

Часть 1.

1. Постановка задачи

На основе паттерна наблюдателя, было необходимо написать программу для слежения за состоянием выбранного файла.

Всего нужно было следить за двумя характеристиками, а именно: существует ли файл, и каков и его размер.

Программа должна выводить в консоль уведомление о произошедших изменениях в файле.

Существует лишь 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.

2. Предполагаемое решение

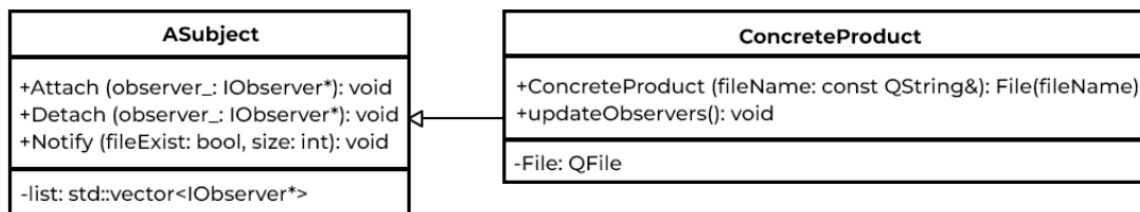
По своей сути, наблюдатель - это поведенческий паттерн проектирования, создающий некий механизм подписки, который позволяет некоторым объектам следить и реагировать на события, которые происходят в других объектах.

Для реализации будет необходим класс, взаимодействующий со всеми наблюдателями, имя этого класса - Subject. Также нам понадобится класс самих наблюдателей - IObserver.

Subject это класс, который отслеживает всех наблюдателей, а также предоставляет возможность удалять или добавлять их. Именно он отвечает за обновления наблюдателей при любых изменениях. Этот класса предоставляет

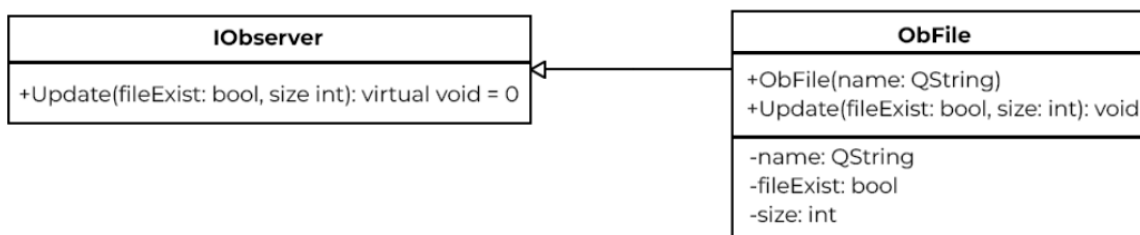
наблюдателям методы, которые им необходимы для добавление или удаление себя из списка.

ConcreteProduct это класс, который реализует класс Subject. ConcreteProduct наследуется от Subject, и, в свою очередь, будет изменять данные, соответственно, в наблюдателях обновится информация.

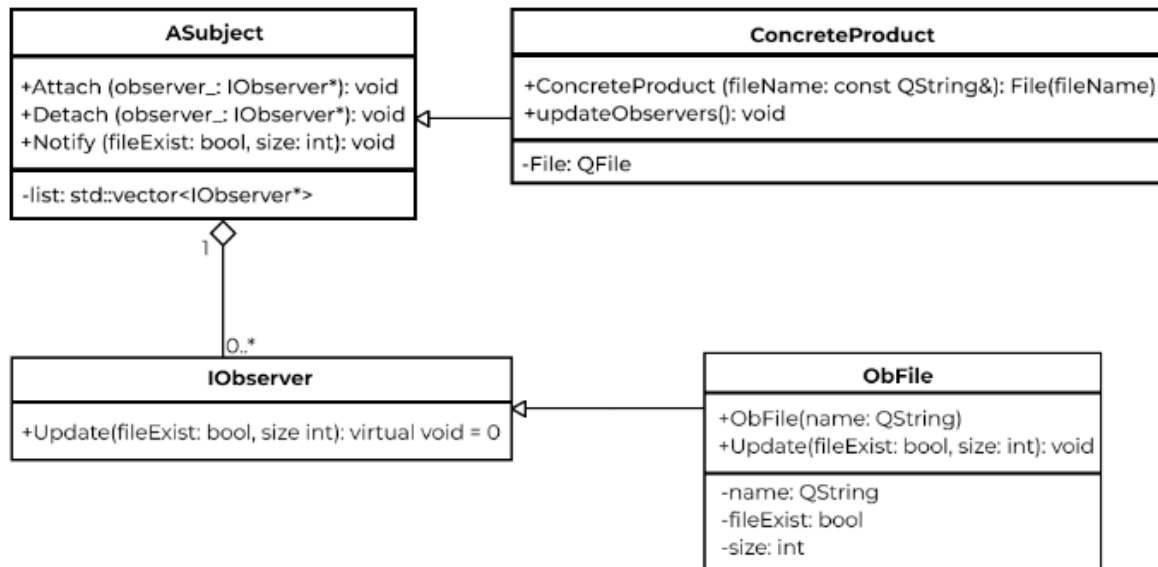


IObserver это абстрактный класс наблюдателя, который предоставляет наследникам метод, вызываемый при каждом изменении.

