Project 1: Mondrian Art

Deadline: due before 11:00 pm on Thursday Sept. 22nd

Point Value: 50

Topics Covered (Modules 01 & 02)

Sequential algorithms, numerical data types, arithmetic operations, simple input, Sedgewick & Wayne StdDraw drawing library.

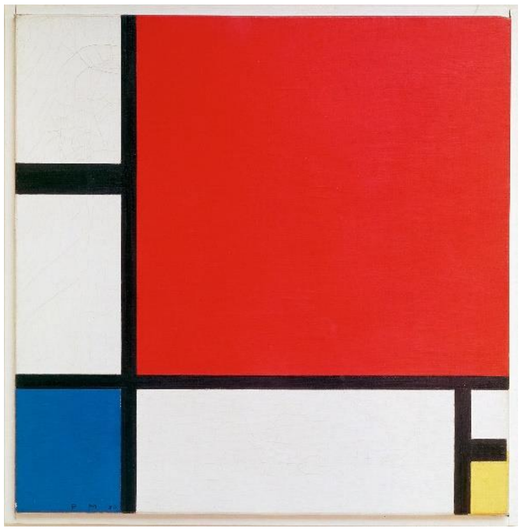
Setting Things Up

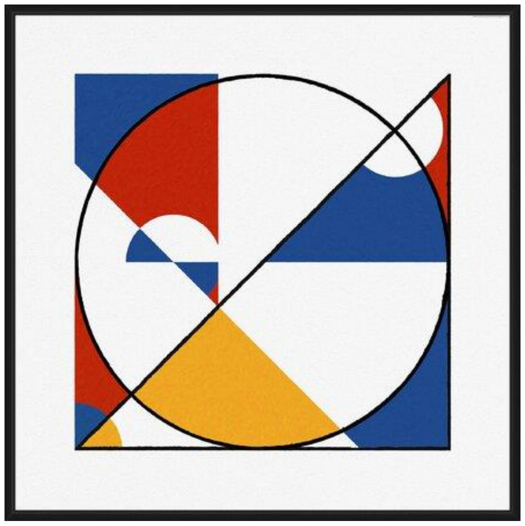
For this project, you will utilize the Sedgewick & Wayne drawing library which is a Java file named StdDraw.java. (We're using it during class in Week 2.) To download the file:

- Go to Canvas and on the left menu click on “Modules”, then “Getting Started”. Under "Programming Tools" click to load the “Drawing Library” item.
- Using a right-click, download the Java file (i.e. the drawing library named StdDraw.java) into the same folder where you plan to save this project’s Java file.
- Click on the link named ‘documentation’ to view the documentation for the library.
- We have created a **starter program file** for you with placeholders to add the code needed to solve this problem. **Right click on Project1.java to download the code**, making sure to store it in the same folder where you saved the StdDraw.java drawing library.

