Programming in Java

with Greenfoot

http://www.greenfoot.org

http://www.greenfoot.org/doc

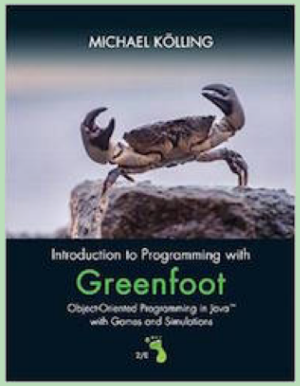
The book

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

The second edition of the book is out now.

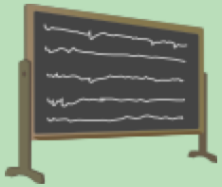
The new edition has several new chapters; new scenarios; end-of-chapter drill and practice sections added; more gradual introduction; improvement of presentaion of key concepts; new Greefoot features included; programming with Microsoft Kinect.

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

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()`	Construct an Actor.
`void act()`	The act method is called by the Greenfoot framework to give objects a chance to perform some action.
`protected void addedToWorld(World world)`	This method is called by the Greenfoot system when the object has been inserted into the world.
`GreenfootImage getImage()`	Return the image used to represent this Actor.
`protected List getIntersectingObjects(Class cls)`	Return all the objects that intersect this object.
`protected List getNeighbours(int distance, boolean diagonal, Class cls)`	Return the neighbours to this object within a given distance.
`protected List getObjectsAtOffset(int dx, int dy, Class cls)`	Return all objects that intersect the given location (relative to this object's location).
`protected List getObjectsInRange(int r, Class cls)`	Return all objects within range 'r' around this object.
`protected Actor getOneIntersectingObject (Class cls)`	Return an object that intersects this object.
`protected Actor getOneObjectAtOffset (int dx, int dy, Class cls)`	Return one object that is located at the specified cell (relative to this object's location).
`int getRotation()`	Return the current rotation of the object.
`World getWorld()`	Return the world that this object lives in.
`int getX()`	Return the x-coordinate of the object's current location.
`int getY()`	Return the y-coordinate of the object's current location.
`protected boolean intersects(Actor other)`	Check whether this object intersects another given object.
`void setImage(GreenfootImage image)`	Set the image for this object to the specified image.
`void setImage(String filename)`	Set an image for this object from an image file.
`void setLocation(int x, int y)`	Assign a new location for this object.
`void setRotation(int rotation)`	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)`	*True* if the mouse has been clicked on the given object.
`static boolean mouseDragEnded(Object obj)`	*True* if a mouse drag has ended.
`static boolean mouseDragged(Object obj)`	*True* if the mouse has been dragged on the given object.
`static boolean mouseMoved(Object obj)`	*True* if the mouse has been moved on the given object.
`static boolean mousePressed(Object obj)`	*True* 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

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

Class GreenfootImage

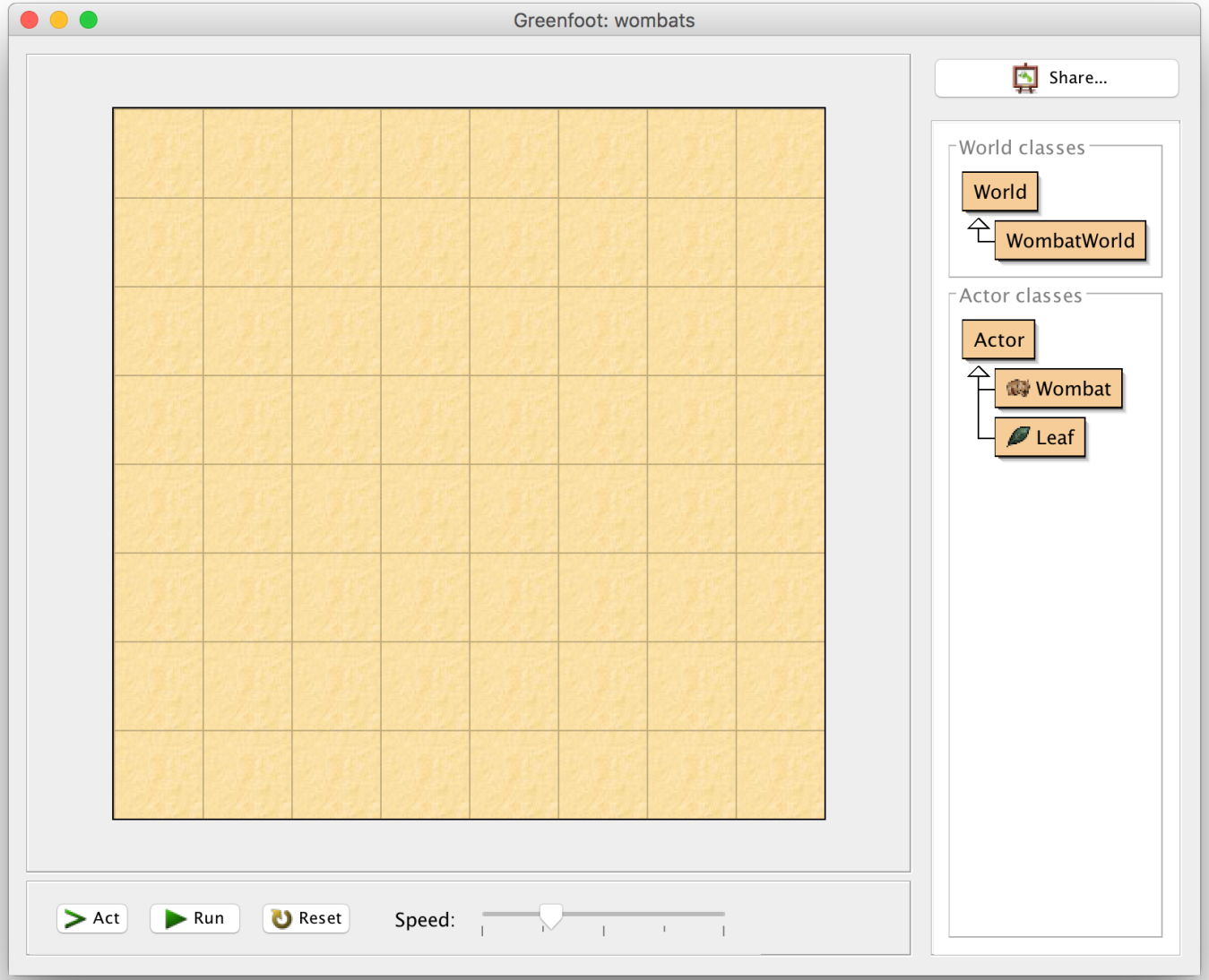
`GreenfootImage``(GreenfootImage image)`	Create a GreenfootImage from another GreenfootImage.
`GreenfootImage``(int width, int height)`	Create an empty (transparent) image with the specified size.
`GreenfootImage``(String filename)`	Create an image from an image file.
`void` **`clear`**`()`	Clear the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

(*continued*)

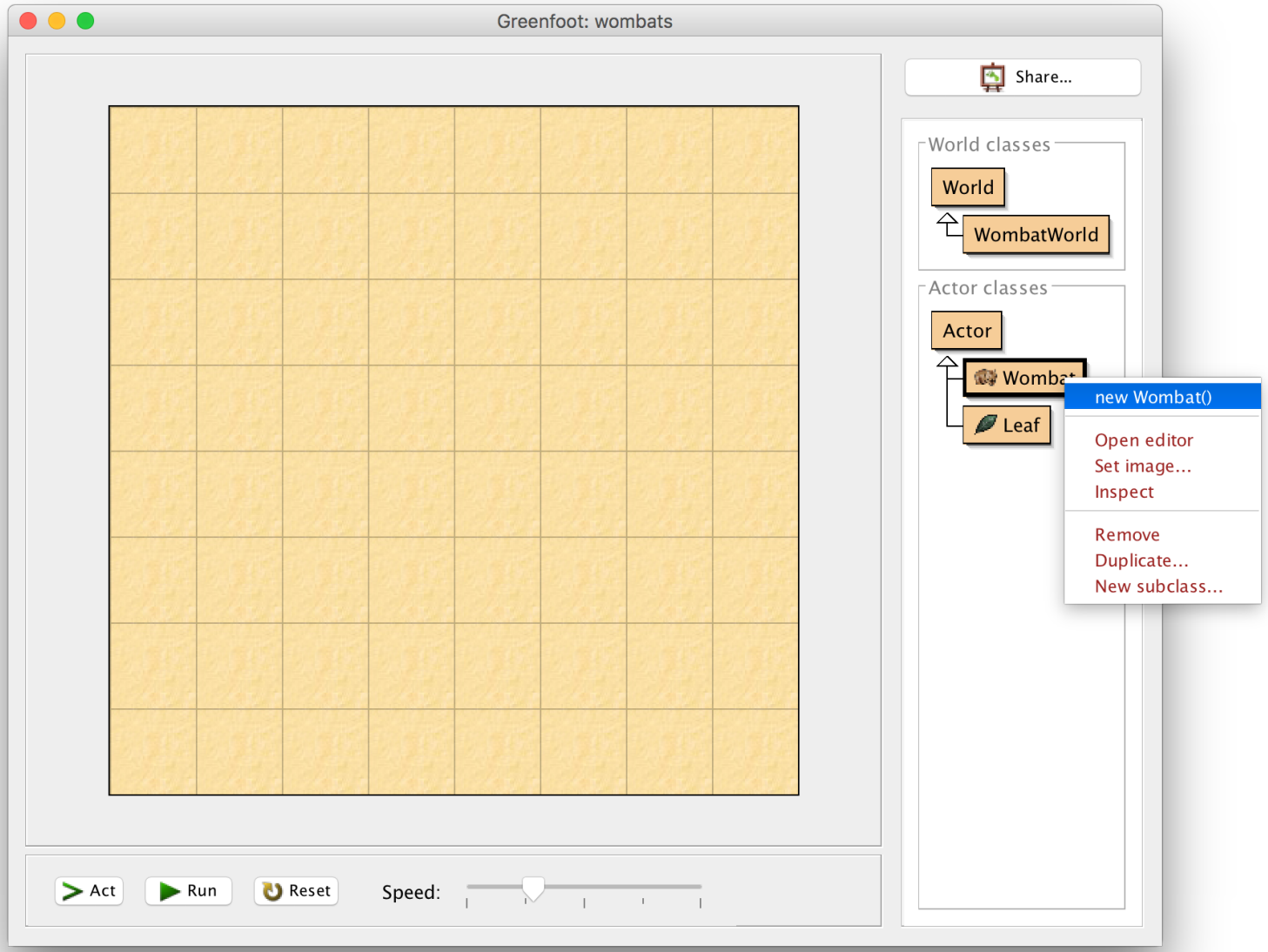
Class GreenfootImage (continued)

