Пиманкин Данил, студент группы 931921

Отчет по лабораторной работе №2

1. Постановка задачи

На основе паттерна Наблюдателя, написать программу для слежения за состоянием выбранного файла.

Ограничимся двумя характеристиками:

1. Существует файл или нет;

2. Каков размер файла.

Программа будет выводить на консоль уведомление о произошедших изменениях в файле.

Существует несколько ситуаций для наблюдаемого файла

1. Файл существует , файл не пустой - на экран выводится факт существования файла и его размер.

2. Файл существует, файл был изменен - на экран выводится факт существования файла, сообщение о том что файл был изменен и его размер.

3. Файл не существует - на экран выводится информация о том что файл не существует.

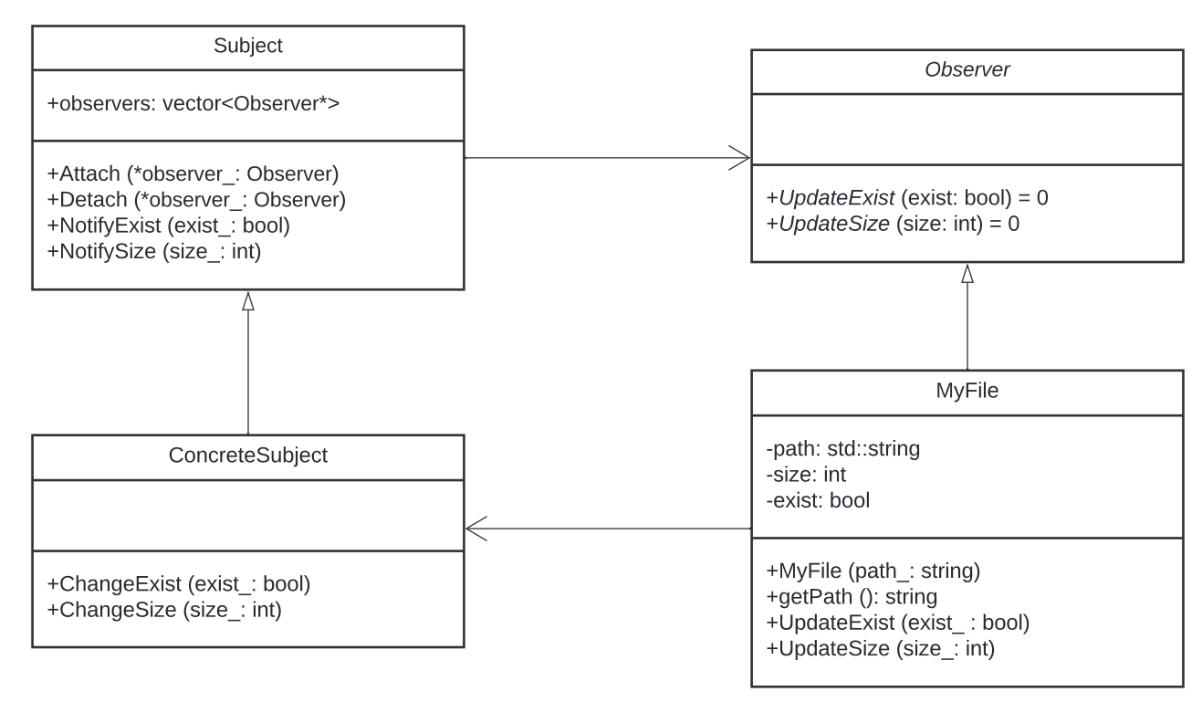
В главной программе создаем объект для отслеживания состояния конкретного файла,

затем определяется объект наблюдатель, после чего связываем Наблюдателя с Источником. Можно использовать бесконечный цикл, в котором будем обновлять состояние объекта Источника каждые 100 миллисекунд

(например так std::this\_thread::sleep\_for( std::chrono::milliseconds( 100 ) );).

