CMSC131 Fall 2021 (https://www.cs.umd.edu/class/fall2021/cmsc131-01XX-03XX/)

Project #3, Drawing App (Due Thursday, Sep 30, 11:55 pm)

Objectives

This project will allow you to practice definition of static methods and nested loops.

Remember that you should develop your code using incremental code development; write a little bit, test your code and move forward.

Overview

For this project you will write methods that draw different diagrams (rectangles, flags, etc.). **For this project you can discuss the project with other classmates, but you MAY NOT write code with other students, nor look at other students' code.** Remember that your goal is to learn concepts and to be ready for quizzes/exams. In an exam, you have to write the code yourself; you will be able to do so, if you have implemented the project by yourself.

Grading

- (46%) Public Tests
- (30%) Release Tests
- (16%) Secret Tests
- (8%) Style

Code Distribution

The project's code distribution is available at DrawingApp.zip (https://www.cs.umd.edu/class/fall2021/cmsc131-01XX-03XX/prot/projects/zipFiles/DrawingApp.zip). Download it and import it as you have imported our class examples. The code distribution provides you with the following:

- **app** → A package (folder) where you will complete the implementation of the **DrawingApp** class.
- **gui** → A package (folder) with classes that support the displaying of a drawing in color. You can ignore this package if you want (you can complete the whole project without ever looking at these files). The file **DisplayDiagramExample.java** illustrates how to display a diagram.
- **tests** → A package (folder) where you will find public tests.
- **expectedResults** - A folder that has the expected results for each of the public test.
- **results** - A folder that the results generated by your code for each of the public tests.

Specifications

You need to implement the following static methods associated with the **DrawingApp** class. Feel free to add any **private** static methods you think can be useful.

1. **private static boolean isValidColor(char color)** - This method returns true if the color parameter corresponds to one of the following characters: 'R' (red), 'G' (green), 'B' (blue), 'Y' (yellow), '*' (black), or '.' (white).

2. **public static char getRandomColor(Random random)** - Using the random.nextInt(6) method call, generate a random integer value between 0 (inclusive) and 5 (inclusive). The character to return for a particular integer will be:

    - 0 → 'R'
    - 1 → 'G'
    - 2 → 'B'
    - 3 → 'Y'
    - 4 → '*'
    - 5 → '.'

    To help you track down problems with your code, you should return another character (e.g., 'X') for a value other than 0 thru 5.

3. **public static String getRectangle(int maxRows, int maxCols, char symbol)** - This method returns a string with a rectangle that has maxRows and maxCols, using the symbol character. The method will return the special value **null** (it means nothing or no address) if maxRows or maxCols (or both) are less than 1; in this case no diagram will be generated. The method MUST NOT rely on System.out.println(). For example, calling DrawingApp.getRectangle(6, 9, '*'); will return the following string:

```
*********
*********
*********
*********
*********
*********
```

4. **public static String getHorizontalBars(int maxRows, int maxCols, int bars, char color1, char color2, char color3)** - This method returns a string with a number of horizontal bars that correspond to the **bars** parameter. To compute the size of each horizontal bar, divide maxRows by the specified number of bars. The first horizontal bar will use color1, the second color2, and the third color3. If more than 3 bars are present, we will start again with color1. If the computed size for a horizontal bar is less than 1, or any of the colors is invalid, the method will return **null** and no diagram will be generated. The method MUST NOT rely on System.out.println(). For example, calling DrawingApp.getHorizontalBars(12, 10, 3, 'R', 'G', 'B'); will return the following string:

```
RRRRRRRRRR
RRRRRRRRRR
RRRRRRRRRR
RRRRRRRRRR
GGGGGGGGGG
GGGGGGGGGG
GGGGGGGGGG
GGGGGGGGGG
BBBBBBBBBB
BBBBBBBBBB
BBBBBBBBBB
BBBBBBBBBB
```

Calling DrawingApp.getHorizontalBars(6, 8, 3, 'R', 'B', 'G'); will return the string:

```
RRRRRRRR
RRRRRRRR
BBBBBBBB
BBBBBBBB
GGGGGGGG
GGGGGGGG
```

5. **public static String getFlag(int size, char color1, char color2, char color3)** - This method returns a string where a triangle appears on the left size of the diagram, followed by horizontal lines. For example, calling DrawingApp.getFlag(9, 'R', '.', 'Y'); will return the following string:

```
R.........................................
RRYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
RRRYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
RRRRYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
RRRRRYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
RRRRRRYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
RRRRRRRYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
RRRRRRRRYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
RRRRRRRRRYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
RRRRRRRRR.................................
RRRRRRRRR.................................
RRRRRRRRRYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
RRRRRRRRYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
RRRRRRRYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
RRRRRRYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
RRRRRYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
RRRRYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
RRRYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
RRYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
R.........................................
```

The diagram has a number of rows that corresponds to **size * 2** and a number of colums that corresponds to **size * 5**. The first and last row will use color2 (except for the first character that will use color1). The center two rows will use color2 and the rest color3. The triangle will rely on color1 and will have a number of

rows corresponding to **size * 2**. If the **size** parameter is less than three, the method will return **null** and will not generate any diagram. For this method you can assume the colors are valid. The method MUST NOT rely on System.out.println().

6. **public static String getVerticalBars(int maxRows, int maxCols, int bars, char color1, char color2, char color3)** - This method returns a string with a number of vertical bars that correspond to the **bars** parameter.To compute the size of each vertical bar, divide maxCols by the specified number of bars. The first vertical bar will use color1, the second color2, and the third color3. If more than 3 bars are present, we will start again with color1. If the computed size for a vertical bar is less than 1, or any of the colors is invalid, the method will return **null** and no diagram will be generated. The method MUST NOT rely on System.out.println(). For example, calling DrawingApp.getVerticalBars(10, 12, 3, 'R', 'G', 'B'); will return the following string:

```
RRRRGGGGBBBB
RRRRGGGGBBBB
RRRRGGGGBBBB
RRRRGGGGBBBB
RRRRGGGGBBBB
RRRRGGGGBBBB
RRRRGGGGBBBB
RRRRGGGGBBBB
RRRRGGGGBBBB
RRRRGGGGBBBB
```

Calling DrawingApp.getVerticalBars(5, 9, 3, 'R', 'B', 'G'); will return the string:

```
RRRBBBGGG
RRRBBBGGG
RRRBBBGGG
RRRBBBGGG
RRRBBBGGG
```

Requirements

- **Diagrams cannot have a newline character ('\n') in the last row. If you add a newline methods that display diagrams in color will fail.**
- Do not define any static variables. Static variables can make public tests fail.
- You may define private static auxiliary methods, but no public methods.
- The methods you implement for this project will be used for a future class project.
- **You may not use Arrays nor ArrayLists for this project. If you do, you will lose all the points associated with tests.**
- You must followed the style guidelines available at style (http://www.cs.umd.edu/~nelson/classes/resources/javastyleguide/).
- Verify that your project passes the submit server public and release tests (https://submit.cs.umd.edu/ (https://submit.cs.umd.edu/fall2021)). Your score on the project will be based on results from the submit server and NOT from your results in Eclipse.
- This space is available for rent 😊.

- Files with the "Results.txt" extension represent the results generated by your code when public tests are run. Remember to refresh the Eclipse project to see these files. To compare files you can use the Eclipse compare option or use the web sites you will find on the class web page (Resources→ Other section).
- You can only release test your project once you have passed all public tests. For this project you have 5 tokens in a 22-hour period.
- Do not use System.exit(0) in your programs.

Displaying Your Diagrams In Color

- You don't need to display your diagrams in color in order to complete this project. You can ignore this section if you prefer.
- If you would like to see your diagrams in color:
  - Take a look at the **DisplayDiagramExample.java** class that is in the **gui** package of the project distribution.
  - Currently the main method displays the flag, but you can replace

```
String diagram = DrawingApp.getFlag(size, color1, color2, color3);

                                    with

String diagram = DrawingApp.(ANY OTHER METHOD THAT CREATES A DIAGRAM);
```

  - If you run the main method of the **DisplayDiagramExample.java** class you will see the diagram in color.

Testing

In previous projects you were writing a full program that defined a main that read data and did some processing. In this project you are defining methods that are called by other methods. You can test the methods you are implementing by defininig a driver class. This class will have main method that will call the methods you need to implement. The provided public tests perform a similar task, but they provide additional options. The following is an example of a driver that allows you to test some of the functionality of one of the methods you need to implement. This class is not part of the code distribution.

```
import app.DrawingApp;
public class SampleDriver {

   public static void main(String[] args) {
      int maxRows = 6, maxCols = 9;
      char symbol = '*';

      String result = DrawingApp.getRectangle(maxRows, maxCols, symbol);
      System.out.println(result);
   }
}
```

Submission

- Submit as usual.

- If you write the project from scratch, without using the provided zip file, you will not be able to submit your work.

Academic Integrity

Please make sure you read the academic integrity section of the syllabus so you understand what is permissible in our programming projects. We want to remind you that we check your project against other students' projects and any case of academic dishonesty will be referred to the University's Office of Student Conduct (https://www.studentconduct.umd.edu/).

Web Accessibility (https://www.umd.edu/web-accessibility/)