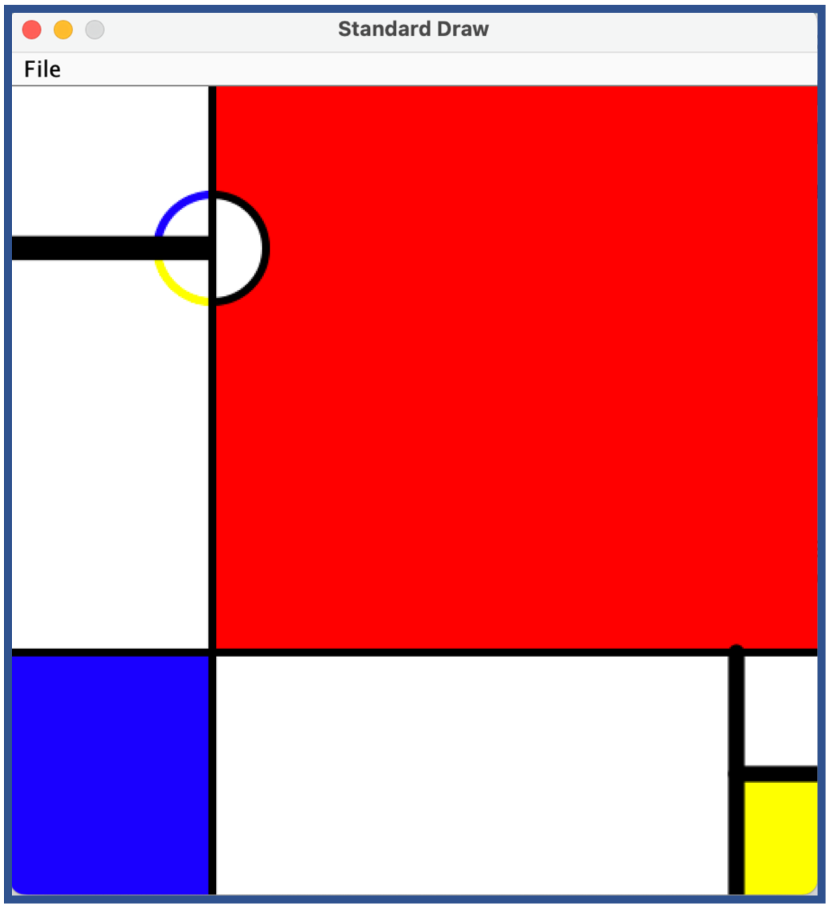
Project Description

This project will enable users to create a customized version of an art piece along the lines of Dutch artist Piet Mondrian. The artwork is largely inspired by his piece *Composition II in Red, Blue, and Yellow* (below left), with additional elements from *Mondrian Circle VI* (below right).





The custom abstract art piece your Java program will produce must have the same elements as this design (excluding the outer border on the window):



The program will enable the user to create variations based on required input to indicate the relative sizes of these four composition elements:

1. The width of the lower left block (blue), 25% in the above example
2. The height of the lower left block (blue), 30% in the above example
3. The height of the upper left block (white), 20% in the above example
4. The width of the lower right block (yellow), 10% in the above example

The following requirements must also be satisfied in order for the produced art to conform to our design:

- The radius of the circle must be one third the height of the upper left block.
- The height of the lower right block (yellow) must be half the height of the blue block
- Your artwork must be a square. Using the default display size of 512 x 512 is fine, but you can make it larger if you want.
- The thinnest lines should be drawn with pen ~~width~~ *radius* set to 0.01
    - The middle thickness should be twice that (lines for the lower right blocks)
    - The widest line should be three times the smallest (top left horizontal line)
    - Note that there is sometimes a rounding effect at the end of a line – you are expected to eliminate this for the widest line (ie, no bump into the circle), and encouraged to eliminate it for others also. (Hint: use a rectangle!)
    - The thinnest width should be used to draw the circle arcs and the cross pieces defining the blue and red blocks.

Below is a sample run to demonstrate the required prompts for input and how the user is expected to provide those values. You must use this exact wording, spelling, and spacing. You can assume the input will be valid.

```
----jGRASP exec: java Project1
Enter percent for blue block width: 25
Enter percent for blue block height: 30
Enter percent for top left block height: 20
Enter percent for bottom right block width: 10
```

