**Рынк Даниил Витальевич группа 931921**

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.

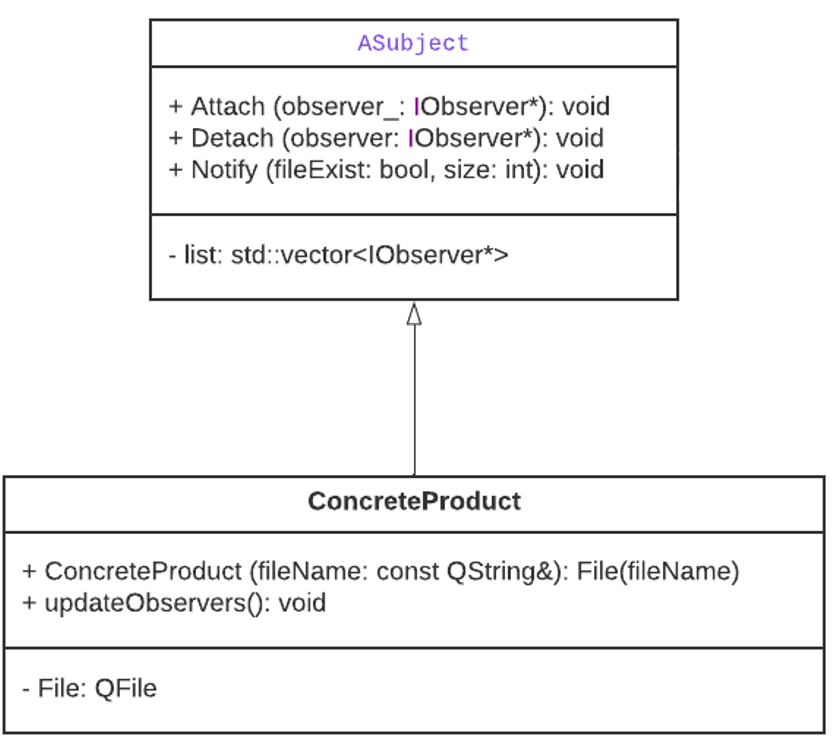
1. Предполагаемое решение  
     
   Наблюдатель- это поведенческий паттерн проектирования, который создаёт механизм подписки, позволяющий одним объектам следить и реагировать на события, происходящие в других объектах.

Для реализации нам понадобится класс который будет взаимодействовать со всеми наблюдателями – **Subject**. И сам класс наблюдателей **IObserver**.

**Subject** – класс, который отслеживает всех наблюдателей и предоставляет возможность добавлять или удалять их. Именно этот класс отвечает за обновление наблюдателей при любых изменениях.

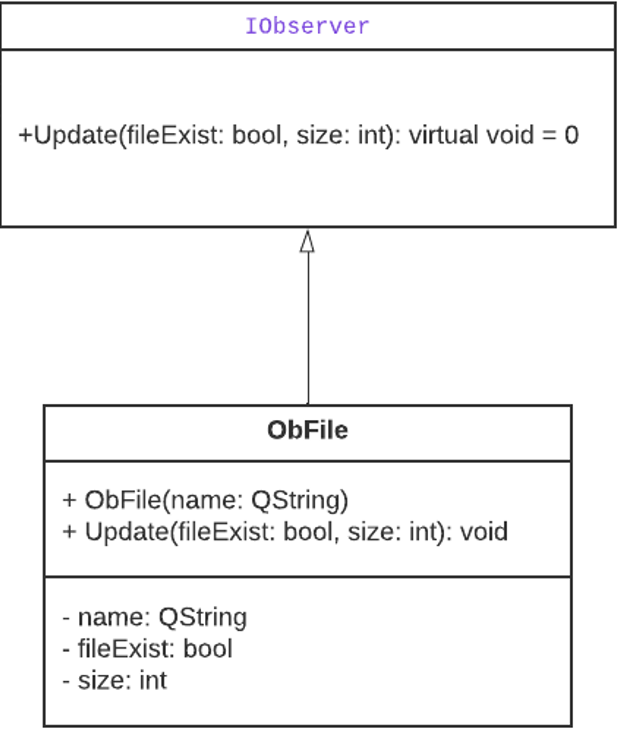
Он предоставляет наблюдателям методы, чтобы они могли добавить или убрать себя из списка.

**ConcreteProduct –** Класс, который реализует Subject.Он наследуется от Subject и будет изменять данные, что приведет к обновлению информации в наблюдателях.

