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.

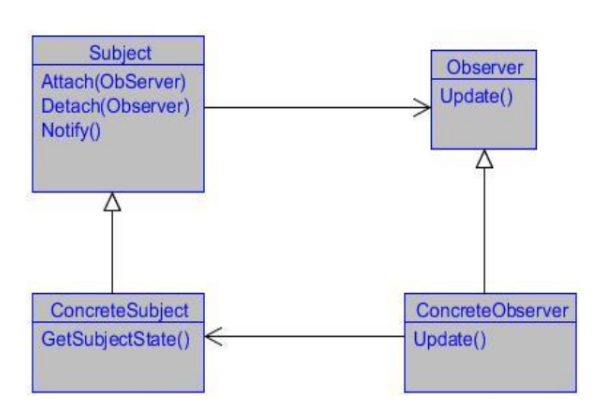
2. Файл существует, файл был изменен - на экран выводится факт существования файла, сообщение о том, что файл был изменен и его размер.

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

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

ConcreteSubject - класс, реализующий интерфейс Subject

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 **ASubject**).

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.

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

Поля класса:

static QString name - поле, содержащее название файла

static bool fileExist - поле, содержащее информацию о существовании файла

static int size - поле, содержащее размер файла

Методы класса:

**ConsoleFile**()= default;//Шаг 4: конструктор, приватный

~**ConsoleFile**()= default;// деструктор, приватный

**ConsoleFile**(const ConsoleFile&)=delete;//закрываем конструктор копирования

ConsoleFile& operator=(const ConsoleFile&)=delete;//и оператор присваивания тоже закрываем

static ConsoleFile& **instance**()

void ***Update***(bool fileExist, int size); // метод, обновляющий информацию о состоянии файла - 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 **ASubject**.

Результат работы метода Update – вывод наблюдения за файлом в консоль. Консоль должна быть единственна, поэтому реализуем ConsoleFile на основе паттерна Singleton.

**ASubject -** 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:

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

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

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

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

3) Коды программ

Файлы можно найти по ссылке

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

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

**subj.h** - содержит классы, ответственные за взаимодействие с наблюдателями: **ASubject и ConcretFile.**

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

**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:

void **Attach**(IObserver \*sbj) - test.*Attach*(*&*ConsoleFile*::instance()*); void **Detach**(IObserver \*product) - test.*Notify*(File.exists(),File.*size*());

void ***Notify***(bool fileExist, int size) - test.*Notify*(File.exists(),File.*size*());

5) Тестирование

Примеры для тестирования реализованы в файле main.cpp.

Протестируем метод void **updateObservers**().

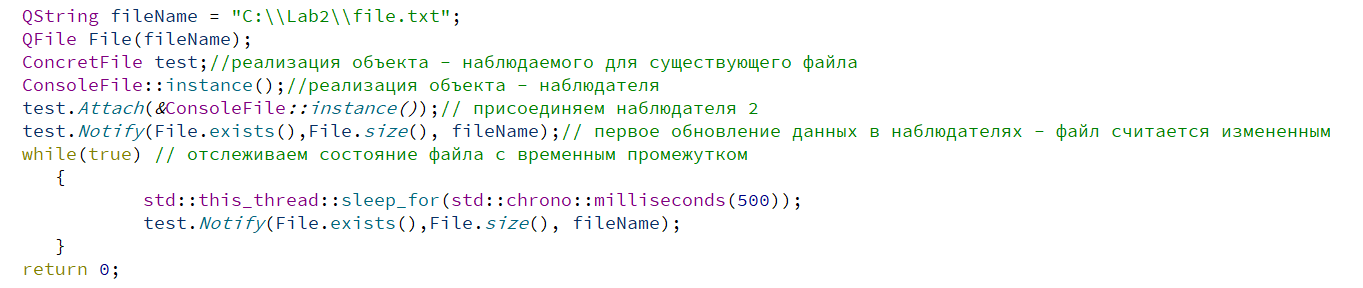
**Сценарий:**

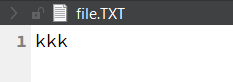
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| состояние файла  Ситуация | существует | пуст | изменен | вывод |
| 1 | 1 | 0 | 0 | *на экран выводится факт существования файла и его размер* |
| 2 | 0 | 1/0 | 1/0 | *на экран выводится информация о том, что файл не существует* |
| 3 | 1 | 1 | 0 | Ничего не выведено |
| 4 | 1 | 0/1 | 1 | *на экран выводится факт существования файла, сообщение о том, что файл был изменен и его размер* |

Ситуации:

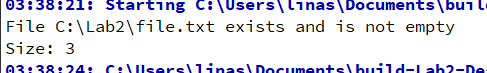
1.

Тест:

****

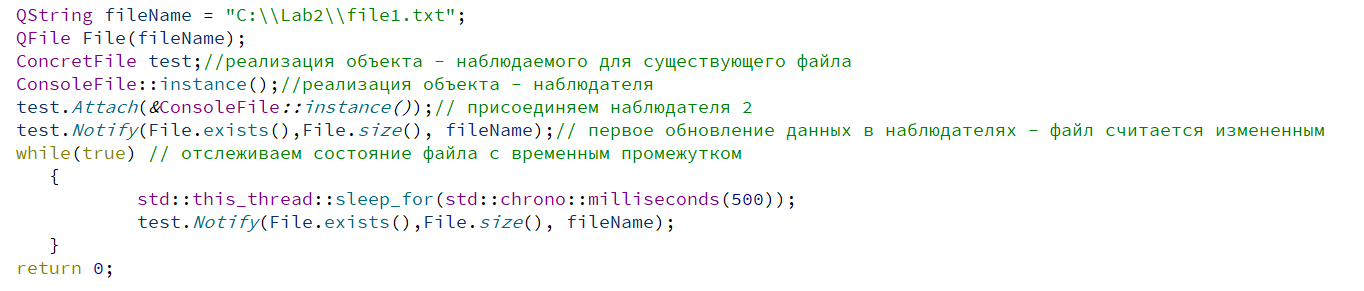
****

**Вывод:**



*2.*

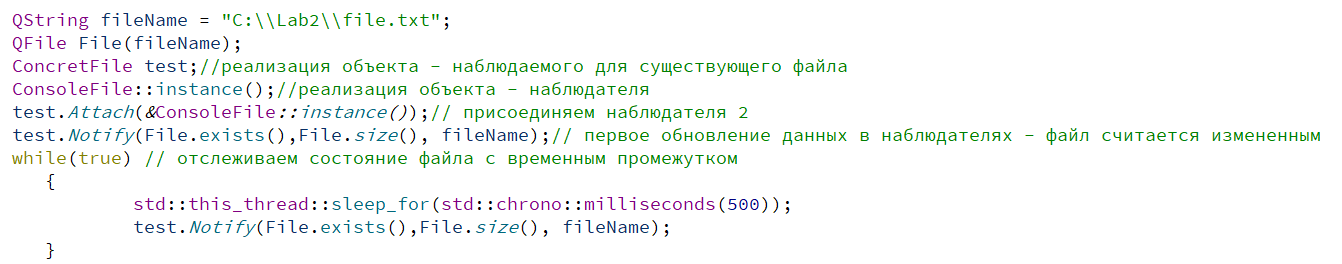
Тест:

**Вывод:**

****

*3.*

Тест:



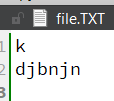
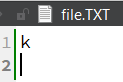
****

**Вывод:**



4.

Тест:

****

**Вывод:**