ObFile наследуется от IObserver и выполняет обновление информации, пришедшей от ASubject.



Итоговая UML диаграмма:



3. Коды программ

Файлы с кодом программы:

1. observer.h
2. observer.cpp
3. subject.h
4. subject.cpp
5. main.cpp
6. test.txt

Находятся на репозитории github.com/timuraknazarov/trpo_lab_2

4. Инструкция пользователя

Слежение за конкретным файлом реализовано следующим образом:

В файле main.cpp необходимо ввести имя файла, который заранее находится в директории сборки программы.

В данном примере, файл называется test.txt .

```
QString fileName = "test.txt"; // Создаём файл
```

В программе реализован бесконечный цикл, в котором обновляется состояние нашего файла, а также выводится информация о состоянии в консоль.

В приложении предусмотрено несколько состояний файла:

- файл не существует или отсутствует в директории

- файл существует и он не пуст, выводится информация о размере файла
- файл был изменен, выводится информация о новом размере

Сообщение о файле, если он существует и пустой, не предусмотрено.

После указания имени файла приложение начинает следить за файлом, соответственно появляется возможность получать информацию об изменении или удалении файла в консоли.

5. Тестирование

В файле main.cpp реализован пример наблюдателя для тестирования. В первую очередь, проверим добавление/удаление наблюдателей и общие взаимодействия с ними.

Создаем объект и добавляем двух наблюдателей.

```
QString fileName = "test.txt"; // Создаём файл
ConcreteProduct test(fileName); // Создаём объект ConcreteProduct

ObFile Ob1(fileName), Ob2(fileName); // Создаём наблюдатели

test.Attach(&Ob1); // Добавляем первый наблюдатель
test.Attach(&Ob2); // второй
```

Обновляем данные в наблюдателях

```
test.updateObservers(); // Обновляем информацию в наблюдателях
```

На выходе получаем информацию о файле от обоих наблюдателей.

```
File: test.txt does not exist
File: test.txt does not exist
```

Теперь попробуем отвязать второго наблюдателя и обновим данные.

```
std::cout<<std::endl; // Выведем пустую строку
test.Detach(&Ob2); // Отвязываем 2ого
test.updateObservers(); // Обновляем данные в наблюдателе
```

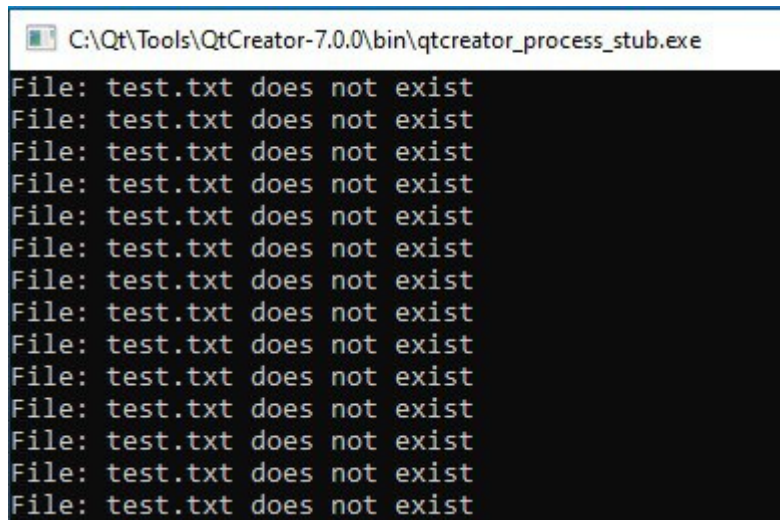
На выходе сначала получаем информацию от обоих наблюдателей, после отвязки только от одного.

```
File: test.txt does not exist
File: test.txt does not exist
File: test.txt does not exist
```

Мы протестировали добавление, удаление и обновление наблюдателей, программа работает корректно.

Наконец, проверим как программа взаимодействует с файлами.

