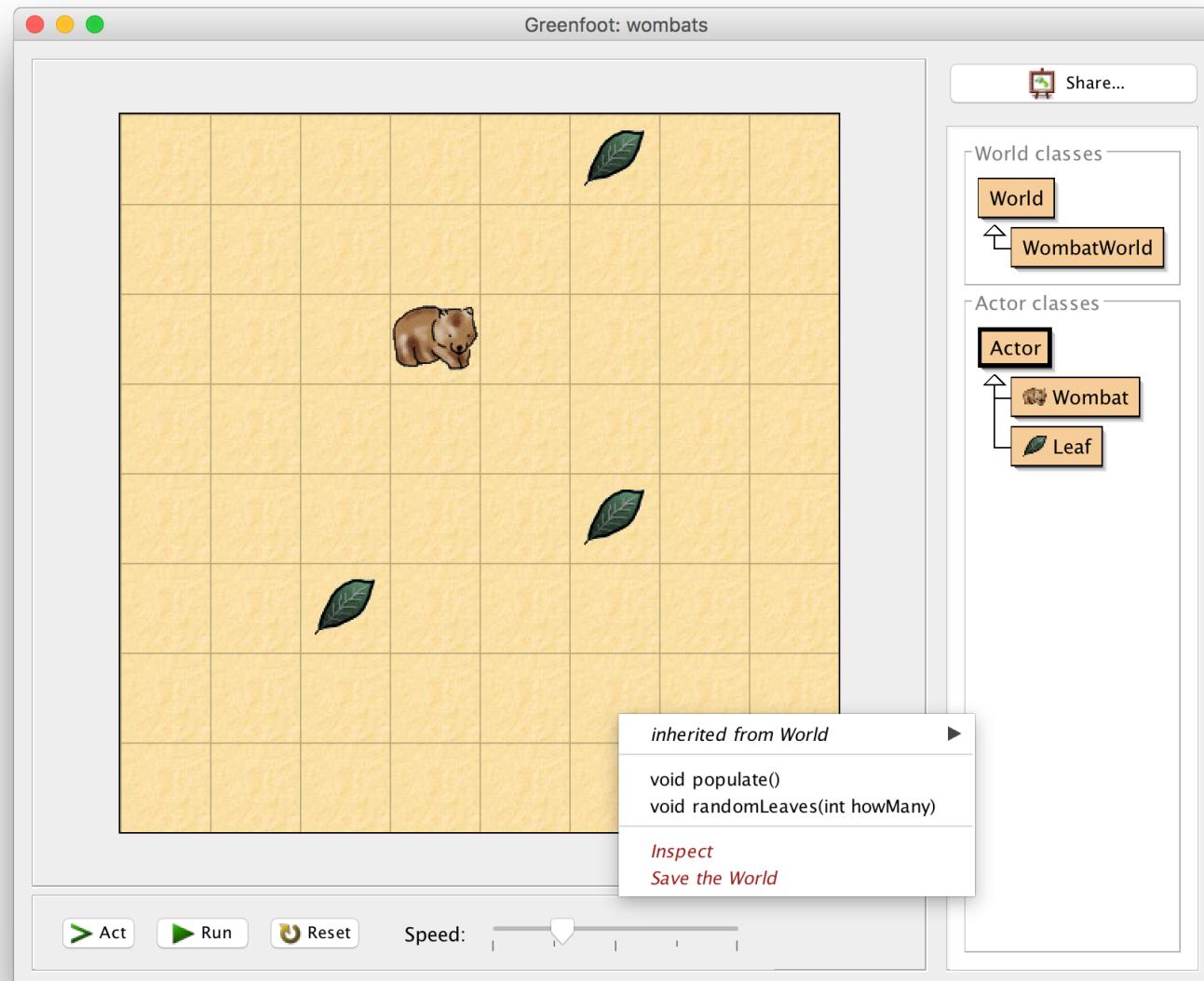
What happens on click of “Act”? “Run”? “Reset”?



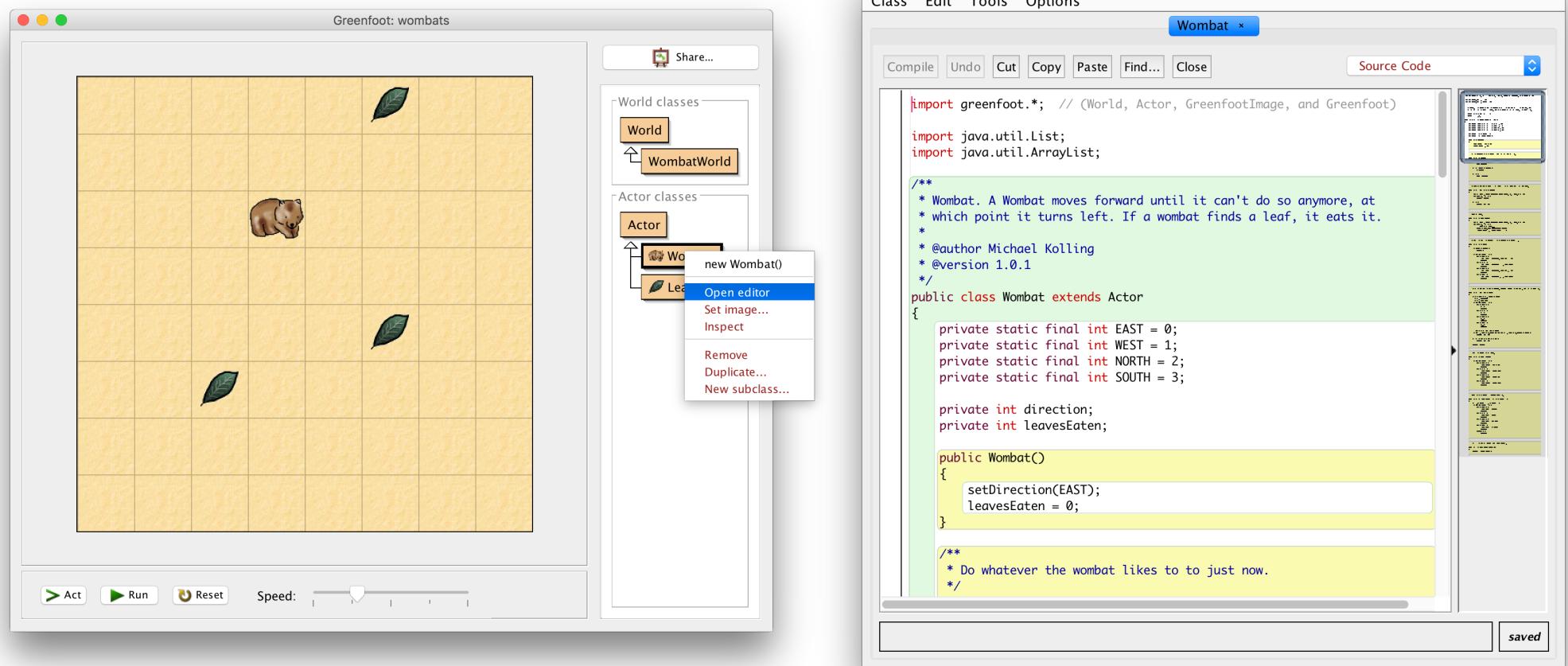
Objects have “Methods”



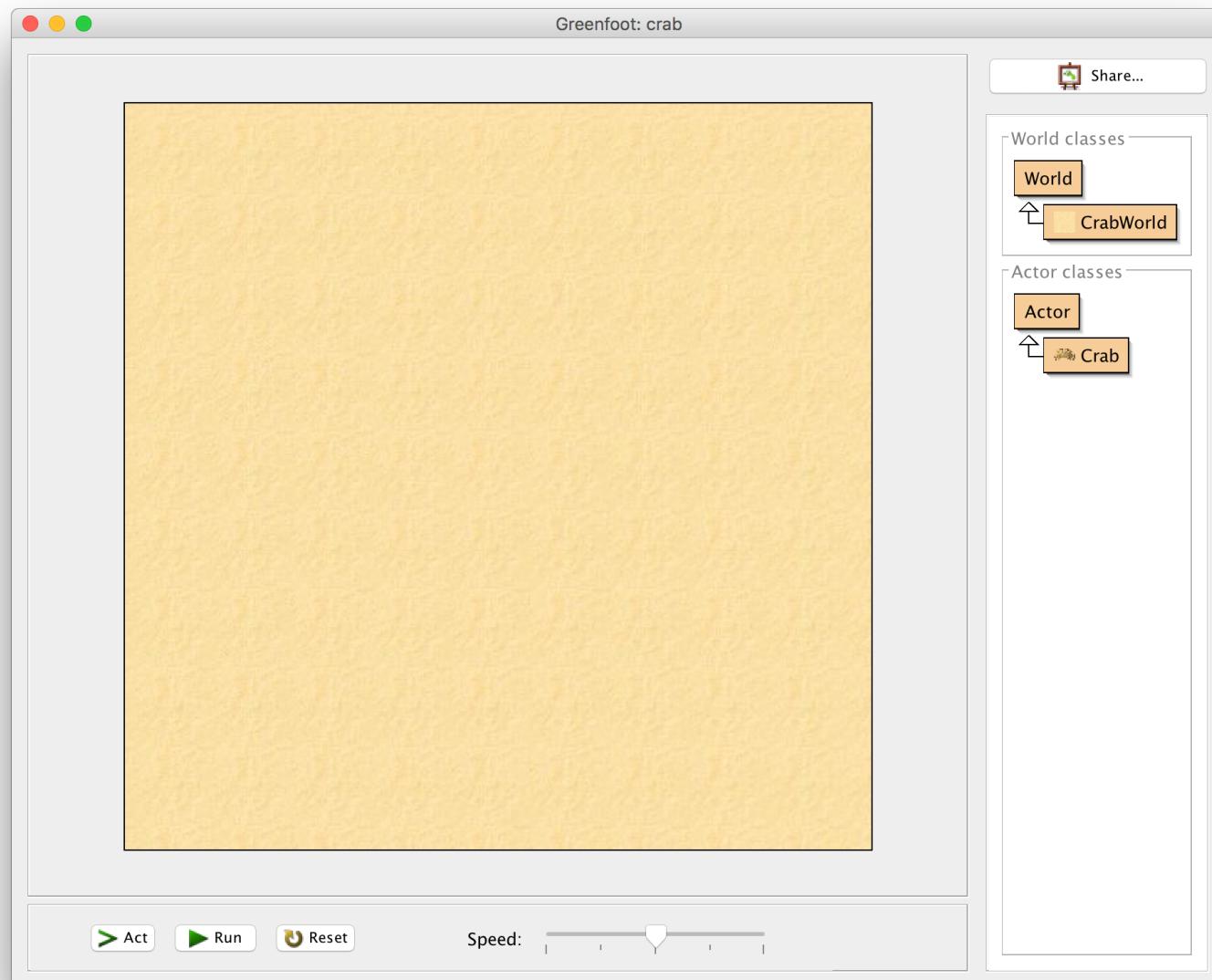
The “World” has Methods.



