**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.

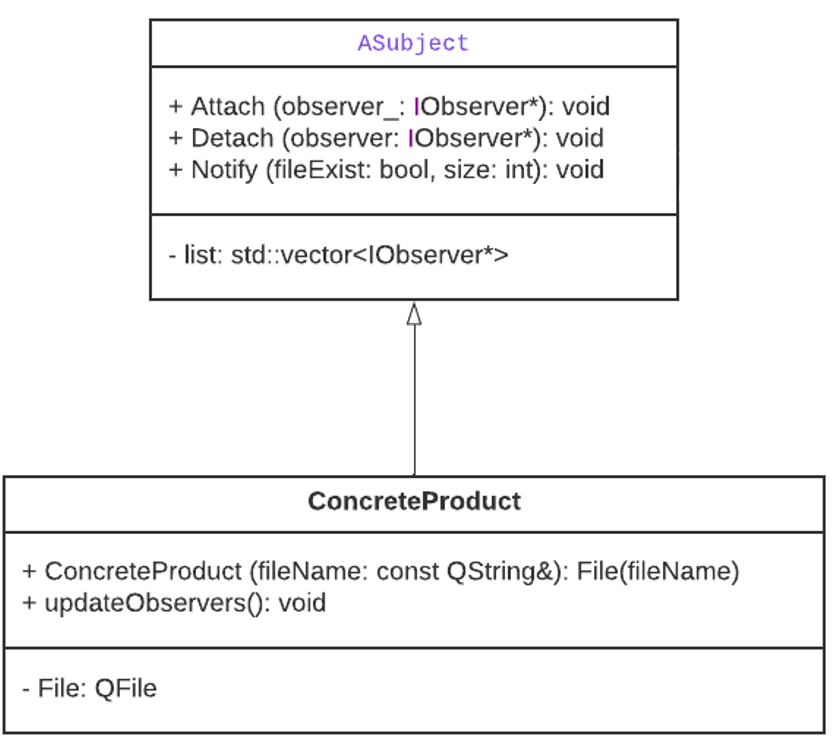
1. Предполагаемое решение  
     
   Наблюдатель- это поведенческий паттерн проектирования, который создаёт механизм подписки, позволяющий одним объектам следить и реагировать на события, происходящие в других объектах.

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.

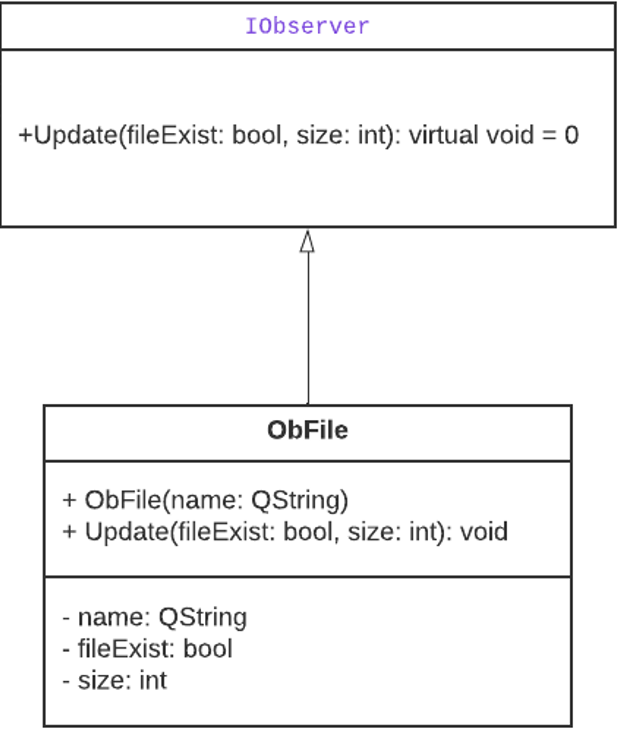
Он предоставляет наблюдателям методы, чтобы они могли добавить или убрать себя из списка.

