Шарко Полина Алексеевна. Группа 931920.

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

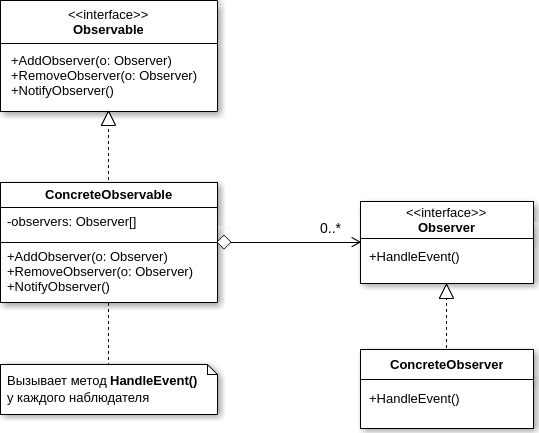
1. Файл существует, файл не пустой - на экран выводится факт существования файла и его размер.

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

2) Предлагаемое решение

Наблюдатель — это поведенческий паттерн проектирования, который создает механизм подписки, позволяющий одним объектам следить и реагировать на события, происходящие в других объектах. Паттерн используется когда в приложении есть несколько наблюдателей, отслеживающих состояние конкретного объекта (Subject). Наблюдатели уведомляются об изменении состояния наблюдаемого объекта.

UML-диаграмма паттерна Observer:

Здесь

Observable - интерфейс, определяет методы для добавления, удаления и оповещения наблюдателей. (класс, ответственный за взаимодействие с наблюдателями)

ConcreteObservable - класс, реализующий интерфейс Observable

Observer - интерфейс, с помощью которого наблюдатель получает оповещение

ConcreteObserver - класс, реализующий интерфейс Observer

Для реализации паттерна нам нужны два класса - класс Наблюдатель(у нас **IObserver**) и класс, ответственный за взаимодействие с наблюдателями(у нас **Subject**).

Рассмотрим реализованные классы.

**IObserver** – базовый класс наблюдателя.

Методы класса:

virtual void *Update*(bool fileExist, int size) = 0 - метод,определяющий состояние файла (существует/ не существует, изменен/не изменен, информация о размере файла) и оповещающий о нем.

**FileObserver –** наследник **IObserver** для работы с объектами типа QFile.

Поля класса:

QString name - поле, содержащее название файла

bool fileExist - поле, содержащее информацию о существовании файла

int size - поле, содержащее размер файла

Методы класса:

**FileObserver**(QString \_name = “”) - конструктор по умолчанию

void ***Update***(bool fileExist, int size) - метод,определяющий состояние файла (существует/ не существует, изменен/не изменен, информация о размере файла) и оповещающий о нем. Метод **Update** применяет при обработке информации, полученной от объектов класса **Subject**.

**Subject -** класс, ответственный за взаимодействие с наблюдателями. Регулирует обновление наблюдателей, предоставляя им методы для контроля своего нахождения в списке(позволяет добавлять/удалять себя из списка).

Поля класса:

vector<IObserver\*> list - список адресов наблюдателей

Методы класса:

void **Attach**(IObserver \*product) - метод для присоединения наблюдателя

void **Detach**(IObserver \*product) - метод для отключения наблюдателя

void **Notify**(bool fileExist, int size) - метод, уведомляющий об обновлении данных

**FileSubject -** наследник **Subject** для работы с объектами типа QFile**,** отвечает за обновление данных в наблюдателях.

Поля класса:

QFile File; - поле, содержащее файл

Методы класса:

**FileSubject**(const QString& fileName): File(fileName) - конструктор по умолчанию

void **updateObservers**() - метод, уведомляющий об обновлении данных при работе с файлами

3) Коды программ

Файлы можно найти по ссылке

**Observer.h** - содержит классы-наблюдатели: **IObserver** и **FileObserver.**

**Observer.cpp** - файл с определением метода класса **FileObserver.**

**subj.h** - содержит классы, ответственные за взаимодействие с наблюдателями: **Subject** и **FileSubject.**

**subj.cpp** - файл с определением метода класса **Subject.**

**main.cpp** - файл, используемый для тестирования

Файлы для тестирования: **file.txt** - наблюдаемый файл

4) Инструкция пользователя

Программа отслеживает состояние файла - существование и изменение.

Примеры использования функций:

**FileSubject**(const QString& fileName): File(fileName) - FileSubject test(fileName);

void **Attach**(IObserver \*product) - test.Attach(&fo2);

void **Detach**(IObserver \*product) - test.**Detach**(&fo2);

**FileObserver**(QString \_name = “”) - FileObserver fo2(fileName);

void **updateObservers**() - test.updateObservers();

5) Тестирование

Примеры для тестирования реализованы в файле main.cpp.

Протестируем метод void **updateObservers**().

**Сценарий:**

Из условия, ситуации может быть 3:

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

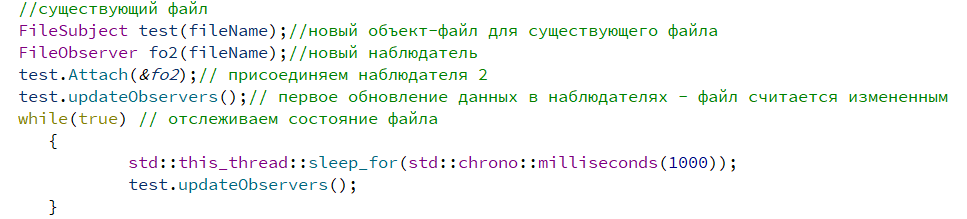
*2. Файл существует, файл был изменен - на экран выводится факт существования файла, сообщение о том, что файл был изменен и его размер.*

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

Рассмотрим поведение программы в этих случаях.

*1. Файл существует, файл не пустой - на экран выводится информация о существовании файла и его размер.*

Тест:

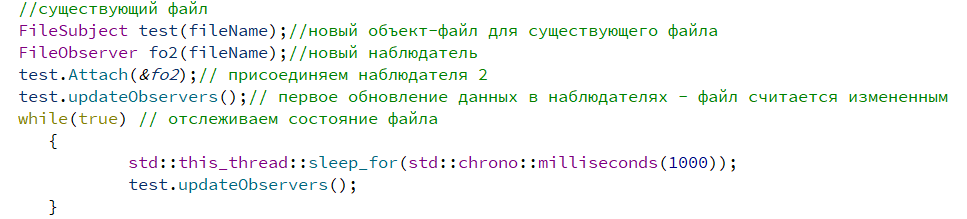


**Вывод:**



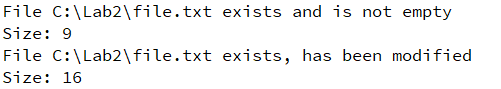
*2. Файл существует, файл был изменен - на экран выводится факт существования файла, сообщение о том, что файл был изменен и его размер.*

Тест:



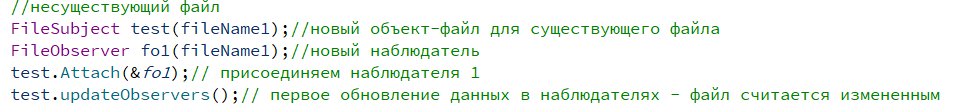
Файл изменяем вручную.

**Вывод:**

****

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

Тест:



**Вывод:**

