Question 1: Stepwise Refinement (30%)

Complete the class called `DrawPicture.java`. The entire interface for the class is a single constructor that draws a pattern consisting of 16 equal sized rectangular regions:

Region 1	Region 2	Region 3	Region 4
Region 5	Region 6	Region 7	Region 8
Region 9	Region 10	Region 11	Region 12
Region 13	Region 14	Region 15	Region 16

The constructor has parameters corresponding to the height and width of a region (assume the numbers are positive). The constructor creates an appropriately sized board and places coloured pegs and lines to generate the following pattern:

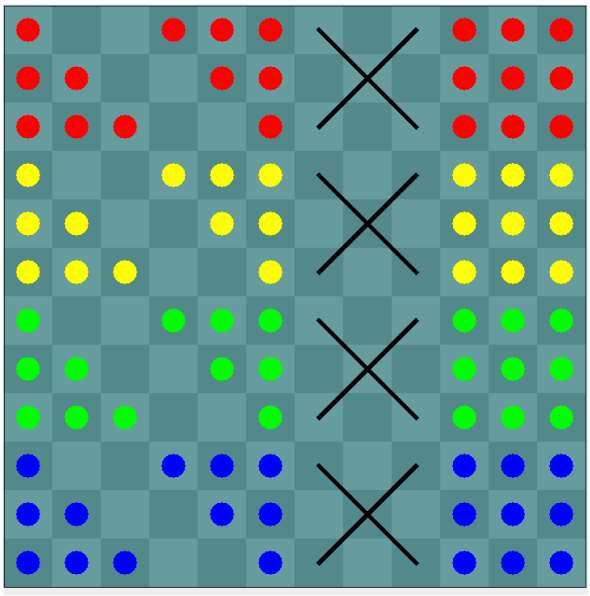
- Regions 1-4 use red pegs. Regions 5-8 use yellow pegs. Regions 9-12 use green pegs. Regions 13-16 use blue pegs.
- Regions 1, 5, 9, 13 contain a triangle pattern. The pattern consists of a peg in the first square of the first row, pegs in the first two squares of the second row, etc. If the region is taller than it is wide, the extra rows will be entirely filled with pegs.
- Regions 2, 6, 10, 14 contain a different triangle pattern. The pattern consists of pegs in every square of the first row, pegs in all but the first square in the second row, etc. If the region is taller than it is wide, the extra rows will not have any pegs.
- Regions 3, 7, 11, 15 contain an "X" going from the corners of the region.
- Regions 4, 8, 12, 16 are completely filled with pegs.

The following page contains three screen captures along with the input that produced them.

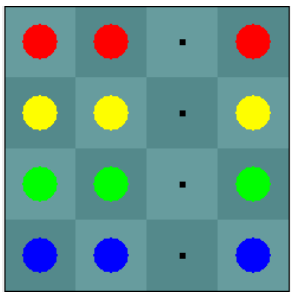
Marks will be given for proper use of concepts taught in lecture like stepwise refinement and constants. Your mark will be **significantly** reduced should you choose to put all of your code inside of one method.

Files to Submit: `DrawPicture.java`

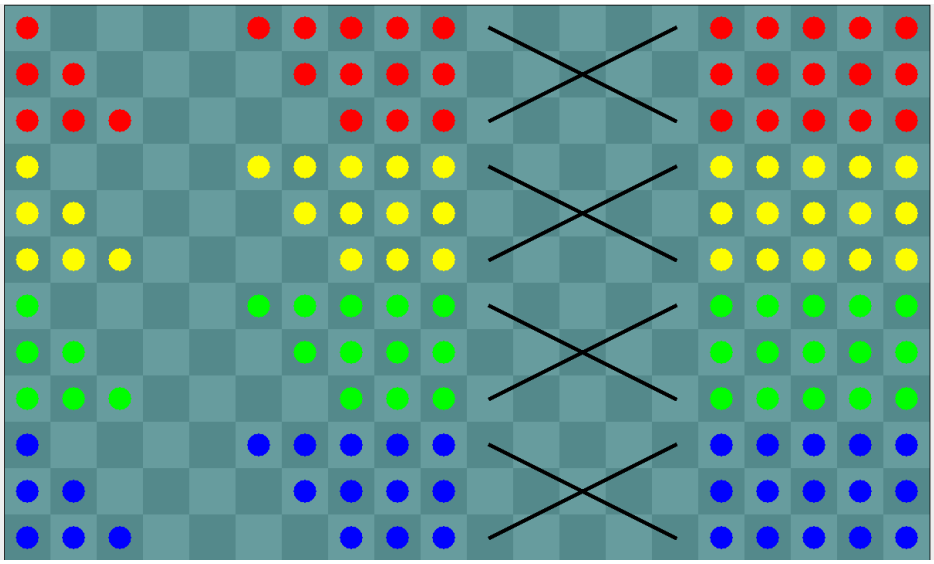
DrawPicture(3, 3)