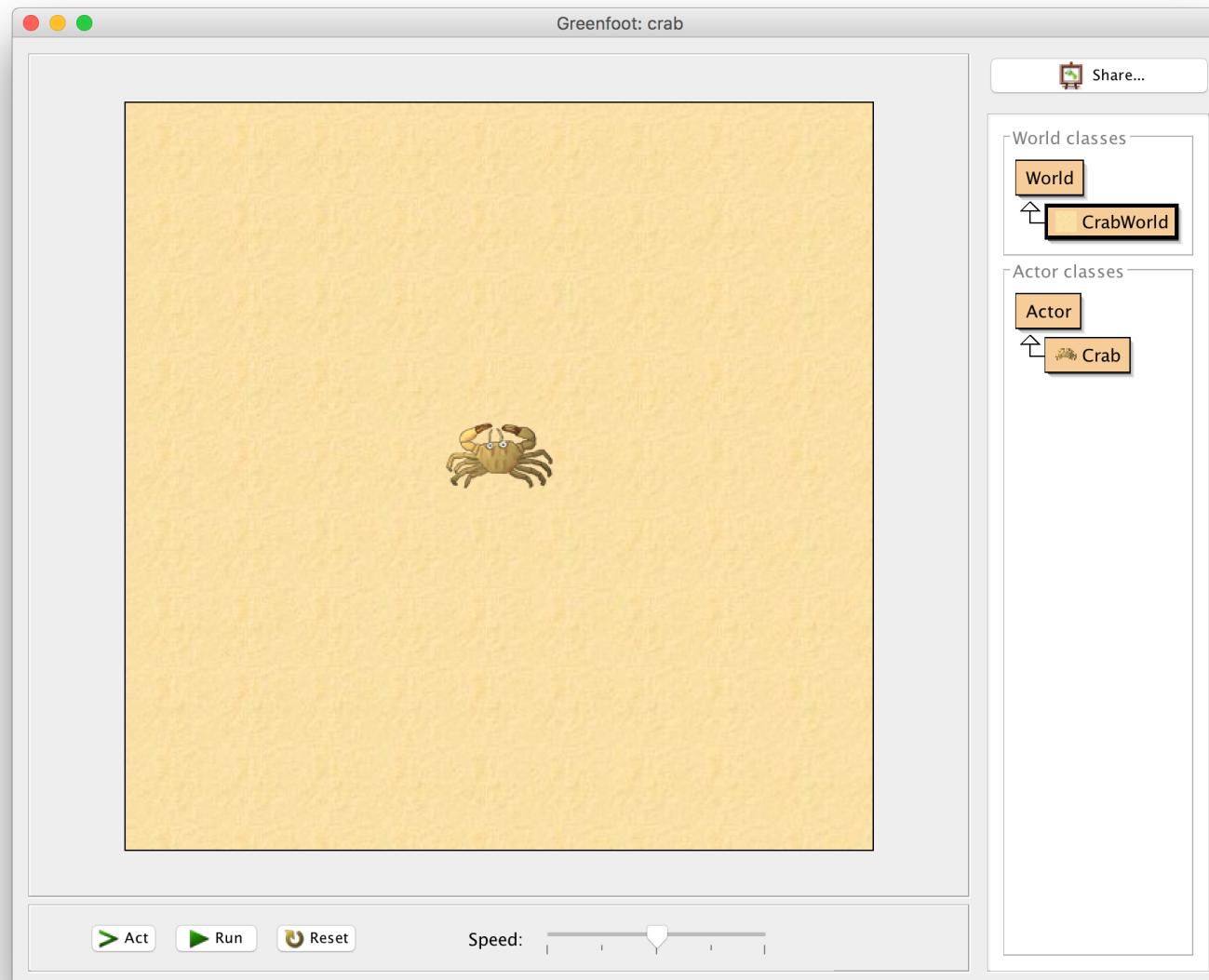
Edit Code of Objects



Part 2 - Crab



Create a “Crab” and click “Run”



The “Lazy” Crab!

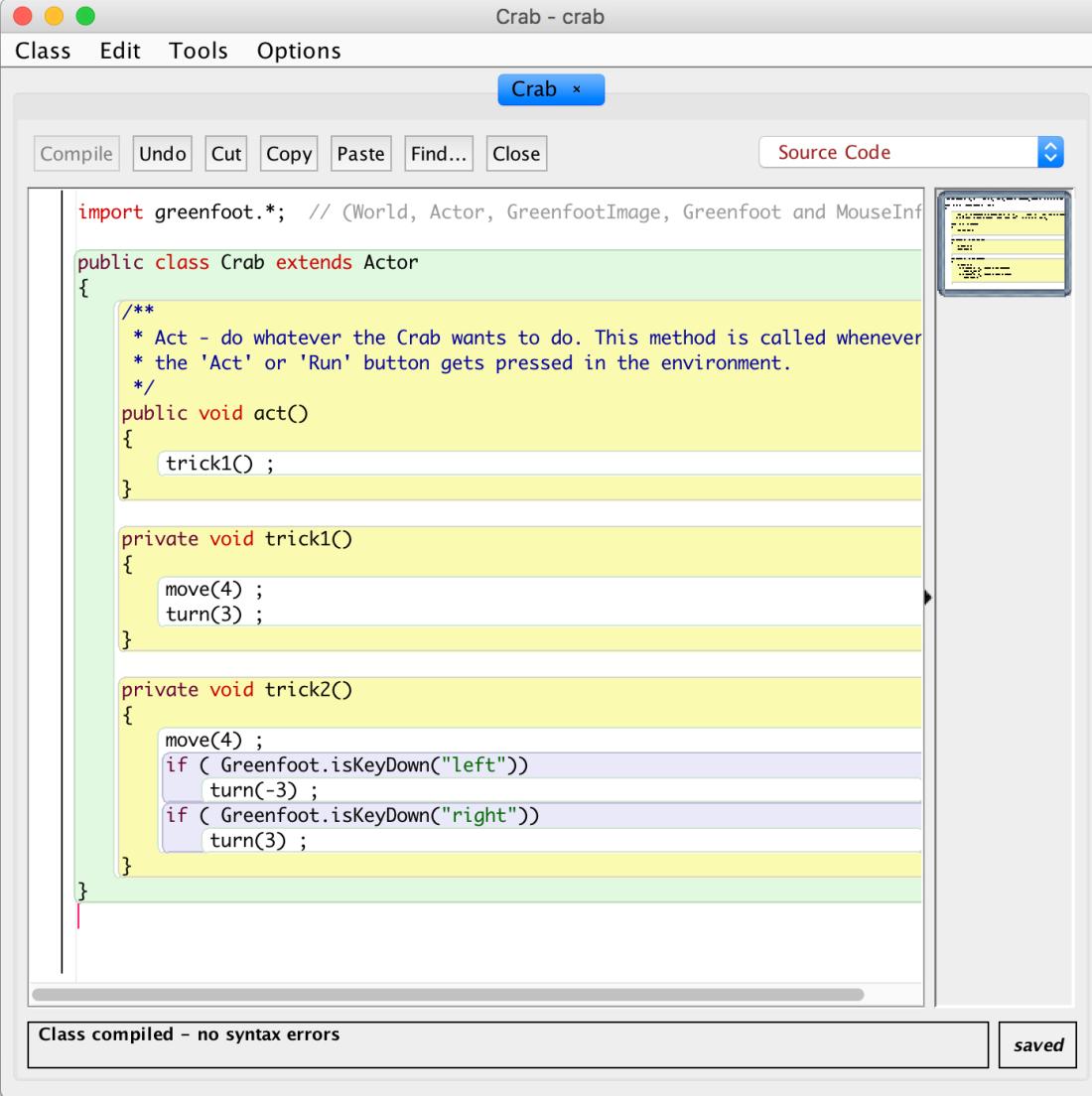
The screenshot shows the Greenfoot IDE interface with a window titled "Crab - crab". The window has a menu bar with "Class", "Edit", "Tools", and "Options". Below the menu is a toolbar with buttons for "Compile", "Undo", "Cut", "Copy", "Paste", "Find...", and "Close". A tab labeled "Crab" is selected. To the right of the tabs is a "Source Code" button. The main area contains the Java code for the "Crab" class:

```
import greenfoot.*; // CWorld, Actor, GreenfootImage, Greenfoot and MouseInfo

public class Crab extends Actor
{
    /**
     * Act - do whatever the Crab wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public void act()
    {
    }
}
```

On the far right of the code editor, there is a vertical scroll bar. At the bottom right of the window, there is a "saved" status indicator.

