**GridWorld - "DixonBug" Assignment**

You will create a new class, *DixonBug*, that extends the Bug class.

***DixonBug* Data Fields**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Purpose** |
| currentColor | String [ ] | Holds the colors the *DixonBug* will change into for each step of its movement |
| direction | int [ ] | Holds the directions the *DixonBug* will move for each step of its movement |
| index | int | Holds the current index of the String and integer arrays being used |
| distance | int | Holds the number of grid spaces the *DixonBug* will move with each "step" |

***DixonBug Constructor***

The *DixonBug* constructor will accept an integer array, a String array, and an integer. You can assume that the two arrays (integer and String) will be the same size. The integer will be used to set the distance variable's value.

**Direction Array**

The integer array holds the direction in which your DixonBug will move. There are eight possible directions that your DixonBug can move. The follow table shows how the integer value will be used to set the direction for your DixonBug:

|  |  |  |
| --- | --- | --- |
| 7 | 0 | 1 |
| 6 | Bug | 2 |
| 5 | 4 | 3 |

Thus, if the value read from the array is 5, your DixonBug will move southwest. The setDirection method allows us to specify the direction our bug is facing.

\*Hint: We can multiply the above values by 45 to get the appropriate degree of angle.

**currentColor Array**

* This String array holds the color your DixonBug will change into for each step.
* Every step, it will change its color to match the color provided at that index in the currentColor array.
* The Dixonbug should only accept the values "red", "green" and "blue", with lowercase being a necessity. Any other value read from the String array will cause the DixonBug to turn yellow for that step.

**Movement**

Your DixonBug will move forward *distance* spaces each "step". If the DixonBug is unable to move forward all steps, it will move forward as many as it can until it is blocked. For example, if the distance is 5 and the bug moves forward 3 step, but then hits a rock or edge of the grid, it will move those 3 steps, then move forward no more. The next step, a new direction will be set for the DixonBug and it will move in the same manner.

**Order of Events** - These happen each "step" in this order:

1. changeColor() - This will always happen first, regardless of how the DixonBug moves. It is trackable because if the color of flowers left behind matches the DixonBug's color.

2. setDirection() - Based on the value in the integer array (0 to 7), point your bug in the appropriate direction.

3. move() - If the DixonBug is able to move, it will move forward *distance* spaces, or as many spaces it can until it is blocked, either by a rock, another Bug, or the edge of the Grid.

\*Note: Your DixonBug will stop moving once it reaches the end of the direction array. The bug will not move at this point, even if step is pressed extra times. If they click on the step button after they reach the end of the direction array, the program should not generate an error, but a message "No more steps to perform" should be outputted to the console."

**DixonBugRunner**

Your DixonBugRunner program will utilize the following code:

**import** info.gridworld.actor.ActorWorld;

**import** info.gridworld.grid.Location;

**import** java.awt.Color;

**public** **class** DixonBugRunner {

**public** **static** **void** main(String[] args) {

 **int** [] nums = {2, 2, 7, 5, 0, 0, 3, 1, 6, 6, 1, 3, 4, 4};

 String[] colors={"red", "blue", "green", "white", "blue", "Blue", "green", "red", "green", "red", "blue", "green", "blue", "green"};

 ActorWorld world = **new** ActorWorld();

 DixonBug dixon = **new** DixonBug(colors, nums, 3);

 dixon.setColor(Color.*BLUE*);

 world.add(**new** Location(9,1), dixon);

 world.show();

 }

}

Your DixonBug should move 14 times and the pattern in the end should match what you see below exactly.

