ISM3230 Individual Assignment 5 – Classes Spring 2017

In this assignment you will model a home automation light controller that can turn on and off individual lights in a building. The controller will be an object that will have public methods to control the lights. The lights will in turn also be objects created from the Light class. A simple driver program will demonstrate the use cases of the controller.

Requirements:

Light class:

The light class represents an individual light. A light can be defective or functional, if it is not defective, it can be on or off (when defective it is off), and has a brightness ranging from 0 to 255. The brightness level is remembered even when the light is off, so when it is turned on again it has the saved brightness level. Finally, each light has a random defect rate that ranges from 5% to 15% (exaggerated for this exercise).

When a light object is created, the light is off, not defective, brightness is set to full and a random defect rate is set. A light can be turned on, turned off, and the brightness can be set from 0 to 255. When the light is being turned on, there is a chance that the light will burn out and consequently will be immediately turned off and marked as defective. A light can indicate its status by returning a character to the caller: a dot '.' means off, 'o' means on with intensity <128, and 'O' means on with intensity>=128.

Controller class:

A controller controls a number of lights. When a controller is created, it must be given the number of lights it will contain. The controller will create the required number of the lights and save their references in an array. The controller has several actions it can perform with the lights. It can turn all the lights on, all the lights off, individual light on (as indicated by a parameter that is the index into the lights array), individual light off, and it can set the brightness of a specific light.

For the purposes of this simulation, the controller prints out the status of all the lights after every action that affects any of the lights. The controller asks each light for its status character and prints the whole set of lights in one line after a light is turned on, off, or the brightness is changed.

The driver program:

This is the testing program for the controller. It contains the following actions:

- it creates one controller with 20 lights
- it cycles all the lights on and off four times
- it turns light 5 on
- it sets the brightness of light 5 to 20
- it sets the brightness of light 10 to 100
- it turns light 10 on

- it sets the brightness of light 10 to 160
- it turns all the lights on

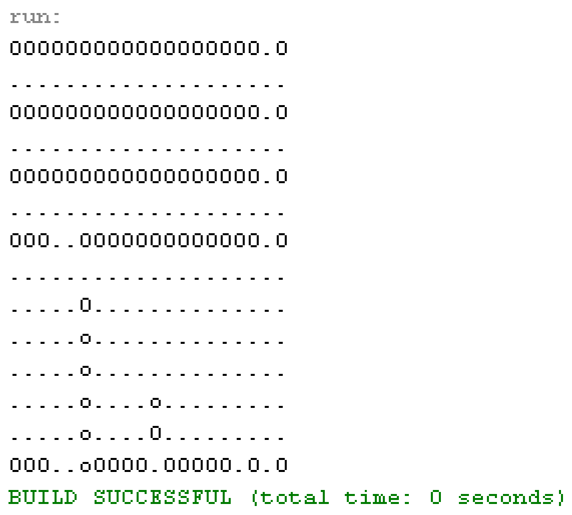
Instructions:

1) Create a new Project. Add new class files for `Light` and `Controller`.
2) `Light` class:
   a) private attribute `boolean isOn`
   b) private attribute `boolean isDefective`
   c) private attribute `int brightness`
   d) private attribute `double defectRate`
   e) a default constructor that will
      i) set brightness to 255, turn light off and set defective to off
      ii) generate a random number between .05 and .15 and assign it to `defectRate`
      Hint: use `Random`'s `nextDouble()` method that returns a number from 0.0 to 1.0. Get this number to the desired range of .05 to .15 by dividing it by 10 and adding an offset of .05.
   f) private method `boolean isBurningOut()` that simulates the probability of the light burning out when turning on. It should return `true` if the light is already defective. Otherwise it should generate a random number from 0.0 to 1.0 (with `nextDouble()`) and compare it to the `defectRate`. If the random number is less than `defectRate`, the light will burn out, so the method should return `true`. Otherwise it should return `false`.
   g) public method `void turnOn()` that attempts to turn on the light. It should check with the `isBurningOut()` method and then set the light's `isOn` to `true` if it is not burning out, otherwise, it should turn the light off and mark it as defective.
   h) public method `void turnOff()` that turns the light off.
   i) public method `void setBrightness(int desiredBrightness)` that checks the parameter for being in allowable range and sets the light brightness to the desired value. If the parameter is less than 0 or more than 255, set it to 0 or 255, respectively.
   j) public method `char getStatus()` that returns a character `'o'` if the light is on with brightness less than 128, `'O'` if it is on with brightness 128 or more, and dot `'.'` if the light is off.
3) `Controller` class:
   a) private array `Light[] lights` that holds the references to light objects
   b) a constructor that takes one parameter `numLights` that indicates how many lights are controlled by the controller
      i) it dimensions the `lights` array to size `numLights`
      ii) it creates new lights objects and stores them in the `lights` array (use a `for` loop)
   c) public method `void printStatus()` that goes through the `lights` array and prints the status of each light on one line. Afterwards it prints a newline to move the printing position to the next line.
   d) public method `void turnLightOn(int lightid)` that turns on the indicated light and calls `printStatus()`

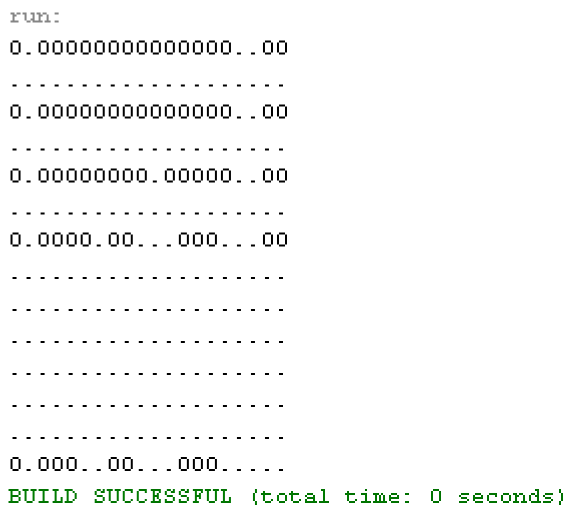
   e) public method `void turnLightOff(int lightid)` that turns off the indicated light and calls `printStatus()`
   f) public method `void setLightBrightness (int lightid, int intensity)` that sets the brightness of the indicated light to the desired intensity and calls `printStatus()`
   g) public method `void turnAllOn()` that turns all the lights on and calls `printStatus()`
   h) public method `void turnAllOff()` that turns all the lights off and calls `printStatus()`
   i)
4) Driver program:
   a) Instantiate one controller object with 20 lights.
   b) perform the operations as described in the Requirements section.

SAMPLE OUTPUT

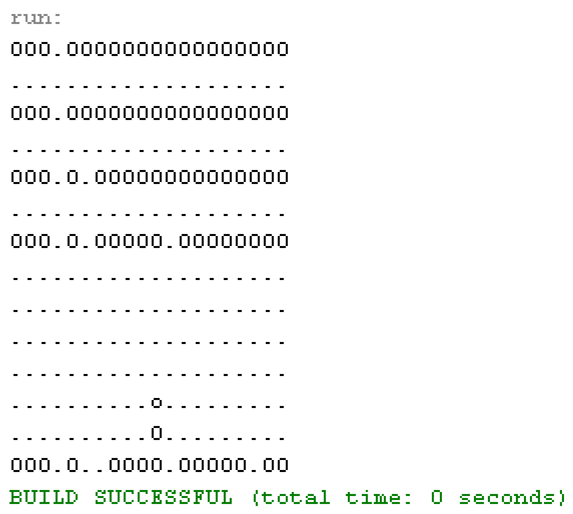
Note: These results have randomly generated defect rates and defect probabilities, results will be different with every program run.



