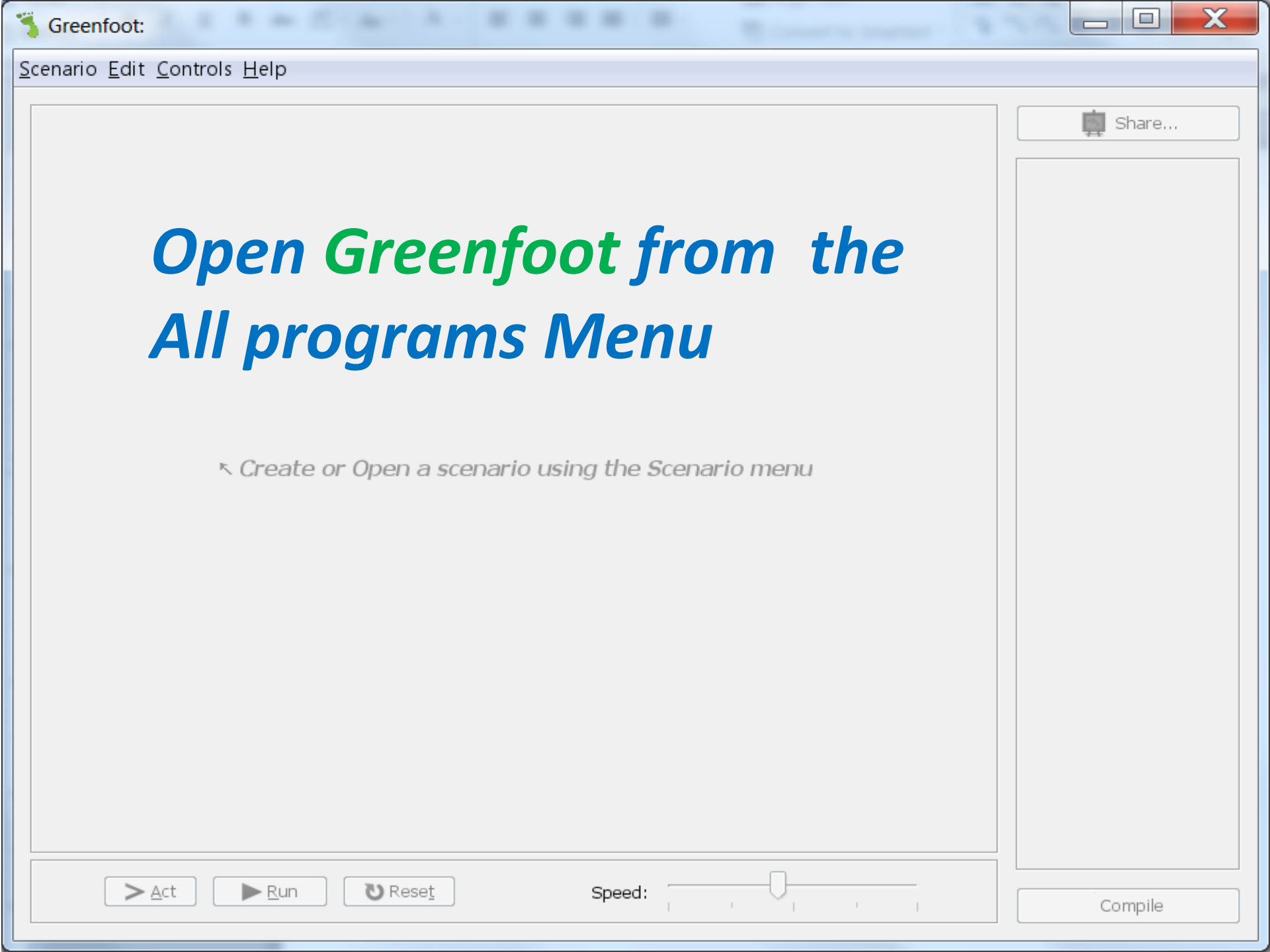
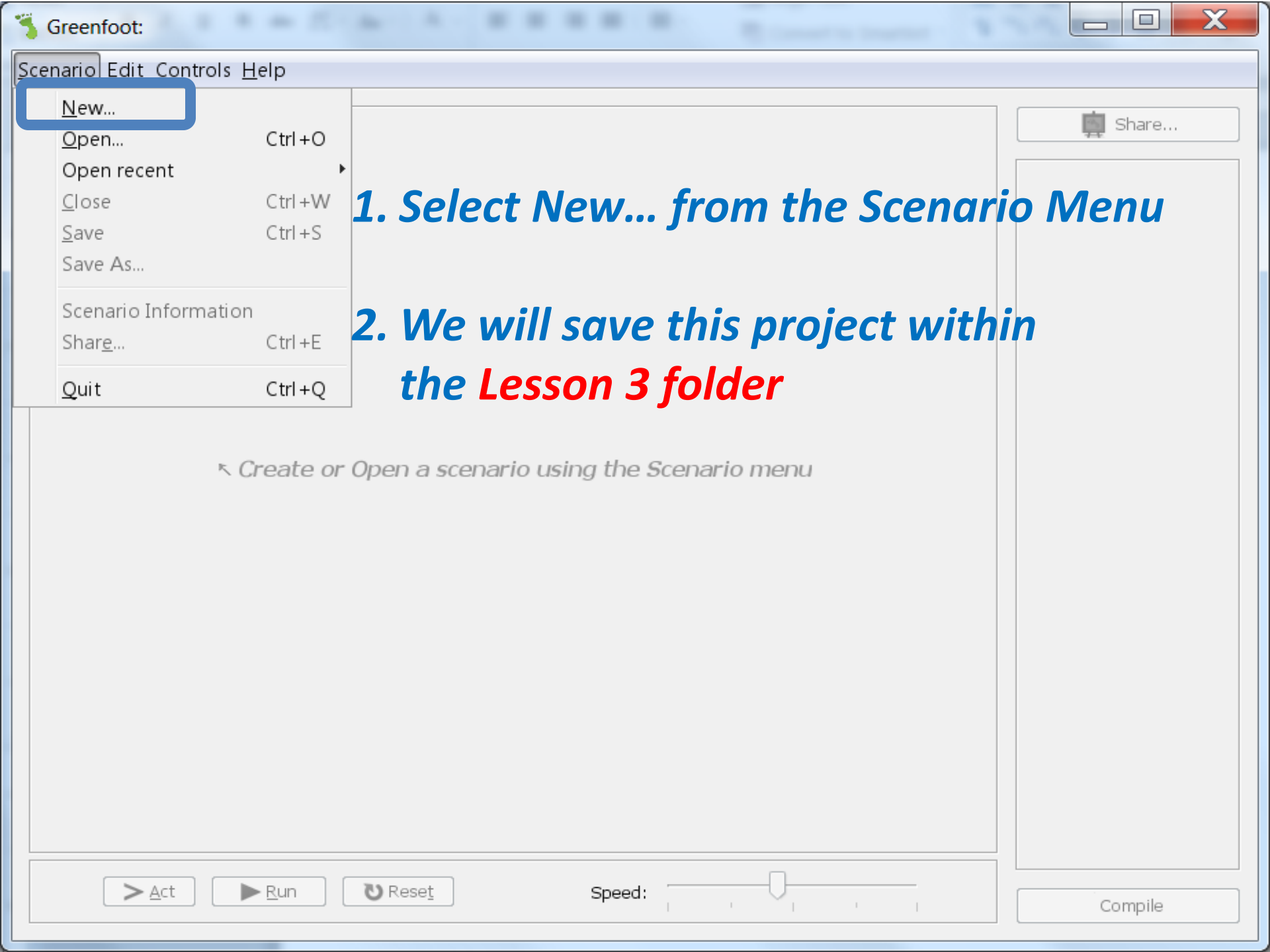
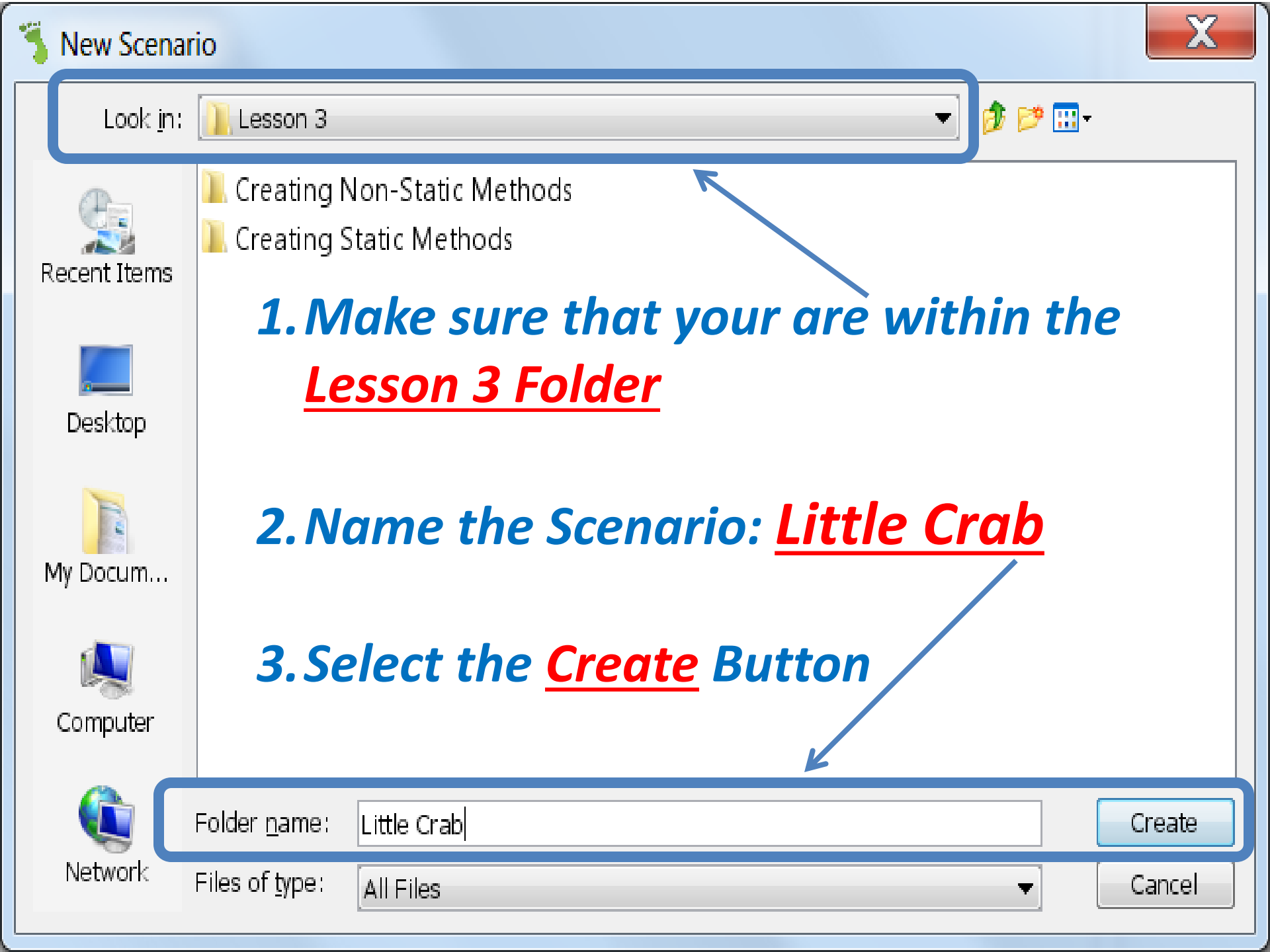
The Crab Game

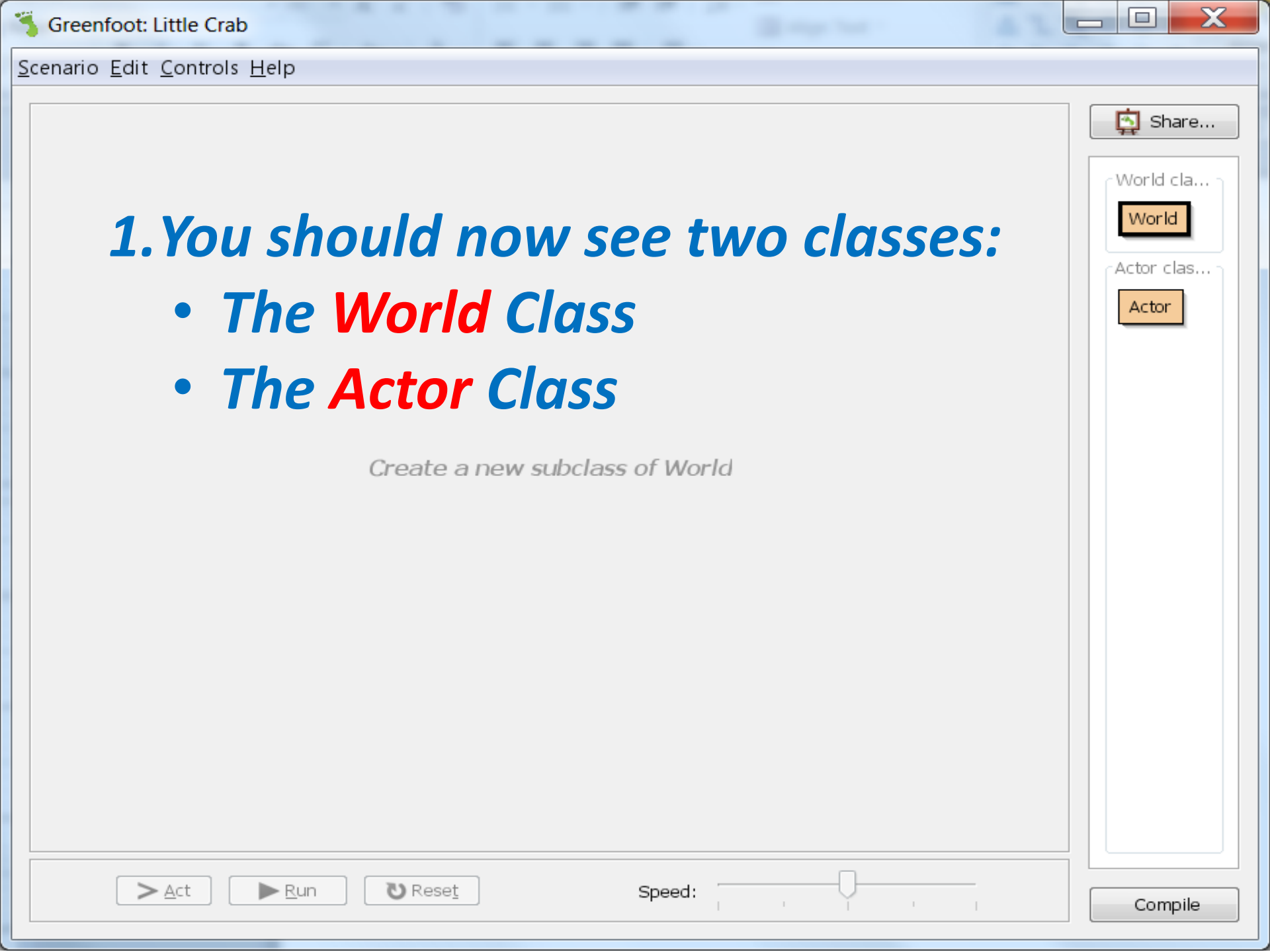
Digital Game Development

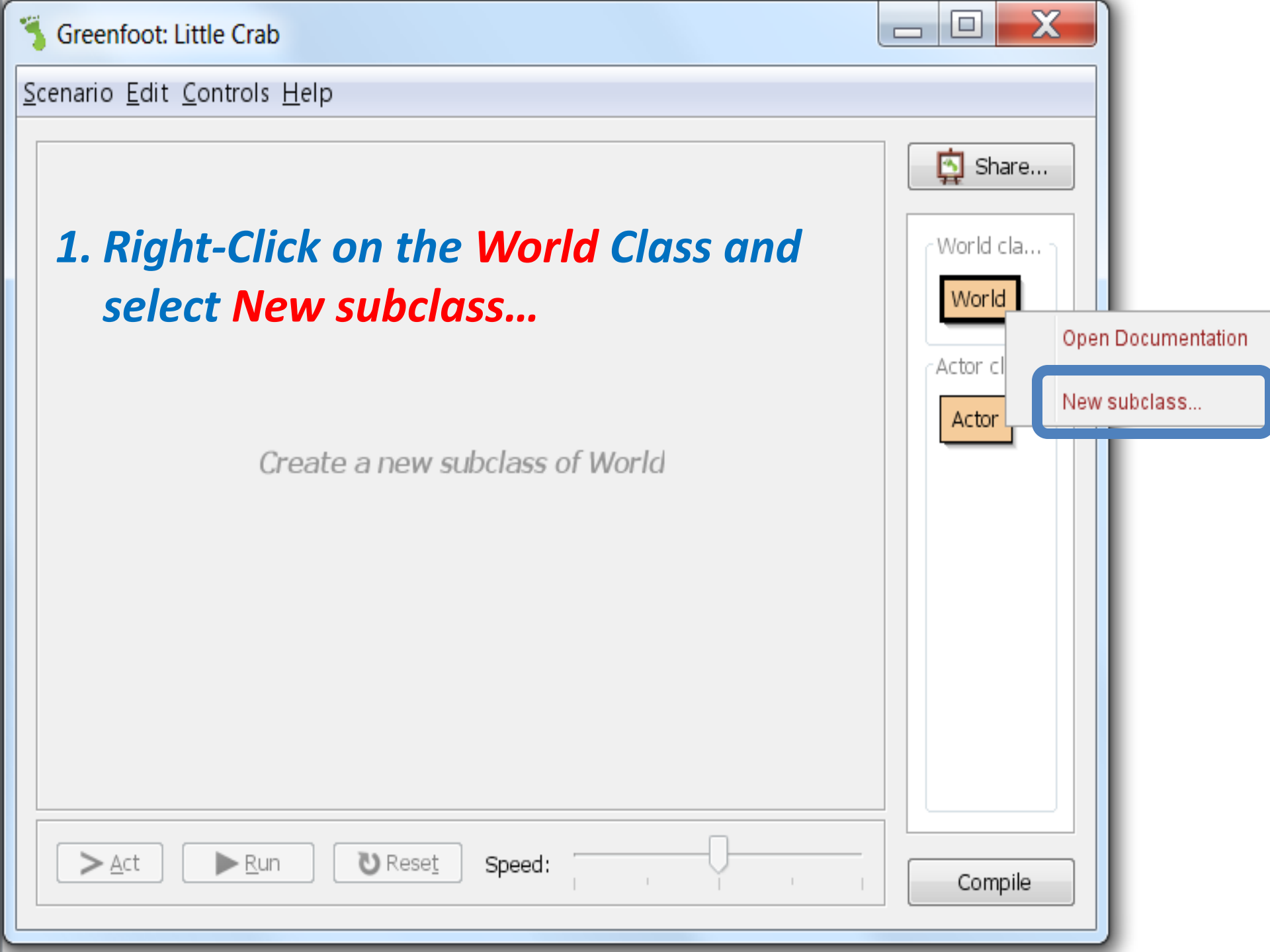
2015-2016

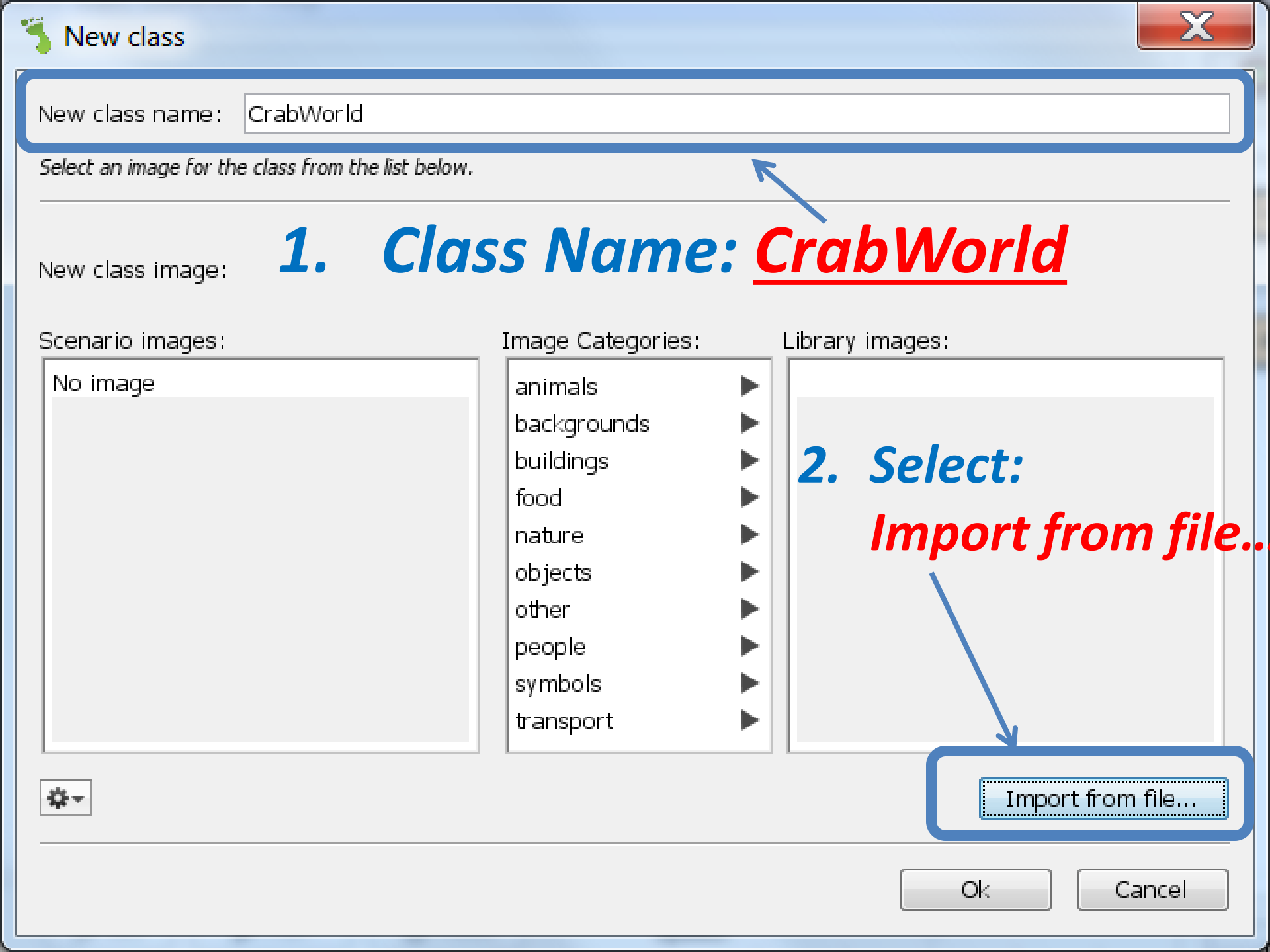

Greenfoot:
Scenario Edit Controls Help
Share...
Open Greenfoot from the All programs Menu
Create or Open a scenario using the Scenario menu
Act
Run
Reset
Speed:
Compile


Greenfoot:
Scenario Edit Controls Help
New...
Open... Ctrl+O
Open recent
Close Ctrl+W
Save Ctrl+S
Save As...
Scenario Information
Share... Ctrl+E
Quit Ctrl+Q
1. Select New… from the Scenario Menu
2. We will save this project within the Lesson 3 folder
Create or Open a scenario using the Scenario menu
Share...
Act
Run
Reset
Speed:
Compile

