Do You Know?

Set 7

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

1. What methods are implemented in Critter?

```
act
```

2. What are the five basic actions common to all critters when they act?

```
ArrayList<Actor> getActors()
void processActors(ArrayList<Actor> actors)
ArrayList<Location> getMoveLocations()
Location selectMoveLocation(ArrayList<Location> locs)
void makeMove(Location loc)
```

3. Should subclasses of Critter override the getActors method? Explain.

Yes, maybe, cuz different critter would select different strategy to eat.

4. Describe the way that a critter could process actors.

Remove actors from the grid.

5. What three methods must be invoked to make a critter move? Explain each of these methods.

```
ArrayList<Location> moveLocs = getMoveLocations();
        Location loc = selectMoveLocation(moveLocs);
        makeMove(loc);
```

6. Why is there no Critter constructor?

No parameters, and nothing need to be initialized when construct a Critter.

Do You Know?
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?
Act invoke makeMove, so when ChameleonCritter override makeMove, the behavior changes.

2. Why does the makeMove method of ChameleonCritter call super.makeMove?
Call the Critter class method makeMove

3. How would you make the ChameleonCritter drop flowers in its old location when it moves?

Add it in makeMove function.

4. Why doesn't ChameleonCritter override the getActors method?
No need.

5. Which class contains the getLocation method?
Actor

6. How can a Critter access its own grid?
getGrid()

Do You Know?

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?

It just eat, only change the getActors is ok.

2. Describe the process a CrabCritter uses to find and eat other actors. Does it always eat all neighboring actors? Explain.

No, it just change the getActors method, and get the neighboring on { Location.AHEAD, Location.HALF_LEFT, Location.HALF_RIGHT } and eat it.

3. Why is the getLocationsInDirections method used in CrabCritter?

Finds the valid adjacent locations of this critter in different directions.

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?

(4,4), (4, 3), (4, 5)

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

Both uses makeMove to specific one's movement, but CrabCritter just move horizentally.

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

setDirection(getDirection() + angle)

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

```
if (!(a instanceof Rock) && !(a instanceof Critter))
            a.removeSelfFromGrid();
```

Exercises

1. Modify the processActors method in ChameleonCritter so that if the list of actors to process is empty, the color of the ChameleonCritter will darken (like a flower).

In the following exercises, your first step should be to decide which of the five methods--~~getActors, processActors, getMoveLocations, selectMoveLocation, and makeMove~~-- should be changed to get the desired result.

2. Create a class called ChameleonKid that extends ChameleonCritter as modified in exercise 1. A ChameleonKid changes its color to the color of one of the actors immediately in front or behind. If there is no actor in either of these locations, then the ChameleonKid darkens like the modified ChameleonCritter.

3. Create a class called RockHound that extends Critter. A RockHound gets the actors to be processed in the same way as a Critter. It removes any rocks in that list from the grid. A RockHound moves like a Critter.

4. Create a class BlusterCritter that extends Critter. A BlusterCritter looks at all of the neighbors within two steps of its current location. (For a BlusterCritter not near an edge, this includes 24 locations). It counts the number of critters in those locations. If there are fewer than c critters, the BlusterCritter's color gets brighter (color values increase). If there are c or more critters, the BlusterCritter's color darkens (color values decrease). Here, c is a value that indicates the courage of the critter. It should be set in the constructor.

5. Create a class QuickCrab that extends CrabCritter. A QuickCrab processes actors the same way a CrabCritter does. A QuickCrab moves to one of the two locations, randomly selected, that are two spaces to its right or left, if that location and the intervening location are both empty. Otherwise, a QuickCrab moves like a CrabCritter.

6. Create a class KingCrab that extends CrabCritter. A KingCrab gets the actors to be processed in the same way a CrabCritter does. A KingCrab causes each actor that it processes to move one location further away from the KingCrab. If the actor cannot move away, the KingCrab removes it from the grid. When the KingCrab has completed processing the actors, it moves like a CrabCritter.