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

Project #7, Diagram System (Due Thu, Nov 18, 11:55 pm / Mon, Nov 22, 11:55 pm)

Overview

For this project you will practice two-dimensional arrays and interfaces. **For this project you MAY NOT discuss the project with classmates, but you may not implement code together and you may not share code (that includes student tests)**.

This project has two deadlines:

- **Thu, Nov 18, 11:55 pm** - Your code must pass the first 5 release tests. This is the only requirement for this deadline. We will not grade the code for style. This first part is worth **0.5%** of your course grade (NOT **0.5%** of this project grade). You can still submit late for this part. For this deadline you must implement the folloowing methods from the TwoDimArrayUtil.java class: isRagged, rotateTopOneRow, rotateLeftOneColumn, appendTopBottom. Make sure you provide empty methods (or throw an exception) for methods you are not planning to implement for the first deadline.
- **Mon, Nov 22, 11:55 pm** - Final deadline for the project. You can still submit late until Wed, Nov 24, 11:55 pm.

Overview

For this project you will implement classes that support displaying and animating diagrams similar to the ones you developed for the DrawingApp project. The video Sample Video (https://umd.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=58052992-5802-4fe1-b1d1-ab050159ad80) illustrates some of the functionality that will be supported by the classes you need to implement. The program we executed for this video is GraphicalExampleDriver.java (part of the system package).

After reading this description, import the code example Example.zip (Example.zip) and go over the README file you will find along with the code distribution. This code provides an example of what you need to do for this project.

Grading

- (82%) Release Tests
- (12%) Secret Tests
- (6%) Style

Code Distribution

The project's code distribution is available at DiagramSystem.zip (https://www.cs.umd.edu/class/fall2021/cmsc131-01XX-03XX/prot/projects/zipFiles/DiagramSystem.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 place your implementation for the **DrawingApp** project. Make sure you fix any problems associated with the DrawingApp project. Notice that just because you got a 100 in the DrawingApp project, it does not mean it is bug-free. The secret and release tests for this project could find

some of these bugs. If you failed some secret/release tests, a TA, during office hours, could provide information about the tests. You can resubmit your code and see release/secret tests results in the submit server for the Drawing App project.

- **gui** → A package (folder) providing code that supports the graphical display and animation of a diagram. You will use the methods provided by the GraphicalUtilities.java class. Do not modify anything in this package.
- **system** → A package (folder) where you will implement the classes associated with this project.
- **tests** → A package (folder) where you will find a file called **StudentTests.java** were you are expected to write tests for your program. **There are no public tests associated with this project.**

Specifications

You need to implement the classes that are part of the **system** package (except GraphicalDriverExample.java). A description of the classes can be found at javadoc (doc/index.html). You need to start with the TwoDimArrayUtil class (other classes depend on it) and continue with the HorizontalBars.java, VerticalBars.java and Flag.java classes. You can leave for last the CombineLeftRight.java and CombineTopBottom.java classes.

You can finish this project without ever running the Graphical utility methods that allow you to display a diagram graphically or that allow you to animate a diagram. If you provide the expected functionality for the classes in the **system** package, the graphical component will work as expected. You can run a Graphical demo by executing the GraphicalDriverExample.java class. Feel free to update that file with your own diagrams.

Before you start implementing this project, you need to copy to the **app** package the DrawingApp.java class you developed. Make sure you fix any problems that might exist with your code.

For style we will be focusing on the following items:

- Good variable names and indentation.
- Avoid code duplication when possible.

Student Tests

Test your code as you develop it, by writing tests in the **StudentTests.java** file. We will not grade your student tests, but you should write them as you only have a finite number of release tokes (you want to test as much as possible before doing a release test). You may not be able to get help during office hours unless you have student tests.

Requirements

- **Using the Arrays class is not allowed. You can use it only for student tests.**
- You can release test your project any time, but for this project you have 5 tokens in a 24-hour period.
- Do not use System.exit(0) in your programs.
- **For your code to compile in the submit server, you need to provide all the classes and the methods (or shells of them) you are expected to implement. You can make methods compile (even though you may not have implemented them) by adding the following statement**

  throw new UnsupportedOperationException("Not implemented");

- If the parameter to the isRagged method is null, the method will throw an IllegalArgumentException. Any error message is fine.
- You don't need to re-implement the DrawingApp class. You will use the class you developed for DrawingApp project. That class will generate the basic diagrams (e.g., Flag) for you.
- It is OK to use code you see in **Sample Driver** section.

- **You may not use ArrayLists or any other Collection class (e.g., maps) for this project. If you do, you will lose most of the points for the project.**
- Do not cut and paste the prototypes of methods from the Javadoc. If you do your code will NOT work as there will be character set problems with your code.

Sample Driver

The following driver (and associated output) illustrates the functionality of some of the methods you need to implement.

```
public class DriverExample {

	public static void main(String[] args) {
		char[][] array1 = { { 'A', 'B' }, { 'C', 'D' }, { 'E', 'F' } };

		String answer = "Original:\n";
		answer += getTwoDimArrayString(array1);

		answer += "\nAfter one top rotation:\n";
		TwoDimArrayUtil.rotateTopOneRow(array1);
		answer += getTwoDimArrayString(array1);

		answer += "\nAfter two top rotations:\n";
		TwoDimArrayUtil.rotateTopOneRow(array1);
		answer += getTwoDimArrayString(array1);

		answer += "\nAfter left rotation:\n";
		TwoDimArrayUtil.rotateLeftOneColumn(array1);
		answer += getTwoDimArrayString(array1);

		char[][] array2 = { { 'G', 'H' }, { 'I', 'J' }, { 'K', 'L' } };

		char[][] array3 = { { 'M', 'N' }, { 'O', 'P' }, { 'Q', 'R' } };

		answer += "\nAfter left right append:\n";
		answer += getTwoDimArrayString(TwoDimArrayUtil.appendLeftRight(array2, array3));

		answer += "\nAfter top bottom append:\n";
		answer += getTwoDimArrayString(TwoDimArrayUtil.appendTopBottom(array2, array3));

		answer += "\nAnimation of horizontal bars:";
		int maxRows = 6, maxCols = 6, bars = 3, animationType = 1;
		char color1 = 'R', color2 = 'G', color3 = 'B';
		HorizontalBars horizontalBars = new HorizontalBars(maxRows, maxCols, bars, color1,
				animationType);
		answer += "\nOriginal:\n";
		answer += getTwoDimArrayString(horizontalBars.getBoard());
		for (int i = 1; i <= 3; i++) {
			answer += "\n\nAnimation Step: " + i + "\n";
			char[][] nextAnimationStep = horizontalBars.nextAnimationStep();
			answer += getTwoDimArrayString(nextAnimationStep);
		}

		System.out.println(answer);
	}

	private static String getTwoDimArrayString(char[][] array) {
		if (array == null) {
			throw new IllegalArgumentException("Invalid parameter getTwoDimArrayString
		}

		String answer = "";
```

```
                for (int row = 0; row < array.length - 1; row++) {
                        answer += Arrays.toString(array[row]) + "\n";
                }
                answer += Arrays.toString(array[array.length - 1]);

                return answer;
        }
}
```

Output

```
Original:
[A, B]
[C, D]
[E, F]
After one top rotation:
[C, D]
[E, F]
[A, B]
After two top rotations:
[E, F]
[A, B]
[C, D]
After left rotation:
[F, E]
[B, A]
[D, C]
After left right append:
[G, H, M, N]
[I, J, O, P]
[K, L, Q, R]
After top bottom append:
[G, H]
[I, J]
[K, L]
[M, N]
[O, P]
[Q, R]
Animation of horizontal bars:
Original:
[R, R, R, R, R, R]
[R, R, R, R, R, R]
[G, G, G, G, G, G]
[G, G, G, G, G, G]
[B, B, B, B, B, B]
[B, B, B, B, B, B]

Animation Step: 1
[R, R, R, R, R, R]
[G, G, G, G, G, G]
[G, G, G, G, G, G]
[B, B, B, B, B, B]
[B, B, B, B, B, B]
[R, R, R, R, R, R]

Animation Step: 2
[G, G, G, G, G, G]
[G, G, G, G, G, G]
[B, B, B, B, B, B]
[B, B, B, B, B, B]
[R, R, R, R, R, R]
[R, R, R, R, R, R]
```

```
Animation Step: 3
[G, G, G, G, G, G]
[B, B, B, B, B, B]
[B, B, B, B, B, B]
[R, R, R, R, R, R]
[R, R, R, R, R, R]
[G, G, G, G, G, G]
```

Submission

- Submit as usual.
- Make sure you submit often so you have a backup in the submit server.
- It takes time for a project to be tested in the submit server, especially the day the project is due; keep this mind.

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/)