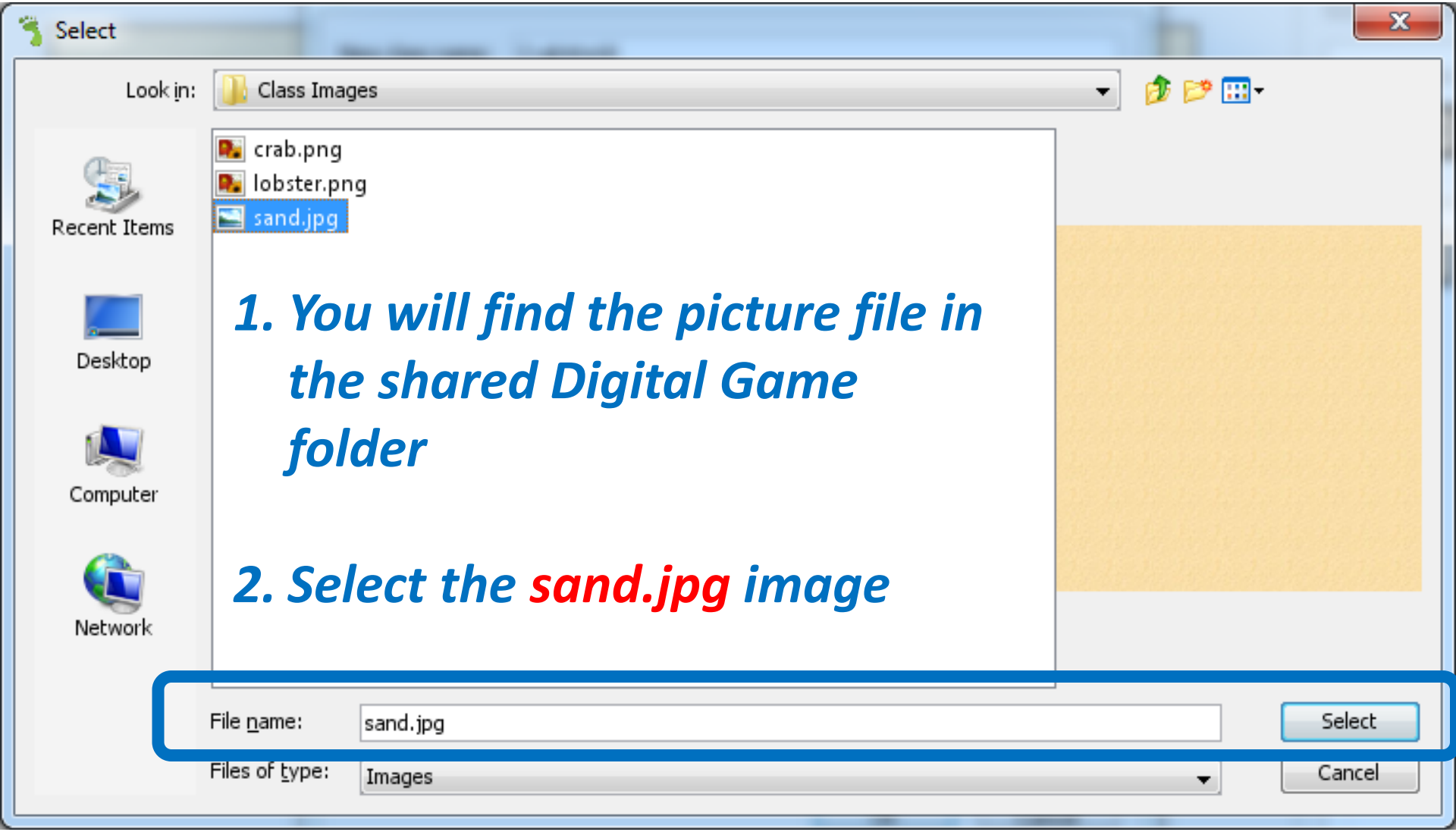

New Scenario
Look in:
Lesson 3
Creating Non-Static Methods
Creating Static Methods
Recent Items
Desktop
My Docum...
Computer
Network
1. Make sure that your are within the Lesson 3 Folder
2. Name the Scenario: Little Crab
3. Select the Create Button
Folder name:
Little Crab
Create
Files of type:
All Files
Cancel

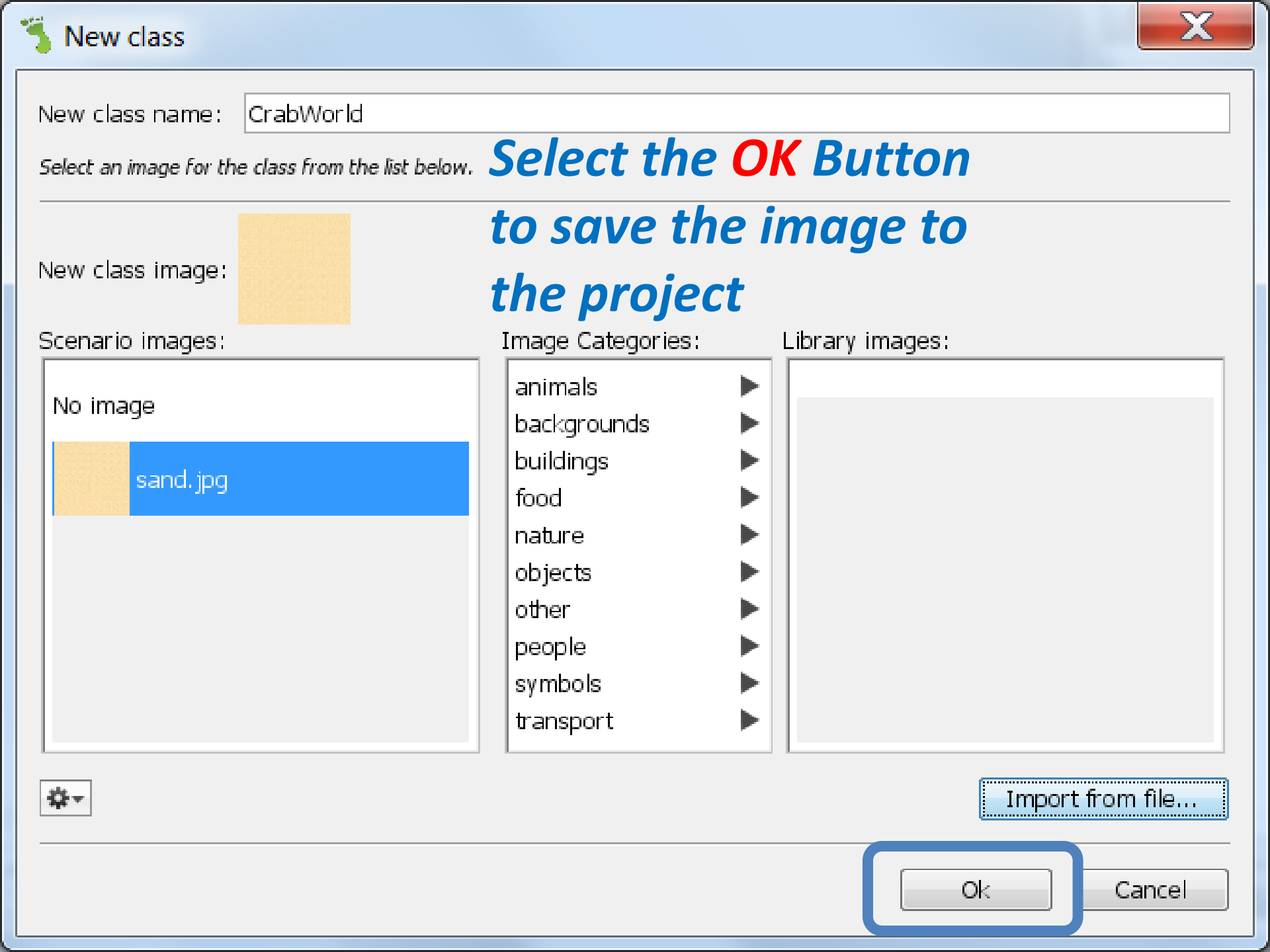

Greenfoot: Little Crab
Scenario Edit Controls Help
1.You should now see two classes:
• The World Class
• The Actor Class
Create a new subclass of World
Share...
World cla...
World
Actor clas...
Actor
Act
Run
Reset
Speed:
Compile

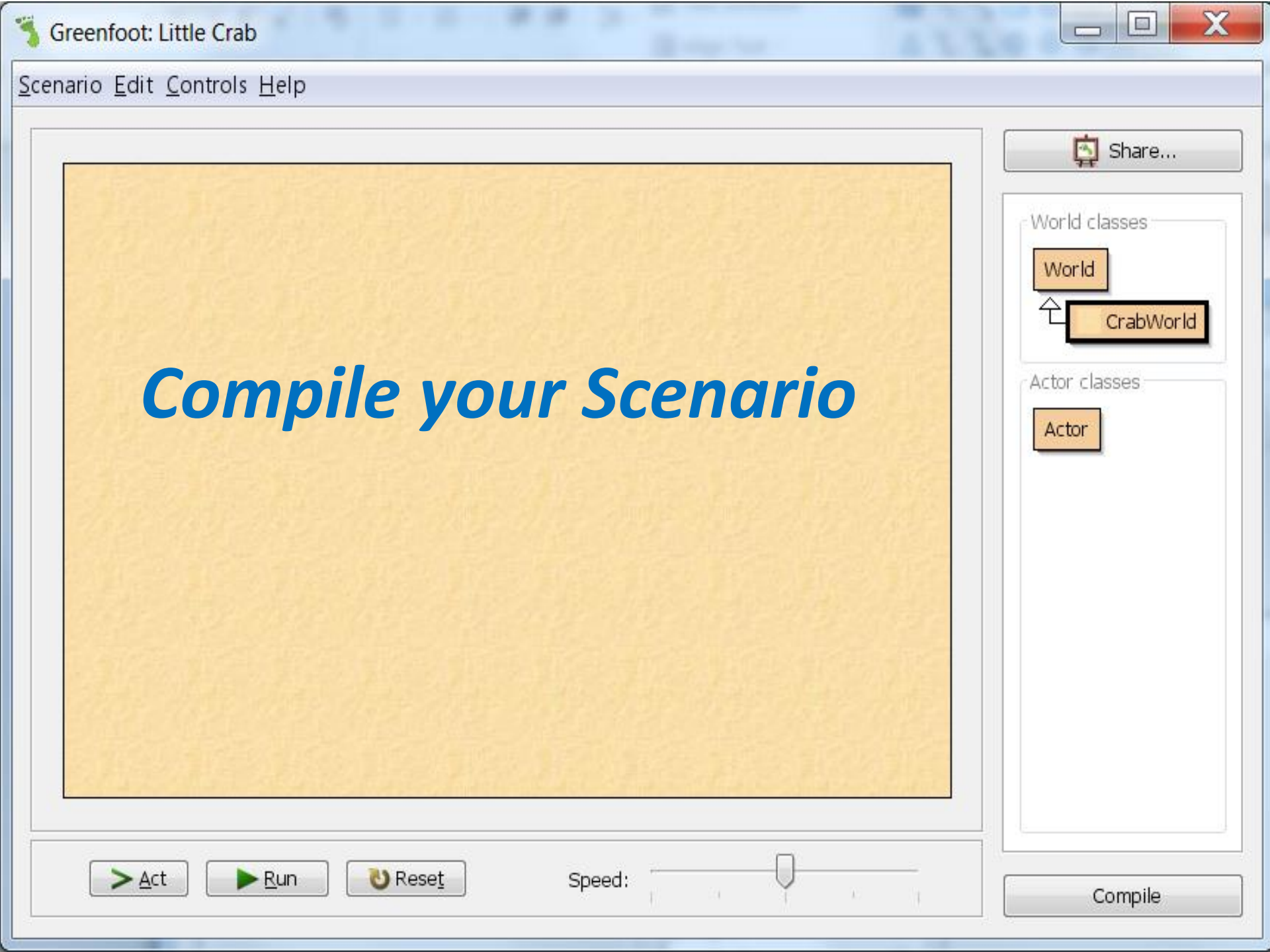

Greenfoot: Little Crab
Scenario Edit Controls Help
1. Right-Click on the World Class and select New subclass…
Create a new subclass of World
Share...
World cla...
World
Actor cl
Actor
Open Documentation
New subclass...
Act
Run
Reset
Speed:
Compile


New class
New class name:
CrabWorld
Select an image for the class from the list below.
New class image:
1. Class Name: CrabWorld
Scenario images:
No image
Image Categories:
animals
backgrounds
buildings
food
nature
objects
other
people
symbols
transport
Library images:
2. Select:
Import from file...
Import from file...
Ok
Cancel

Data:\Shared\Students\Digital Game\Class Images\sand.jpg




New class
New class name:
CrabWorld
Select an image for the class from the list below.
Select the OK Button to save the image to the project
New class image:
Scenario images:
Image Categories:
Library images:
No image
sand.jpg
animals
backgrounds
buildings
food
nature
objects
other
people
symbols
transport
Import from file...
Ok
Cancel


Greenfoot: Little Crab
Scenario Edit Controls Help
Compile your Scenario
Share...
World classes
World
CrabWorld
Actor classes
Actor
Act
Run
Reset
Speed:
Compile

Class Structure

```
public class World
{
    Inner part omitted.
}
```

The outer wrapper of World

Class Structure

```
public class World
{
    Inner part omitted.
}
```

The outer wrapper of World

```
public class CrabWorld
{
    Fields
    Constructors
    Methods
}
```

The inner contents of a class

Inheritance

- ***Inheritance*** allows us to define one class as an extension of another
  - We saw this with the Game Board

```
public class CountingSeconds extends DrawableAdapter
{
```

Class Structure with Inheritance

```
public class World
{
    Inner part omitted.
}

public class CrabWorld extends World
{
    Fields
    Constructors
    Methods
}
```

The outer wrapper of World

The inner contents of a class

Class ***Fields***

1. ***Fields*** store values for an object
2. They are also known as ***instance variables***
3. ***Fields*** define the state of an object

Class ***Constructors***

1. ***Constructors*** initialize an object

2. ***Constructors*** have the same name as their class

3. ***Constructors*** store initial values into the classes fields
   - External parameters are used for this

Class ***Methods***

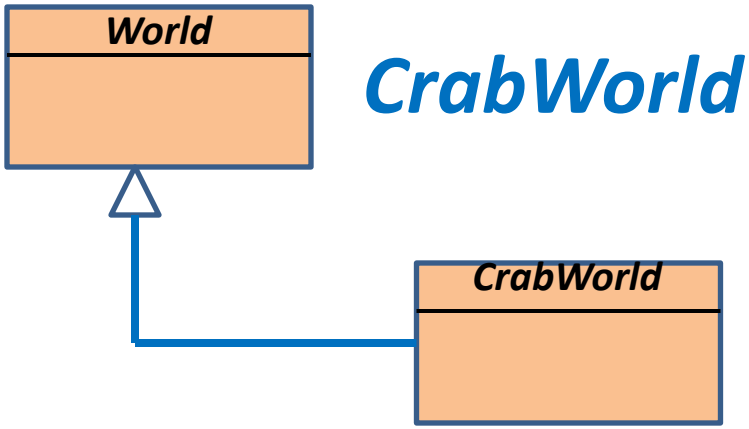
1. ***Methods*** implement the behavior of objects

2. **Accessor methods** provide information about an object

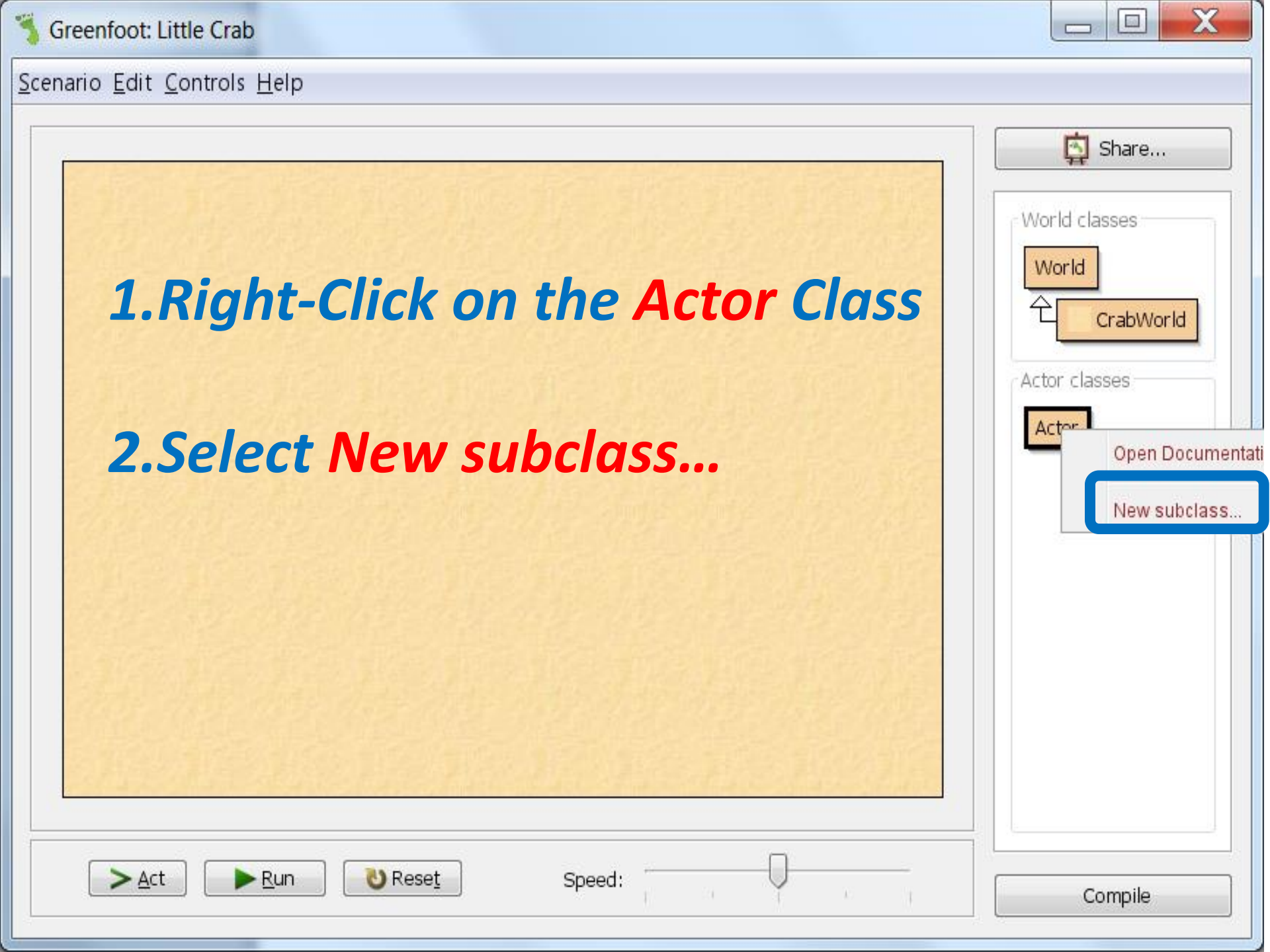
3. ***Mutator methods*** alter the state of an object

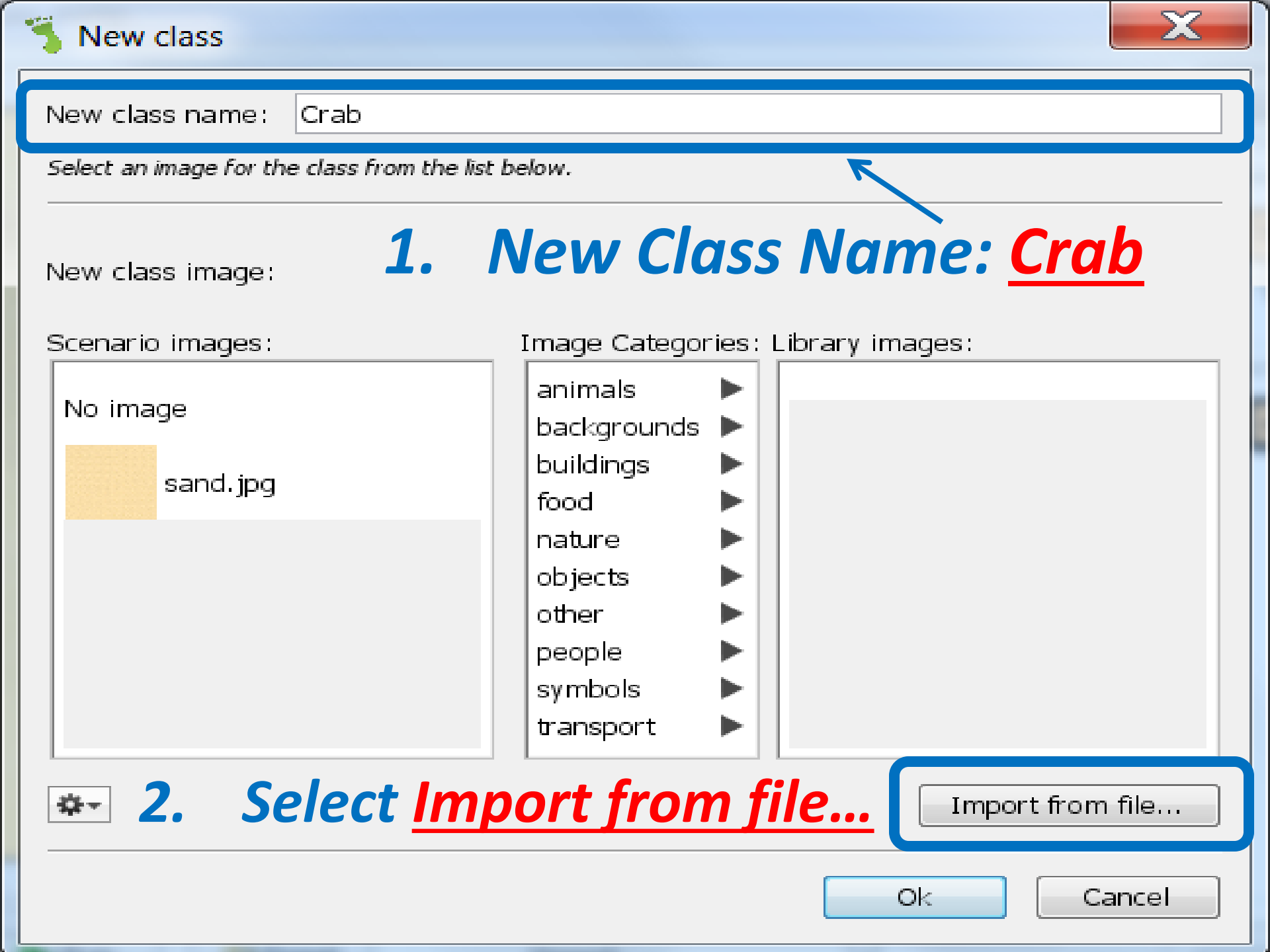
Inheritance

- ***Inheritance*** creates a hierarchy of classes
  1. ***SuperClass*** or ***Parent Class***
     - A class that is extended by another class
  2. ***SubClass*** or ***Child Class***
     - A class that extends or inherits from another class
     - It inherits all fields and methods from its superclass

