Do You Know?

Set 2

1. What is the role of the instance variable sideLength?

It can determine how far a bug can move in a direction

2. What is the role of the instance variable steps?

It can see whether the distance that a bug has moved equals to the sideLength

3. Why is the turn method called twice when steps becomes equal tosideLength?

Because a bug makes a turn by 45degree, in order to move as a box, the bug has to turn by 90degree per time, that means the turn method has to be called twice.

4. Why can the move method be called in the BoxBug class when there is no move method in the BoxBug code?

Because the BoxBug class has import the class by write "import info.gridworld.actor.Bug;" at the beginning of the class, which provides the move method to the bug

5. After a BoxBug is constructed, will the size of its square pattern always be the same? Why or why not?

Of course not. The size of its square pattern will be the same at last, but not always at the beginning.

6. Can the path a BoxBug travels ever change? Why or why not?

Yes. Sometimes if a bug crack into the wall it will change its direction and finally it will form it own path.

7. When will the value of steps be zero?

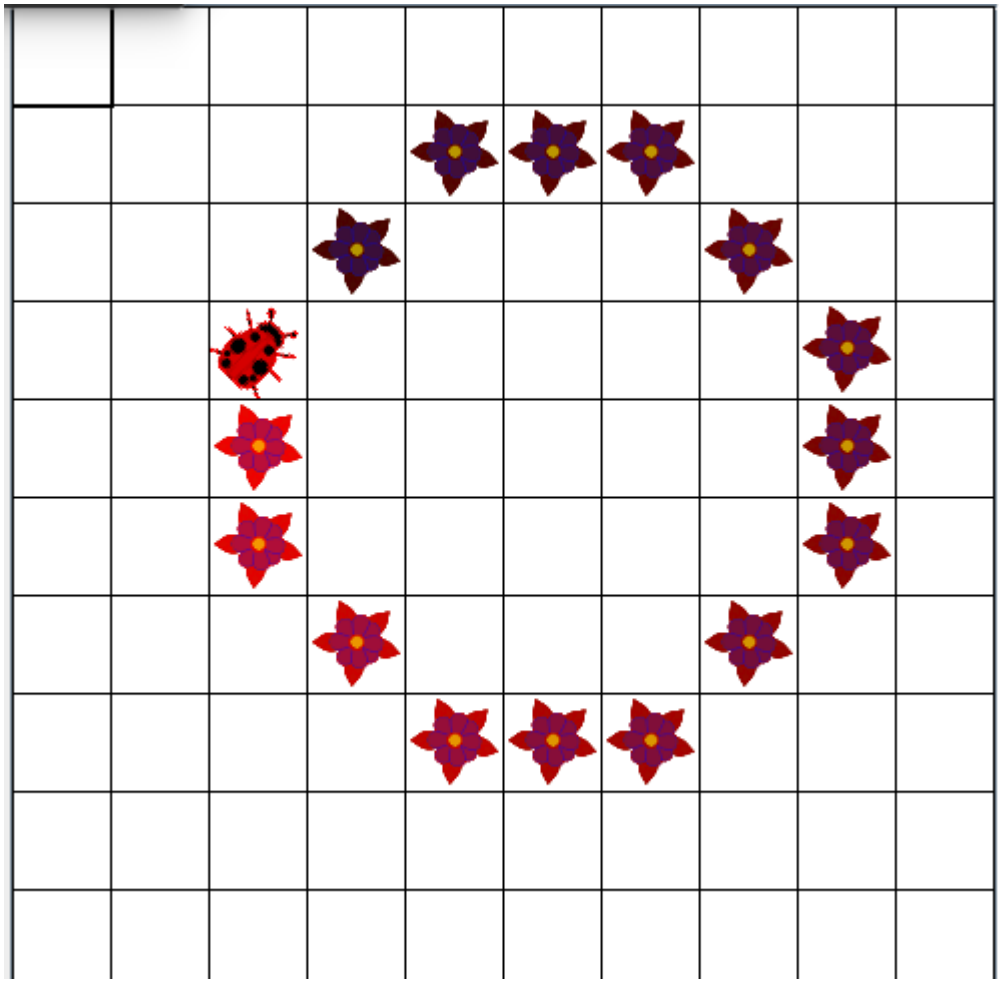
Every time when the bug move by the length of sideLength or the bug crack into to the wall.

