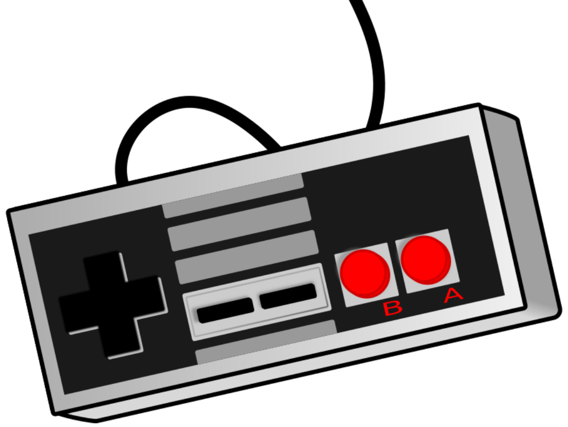


Lab_2

Little Crab

420-141-VA - GAME PROGRAMMING 1 - VANIER COLLEGE

Outline

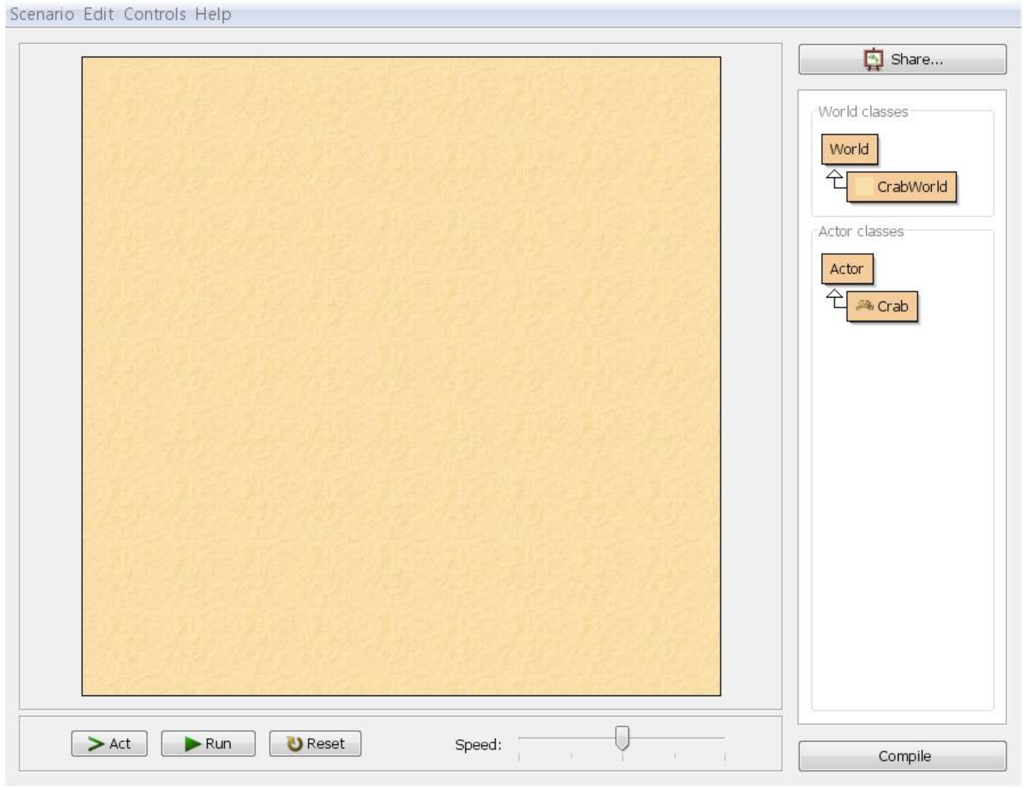
Adaptation of the Crab Tutorial from Greenfoot.org by using Stride instead of Java