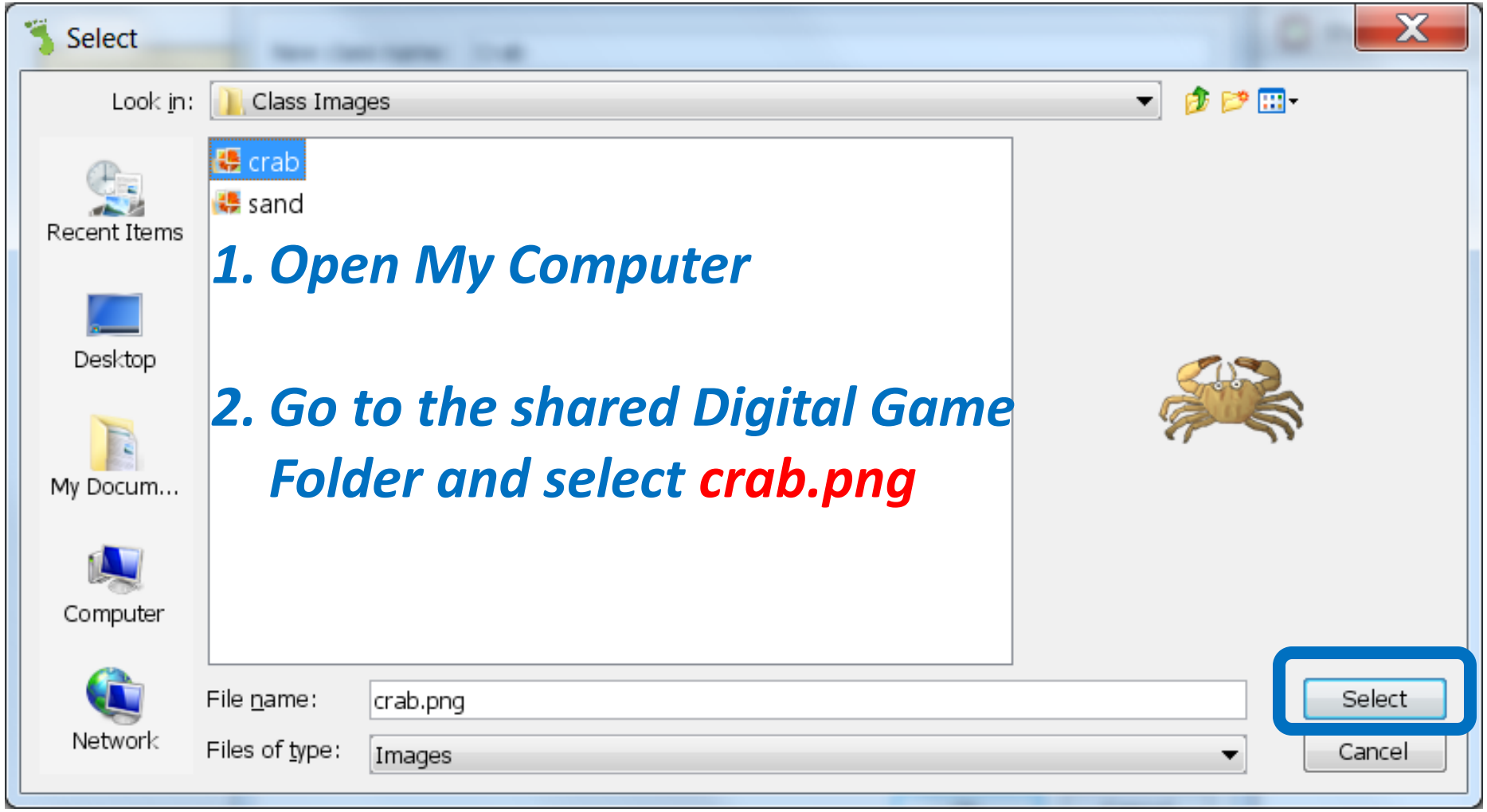


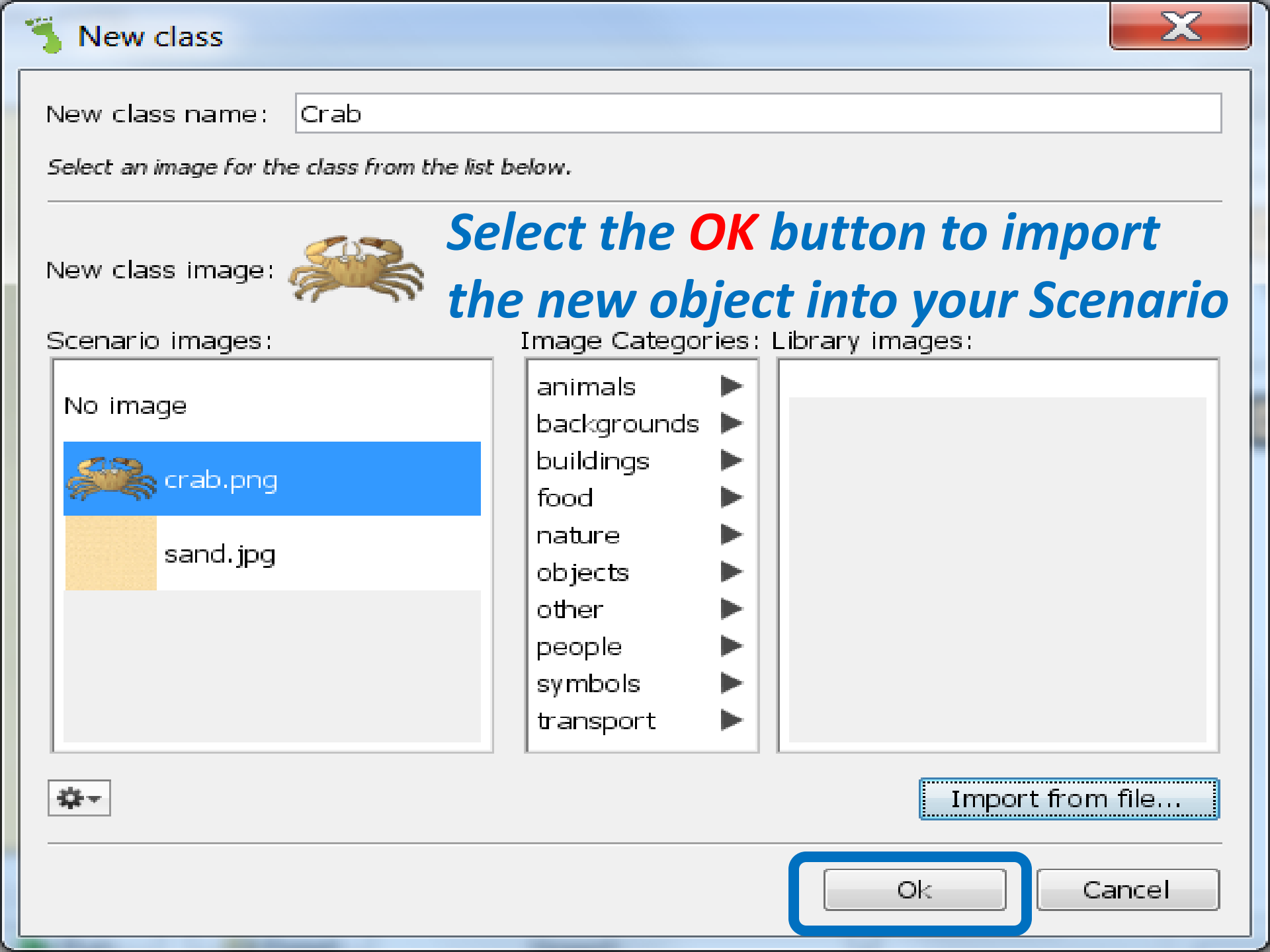
CrabWorld "is a" World

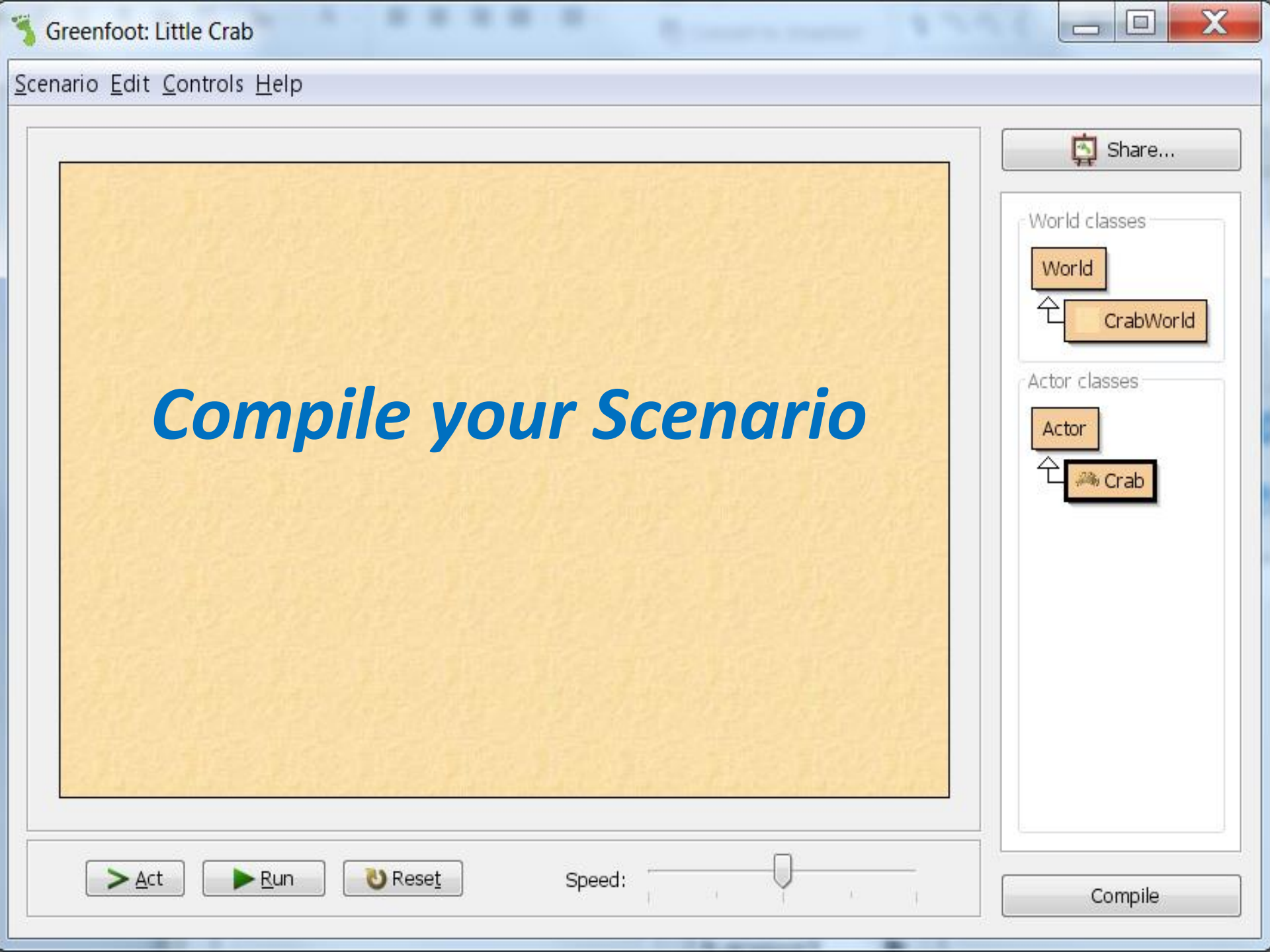

Greenfoot: Little Crab
Scenario Edit Controls Help
1.Right-Click on the Actor Class
2.Select New subclass…
Share...
World classes
World
CrabWorld
Actor classes
Open Documentati
New subclass...
Act
Run
Reset
Speed:
Compile


New class
New class name:
Crab
Select an image for the class from the list below.
1. New Class Name: Crab
New class image:
Scenario images:
No image
sand.jpg
Image Categories:
animals
backgrounds
buildings
food
nature
objects
other
people
symbols
transport
Library images:
2. Select Import from file...
Import from file...
Ok
Cancel

Data:\Shared\Students\Digital Game\Class Images\crab.png

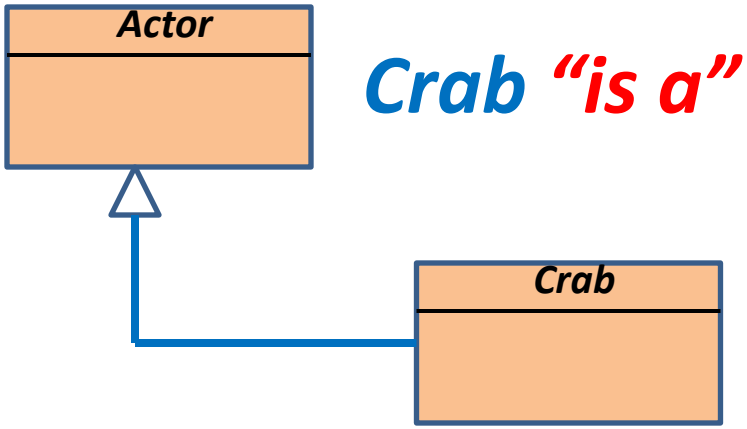


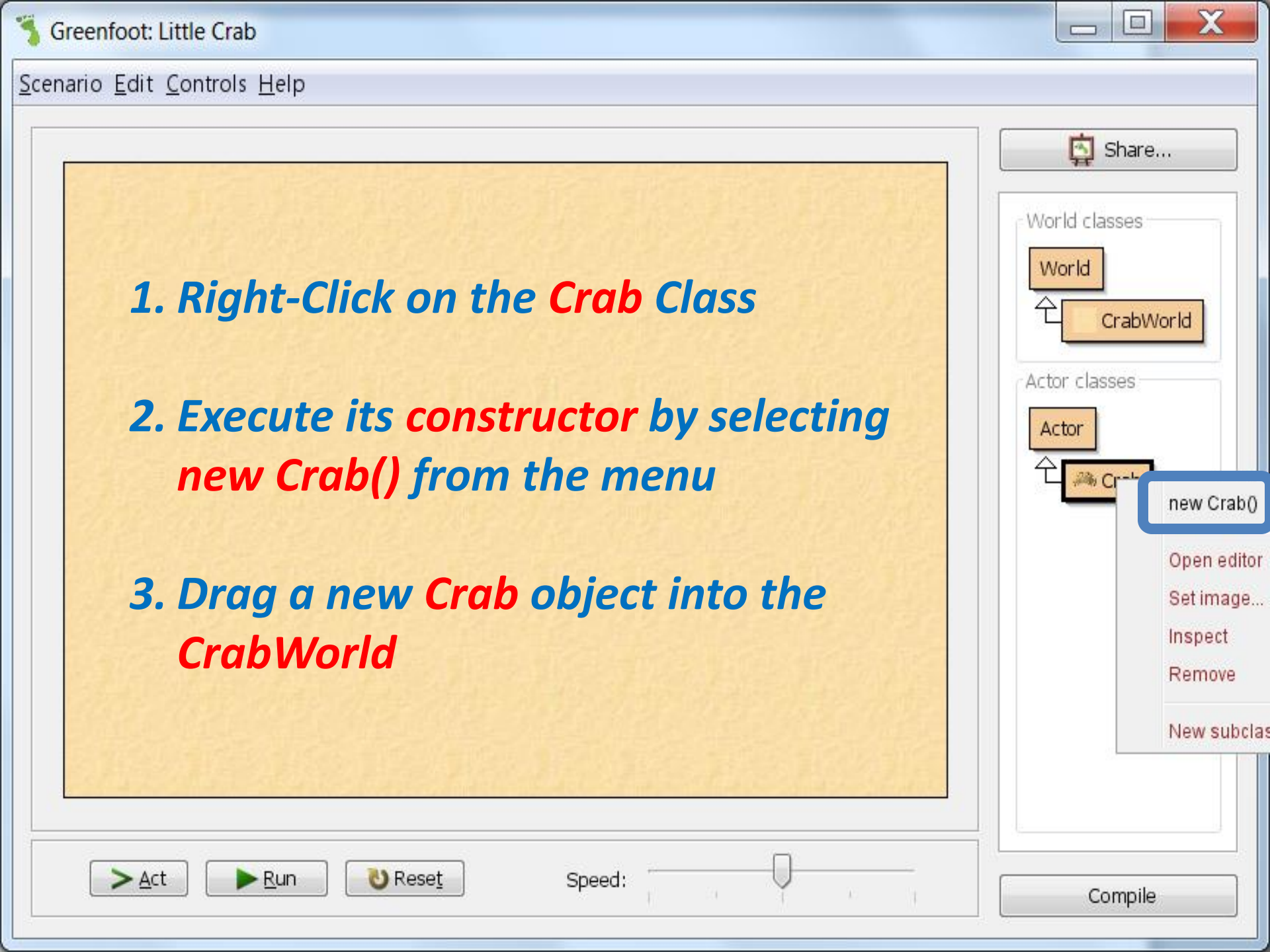

New class
New class name:
Crab
Select an image for the class from the list below.
New class image:
Select the OK button to import the new object into your Scenario
Scenario images:
Image Categories:
Library images:
No image
crab.png
sand.jpg
animals
backgrounds
buildings
food
nature
objects
other
people
symbols
transport
Import from file...
Ok
Cancel


Greenfoot: Little Crab
Scenario Edit Controls Help
Compile your Scenario
Share...
World classes
World
CrabWorld
Actor classes
Actor
Crab
Act
Run
Reset
Speed:
Compile

Inheritance

- ***Inheritance*** creates a hierarchy of classes
  1. ***SuperClass*** or ***Parent Class***
     - A class that is extended by another class
  2. ***SubClass*** or ***Child Class***
     - A class that extends or inherits from another class
     - It inherits all fields and methods from its superclass




Greenfoot: Little Crab
Scenario Edit Controls Help
1. Right-Click on the Crab Class
2. Execute its constructor by selecting new Crab() from the menu
3. Drag a new Crab object into the CrabWorld
Share...
World classes
World
CrabWorld
Actor classes
Actor
new Crab()
Open editor
Set image...
Inspect
Remove
New subclas
Act
Run
Reset
Speed:
Compile

Greenfoot: Little Crab

Scenario Edit Controls Help

1. Click on the Run button
2. Nothing happens

Act Run Reset Speed:

Share...

World classes

World

CrabWorld

Actor classes

Actor

Crab

Compile

Crab - Little Crab

Class Edit Tools Options

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

```
import greenfoot.*;

/**
 * Write a description of class Crab here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
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()
    {
        // Add your action code here.
    }
}
```

1. Open the editor for The Crab Class

When Run is Clicked the Crab does nothing

This is because there is no Source Code in the act Method for the Crab.

saved

Crab - Little Crab

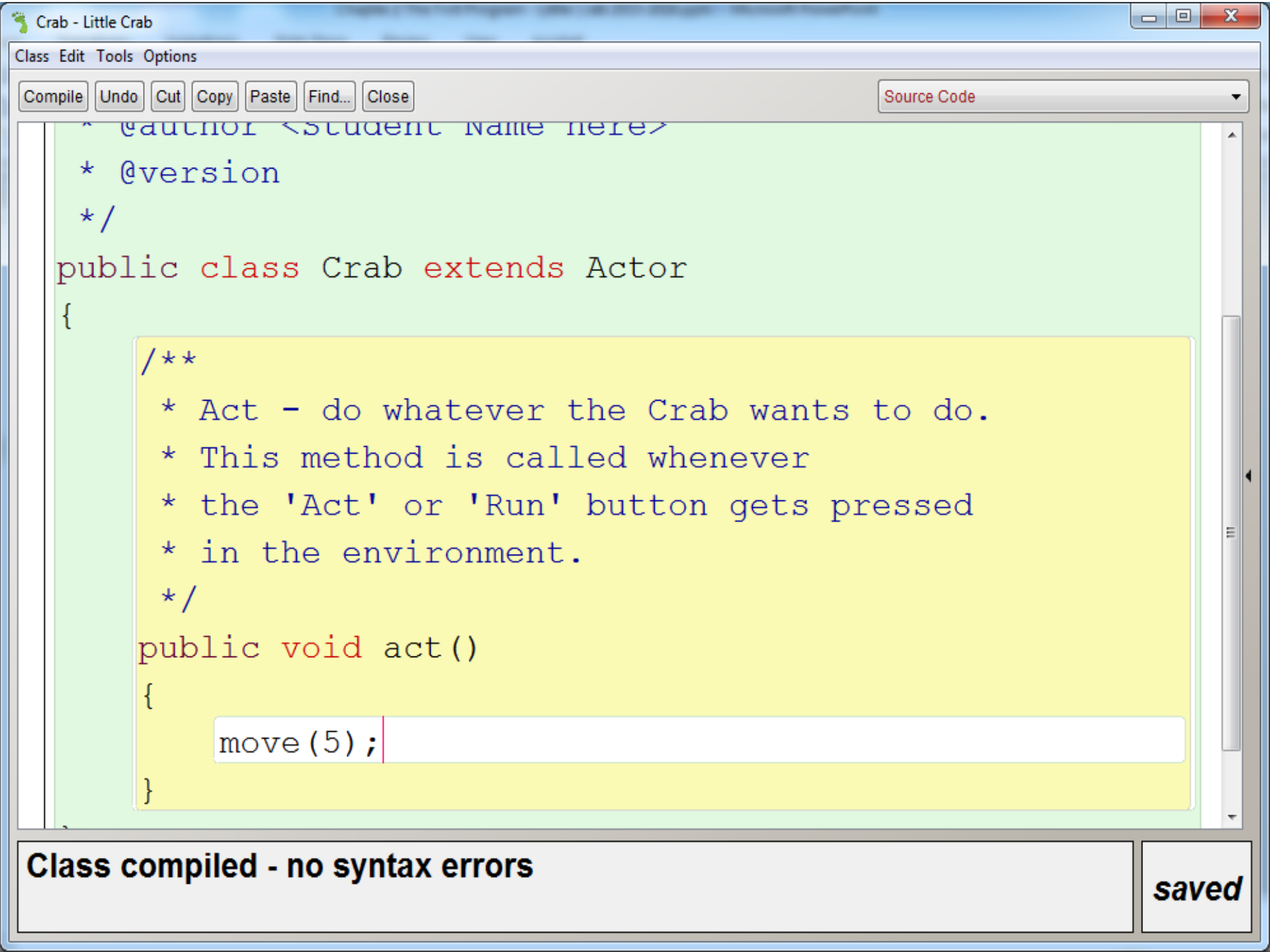
Class Edit Tools Options

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

```
import greenfoot.*;

/**
 * This class defines a crab. Crabs live on the beach.
 *
 * @author <Student Name here>
 * @version
 */
public class Crab extends Actor
{
    /**
     * Act - do whatever the Crab wants to do. This method
     * the 'Act' or 'Run' button gets pressed in the enviro
     */
```

Class compiled - no syntax errors

saved


Crab - Little Crab
Class Edit Tools Options
Compile Undo Cut Copy Paste Find... Close
Source Code
* @author <Student Name here>
* @version
*/
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()
{
move(5);
}
Class compiled - no syntax errors
saved

Greenfoot: Little Crab

Scenario Edit Controls Help

1. Click on the Act button

2. Click on the Run button

Act Run Reset Speed:

Share...

World classes

World

CrabWorld

Actor classes

Actor

Crab

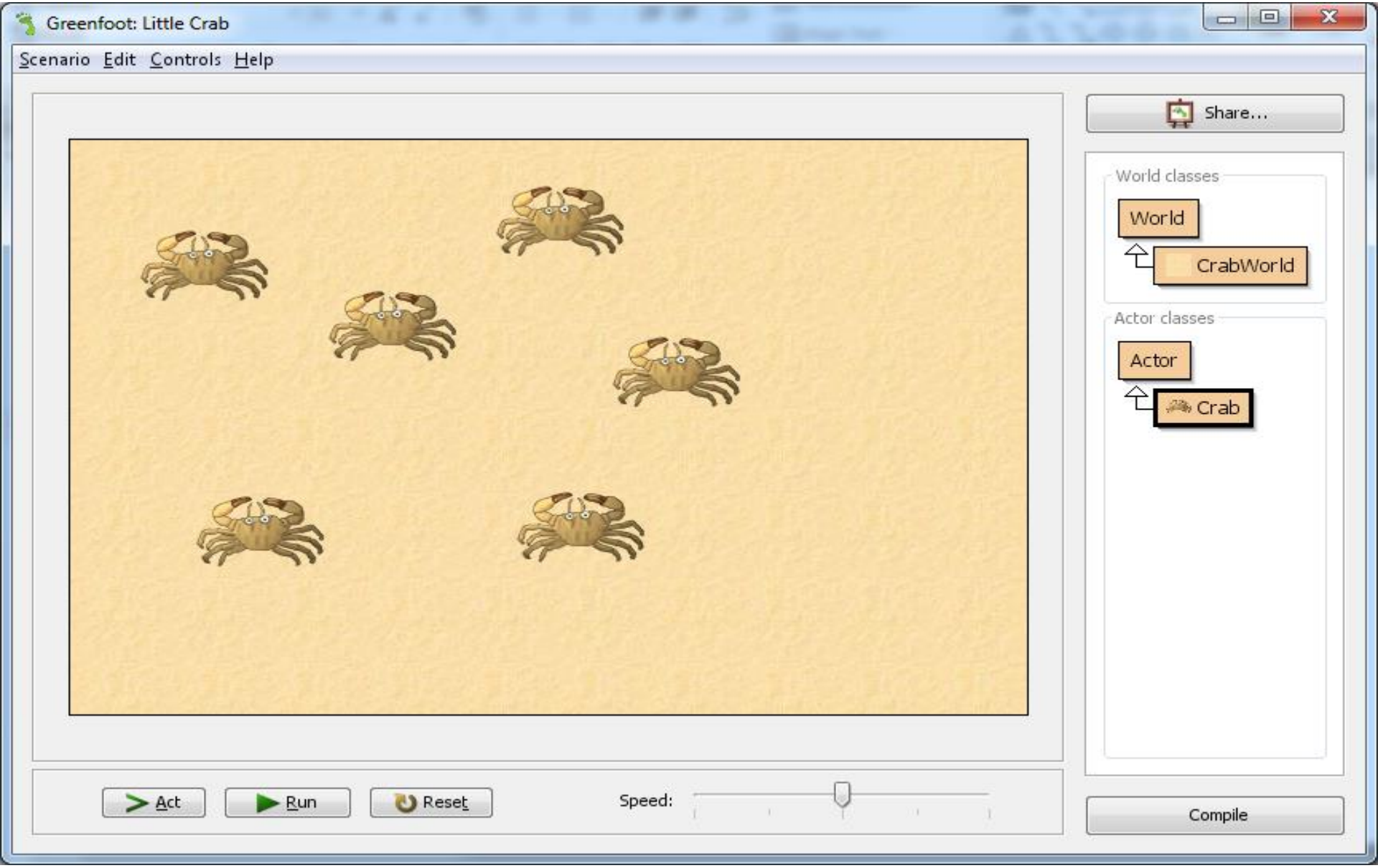
Compile

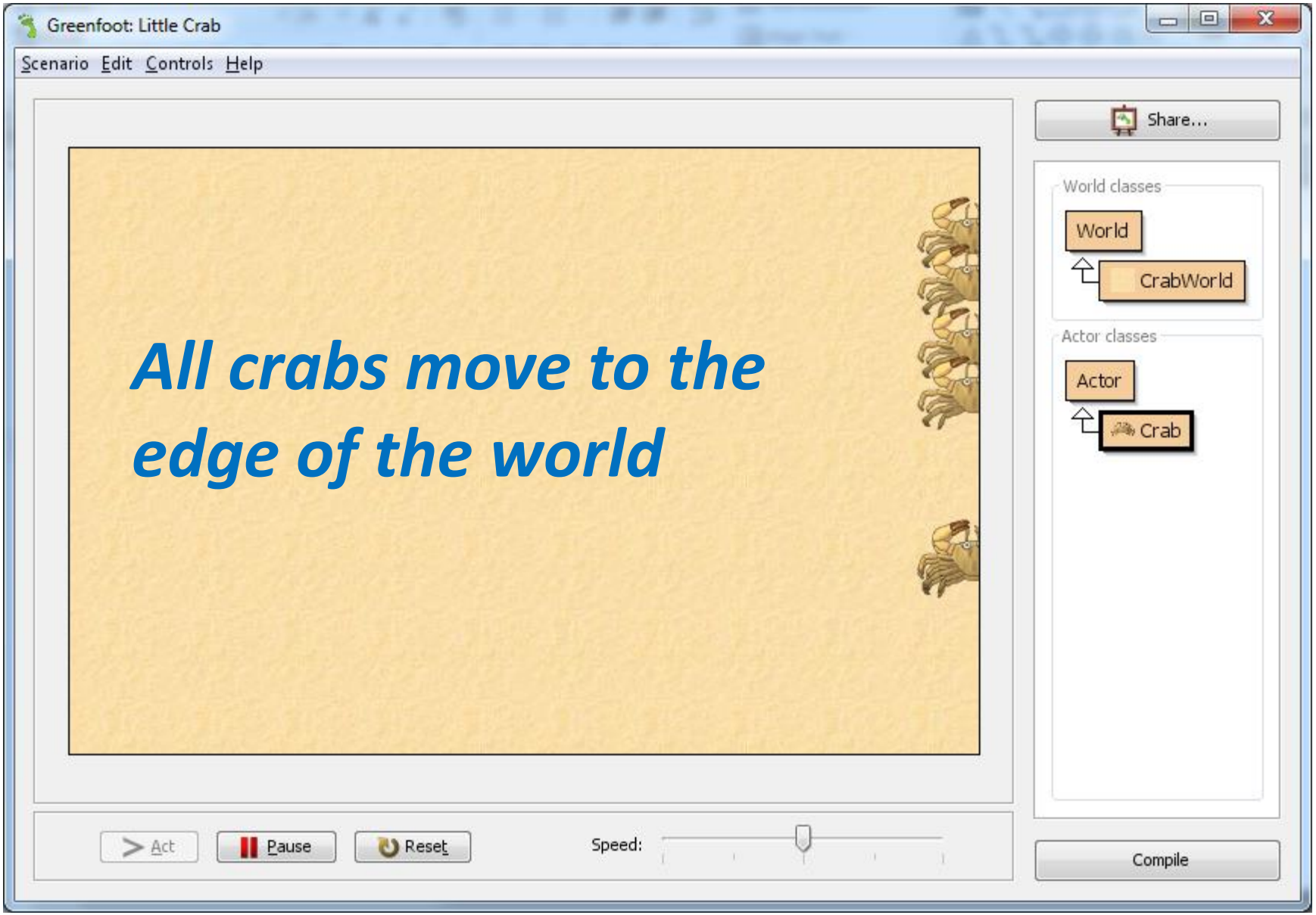
The Act and Run Buttons

- Clicking the ***Act*** button executes the code in the act() method once

- Clicking the ***Run*** button is just lick clicking the ***Act*** button repeatedly

