|  |  |
| --- | --- |
| **CMSC 131** |  |
|  |

Assignment:     Project #2

Due Date:     Thursday 10/04, 11:00PM

Open/Closed policy:     OPEN

**Flags of the World**

****

Objective

This project will give you some practice with loops and methods, and also introduces the notion of having multiple classes working together in a single project.

Overview

The project consists of several classes working together -- most of them are provided for you -- only one class (the one named FlagMaker) will be written by you.  The final product will draw flags of different countries and at different sizes as selected by the input from the console.

**This project is considered "open". Please visit the course web page for information regarding the** [**open/closed policy for projects**](http://docs.google.com/openClosedPolicy.shtml) **of this course.**

Color Class

One of the classes  you will use for this project is provided in one of the available Java graphics packages.  It is the "Color" class, and it is part of a package called "java.awt".  That means that any class that wants to use Color objects must begin with the following statement at the top of the file:

 import java.awt.Color;

If you wanted to create your own Color object you could use a statement like the one below:

Color myColor = Color.BLACK;  // myColor now refers to the color "black"

There are other ways to generate just about any color you could imagine, but the simple syntax shown above works for the following built-in colors that are needed for the flags:  BLACK, BLUE, GREEN, ORANGE, RED, WHITE, YELLOW.  In real life some flags are made from different variations of these colors for example one flag might have a dark green and another might have a light green - for ease on this project we will use the built in green for both.

MyGrid Class

The MyGrid class has been written for you.  It is part of a package called "GridTools".  That means that any class that wants to use MyGrid objects must begin with the following import statement at the top of the file:

import GridTools.MyGrid;

A MyGrid object is a window that you can draw on.  In the center of the window there is a **rectangular** grid (twice as wide as it is high).  You can color in each region of the grid with any color you want.  Below is an example of an "empty" MyGrid object of size 8 (that means 8 high and 16 wide).



**Creating a DrawingGrid Object**

A MyGrid object is created in the example driver provided for you in your CVS repository. **You do NOT need to ever create a grid yourself in order to complete this assignment.** But if you ever wanted to create a grid for your own purposes, you could use a statement like this one:

MyGrid aGrid = new MyGrid(8);  //aGrid will refer to a 8 by 16 drawing grid

This statement will create a MyGrid object named aGrid where each of the 128 squares are already colored grey.

**Coloring the Squares on the Grid**

Once you have a MyGrid object, you can color in any of the squares you want using any color you want.  To do this, you use the method of the MyGrid class called setColor.  The setColor method of the MyGrid class takes three parameters.  The prototype for the method appears below.

public void setColor(int row, int col, Color color)

The parameters "row" and "col" specify the location of the square on the grid that you would like to color in; the parameter "color" refers to the color you want to use. Note that the first row is row number 0 (not row number 1), and the first column is column number 0.  For example, below is a picture of what you would see if you executed the code fragment that follows.

MyGrid aGrid = new MyGrid(6);  // creates the 6 by 12 MyGrid object

aGrid.setColor(3, 3, Color.RED);     // colors in the square at row 3, col 3, using red

aGrid.setColor(0, 2, Color.BLUE);    // colors in the square at row 0, col 2, using blue



If a particular square on the grid has already been colored, there is no harm in re-coloring that same square -- the previous color will simply be replaced with the new color.  This will be a very useful fact when you look for similarities between the flags so you can reuse code you have already written.

**Determining the size of the MyGrid object**

There is are two other methods of the MyGrid class that you will need.  Suppose you have a variable called "aGrid" that refers to an existing MyGrid, and you want to know how big it is.  You can use the methods "getHt" and "getWd" to find out.  For example, if the variable "aGrid" refers to an existing grid, the following statements could be used to store the dimensions of the grid in two helpful variables:

int height = aGrid.getHt();

int width = aGrid.getWd();

Be careful - DO NOT USE getWidth() and getHeight() - these are defined but will not give you the numbers you are looking for.

FlagMaker Class

OK, this is where YOU come in!  We have provided a skeleton for this class that you must complete.  There is basically just one method that you must fill in, although it is okay for you to create other methods in this class if you find it useful to do so.  You may find that writing additional methods can help to eliminate duplicative code.  Since we will not be creating any FlagMaker objects, any methods you choose to write in this class should be declared using the keyword  *"static"*.  Since these other methods should only be called from another method in the one class you are writing, it should also have the keyword private.

The method that you *must* write has the following prototype:

public static void drawFlag(MyGrid grid, int countryCode)

Remember that the two parameters (grid and countryCode) are provided to your method by whoever called it.  Those parameters contain the information that your method needs in order to do its job.

The parameter "grid" will refer to an existing MyGrid object that has already been created by someone else.  **Your method should not create a MyGrid object anywhere within the methods of this class**, it is already there!  Your method will draw on the grid that is passed in via this parameter.  You may assume that the grid starts off empty (all grey squares).

The parameter "countryCode" will be an integer.  The number should correspond to one of the country flags listed in the menu of the example driver provided for you in your CVS repository (also corresponding to the numbers of  [The Flag Information Pages](http://docs.google.com/InformationAndPictures.htm)).  If it is not one of these values, it should display "the error flag" - more description of this below.  If the height specified by the user is invalid (as described for that flag) for the flag corresponding to that country code, the error flag should be displayed.  If it is a valid countryCode and a valid height for that country, the drawFlag method should display the flag indicated by that countryCode in the grid that was passed as a parameter to the method.

The style for this flag must conform **exactly** to the examples and descriptions below using the basic colors of the Color class (even though some are not  exactly the accurate shade for the flag chosen).  Note that the size of the grid can be determined by calling the "getHt" and "getWd" methods of the MyGrid class.  The size of the grid dictates how big a flag you are supposed to draw (the flag must fill up the entire grid, as in the examples shown below.)  The size typed by the user when it is requested in the console corresponds to the height of the flag - the width is always twice the height.

**Examples and descriptions of flags**

For examples and further descriptions of the flags  [click here](http://docs.google.com/InformationAndPictures.htm). These are the exact style that you must use for drawing your flags. The hyperlinked file shows two examples of different sizes for each flag.   Your flag must conform **exactly** to the styles below.  The flag must take up the entire MyGrid, no matter what size the grid is. Your drawFlag method must be capable of drawing flags of any size that is appropriate for that country's flag (as defined here).  If the grid size given is not appropriate for the flag selected, an error flag should be drawn instead.  The smallest flag size allowed will be 4 high by 8 wide; if a size smaller than 4 is requested, the example driver creates a MyGrid object of size 4x8 in which you must display the error flag.  The maximum size for any flag is one with a height of 30; if a size larger than 30 is requested, an the example driver creates a MyGrid object of size 4x8 in which you must display the error flag.  These limits are taken care of in the exampleDriver and you do not have to check it again in your code (read the exampleDriver for more explanation) - the example driver simply passes an invalid country code so you know that the error flag should be displayed.

Public Tests

On the submit server you'll notice that there are three tests labeled "Public Tests". They are "testIndonesia6", "testLithuania6", and "testErrorFlag".  **You must pass these three public tests before your code will be tested on the remaining (release) tests.**  In other words, if you aren't passing the public tests, then your score on the remaining tests will be 0.

ExampleDriver Class

Please note that the ExampleDriver class is not really part of the project assignment!  It is only provided to help you test out your code.  In fact, when we test your FlagMaker class ourselves, we will be using a completely different driver class.  Feel free to write your own driver class to test your FlagMaker class a different way, if you want to.

Getting Started

Begin the project by "checking out" the project called " 131Fall18Proj2".  In order to help you get started, we have included a FlagMaker class with a skeleton version of the "drawFlag" method, which you must complete.

**If you write the project from scratch, without checking out the "131Fall18Proj2" files from your CVS repository, you will not be able to submit your work.**

Requirements

* You may not create any new classes on your own.
* You may not use any static variables unless they are declared "final". You may not use instance variables. (If you don't know what these are, don't worry about it.) If you write methods of your own you will probably need to pass the grid as an argument to those methods.
* You must use meaningful variable names and good indentation.
* You must used named constants where appropriate to assist in readability.
* You should not get creative and modify the project specifications. For example, do not add extra behavior or output.
* We use an automated system to grade projects and you will not pass our tests if you do not follow the project specifications precisely. **In particular, the coloring and sizes of your Flags must match the examples shown in**  [**the linked file**](http://docs.google.com/InformationAndPictures.htm) **EXACTLY.  Pay particular attention to how the edges where the color changes look.  You are not allowed to be "off-by-one-square" or deviate in any way from the examples.**
* Just in case you know what an array is -- you may not use arrays (or any Java collections of any kind) while implementing this project.
* Your program must terminate without using anything like *System.exit().  (*Don't worry if you don't know what that is.)
* You may assume that the user will type the correct type of input when requested.  For example, if an integer is requested, the user will not type a string.
* If the countryCode is not one that is given in the list, the Error Flag should be displayed.
* If the flag height requested is not appropriate for the countryCode given, the Error Flag should be displayed.
* If the flag height requested is not between 4 and 30, the main method that we have provided will create a 4 by 8 grid and pass an illegal country code to your method. This should cause your method to display the error flag on the 4 by 8 grid.

Submission

Submit your project from Eclipse by right-clicking the project folder and selecting "submit".  You may submit as many times as you want -- we we only grade the submission that scores the highest on the automated tests.  After you have submitted your project, you should visit the [submit server](http://submit.cs.umd.edu/Fall2018).  There you can obtain limited feedback about how well your project is performing.  The number of times you can run our tests on your project (before the due date) is limited.  **The earlier you begin working on the project, the more opportunities you will have to see how your project performs on our tests before the due date!**

Grading

90% of your grade on this project will be from the automated tests on the submit server (both public tests and release tests). 10% of your grade on the project will be based on using correct "style" for Java programming.

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