**Set 7**

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

1. What methods are implemented in Critter?

Answer:

public void act()

public ArrayList<Actor> getActors()

public void processActors(ArrayList<Actor> actors)

public ArrayList<Location> getMoveLocations()

public Location selectMoveLocation(ArrayList<Location> locs)

public void makeMove(Location loc)

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

Answer:getActors(),processActors(),getMoveLocations(),selectMoveLocation() and makeMove()

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

Answer: Yes, when we want our own critter move differently from the Critter, we can override the getActors method.

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

Answer: It could eat all of the actors in its list, it could makethem all change colors, or it could ask them all to move.

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

Answer: public ArrayList<Location> getMoveLocations()

public Location selectMoveLocation(ArrayList<Location> locs)

public void makeMove(Location loc)

getMoveLocations method returns a list of all the empty adjacent locations around the critter.

the selectMoveLocation

randomly chooses one of the locations and returns that location. If there are no empty locations to choose from, selectMoveLocation returns the current location of the critter.

the makeMove method

and the critter is moved to the new location

6. Why is there no Critter constructor?

Answer: Critter extends Actor.  The Actor class has a default constructor. If you do not create a constructor in a class, Java will write a default constructor for you. The Critter default constructor that Java provides will call super(), which calls the Actor default constructor.

**SET8**

1. Why does act cause a ChameleonCritter to act differently from a Critter even though ChameleonCritter does not override act?

Answer:

The act method calls getActors, processActors, getMoveLocations, selectMoveLocation, and makeMove.  The ChameleonCritter class overrides the processActors and makeMove

methods. Therefore, calling act for a ChameleonCritter will produce different behavior than calling act for a Critter.

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

Answer:

it calls super.makeMove of the Critter class to actually move to the new location.   
3. How would you make the ChameleonCritter drop flowers in its old location when it moves?

Answer:

public void makeMove(Location loc)

{

  Location oldLoc = getLocation();

  setDirection(getLocation().getDirectionToward(loc));

  super.makeMove(loc);

  if(!oldLoc.equals(loc)) //don't replace yourself if you did not move

  {

    Flower flo = new Flower(getColor());

    flo.putSelfInGrid(getGrid(), oldLoc);

  }

}

4. Why doesn't ChameleonCritter override the getActors method?

Answer:

 Since ChameleonCritter does not define a new behavior for getActors, it does not need to override this method.

5. Which class contains the getLocation method?

Answer: Actor class

6. How can a Critter access its own grid?

Answer: getGrid()

**SET9**

1. Why doesn't CrabCritter override the processActors method?

Answer: There is no need 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:

The CrabCritter’s getActors method only looks for neighbors that are immediately in front of the crab critter and to its right-front and left-front locations. Any neighbors found in these locations will be “eaten” when the processActors method is called. Actors in the other neighboring locations will not be disturbed.

3. Why is the getLocationsInDirections method used in CrabCritter?

Answer:

The parameter for this method brings in an array of directions. For the crab critter, this array contains the directions of the possible neighbors that this crab can eat. The method getLocationsInDirections uses this array to determine and return valid adjacent locations of this critter in the directions given by the array parameter.

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?

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

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