The Answer of Part3

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?

   Ans: Use the method getRow().

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

```
boolean b = loc1.equals(loc2);
```

   Ans: The value of b is false.

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

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

   Ans: loc3 = Location(4, 4).

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

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

  Ans: The value of dir is 135.

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

  Ans: This method get the adjacent location according to the direction parameter.

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?

   Ans: int objNum = grid.getOcuupiedLocations().size();

        int emptyNum = grid.getNumRows() *

grid.getNumCols() - objNum;

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

   Ans: If the grid is unbounded, it is in the grid. If not, there must exist the number of rows and columns of the grid. Use the method getRow() and getCol() to get the row and col of the location, which are 10 and 10. Then only if one of the two attribute is not smaller than the corresponded row and col of the grid, the location is not in the grid.

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

   Because Grid is a abstract class, whose methods are implemented by its extended classes. So the implementations can be founded in the extended classes of Grid.

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.

   Ans: No, array must defined the length, and the length is fixed. Instead, ArrayList can be extended easily. Usually we can not know how mant objects will be returned before the function executes, so using ArrayList is better.

Set 5

1. Name three properties of every actor.

   Ans: The location, direction and the color of the actor.

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

   Ans: The direction is north(0) and the color is blue.

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

Because the actor is a kind of object, which has its own properties, not only methods.

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

   Ans: An actor can put itself into a grid twice because when it puts itself on the second time, it will removes itself and put itself again. But an actor can not remove itself from a grid twice, for the reason that in the second remove procedure, the location has not object, which will cause an IllegalException. The actor can be placed into a grid, remove itself and the put itself back.

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

Ans: Add Location.RIGHT to the direction attribute of the actor.

Set 6

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

   Ans: The statement if (!gr.isValid(next)) ensures the moving is within the grid.

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

   Ans: return (neighbor ==null) || (neighbor instance of Flower).

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

   Ans: isValid() method and get method, because only the Grid can judge whether the location is valid for itself so that the actor can move and we should know whether there is an object on the destination according to the result of get(next).

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

   Ans: getAdjacentLocation(), because we need to know the next place the bug will move to.

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

   Ans: getLocation(), getGrid() and getDirection().

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

   Ans: The bug removes itself from the grid.

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

   Ans: The loc is not needed, but use it will increase the effciency.

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

   Ans: Because in the move() method, the color parameter of the constructed function of flower is get from the method getColor(), which returns the bug’s color.

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

   Ans: When the remove method is called by the move method, the answer is yes.

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

    Ans: 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?

Ans: Four times.