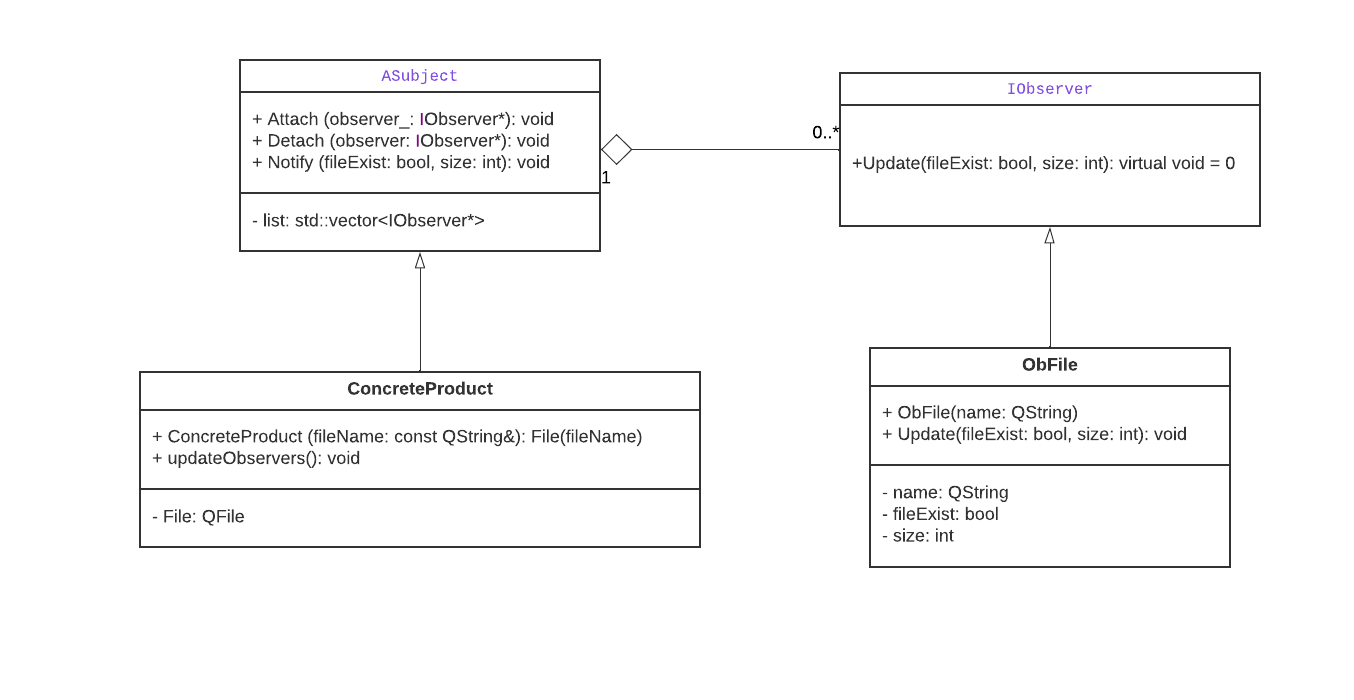


**IObserver –** абстрактный класс наблюдателя. Он предоставляет наследникам метод, который вызывается при каждом изменении.

**ObFile –** наследуется от IObserver. Он выполняет обновление информации, пришедшей от ASubject



**Итоговая UML диаграмма:**



1. Коды программ

Все файлы находятся в репозитории: [https://github.com/rynkdesign/TRPO\_lab](https://github.com/rynkdesign/TRPO_lab1)2

А именно:

observer.h - Наблюдатель

observer.cpp  
subject.h – Описание субъекта за которым мы будем наблюдать

subject.cpp

test.txt – Сам файл за которым мы будет наблюдать

main.cpp – Тестовый пример отработки программы

1. Инструкция пользователя

На данный момент мы можем использовать программу в качестве наблюдателя за файлом формата “.txt”. На **вход** мы подаём файл формата “.txt”, который размещаем в директории сборки и подключаем к нему наблюдателя.