DrawPicture(1, 1)



DrawPicture(3, 5)



Question 2: Arrays (30%)

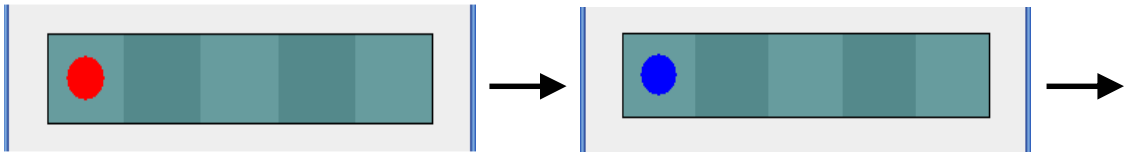
Write a program called `Lights.java`. The program will read in a number (you may assume it is positive) from the user and create a 1-D board with that many squares in it. The user will then click repeatedly on the board.

Whenever the user clicks on an empty square, a red peg is placed on the square. Whenever the user clicks on a non-empty square, **all** squares with a peg on it advance one colour in the following sequence:

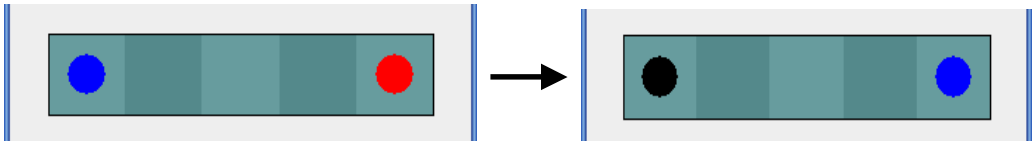
red → blue → black → white → empty

The program ends when the board gets back to being empty.

For instance, if the user clicks on the leftmost square twice the board would be:



Then if the user clicks on the rightmost square twice the board would be:



Marks will be given for proper use of concepts taught in lecture like using constants.

Files to Submit: `Lights.java`

Question 3: Implementing Classes with Arrays (40%)

Complete the class called `Polynomial.java` which uses an array to keep track of the integer coefficients for a polynomial. The class includes methods for various operations such as adding, multiplying, evaluating and printing polynomials. Here is an example of a polynomial:

$$3 + x - 12x^3 + 7x^5$$

Here is a class diagram of the operations defined for a Polynomial:

Polynomial
- int[] coefficient - int maxDegree
+ Polynomial(int maxDeg) + int getDegree() + void setCoeff(int degree, int val) + int eval(int x) + void add(Polynomial other) + void multiply(Polynomial other) + String toString()

Read the pre and post conditions to fully understand how the operations work. Here is a brief explanation of some of the variables and methods

- A polynomial has only non-negative integer powers of *x*
- The constructor creates the zero polynomial. The maximum degree that the polynomial will ever have is provided. This allows you to create the integer coefficient array, and you need not worry about having to enlarge it later on when performing an add or multiply.
- The degree of the zero polynomial is defined to be -1
- The `eval` method evaluates the polynomial with a particular value of x. For instance, the above polynomial evaluated at *x=2* would be *133*.
- It is the other polynomial that is added/multiplied to this. It is assumed this has a maximum degree that is large enough to hold the polynomial.
- The method `toString` returns a string representation of the polynomial (from highest to lowest degree terms). For example, the polynomial at the start of the question would be displayed as:

```
+7x^5 -12x^3 +1x +3
```

Notice that terms with a coefficient of zero are not present in the string. The only exception is the zero polynomial; its string should simply be "0".

- As a reminder, to add two polynomials, simply add the coefficients of terms with the same degree. For example:

$$(6x^4 + 2x^2 + 4) + (7x^4 - 2x^2 + 9x)$$
$$= (6+7)x^4 + (0+0)x^3 + (2-2)x^2 + (0+9)x + (4+0)$$
$$= 13x^4 + 9x + 4$$

- As a reminder, to multiply two polynomials, each term of one polynomial multiplies each term of the other polynomial. Then, terms of the same degree are combined together and their coefficients are added to form a single coefficient. For example:

$$(2x - 4) + (x^2 + 3x + 5)$$
$$= 2x(x^2 + 3x + 5) + (-4)(x^2 + 3x + 5)$$
$$= 2x^3 + 6x^2 + 10x - 4x^2 - 12x - 20$$
$$= 2x^3 + 2x^2 - 2x - 20$$

Files to Submit: `Polynomial.java`