1.

ArrayList<Actor> actors = getActors();

processActors(actors);

ArrayList<Location> moveLocs = getMoveLocations();

Location loc = selectMoveLocation(moveLocs);

makeMove(loc);

2.

getActors, processActors, getMoveLocations, selectMoveLocation, makeMove

//@file:info/gridworld/actor/Critter.java

//@line:42~46

ArrayList<Actor> actors = getActors();

processActors(actors);

ArrayList<Location> moveLocs = getMoveLocations();

Location loc = selectMoveLocation(moveLocs);

makeMove(loc);

3.

Yes.If the subclasses of Critter select different the actors in locations from the Critter class, then it must override the getActor() method.

4.

A critter may could eat the actors, or it could causes the actors to move away(the KingCrab).

5.

\*\*getMoveLocations, selectMoveLocation, makeMove.\*\*

The getMoveLocations() method returns all the empty and adjacent locations of the critter:

//@file:info/gridworld/actor/Critter.java

//@line:90

return getGrid().getEmptyAdjacentLocations(getLocation());

The selectMoveLocation() method returns a randomly selected location of the locations get from the getMoveLocations() method:

//@file:info/gridworld/actor/Critter.java

//@line:109~110

int r = (int) (Math.random() \* n);

return locs.get(r);

The makeMove() method make the critter move to the location selected from the selectMoveLocation() method:

//@file:info/gridworld/actor/Critter.java

//@line:130

moveTo(loc);

6.

Because the Critter class extends the Actor class and the Actor class has a constructor which does't have parameter.The Critter class will have a default constructor which calls super() method.

//@file:info/gridworld/actor/Actor.java

//@line:39

public Actor()

7.

Because the ChameleonCritter class \*\*override the processActors() method and the makeMove() method\*\*.Both of the two methods will affect the act() method, so a ChameleonCritter will act differently from a Critter.

//@file:projects/critters/ChameleonCritter.java

//@line:36 and 50

public void processActors(ArrayList<Actor> actors)

public void makeMove(Location loc)

8.

Because when a ChameleonCritter moves, it turns toward the new location.To achieve this, we just need to set it's direction in the makeMove() method, and then it can act like a Critter.

//@file:projects/critters/ChameleonCritter.java

//@line:52

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

9.

We can write the code just like in the Bug class, and we need to rewrite the makeMove() method.

public void makeMove(Location loc)

{

Location oldloc = getLocation();

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

super.makeMove(loc);

if (oldloc != loc) {

Flower flower = new Flower(getColor());

flower.putSelfInGrid(getGrid(), oldloc);

}

}

//@file:info/gridworld/grid/actor/Bug.java

//@line:82~83

Flower flower = new Flower(getColor());

flower.putSelfInGrid(gr, loc);

10.

Because the ChameleonCritter class \*\*get the same neightborings actor as a Critter.\*\*(From the github, Part4).Since the ChameleonCritter can get the actors as a Critter, it does't need to override the getActors() method.

11.

Since the \*\*Actor\*\* class contains the getLocation() method, then all the subclasses that extends the Actor class will contain the get Location() method.

//@file:info/gridworld/actor/Actor

//@line:104

return location;

12.

It can use the \*\*getGrid()\*\* method(e.g.\*\*critter.getGrid()\*\*), though it does't implement the method, but the Actor class implements it, so the Critter class inherits the getGrid() method from the Actor class.

//@file:info/gridworld/actor/Actor.java

//@line:94

return grid;

13.

Because the CrabCritter eats whatever the actors it find just as the Critter(From github, Part4), so there is no need for the CrabCritter class to override the processActors() method, it can use the method of Cirtter class directly.

//@file:info/gridworld/actor/Critter.java

//@line:73~76

for (Actor a : actors)

{

if (!(a instanceof Rock) && !(a instanceof Critter))

a.removeSelfFromGrid();

14.

First it find the actors in the locations immediately in front, to the right-front, or to the left-front of it, (From github, Part4)，then it will eat the actors that are not rock and Critters.So it won't eat all the neighboring actors.

//@file://projects/critters/CrabCritter.java

//@line:47~54

int[] dirs =

{ Location.AHEAD, Location.HALF\_LEFT, Location.HALF\_RIGHT };

for (Location loc : getLocationsInDirections(dirs))

{

Actor a = getGrid().get(loc);

if (a != null)

actors.add(a);

}

15.

Because the CrabCritter need this method to find the locations immediately in front, to the right-front, or to the left-front of it(from github, Part4), then it can get the actors in this locations.

//@file:projects/critters/CrabCritter.java

//@line:107~112

for (int d : directions)

{

Location neighborLoc = loc.getAdjacentLocation(getDirection() + d);

if (gr.isValid(neighborLoc))

locs.add(neighborLoc);

}

16.

Accoding to the getLocationsInDirections() method, it will find the locations immediately in front, to the right-front, or to the left-front of it., so the locations will ba (4,4), (4,3), (4,5)

//@file:projects/critters/CrabCritter.java

//@line:107~112

for (int d : directions)

{

Location neighborLoc = loc.getAdjacentLocation(getDirection() + d);

if (gr.isValid(neighborLoc))

locs.add(neighborLoc);

}

17.

Similarities:Both of them select a location to move to randomly(\*\*since the CrabCritter doesn't override the selectMoveLocation\*\*) from their valid locations.

//@file://projects/critters/CrabCritter.java

Differences:The CrabCritter can only move to it's right or left while the Critter doesn't have this limit. The CrabCritter will turn to it's right or left while the Critter won't.

//@file://projects/critters/CrabCritter.java

//@line:65~67 and 79~88

int[] dirs =

{ Location.LEFT, Location.RIGHT };

for (Location loc : getLocationsInDirections(dirs))

if (loc.equals(getLocation()))

{

double r = Math.random();

int angle;

if (r < 0.5)

angle = Location.LEFT;

else

angle = Location.RIGHT;

setDirection(getDirection() + angle);

}

18.

When it's current location equals the location it will move to , it will turn.

//@file:projects/critters/CrabCritter.java

//@line:79

if (loc.equals(getLocation()))

19.

Because the CrabCritter class doesn't override the processActors() method, since the Critter won't eat a rock and a Critter, and the CrabCritter is a Critter, so the CrabCritter objects won't eat each other.

//@file:info/gridworld/actor/Critter.java

//@line:75

if (!(a instanceof Rock) && !(a instanceof Critter))