- Instantiate Objects
- Implement **act**() method by invoking other methods in the **Actor** class
- Access **Greenfoot** Documentation
- Handle Keyboard Inputs using the Greenfoot.**isKeyDown**() method
- Understand the **if** statement

Step 1 - Setting up the Crab Scenario

The Crabs Scenario

- Download the **Lab_2.zip** file from Omnivox, which contains the **modern-crab** Scenario
- Unzip the contents to somewhere on your USB key or hard disk.
- Open the scenario in that location with **Greenfoot**
- You should see the standard Greenfoot interface, with an empty sandy world



Step 2 - Instantiate a Crab object in the World

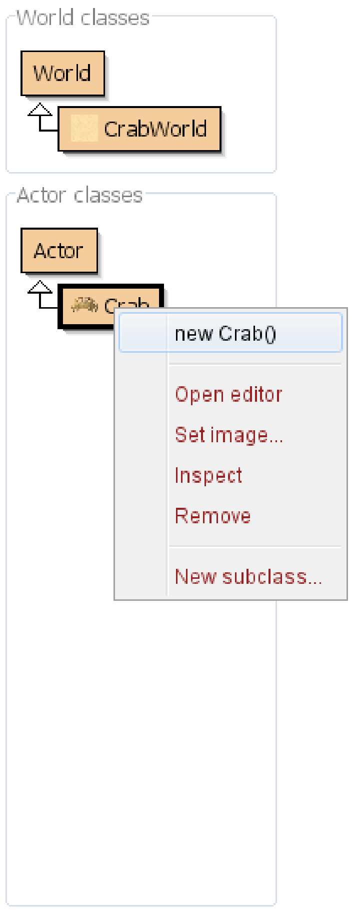
Right-click on the **Crab** class and select **new Crab()**, then left click on the world to place the crab.

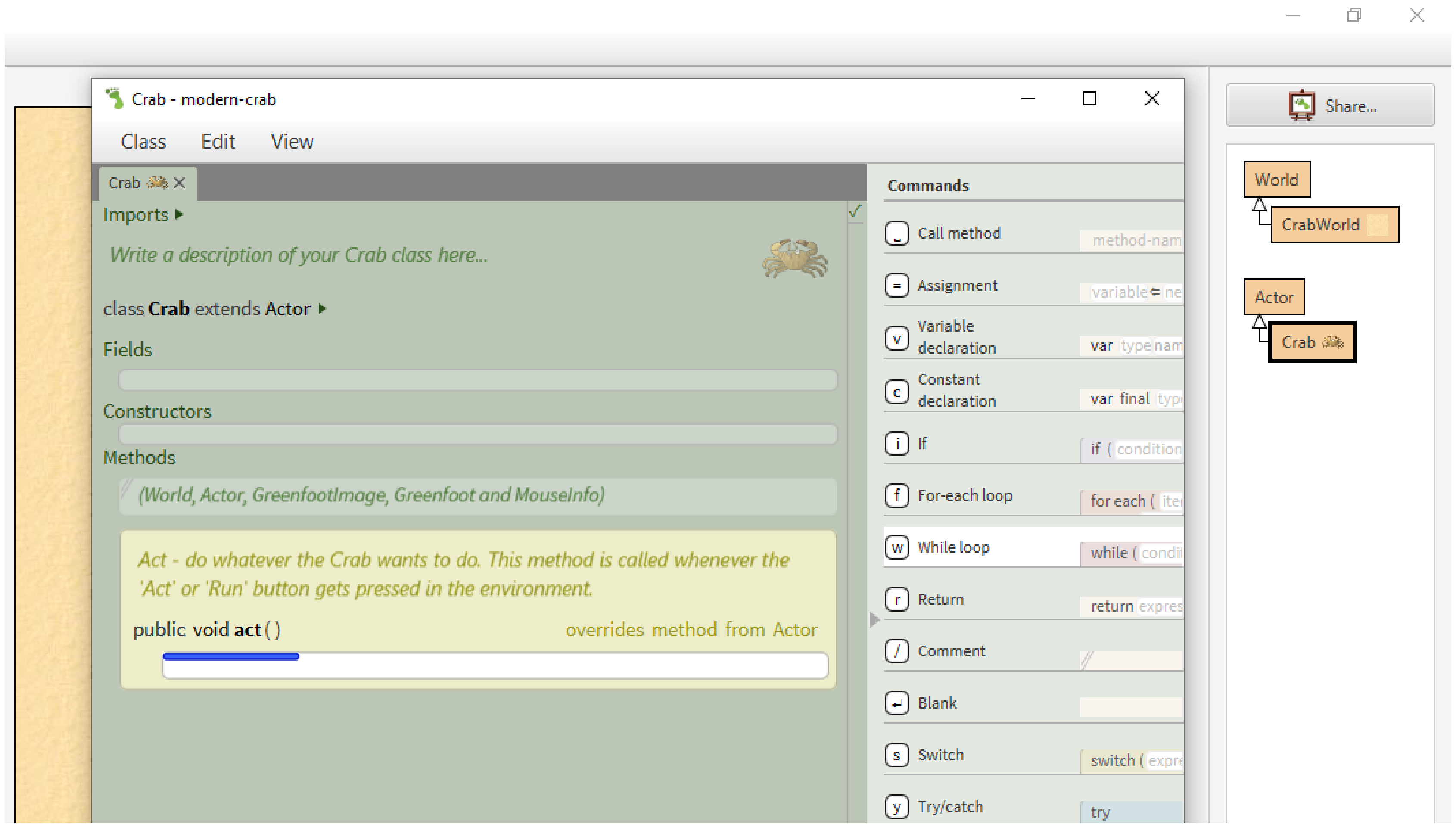
After that, click **Run**.