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

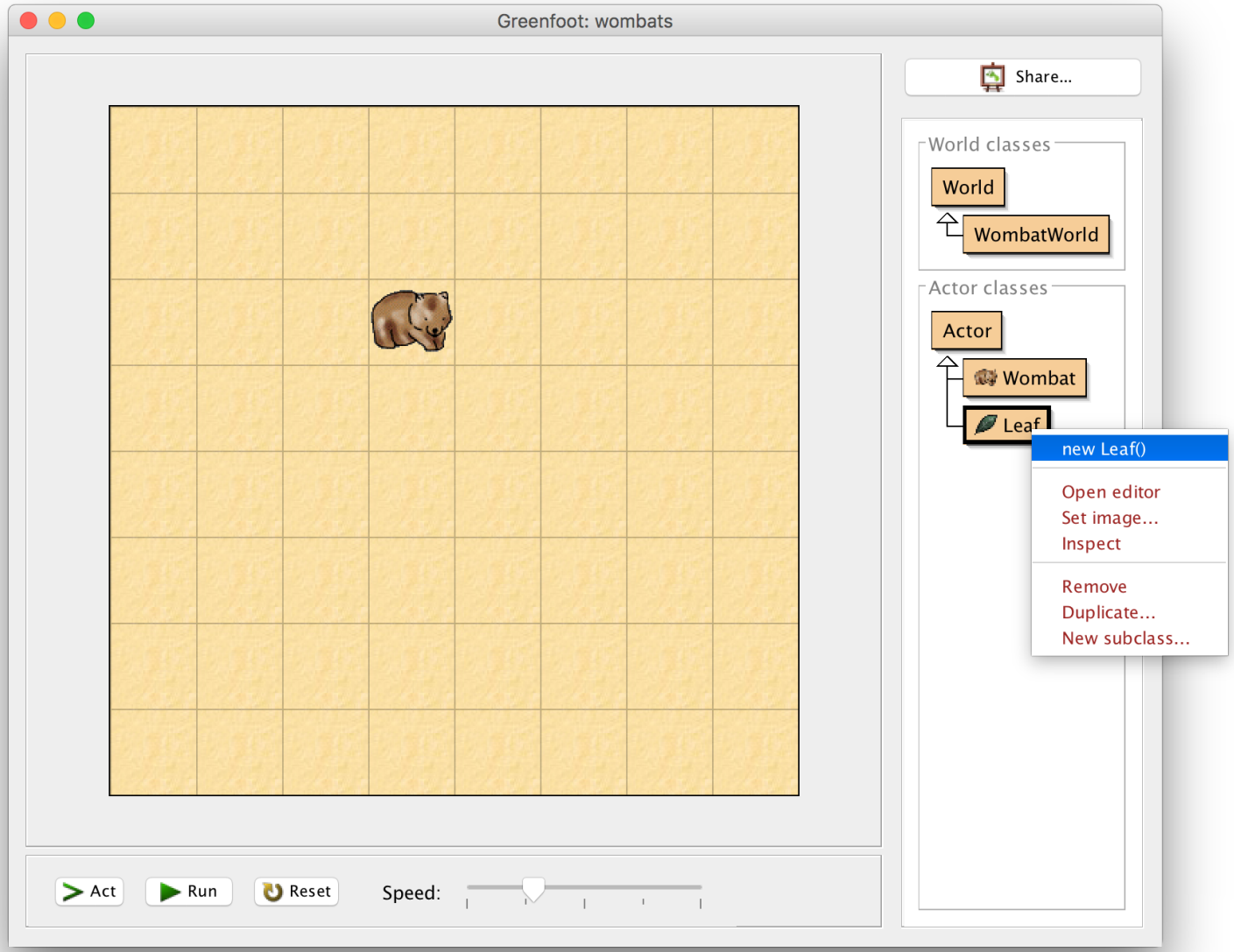
Part 1 - Wombat



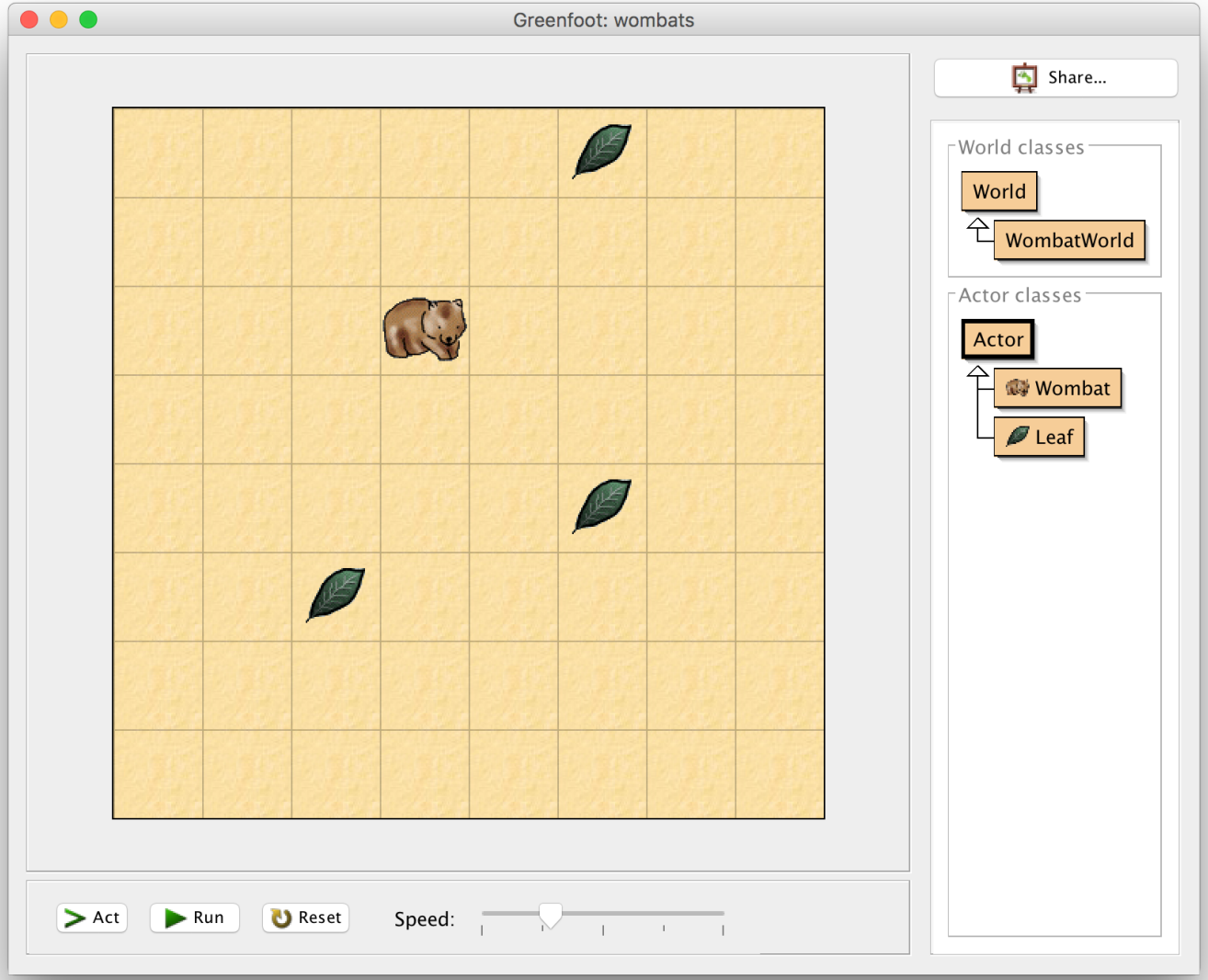
Create a Wombat



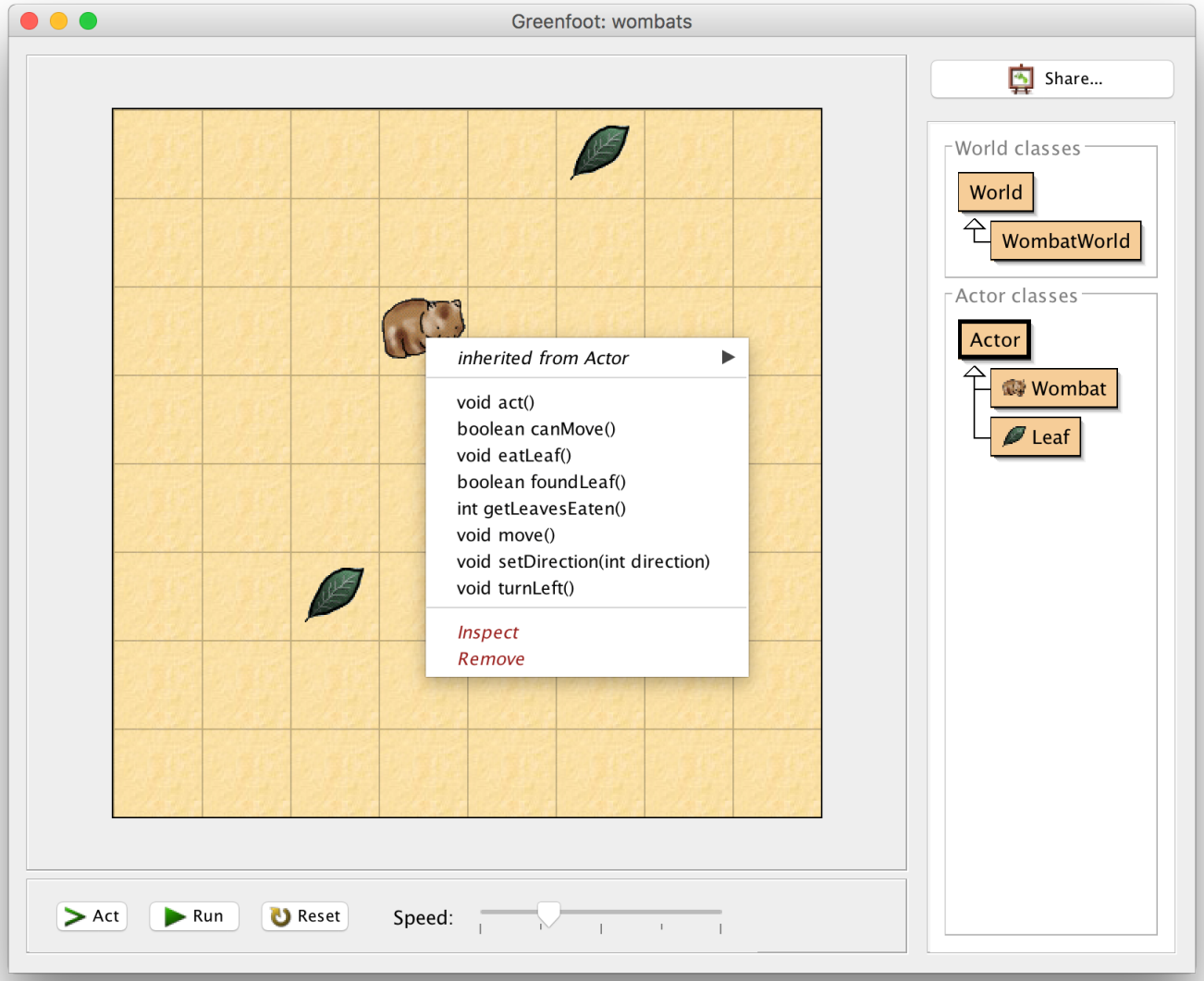
Create one or more Leaves



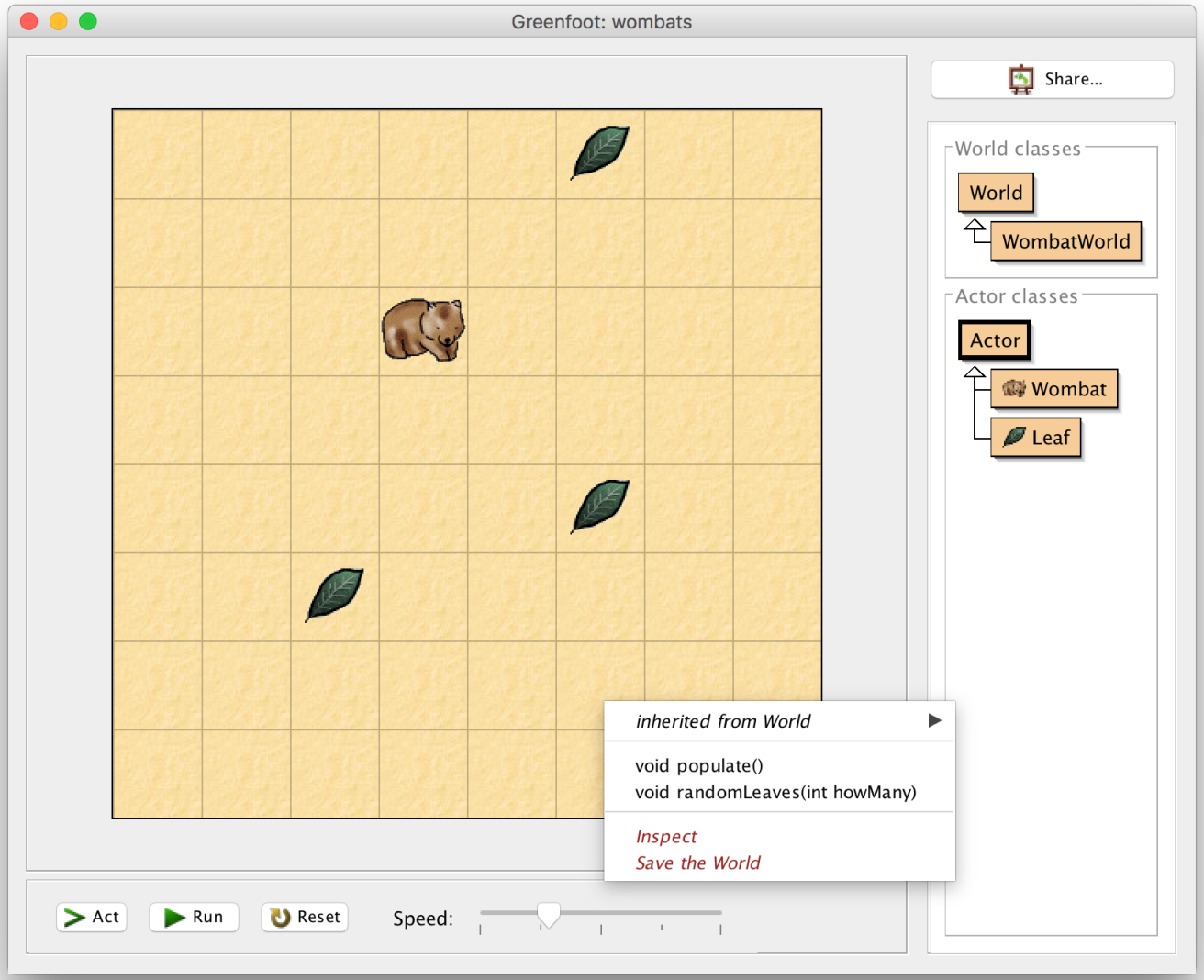
What happens on click of “Act”? “Run”? “Reset”?



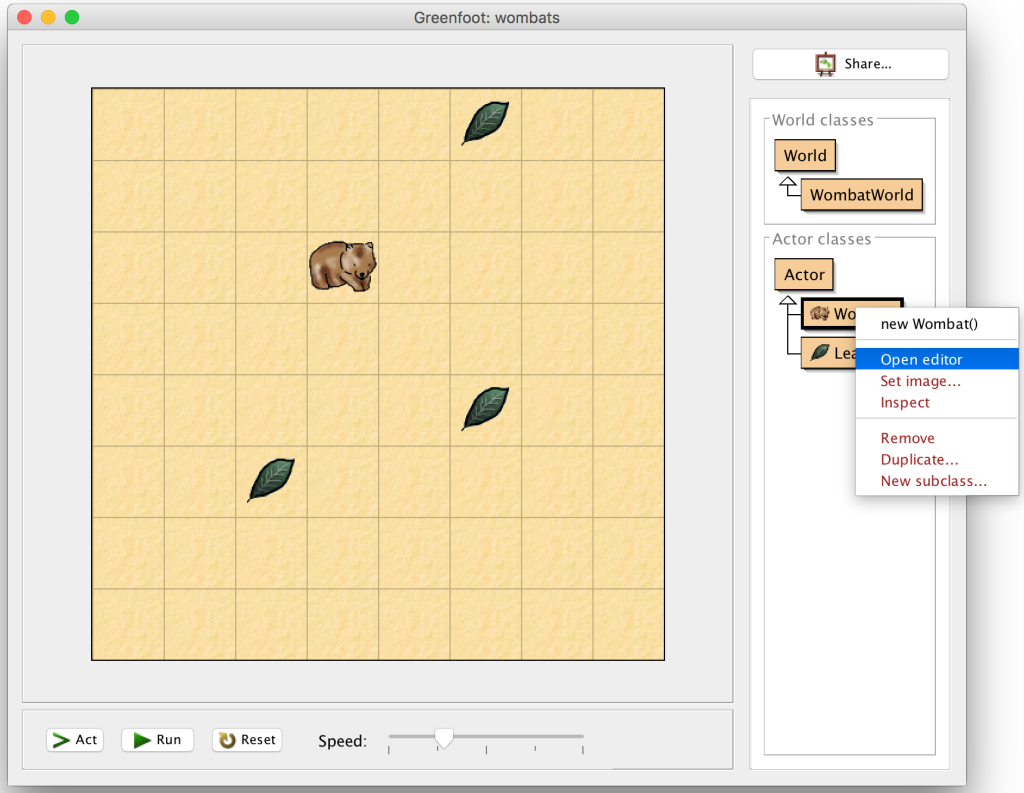
Objects have “Methods”

