APCS Gridworld:Chapter 2 Name: Per: F

Set 2, Page 12
Source code for the BoxBug class is in Appendix C.

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

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

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

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

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

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

7. When will the value of steps be zero?

1. (a) Write a class CircleBug that is identical to BoxBug, except that in the act method the turn method is called once instead of twice.

(b) How is its behavior different from a BoxBug?

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.

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 cant 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 a parameter 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.

```
public class BoxBugRunner
{
     public static void main(String[] args)
     {
          ActorWorld world = new ActorWorld();
          BoxBug alice = new BoxBug(6);
          alice.setColor(Color.ORANGE);
          BoxBug bob = new BoxBug(3);
          world.add(new Location(7, 8), alice);
          world.add(new Location(5, 5), bob);
          // your code here









          world.show();
          }
}
```