Exercise

1. Write a class CircleBug that is identical to BoxBug, except that in the act method the turn method is called once instead of twice. How is its behavior different from a BoxBug?

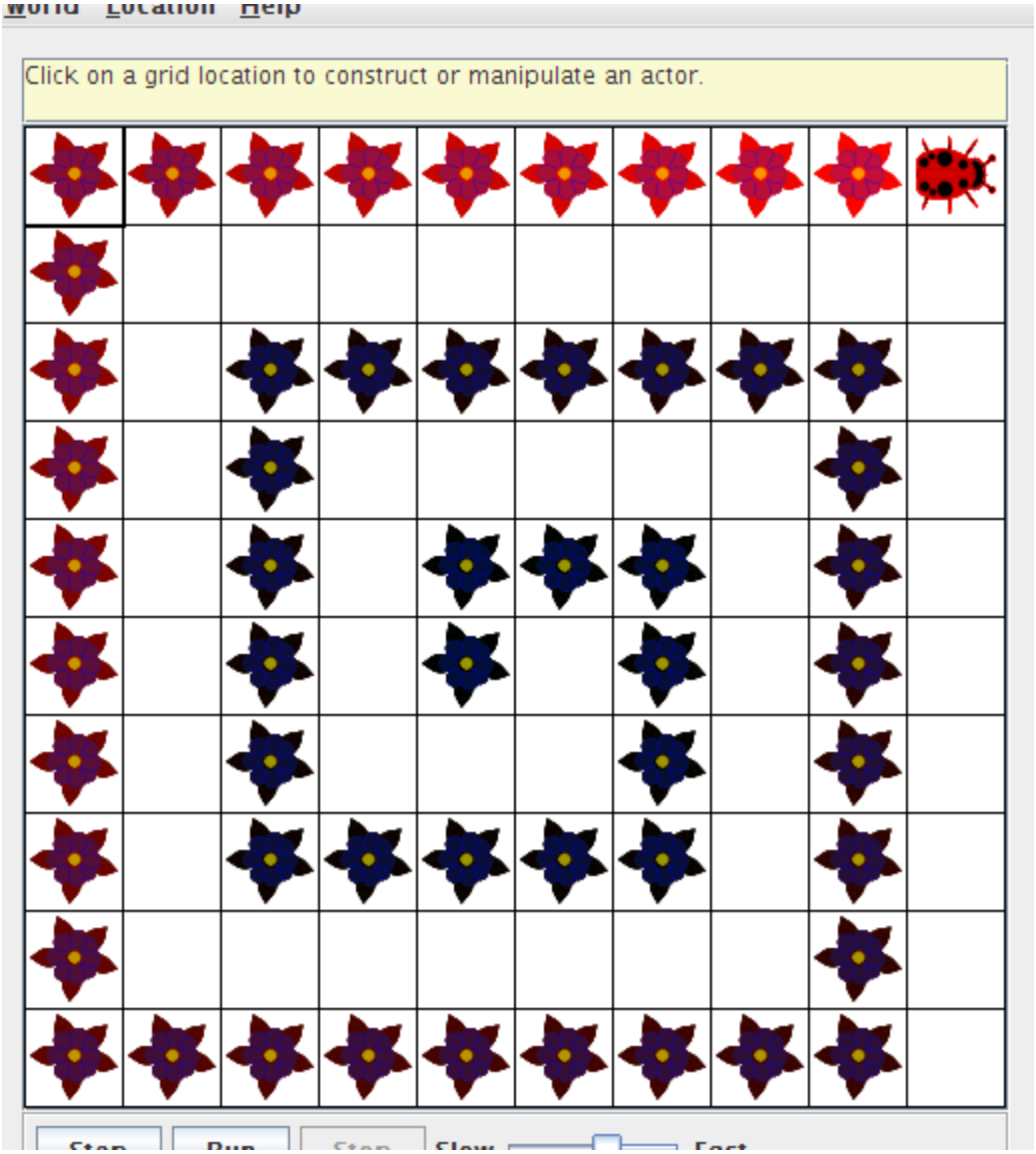
!!!The code is attch to the zip file

```
     */
    public void act()
    {
        if (steps < sideLength && canMove())
        {
            move();
            steps++;
        }
        else
        {
            turn();
            steps = 0;
        }
    }
```