You might be hoping you could watch the **Crab** do an amazing dance around the screen. **Unfortunately, it seems we have a lazy Crab!**

Let's open up the code and have a look; you can double-click on the **Crab** class in the class browser.

You should see the **Stride** code in the Editor, notice the **act()** method contains nothing.





```
Imports
Write a description of your Crab class here...
class Crab extends Actor
Fields
Constructors
Methods
(World, Actor, GreenfootImage, Greenfoot and MouseInfo)
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()                         overrides method from Actor
```

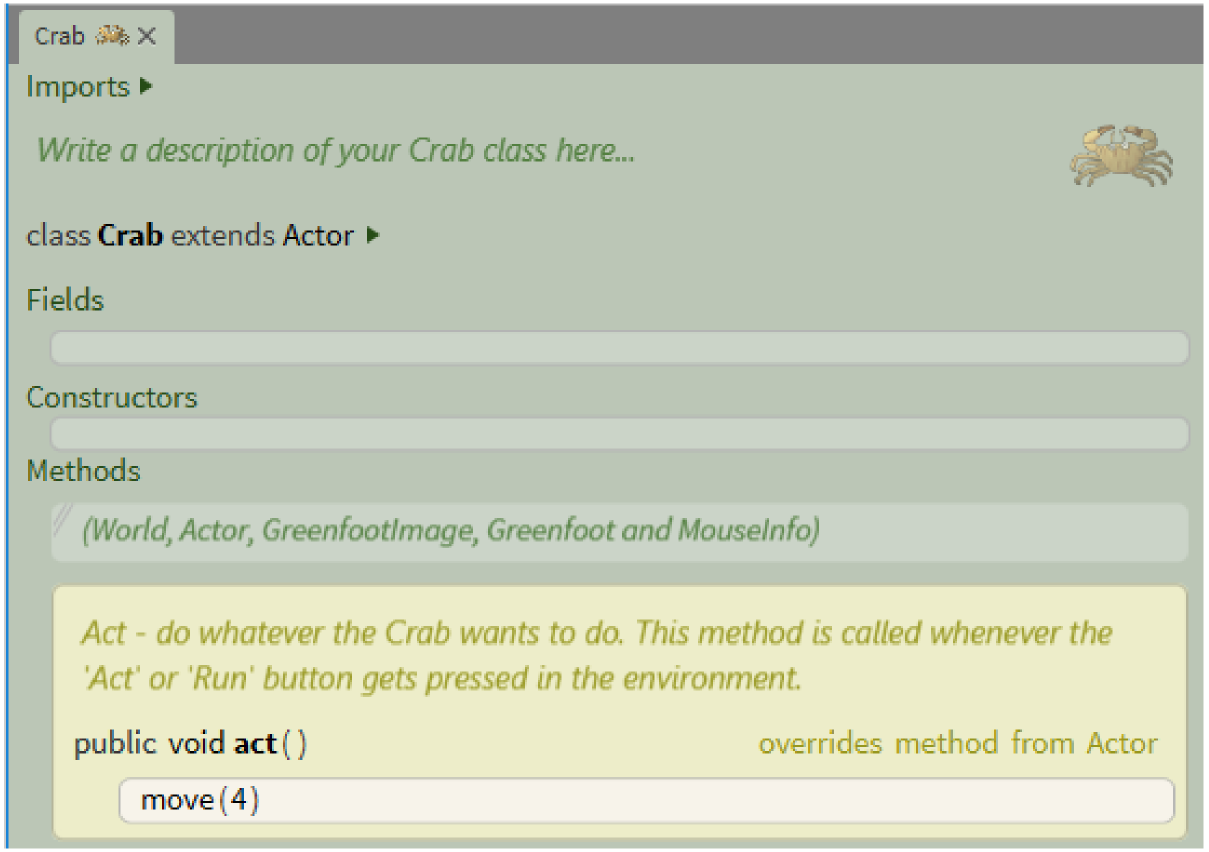
Step 3 - Making the Crab move!

If we want our crab to do anything, we must implement the **act()** method.

Let's get it moving, by adding a move instruction to the code:

- Click inside the **act()** method, you should see a blue rectangle
- Press the [SPACE] shortcut key to invoke a method, the blue rectangle should be replaced by *method-name()*
- Replace *method-name* by **move**
- Press [TAB] to enter the parameter value (**4**)

Close the code editor, hit **reset**, and instantiate a crab again, then hit **Run**!



Write a description of your Crab class here...

class **Crab** extends Actor

(World, Actor, GreenfootImage, Greenfoot and MouseInfo)