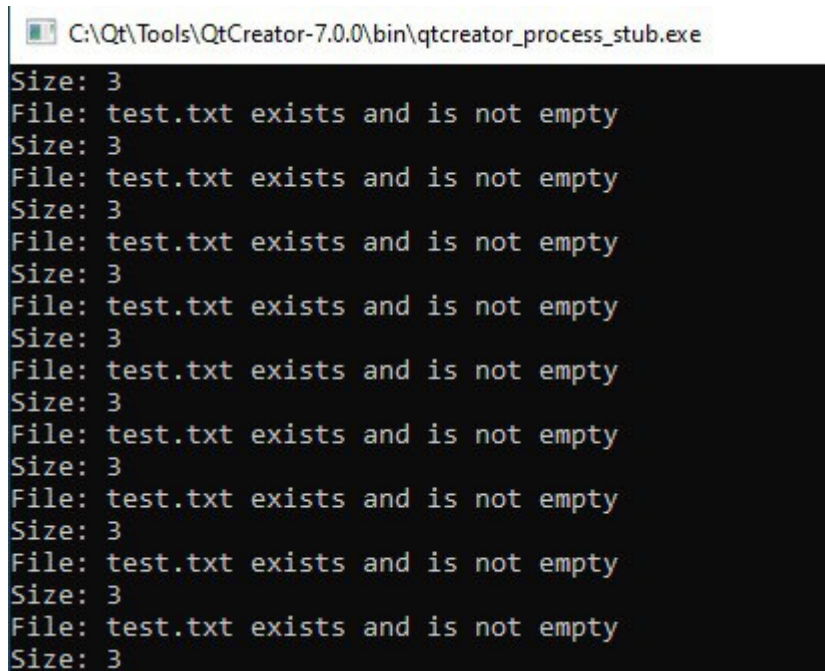
- **Тест 1: Файл в директории отсутствует**



```
C:\Qt\Tools\QtCreator-7.0.0\bin\qtcreator_process_stub.exe
File: test.txt does not exist
File: test.txt does not exist
File: test.txt does not exist
File: test.txt does not exist
File: test.txt does not exist
File: test.txt does not exist
File: test.txt does not exist
File: test.txt does not exist
File: test.txt does not exist
File: test.txt does not exist
File: test.txt does not exist
File: test.txt does not exist
File: test.txt does not exist
File: test.txt does not exist
File: test.txt does not exist
```

Так как файла нет, программа раз в секунду выдает сообщение о его отсутствии.

- **Тест 2: Файл есть в директории, и он не пустой**



```
C:\Qt\Tools\QtCreator-7.0.0\bin\qtcreator_process_stub.exe
Size: 3
File: test.txt exists and is not empty
Size: 3
File: test.txt exists and is not empty
Size: 3
File: test.txt exists and is not empty
Size: 3
File: test.txt exists and is not empty
Size: 3
File: test.txt exists and is not empty
Size: 3
File: test.txt exists and is not empty
Size: 3
File: test.txt exists and is not empty
Size: 3
File: test.txt exists and is not empty
```

Так как файл существует в директории и не пустой, программа раз в секунду выводит соответствующее сообщение и размер файла.

- **Тест 3: Файл есть, он не пустой, во время работы меняется его содержимое.**

C:\Qt\Tools\QtCreator-7.0.0\bin\qtcreator_process_stub.exe

```
Size: 3
File: test.txt exists and is not empty
Size: 3
File: test.txt exists and is not empty
Size: 3
File: test.txt exists and is not empty
Size: 3
File: test.txt exists and is not empty
Size: 3
File: test.txt exists and is not empty
Size: 3
File: test.txt exists and has been modified
New size: 15
File: test.txt exists and is not empty
Size: 15
File: test.txt exists and is not empty
Size: 15
File: test.txt exists and is not empty
Size: 15
```

Так как файл существует в директории, он не пустой программа выдает соответствующие сообщения. После того, как содержимое файла изменяется, программа также сообщает нам об этом, после чего выдает сообщения с новым размером файла.

- **Тест 4: Файл существует, но изначально пустой**

C:\Qt\Tools\QtCreator-7.0.0\bin\qtcreator_process_stub.exe

```
File: test.txt exists and has been modified
New size: 24
File: test.txt exists and is not empty
Size: 24
File: test.txt exists and is not empty
Size: 24
File: test.txt exists and is not empty
Size: 24
File: test.txt exists and is not empty
Size: 24
File: test.txt exists and is not empty
Size: 24
File: test.txt exists and is not empty
Size: 24
File: test.txt exists and is not empty
Size: 24
File: test.txt exists and is not empty
Size: 24
```

Так как файл существует в директории, но изначально пуст, программа не выдает сообщения, но после его изменения появляется соответствующее оповещение и сообщения с новым размером файла.

- **Тест 5: Файл существует и не пустой, но удаляется во время работы программы.**

 C:\Qt\Tools\QtCreator-7.0.0\bin\qtcreator_process_stub.exe

```
File: test.txt exists and is not empty
Size: 24
File: test.txt exists and is not empty
Size: 24
File: test.txt exists and is not empty
Size: 24
File: test.txt exists and is not empty
Size: 24
File: test.txt does not exist
File: test.txt does not exist
File: test.txt does not exist
File: test.txt does not exist
File: test.txt does not exist
File: test.txt does not exist
File: test.txt does not exist
```

Так как изначально файл существует, и не пуст, программа выдает соответствующие сообщения. После того, как файл удаляется из директории программы, появляются сообщения об отсутствии файла.

Программа работает корректно.