13331235 TanXiao

1 Part 3 - Set3

1.1 How would you access the row value for loc1?

Use the getRow method.

1.2 What is the value of b after the following statement is executed?

The value of b is false.

1.3 What is the value of loc3 after the following statement is executed?

(4, 4)

1.4 What is the value of dir after the following statement is executed?

135

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

This method has a parameter to indicates the direction of the adjacent neighbor to find so that if we give it a direction, it will find a closest location in the given direction.

2 Part3 - Set4

2.1 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?

Use getOccupiedLocations method to get an array of the objects. Then use size method to get the array's size, the size is the count of objects.

Use getNumRows() * getNumCols() to find the total number of locations. Then use the total number to subtract the count of objects(we've got it before). The result is the count of empty locations.

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

Use isValid(new Location(10, 10)) to check. If the result is true, this location is in the grid.

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

Grid is an interface. All of methods in an interface are abstract method, they don't need to be implemented in the interface. These methods can be implemented by classes. We can find the implementations in AbstractGrid class and it's subclass BoundedGrid and UnboundedGrid.

2.4 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.

Using an array is more convenient for users but is more inconvenient to design grid classes. For example, you have to size the array when you want to use it, but the grid class doesn't count how many objects in the grid. When we want to get this number, grid class has to count objects one by one.

3 Part3 – Set5

3.1 Name three properties of every actor.

Color, Direction, Location

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

color = Color.BLUE;

direction = Location.NORTH;

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

Because the Actor class has some implemented methods, instance variables.

3.4 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?

An actor can't put itself into a grid twice without first removing itself. It will cause an IllegalStateException saying that “This actor is already contained in a grid.”.

An actor can't remove itself from a grid twice. It will cause an IllegalStateException saying that “This actor is not contained in a grid.”.

An actor can be placed into a grid, remove itself, and then put itself back. This part of code can run successfully.

3.5 How can an actor turn 90 degrees to the right?

Use setDirection method like this:

```
setDirection(getDirection() + Location.RIHGT);
```

4 Part3-Set6

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

```
Location next = loc.getAdjacentLocation(getDirection());
if (!gr.isValid(next))
   return false;
```

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

```
Actor neighbor = gr.get(next);
```

return (neighbor == null) || (neighbor instanceof Flower);

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

get method and isValid method. At first, the isValid method is used to judge if the location is valid in the grid, so that the canMove method invokes these methods. Then, canMove method has to judge if there is another actor on the next location.

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

getAdjacentLocation method. Use getAdjacentLocation method to find next location with the bug's current location and current direction.

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

getGrid, getLocation, getDirection

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

removeSelfFromGrid();

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

Yes. The move method uses it to store the bug's original location because the bug will leave a flower in it's original location.

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

Look at these code in the move method:

Flower flower = new Flower(getColor());

flower.putSelfInGrid(gr, loc);

Obviously the flower's color is same to the bug.

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

Yes, it will. After it called the removeSelfFromGrid method, the following code should run:

Flower flower = new Flower(getColor());

flower.putSelfInGrid(gr, loc);

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

Flower flower = new Flower(getColor());

flower.putSelfInGrid(gr, loc);

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

180 / 45 = 4 times