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?

loc1.getRow()

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

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

false

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

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

row4col4

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

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

135

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

Through the parameter.

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?

1 `getOccupiedLocations().length 2 getNumRows() * getNumCols() - getOccupiedLocations().length`

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

getNumRows >= 10 && getNumCols >= 10

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

1 Grid is an interface. 2 the class that implement the interface-->BoundedGrid and 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.

No, cuz translating the Object cost more space, which result in the less efficient.

Do You Know?

Set 5

1. Name three properties of every actor.
location color grid

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

3. Why do you think that the Actor class was created as a class instead of an interface?
Actor is a base class that could be satisfied “is a” relation with the class Bug, flower, etc. while interface should satisfied the relation “like a”.

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?
1 No 2 No 3 Yes

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

setDirection(getDirection() + 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?

```
if (!gr.isValid(next))
        return false;
```

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

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

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

gr.isValid(next), test if the next location is valid

gr.get(next), get the obj on the next location

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

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

GetAdjacentLocation get the next location for the bug to move

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

turn

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

Yes

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

Flower flower = new Flower(getColor());

flower.putSelfInGrid(gr, loc);

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?

4

Group Activity

Organize groups of 3--5 students.

1. Specify: Each group creates a class called Jumper. This actor can move forward two cells in each move. It "jumps" over rocks and flowers. It does not leave anything behind it when it jumps.

In the small groups, discuss and clarify the details of the problem:

a. What will a jumper do if the location in front of it is empty, but the location two cells in front contains a flower or a rock?

turn

b. What will a jumper do if the location two cells in front of the jumper is out of the grid?

turn

c. What will a jumper do if it is facing an edge of the grid?

turn

d. What will a jumper do if another actor (not a flower or a rock) is in the cell that is two cells in front of the jumper?

turn

e. What will a jumper do if it encounters another jumper in its path?

turn

f. Are there any other tests the jumper needs to make?

probably

2. Design: Groups address important design decisions to solve the problem:
   a. Which class should Jumper extend?

   bug

   b. Is there an existing class that is similar to the Jumper class?

   bug

   c. Should there be a constructor? If yes, what parameters should be specified for the constructor?

   No

   d. Which methods should be overridden?

   move canMove

   e. What methods, if any, should be added?

   no

   f. What is the plan for testing the class?

   Put a jumper into different states and test the correctness of its behavior
3. Code: Implement the Jumper and JumperRunner classes.
4. Test: Carry out the test plan to verify that the Jumper class meets the specification.