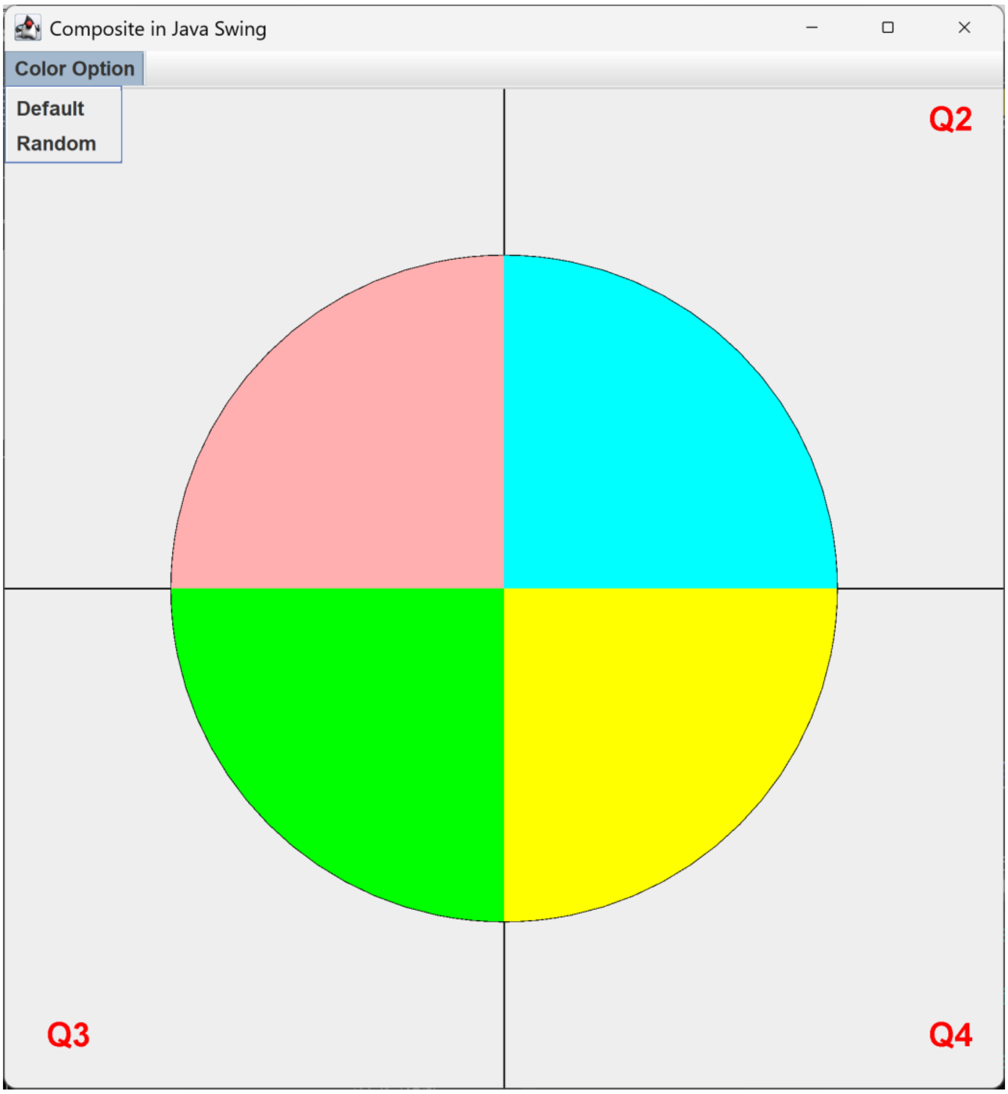
CSC319 Advanced Java (2/2024)

Workshop #1: Java Swing Application

In this workshop, we will observe the application of the Composite and Template Method design patterns as implemented in the Swing Framework. Swing, as it stands today, is a legacy GUI framework of the Java language. While it is still fully functional, it has been largely deprecated (replaced by JavaFX) due to its lack of support for some of the modern UX/UI features, e.g. responsive design. However, the framework is still quite valuable as a learning tool. In this workshop, in addition to discussing how Composite has been implemented into the framework, we will also attempt to modify the design to incorporate the Strategy design pattern into our Swing application.

Here is what we want…



Conceptually, writing a Java Swing application is fairly straightforward. If there is ever a challenge, that tends to lie in how we can make the Swing GUI window behaves exactly the way we'd like it to. This requires mastering the framework and it takes some time.

