README

Introduction

The provided class have been redesigned and refactoring to provide maximum reusability, and extensibility and eliminates as many dependencies as possible in the design

Changes

Interfaces Introduced:

MathOperation Interface – This interface was created to help do the math jobs is possible to add new math operations without messing with existing code.

Executable Interface - Introduced to encapsulate the execution logic. This now allows us to easily add new features later, such as remote execution and access control.

Class Modifications:

ObjectQueue - Refactored to use the Executable interface, transforming it so it can handle a variety of tasks. By doing this it won't need to rely on specific classes and made it more flexible in managing different types of operations

Flipper – Now implements the Executable interface, encapsulating up its logic for reversing strings to follow the Command pattern.

Summation and LogFunction – Both classes have been updated to implement the MathOperation interface, and also the Executable interface. So that they can be executed by the ObjectQueue and are prepared for potential extensions.

Priority - No changes required. The enum serves its purpose without violating any design principles.

Classes Removed:

No classes have been removed in this refactoring process. Instead, their responsibilities have been realigned to fit within the new design pattern, making them more coherent and maintainable.

Design Pattern

Command Pattern – By using the Executable interface, all information/details needed to execute an operation in the ObjectQueue.

Strategy Pattern – Used in the MathOperation. The MathOperation interface acts like a game plan for doing math calculations. It use's different math methods. This way, we can easily change or add new math strategies without disrupting the rest of the system.

Conclusion

The refactoring has improved the design of the application.