1. *Place multiple crabs into the World. Run the scenario*
2. *Change the number in the move method*




Greenfoot: Little Crab
Scenario Edit Controls Help
All crabs move to the edge of the world
Share...
World classes
World
CrabWorld
Actor classes
Actor
Crab
Act
Pause
Reset
Speed:
Compile

Turning

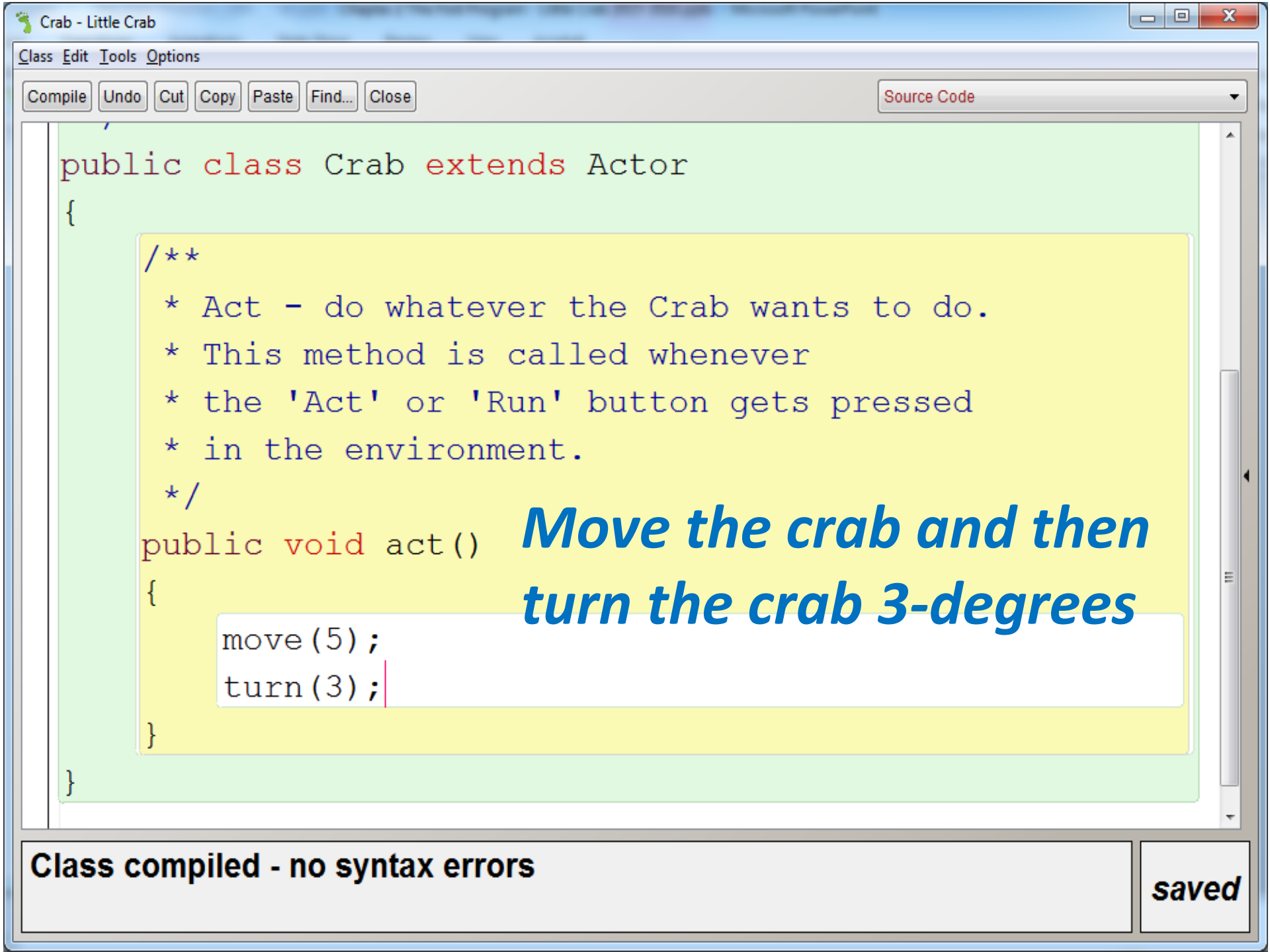
turn

```
public void turn(int amount)
```

Turn this actor by the specified amount (in degrees).

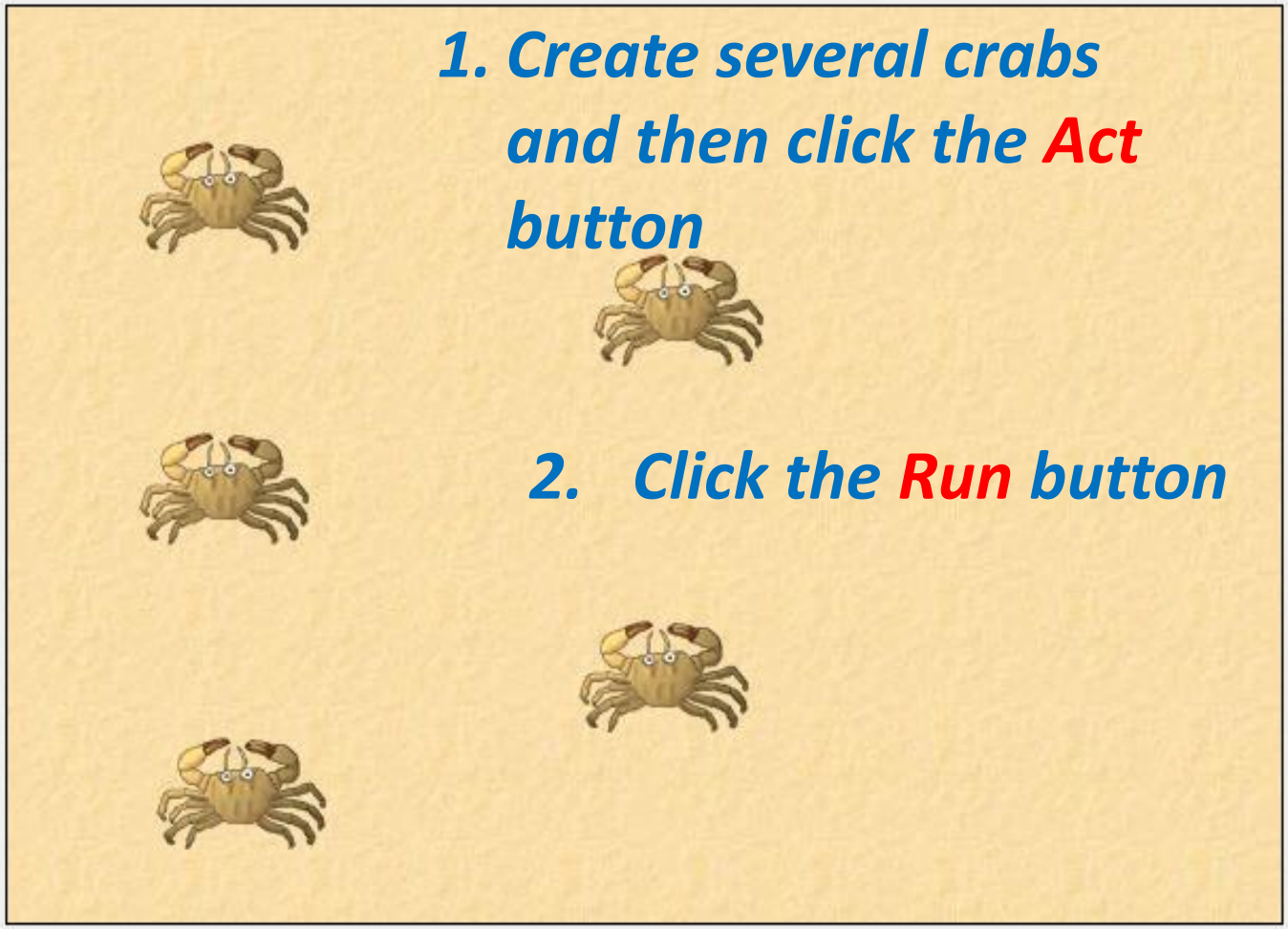
Parameters:

`amount` - the number of degrees to turn; positive values turn clockwise


Crab - Little Crab
Class Edit Tools Options
Compile Undo Cut Copy Paste Find... Close
Source Code
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()
{
move(5);
turn(3);
}
}
Move the crab and then turn the crab 3-degrees
Class compiled - no syntax errors
saved

Greenfoot: Little Crab

Scenario Edit Controls Help



Act Run Reset Speed:

Share...

World classes

World

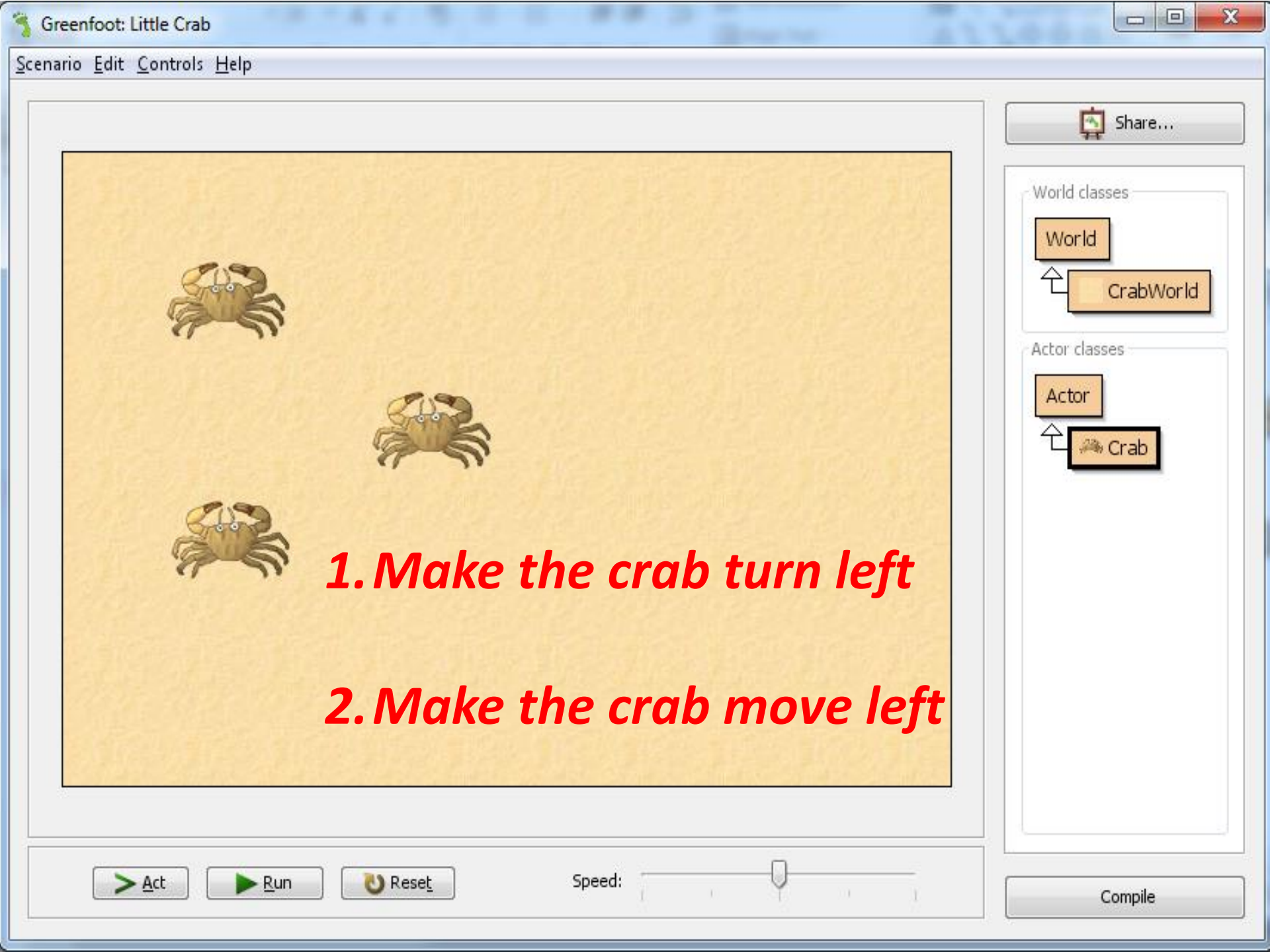
CrabWorld

Actor classes

Actor

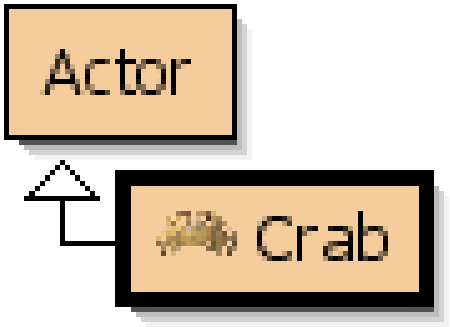
Crab

Compile


Greenfoot: Little Crab
Scenario Edit Controls Help
Share...
World classes
World
CrabWorld
Actor classes
Actor
Crab
1. Make the crab turn left
2. Make the crab move left
Act
Run
Reset
Speed:
Compile

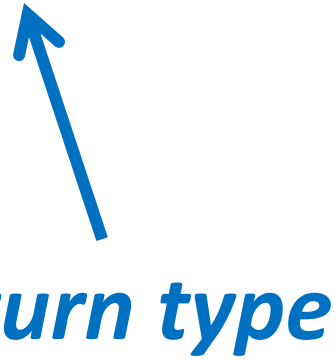
Inheritance

- Inheritance
  - The methods turn() and move() come from the Actor class
  - A crab is an Actor
    - Therefore, a crab can do whatever an Actor can do



Dealing with Screen Edges

- ***isAtEdge***
  - public boolean isAtEdge()



Dealing with Screen Edges

- ***isAtEdge***
  - public boolean isAtEdge()

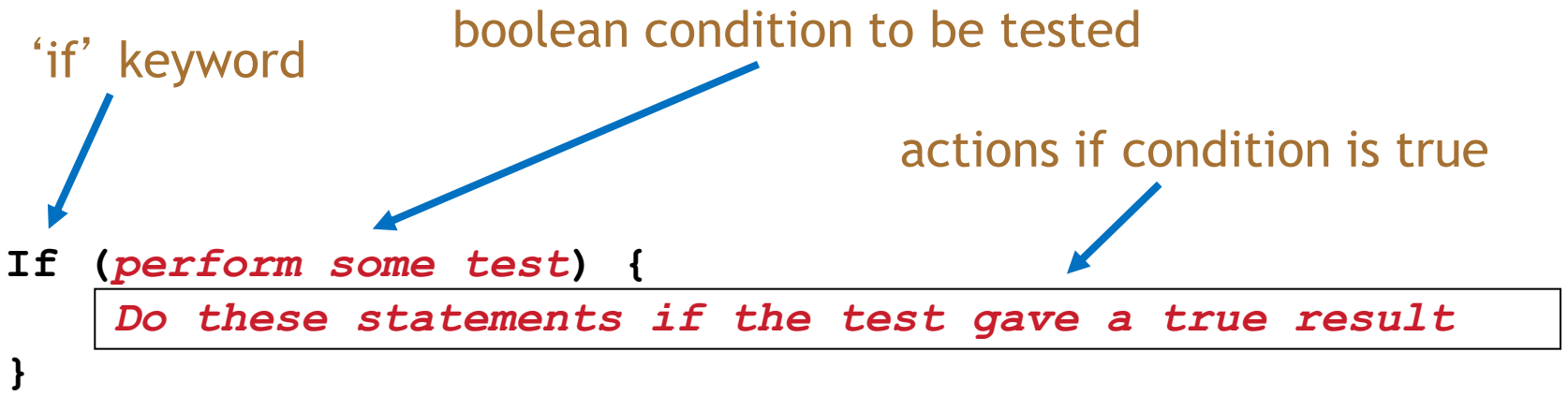
    1. Detect whether the actor has reached the edge of the world
    2. The actor is at the edge of the world if their position is at, or beyond, the cells at the very edge of the world

  - ***Returns***
    - ***True*** if the actor is at or beyond the edge cell of the world, and ***false*** otherwise

Making a choice in everyday life

```
if(I have enough money left) {
     go out for a meal;
}
else {
     stay home and watch a movie;
}
```

Making choices in Java



Making choices in a Program

- If (an object is at the edge of the world)
  - Tell the object to turn
  - Tell the object to move

Crab - Little Crab

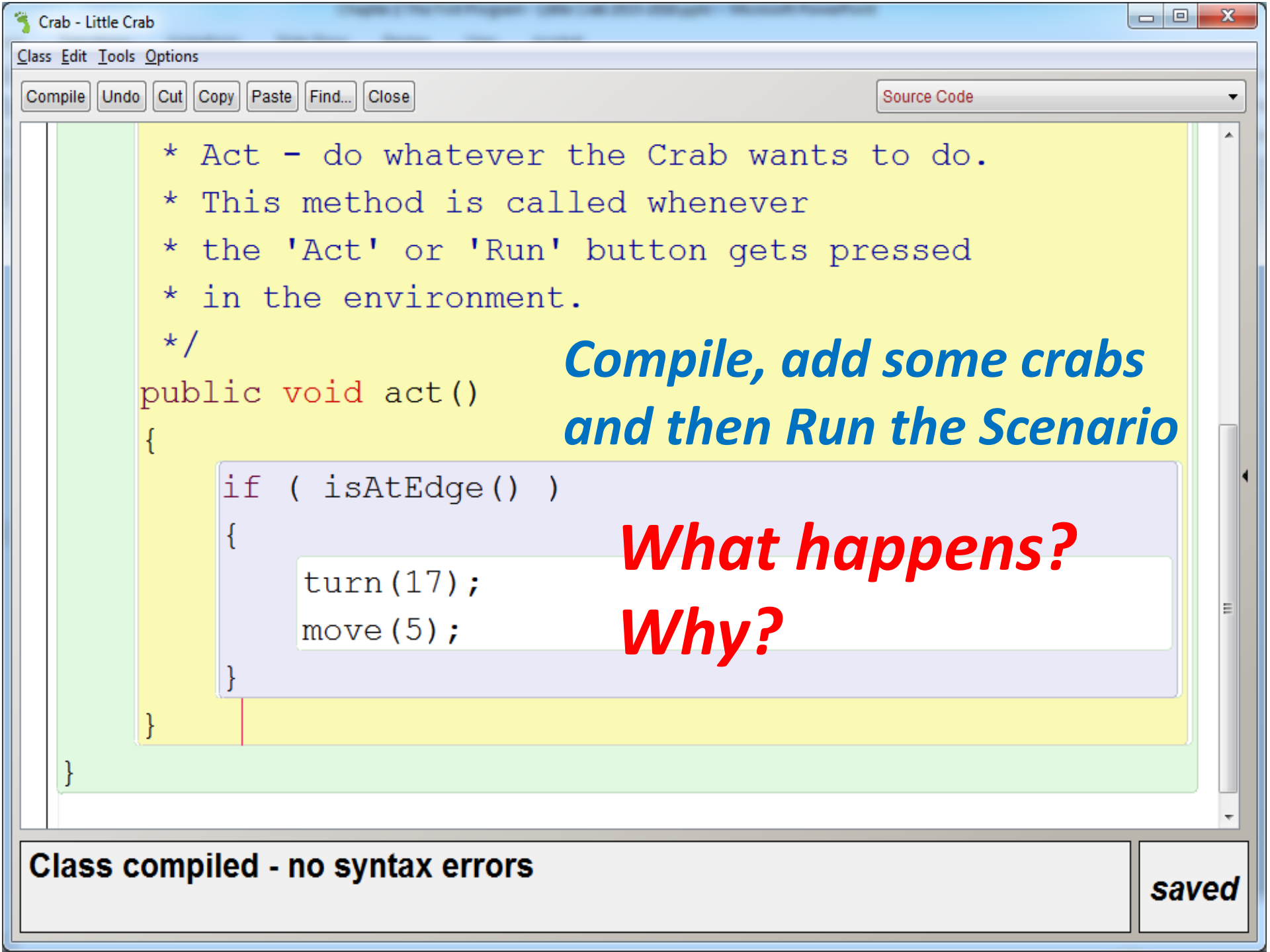
Class Edit Tools Options

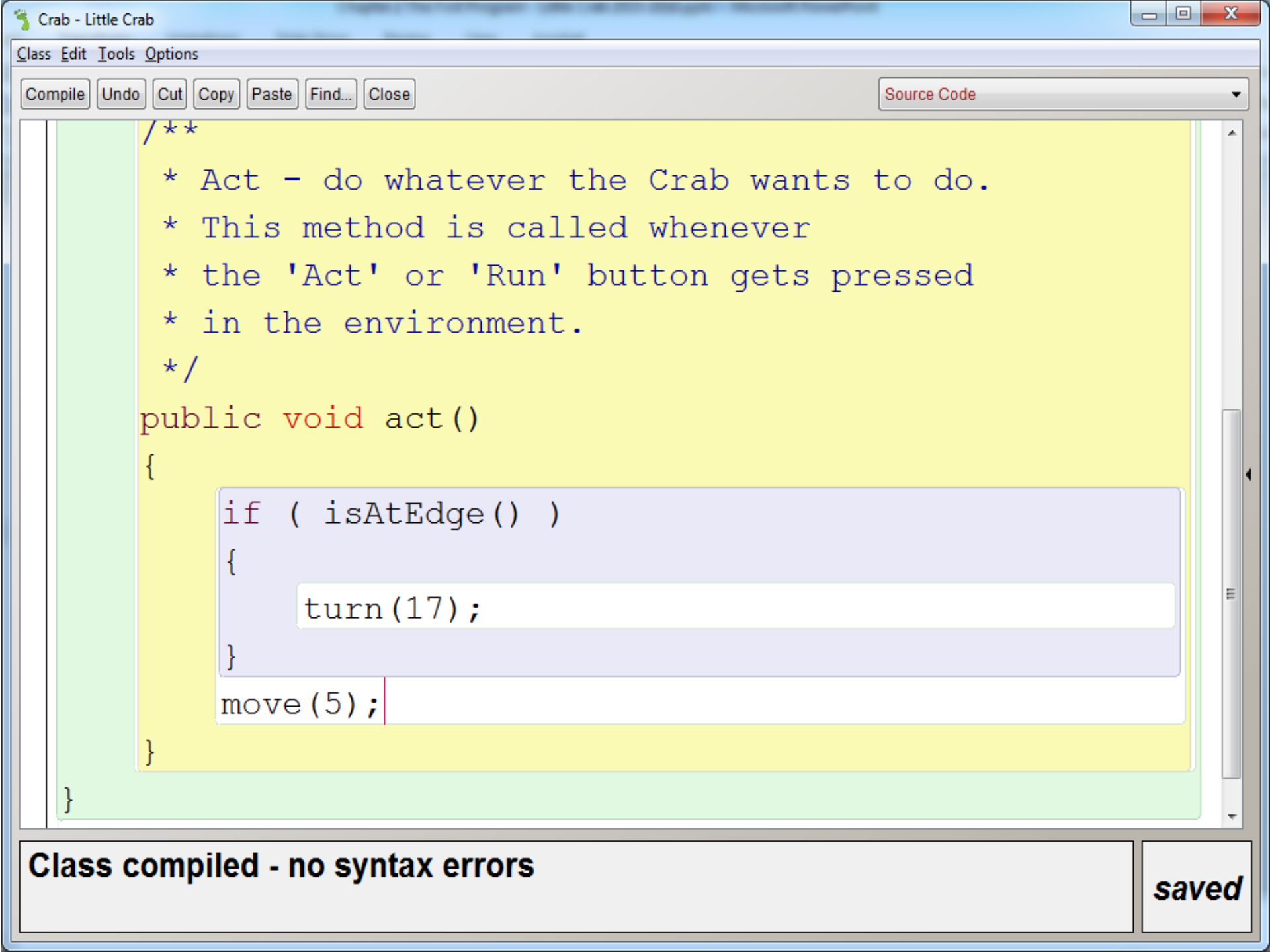
Compile Undo Cut Copy Paste Find... Close Source Code

```
 */
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()
    {
        //Write your conditional statement here
    }
}
```

