13331093
黄雄镖

Set 3
1. How would you access the row value for loc1?

Use the getRow method to access the row value for a location :
loc1 .getRow()

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

false.

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

(4, 4)

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

135(degrees)

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

The getAdjacentLocation method returns the adjacent location in the compass direction that is closest to *direction, by compare between the parameter and the eight constants of the compass directions*

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?

Use the getOccupiedLocations method to get an arraylist of all occupied locations in a grid, then use size() to return its size, the size is the count of the objects in a grid.

Use the getNumRows method and getNumCols method to get the rows and columns of a grid, then multiply them to get the total cells' number of the grid. Then the count of the empty locations in a bounded grid is the number of total cells subtract the count of the objects in a grid.

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

Use the *isValid* method to check like this: grid.*isValid(new Location(10, 10)), return true if it is in the grid, else false.*

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

Because the Grid is an interface, in its most common form, an interface is a group of related methods with empty bodies. A class that implements an interface must implement all of the methods described in the interface, or be an abstract class. The implementations of these methods can be found in the AbstractGrid class, BoundedGrid class and the UnboundedGrid class.

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.

I think array is similar to ArrayList. ArrayList is internally backed by array in Java, any resize operation in ArrayList will slow down performance as it involves creating new Array and copying content from old array to new array. It is depend on whether we know the size of the Array or not before hand. We should use ArrayList If we don't know about it. Another difference is the method to get object, using [] is easier than using the get method. If the Grid know the number of objects in the grid all the time, use the array may be better. I think ArrayList is more convenient in a way.

Set 5

1.Name three properties of every actor.

Location, direction, color.

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

North, bule

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

Because an interface is a group of related methods with empty bodies, we cann't implement th methods of it. but the actor has its own behavior.

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?

If I try to let the actor put itself into a grid twice without first removing itself, it will raise an exception like that: Exception in thread "main" java.lang.IllegalStateException: This actor is already contained in a grid.

If I try to let the actor remove itself from a grid twice, it will raise an exception like that: Exception in thread "main" java.lang.IllegalStateException: This actor is not contained in a grid.

It is OK to let the actor placed into a grid, remove itself, and then put itself back.

Test code:

```
import info.gridworld.actor.ActorWorld;
import info.gridworld.actor.Bug;
import info.gridworld.grid.Location;

import java.awt.Color;
public class Part3 {

	/**
	 * @param args
	 */
	public static void main(String[] args) {
		// TODO Auto-generated method stub
        final int aliceLocationX = 7;
        final int aliceLocationY = 8;
        final int aliceSideLength = 6;
        // construct a ActorWorld
        ActorWorld world = new ActorWorld();
        // construct two bugs
        Bug alice = new Bug();
        alice.putSelfInGrid(world.getGrid(), new Location(aliceLocationX, aliceLocationY));
        //put it twice
        //alice.putSelfInGrid(world.getGrid(), new Location(aliceLocationX, aliceLocationY));
        //remove it
        alice.removeSelfFromGrid();
        //remove it again
        //alice.removeSelfFromGrid();
        //put itself again
        alice.putSelfInGrid(world.getGrid(), new Location(aliceLocationX, aliceLocationY));
        world.show();
	}

}
```

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

First, use the getDirection method to get the actor's current direction, then add 90 to it, pass it to the method setDirection like this:

actor.setDirection(actor.getDirection() + 90)

Set 6

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

```
if (!gr.isValid(next))

    return false;
```

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);
```

It determines that a bug will only walk into a location which contains nothing or a flower.

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

The isValid method and get method, because if we try to move, we must know about whether it can move or not and what next is.

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

The getAdjacentLocation method, if we try to move forward, we should know the forward's location. The getAdjacentLocation(direction) will return the location of the give direction next to the object.

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

GetLocation, getGrid, getDirection

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

It will 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?

Yes, because we need to place a flower in the bug's old position after the bug move.

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

Because it use the getColor() the get the bug's color, one of the benefit is that we can know the bug path easier.

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

No.

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);
```

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

180 / 45 = 4 times