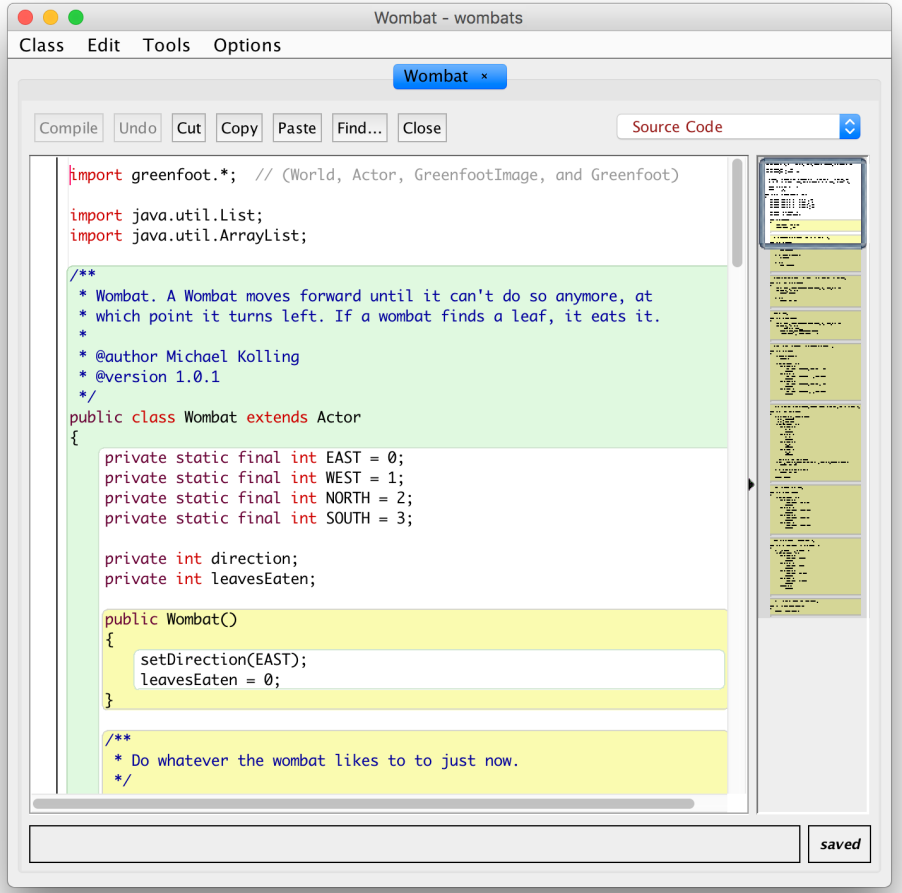


The “World” has Methods.

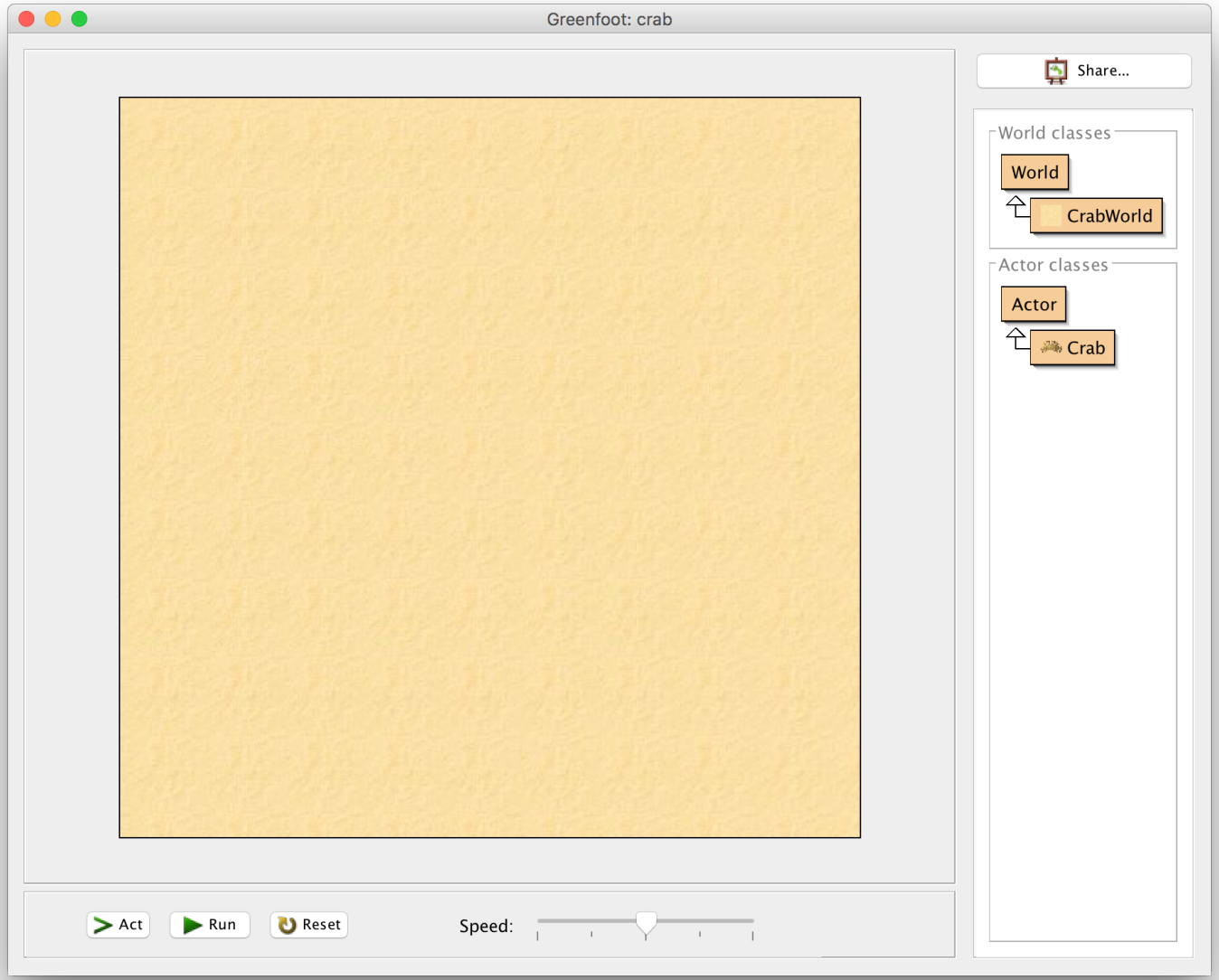


Edit Code of Objects

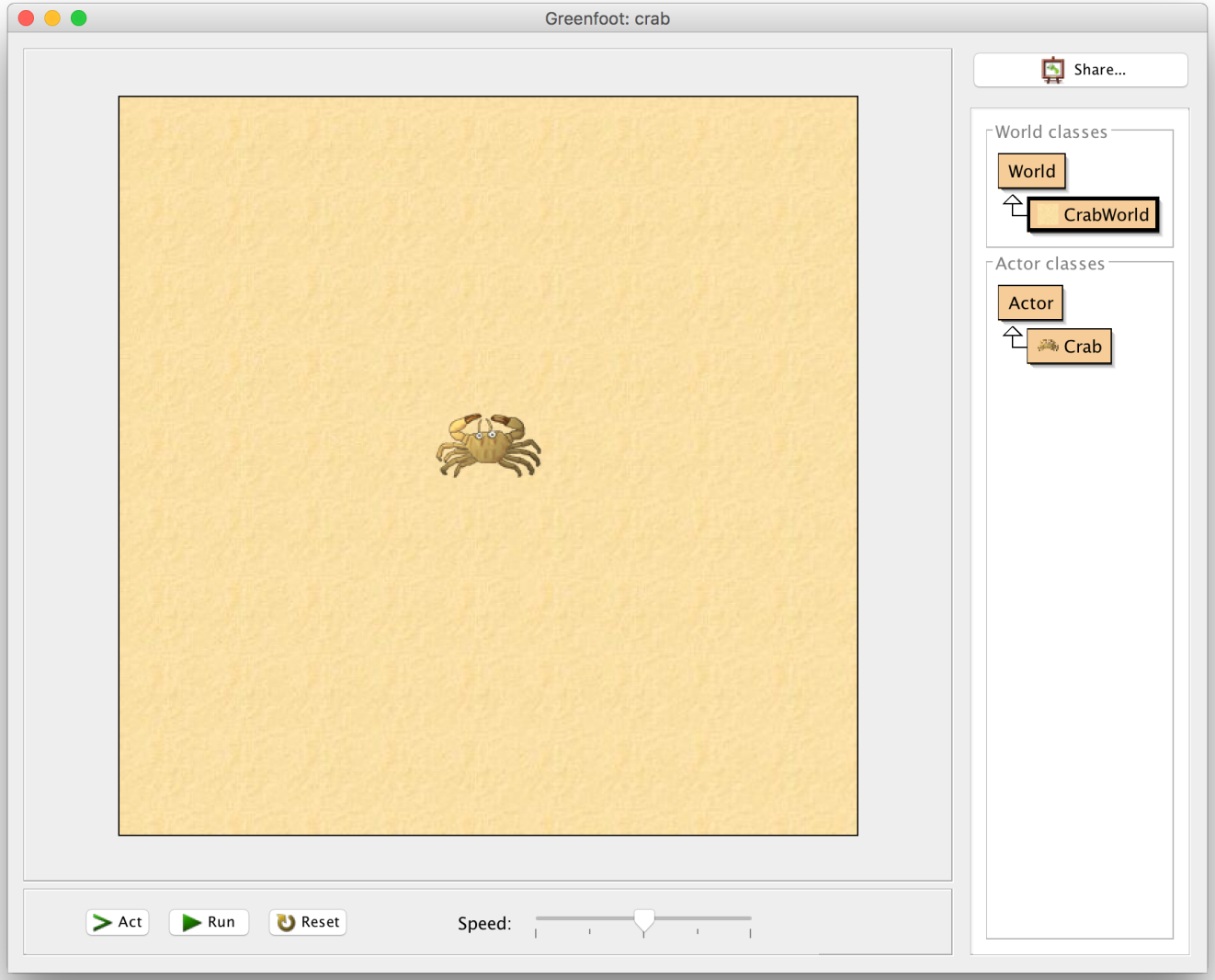




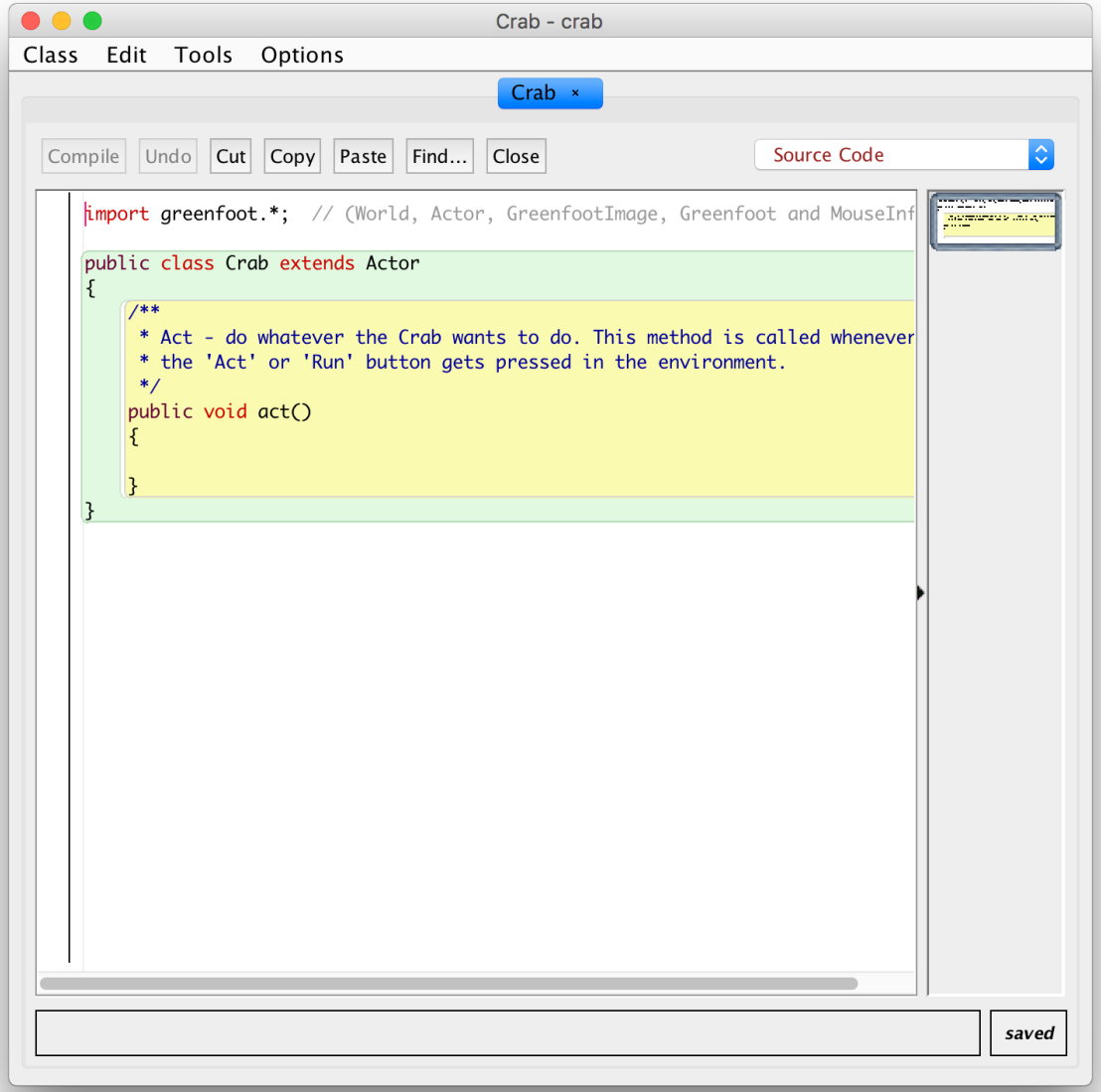
Part 2 - Crab


Greenfoot: crab
Share...
World classes
World
CrabWorld
Actor classes
Actor
Crab
Act
Run
Reset
Speed:

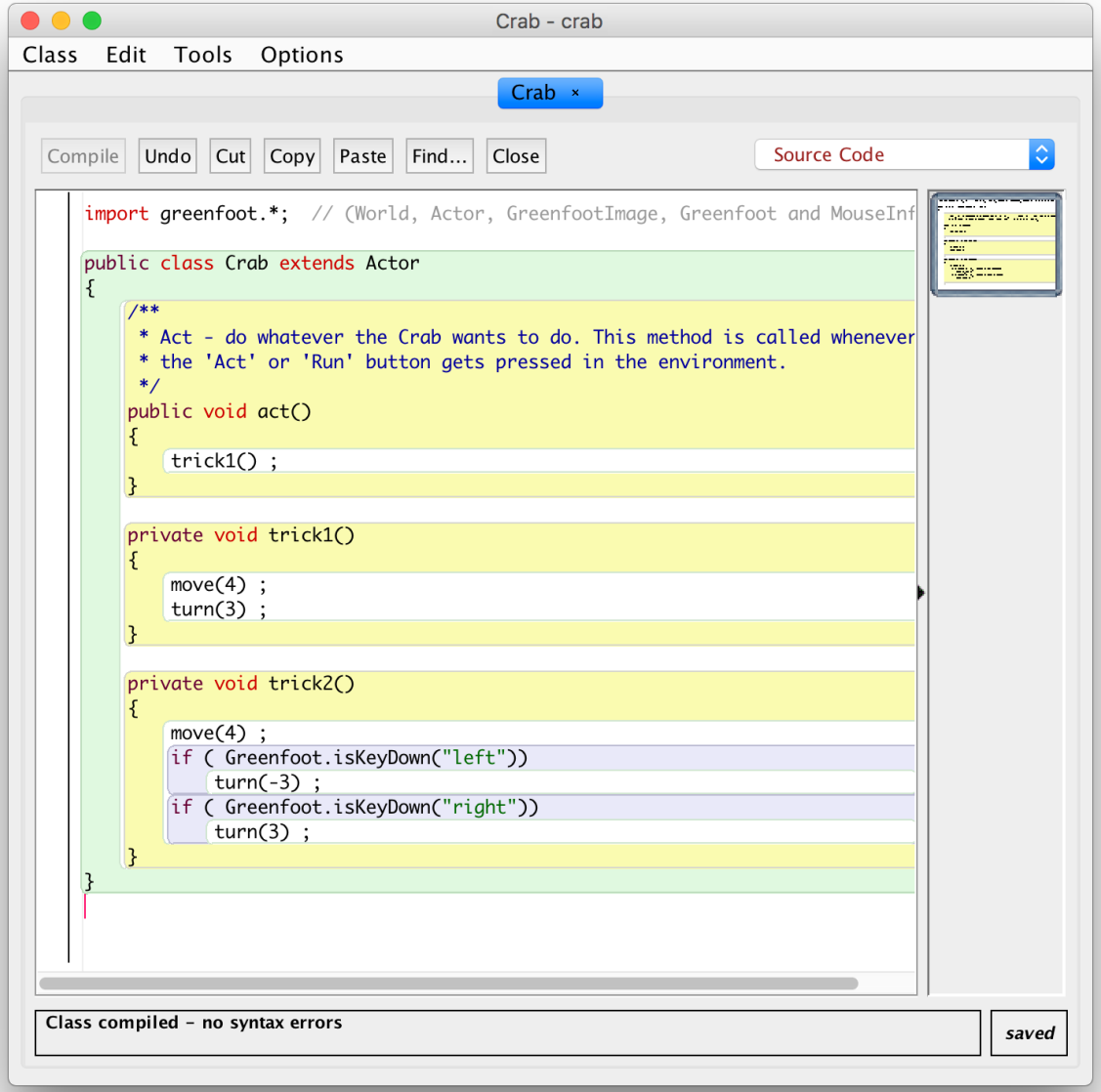
Create a “Crab” and click “Run”



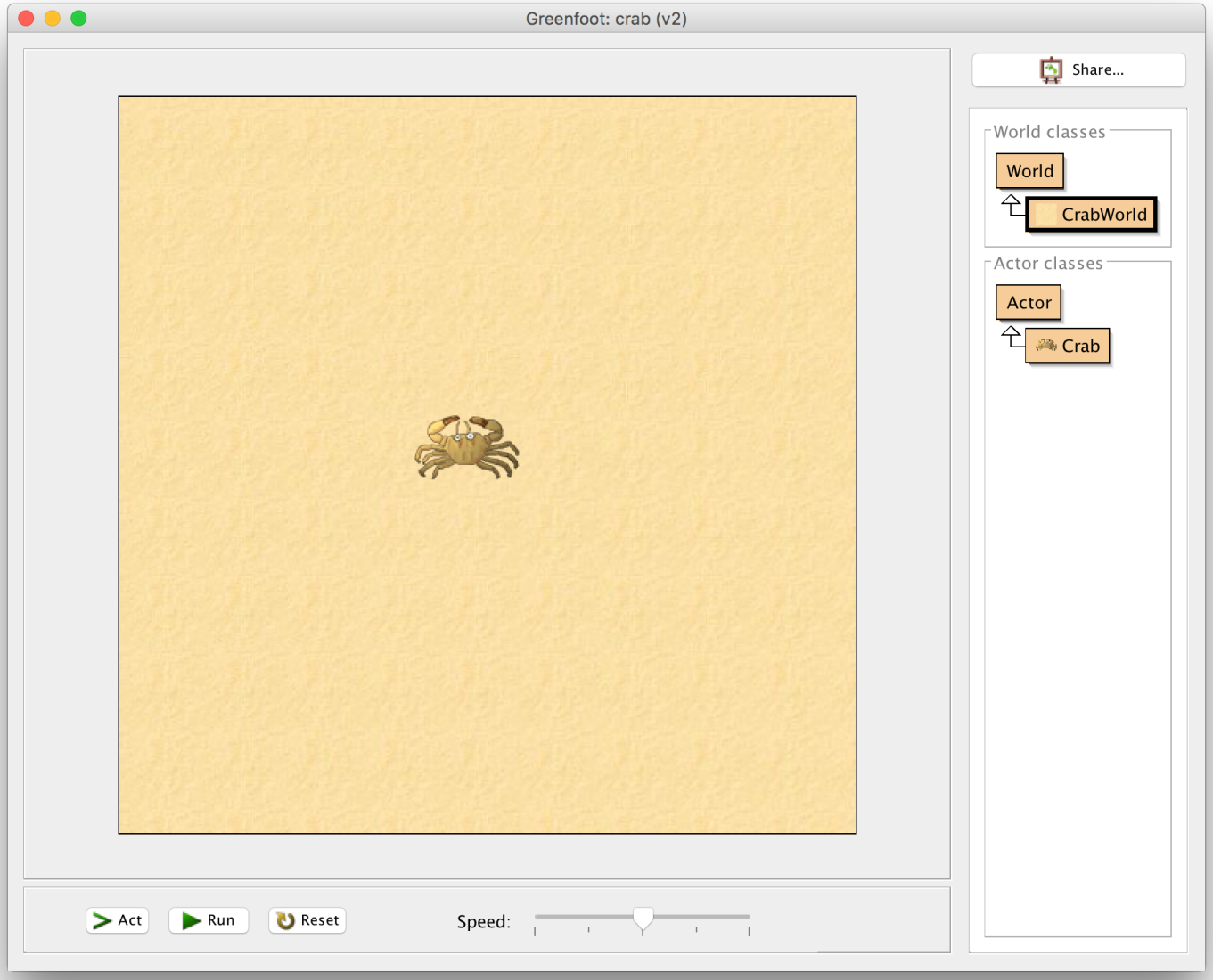
The “Lazy” Crab!



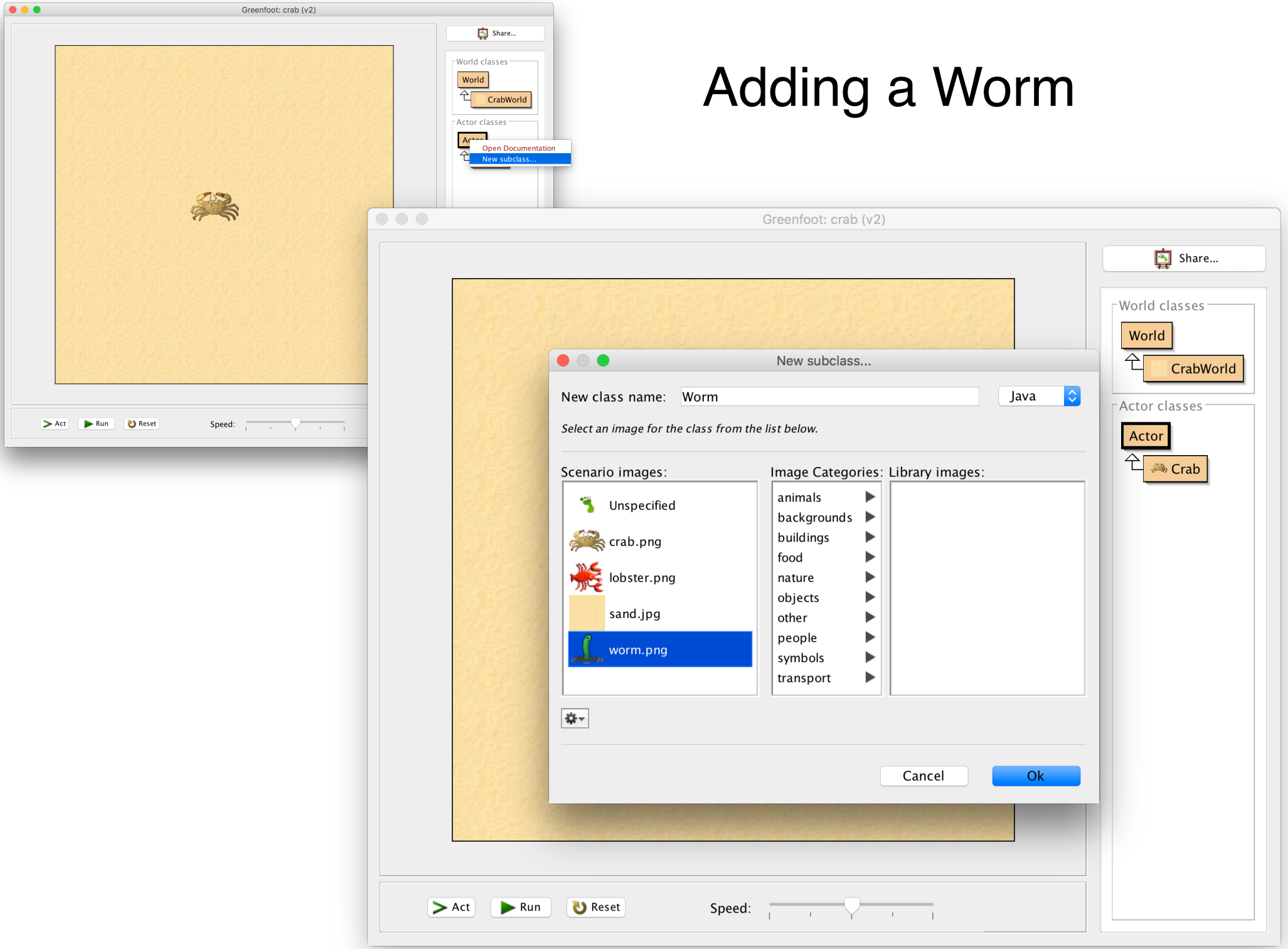
Teach the “Crab” some Tricks!