Class compiled - no syntax errors

saved


Crab - Little Crab
Class Edit Tools Options
Compile Undo Cut Copy Paste Find... Close
Source Code
* 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()
{
if (isAtEdge())
{
turn(17);
move(5);
}
}
}
Compile, add some crabs
and then Run the Scenario
What happens?
Why?
Class compiled - no syntax errors
saved


Crab - Little Crab
Class Edit Tools Options
Compile Undo Cut Copy Paste Find... Close
Source Code
/**
* 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()
{
if (isAtEdge())
{
turn(17);
}
move(5);
}
}
Class compiled - no syntax errors
saved

Concept Summary

- ***Method Call***
  1. An instruction that tells an object to perform an action

  2. The action is defined by a method of the object

Concept Summary

- ***Parameter***
  1. Additional information passed to a method within the parentheses

  2. Examples:
     - move(5);
     - turn(17);

Concept Summary

- A subclass ***inherits*** all the methods from its superclass
  - This means that it has, and can use, all methods that its superclass defines

Concept Summary

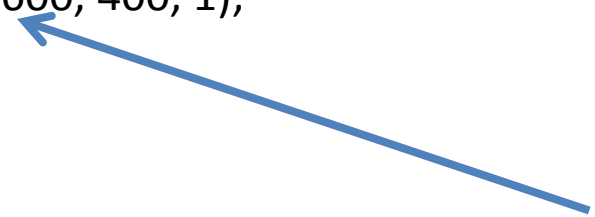
- ***Void versus non-void***
  - Calling a method with a ***void return type*** issues a command
  - Calling a method with a ***non-void return-type*** asks a question

Concept Summary

- ***If-Statements***
  - Used to write instructions that are only executed when a certain condition is true

Calling the Parent class Constructor

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

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

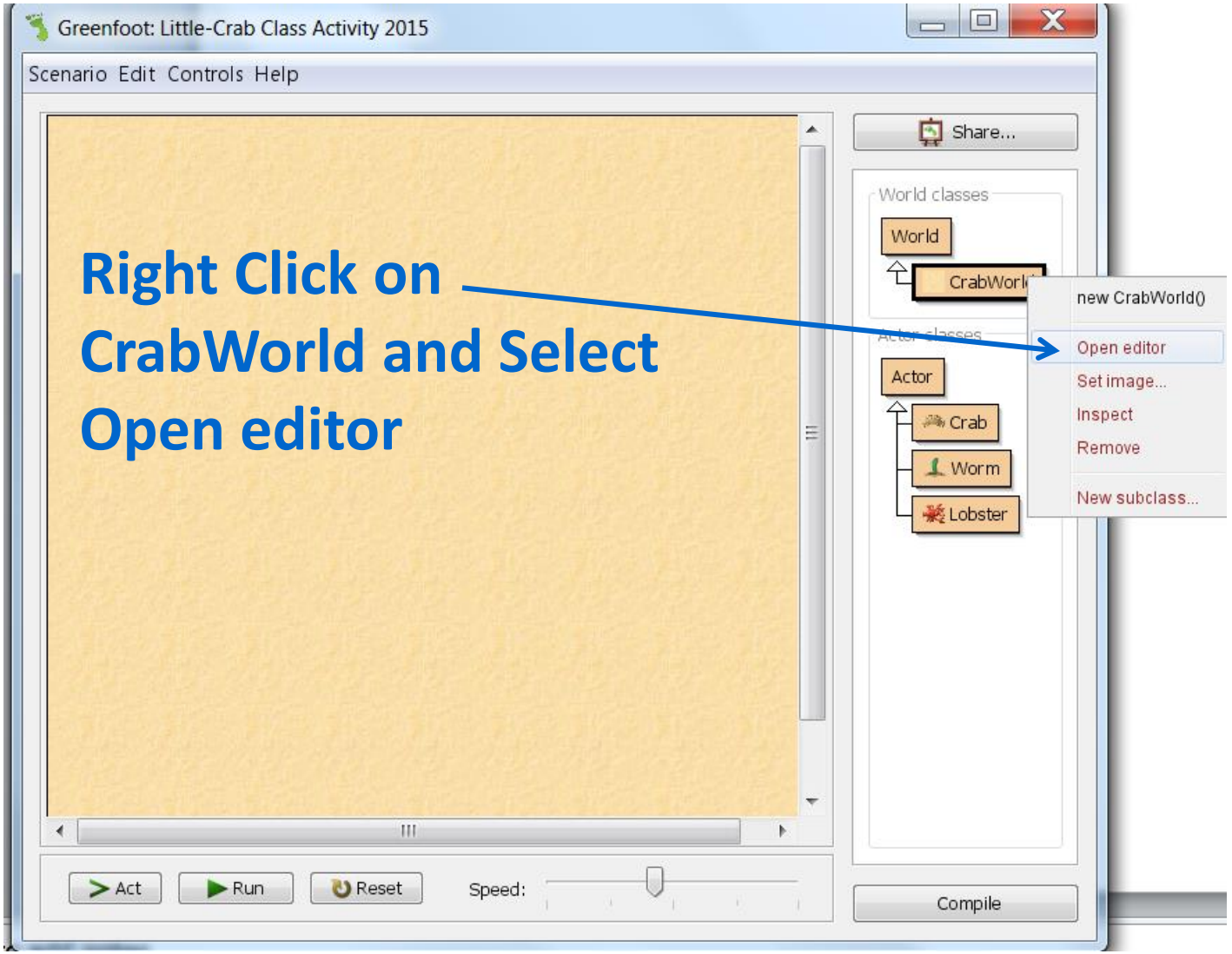


Calls the World class constructor to create the game area

Calling the Parent class Constructor

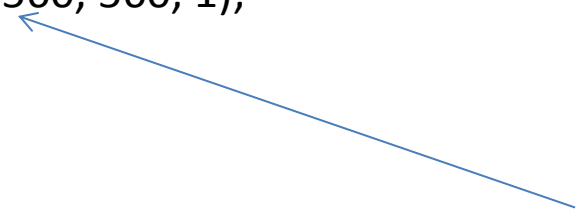
```
public class CrabWorld extends World
{
	public CrabWorld()
	{
		super(600, 400, 1);
	}
}
```

Adding Objects Automatically



Calling the Parent class Constructor

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

public class CrabWorld extends World
{
    /**
     * Create the crab world (the beach). Our world has a size
     * of 560x560 cells, where every cell is just 1 pixel.
     */
    public CrabWorld()
    {
        super(560, 560, 1);
    }
}
```



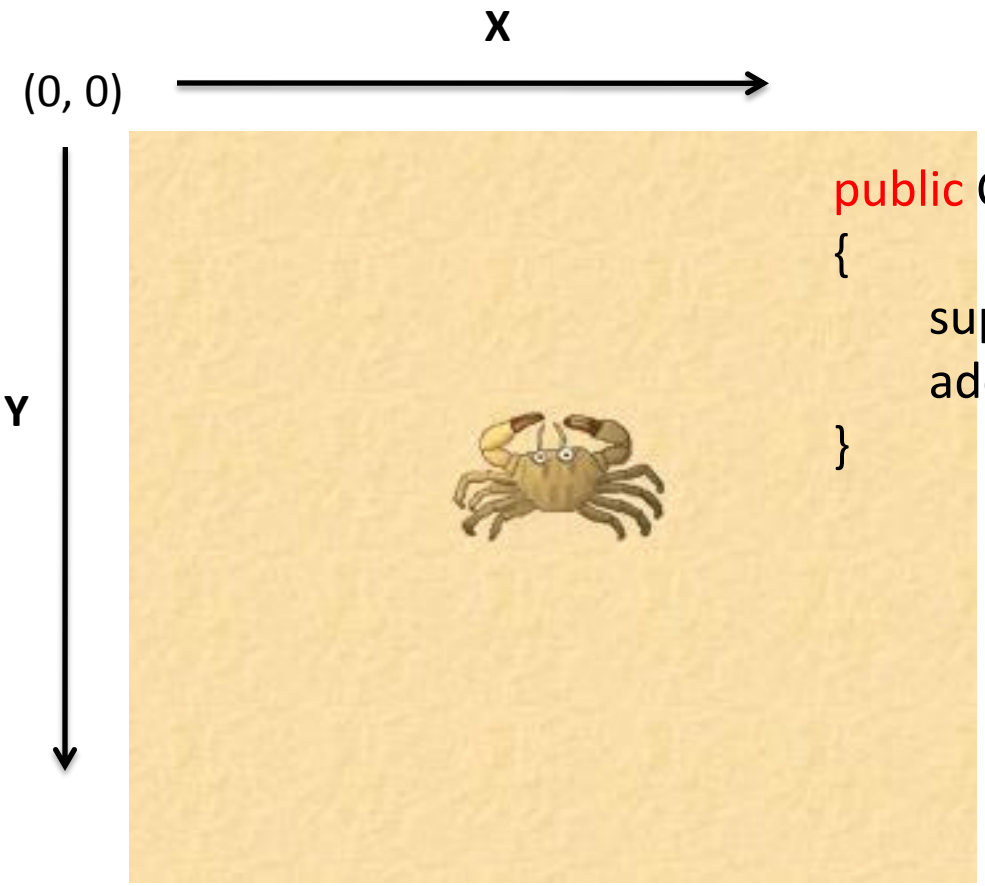
Calls the World class constructor to create the game area

Constructor for CrabWorld

```
public CrabWorld()
    {
        super(560, 560, 1);
    }
```

- Constructor has no Return Type Specified Between Keyword “public” and the Name
- Constructor Always has the Same Name as the Class
- Constructor is Automatically Executed Whenever and Instance of the Class is Created

Add a Crab to the World




```
public CrabWorld()
{
        super(560, 560, 1);
        addObject ( new Crab(), 150, 100);
}
```

Create a New Crab and Add it at Location x=150 and y=100

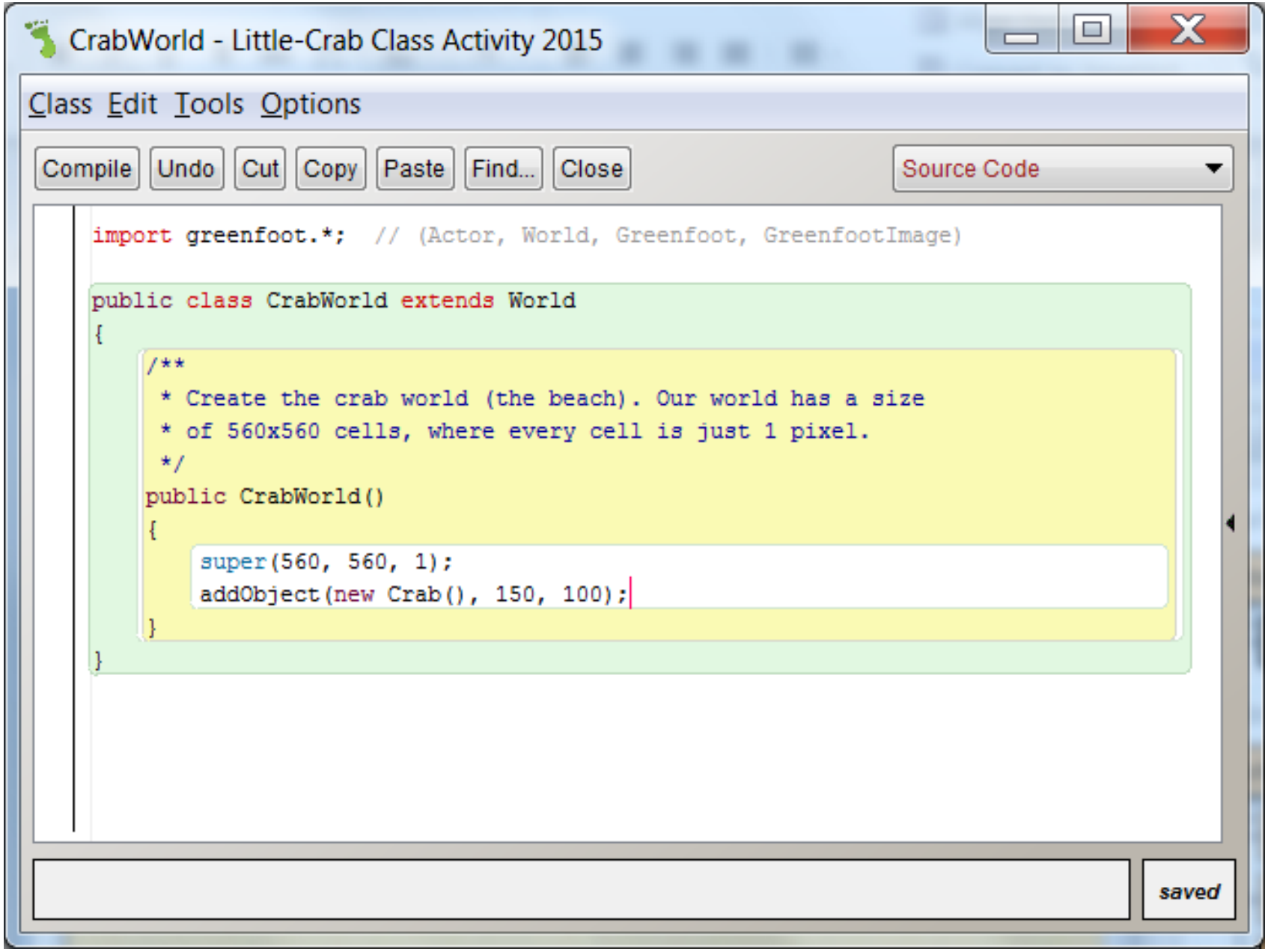
Creating New Objects