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()                    overrides method from Actor
    move(4)
```

Step 4 - Making the Crab Turn!

Open the **Crab** class again in the Editor

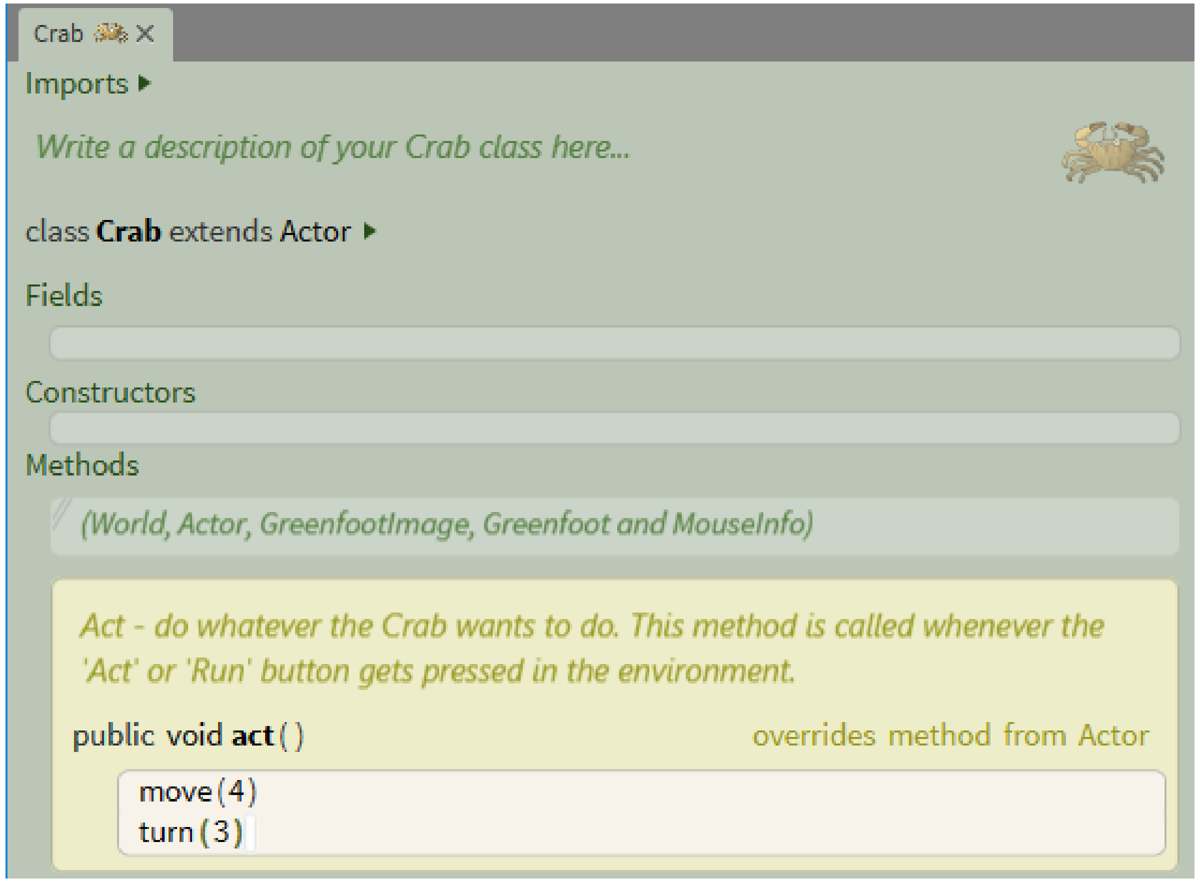
- Click below the **move(4)** statement, you should see the blue rectangle below it.
- Press [SPACE] again to invoke a method
- Type the method **turn** with parameter **(3)**

Close the editor, Hit **Reset**, instantiate another crab, and press **Run**!

Notice the crab now moves on a circle

Determine different turning amounts to get the circle:

a) **tighter**
b) **larger**


Crab ×

Imports ▸

Write a description of your Crab class here...

class **Crab** extends Actor ▸

Fields

Constructors

Methods

(World, Actor, GreenfootImage, Greenfoot and MouseInfo)