Teach the “Crab” some Tricks!



The screenshot shows the Greenfoot IDE interface with the title bar "Crab - crab". The menu bar includes "Class", "Edit", "Tools", and "Options". A toolbar below the menu has buttons for "Compile", "Undo", "Cut", "Copy", "Paste", "Find...", and "Close". The main window is titled "Crab x" and contains the "Source Code" tab. The code editor displays the following Java code:

```
import greenfoot.*; // CWorld, Actor, GreenfootImage, Greenfoot and MouseInfo

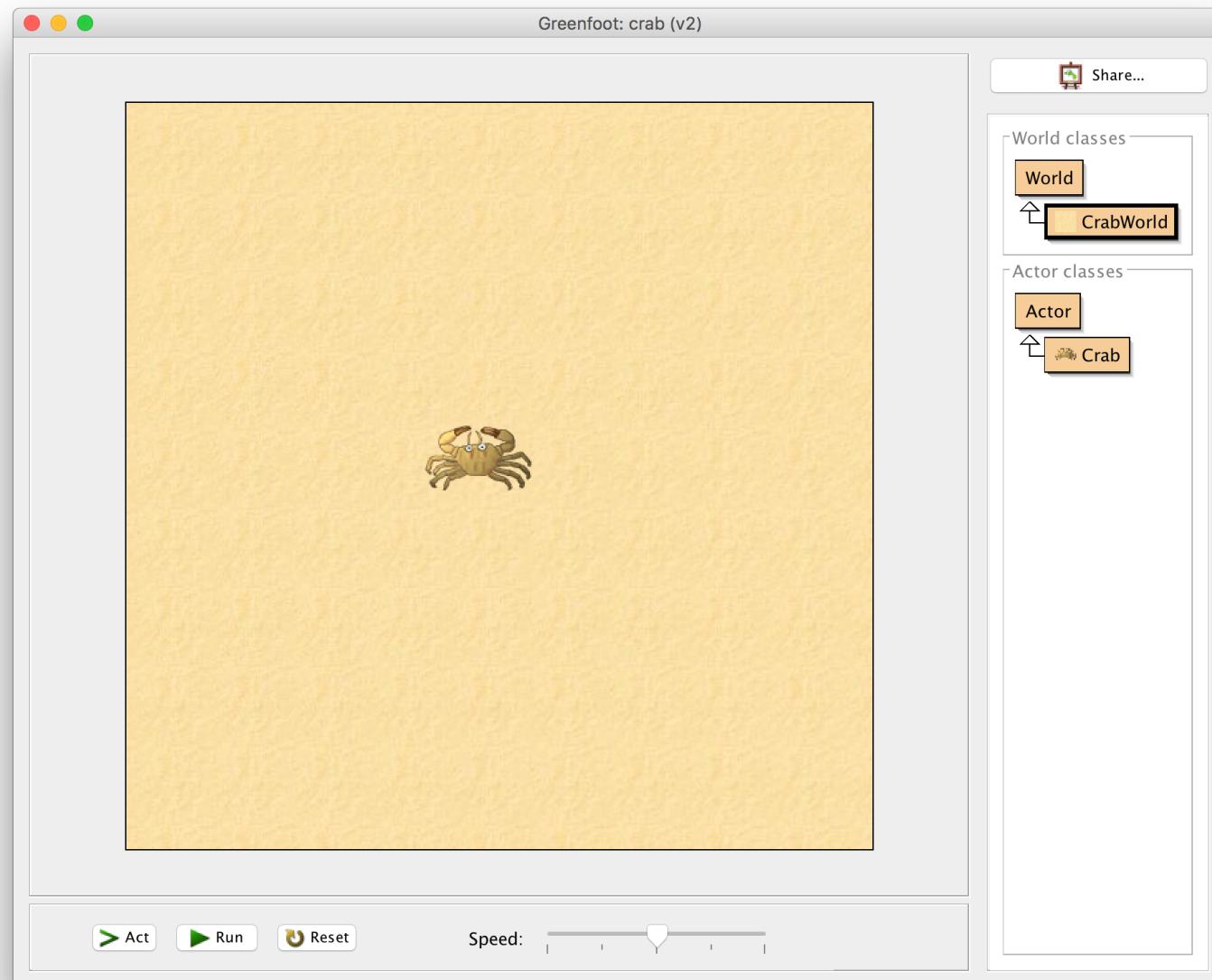
public class Crab extends Actor
{
    /**
     * Act - do whatever the Crab wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public void act()
    {
        trick1();
    }

    private void trick1()
    {
        move(4);
        turn(3);
    }

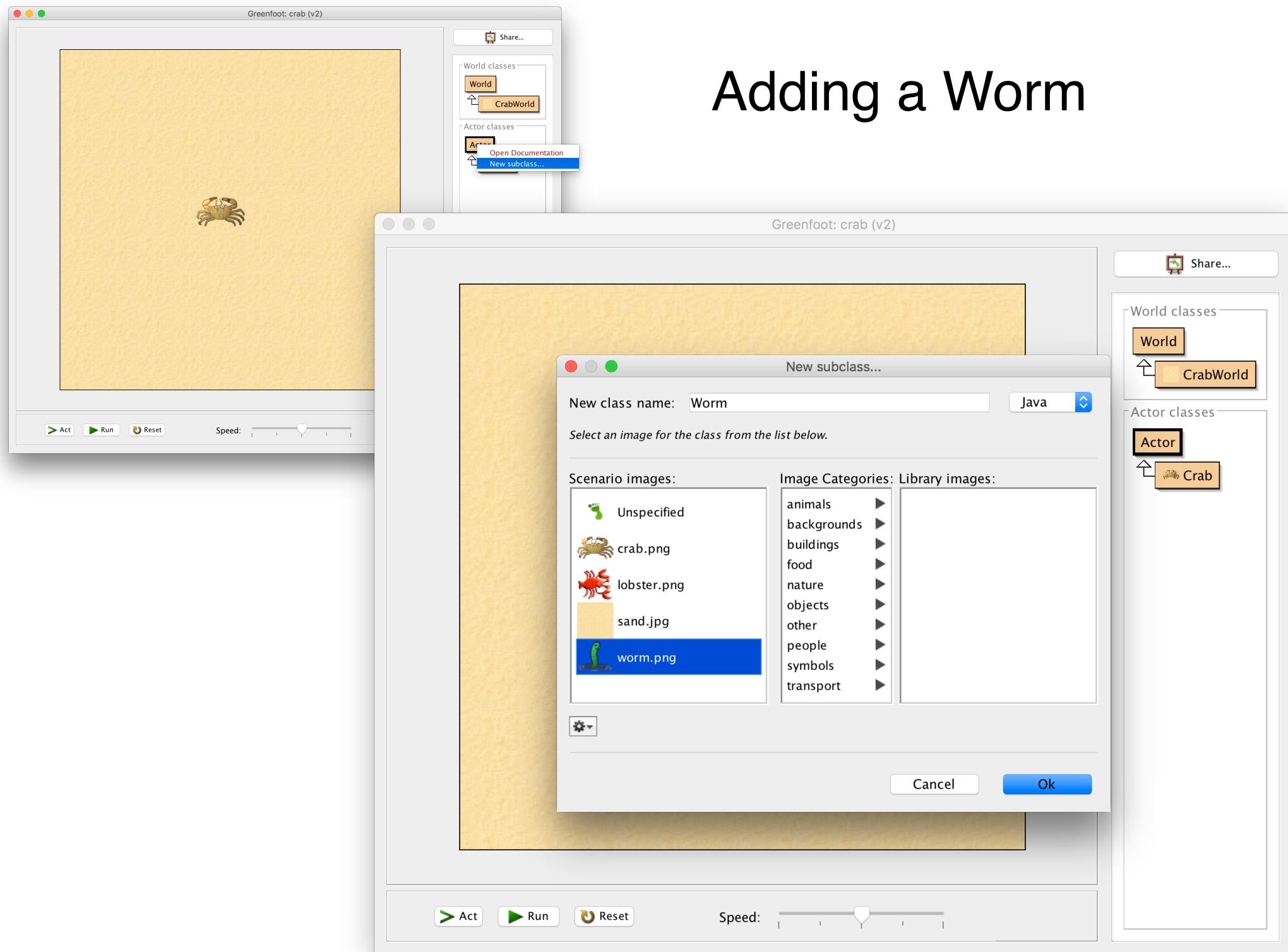
    private void trick2()
    {
        move(4);
        if ( Greenfoot.isKeyDown("left"))
            turn(-3);
        if ( Greenfoot.isKeyDown("right"))
            turn(3);
    }
}
```