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

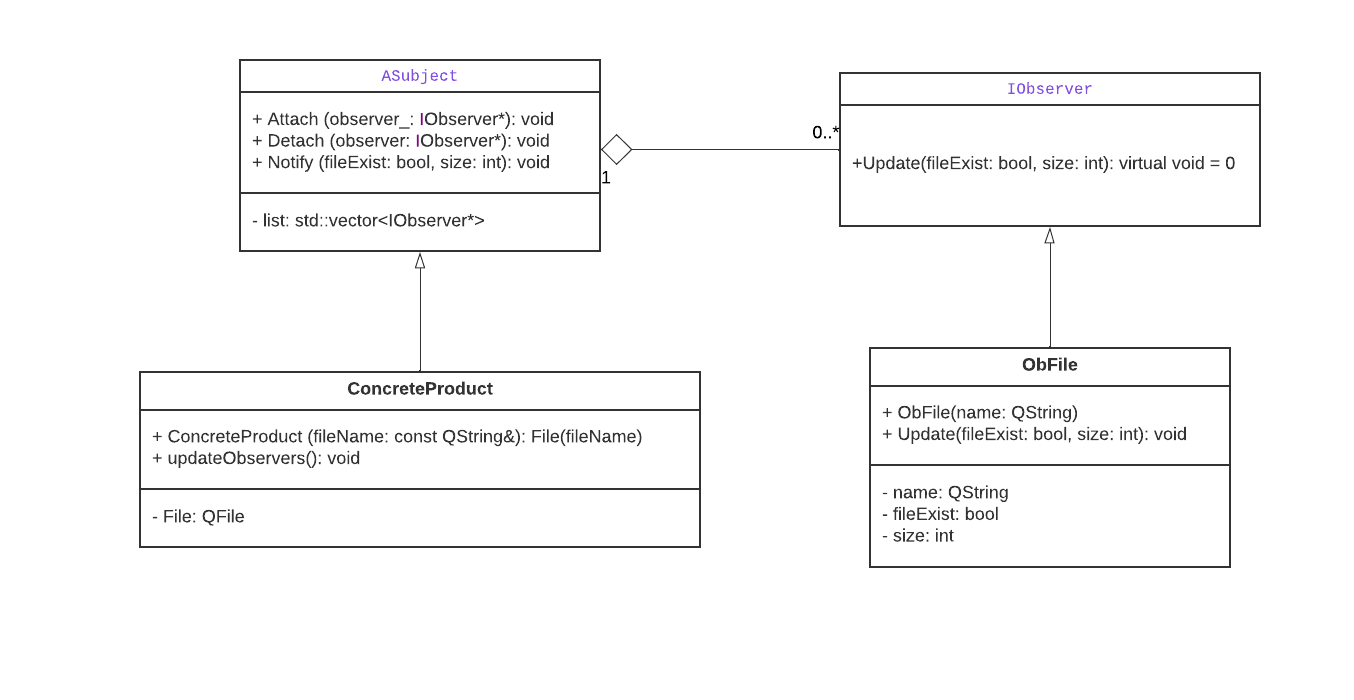


**IObserver –** абстрактный класс наблюдателя. Он предоставляет наследникам метод, который вызывается при каждом изменении.

**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, слежение за определённым файлом реализовано в виде хард кода:

Необходимо в файле main.cpp ввести имя файла (который заранее находится в директории сборки программы) тем самым проинициализировать filename нужным файлом.

В нашем примере файл называется “test.txt”

QString fileName = "test.txt";  
  
Далее предполагается бесконечный цикл, который с периодичностью в 1 секунду обновляет состояние этого файла и выводит их в консоль.

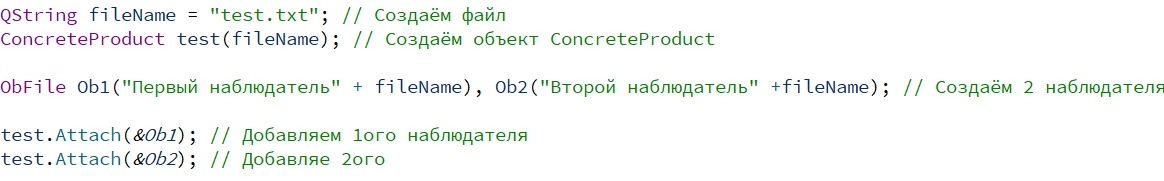
Состояния файла предусмотренные программой:  
- Файл не существует (Или его нет в директории)  
- Файл существует и не пустой + показывает его размер  
- Файл изменили + показывает его новый размер  
  
(Состояние о том, что файл существует и пустой намеренно не предусмотрен, т.к. не обговаривается в задаче)

То есть после указания имени файла программа начинает следить за файлом, а вы в это время самостоятельно можете изменять или удалять этот файл и получать соответствующие обновления в консоли.

