**Weather Station with Observer**

The Weather Station with Observer is a library that depends on a library. To explain this I say that Weather Station uses the Observer class. (This means that both the Observer and the Weather Station libs need to be used.)

**Methods for Station:**

**Class Station**

Temperature getTemperature() const;

Returns a Class Temperature with a value.

Humidity getHumidity() const;

Returns a Class Humidity with a value.

Pressure getPressure() const;

Return a Class Pressure with a value.

static Station & getWeatherStation();

Returns an instance of Station.

Temperature getMeanTemperature(std::chrono::system\_clock::time\_point const t0, std::chrono::system\_clock::time\_point const t1) const;

Returns a Class Temperature after multiple Temperature changes.

Humidity getMeanHumidity(std::chrono::system\_clock::time\_point const t0, std::chrono::system\_clock::time\_point const t1) const;

Returns a Class Humidity after multiple Humidity changes.

Pressure getMeanPressure(std::chrono::system\_clock::time\_point const t0,

std::chrono::system\_clock::time\_point const t1) const;

Returns a Class Humidity after multiple Pressure changes.

void measure();

Causes an update to Temperature Humidity and Pressure because I do not have an actual weather station but a representation of one instead.

**Example Usage with a Viewer:**

#include "stdafx.h"

#include <iostream>

#include <random>

#include <chrono>

#include <thread>

#include "station.h"

#include "current.h"

int main()

{

WeatherStation::Station weatherStation;

const WeatherViewer::Current currentWeather(weatherStation);

std::random\_device rd;

std::mt19937 mt{ rd() };

std::uniform\_int\_distribution<int> const dist{ 0, 10000 };

for (int i{ 0 }; i != 10; ++i)

{

auto const sporadic{ dist(mt) };

std::this\_thread::sleep\_for(std::chrono::milliseconds(sporadic));

weatherStation.measure();

std::cout<< currentWeather;

}

return 0;

}

**Note:**

Current is a viewer class and is not part of the Station.

Weather only changes upon a measure call.

Current inherits Observer and Station inherits subject.

**Example Usage inside a Viewer:**

class Current : public Observer

{

friend std::ostream& operator<<(std::ostream& os, Current const& current);

private:

WeatherStation::Station const& station\_;

int\* temperature;

int\* humidity;

double\* pressure;

public:

explicit Current(WeatherStation::Station const& station);

WeatherStation::Station const& getStation() const;

void Update() override;

};

void Current::Update()

{

\*temperature = station\_.getTemperature().get();

\*humidity = station\_.getHumidity().get();

\*pressure = station\_.getPressure().get();

}

**SAMPLE OUTPUT:**

TEMP | Humidity | Pressure

37 (C)| 103% | 58 in. Hg

TEMP | Humidity | Pressure

36 (C)| 100% | 36 in. Hg