**GridWorld Objects**

**Actor** – Actor is the superclass for Objects that we place in the GridWorld simulation. It has an act() method that is called each “step” of the simulation.

**Grid** – This is the 2D Array of Locations. By default, it has 10 rows and 10 columns. We can place various Actors (its subclasses), such as Rocks, Flowers, Bugs and Critters into different Locations within the Grid.

**Location** – This is what the Grid is composed of and has the capacity to hold a single Actor.

**Bug** – This Actor has the following behavior, which occurs during its act() method, thus each “step” of the simulation.

* If the Bug can move forward in the direction its facing, it will, leaving a Flower (that matches the Bugs color) behind the Location it just left.
* If the Bug cannot move forward, it will turn clockwise 45 degrees.
* Bugs will eat flowers, meaning they do not “block” the Bug from moving. They will not eat other Bugs, Critters or Rocks.

**Flower** – This Actor doesn’t move. It just darkens each step, until it eventually becomes black.

**Rock** – This Actor also doesn’t move. Its only purpose is to block Bug and Critters from moving through it.

**Critter** – This Actor is like a Bug, only it has a more sophisticated act() method. Each time act is called, it calls five other methods, in this order:

1. getActors() – Locates neighboring Actors to the Critter instance and puts them into an actors ArrayList.
2. processActors() – The Critter will eat (remove) all Actors that are not Rocks or Critters.
3. getMoveLocations() – Creates an ArrayList of Locations the Critter can move to (Empty and neighboring)
4. selectMoveLocation() – Choose a random Location from the ArrayList created in step 3
5. makeMove() – This method will move the Critter to the randomly selected Location in step 4

Our goal will be to create custom Actors (Bugs, Critter, etc…) that utilize the framework and provided methods to show our capacity to work in an Object oriented programming environment.

BoxBug is our first example.