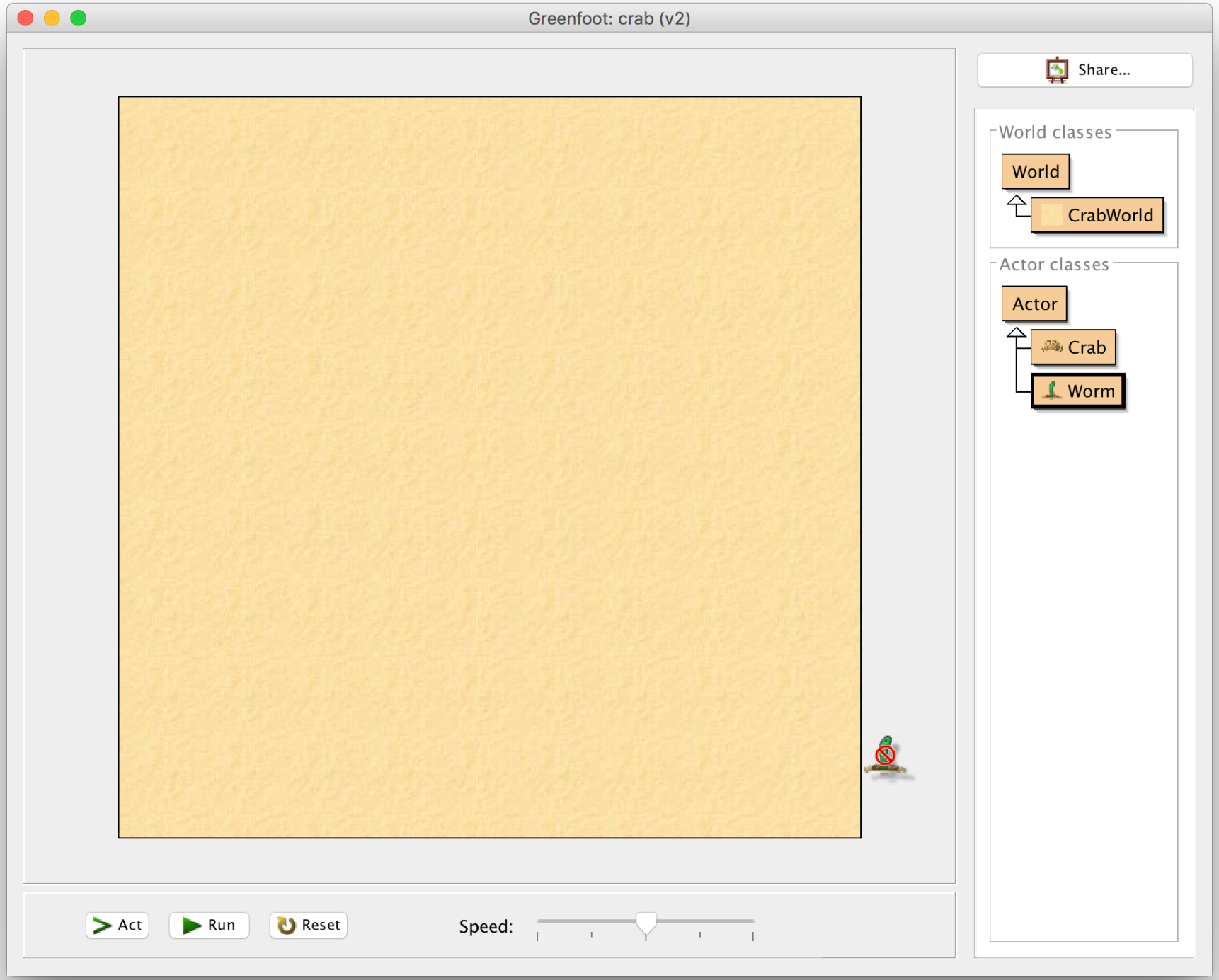


Part 3 - Eating Worms

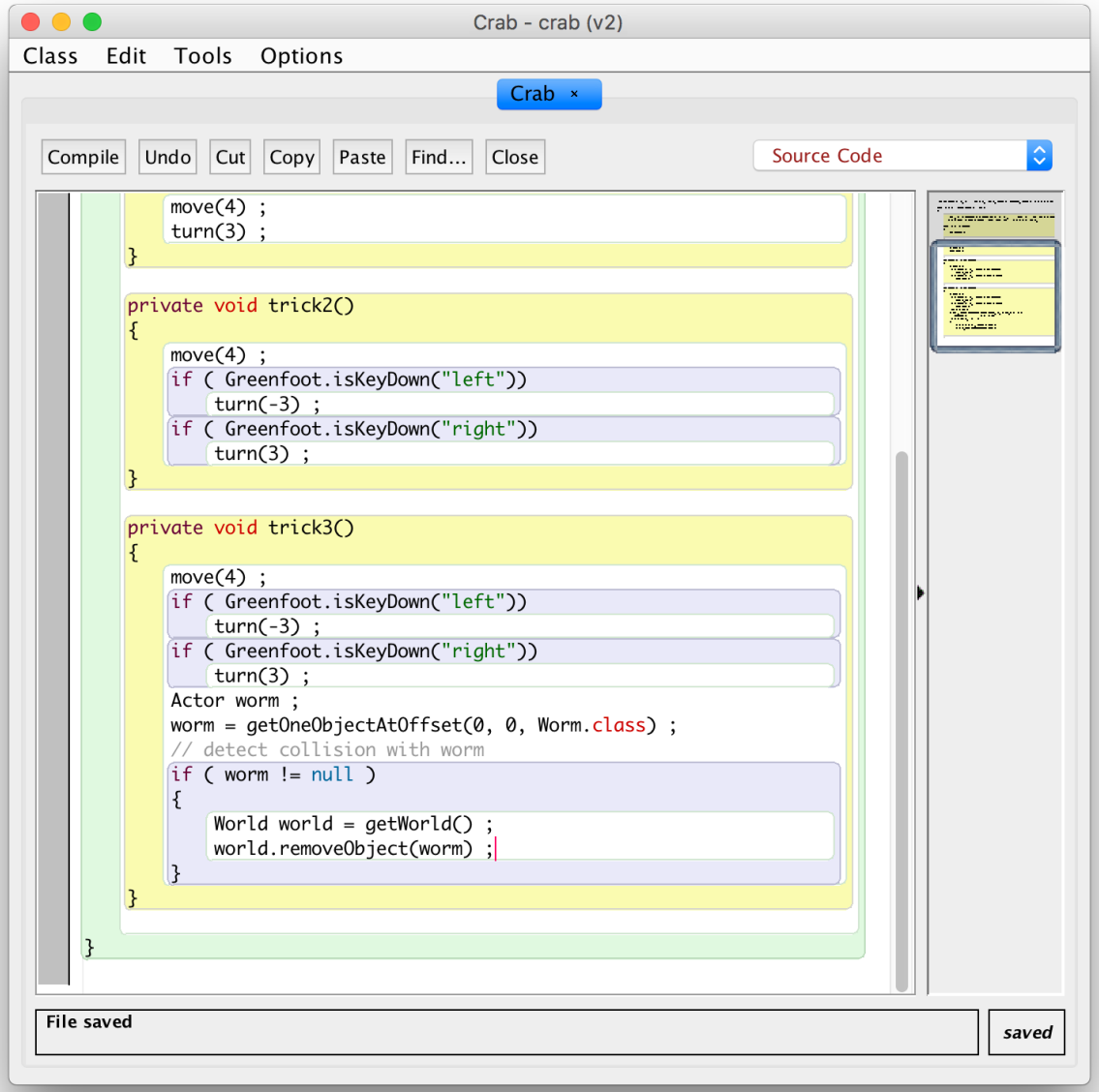


Adding a Worm

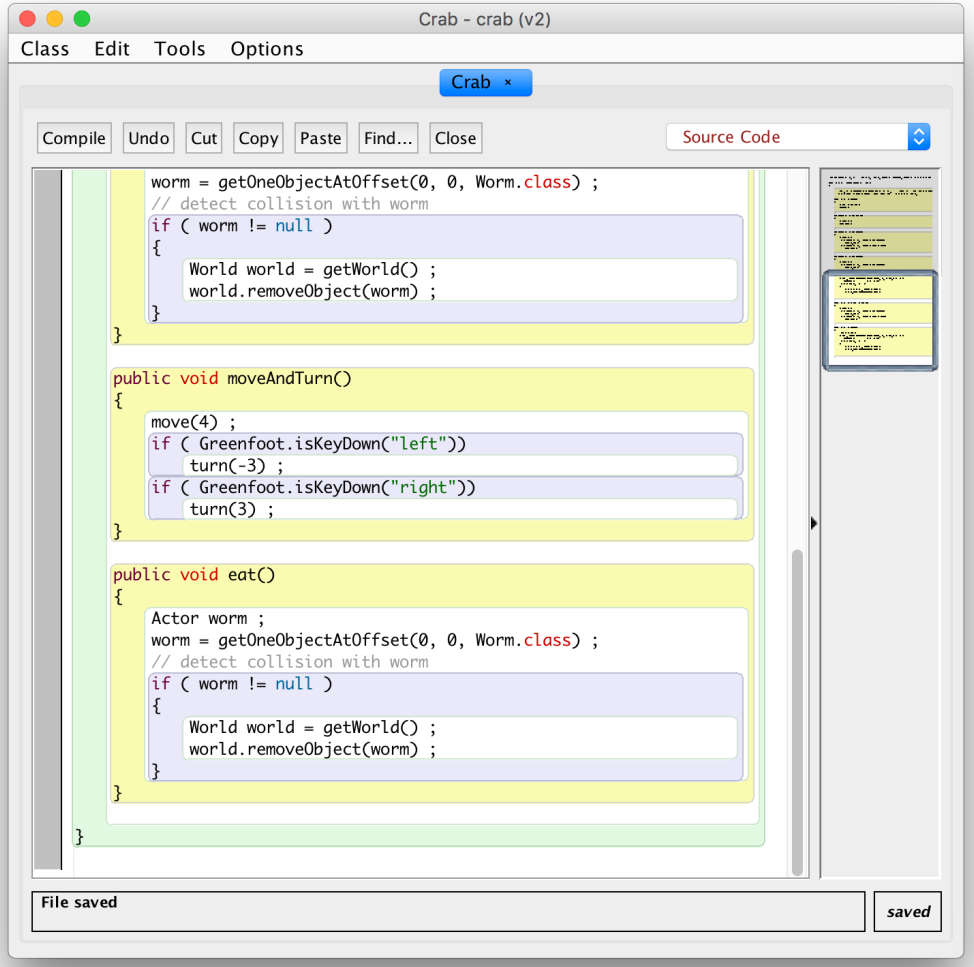


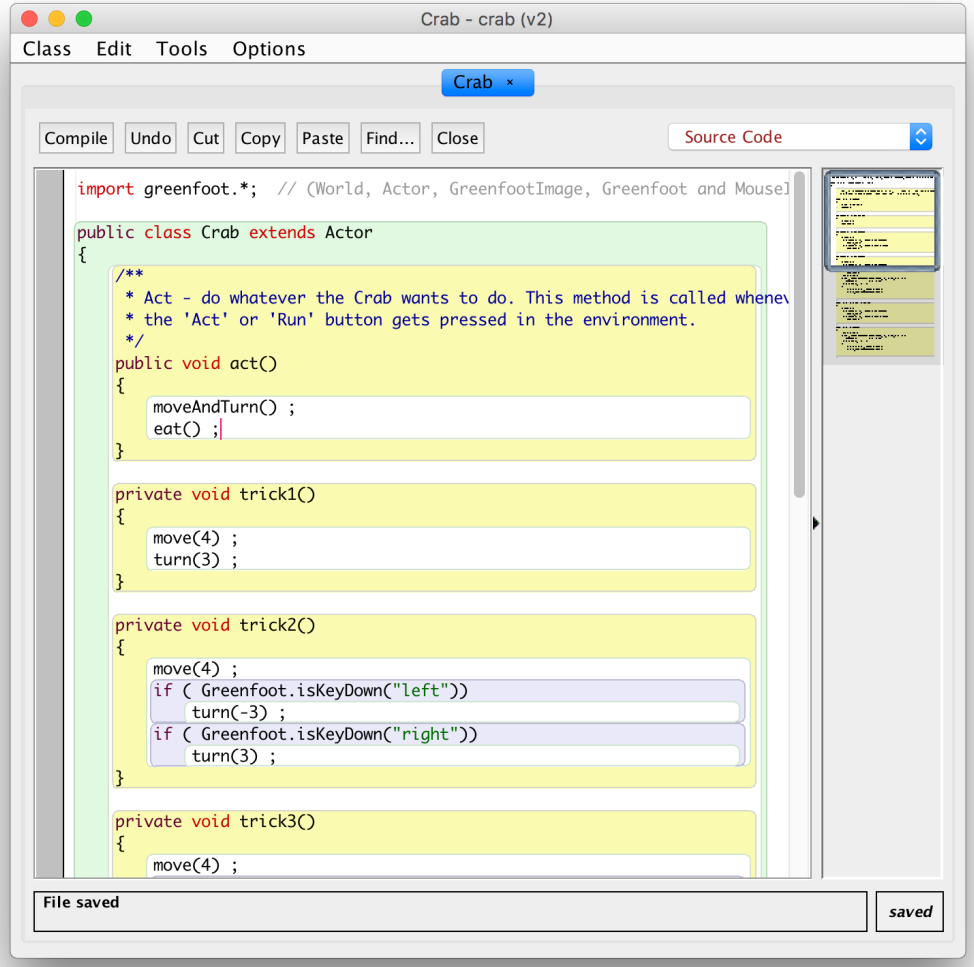

Greenfoot: crab (v2)
Share...
World classes
World
CrabWorld
Actor classes
Actor
Crab
Worm
Act
Run
Reset
Speed:

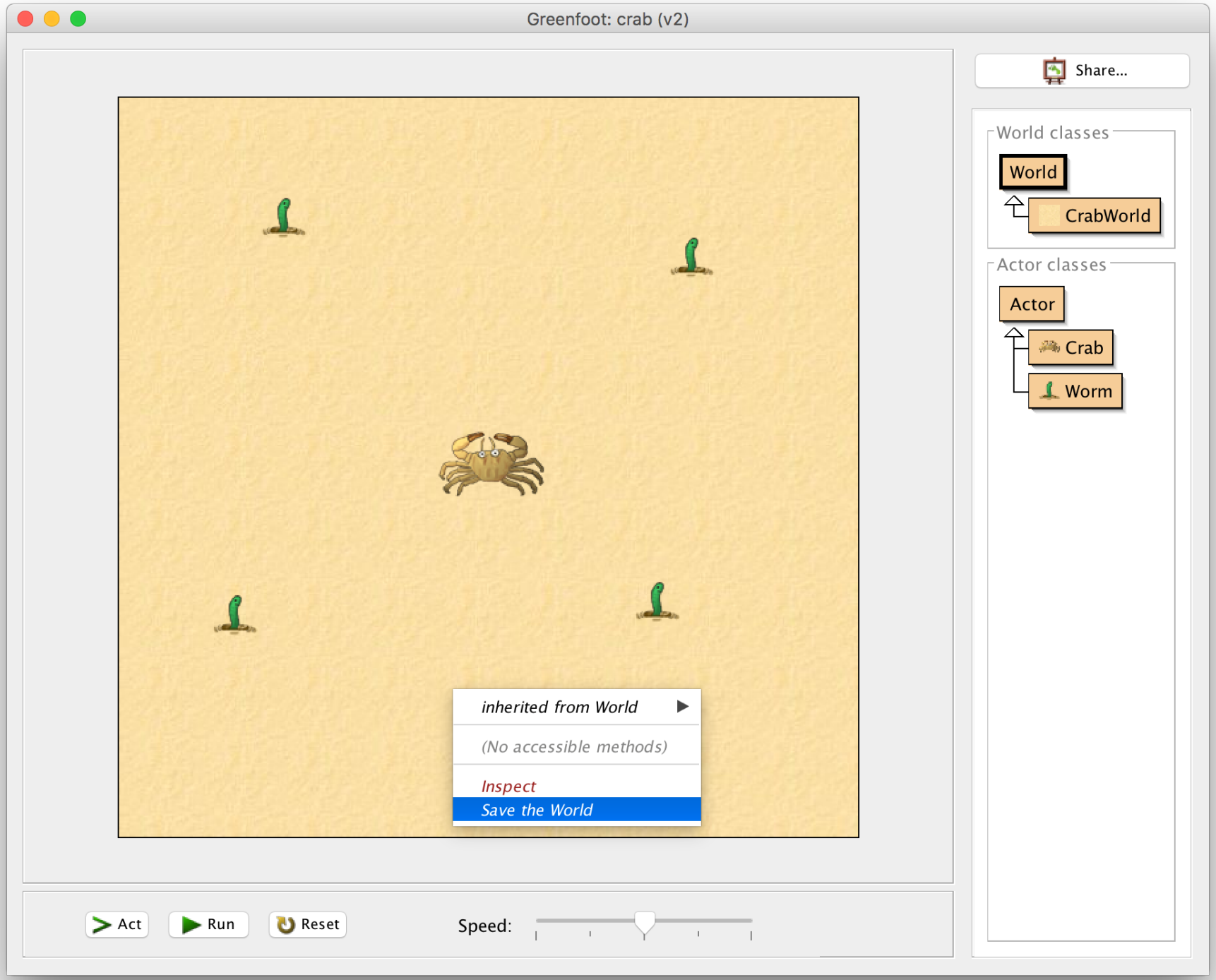
Collision Detection and Removing Objects