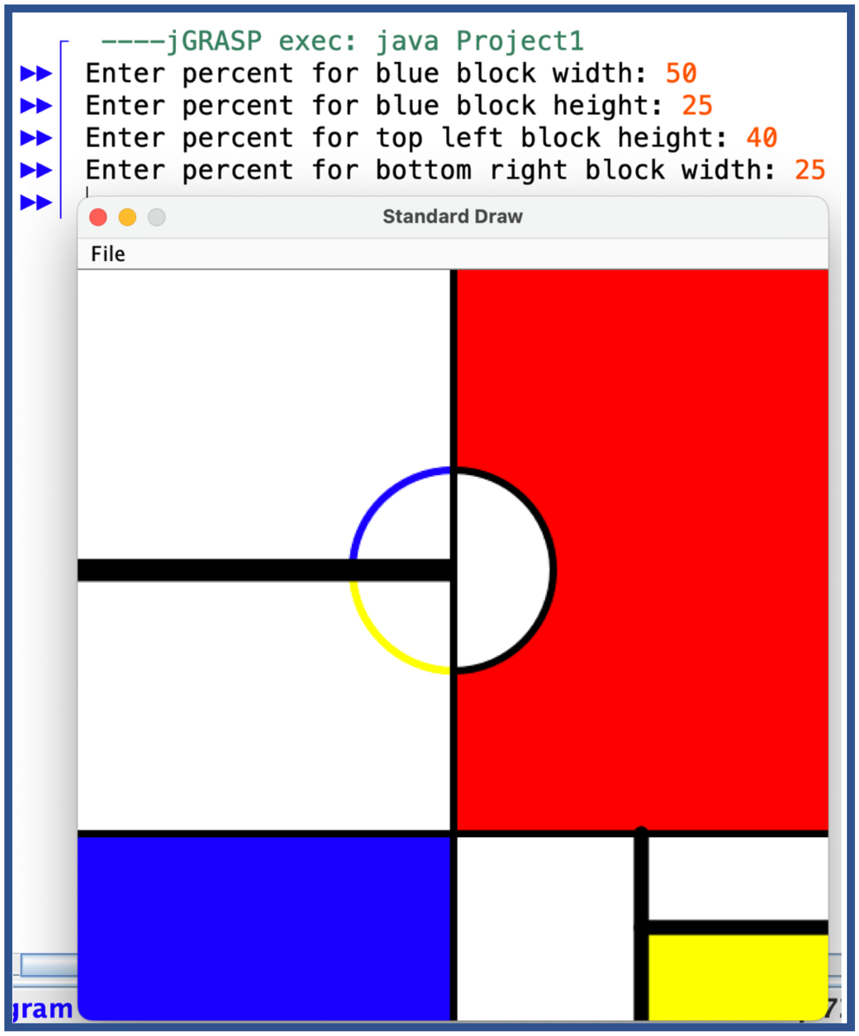
Getting Started & Hints

We have created a **starter program file** for you with placeholders to add the code needed to solve this problem. **Right click on Project1.java to download the code**, making sure to store it in the same folder where you saved the StdDraw.java drawing library.

- Many colors are available as pre-defined values in StdDraw.java. While using the StdDraw.java library you may find it useful to consult this documentation.Here

are the color expectations for clarity, but variations in tone are permitted if you want to make yours more "Mondrian-like".
  - the bottom left block and the top left quadrant of the circle are blue
  - the top right block is red
  - the bottom right block and bottom left quadrant of the circle are yellow
  - the lines are black
  - all else is white

- Another sample run showing the input and resulting artwork is provided here to further your understanding of the input values:



```
 ----jGRASP exec: java Project1
Enter percent for blue block width: 50
Enter percent for blue block height: 25
Enter percent for top left block height: 40
Enter percent for bottom right block width: 25
```

- You will need to calculate the center of the blocks to be drawn, based on their sizes and positions relative to the whole canvas.
- You will also need to calculate the start and end points of the lines.
- The order in which you draw the various shapes will determine the results!

- We strongly suggest drawing this out on paper first to get the geometry of the problem correct.

General Project (Hard) Requirements:

- You will receive a 0 for any code that does not compile!
- Your solution must be named Project1.java (capitalization matters!)
- Your file must have a javadoc style comment for the class definition that describes 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 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 Project1.java (NOT Project1.class) to the Project 1 assignment on Gradescope before the deadline date.
- Please note the late policies as stated on the course syllabus. You can resubmit as many times as you want before the deadline - we will only grade your final (most recent) submission, and will penalize this last submission for lateness if applicable, according to the policy on the syllabus.

Grading Breakdown:

- [44] Functionality (only if it compiles)
    - -.5 for minor errors (eg, off by one)
    - -1 to -3 for each computational error, partially missing requirement, extra output, etc.
    - -3 to -5 for each missing feature or requirement
- [4] Style (only if it compiles), based on checkstyle errors, variable names, and meaningful comments
- [2] Submission & (no) run-time errors (only if it compiles)
- IMPORTANT: Assignments that don't compile get a 0 grade!