```
new Crab()
```

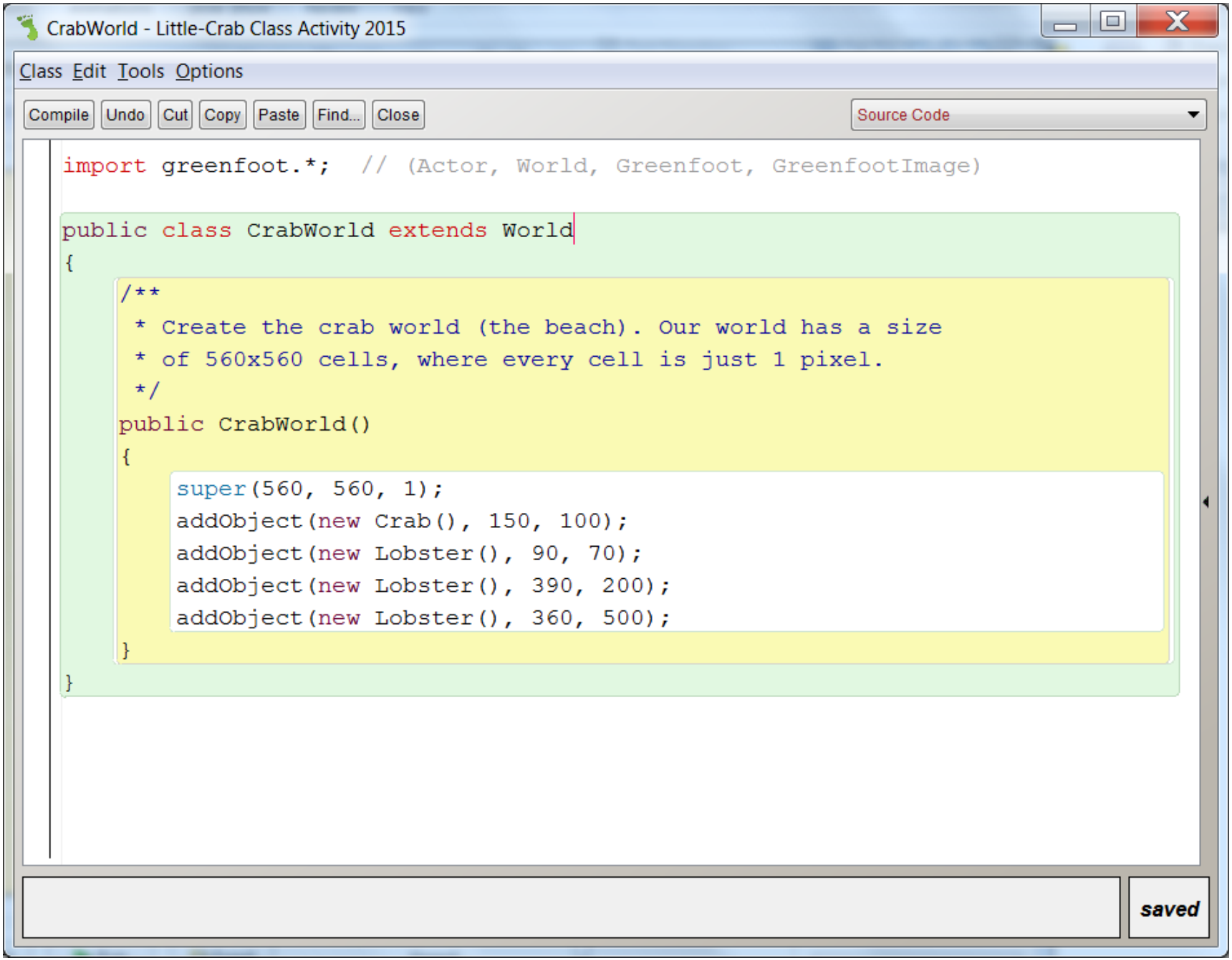
X-Position Y-Position

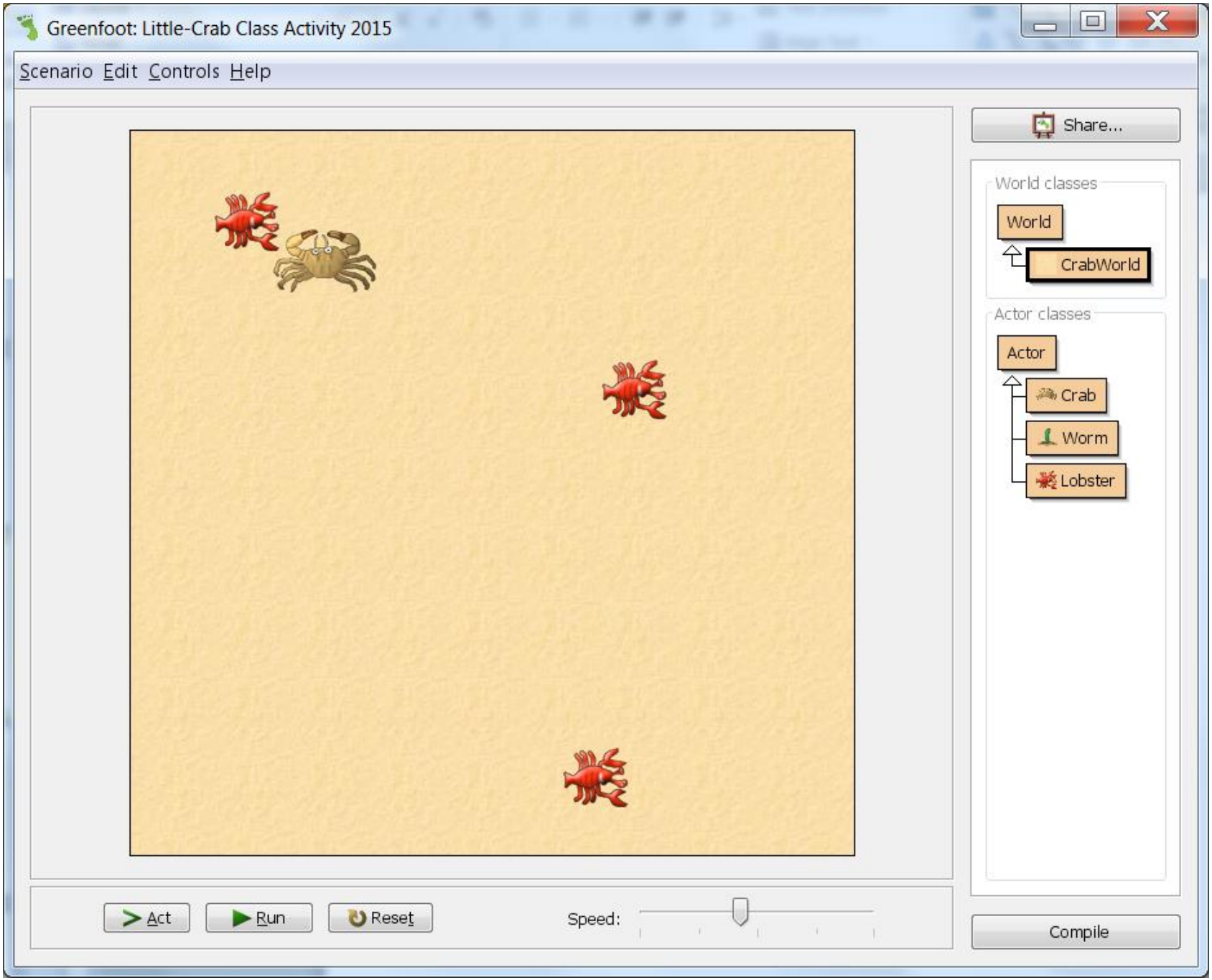
```
addObject ( new Crab(), 150, 100);
```

- Creates a New Instance of the Class Crab
- When We Create a New Object , We Must Do Something with It


CrabWorld - Little-Crab Class Activity 2015
Class Edit Tools Options
Compile Undo Cut Copy Paste Find... Close
Source Code
import greenfoot.*; // (Actor, World, Greenfoot, GreenfootImage)
public class CrabWorld extends World
{
/**
* Create the crab world (the beach). Our world has a size
* of 560x560 cells, where every cell is just 1 pixel.
*/
public CrabWorld()
{
super(560, 560, 1);
addObject(new Crab(), 150, 100);
}
}
saved


Greenfoot: Little-Crab Class Activity 2015
Scenario Edit Controls Help
The Crab is Automatically
Created in CrabWorld
Share...
World classes
World
CrabWorld
Actor classes
Actor
Crab
Worm
Lobster
Act
Run
Reset
Speed:
Compile


CrabWorld - Little-Crab Class Activity 2015
Class Edit Tools Options
Compile Undo Cut Copy Paste Find... Close
Source Code
import greenfoot.*; // (Actor, World, Greenfoot, GreenfootImage)
public class CrabWorld extends World
{
/**
* Create the crab world (the beach). Our world has a size
* of 560x560 cells, where every cell is just 1 pixel.
*/
public CrabWorld()
{
super(560, 560, 1);
addObject(new Crab(), 150, 100);
addObject(new Lobster(), 90, 70);
addObject(new Lobster(), 390, 200);
addObject(new Lobster(), 360, 500);
}
}
saved


Greenfoot: Little-Crab Class Activity 2015
Scenario Edit Controls Help
Share...
World classes
World
CrabWorld
Actor classes
Actor
Crab
Worm
Lobster
Act
Run
Reset
Speed:
Compile

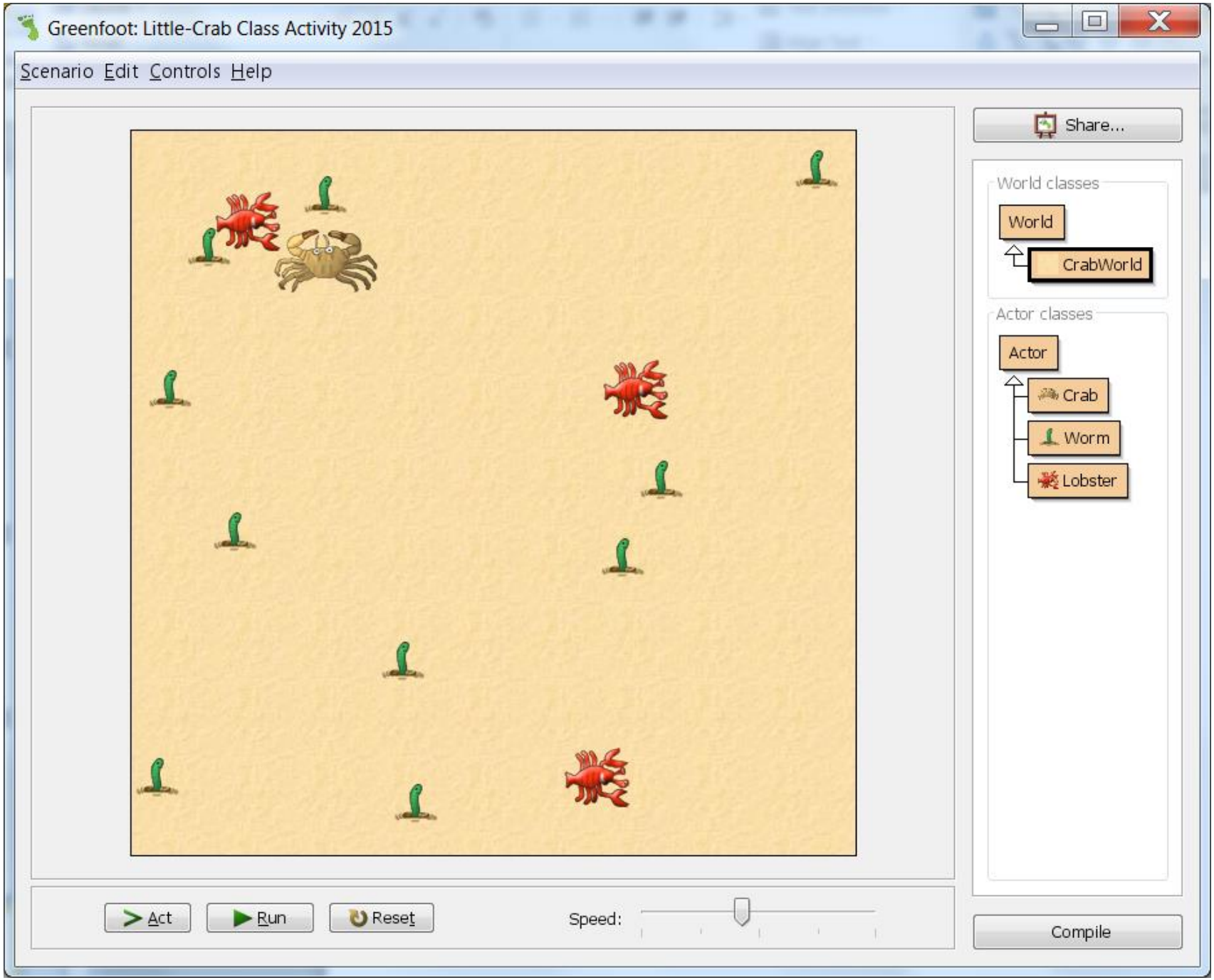
CrabWorld - Little-Crab Class Activity 2015

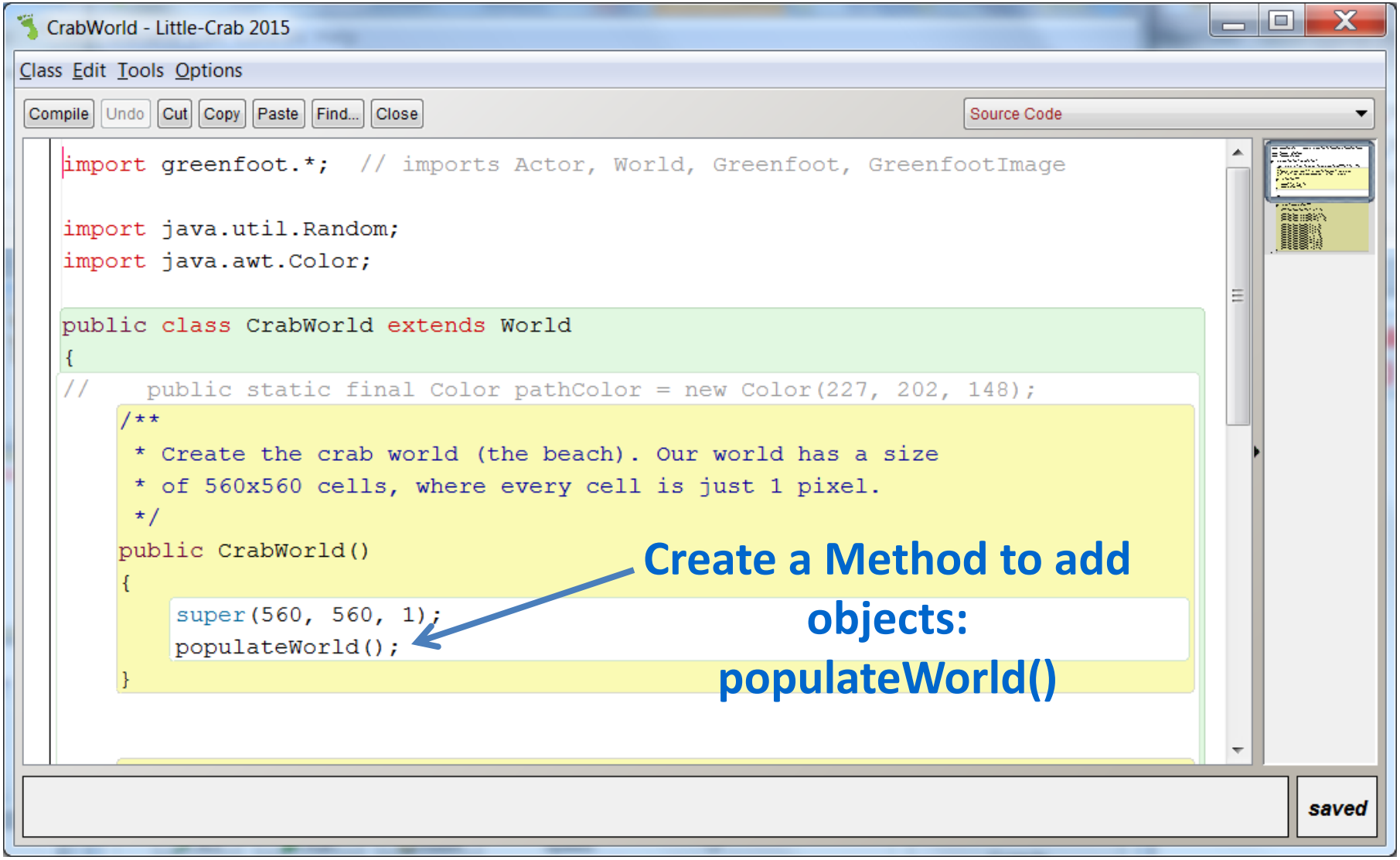
Class Edit Tools Options

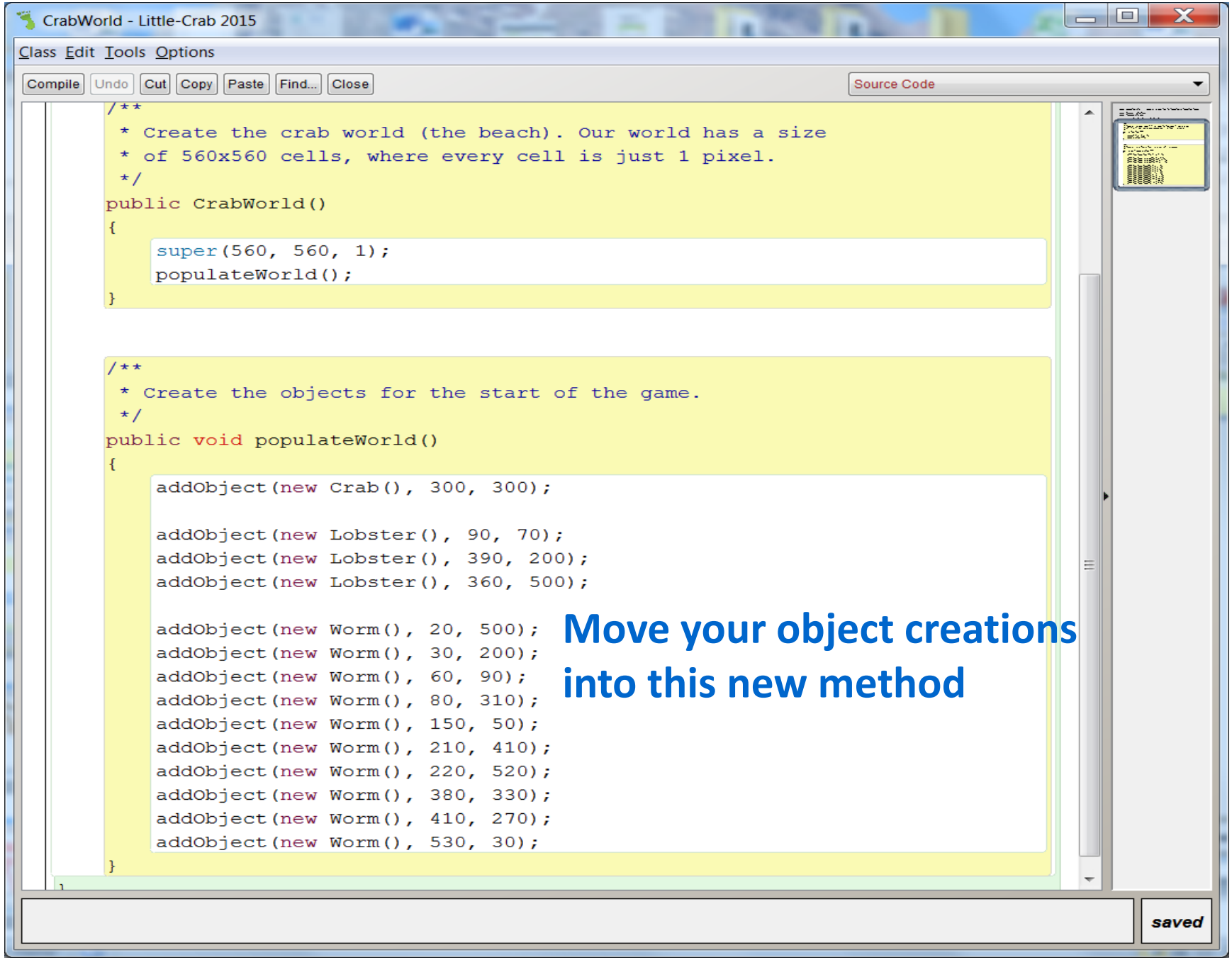
Compile Undo Cut Copy Paste Find... Close Source Code

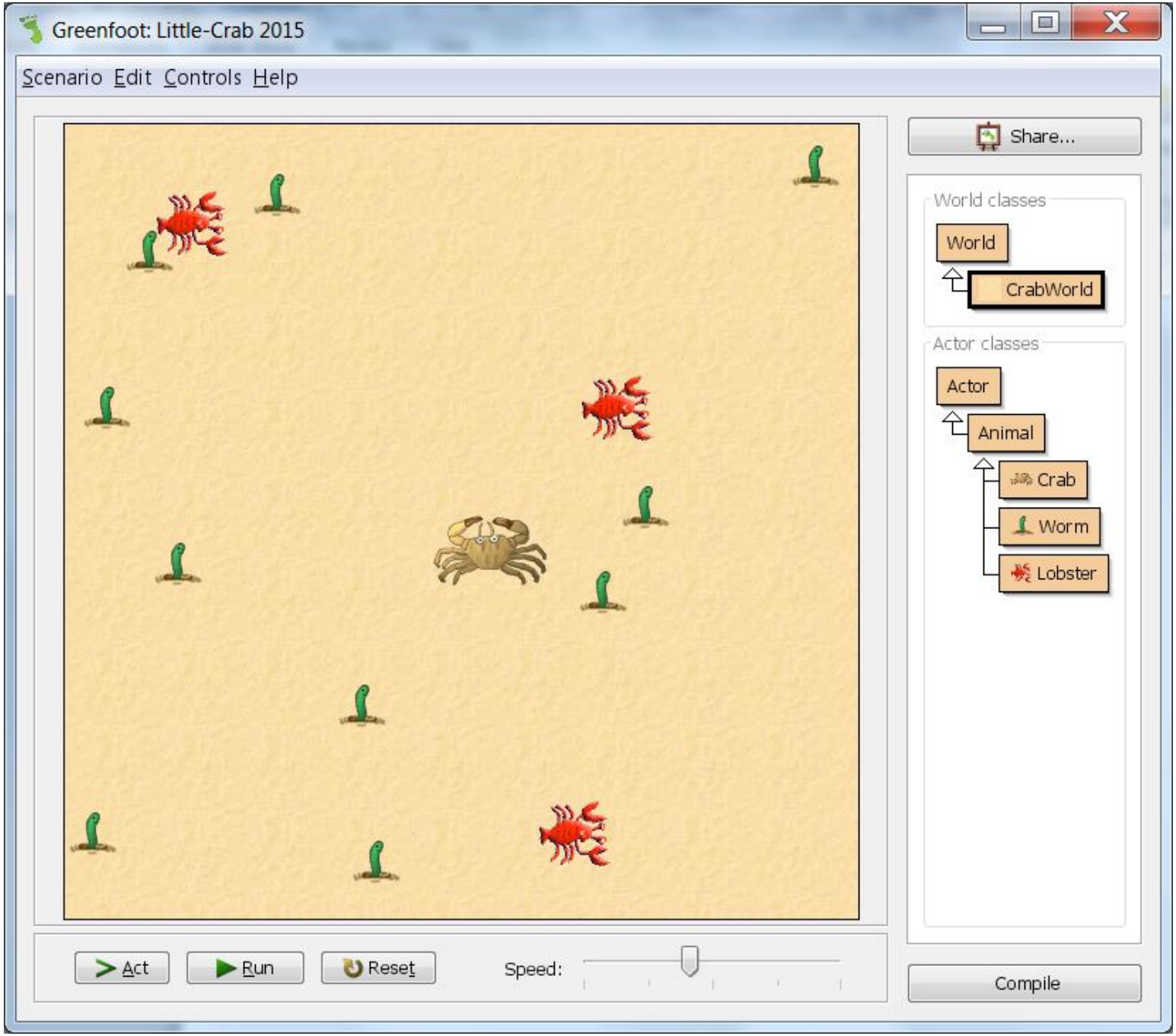
```
import greenfoot.*;  // (Actor, World, Greenfoot, GreenfootImage)

public class CrabWorld extends World
{
    /**
     * Create the crab world (the beach). Our world has a size
     * of 560x560 cells, where every cell is just 1 pixel.
     */
    public CrabWorld()
    {
        super(560, 560, 1);
        addObject(new Crab(), 150, 100);
        addObject(new Lobster(), 90, 70);
        addObject(new Lobster(), 390, 200);
        addObject(new Lobster(), 360, 500);

        addObject(new Worm(), 20, 500);
        addObject(new Worm(), 30, 200);
        addObject(new Worm(), 60, 90);
        addObject(new Worm(), 80, 310);
        addObject(new Worm(), 150, 50);
        addObject(new Worm(), 210, 410);
        addObject(new Worm(), 220, 520);
        addObject(new Worm(), 380, 330);
        addObject(new Worm(), 410, 270);
        addObject(new Worm(), 530, 30);
    }
}
```

changed


Greenfoot: Little-Crab Class Activity 2015
Scenario Edit Controls Help
Share...
World classes
World
CrabWorld
Actor classes
Actor
Crab
Worm
Lobster
Act
Run
Reset
Speed:
Compile


CrabWorld - Little-Crab 2015
Class Edit Tools Options
Compile Undo Cut Copy Paste Find... Close
Source Code
import greenfoot.*; // imports Actor, World, Greenfoot, GreenfootImage
import java.util.Random;
import java.awt.Color;
public class CrabWorld extends World
{
// public static final Color pathColor = new Color(227, 202, 148);
/**
* Create the crab world (the beach). Our world has a size
* of 560x560 cells, where every cell is just 1 pixel.
*/
public CrabWorld()
{
super(560, 560, 1);
populateWorld();
}
Create a Method to add objects: populateWorld()
saved


CrabWorld - Little-Crab 2015
Class Edit Tools Options
Compile Undo Cut Copy Paste Find... Close
Source Code
/**
* Create the crab world (the beach). Our world has a size
* of 560x560 cells, where every cell is just 1 pixel.
*/
public CrabWorld()
{
super(560, 560, 1);
populateWorld();
}
/**
* Create the objects for the start of the game.
*/
public void populateWorld()
{
addObject(new Crab(), 300, 300);
addObject(new Lobster(), 90, 70);
addObject(new Lobster(), 390, 200);
addObject(new Lobster(), 360, 500);
addObject(new Worm(), 20, 500);
addObject(new Worm(), 30, 200);
addObject(new Worm(), 60, 90);
addObject(new Worm(), 80, 310);
addObject(new Worm(), 150, 50);
addObject(new Worm(), 210, 410);
addObject(new Worm(), 220, 520);
addObject(new Worm(), 380, 330);
addObject(new Worm(), 410, 270);
addObject(new Worm(), 530, 30);
}
Move your object creations into this new method
saved


Greenfoot: Little-Crab 2015
Scenario Edit Controls Help
Share...
World classes
World
CrabWorld
Actor classes
Actor
Animal
Crab
Worm
Lobster
Act
Run
Reset
Speed:
Compile

An easier Way

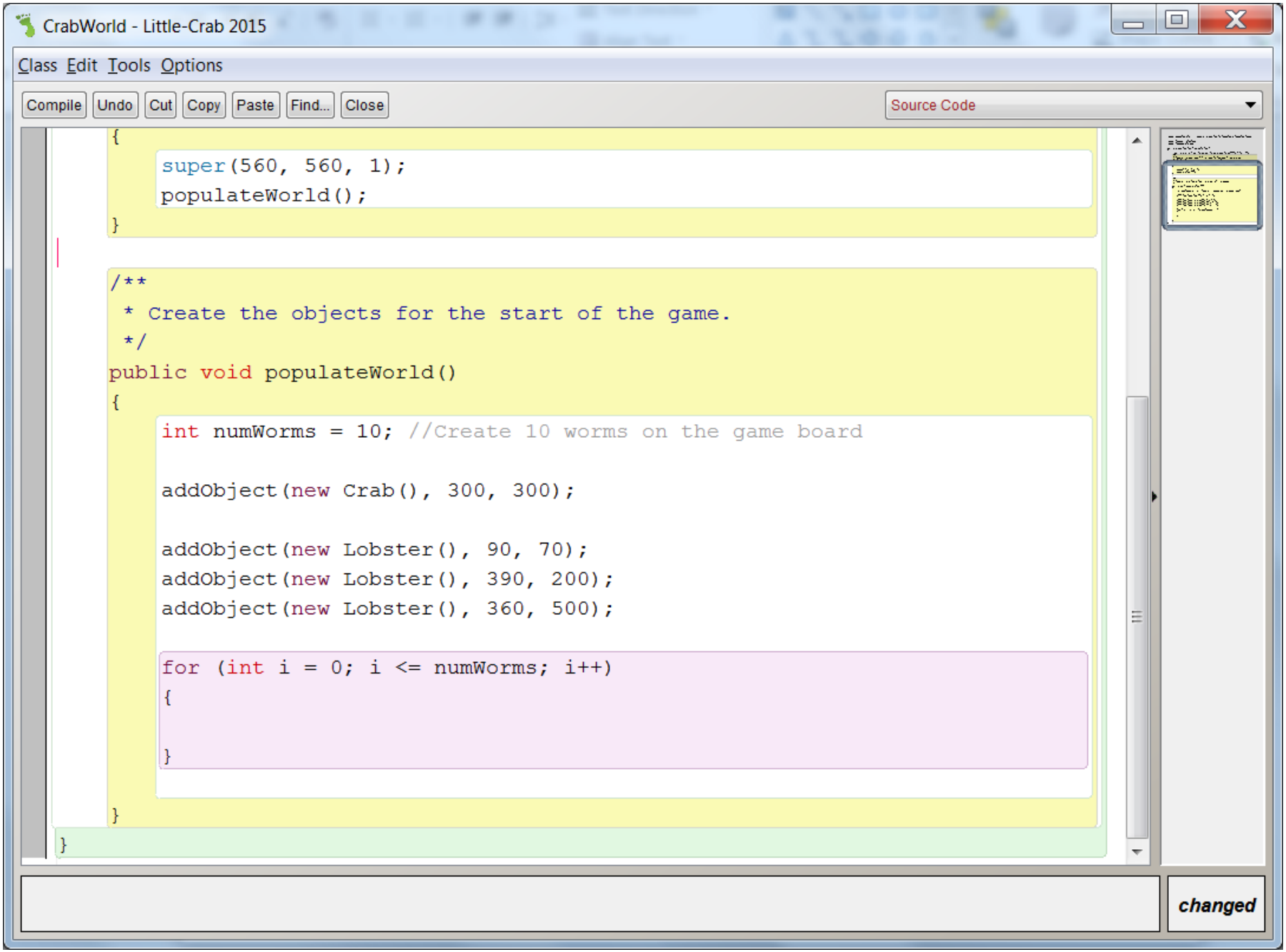
- We now have a way to create a new Crab World

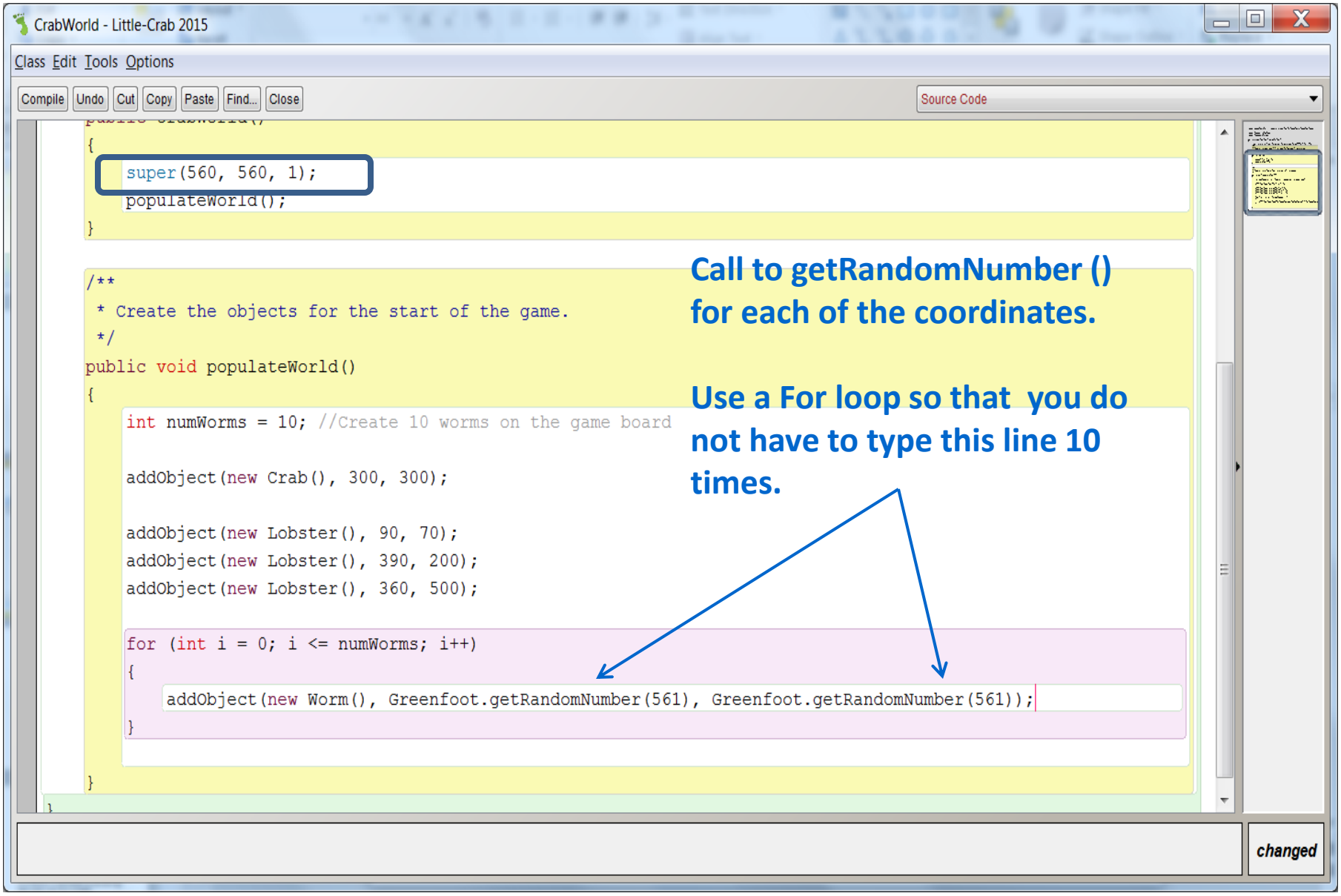
- **Problem:**
  - If we need to add many different objects to the game board we will have to type every statement one at a time – **There is an easier way**

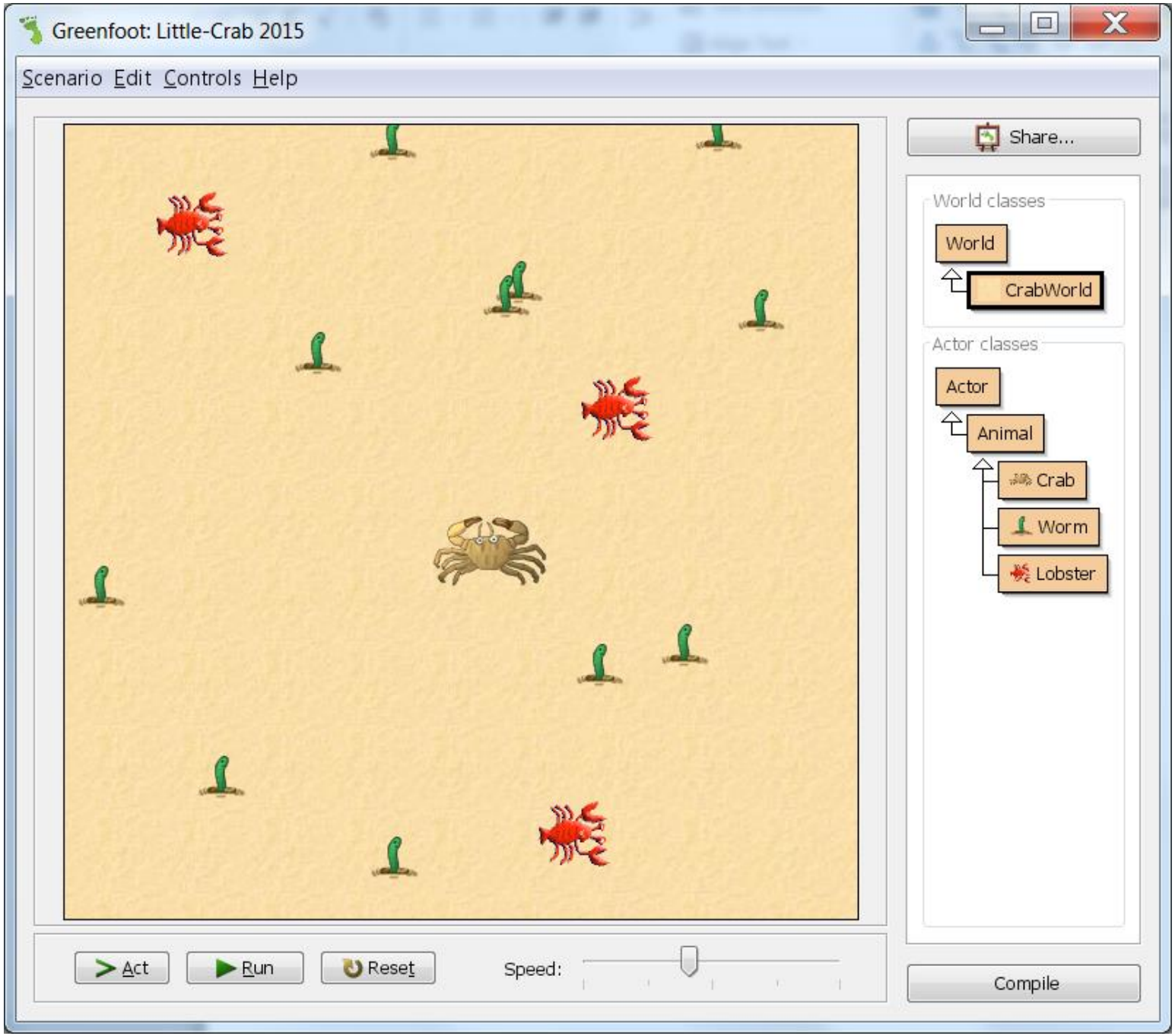
Using Random Numbers

- Modify the **CrabWorld** class to use random numbers for the coordinates of the worms

- We will use a loop statement to help us create the random worms


CrabWorld - Little-Crab 2015
Class Edit Tools Options
Compile Undo Cut Copy Paste Find... Close
Source Code
{
super(560, 560, 1);
populateWorld();
}
/**
* Create the objects for the start of the game.
*/
public void populateWorld()
{
int numWorms = 10; //Create 10 worms on the game board
addObject(new Crab(), 300, 300);
addObject(new Lobster(), 90, 70);
addObject(new Lobster(), 390, 200);
addObject(new Lobster(), 360, 500);
for (int i = 0; i <= numWorms; i++)
{
}
}
}
changed


CrabWorld - Little-Crab 2015
Class Edit Tools Options
Compile Undo Cut Copy Paste Find... Close
Source Code
{
super(560, 560, 1);
populateWorld();
}
/**
* Create the objects for the start of the game.
*/
public void populateWorld()
{
int numWorms = 10; //Create 10 worms on the game board
addObject(new Crab(), 300, 300);
addObject(new Lobster(), 90, 70);
addObject(new Lobster(), 390, 200);
addObject(new Lobster(), 360, 500);
for (int i = 0; i <= numWorms; i++)
{
addObject(new Worm(), Greenfoot.getRandomNumber(561), Greenfoot.getRandomNumber(561));
}
}
Call to getRandomNumber () for each of the coordinates.
Use a For loop so that you do not have to type this line 10 times.
changed


Greenfoot: Little-Crab 2015
Scenario Edit Controls Help
Share...
World classes
World
CrabWorld
Actor classes
Actor
Animal
Crab
Worm
Lobster
Act
Run
Reset
Speed:
Compile

Animating Images

Crab with legs out

Crab with legs in

Animation is Achieved by Switching Between the Two Images

Crab Images

- Go to the shared folder and copy and paste the **Crab2** image to your images folder within your Little-Crab folder

Crab with legs out:
Crab.png

Crab with legs in:
Crab2.png

setImage method

---

setImage