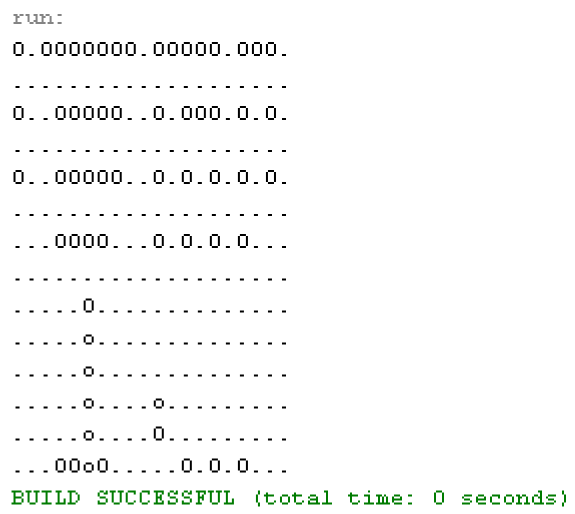
```
run:
OOOOOOOOOOOOOOOOOO.O
--------------------
OOOOOOOOOOOOOOOOOO.O
--------------------
OOOOOOOOOOOOOOOOOO.O
--------------------
OOO..OOOOOOOOOOOOO.O
--------------------
.....O..............
.....o..............
.....o..............
.....o....o.........
.....o....O.........
OOO..oOOOO.OOOOO.O.O
BUILD SUCCESSFUL (total time: 0 seconds)
```



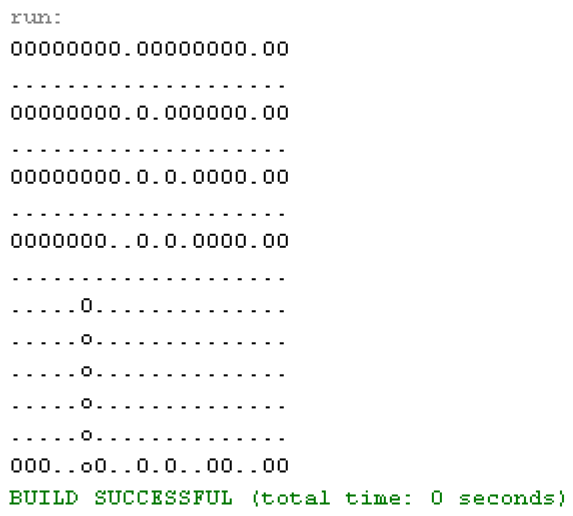
```
run:
O.OOOOOOOOOOOOOO..OO
--------------------
O.OOOOOOOOOOOOOO..OO
--------------------
O.OOOOOOOO.OOOOO..OO
--------------------
O.OOOO.OO...OOO...OO
--------------------
--------------------
--------------------
--------------------
--------------------
--------------------
O.OOO..OO...OOO.....
BUILD SUCCESSFUL (total time: 0 seconds)
```



```
run:
000.0000000000000000
....................
000.0000000000000000
....................
000.0.00000000000000
....................
000.0.00000.00000000
....................
....................
....................
....................
..........o.........
..........0.........
000.0..0000.00000.00
BUILD SUCCESSFUL (total time: 0 seconds)
```



```
run:
0.0000000.00000.000.
....................
0..00000..0.000.0.0.
....................
0..00000..0.0.0.0.0.
....................
...0000...0.0.0.0...
....................
.....0..............
.....o..............
.....o..............
.....o....o.........
.....o....0.........
...00o0.....0.0.0...
BUILD SUCCESSFUL (total time: 0 seconds)
```



```
run:
00000000.00000000.00
....................
00000000.0.000000.00
....................
00000000.0.0.0000.00
....................
0000000..0.0.0000.00
....................
.....0..............
.....o..............
.....o..............
.....o..............
.....o..............
000..o0..0.0..00..00
BUILD SUCCESSFUL (total time: 0 seconds)
```