Chapter 3 (Part 1) - Improving the Crab - More Sophisticated Programming

Original slides by Bruce Chittenden
Exercise edits by Scott Blanck

Setup: Use your modified little-crab scenario from the Chapter 2 exercises.

3.1 Adding Random Behavior

The Greenfoot Environment has a Method to get a Random Number.

```
Greenfoot.getRandomNumber (20);
```

The call would return a random number range 0 to 19

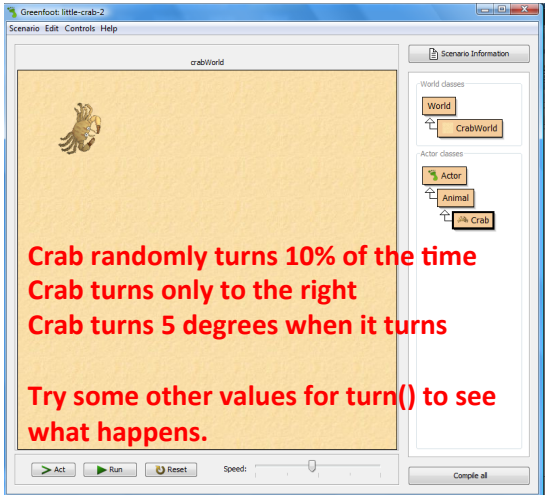
Example – If we wanted a % chance of something happening -

```
if ( Greenfoot.getRandomNumber (100) < 50)
{
        // Code Here Executes 50% of the Time
}

if ( Greenfoot.getRandomNumber (100) < 25)
{
        // Code Here Executes 25% of the Time
}
```

Exercise 3.3 – Add this code to the Crab act method. (10% chance turn)

```
public class Crab extends Animal
{
    public void act()
    {
        if ( atWorldEdge() )
        {
            turn(17);
        }

        if ( Greenfoot.getRandomNumber(100) < 10 )
        {
            turn( 5);
        }

        move();
    }
}
```

Exercise 3.3

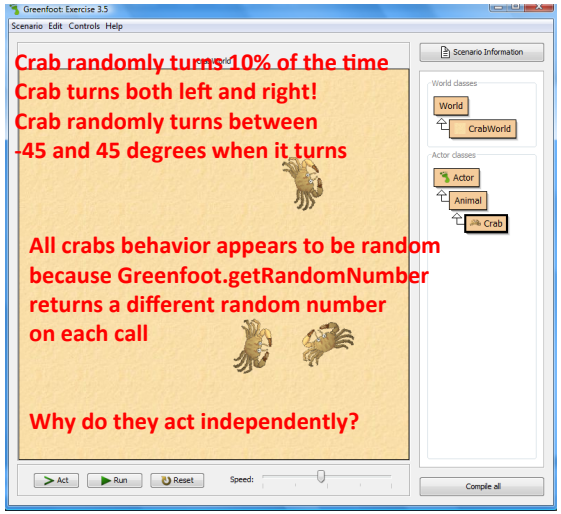


Exercise 3.4 – Add code to make the crab turn random degrees!

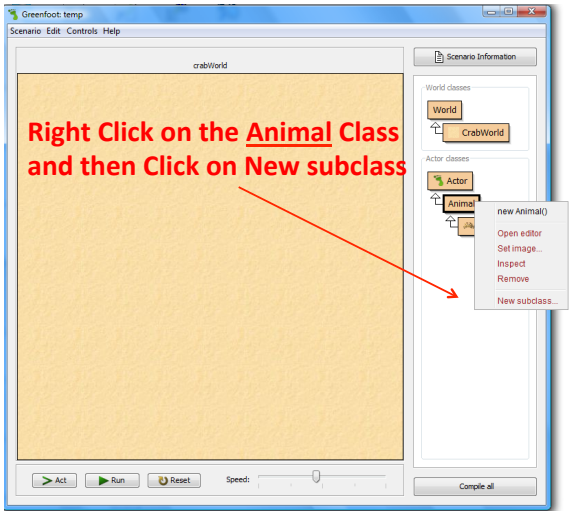
```
public void act()
{
    if ( atWorldEdge() )
    {
        turn(17);
    }

    if ( Greenfoot.getRandomNumber(100) < 10 )
    {
        turn( Greenfoot.getRandomNumber(90)-45 );
    }

    move();
}
```

- Now we are turning 10% of the time and turning a random amount of degrees.
- Try it out to see how your crab behaves!
- The crab will now turn left and right. Why?