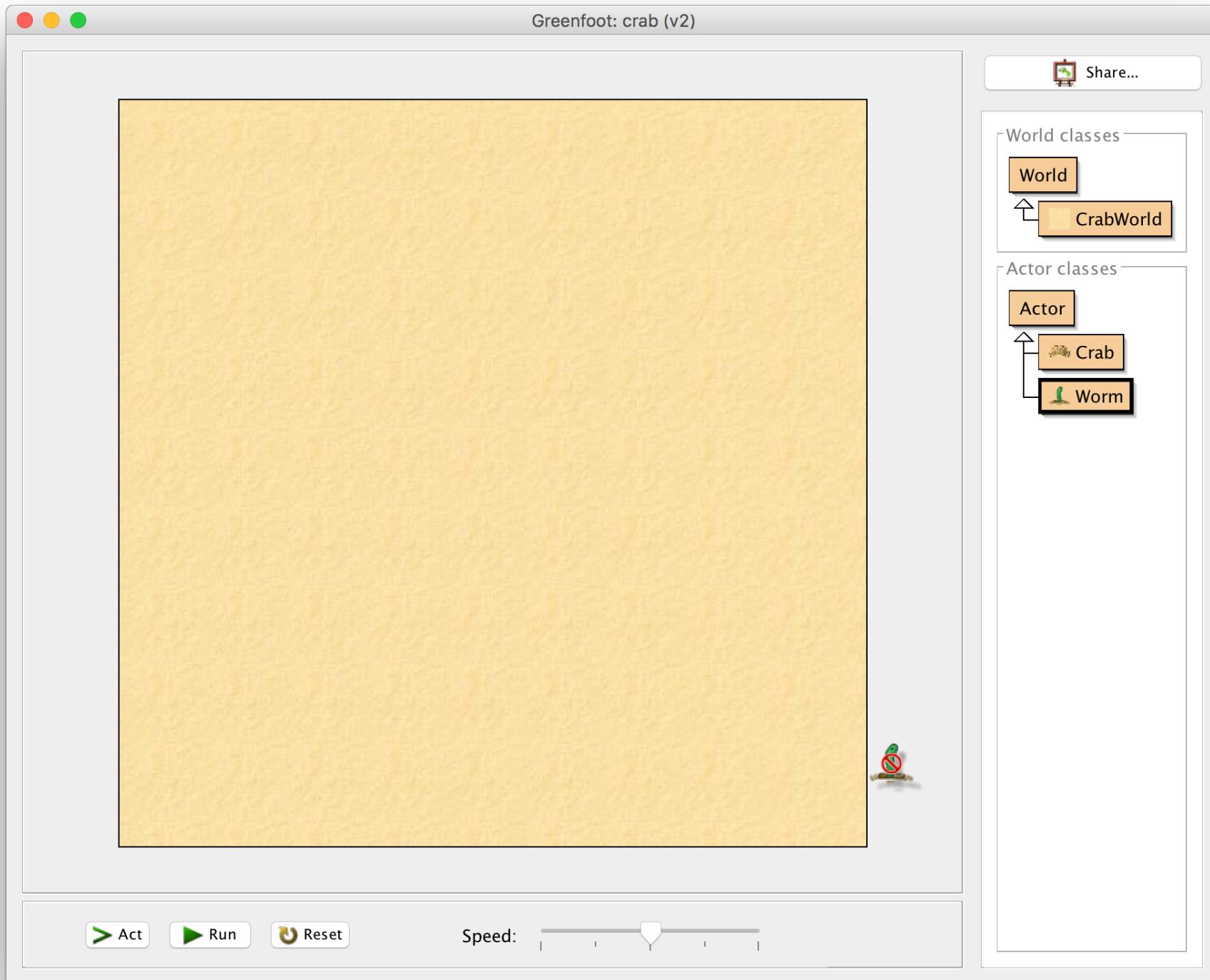
The code uses the Greenfoot framework to define a class named "Crab" that extends the "Actor" class. The "act()" method contains a call to the "trick1()" method. The "trick1()" method moves the crab 4 units and turns it 3 degrees clockwise. The "trick2()" method moves the crab 4 units and performs a turn based on the left or right arrow key being pressed. The code editor has syntax highlighting and a code completion dropdown visible on the right side.

Part 3 - Eating Worms

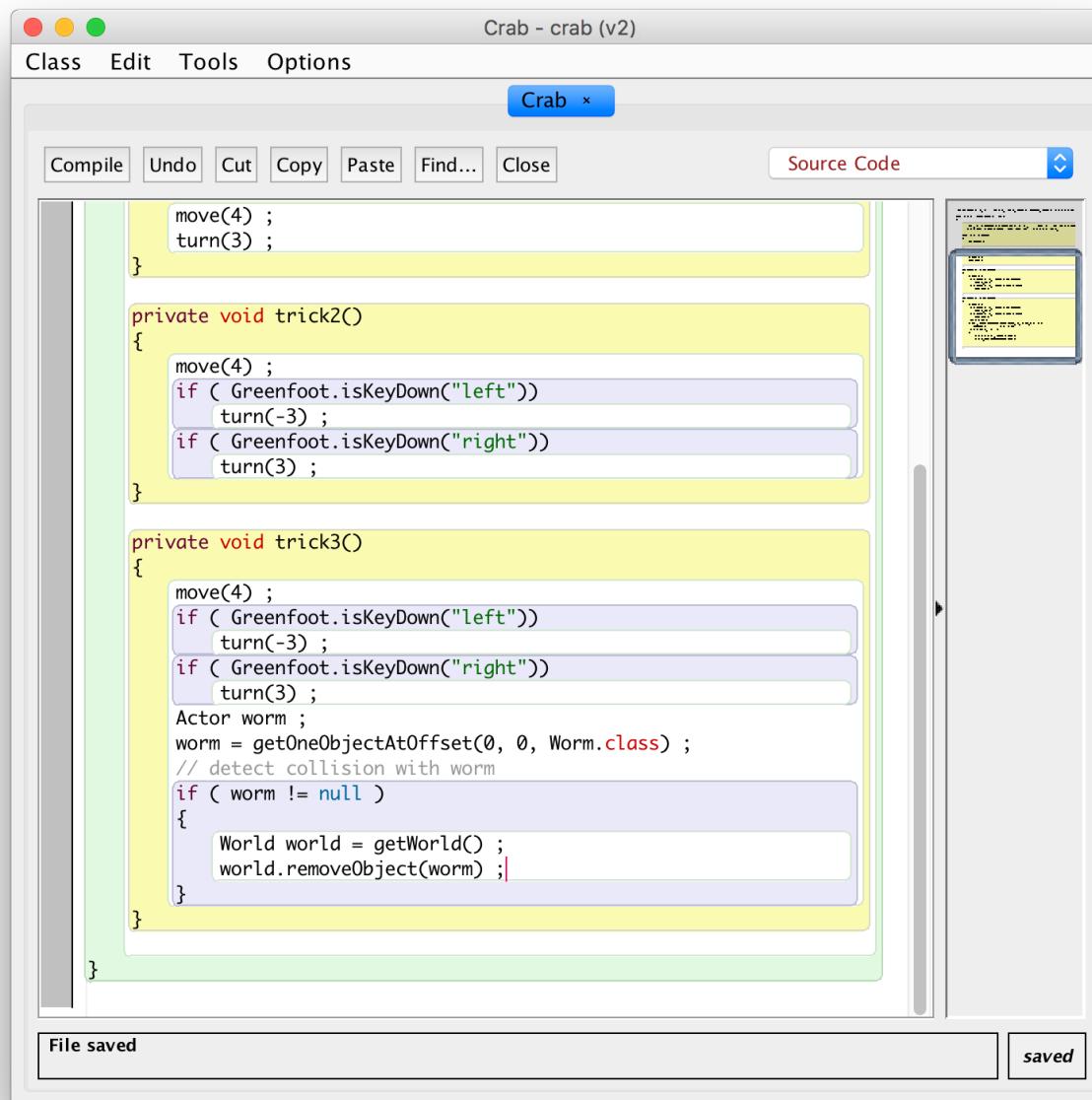


Adding a Worm





Collision Detection and Removing Objects



The screenshot shows the Greenfoot IDE interface with the title bar "Crab - crab (v2)". The menu bar includes "Class", "Edit", "Tools", and "Options". The tab bar shows "Crab x". The toolbar contains "Compile", "Undo", "Cut", "Copy", "Paste", "Find...", and "Close". A dropdown menu "Source Code" is open. The main area displays Java code for a class named "Crab". The code includes methods for movement and collision detection:

```
move(4) ;
turn(3) ;

private void trick2()
{
    move(4) ;
    if ( Greenfoot.isKeyDown("left"))
        turn(-3) ;
    if ( Greenfoot.isKeyDown("right"))
        turn(3) ;
}

private void trick3()
{
    move(4) ;
    if ( Greenfoot.isKeyDown("left"))
        turn(-3) ;
    if ( Greenfoot.isKeyDown("right"))
        turn(3) ;
    Actor worm ;
    worm = getOneObjectAtOffset(0, 0, Worm.class) ;
    // detect collision with worm
    if ( worm != null )
    {
        World world = getWorld() ;
        world.removeObject(worm) ;
    }
}
```

The status bar at the bottom left says "File saved" and the bottom right says "saved".

Refactored

Crab - crab (v2)

Class Edit Tools Options

Crab x

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

```
worm = getOneObjectAtOffset(0, 0, Worm.class) ;
// detect collision with worm
if ( worm != null )
{
    World world = getWorld() ;
    world.removeObject(worm) ;
}

public void moveAndTurn()
{
    move(4) ;
    if ( Greenfoot.isKeyDown("left"))
        turn(-3) ;
    if ( Greenfoot.isKeyDown("right"))
        turn(3) ;
}

public void eat()
{
    Actor worm ;
    worm = getOneObjectAtOffset(0, 0, Worm.class) ;
    // detect collision with worm
    if ( worm != null )
    {
        World world = getWorld() ;
        world.removeObject(worm) ;
    }
}
```

File saved saved

Crab - crab (v2)