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**( )   overrides method from Actor

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

Step 5 - Greenfoot isKeyDown Documentation

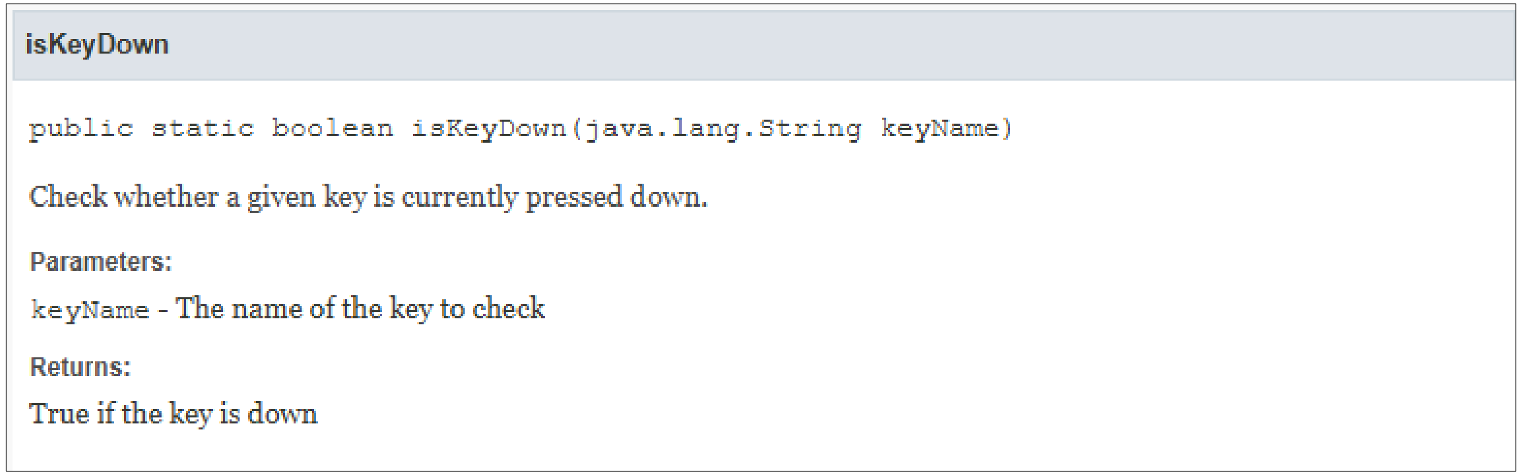
Double-click the **Actor** class to see the documentation. Click on **Package** (top-left portion)

Click on the **Greenfoot** class hyperlink to access the Greenfoot class documentation

Scroll down to see all available methods in the Greenfoot class, and click on the **isKeyDown** method hyperlink to access its documentation

From the method signature below, notice the return type is **boolean** and the parameter is a **String**

- This method checks whether a key is pressed while a scenario is running



isKeyDown

