Do You Know?
Set 2

The source code for the BoxBug class can be found in the boxBug directory。.

1. What is the role of the instance variable sideLength?
The length for bug to move until it turn.

2. What is the role of the instance variable steps?
Record how many steps the bug has moved since the last turn.

3. Why is the turn method called twice when steps becomes equal to
   sideLength?
One turn is 45 degree clockwise, it should turn 90 degree.

4. Why can the move method be called in the BoxBug class when there is no
move method in the BoxBug code?
BoxBug inherit Bug which is in gridworld.jar, and the method is from Bug.

5. After a BoxBug is constructed, will the size of its square pattern always be the
same? Why or why not?
No, the constructor has a parameter length which used to declare the size of the square pattern.

6. Can the path a BoxBug travels ever change? Why or why not?
Yes, if there is a actor in front of his path, it would turn before the sideLength is finished.

7. When will the value of steps be zero?
1 constuct a bug 2 when it turn.

Exercises

In the following exercises, write a new class that extends the *Bug* class. Override the *act* method to define the new behavior.

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?

It run in a positive octagon track.

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.

When it turn, make sideLength+1.

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.

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.

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

1 new BoxBug() 2 world.add(location, bug)