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?

Answer: int rowValue = loc1.getRow();

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

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

Answer: b = false;

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

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

Answer: (4, 4)

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

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

Answer: 135 degree.

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

Answer: it knows the adjacent location by being called by a location instance using one of the eight compass directions, thus it can get the adjacent loaction through calculation.

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?

Answer: use the getValidAdjacentLocations(Location loc) method.

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

Answer: 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?

Answer: because Grid is an interface class and no need to implement it. The implementations of these methods are in class BoundedGrid and class UnboundedGrid.

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.

Answer: NO, IT WON'T BE BETTER! Although using an array is more efficient, but its size can't be change. So using an ArrayList is more flexible and can reduce potential bugs.

Set 5

1. Name three properties of every actor.

Answer: color, direction, location.

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

Answer: it has default color of blue and default direction of Location.NORTH when it is constructed.

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

Answer: because some of its methods have implementations. If the Actor class is an interface, there is no necessity to do this.

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?

Answer:

No, it can't. When the first case happens, the drop-down menu will not show the method of putting an actor at the location.

No, it can't. When the second case happens, the drop-down menu will not show the method of removing an actor at the location.

Yes, it can. When the third case happens, the drop-down menu will first show the method of removing an actor at the location, then after removing the actor, the method of putting an actor at the location will show.

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

Answer: use the method of setDirection.

Set 6

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

Answer: `if (gr == null) return false;`

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

Answer: `return (neighbor == null) || (neighbor instanceof Flower);`

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

Answer: the method of get(). In order to know whether the next location is occupied bya rock
the method of isValid(). Shows whether the next location is a legal location.

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

Answer: the method of getAdjacentLocation(). Gets the position of the next location.

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

Answer: the method of getGrid(). Get the grid instance the bug is in.
the method of getLocation. Get the location of the bug is at.

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

Answer: the bug will be removed from the grid.

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

Answer: Yes, it is needed. Yes, it could, but it makes the codes no so concise.

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

Answer: because the color of the flowers is gotten from the color of the bug.

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

Answer: no, it won't

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

Answer: flower.putSelfInGrid(gr, loc);

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

Answer: 4 times.