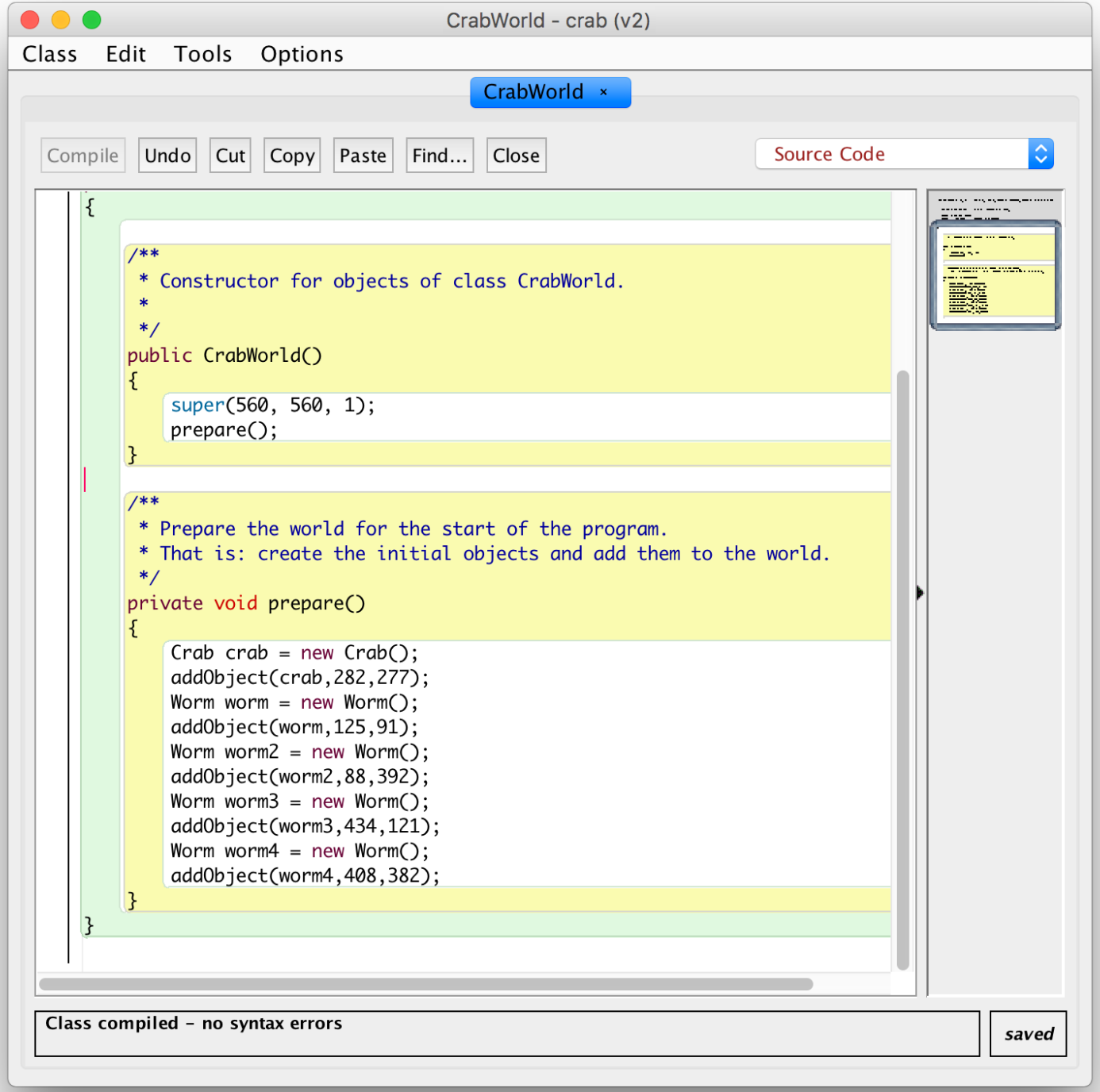


Refactored

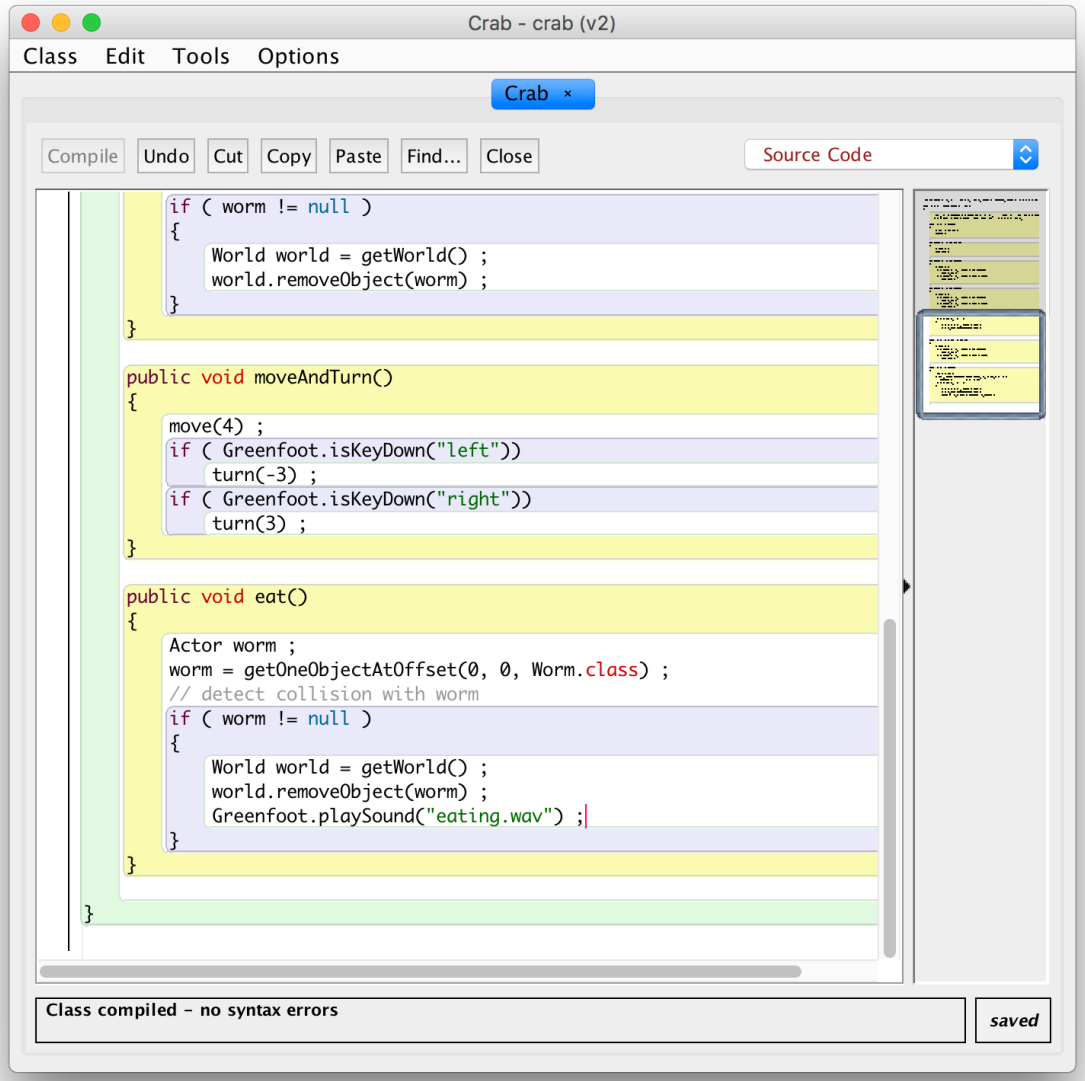

Crab - crab (v2)
Class Edit Tools Options
Crab
Compile Undo Cut Copy Paste Find... Close
Source Code
worm = getOneObjectAtOffset(0, 0, Worm.class) ;
// detect collision with worm
if (worm != null)
{
World world = getWorld() ;
world.removeObject(worm) ;
}
}
public void moveAndTurn()
{
move(4) ;
if (Greenfoot.isKeyDown("left"))
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
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 whene
* 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

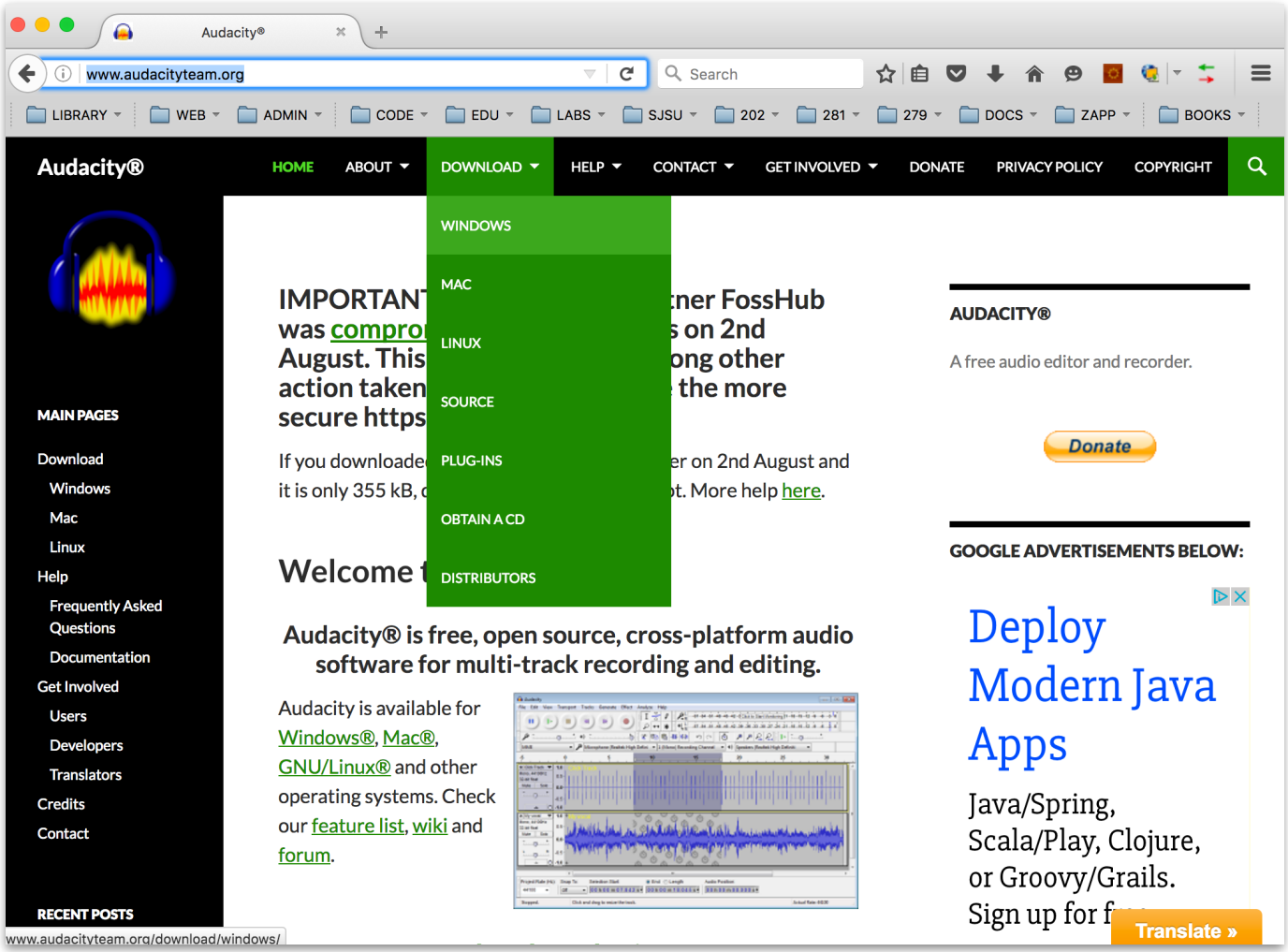

Greenfoot: crab (v2)
Share...
World classes
World
CrabWorld
Actor classes
Actor
Crab
Worm
inherited from World
(No accessible methods)
Inspect
Save the World
Act
Run
Reset
Speed:


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

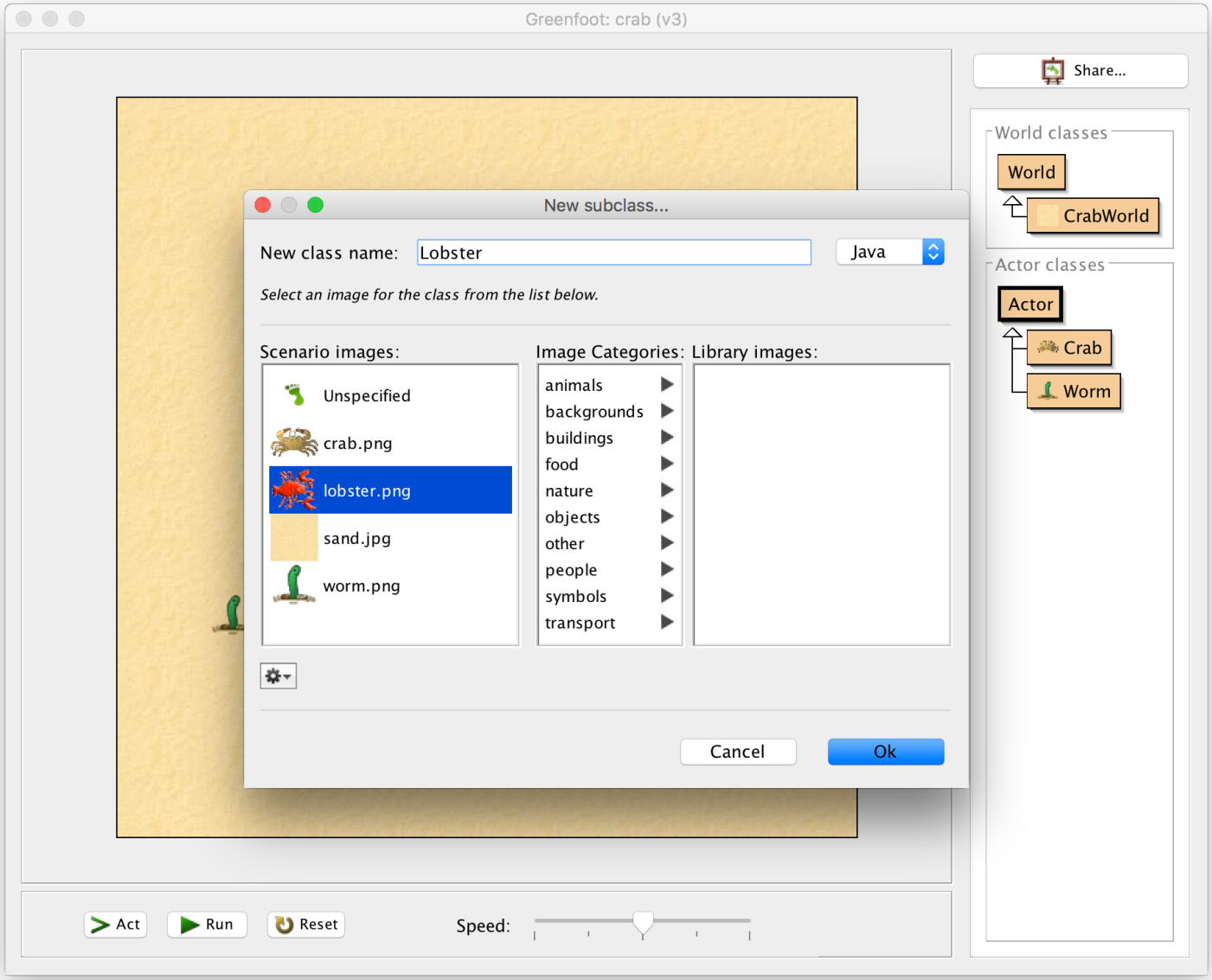


Try Recording your own Sounds

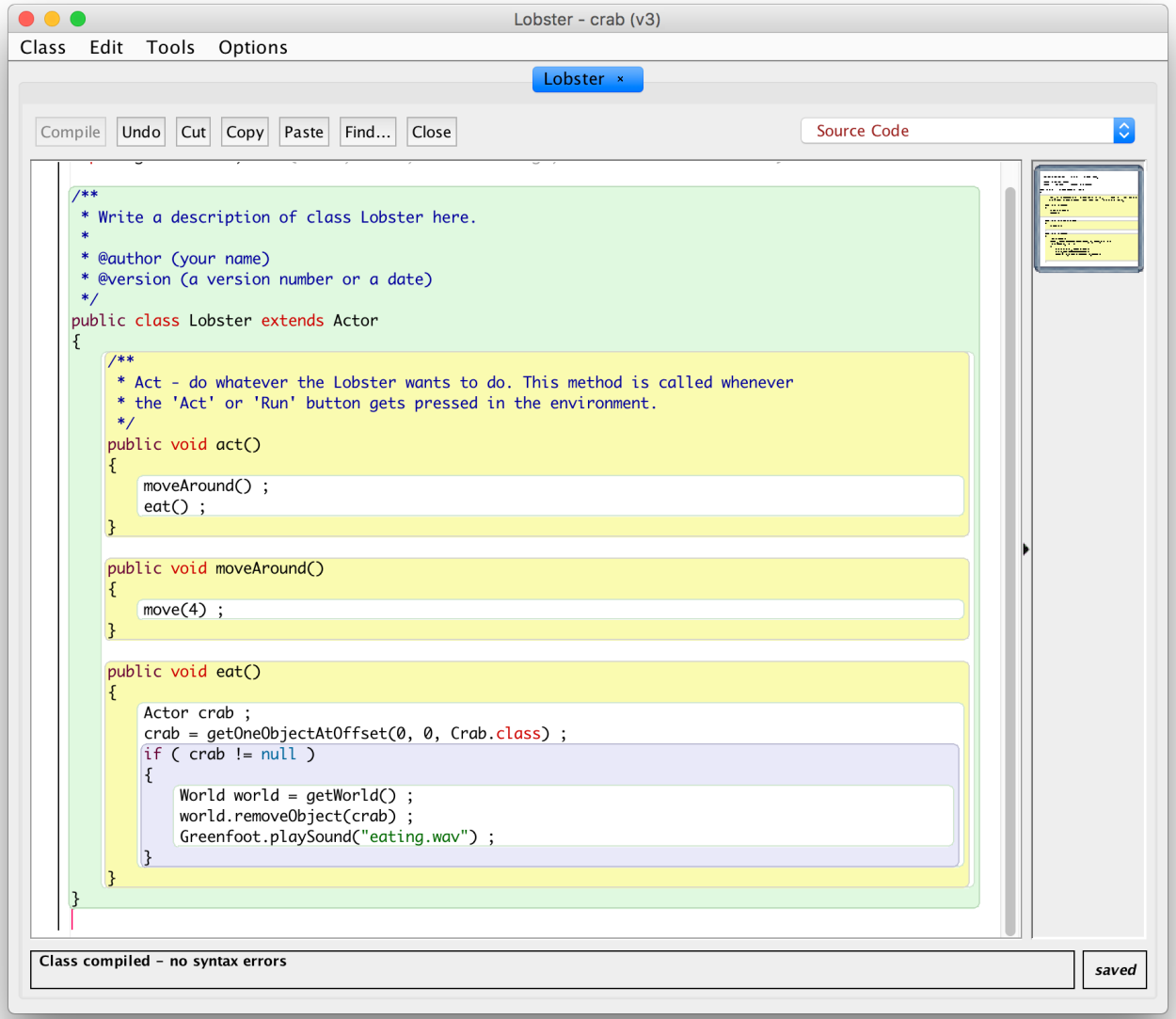


http://www.audacityteam.org/

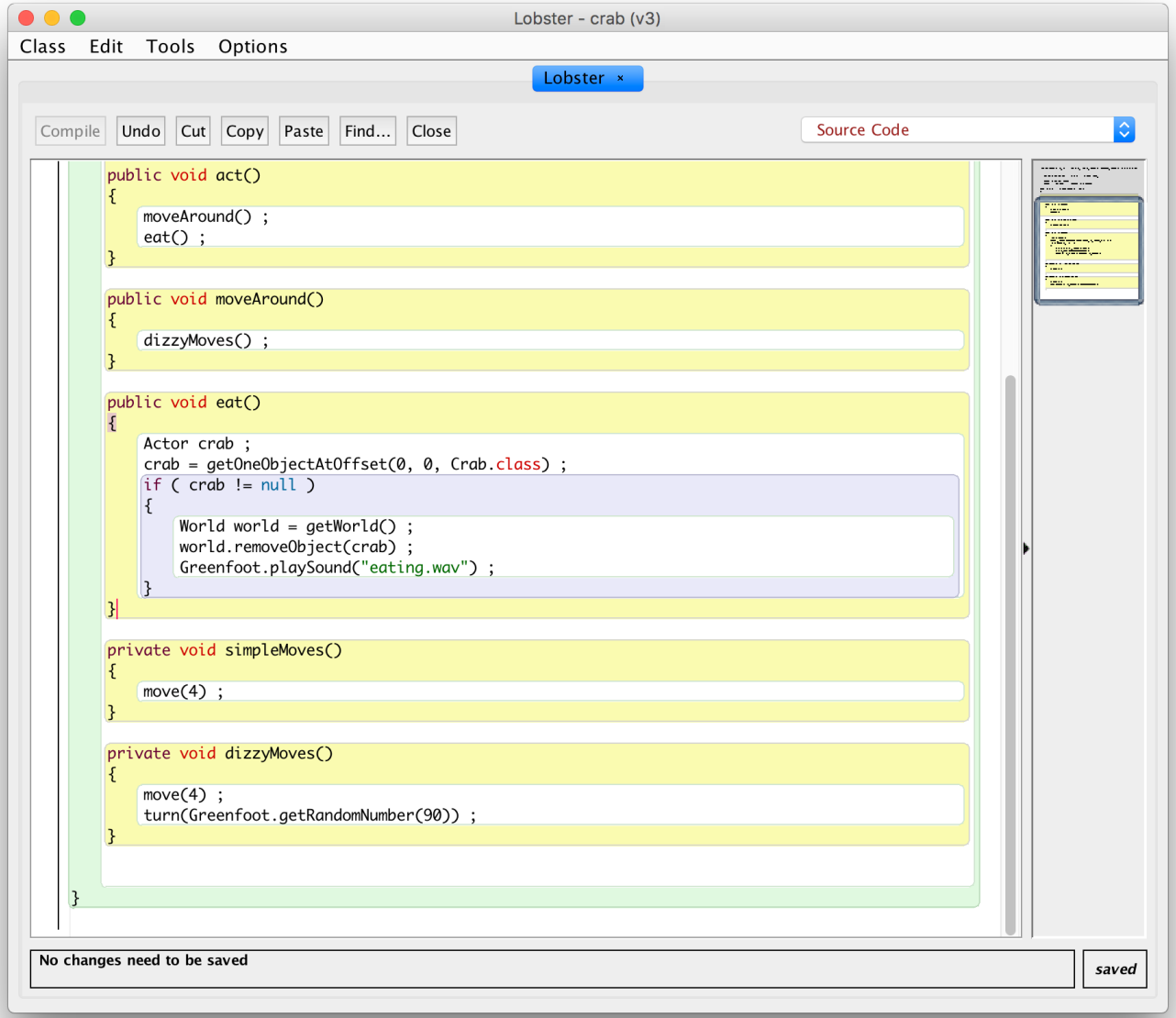
Part 5 - Adding Enemies



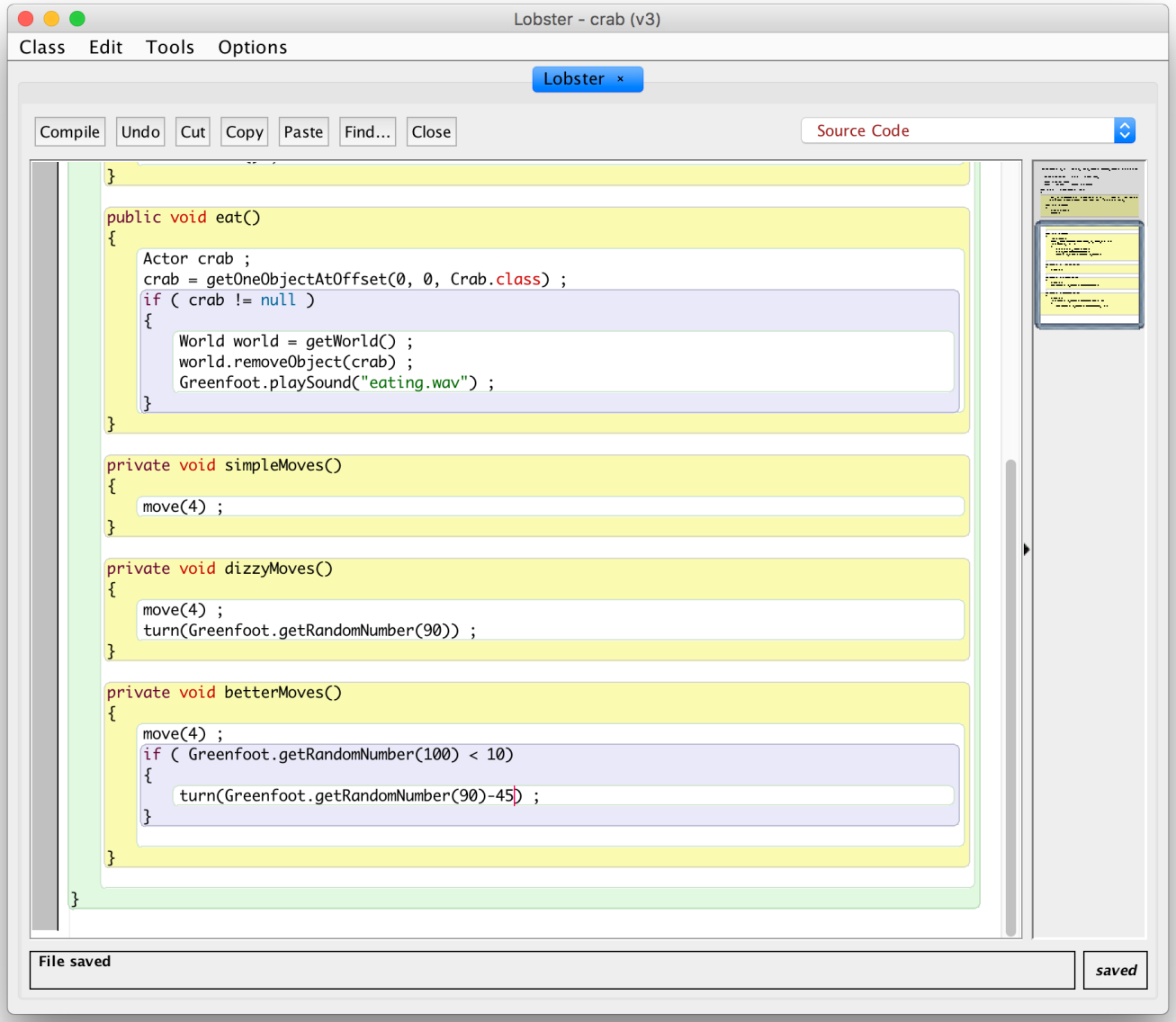
Simple Lobster



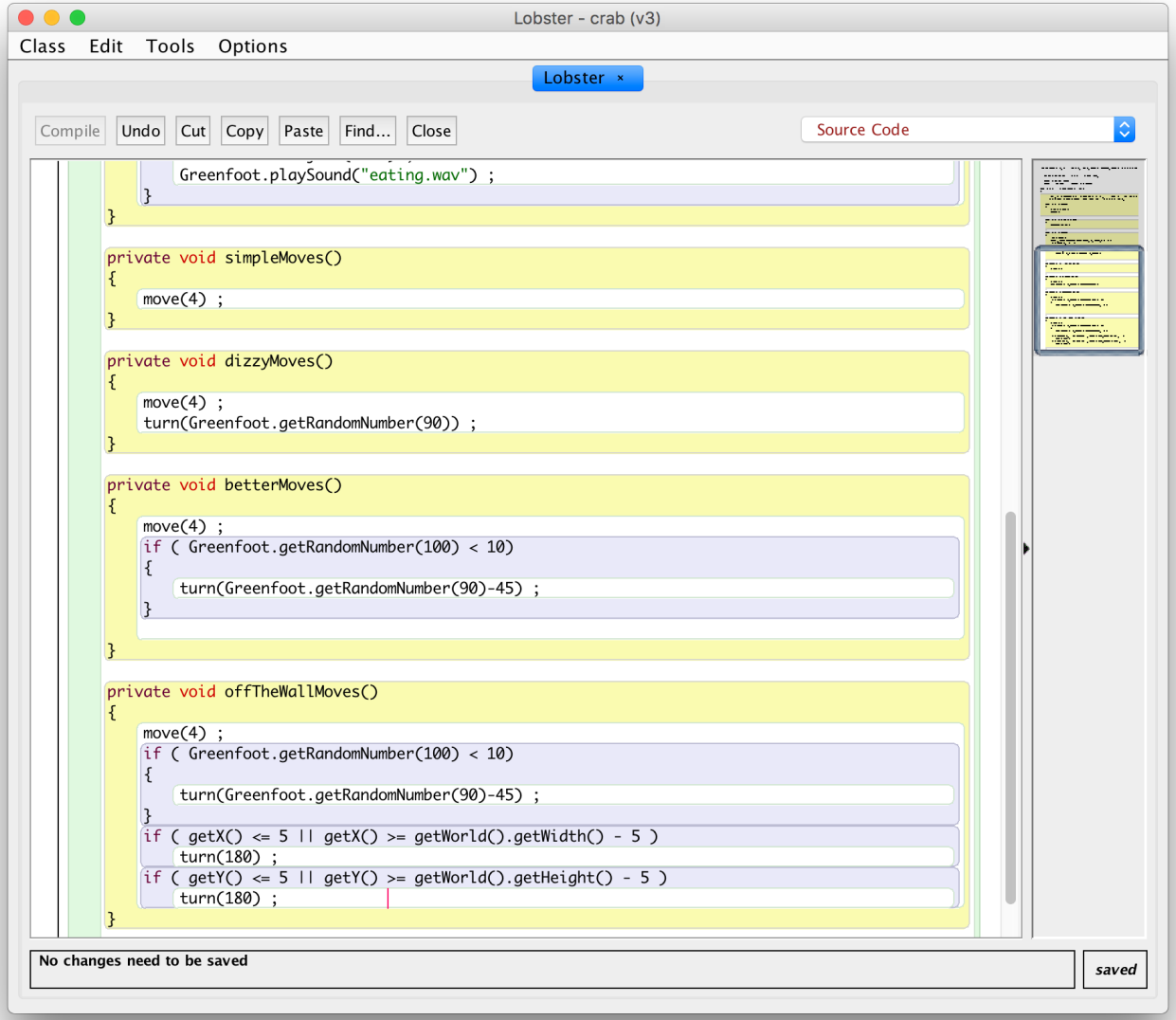
Dizzy Lobster

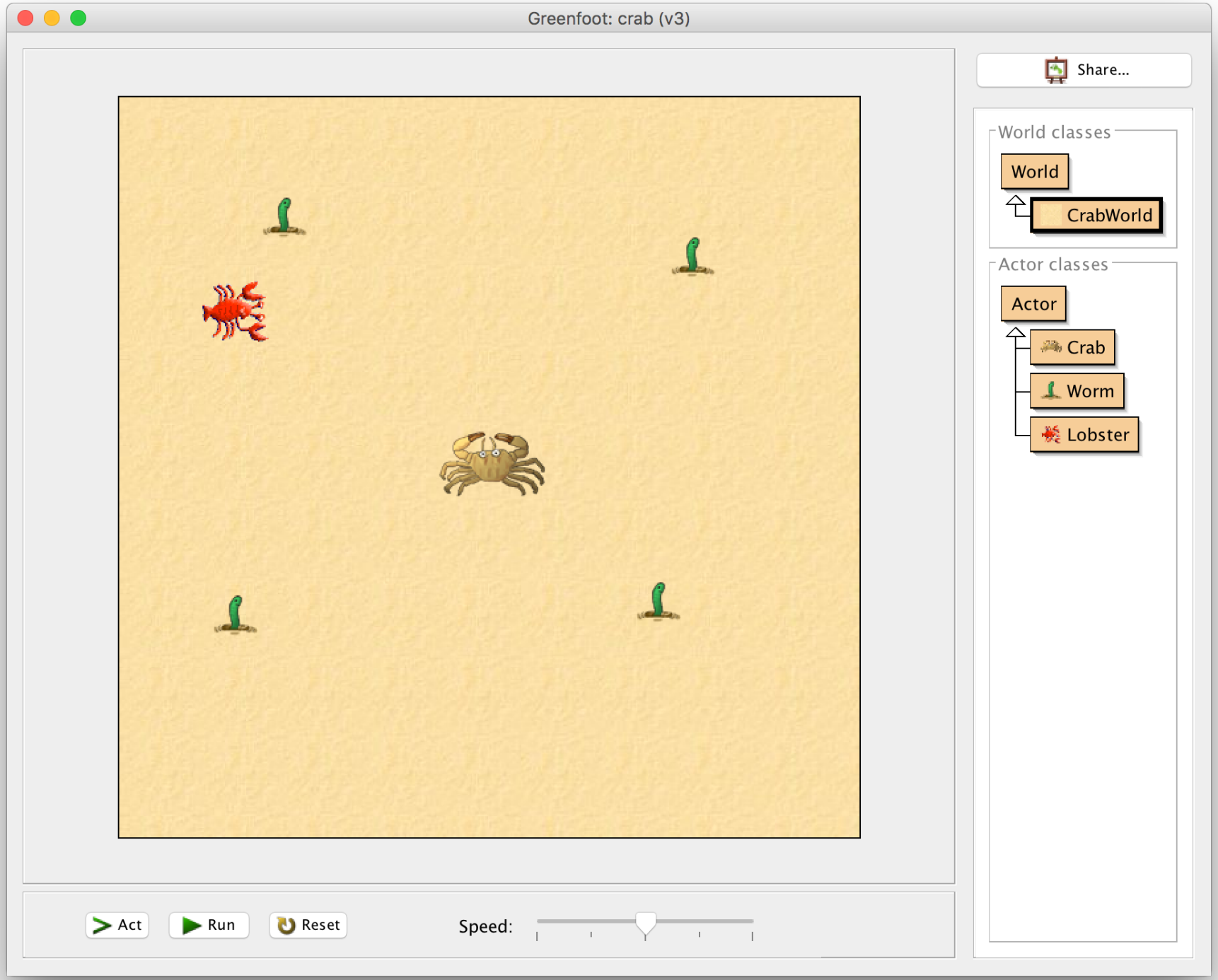


Better Lobster

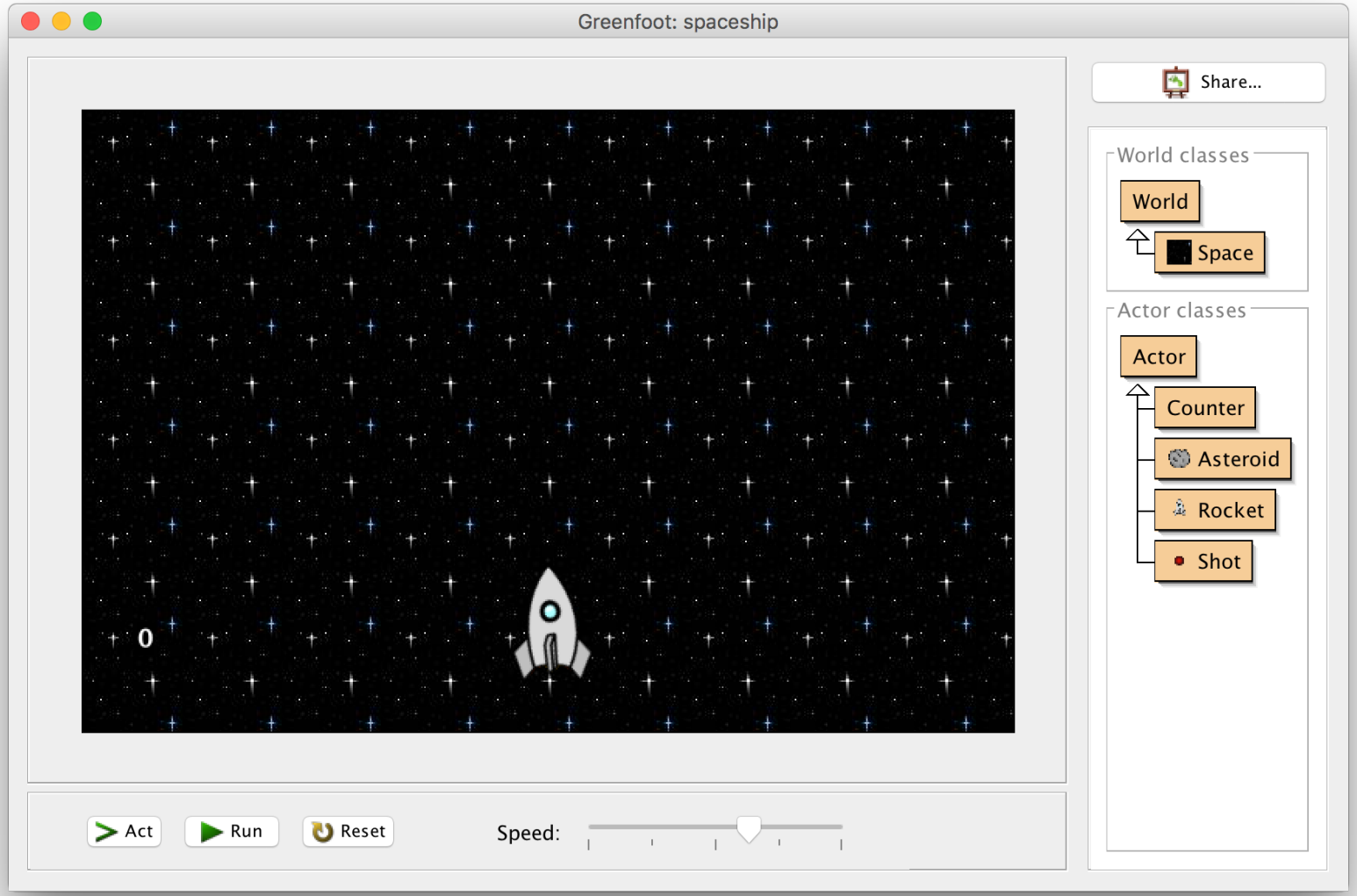


Off the Wall!


Lobster - crab (v3)
Class Edit Tools Options
Lobster
Compile Undo Cut Copy Paste Find... Close
Source Code
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) ;
}
No changes need to be saved
saved


Greenfoot: crab (v3)
Share...
World classes
World
CrabWorld
Actor classes
Actor
Crab
Worm
Lobster
Act
Run
Reset
Speed:

Part 6 - Space Ships



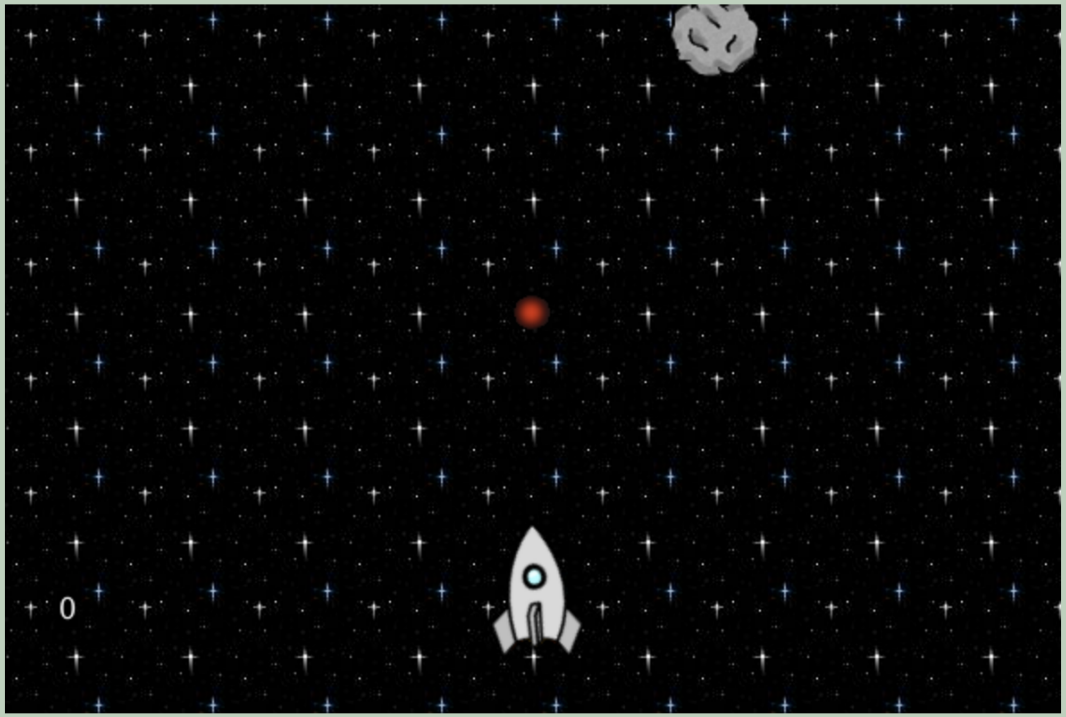
How to access one object from another

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

Search

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 Popu 1 2 3

little-crab 2-player
by MadsOkkels, 14 hours ago

RobTheBank
by Sauerbraten, yesterday

crazy game my Moritz

DonkeyKong with guns
by TheMatrixSheep, 2016/8/26

The Kong will have his Revenge

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).

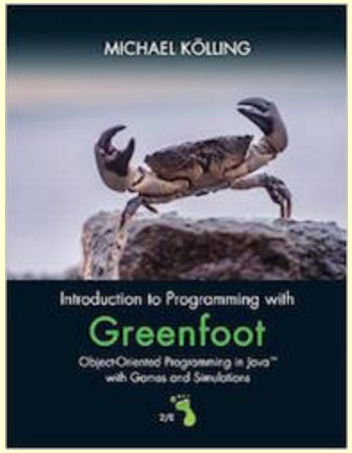
Going Further

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



Michael Kölling

Introduction to Programming with Greenfoot

Object-Oriented Programming in Java with Games and Simulations

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 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)			
Scenarios discussed in the	Discussion forum	Introduction to	Introduction to	
	Scenario gallery			