Do You Know?

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: loc1.getRow();

2. What is the value of b after the following statement is executed?
   ```
   boolean b = loc1.equals(loc2);
   ```
   Answer: The value of b is false.

3. What is the value of loc3 after the following statement is executed?
   ```
   Location loc3 = loc2.getAdjacentLocation(Location.SOUTH);
   ```
   Answer: loc3 has row 4, column 4.

4. What is the value of dir after the following statement is executed?
   ```
   int dir = loc1.getDirectionToward(new Location(6, 5));
   ```
   Answer: Dir has value 135 (degrees).

5. How does the getAdjacentLocation method know which adjacent location to return?
   Answer: The getAdjacentLocation method has a parameter Location to find the direction of the adjacent neighbor. It returns the closest position of the compass direction on the given direction.

Do You Know?

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: Suppose a grid object named gr.
   Obtain a count of the object: gr.getOccupiedLocations().size();
   Obtain a count of the empty locations:
   gr.getNumRows * gr.getNumCols - gr.getOccupiedLocations().size();

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

   Answer: Suppose a grid object named gr.
   boolean va = gr.isValid(new location(10, 10));
   If the value of va is true, the location is in the grid, else 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?

   Answer: The class Grid in an interface. Another class must implement these unimplemented. We can find the implementations of these methods 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. ArrayList does not require a fixed size. But array does. What' more, ArrayList can dynamically increase and decrease the element.

Do You Know?

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: The direction is north and its color is orange.

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

Answer: An interface is used to define an abstract type that contains no data or code, but defines behaviors as method signatures. We know that the Actor has itself state and behavior, so the Actor class is a class instead an interface.

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, the actor cannot put itself into a grid twice without first removing itself. It will throw an IllegalStateException when running.

No, the actor cannot remove itself from a grid twice. It will throw an IllegalStateException when running.

Yes, an actor can be placed into a grid, remove itself, and then put itself back.

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

Answer: Suppose an actor object named ac.

```
ac.setDirection(getDirection() + Location.RIGHT);
```

Do You Know?
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.isValid(next)) 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: Method isValid. It is called to ensure the next location is valid.

Method get. It is called to ensure that the next location whether can be replaced by a bug or not.

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

Answer: Method getAdjacentLocation. It is called to find the adjacent location in the compass direction that is closest to current direction.

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

Answer: Method getLocation.

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

Answer: Remove itself 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, the variable loc is needed in the move method.

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

Answer: It is convenient to find which bug drop the flower.

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

Answer: The answer is yes when calling the method move. However, the answer is no when just only calling the method removeSelfFromGrid.

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

Answer: 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?

Answer: 4.