Part 4

Set 7

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

1. What methods are implemented in Critter?

A: act, getActors, processActors, getMoveLocations, selectMoveLocation, makeMove

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

A: getActors, processActors, getMoveLocations, selectMoveLocation, makeMove

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

A: Yes. If a new subclass need to get actors form other location, it should override the getActor method.

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

A: Different critters do different things. The critter can eat them, just change their color, and so on.

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

A: a. getMoveLocations

This method gets all locations that the critter can move to.

b.selectMoveLocation

This method selects one of the locations got by getMoveLocations method.

c.makeMove

This method make critter move to a location by passing a param.

6. Why is there no Critter constructor?

A: Because Critter extends Actor, the default construction of Critter is the construction of Actor if there is no construction in the Critter class.

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?
A: Act method contains five methods, which are getActors, processActors, getMoveLocations, selectMoveLocation, and canMove sequentially. Although ChameleonCritter act differently from a Critter, the process is identical. Therefore, we only need to override processActors and canMove methods.

2. Why does the makeMove method of ChameleonCritter call super.makeMove?
A: Because the difference between ChameleonCritter's makeMove and Critter's canMove is only that ChameleonCritter changes its direction after moving. Therefore, it can call super.makeMove() to call Critter's canMove method and then add some statements to achieve its special behavior.

3. How would you make the ChameleonCritter drop flowers in its old location when it moves?
A:
```
public void makeMove(Location loc) {
      Location lastLoc = getLocation();
      super.makeMove(loc);
      if(!oldLoc.equals(loc)) {
            Flower flower = new Flower(getColor());
            flower.putSelfInGrid(getGrid(), lastLoc);
      }
      setDirection(lastLoc.getDirectionToward(loc));
  }
```

4. Why doesn't ChameleonCritter override the getActors method?
A: Because the way ChameleonCritter get actors is the same as Critter.

5. Which class contains the getLocation method?
A: Actor

6. How can a Critter access its own grid?
A: By calling getGrid method which is implements in Actor class.

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?

A: Because the way CrabCritter process actors is different from Critter, so it override it to determine its own way to process actors.

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

A: A CrabCritter looks for other actors whose location immediately in front, to the right-front, or to the left-front of it. CrabCritter doesn't eat all neighboring actors, it only eat the actors found by getActors method overrode by CrabCritter.

3. Why is the getLocationsInDirections method used in CrabCritter?

A: Because CrabCritter need to move or get other actors according to directions, so it call getLocationsInDirections method instead of calling getNeighbors method which is declared by Grid.

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?

A: (4, 3), (4, 4), (4, 5)

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

A: Similarities: CrabCritter and Critter move to a random locations that is possible for them to move to. And they never trun direction to the location to which they will move.

Differences: CrabCritter can only move to right or left location, while Critter can move to its eight adjacent locations. And when there is no location accessable, CrabCritter will trun to right or left randomly, but Critter will do nothing.

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

A: If the location passed as param is equal to its current location, the CrabCritter will turn.

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

A: Because CrabCritter inherits the processActors method from the Critter, in which determines that Critter cannot eat each other.