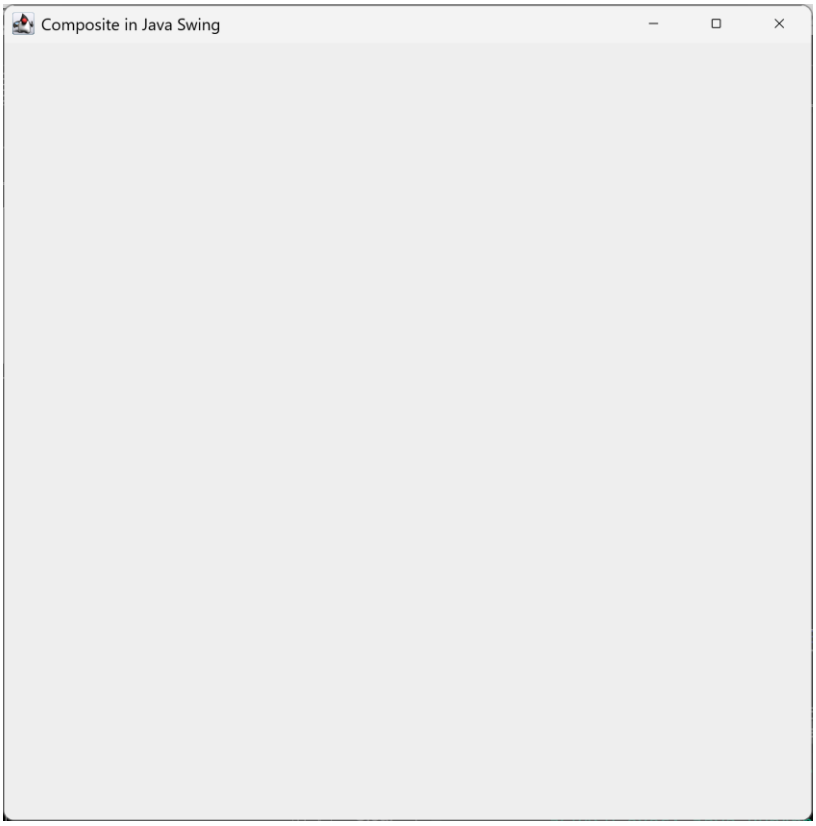
It should also be noted that Swing built on top of Java's Abstract Window Toolkit (AWT), a predecessor to Swing. Thus, it has access to and makes use of some AWT components.

The steps in using the Swing framework typically involve:

1. Creating the JFrame object to link Java's virtual environment to the underlying operating system resources. Normally, there is exactly 1 JFrame object per a Swing-based application.
2. Initialize the JFrame's container as the top-level, main container for the application.
3. Build subcontainers (typically, outside of JFrame) using JPanel. With each JPanel's subcontainer, we can add the needed Swing GUI components as well as other relevant AWT components.
    a. Each subcontainer can define its own rendering characteristics, i.e. layouts, styles, typography, etc.
4. Add the subcontainers into the main contain to have them rendered on the main application window.
    a. The main application window is taken from the contents of the JFrame object's container.
5. Repeat Steps 2-4.

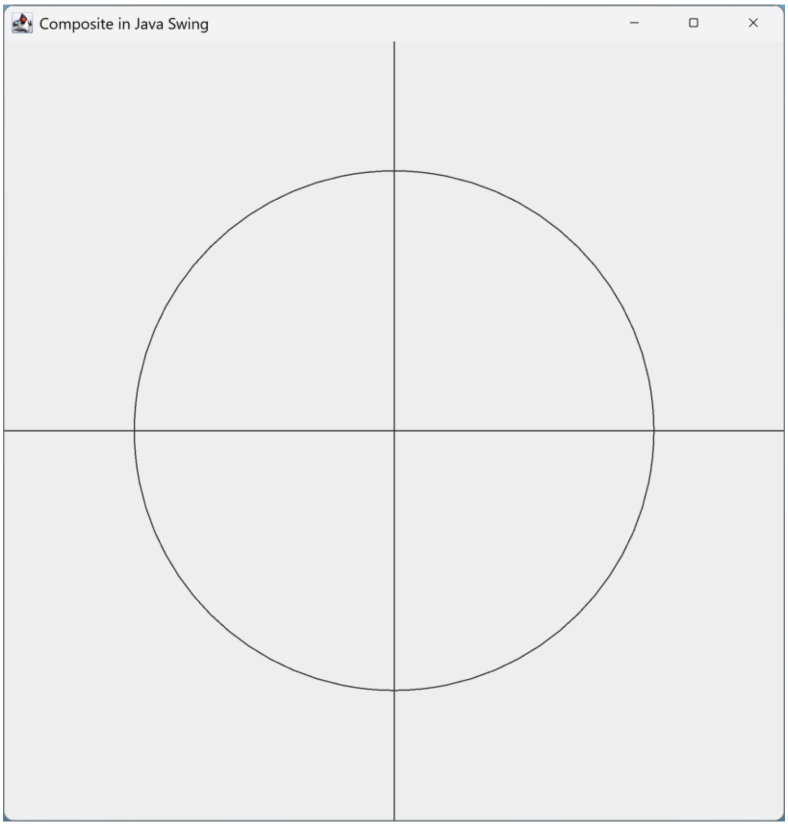
In this workshop, we are going to build the Swing GUI window in iterations as follows.

Iteration 1: Getting the JFrame object up and initializing the top-level container.