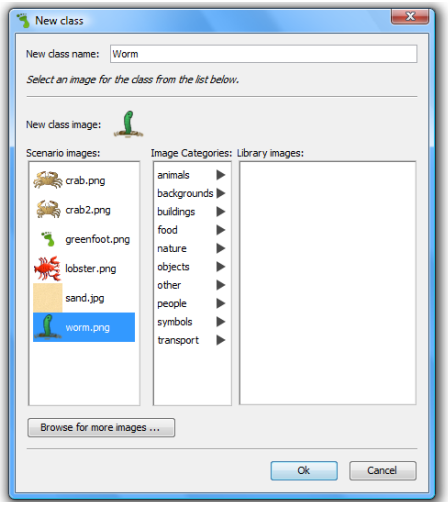
Exercise 3.6 – Add a bunch of crabs and watch their behavior.



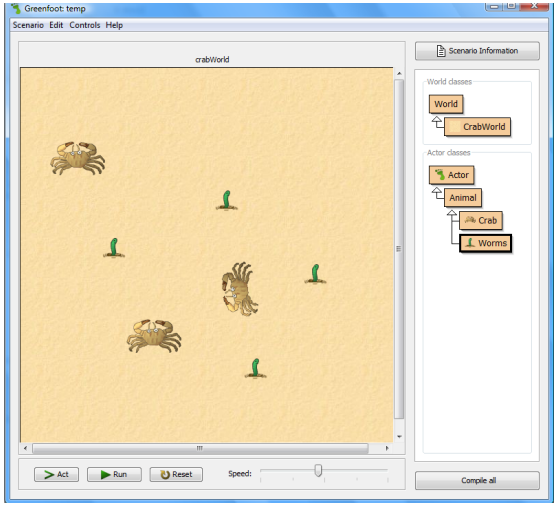
3.2 Adding Worms – Something for the crabs to eat.



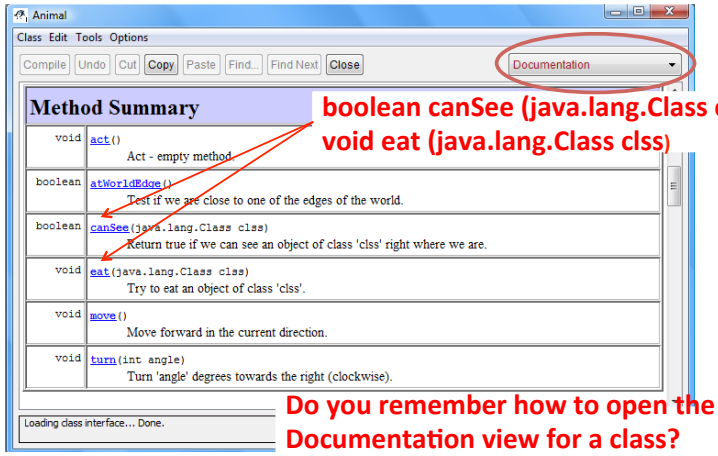
Name the class Worm (classes should be capitalized). Choose the worm graphic png file.



Exercise 3.7 – Add worms and crabs to the world. What happens?



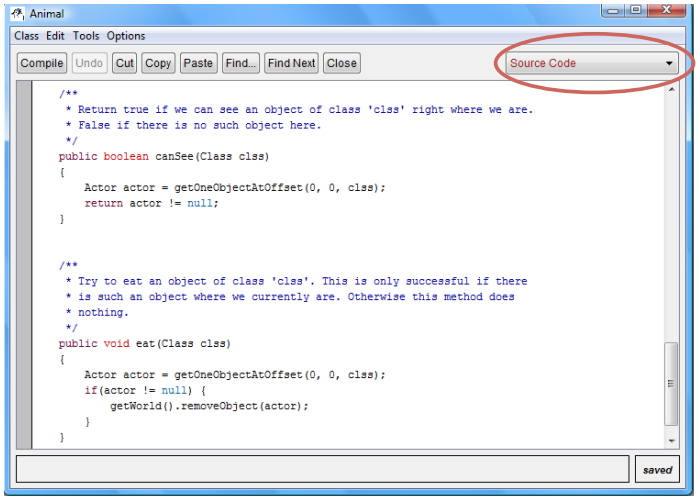
3.3 Eating Worms – Open the documentation for the Animal class.



boolean canSee (java.lang.Class clss)
void eat (java.lang.Class clss)

Do you remember how to open the Documentation view for a class? Ask a classmate for help if not.