2. Предлагаемое решение

Наблюдатель — это поведенческий паттерн проектирования, который создаёт механизм подписки, позволяющий одним объектам следить и реагировать на события, происходящие в других объектах. Предлагаемая архитектура реализации паттерна представлена на uml-диаграмме.



1. Архитектура наблюдателя

Класс Subject отслеживает всех наблюдателей класса Observer и предоставляет возможность добавлять или удалять наблюдателей. Кроме того, именно класс Subject отвечает за обновление наблюдателей при любых изменениях. Внутри объекта Subject хранится список ссылок на объекты наблюдателей. Когда в Subject будет происходить важное событие, он будет проходиться по списку подписчиков и оповещать их об этом, вызывая определённый метод объектов-наблюдателей.

Класс ConcreteSubject является реальным классом, который реализует Subject. Этот класс является сущностью, изменение которой повлияет на другие объекты. Класс Observer представляет интерфейс, который определяет метод, который должен вызываться при каждом изменении. MyFile выполняет изменение своих данных и выводит изменение на консоль в ответ на оповещение, пришедшее от Subject. Эти классы следуют общему интерфейсу Observer чтобы Subject не зависел от конкретных классов наблюдателей.

В главной программе создаются объекты ConcreteSubject и MyFile, при этом MyFile регистрируется на обновления в ConcreteSubject.

3. Коды программ

Коды программ размещены на https://github.com/danilpimankin/TRPO/tree/main/Lab\_2-develop

* ConcreteSubject.h – класс ConcreteSubject с реализацией.
* MyFile.h - класс MyFile с релизацией в MyFile.cpp.
* Observer.h – класс-интерфейс Observer.
* Subject.h – класс Subject с реализацией в Subject.cpp.
* main.cpp – главный файл с основной программой.

4. Инструкция пользователя

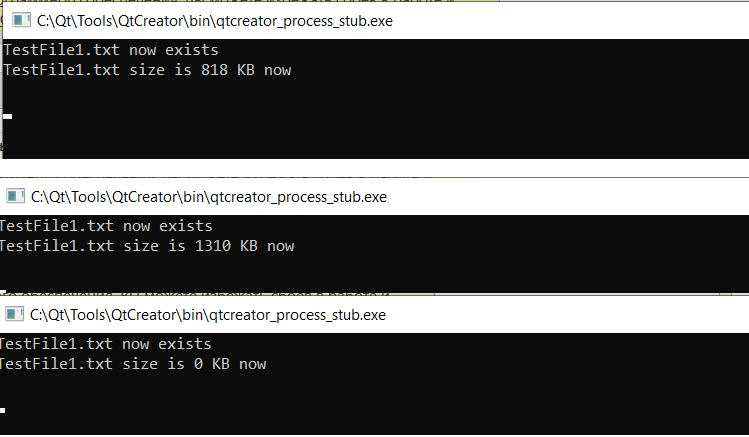
Чтобы начать отслеживание файла, необходимо ввести его полное название, включая путь (либо только название файла, если файл находится в папке проекта). Пример формата ввода: “C:\\Users\User\Documents\Lab\_2\Testfile1.txt”. После ввода начнется отслеживание файла и его состояние будет отображаться на консоли. Чтобы остановить отслеживание, нажмите любую клавишу. После этого можно указать новый отслеживаемый файл и начать отслеживание снова.

