Set 7

The source code for the Critter class is in the critters directory

1. What methods are implemented in Critter?
   Answer: the methods of act, getActors, processActors, getMoveLocations, selectMoveLocation, makeMove are implemented.
2. What are the five basic actions common to all critters when they act?
   Answer: getActors, processActors, getMoveLocations, selectMoveLocation, makeMove are the common basic actions.
3. Should subclasses of Critter override the getActors method? Explain.
   Answer: Yes. Because a new critter may selects its actor from different locations.
4. Describe the way that a critter could process actors.
   Answer: the ways are many. Etc, changing their color, eating them, or asking them to move.
5. What three methods must be invoked to make a critter move? Explain each of these methods.
   Answer: getMoveLocations—knows where can move to, selectMoveLocation-- choose one location to move, makeMove—if can move to the location, move to it.
6. Why is there no Critter constructor?
   Answer: because there is a default constructor in the superclass Actor and java can write a defualt constructor that will call super() method which calls Actor's default constructor.

Set 8

The source code for the ChameleonCritter class is in the critters directory
1. Why does act cause a ChameleonCritter to act differently from a Critter even though ChameleonCritter does not override act?
   Answer: because the methods involve in the act() method for a Critter are voerrided which makes different actions from the Critter.
2. Why does the makeMove method of ChameleonCritter call super.makeMove?
   Answer: because it behaves like a except that it need a new direction.
3. How would you make the ChameleonCritter drop flowers in its old location when it moves?
   Answer: modify the makeMove methods to a bug class that can drop flowers in the old location.
4. Why doesn't ChameleonCritter override the getActors method?
   Answer: Because it processes the same actors that a Critter does. It behaves the same as Critter and no necessaty to override this method.
5. Which class contains the getLocation method?
   Answer: the Actor class. All subclasses of Actor inherit this method.
6. How can a Critter access its own grid?
   Answer: by calling the getGrid method inherited from Actor.

Set 9

The source code for the CrabCritter class is reproduced at the end of this part of GridWorld.

1. Why doesn't CrabCritter override the processActors method?
   Answer:it needs the same actorlist as processes by a Critter, so it needn't to override this method.
2. Describe the process a CrabCritter uses to find and eat other actors. Does it always eat all neighboring actors? Explain.

Answer: it only finds the neighbors in front of it and to its half-left and to its half-right, and eats all non-rock and non-critter nerghbors immediately. All other neighbors will not be disturbed.

3. Why is the getLocationsInDirections method used in CrabCritter?

Answer:

4. If a CrabCritter has location (3, 4) and faces south, what are the possible locations for actors that are returned by a call to the getActors method?

Answer: (4, 3), (4, 4) and (4, 5).

5. What are the similarities and differences between the movements of a CrabCritter and a Critter?

Answer:

Similarities: 1. don't turn to the direction that they are moving. 2.randomly select a move location from their possible move locations.

Differences: 1. a Crab crriter only moves to left or right, while a Critter moves either of its eight neighbor locations. 2. when can't move, a CrabCritter turns while a Critter doesn't.

6. How does a CrabCritter determine when it turns instead of moving?

Answer: if the parameter loc equals to its current location, it turns.

7. Why don't the CrabCritter objects eat each other?

Answer: the processActors method of CrabCritter class is inherited from class Critter's which doesn't process rocks and critters.