```
public void setImage(GreenfootImage image)
```

Set the image for this object to the specified image.

Parameters:

image - The image.

See Also:

setImage(String)

Instance Variables (Fields)

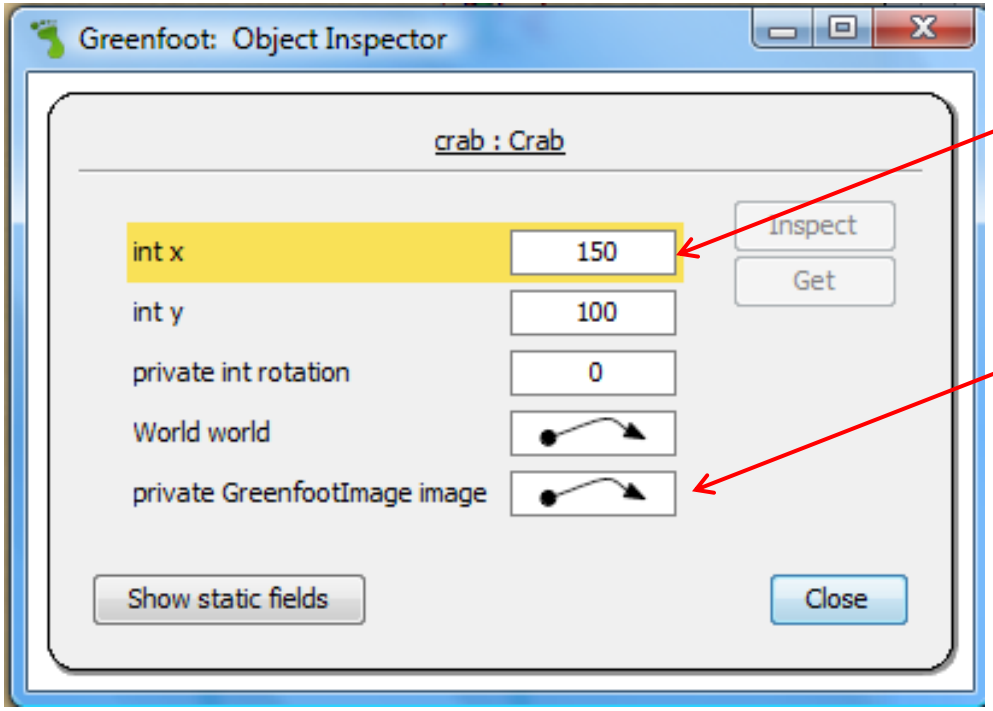
private *variable-type* *variable-name*

An instance variable is a variable that belongs to the object (an instance of a class)

Add the following **fields** (**Instance Variables**) to the Crab Class

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

//  comment omitted

public class Crab extends Animal
{
        private GreenfootImage image1;
        private GreenfootImage image2;

        // methods omitted

}
```

Exercise 4.8



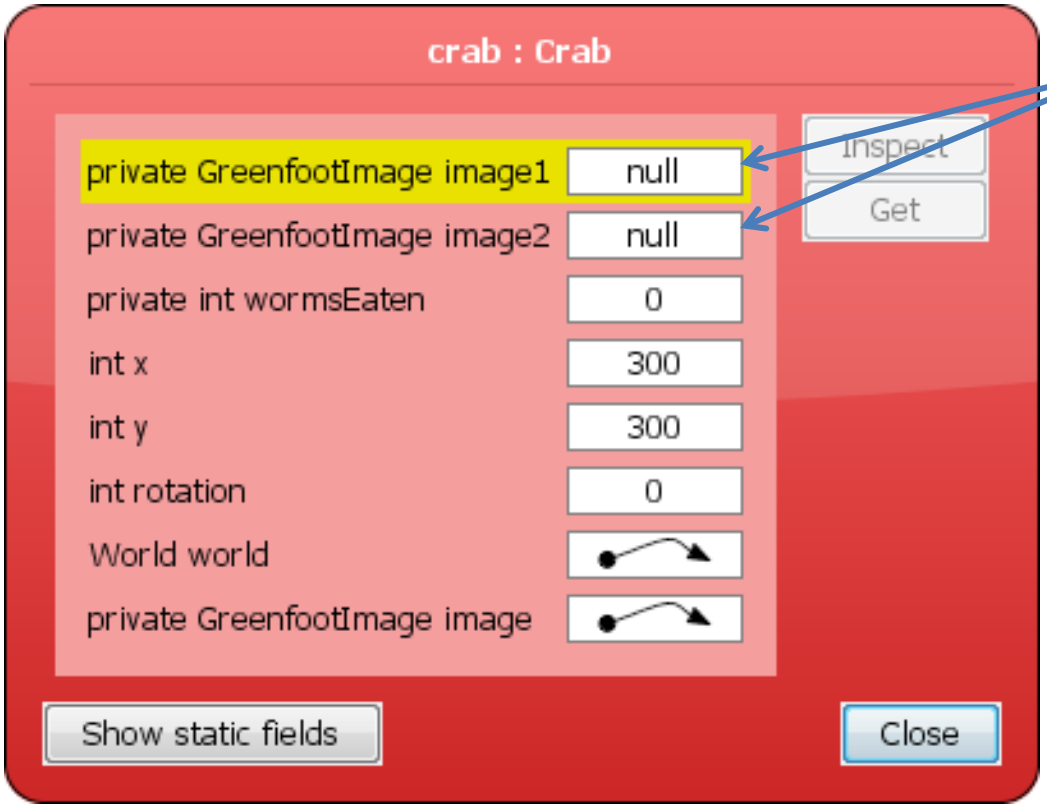
Variables for the Crabs Location in the World

A Variable of the Crabs Image

The Crab Class – So far

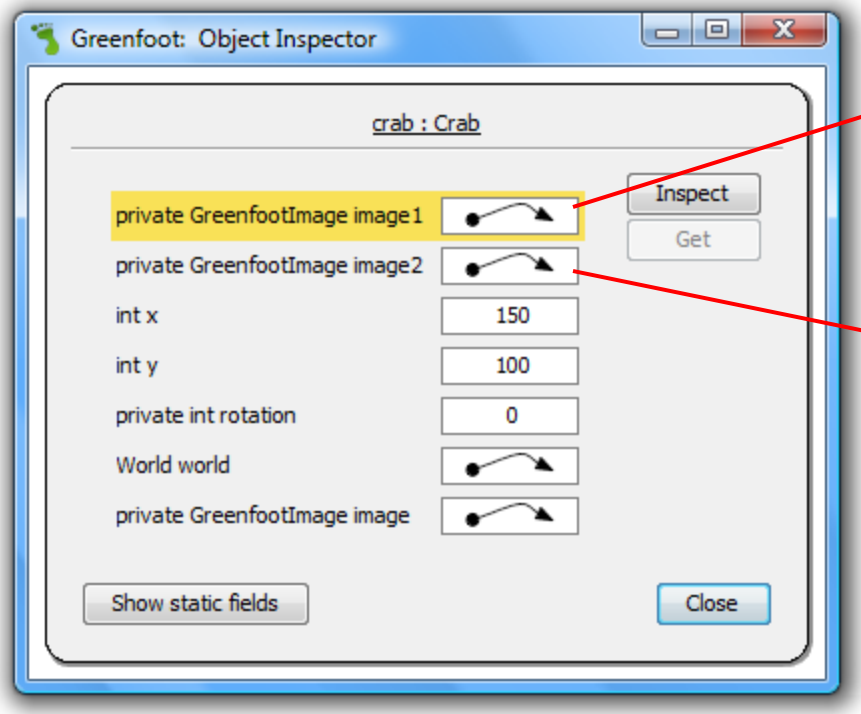
```
public class Crab extends Animal
{
        private GreenfootImage image1;
        private GreenfootImage image2;

        /*
        * 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()
        {
                checkKeypress();
                move();
                lookForWorm();
         }
}
```

Right-Click on a Crab object and select the Inspect option



The Variables Have Not Been Initialized and Contain null

Creation and Assignment



crab.png

crab2.png

image1 = new GreenfootImage (“crab.png”);
image2 = new GreenfootImage (“crab2.png”);

4.7 Using actor Constructors

```
public Crab()
{
        image1 = new GreenfootImage ("crab.png");
        image2 = new GreenfootImage ("crab2.png");
        setImage (image1);
}
```

The constructor for the Crab class creates two images and assigns them to variables for use later.

Code 4.3

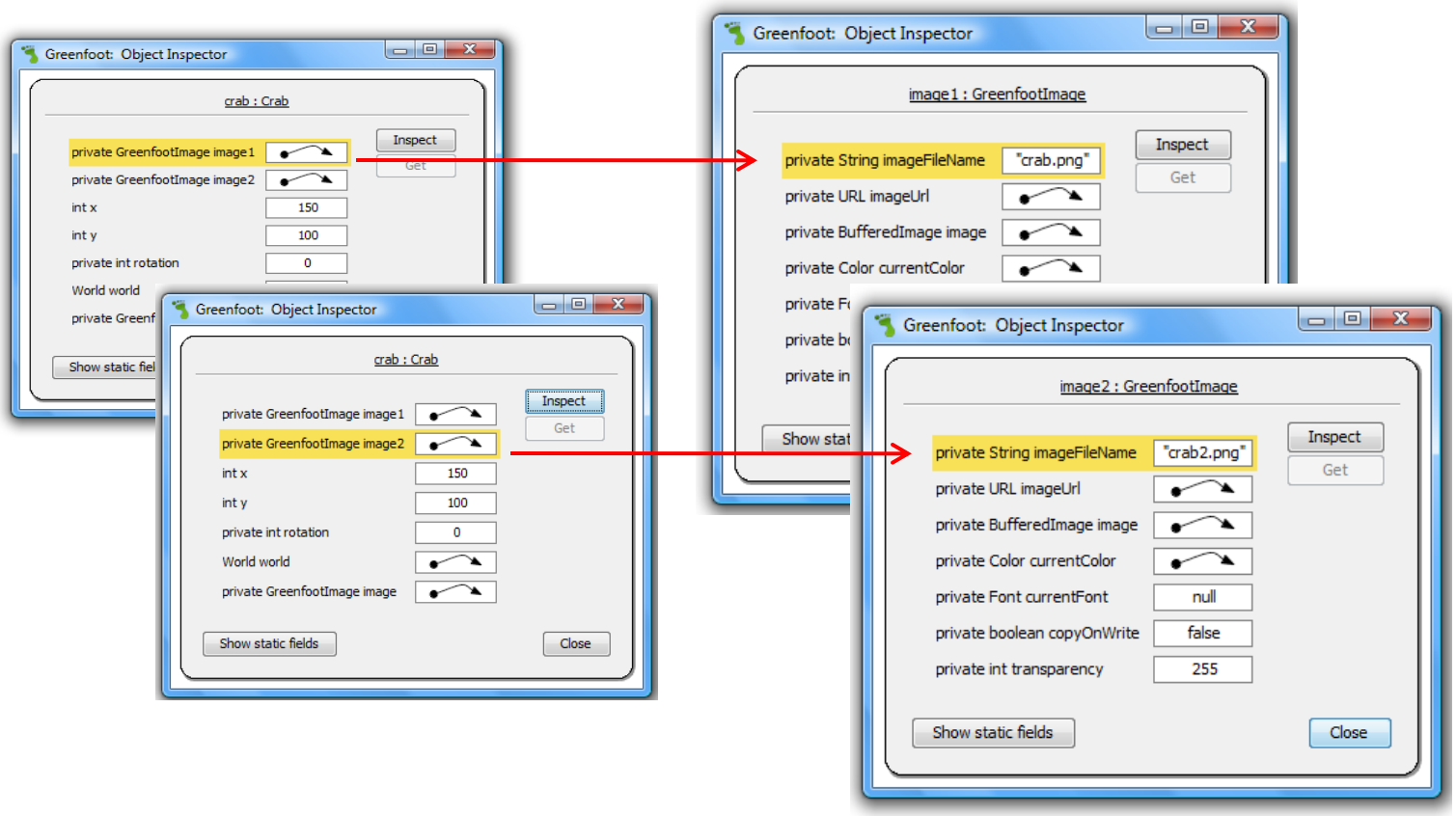
```
import greenfoot.*;  // (World, Actor, GreenfootImage, and Greenfoot)

// comment omitted

public class Crab extends Animal
{
        private GreenfootImage image1;
        private GreenfootImage image2;

        /*
         * Create a crab and initialize its two images.
         */
        public Crab()
        {
                image1 = new GreenfootImage ("crab.png");
                image2 = new GreenfootImage ("crab2.png");
                setImage (image1);
        }

        // methods omitted
}
```

- **The signature of a constructor does not include a return type.**
- **The name of the constructor is the same as the name of the class.**
- **The constructor is automatically executed when a crab object is created.**

Exercise 4.11

```
public class Crab extends Animal
{
        private GreenfootImage image1;
        private GreenfootImage image2;

        /*
          * Create a crab and initialize its two images.
          */
        public Crab()
        {
                image1 = new GreenfootImage ("crab.png");
                image2 = new GreenfootImage ("crab2.png");
                setImage (image1);
        }
```

Exercise 4.12



4.8 Alternating The Images

Pseudo Code

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

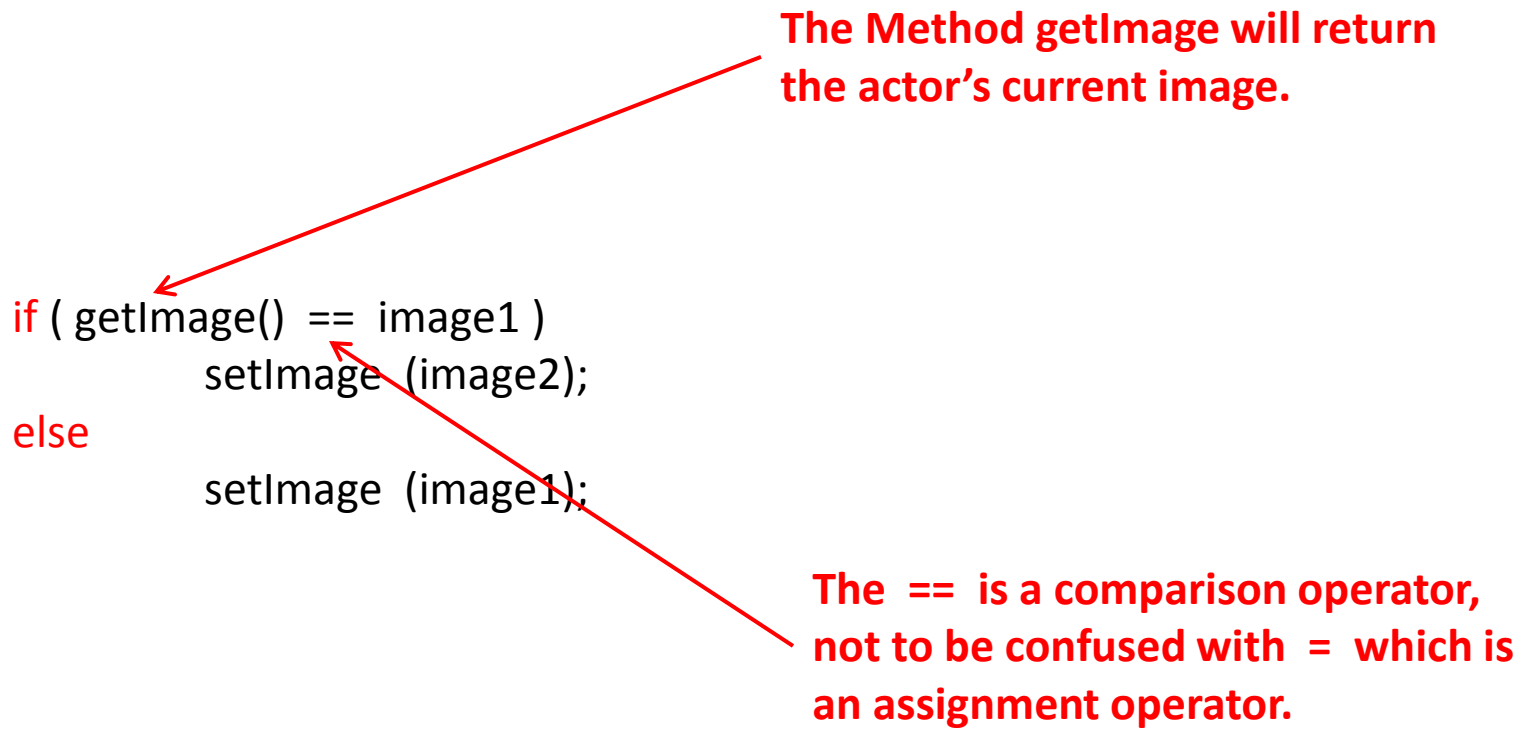
Actual Code

```
if  ( getImage()  ==  image1 )
        setImage  (image2);
else
        setImage  (image1);
```

Code 4.4

```
if ( getImage()  ==  image1 )
{
        setImage  (image2);
}
else
{
        setImage  (image1);
}
```

if/else When Only One Statement



Comparison Operators

Operator	*Meaning*
<	less than
<=	less than or equal to
==	equal to
>=	greater than or equal to
>	greater than
!=	not equal

4.9 The if/else Statement

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

if-clause

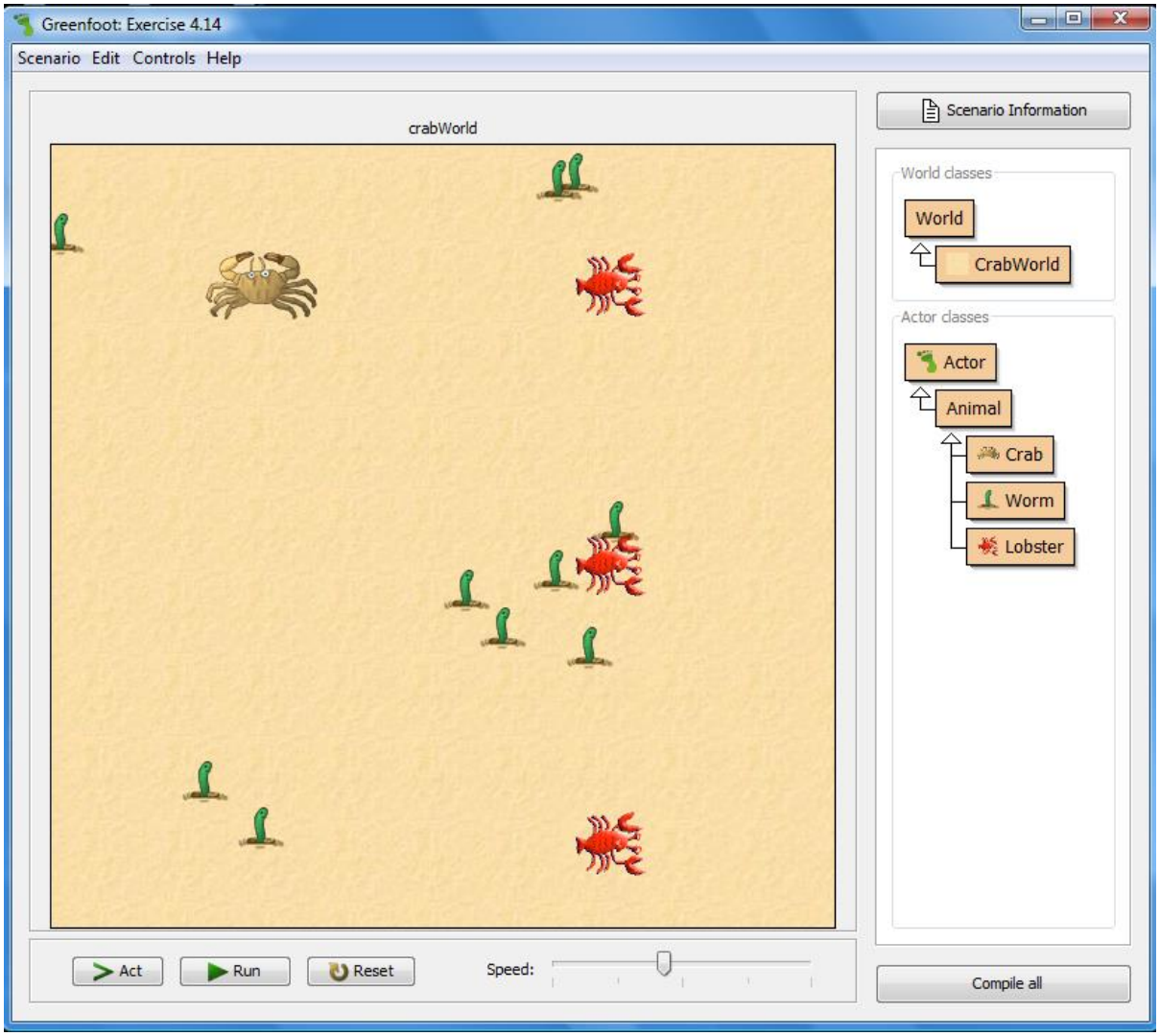
else-clause

If the condition is TRUE the if-clause will be executed, otherwise the else-clause will be executed

Exercise 4.13

```
/*
 * 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()
{
        if (getImage() == image1)
        {
                setImage (image2);
        }
        else
        {
                setImage (image1);
        }
        checkKeypress();
        move();
        lookForWorm();
}
```