1. Тестирование

Для тестирование реализован пример наблюдателя в файле main.cpp.

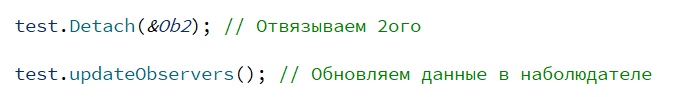
Для начала протестируем добавление/удаление и общую работу с наблюдателями.

Создадим наш объект и добавим к нему 2 наблюдателя подписав каждый из них.  


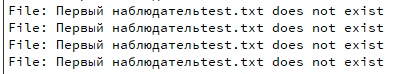
Теперь обновим информацию.  


Получим на вывод информацию по файлу от обоих наблюдателей.

  
  
Затем отвяжем 2ого наблюдателя и ещё раз обновим данные.



Получим на выход теперь только первого наблюдателя, который дальше и будет работать.

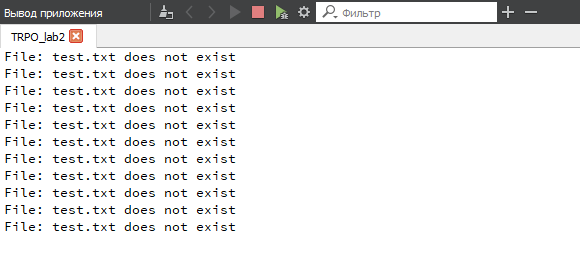


Тем самым мы протестировали добавление, удаление и обновление самих Наблюдателей, а теперь разберем ситуации тестирования конкретно с файлами.

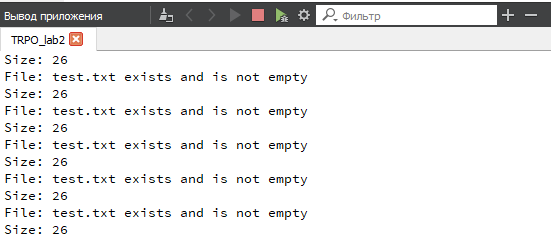
**Ситуации тестирования с файлами:**

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

1. Тестирование   
   Если в данной директории файл “test.txt” отсутствует (то есть мы запустили программу без файла в директории, либо удалили файл во время работы программы), то программы с периодичностью в 1 секунду выдает в консоль соответствующее сообщение: “File: Test.txt does not exist. “.

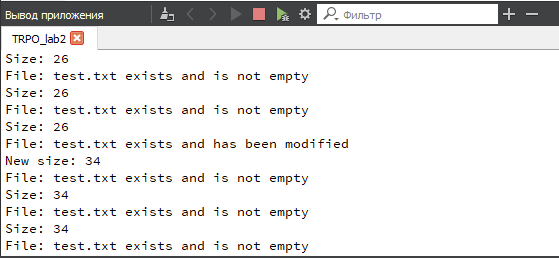
  
2. Тестирование   
Если перед запуском программы файл “test.txt” находится в папке сборки и файл не пустой, тогда программа с периодичностью в 1 секунду выводит в консоль информацию о том, что файл существует и он не пустой и размер файла.   
(“File: test.txt exists and is not empty

Size: 26”)



3. Тестирование  
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

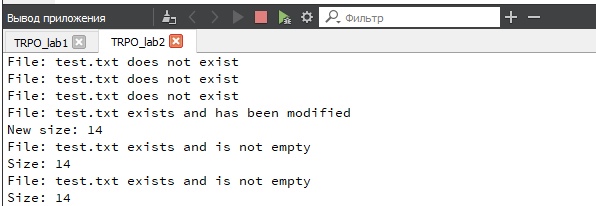
New size: 34”)



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.

5. Если файл был в директории перед запуском, но пустой, а затем мы решим

заполнить его каким-то данными. То вывод будет такой:



6. Если затем мы удалим наш файл, то консоль выведет это:

