Title : Rectangle of Rectangles
Modules : 07 Advanced Arrays & 08 Applications
Point Value : 50
Deadline : Due before 11pm on Thursday, November 3rd

Topics Covered

Two dimensional arrays, nested loops and conditionals, file I/O. [You absolutely may not use ArrayLists or other Java Collection classes in this assignment.]

Overview

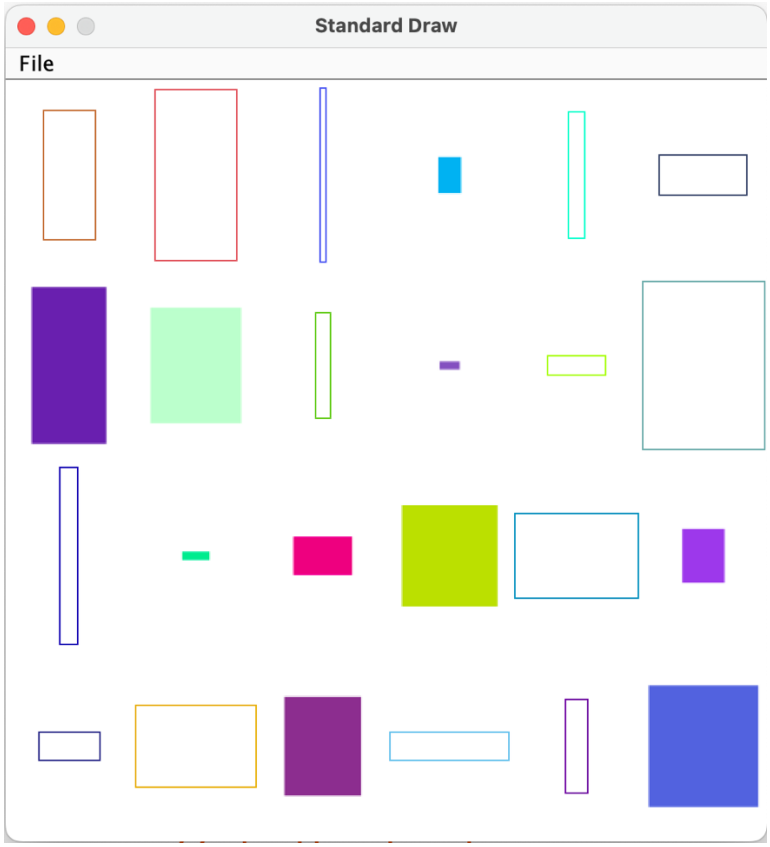
In this project you will create and display rectangular grids (2-dimensional arrays) of colorful rectangles, using the `StdDraw` library from earlier in the semester (Project 1). With this program you can make common graphical patterns such as a checkerboard, and display the graphics with different types of animations. All the java code files and sample input/output files you will need for this project are included in the **project4.zip** starter file.

The main idea is to create a 2D array (grid) that will be mapped to (displayed on) a default (square) `StdDraw.java` display window. Each element of the array (cell) is a `Rectangle` object with several attributes that determine its color, size, whether it is solid (filled) or outline form, and the location of its center in the display window. By iterating through the grid in different cell orders, your project can display some animations of the patterns.

`Rectangle` is a class that we have written for you, along with a main method that demonstrates its operation. This class (and the project code you write) relies on the `StdDraw.java` library to visually display the objects. However, it can also create a text (String) version of each `Rectangle` object that can be written to or read from a plain text file.

The provided demo code (main method in `Rectangle.java`) creates a random 4 x 6 grid of shapes, saving the pattern to a file called `random.txt` and displaying the cells in standard row order in the drawing window. The `Rectangle.java` code, along with `StdDraw.java` for convenience, and some sample grid files are included in the starter zip file for this project. Below is a screenshot of the completed display that results from running the `Rectangle` main method. Your first task is to run this code yourself and read through its documentation (Javadoc comments) to be sure you understand how it works, and subsequently how to use the `Rectangle` class to achieve the project goals

below. (Feel free to change the Random seed (10) to another value, or remove it entirely to generate different random patterns!)



The `Rectangle` class stores a value for each of red, green, and blue to represent the color of each shape. We use the Java `Color` class directly in one of the `Rectangle` class constructors, so you might want to look through its documentation. You will also need to use the Java Color constant `Color.LIGHT_GRAY` as part of your project implementation as described below.

Fun Fact: **RGB** stands for Red, Green, Blue, and integer values for each color in the inclusive range [0, 255] (2^8 possibilities) can be used in combination to make 16,777,216 (2^24) different colors. When all three values are set to 0, the result is black. But when all three are maxed out at 255, the result is white! There are many websites you can use to get the RGB values for a wide range of colors.

The format of each plain text pattern file (output result or used for input) is that the first line contains two integers: the numbers of rows and columns in the grid, in that order. Each subsequent line contains the data for one Rectangle, as provided by the toString()

method in the provided Rectangle class. Cells are written to the output file in standard row order, starting with array location [0][0]. Note that the Rectangle class also contains a constructor which when passed a line of a pattern file, will create a corresponding Rectangle object that can then be stored into a 2D array of type Rectangle. You will need to use this when reading in an pattern file for some of the required operations.

Program Operation

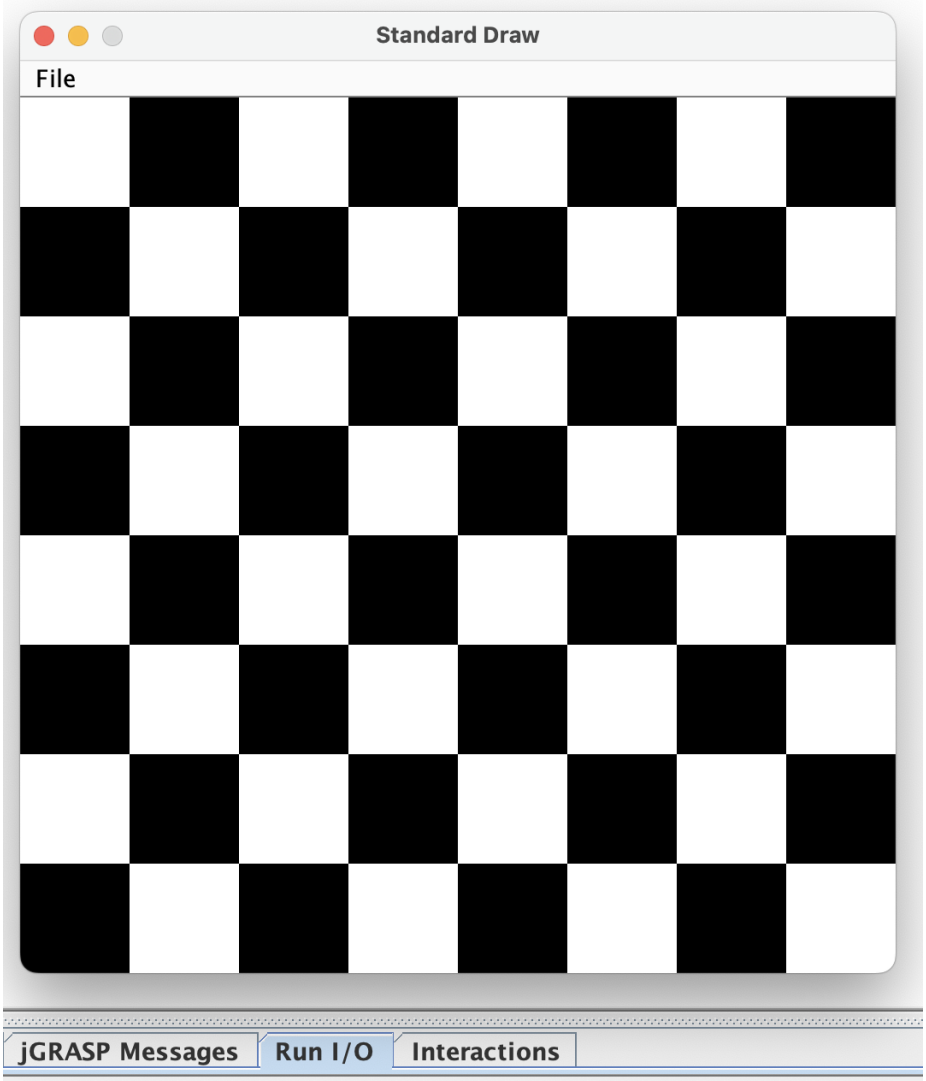
Unlike recent projects that had a menu-driven interface, this program is required to do three main tasks in sequence: Checkerboard, Snake, Spiral. Each operation requires some user input that must be prompted for exactly as displayed below and read from the keyboard. You must also follow these conventions so that your project runs as expected:

- Before each operation, the StdDraw window must be cleared and have its background Color set to Color.LIGHT_GRAY.
- Use StdDraw.pause(200) after each cell is drawn so that the "animations" for the different cell drawing orders run slowly enough to be evident.

Checkerboard

This operation requires the creation of a square grid where each cell alternates between white and a color the user designates by inputting values for red, green and blue. The user must also specify the size of the checkerboard. Your program is required to build the 2D array that represents the checkerboard pattern, display it on the drawing window, and print the details of the grid to a plain text output file. The name of the file must be `checkerboard#.txt` where "#" is substituted by the actual size of the board.

For example, this screenshot captures a classic 8x8 black and white checkerboard generated and displayed by the program, as well as the prompts and input values used to create it. The output for this pattern would be saved in file `checkerboard8.txt` by the program and is included in the project zip file for your reference.



Snake

This operation gets user input for the name of an input file and reads the pattern file into a grid. It then displays the graphic in a "snake" like animation based on column order, iterating from the top left corner to the bottom of the first column, then up the second column, down the third column, etc. See the video below for a demo.

Spiral

This operation is very similar to Snake, but the animation order is a little trickier. The bottom right corner of the grid must be the first one displayed. From there, iterate across the bottom row in the left (west) direction. Next display up the first column (north), then across the first row from left to right (east), and finally down the last column (south),

stopping on the second to last row. Now repeat the west-north-east-south rotation, spiraling gradually inwards towards the center of the pattern. As you proceed, you must not repeat any cells, but continue to smoothly "turn the corners", until there are no more cells to display. See the video below for a demo.

Here are the exact prompts you must use in your program, shown with sample input.

```
 ----jGRASP exec: java Project4Solution
Enter checkerboard size: 8
Enter RGB values, each [0,255]: 0 0 0
Enter snake input filename: snake.txt
Enter spiral input filename: transition.txt
```

We have recorded a video of two complete sample runs to illustrate the expected features of your program. All the sample files we've used are included in the project zip file, as well as some extra ones.

Project Implementation

You must write the entire project file, naming it Project4.java. Here are some import statements to get you started:

```
import java.util.Scanner;
import java.io.PrintWriter;
import java.io.IOException;
import java.io.FileInputStream;
import java.awt.Color;
```

Make sure that all three java files (Project4.java, StdDraw.java and Rectangle.java) are in the same folder. You should compile StdDraw and Rectangle into this folder as well so that your Project4 implementation will be able to use them.

We'll be learning how to define our own class types like Rectangle in the next module (09). In the meantime, you can simply use the provided Rectangle.java class as-is. You must not change anything in the provided Rectangle.java file! We will grade your project solutions with our original version of this Rectangle class, and any changes you make are likely to result in huge point deductions, possibly even 0 credit for a non-compiling program.

You are expected to design and implement helper methods in your solution, not put all the code into main! Continue to fully document these with Javadoc comments. We strongly suggest writing methods to read a file and create the corresponding grid (2D array), to display a grid in basic row order, to write a grid to a file, to create a checkerboard pattern, to display a pattern in snake order, and to display a pattern with the spiral animation. Remember to inspect the main method in the provided Rectangle class for examples of how to do many of these tasks.

As an additional hint, the statement

```
grid[r][c] = new Rectangle(scan.nextLine());
```

will read a line of input with a Scanner called `scan`, create a new `Rectangle` from the data in the line, and store it into a 2D array of type Rectangle called `grid` in cell location row `r`, column `c`.

General Project (Hard) Requirements:

- **You will earn a 0 for any code that does not compile!**
- **You absolutely may not use ArrayLists or any other Java Collection classes in this assignment.**
- Your solution must be named **`Project4.java`** (capitalization matters!).
- Your file(s) must have a javadoc style class comment that explains briefly the purpose of the program, the author's name, JHED, and the date.
- The program must be fully checkstyle-compliant using our course required `check112.xml` configuration file. This means that the checkstyle audit completes without reporting any errors or warnings.
- You must also use good style with respect to class/method/variable names, etc.
- Submit **only the `.java` file named `Project4.java`** to the Project 4 assignment on Gradescope before the deadline date. We will grade with our own version of the `Rectangle.java` file and `StdDraw.java` file (exactly as given to you). Please DO NOT submit any `.txt` files.
- You can resubmit as many times as you want before the deadline. We will only grade your final (most recent) submission.

Grading Breakdown: The following points are awarded only if your submitted files compile, since **assignments that don't compile earn a 0 grade!**

- [39] Implementation of required features
- [5] Appropriate use of methods
- [6] Style & submission