```
public static boolean isKeyDown(java.lang.String keyName)
```

Check whether a given key is currently pressed down.

Parameters:

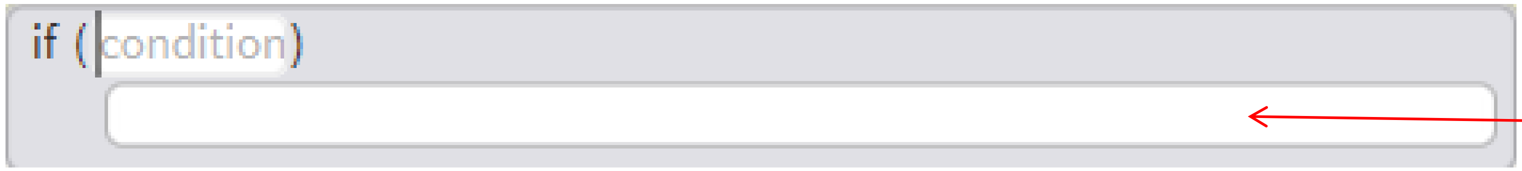
`keyName` - The name of the key to check

Returns:

True if the key is down

The IF statement

The **if** statement will contain code that will conditionally execute


```
if (condition)
```

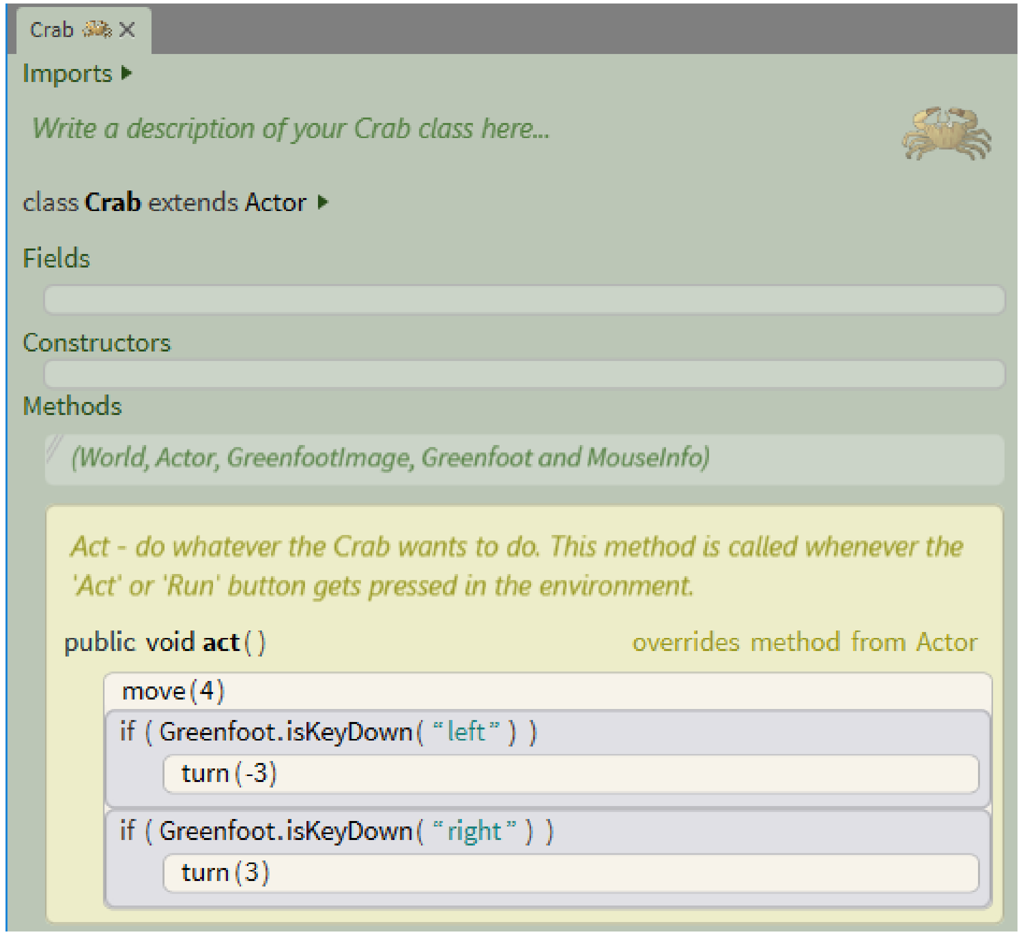
Condition: In **Stride** and **Java**, the condition is a **boolean expression**, evaluates to **true** or **false**.

- The boolean expression can be:
  - the return value of a method, for example: **Greenfoot.isKeyDown** (**"w"**)
  - a mathematical expression, for example, check if variable x is less than 5: (**x < 5**)
- When the boolean expression is:
  - **True**: the conditional code will executes
  - **False**: the conditional code will be skipped

Step 6 - Moving Crab with the Keyboard

Let's make the crab turn when we press the **left** or **right** arrow keys!

- Open **Crab** code in Editor
- Position the cursor (blue rectangle) between the **move** and **turn** statements
- Press **[i]** to insert a **if** statement
- Enter **Greenfoot.isKeyDown("left")** as the condition
- Move the **turn** statement inside the **if**
- Repeat for the **right** key, see code on the right!



Write a description of your Crab class here...

class **Crab** extends Actor

(World, Actor, GreenfootImage, Greenfoot and MouseInfo)

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()                overrides method from Actor
    move(4)
    if ( Greenfoot.isKeyDown( "left" ) )
        turn(-3)
    if ( Greenfoot.isKeyDown( "right" ) )
        turn(3)
```

Step 7 - Understand Crab game

- Put multiple crabs in the world, see how they move.
- Press **left** and **right** simultaneously, see what happens!
- Do you understand? Explain why?

Step 8 - 2 player version?

Create another type of **Actor** and implement the **act()** with similar controls, but using the keys:

- **"a"** instead of **“left”**
- **"d"** instead of **“right”**

Look at **Greenfoot** documentation to implement more advanced behaviors for your new actor (at least one new method)

Questions

?