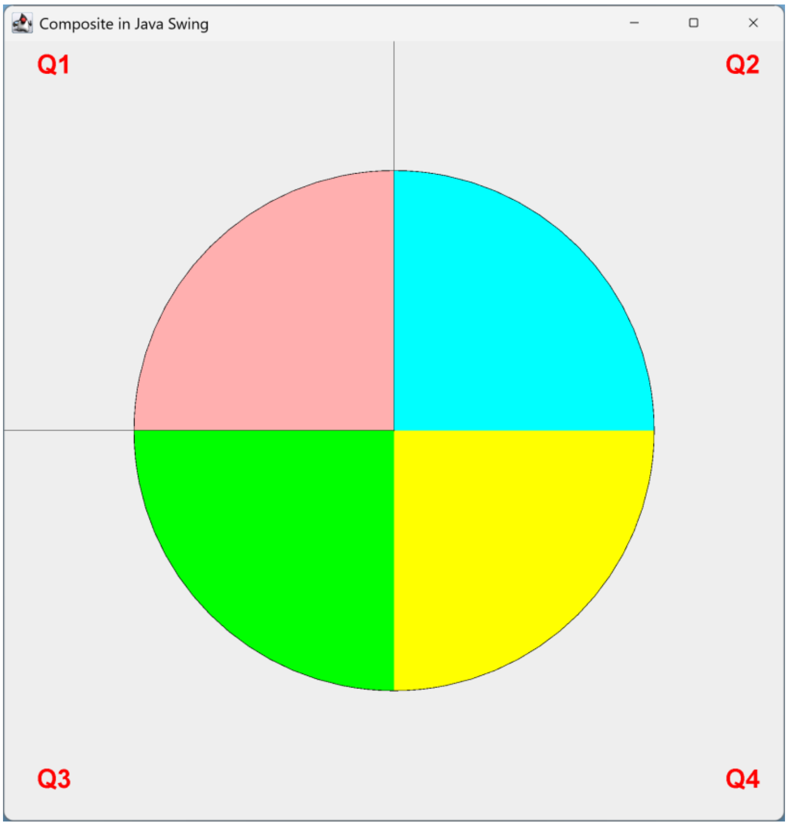


Iteration 2: Building and adding the main content. In our workshop, the main content is a JPanel object that acts as a canvas for drawing graphics. To draw a graphics, we will have to implement the *paintComponent* method of the canvas object (i.e. JPanel object).

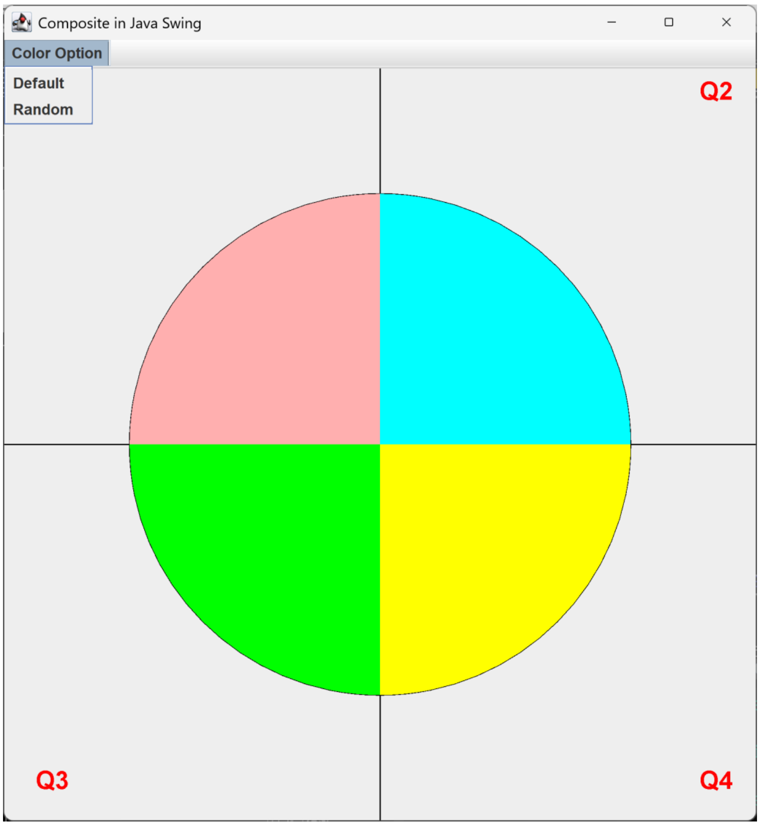
At this point, when completing the tasks successfully, we'd have already encountered 2 design patterns at work: Composite and Template Method.



Iteration 3: Enhance the existing canvas with Grid layout and Color.



Iteration 4: Let the user decide whether the colors we painted on the 4 quadrants of the circle should be (a) the default colors, or (b) the random colors. For this iteration, we will design the solution to specifically implement the Strategy design pattern. The user will interact with the Menu options to decide on which of the 2 strategies will be executed.


Composite in Java Swing
Color Option
Default
Random
Q2
Q3
Q4