Class Edit Tools Options

Crab x

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

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

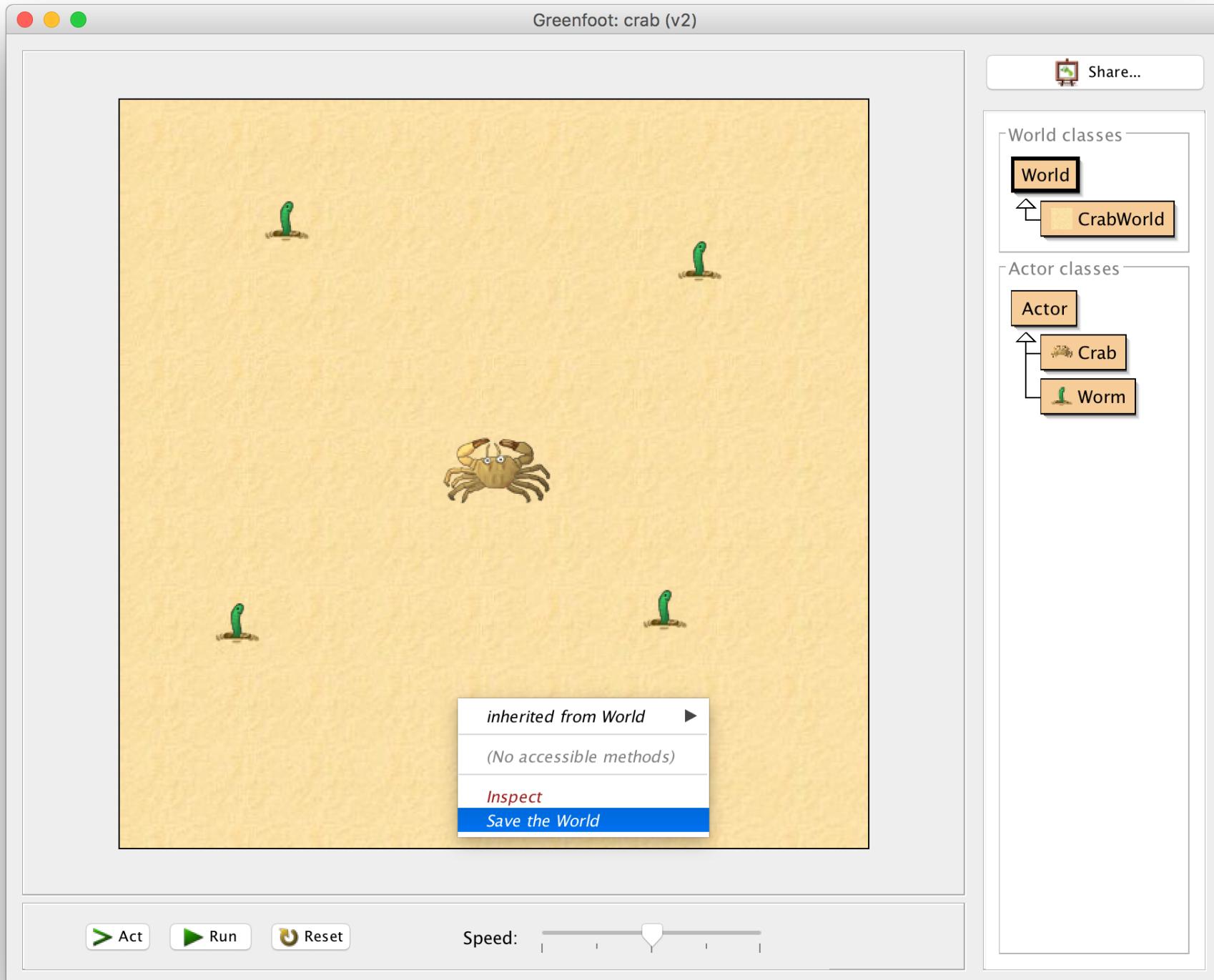
public class Crab extends Actor
{
    /**
     * Act - do whatever the Crab wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public void act()
    {
        moveAndTurn() ;
        eat() ;
    }

    private void trick1()
    {
        move(4) ;
        turn(3) ;
    }

    private void trick2()
    {
        move(4) ;
        if ( Greenfoot.isKeyDown("left"))
            turn(-3) ;
        if ( Greenfoot.isKeyDown("right"))
            turn(3) ;
    }

    private void trick3()
    {
        move(4) ;
    }
}
```

File saved saved



CrabWorld - crab (v2)

Class Edit Tools Options

CrabWorld ×

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

```
{  
    /**  
     * Constructor for objects of class CrabWorld.  
     *  
     */  
    public CrabWorld()  
    {  
        super(560, 560, 1);  
        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()  
    {  
        Crab crab = new Crab();  
        addObject(crab,282,277);  
        Worm worm = new Worm();  
        addObject(worm,125,91);  
        Worm worm2 = new Worm();  
        addObject(worm2,88,392);  
        Worm worm3 = new Worm();  
        addObject(worm3,434,121);  
        Worm worm4 = new Worm();  
        addObject(worm4,408,382);  
    }  
}
```

Class compiled – no syntax errors

saved

Part 4 - Playing Sounds

The screenshot shows the Greenfoot IDE interface with the title bar "Crab - crab (v2)". The menu bar includes Class, Edit, Tools, Options, and a tab labeled "Crab". Below the menu is a toolbar with buttons for Compile, Undo, Cut, Copy, Paste, Find..., and Close. A "Source Code" tab is selected. The main area contains the following Java code:

```
if ( worm != null )
{
    World world = getWorld() ;
    world.removeObject(worm) ;
}

public void moveAndTurn()
{
    move(4) ;
    if ( Greenfoot.isKeyDown("left"))
        turn(-3) ;
    if ( Greenfoot.isKeyDown("right"))
        turn(3) ;
}

public void eat()
{
    Actor worm ;
    worm = getOneObjectAtOffset(0, 0, Worm.class) ;
    // detect collision with worm
    if ( worm != null )
    {
        World world = getWorld() ;
        world.removeObject(worm) ;
        Greenfoot.playSound("eating.wav") ;
    }
}
```

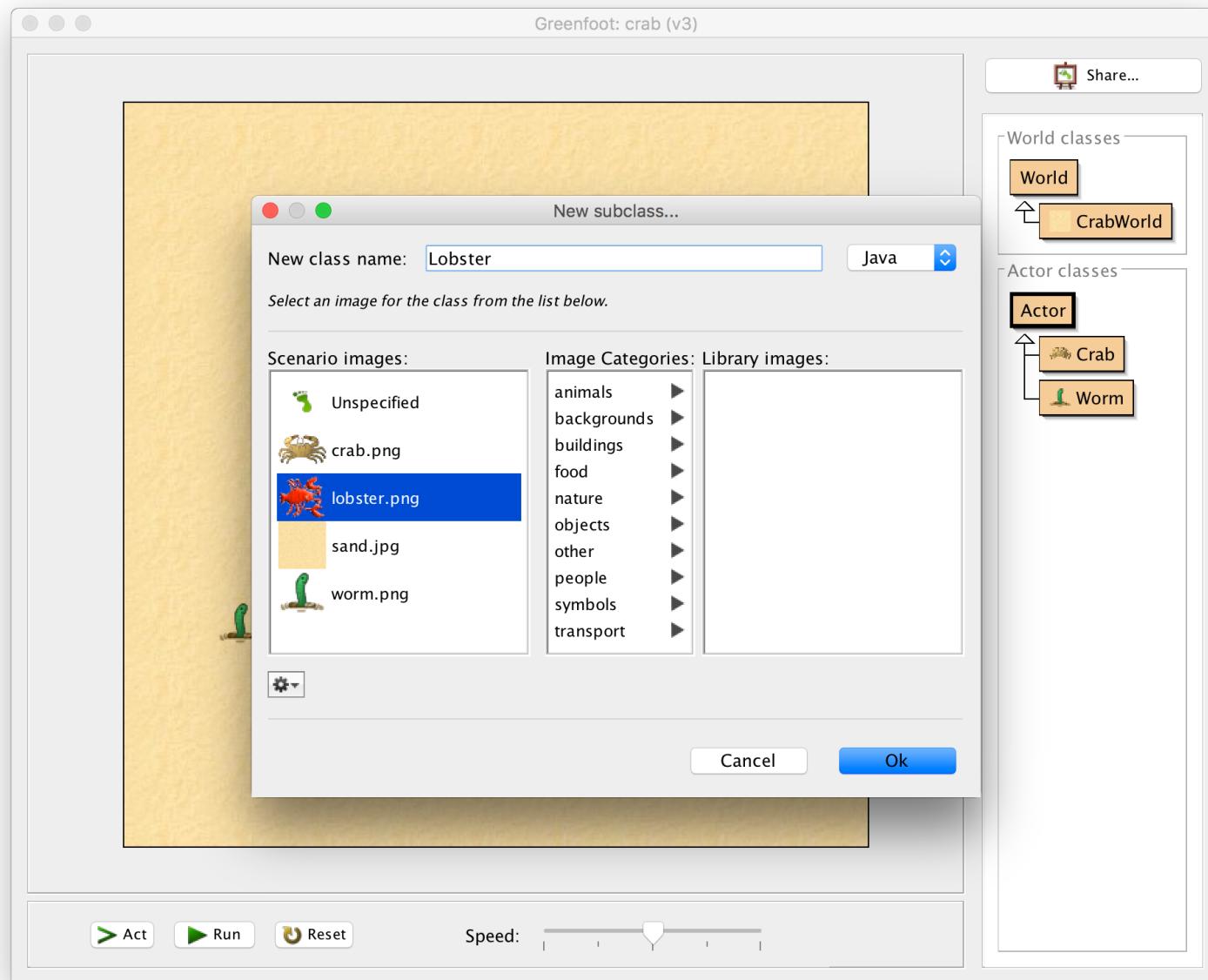
The code uses color-coded syntax highlighting. The right side of the interface features a project tree with several files listed under "Project Files". At the bottom, a status bar displays "Class compiled - no syntax errors" and "saved".

Try Recording your own Sounds

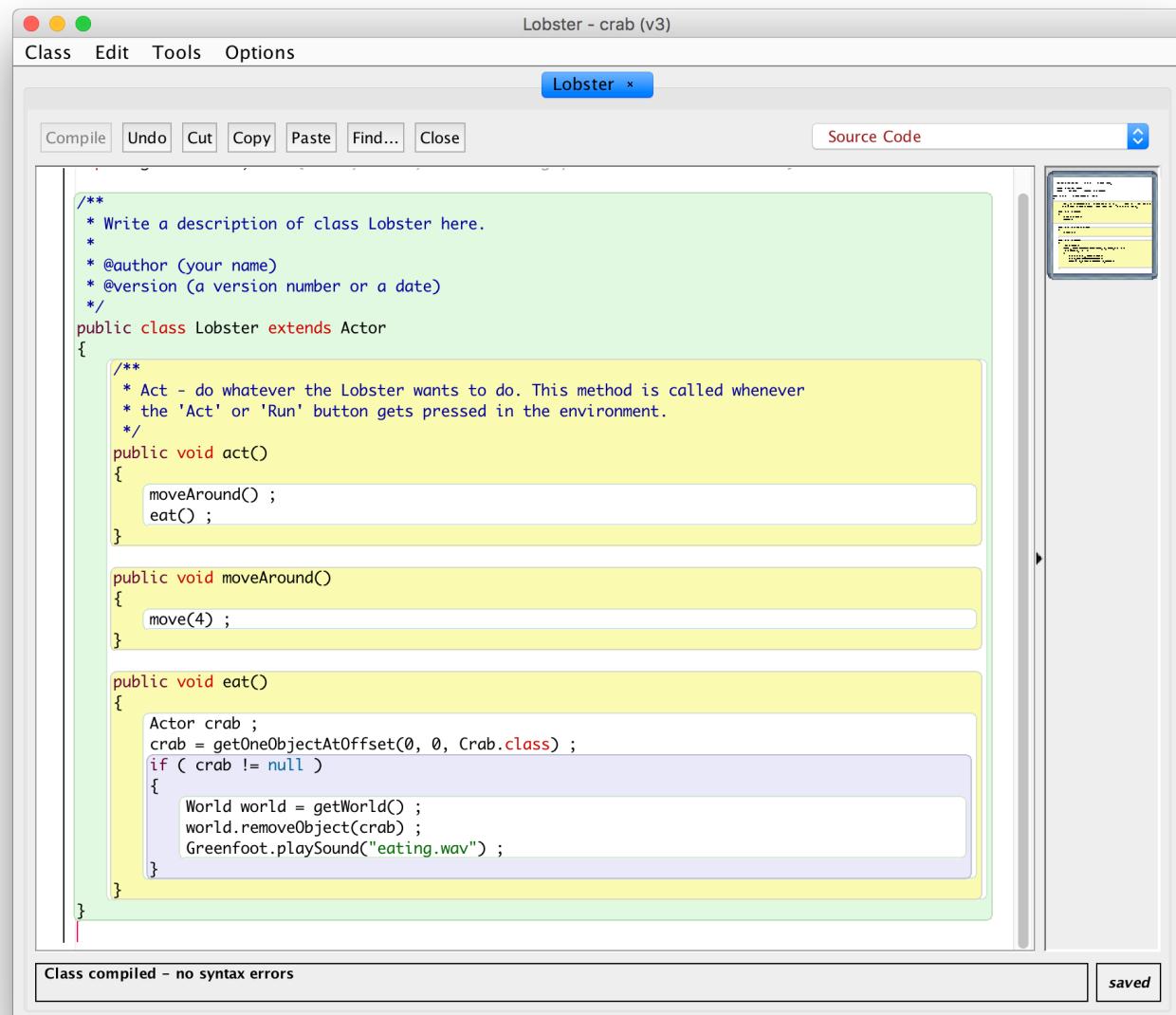
The screenshot shows a web browser window displaying the official Audacity website at www.audacityteam.org. The page has a dark theme with a green header bar. The header includes links for HOME, ABOUT, DOWNLOAD, HELP, CONTACT, GET INVOLVED, DONATE, PRIVACY POLICY, and COPYRIGHT. A search bar is also present. The main content area features a large image of headphones with a sound wave graphic. Below it, there's a sidebar with links for MAIN PAGES (Download, Windows, Mac, Linux, Help, FAQ, Documentation, Get Involved, Users, Developers, Translators, Credits, Contact) and RECENT POSTS. The central content area highlights the importance of secure HTTPS and provides download links for Windows, Mac, and Linux, along with source code and plug-in options. It also mentions obtaining a CD and lists distributors. A welcome message is displayed, followed by a brief description of what Audacity is and a screenshot of the software interface.

<http://www.audacityteam.org/>

Part 5 - Adding Enemies



Simple Lobster



The screenshot shows the Greenfoot IDE interface with the title bar "Lobster - crab (v3)". The menu bar includes Class, Edit, Tools, Options, and a tab labeled "Lobster". Below the menu is a toolbar with buttons for Compile, Undo, Cut, Copy, Paste, Find..., and Close. A "Source Code" dropdown menu is open. The main area displays the Java code for the Lobster class:

```
/*
 * Write a description of class Lobster here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Lobster extends Actor
{
    /*
     * Act - do whatever the Lobster wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public void act()
    {
        moveAround();
        eat();
    }

    public void moveAround()
    {
        move(4);
    }

    public void eat()
    {
        Actor crab;
        crab = getOneObjectAtOffset(0, 0, Crab.class);
        if (crab != null)
        {
            World world = getWorld();
            world.removeObject(crab);
            Greenfoot.playSound("eating.wav");
        }
    }
}
```

The code uses color-coded syntax highlighting. The right side of the interface features a vertical toolbar with various icons for file operations like Open, Save, and Print.

Dizzy Lobster

Lobster - crab (v3)

Class Edit Tools Options Lobster x

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

```
public void act()
{
    moveAround();
    eat();
}

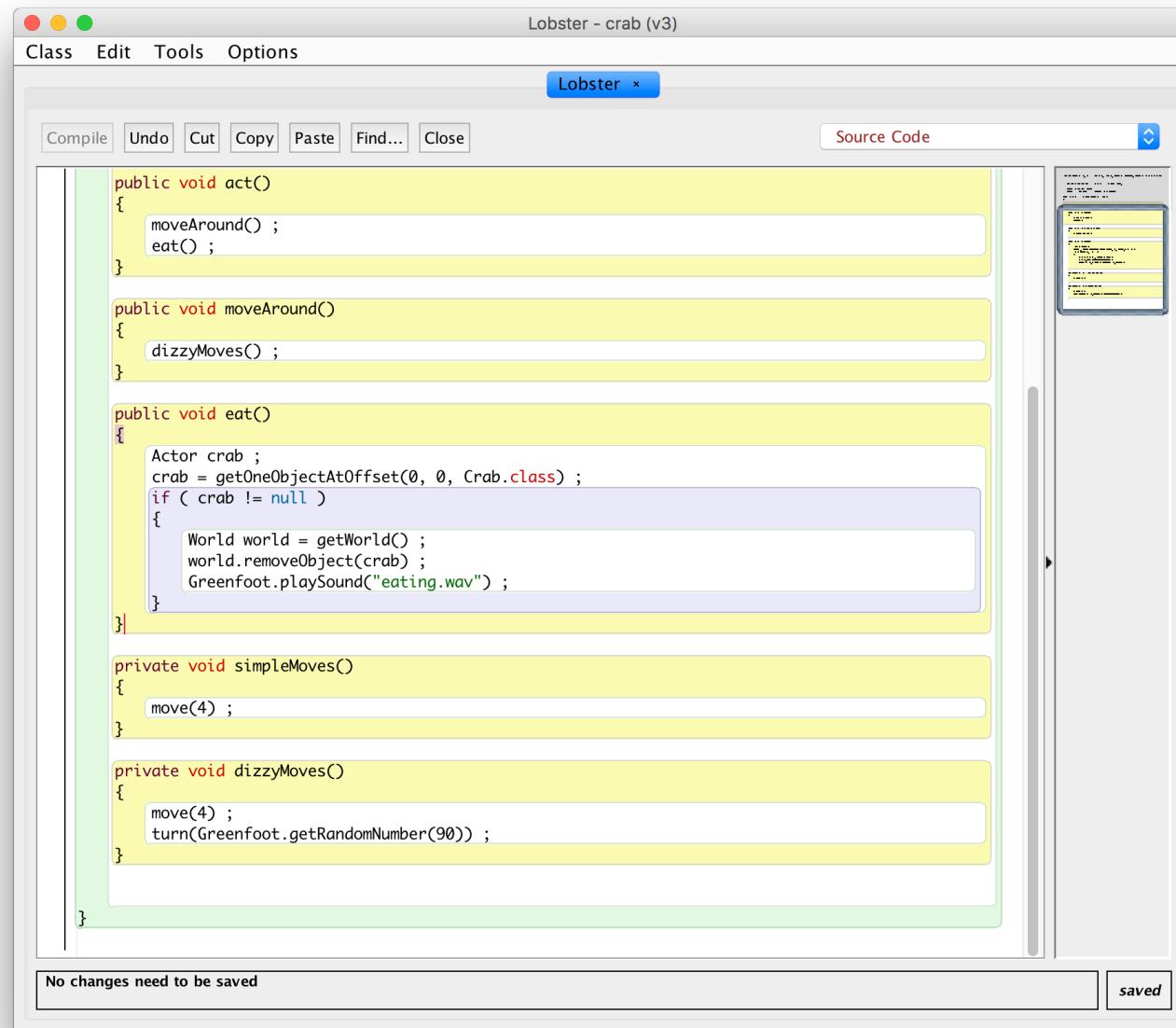
public void moveAround()
{
    dizzyMoves();
}

public void eat()
{
    Actor crab;
    crab = getOneObjectAtOffset(0, 0, Crab.class);
    if ( crab != null )
    {
        World world = getWorld();
        world.removeObject(crab);
        Greenfoot.playSound("eating.wav");
    }
}