5. Тестирование

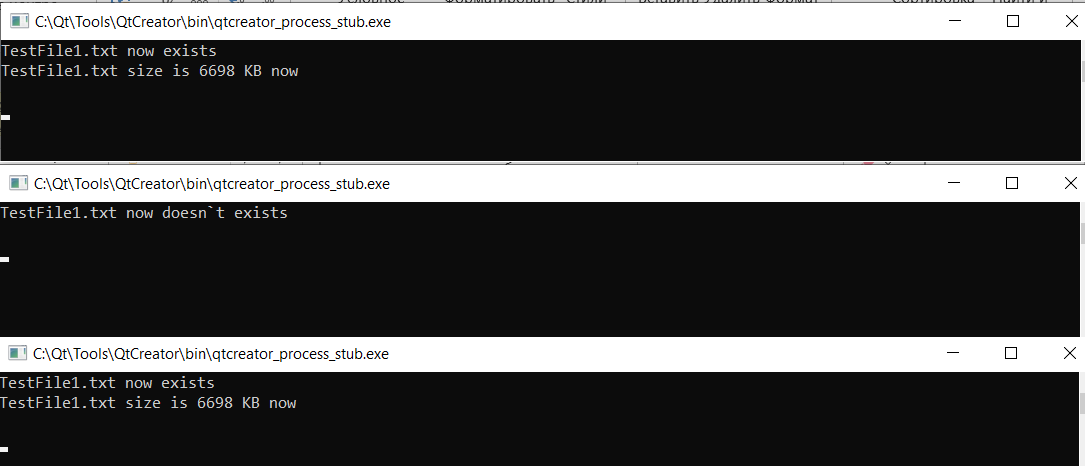
Описание тестирования приведено в приложенном файле “Тестирование лаб\_2.xlsx”. Результаты тестирования представлены на скриншотах.



Тест кейс 1

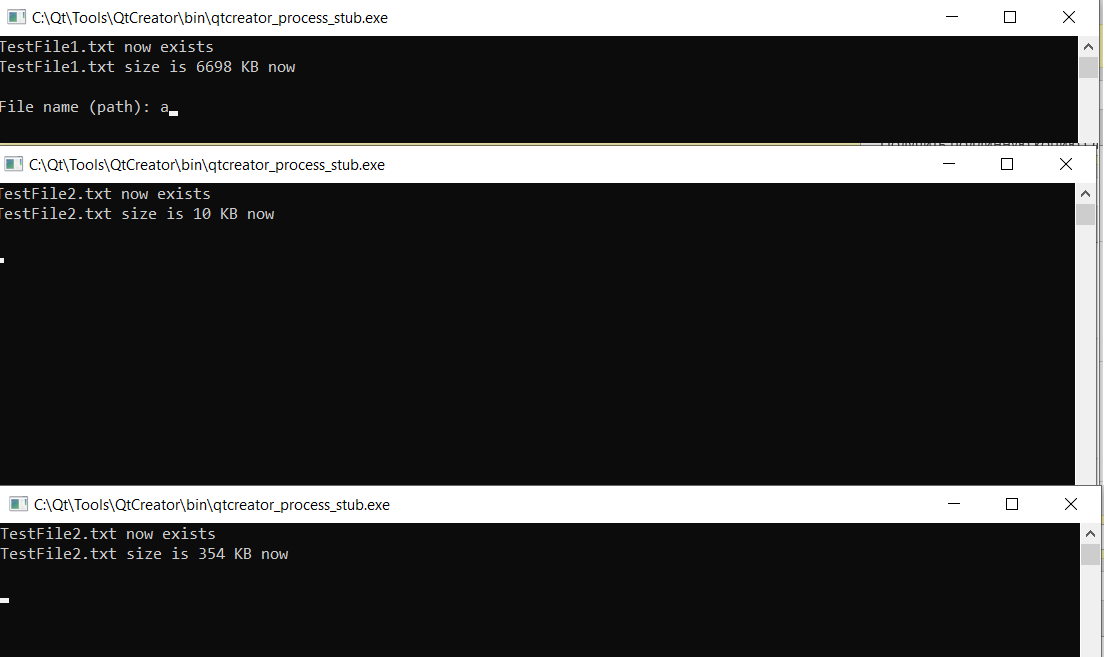


Тест кейс 2



Тест кейс 3



Тест кейс 4

Тест кейс 5

Тест кейс 6