Bridge Pattern Report

Name: Zhumasheva Akerke

Group: SE-2413

1. Introduction

As a product for demonstrating the Bridge Design Pattern, the chosen example is flowers. Flowers (such as roses and tulips) can have different colors (red, white, yellow). This example was chosen because flowers are a universal object that can easily be extended: we can add new flower types or new colors without modifying existing code.

2. Explanation of the Bridge Pattern

The Bridge Pattern decouples abstraction (e.g., "Flower") from its implementation (e.g., "Color"), allowing both to vary independently.

- Abstraction defines high-level logic.
- Implementation contains the details that may change.

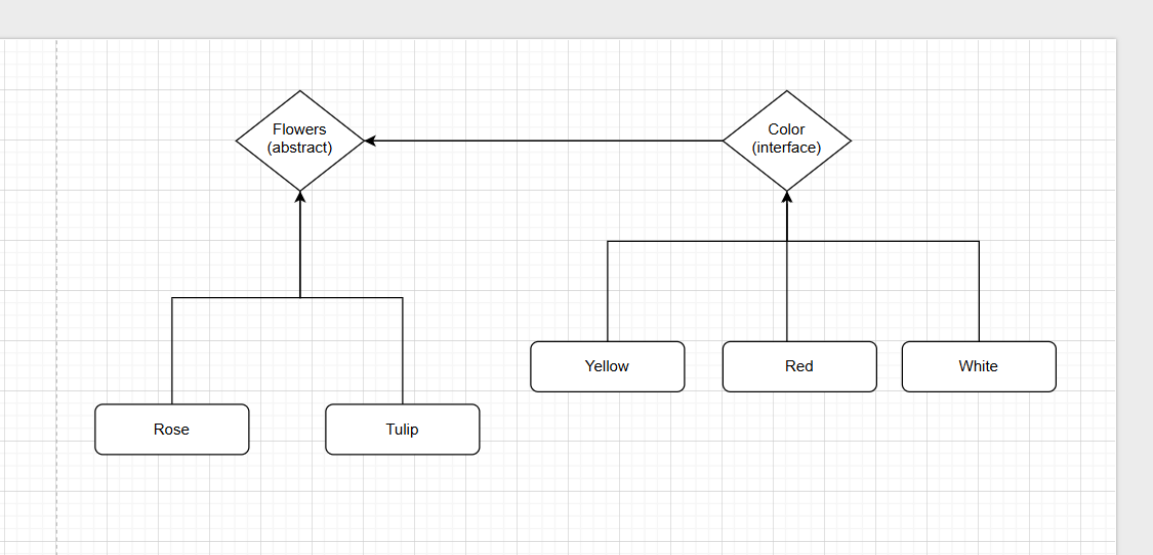
In this project:

- Abstraction: the Flowers abstract class (and its subclasses Rose, Tulip).
- Implementation: the Color interface (with implementations Red, White, Yellow).

This makes it possible to combine any type of flower with any color without changing the existing code.

3. Project Implementation

UML Diagram



Class Explanation Flowers — abstract class that holds a reference to Color.

Rose and Tulip — specific flower types (abstractions).

Color — interface that defines toGiveColor().

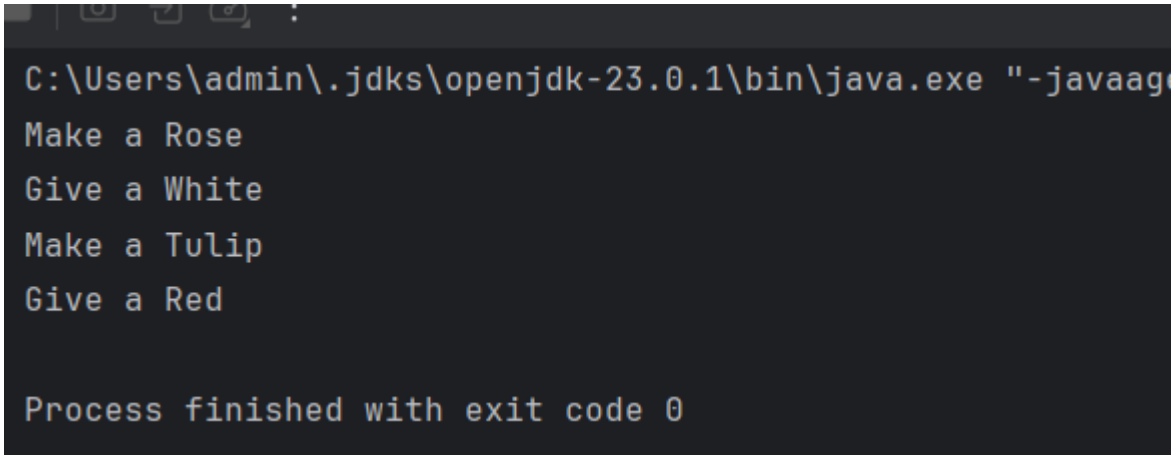
Red, White, Yellow — concrete implementations of Color.

Main — client class that creates different combinations of flowers and colors.

Example program output:

new Rose(new White()) → prints “Make a Rose” + “Give a White”.

new Tulip(new Red()) → prints “Make a Tulip” + “Give a Red”.



5. Conclusion

The project demonstrates the Bridge Design Pattern with the application of Clean Code principles. By separating abstraction from implementation, it is possible to extend both flower types and colors independently. This design improves readability, maintainability, and scalability of the code.

Link: https://github.com/xzxz8741/Bridge-pattern