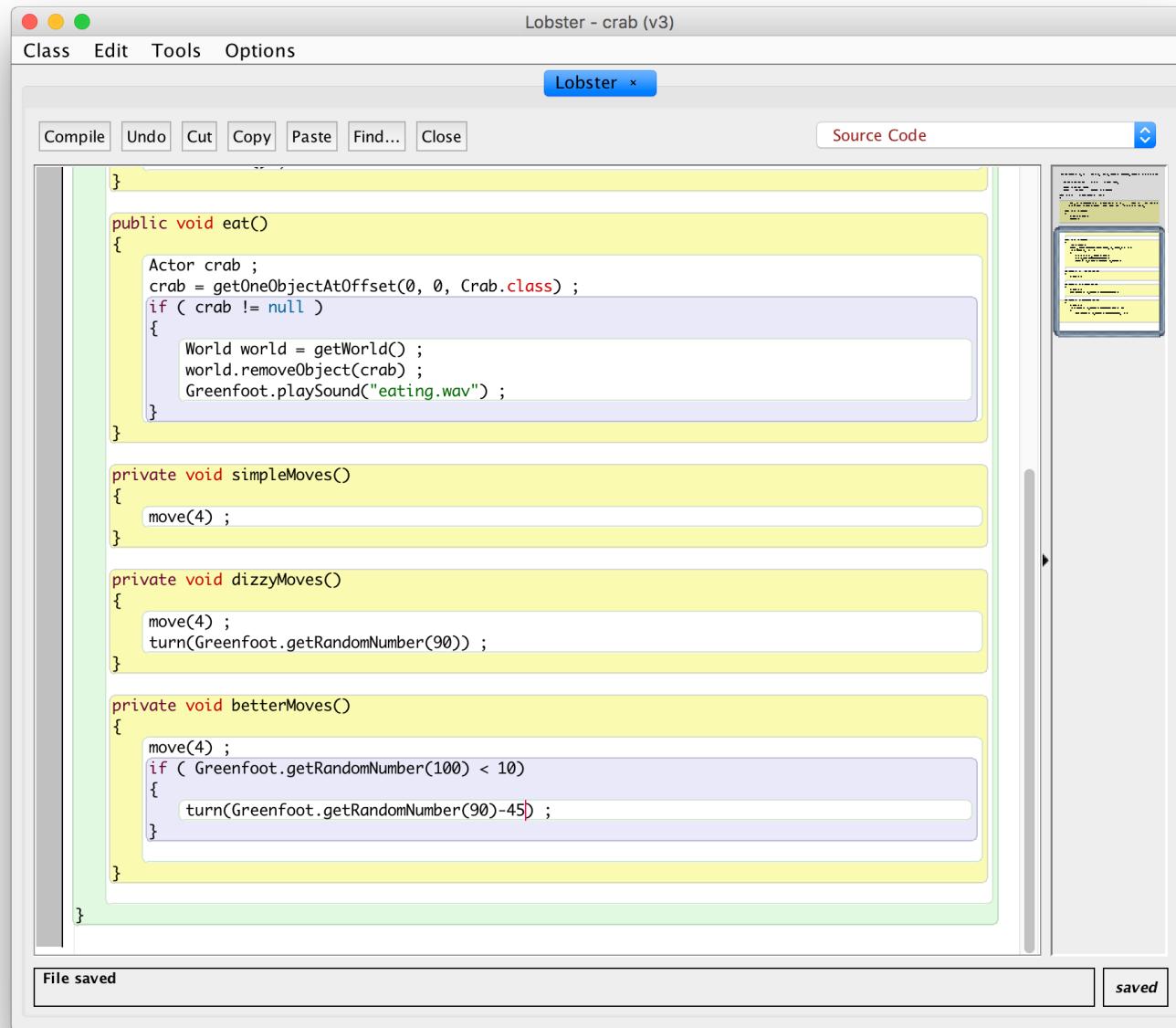
private void simpleMoves()
{
    move(4);
}

private void dizzyMoves()
{
    move(4);
    turn(Greenfoot.getRandomNumber(90));
}
```

No changes need to be saved saved

A screenshot of the Greenfoot IDE interface. The title bar says "Lobster - crab (v3)". The menu bar includes "Class", "Edit", "Tools", "Options", and "Lobster x". Below the menu is a toolbar with buttons for "Compile", "Undo", "Cut", "Copy", "Paste", "Find...", and "Close". To the right of the toolbar is a dropdown menu labeled "Source Code". The main area contains the source code for the "Lobster" class. The code defines three methods: "act()", "moveAround()", and "eat()". The "eat()" method uses the "getOneObjectAtOffset" method to find a "Crab" object and removes it from the world, playing a sound effect. The "dizzyMoves()" method is a private helper method that moves the actor 4 units and turns it at a random angle. A vertical scroll bar is visible on the right side of the code editor. On the far right, there is a small preview window showing a yellow crab character. At the bottom of the screen, a status bar says "No changes need to be saved" and has a "saved" button.

Better Lobster



The screenshot shows the Greenfoot IDE interface with the title bar "Lobster - crab (v3)". The menu bar includes "Class", "Edit", "Tools", and "Options". A tab bar at the top right shows "Lobster x". Below the tabs is a toolbar with "Compile", "Undo", "Cut", "Copy", "Paste", "Find...", and "Close" buttons. To the right of the toolbar is a "Source Code" button with a dropdown arrow. The main area displays the source code for the "Lobster" class:

```
public void eat()
{
    Actor crab ;
    crab = getOneObjectAtOffset(0, 0, Crab.class) ;
    if ( crab != null )
    {
        World world = getWorld() ;
        world.removeObject(crab) ;
        Greenfoot.playSound("eating.wav") ;
    }
}

private void simpleMoves()
{
    move(4) ;
}

private void dizzyMoves()
{
    move(4) ;
    turn(Greenfoot.getRandomNumber(90)) ;
}

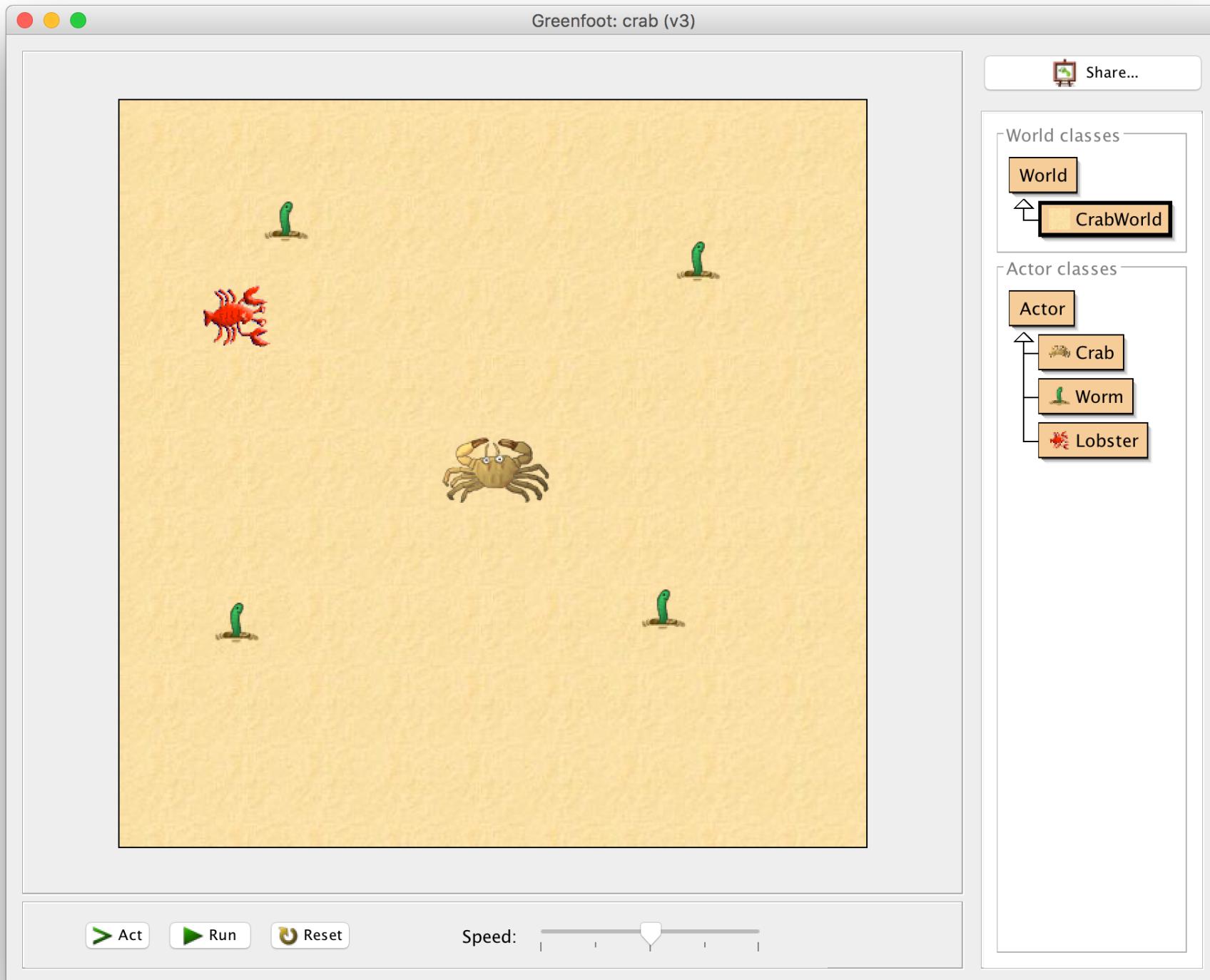
private void betterMoves()
{
    move(4) ;
    if ( Greenfoot.getRandomNumber(100) < 10 )
    {
        turn(Greenfoot.getRandomNumber(90)-45) ;
    }
}
```

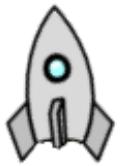
The code uses Java syntax with Greenfoot-specific methods like `getOneObjectAtOffset`, `removeObject`, and `playSound`. The `turn` method is used to rotate the actor.

Off the Wall!

The screenshot shows the Greenfoot IDE interface with the title bar "Lobster - crab (v3)". The menu bar includes Class, Edit, Tools, and Options. A toolbar with buttons for Compile, Undo, Cut, Copy, Paste, Find..., and Close is visible. The main window displays the source code for the Lobster class. The code includes methods for sound playback, simple moves, dizzy moves, better moves, and off-the-wall moves. The "Source Code" tab is selected. On the right side, there is a panel with various icons and settings. At the bottom, a status bar indicates "No changes need to be saved" and "saved".