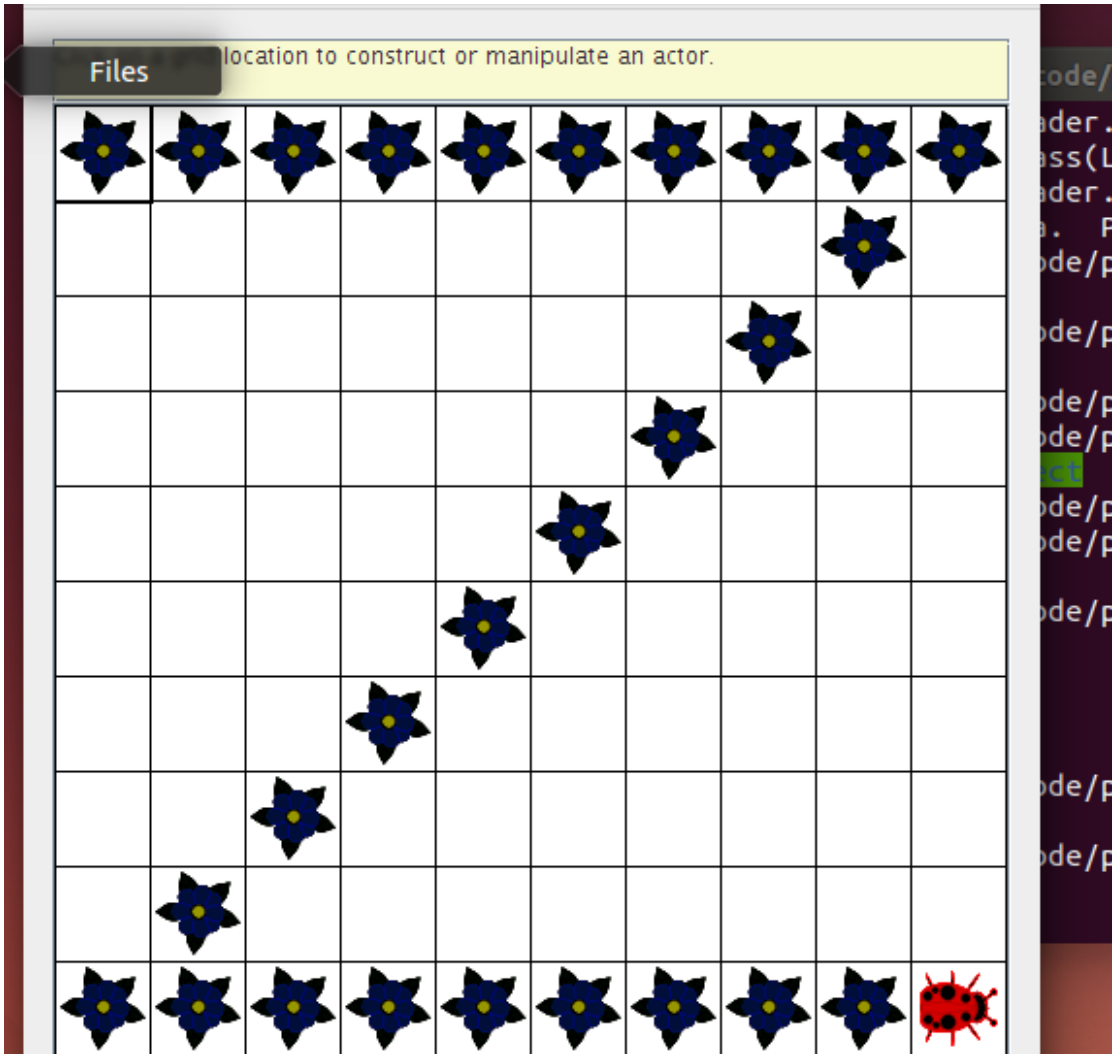
2. Write a class SpiralBug that drops flowers in a spiral pattern. Hint: Imitate BoxBug, but adjust the side length when the bug turns. You may want to change the world to an UnboundedGrid to see the spiral pattern more clearly.