Exercise 4.13



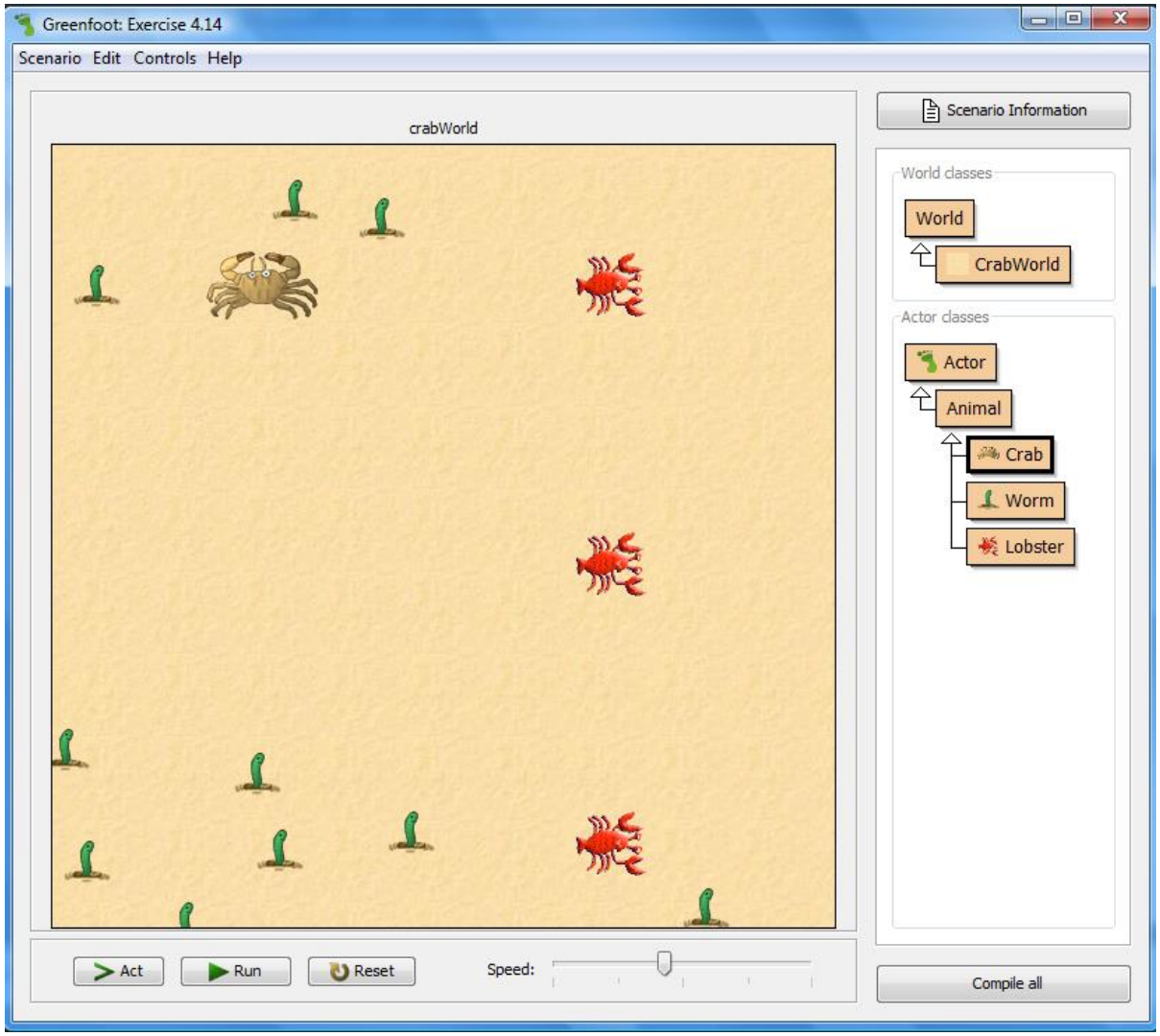
Exercise 4.14

```
/*
  * Switch the images of the Crab to make it appear as if the Crab is moving it's legs.
  * If the current image is image1 switch it to image2 and vice versa.
  */
public void switchImage()
{
		if (getImage() == image1)
		{
			setImage (image2);
		}
		else
		{
			setImage (image1);
		}
}
```

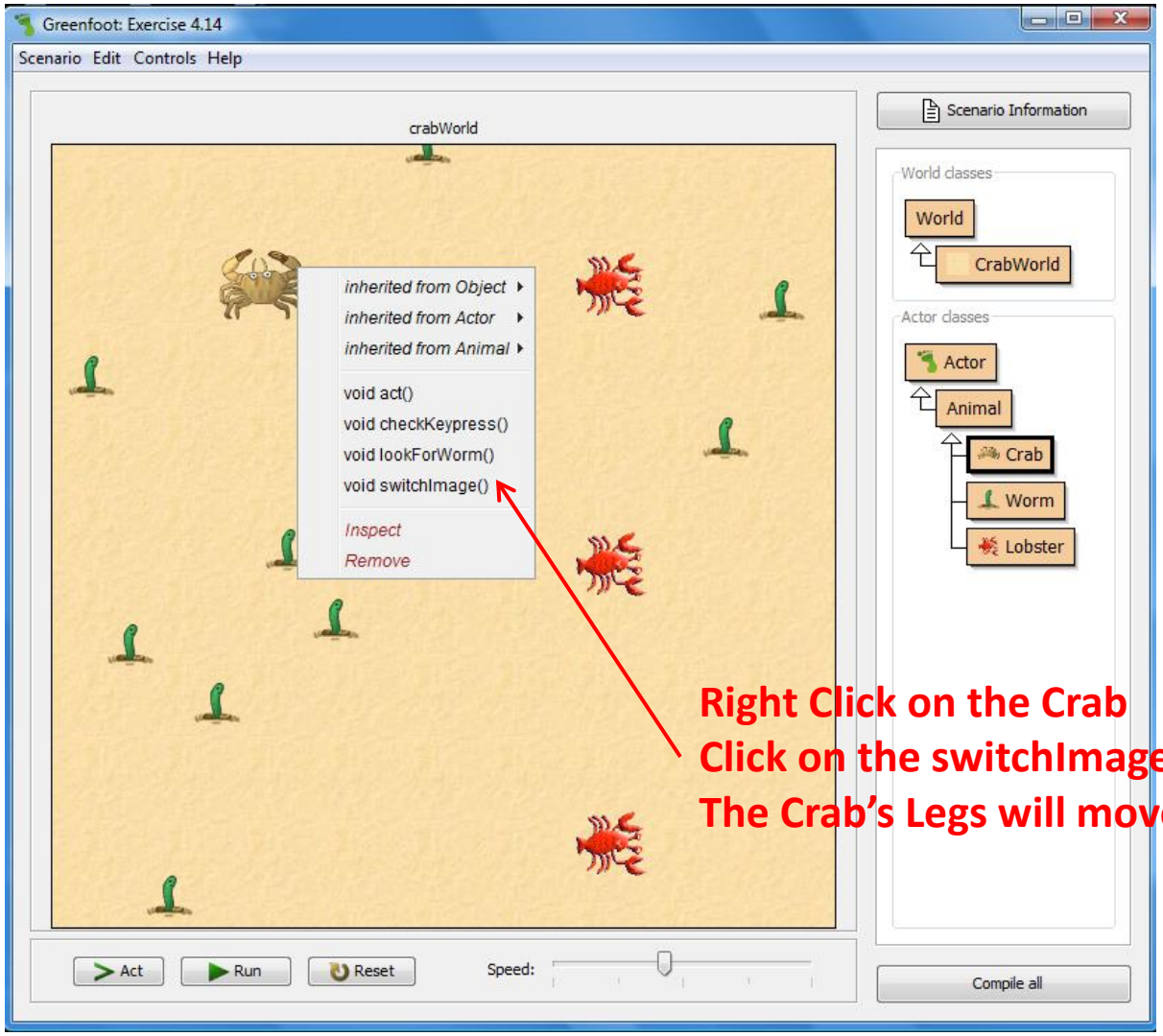
act Method

```
/*
  * 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()
{
        checkKeypress();
        switchImage();
        move();
        lookForWorm();
}
```

Exercise 4.14



Exercise 4.15



4.10 Counting Worms

- An instance variable to store the current count of worms eaten

```
private int wormsEaten;
```

- An assignment that initializes this variable to 0 at the beginning

```
wormsEaten = 0;
```

- Code to increment our count each time we eat a worm

```
wormsEaten = 0;
```

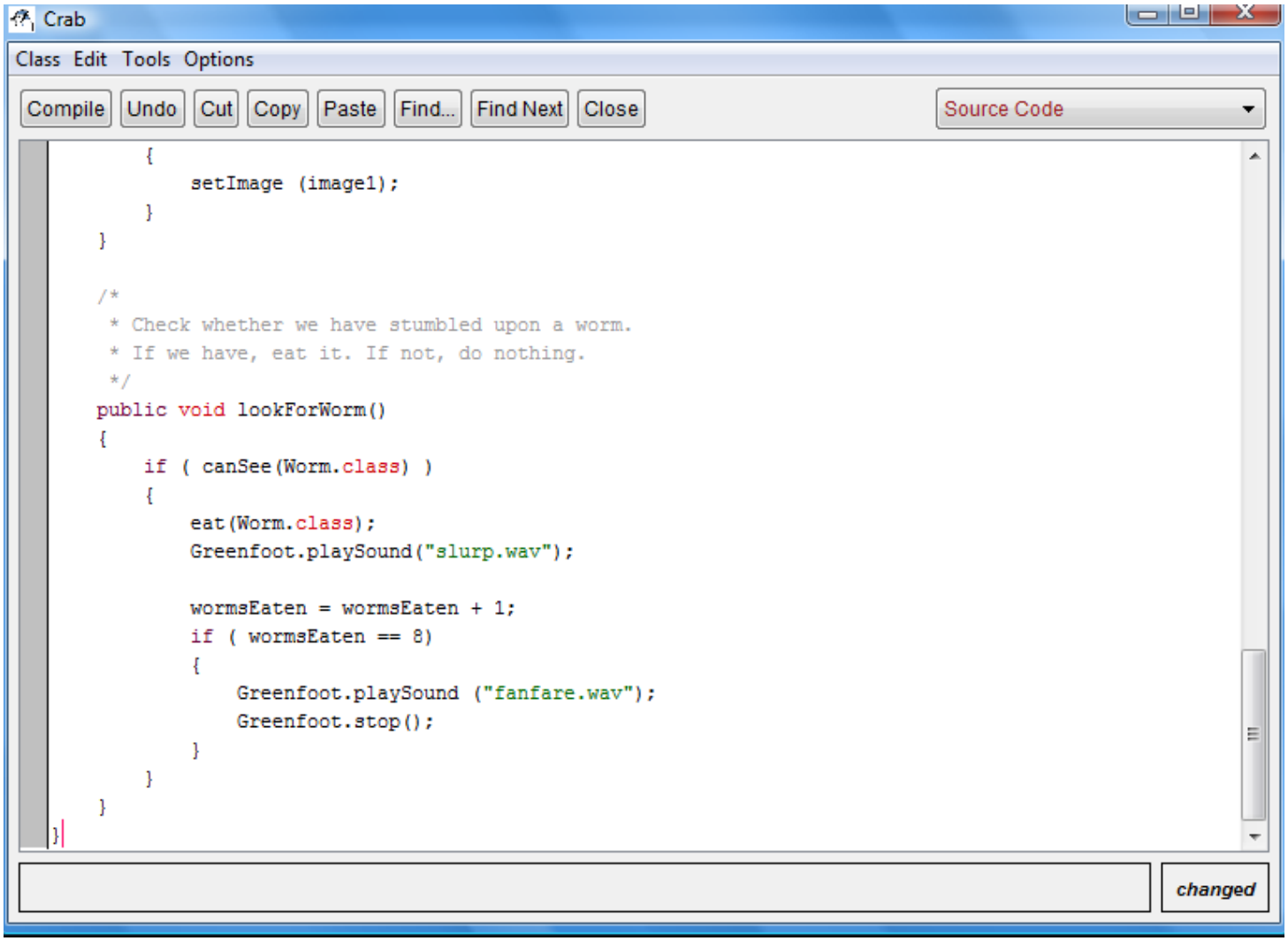
- Code that checks whether we have eaten eight worms and stops the game and plays the sound it we have

```
wormsEaten = wormsEaten + 1;
```

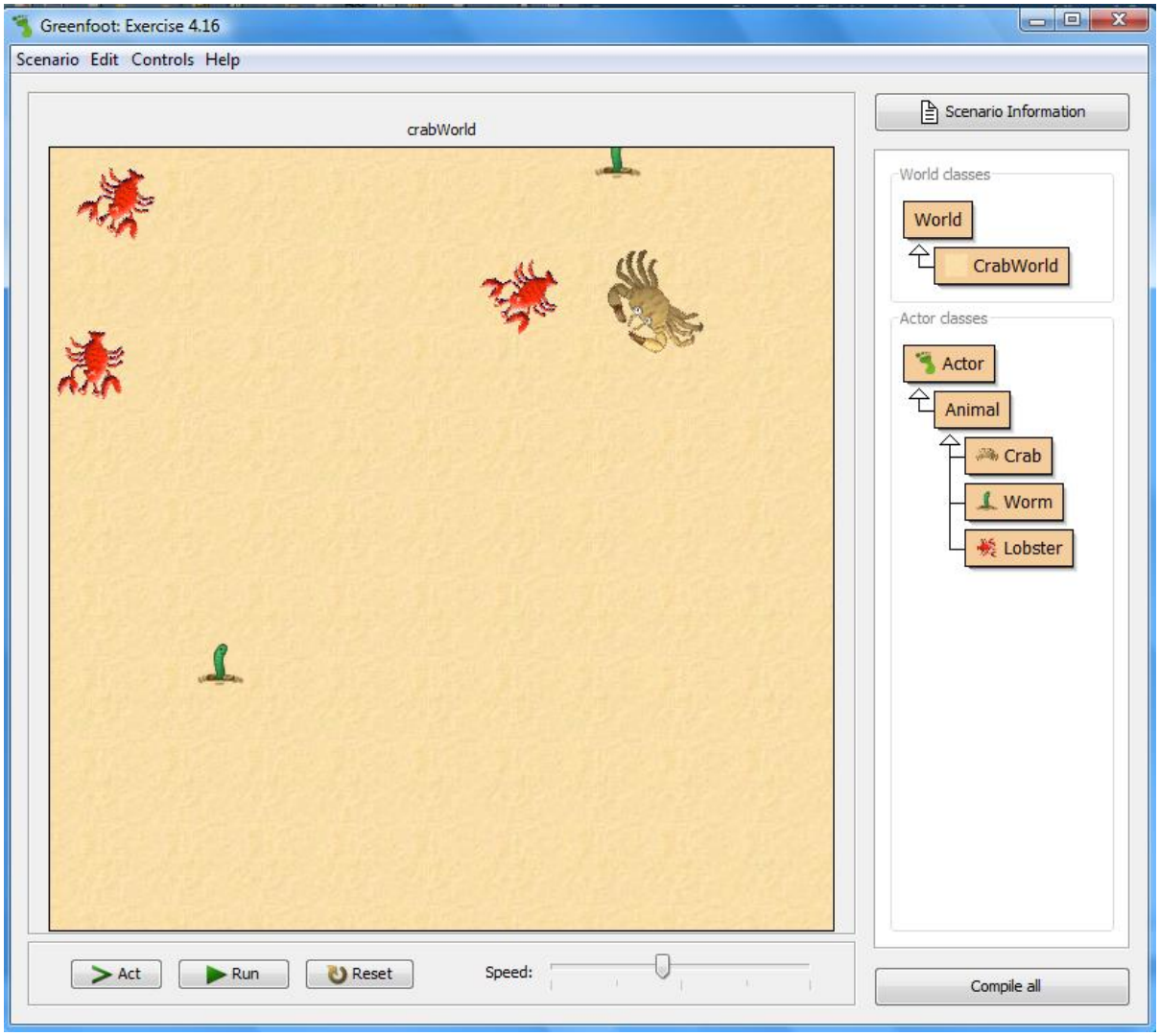
Code 4.5

```
/*
  * Check whether we have stumbled upon a worm.
   * If we have, eat it. If not, do nothing.
   */
public void lookForWorm()
{
                if ( canSee(Worm.class) )
                {
                                eat(Worm.class);
                                Greenfoot.playSound("slurp.wav");

                                wormsEaten = wormsEaten + 1;
                                if ( wormsEaten ==8)
                                {
                                Greenfoot.playSound ("fanfare.wav");
                                Greenfoot.stop();
                                }
                }
}
```

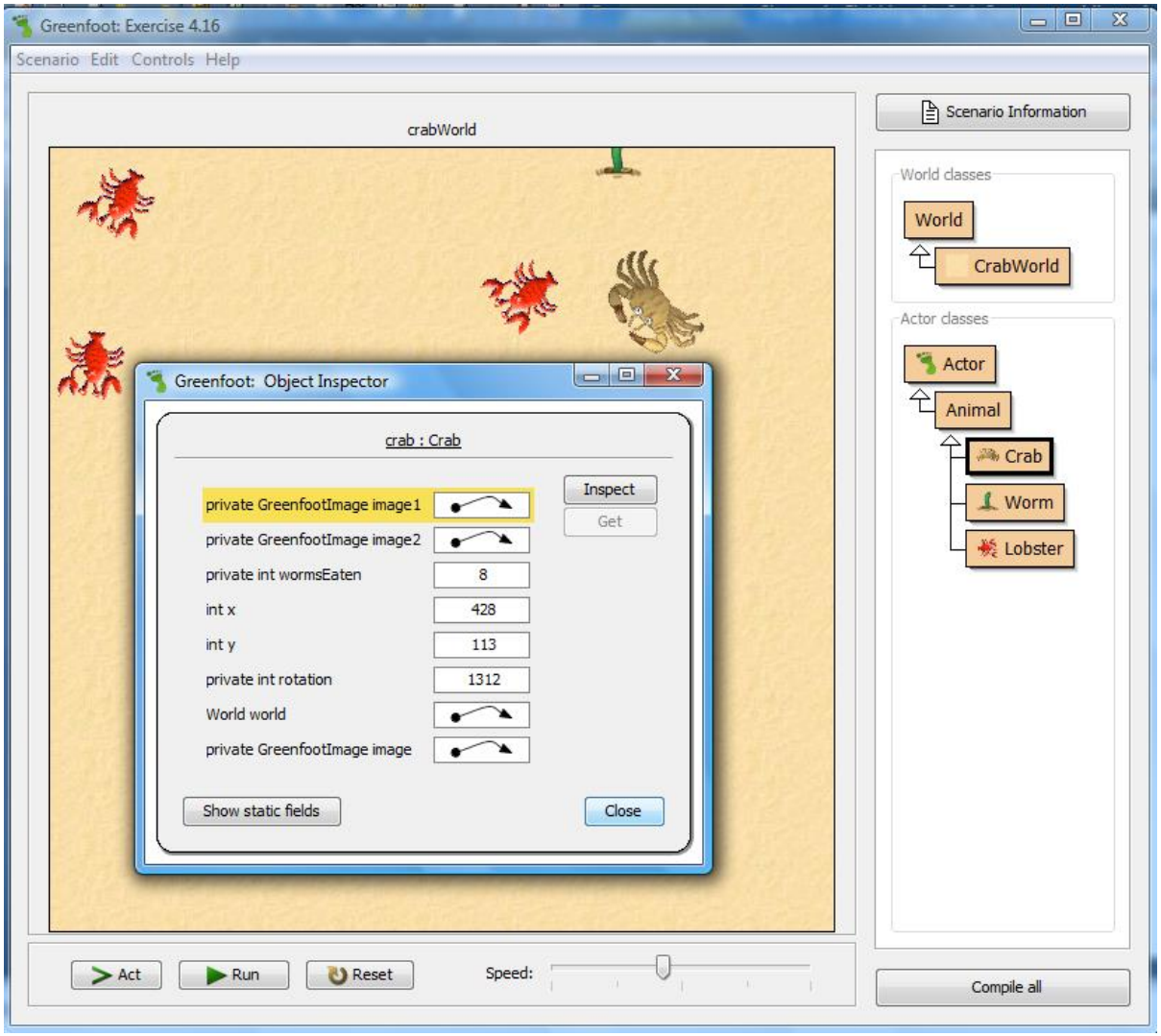
Exercise 4.16



Exercise 4.16



Exercise 4.17



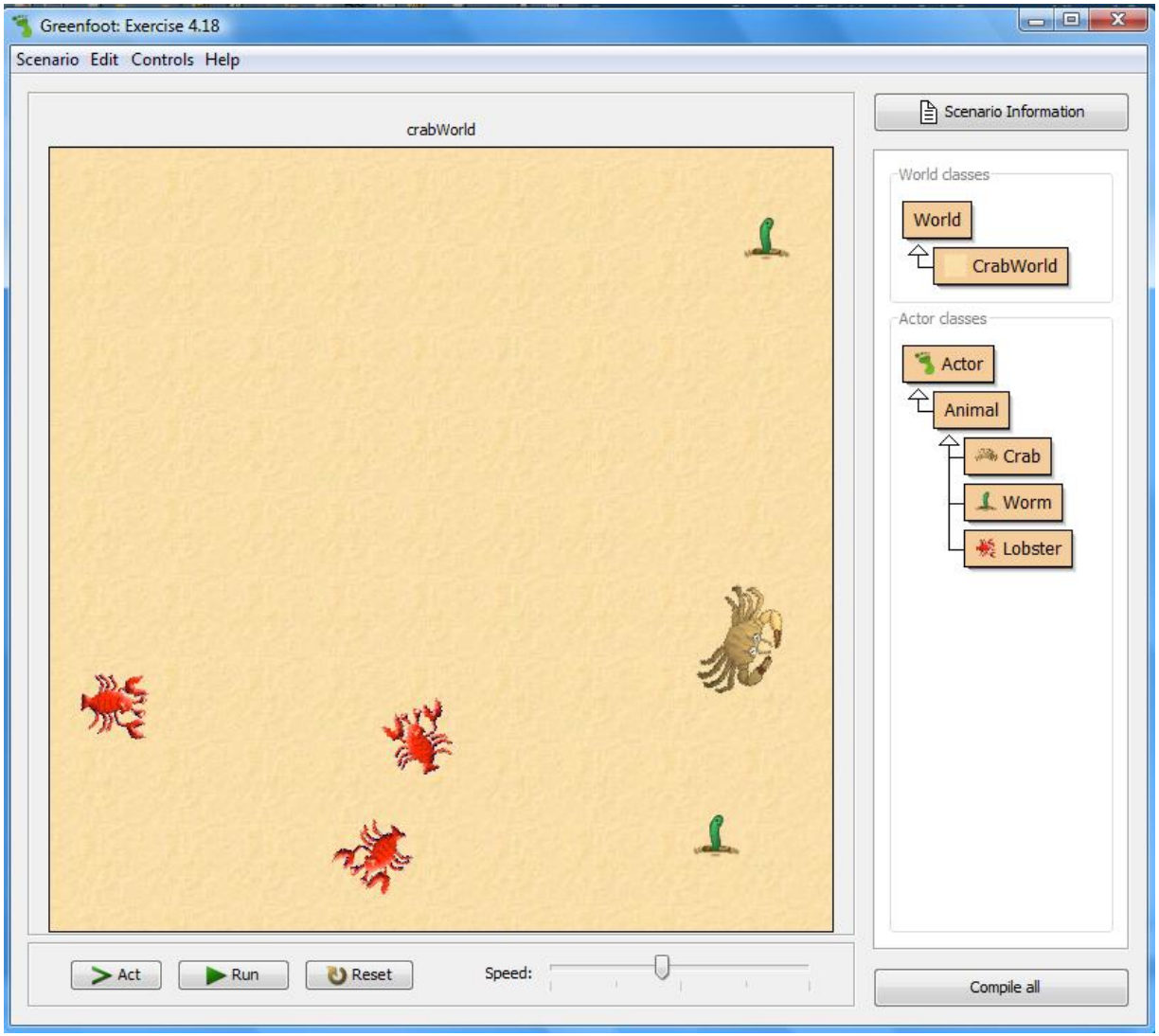
4.11 More Ideas

- Using different images for the background and the actors
- using more different kinds of actors
- not moving forward automatically, but only when the up-arrow key is pressed
- building a two-player game by interdicting a second keyboard-controlled class that listens to different keys
- making new worms pop up when one is eaten (or at random times)
- many more that you can come up with yourselves

Exercise 4.18

```
/*
 * Switch the images of the Crab to make it appear as if the Crab is moving it's legs.
 * If the current image is image1 switch it to image2 and vice versa.
 */
public void switchImage()
{
        if (timeToSwitchImages >= 3)
        {
                if (getImage() == image1)
                        setImage (image2);
                else
                        setImage (image1);
                timeToSwitchImages = 0;
        }
        else
                timeToSwitchImages++;
}
```

Exercise 4.18



4.12 Summary of Programming Techniques

In this chapter, we have seen a number of new programming concepts. We have seen how constructors can be used to initialize objects. Constructors are always executed when a new object is created.

We have seen how to use instance variables and assignment statements to store information, and how to access that information later.

Concept Summary

Concept summary

- A **constructor** of a class is a special kind of method that is executed automatically whenever a new instance is created.
- Java objects can be created programmatically (from within your code) by using the **new** keyword.
- Greenfoot actors maintain their visible image by holding an object of type **GreenfootImage**. These are stored in an instance variable inherited from class Actor.
- **Instance variables** (also called **fields**) can be used to store information (objects or values) for later use.
- An **assignment statement** assigns an object or a value to a variable.
- When an object is assigned to a variable, the variable contains a **reference** to that object.
- We can test whether two things are **equal** by using a double equals symbol: ==.
- The **if/else statement** executes a segment of code when a given condition is true, and a different segment of code when it is false.