```
Greenfoot.playSound("eating.wav") ;  
}  
}  
  
private void simpleMoves()  
{  
    move(4) ;  
}  
  
private void dizzyMoves()  
{  
    move(4) ;  
    turn(Greenfoot.getRandomNumber(90)) ;  
}  
  
private void betterMoves()  
{  
    move(4) ;  
    if ( Greenfoot.getRandomNumber(100) < 10 )  
    {  
        turn(Greenfoot.getRandomNumber(90)-45) ;  
    }  
}  
  
private void offTheWallMoves()  
{  
    move(4) ;  
    if ( Greenfoot.getRandomNumber(100) < 10 )  
    {  
        turn(Greenfoot.getRandomNumber(90)-45) ;  
    }  
    if ( getX() <= 5 || getX() >= getWorld().getWidth() - 5 )  
        turn(180) ;  
    if ( getY() <= 5 || getY() >= getWorld().getHeight() - 5 )  
        turn(180) ;  
}
```





Part 6 - Space Ships

Greenfoot: spaceship

The screenshot shows a Greenfoot application window titled "Greenfoot: spaceship". The main area displays a dark blue background filled with numerous small white stars of varying sizes. In the center, there is a white rocket ship with a blue circular window and a red base, oriented vertically. On the left side of the screen, there is a digital score or counter displaying the number "0". At the bottom of the screen, there is a toolbar with three buttons: "Act" (with a play icon), "Run" (with a play icon), and "Reset" (with a circular arrow icon). To the right of the toolbar, there is a "Speed:" slider with a shield icon. On the far right of the interface, there is a sidebar titled "World classes" which contains a class hierarchy diagram:

```
graph TD; World[World] --> Space[Space]
```

Below the world classes, there is another section titled "Actor classes" which contains the following class hierarchy diagram:

```
graph TD; Actor[Actor] --> Counter[Counter]; Actor --> Asteroid[Asteroid]; Actor --> Rocket[Rocket]; Actor --> Shot[Shot]
```

At the top right of the sidebar, there is a "Share..." button with a paintbrush icon.

Greenfoot

www.greenfoot.org/doc/howto-1

LIBRARY WEB ADMIN CODE EDU LABS SJSU 202 281 279 DOCS ZAPP BOOKS

Activity About Documentation Download Scenarios Discuss

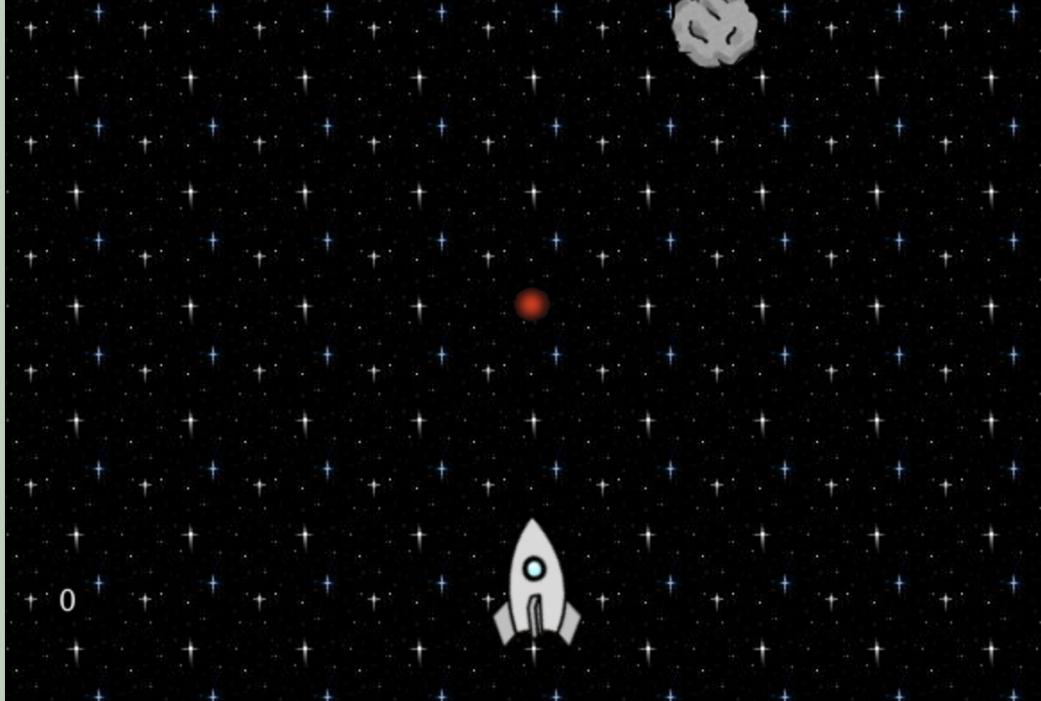
How to access one object from another

A common question that Greenfoot beginners ask is:

How can I access the variables (or methods) of one object/class from another one?

But of course usually it is phrased differently... sometimes beginners don't yet understand what "variables" or "objects" are. So the question might look more like this:

How can I update the score counter whenever an asteroid is destroyed by the rocket?



Now, take a look at the Greenfoot scenario pictured above:

[tut-access-p1](#) (right-click and open the link in a new window or tab).

Additional Resources

Greenfoot | Scenarios

www.greenfoot.org/scenarios

LIBRARY WEB ADMIN CODE EDU LABS SJSU 202 281 279 DOCS ZAPP BOOKS

Greenfoot

Search ...

Username

Password

Remember Me? Login

Sign Up, Lost Password

Activity About Documentation Download Scenarios Discuss

Scenarios

Newest Just Updated Most Plays (this week) Most Votes (this week) Most Popular 1 2 3 ►


little-crab 2-player
by MadsOkkels, 14 hours ago
plays 51 / votes 0


RobTheBank
by Sauerbraten, yesterday
crazy game my Moritz
plays 123 / votes 0


DonkeyKong with guns
by TheMatrixSheep, 2016/8/26
The Kong will have his Revenge
plays 372 / votes 0

www.greenfoot.org/scenarios/17515

Greenfoot | +

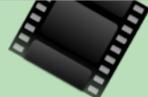
www.greenfoot.org/doc

LIBRARY WEB ADMIN CODE EDU LABS SJSU 202 281 279 DOCS ZAPP BOOKS

5. Adding a Randomly Moving Enemy
6. How to Access One Object From Another

introduction to Greenfoot. A wide range of other short videos are also available.

German translations are available for some videos (thanks to Frajo Ligmann).



Going Further

There are many more things you can do with Greenfoot, such as:

- Running Greenfoot projects on Netbeans. (New)
- Dynamic helper classes. (New)
- Kinect with Greenfoot.
- PicoBoard with Greenfoot.
- Finch with Greenfoot.
- Loading native libraries.
- Gamepads with Greenfoot.
- CS Unplugged with Greenfoot.
- AP Computer Science with Greenfoot.
- Learn Maths with Greenfoot



An installation and configuration FAQ is available.

About, Supported By

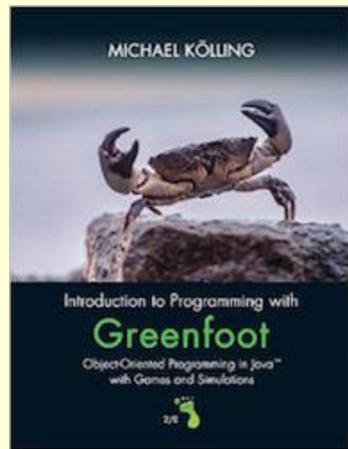
University of Kent

ORACLE®

Introduction to Programming with Greenfoot

www.greenfoot.org/book/

LIBRARY WEB ADMIN CODE EDU LABS SJSU 202 281 279 DOCS ZAPP BOOKS



MICHAEL KÖLLING
Introduction to Programming with Greenfoot
Object-Oriented Programming in Java™ with Games and Simulations

Michael Kölling
Introduction to Programming with Greenfoot
Object-Oriented Programming in Java™ with Games and Simulations

PEARSON Education

Second edition, Pearson, 2016
ISBN-10: 013-405429-6
ISBN-13: 978-013-405429-2

New in the second edition: Several new chapters; new scenarios; end-of-chapter drill and practice sections added; more gradual introduction; improvement of presentation of key concepts; new Greenfoot features included;

Book features

- objects-first approach
- project driven
- hands-on learning
- engaging, fun scenarios
- develop games and simulations
- fully integrated with Greenfoot
- teaches standard Java

Teachers: Sign up to the Greenroom

The Greenroom is a teacher community and provides resources (slides, worksheets, project ideas, tests, etc.) and a teacher discussion forum.

Book Information	Resources	Amazon US	Amazon UK	Amazon DE
Introduction	Get Greenfoot	amazon	amazon.co.uk	amazon.de
About the 2nd edition	The Book scenarios (21.2 Mb)	Introduction to	Introduction to	Introduction to
Scenarios discussed in the book	Discussion forum			
	Scenario gallery			

Transferring data from wms-na.amazon-adsystem.com...