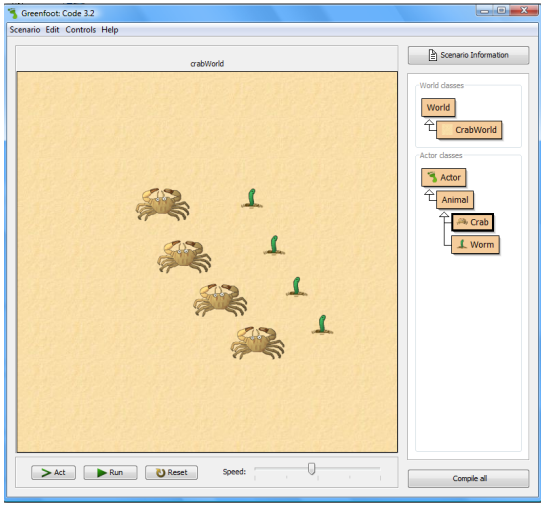
canSee and eat – Animal Source Code view instead of Documentation view



Code 3.2 – Add this code to the crab act method to see and eat worms.

```
public void act()
{
    if ( atWorldEdge() )
    {
        turn(17);
    }

    if ( Greenfoot.getRandomNumber(100) < 10 )
    {
        turn( Greenfoot.getRandomNumber(90)-45 );
    }

    move();

    if ( canSee (Worm.class) )
    {
        eat (Worm.class);
    }
}
```

Crabs Eat the Worms – Try it. What happens?



3.4 Creating New Methods – Create a new method at the bottom of the crab class.

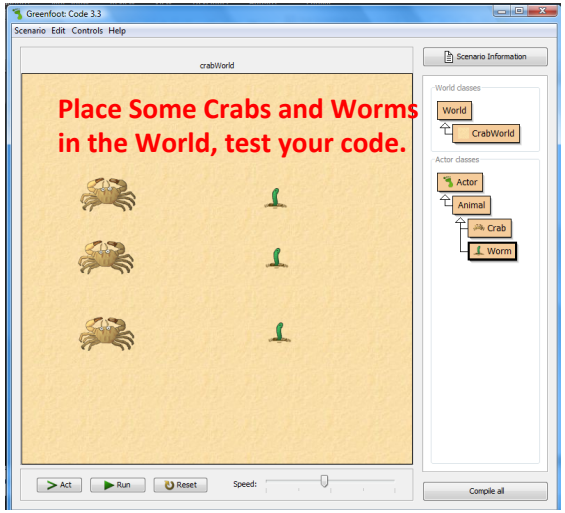
- Add all of this code to the end of the Crab Class instead of in the act method.
- Remember this needs to be inside the Crab class braces!

```
/**
 * 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);
    }
}
```

Code 3.3 – Change act to instead call your new lookForWorm method

```
public void act()
{
    if ( atWorldEdge() )
    {
        turn(17);
    }

    if ( Greenfoot.getRandomNumber(100) < 10 )
    {
        turn(5);
    }

    move();
    lookForWorm ();
}
```

Code 3.3 – Test it.



Exercise 3.8 – Do the same for randomTurn method

Move this code out of the act method and into a method called randomTurn. Do the same thing you just did with lookForWorm.

```
public void randomTurn()
{
    if ( Greenfoot.getRandomNumber(100) < 10 )
    {
        turn( Greenfoot.getRandomNumber(91)-45 );
    }
}
```

Exercise 3.9 – Do the same for the turnAtEdge method

Move this code out of the act() method and into a new method called **turnAtEdge()** by repeating the previous steps.

```
if ( atWorldEdge() )
{
    turn( 17);
}
```

Code 3.4 – This is what your final crab act method should look like.

```
public void act()
{
    turnAtEdge();
    randomTurn();
    move();
    lookForWorm();
}
```

Isn't that easier to read? That's why we moved the code out into their own methods.

Now the act method is simpler to understand because the detailed code is in the methods.