21st Aug, 2015

Name：陈炜健 **Student ID**：13331018

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: False.

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

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

answer: The row of loc3 is 4, the column is 4.

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

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

answer: The value of dir is 135;

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

answer: The getAdjacentLocation method has a int parameter "direction", the method mods the "direction" by 360 and make the "direction" round to closest multiple of 45.

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:

- Obtain a count of the objects by using the **getOccupiedLocations** method.
- Obtain a count of the empty locations by using the **getEmptyAdjacentLocations** method.

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

answer:

```
grid.isValid(new Location(10, 10);
```

If the value is **true**, location(10, 10) is in a 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: In the software, the grid has two type, the BoundedGrid and the UnboundedGrid, which has different features. Some of the method of the **Grid** class must be implement in two different way, so it has the **AbstractGrid** class to implement the **Grid** Class. Then the **BoundedGrid** class and the **UnboundedGrid** inherit the **AbstractGrid** class. Because of that, the implementation of some Gird method will be different in the BoundedGrid and the 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. Because the ArrayList can dynamically increase but the Array must have a fixed size.

Do You Know? - Set 5

1. Name three properties of every actor.

answer: Location, Direction, Color.

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

answer: Its direction is blue, its color is north.

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

answer: An interface has no data, only abstract method. It is unable to use an interface to create an object. An class has data, no abstract method and it is able to use a class to create an object. We know that Actor has data, defined method and can be created as an object, so the Actor is a class.

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: An Actor can not put itself into a grid twice without first removing itself, can not remove itself from a grid twice, but 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:

```
actor.setDirection(90);
```

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 == 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:

- isValid method: Checks whether the adjacent location is valid int this grid.
- get method: Get the object of the adjacent location.

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

answer:

- getAdjacentLocation method: Get the location in front of the bug direction.

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

answer: getGrid and getLocation method.

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

answer: Removes the bug itself form the grid.

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

answer: Each actor has a private parameter **location** which can be used in the move method directly.

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

answer:

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

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

answer: No.

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: Four times.