Для начала мы копируем полное имя нашего файла.  
QString fileName = “test.txt”;

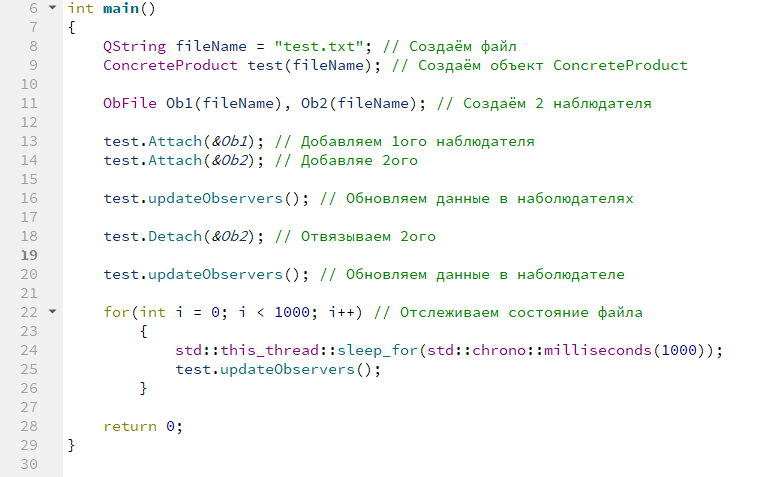
Теперь по этому имени мы можем создать наблюдателя, который будет следить за нашим файлом.  
ObFile Ob1(fileName);  
  
Теперь мы можем создать конкретный объект, который будет добавлять/удалять или обновлять информацию наблюдателям.

ConcreteProduct test(fileName);

Добавляем наблюдателя в список подписчиков обновлений:  
test.Attach(&Ob1);  
  
Затем мы можем использовать эту функцию, для обновление данных в наблюдателях:  
test.updateObservers();  
  
После использования этой функции на **выходе** мы получим информацию в консоли, которая будет отображать статус нашего файла с периодичностью в 1 секунду.  
  
Если понадобится, то мы можем отвязать наблюдателя от обновлений:  
test.Detach(&Ob1);

1. Тестирование

Для тестирование реализован пример в файле main.cpp.



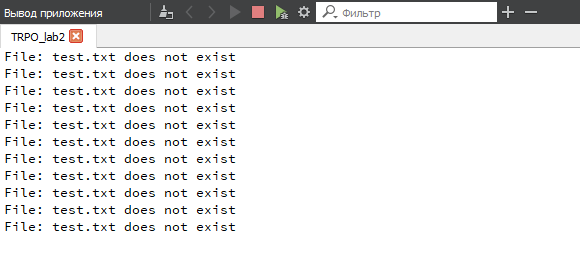
Мы создаём файл, за которым будет следить 2 наблюдателя. Обновляем данные в них и мы увидим 2 сообщения в консоле, затем отвязываем 2ого наблюдателя и снова обновляем данные, увидев теперь только 1.

Далее мы начинаем работу с нашим оставшимся Наблюдателем. Мы производим манипуляции с файлом “test.txt” вручную и постоянно обновляем информацию о нём, которая выводится в консоль.

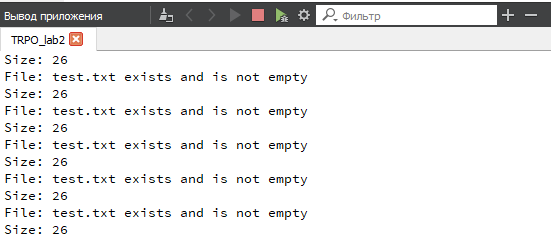
**Ситуации тестирования:**

Для тестирования помещаем файл “test.txt” в папку сборки программы и производим манипуляции с ним.

1. Если в данной директории файл “test.txt” отсутствует, то программы с периодичностью в 1 секунду выдает в консоль соответствующее сообщение: “File: Test.txt does not exist. “.

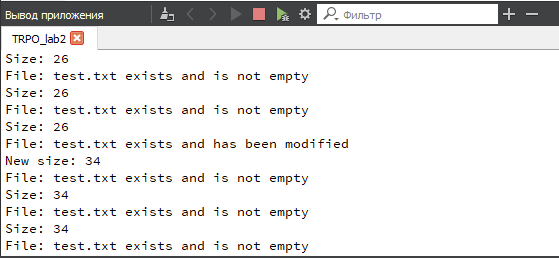
  
2. Если перед запуском программы файл “test.txt” находится в папке сборки и файл не пустой, тогда программа с периодичностью в 1 секунду выводит в консоль информацию о том, что файл существует и он не пустой и размер файла.   
(“File: test.txt exists and is not empty

Size: 26”)



3. Если во время работы программы изменить файл “test.txt” находится в папке сборки, тогда программа с периодичностью в 1 секунду выводит в консоль информацию о том, что файл существует и был изменён и показывает его новый размер.   
(“File: test.txt exists and has been modified

New size: 34”)



4. Если перед запуском программы файл “test.txt” находится в папке сборки, но файл пустой, тогда программа ничего не выдаёт в консоль, т.к. такая обработка не предусмотрена заданием.