!!!The code is attch to the zip file



3. Write a class *ZBug* to implement bugs that move in a "Z" pattern, starting in the top left corner. After completing one "Z" pattern, a *ZBug* should stop moving. In any step, if a *ZBug* can't move and is still attempting to complete its "Z" pattern, the *ZBug* does not move and should not turn to start a new side. Supply the length of the "Z" as a parameter in the constructor. The following image shows a "Z" pattern of length 4. Hint: Notice that a *ZBug* needs to be facing east before beginning its "Z" pattern

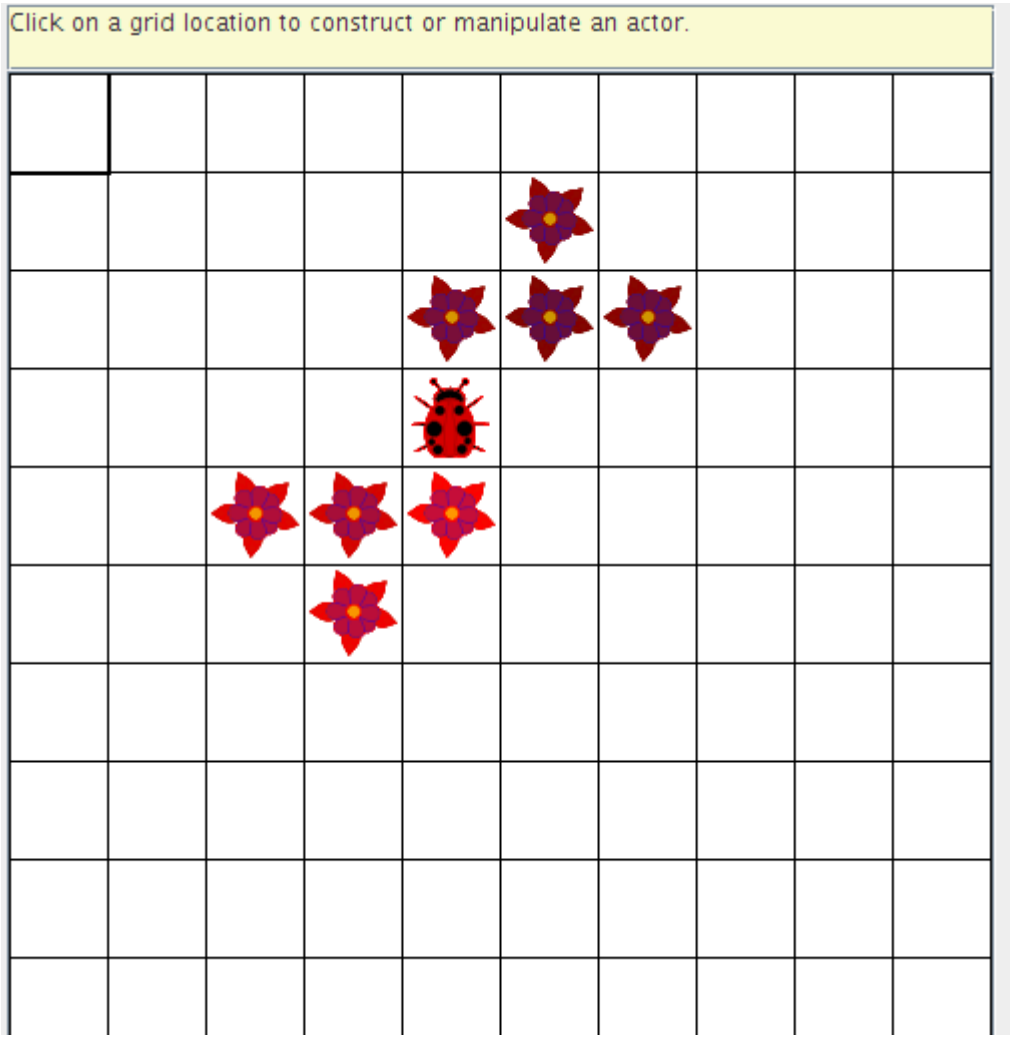
!!!The code is attch to the zip file



4. Write a class *DancingBug* that "dances" by making different turns before each move. The *DancingBug* constructor has an integer array as parameter. The integer entries in the array represent how many times the bug turns before it moves. For example, an array entry of 5 represents a turn of 225 degrees (recall one turn is 45 degrees). When a dancing bug acts, it should turn the number of times given by the current array entry, then act like a Bug. In the next move, it should use the next entry in the array. After carrying out the last turn in the array, it should start again with the initial array value so that the dancing bug continually repeats the same turning pattern.

The DancingBugRunner class should create an array and pass it as aparameter to the DancingBug constructor.

!!!The code is attch to the zip file



5. Study the code for the BoxBugRunner class. Summarize the steps you would use to add another BoxBug actor to the grid.

First, new a BoxBug actor
Second, add the BoxBug into the World