Part 3

Set 3

Assume the following statements when answering the following questions.

Location loc1 = new Location(4, 3);

Location loc2 = new Location(3, 4);

1.How would you access the row value for loc1?

A: loc1.getRow();

2. What is the value of b after the following statement is executed?

    boolean b = loc1.equals(loc2);

A: b = false;

3. What is the value of loc3 after the following statement is executed?

    Location loc3 = loc2.getAdjacentLocation(Location.SOUTH);

A: (4,4)

4. What is the value of dir after the following statement is executed?

    int dir = loc1.getDirectionToward(new Location(6, 5));

A:Southeast.

5. How does the getAdjacentLocation method know which adjacent location to return?

A:The param of the method indicates the direction in which the method finds the closest location to return.

Set 4

1. How can you obtain a count of the objects in a grid? How can you obtain a count of the empty locations in a bounded grid?

A: get the count of the objects in a grid: grid.getOccupiedLocations().size();

get the count of the empty locations: grid.getNumRows()*grid.getNumCols() - grid.getOccupiedLocations().size()

2. How can you check if location (10,10) is in a grid?

A: grid.isValid(new Location(10, 10));

3. Grid contains method declarations, but no code is supplied in the methods. Why? Where can you find the implementations of these methods?

A: Grid is an interface, so it only need to declare some method which will be implemented by the class who implements Grid.

I can find that AbstractGrid who is an abstract class that implements some of methods declared in Grid. In addition, BoundedGrid and UnboundedGrid that extend AbstractGrid implement the rest method.

4. All methods that return multiple objects return them in an ArrayList. Do you think it would be a better design to return the objects in an array? Explain your answer.

A: I don't think so. If we use array instead of ArrayList, it might be more quickly to use an index to find an object than using get(i) method, but we have to calculate the count of objects before we new an array. However, ArrayList makes it easy, because ArrayList doesn't have a constant size.

Set 5

1. Name three properties of every actor.

A: location, direction and color.

2. When an actor is constructed, what is its direction and color?

A: The original color is blue and the original direction is North.

3. Why do you think that the Actor class was created as a class instead of an interface?

A: Because all Actors has common states and methods. If Actor is created as an interface, it cannot contain any instance and also cannot implement any method.

4. (a)Can an actor put itself into a grid twice without first removing itself? (b)Can an actor remove itself from a grid twice? (c)Can an actor be placed into a grid, remove itself, and then put itself back? Try it out. What happens?

A: (a)(b)No, if you do them, an IllegalStateException will be thrown.

(c)Yes, It is valid.

5. How can an actor turn 90 degrees to the right?

A: There are two ways to do it. `actor.setDirection(actor.getDirection() + 90)` or `actor.setDirection(actor.getDirection() + Location.RIGHT)`

If the actor is a Bug, you can call `turn()` twice.

Set 6

1. Which statement(s) in the canMove method ensures that a bug does not try to move out of its grid?

A:
```
if(!gr.isValid(next))
    return false;
```

2. Which statement(s) in the canMove method determines that a bug will not walk into a rock?

A:
```
Actor neighbor = gr.get(next);
return (neighbor == null) || (neighbor instanceof Flower);
```

3. Which methods of the Grid interface are invoked by the canMove method and why?

A:
```
if (!gr.isValid(next))
    return false;
Actor neighbor = gr.get(next);
```

isValid() and get() methods are invoked by the canMove. They are used to make sure whether the location of next step is valid and empty or located a flower.

4. Which method of the Location class is invoked by the canMove method and why?

A: getAdjacentLocation(). Because we need to get the location of next step and judge if the bug can move to it.

5. Which methods inherited from the Actor class are invoked in the canMove method?

A: getLocation(), getDirection(), getGrid().

6. What happens in the move method when the location immediately in front of the bug is out of the grid?

A: The bug will remove itself.

7. Is the variable loc needed in the move method, or could it be avoided by calling getLocation() multiple times?

A: Yes, it is. Loc stores the location of the Bug, it can be used to insert a flower in the Bug's original location after moving.

8. Why do you think the flowers that are dropped by a bug have the same color as the bug?

A:
```
Flower flower = new Flower(getColor());
```

This statement determined that the color of the flower dropped by a bug have the same color as the bug.

9. When a bug removes itself from the grid, will it place a flower into its previous location?

A: If the method removeSelfFromGrid() is called alone, it won't drop a flower. However, if the method is called in the move() method, a flower will be placed into its previous location.

10. Which statement(s) in the move method places the flower into the grid at the bug's previous location?

A:
```
Flower flower = new Flower(getColor());
flower.putSelfInGrid(gr, loc);
```

11. If a bug needs to turn 180 degrees, how many times should it call the turn method?

A: 4 times.