Grid World Part 4 - CS 30

By Eric George

**Part 4**

**Set 7**

1. Methods implemented in the Critter class are act, getActors, processActors, getMoveLocations, selectMoveLocation, and makeMove.
2. The five basic actions made by every critter is to get the actors on the grid, process the Actors list, getting the location it can move to, selecting a location from that, then making a move.
3. Yes, if necessary the critter may get actors differently from the base class.
4. A critter can process actors by keeping the flowers too, or removing other critters, or by removing all actors from the grid.
5. The methods getMoveLocations, selectMoveLocation, and makeMove are necessary to make the critter move. In its most basic form, the critter will find empty adjacent locations with the getMoveLocations method, randomly choose one with the selectMoveLocation method, and either move to that location or remove itself from the grid with the makeMove method.
6. The class that critter extends, Actor, has a default constructor. By default, it uses the actor constructor when the critter class doesn’t have a constructor.

**Set 8**

1. This is because a ChameleonCritter overrides methods that are implemented by the act method, processActors and makeMove.
2. The makeMove method for ChameleonCritter calls super.makeMove to implement the same code.
3. This could be done by modifying the makeMove method to create a new flower object in its past location.
4. ChamaeleonCritter doesn’t override the getActors method because it does specifically the same thing for that specific method: it gets the actors of the grid in the same way as its base class.
5. The actor class contains the getLocation method.
6. A critter can access its own grid by using the getGrid method inherited from the actor class.

**Set 9**

1. This is because it wants to implement the same code from the base class Critter.
2. The crab critter only looks at the actors in the 3 cells in front of it. If an actor is in those cells but is not a rock or a critter, it will remove that actor from the grid. It cannot eat actors that aren’t in front of it or that are rocks or critters.
3. The method is used to determine what locations in the directions from the int array argument are valid adjacent locations. The method is used to determine the possible neighbours that the CrabCritter can eat.
4. The possible locations are (4,3), (4,4), (4,5).
5. A CrabCritter can only move left or right, or turn 90 degrees clockwise or counterclockwise. In contrast, a Critter has the normal full range of motion although with the same turning mechanics.
6. If the location argument for the makeMove method is the location of the CrabCritter, it chooses to move.
7. CrabCritter objects don’t eat each other because this restriction originates from the base class Critter in a method that hasn’t been overridden (processActors) that objects that are instances of the Critter class (and the Rock class) are not removable by ‘eating’.

**Group Activity (Eric)**

1. BeastCritter: the beast critter will dig underneath the grid and appear in random places anywhere the grid is occupied, and replace. Other than movement mechanics, the beast acts like a critter.
2. Needed variables: None. Needed Algorithms: algorithm to place a beast on an existing grid cell with an actor. If another actor is vying for the location, the beast critter will remove itself.
3. Done.
4. No group exchanged with.