

WZORCE PROJEKTOWE

# SPRAWOZDANIE

ZADANIE OBSERVER STACJA POGODOWA

Patryk Figas  
Informatyka, programowanie  
Grupa 34\_Inf\_P\_NW\_6

## 1. Cel

Celem dokumentu jest przedstawienie rozwiązania zadania polegającego na zaprojektowaniu i zaimplementowaniu aplikacji stacji pogodowej z wykorzystaniem wzorca projektowego **Observer**. Stworzono system, który automatycznie powiadamia komponenty wyświetlające dane meteorologiczne o zmianach warunków pogodowych.

W ramach ćwiczenia zaprojektowano klasę za pomocą „pseudokodu”, diagramu UML i implementacji klasy do programu oraz użycie jej w programie Main.

W zadaniu został zastosowany **wzorzec projektowy Observer**, który umożliwia niezależne powiązanie wielu obserwatorów z jednym źródłem danych (podmiotem).

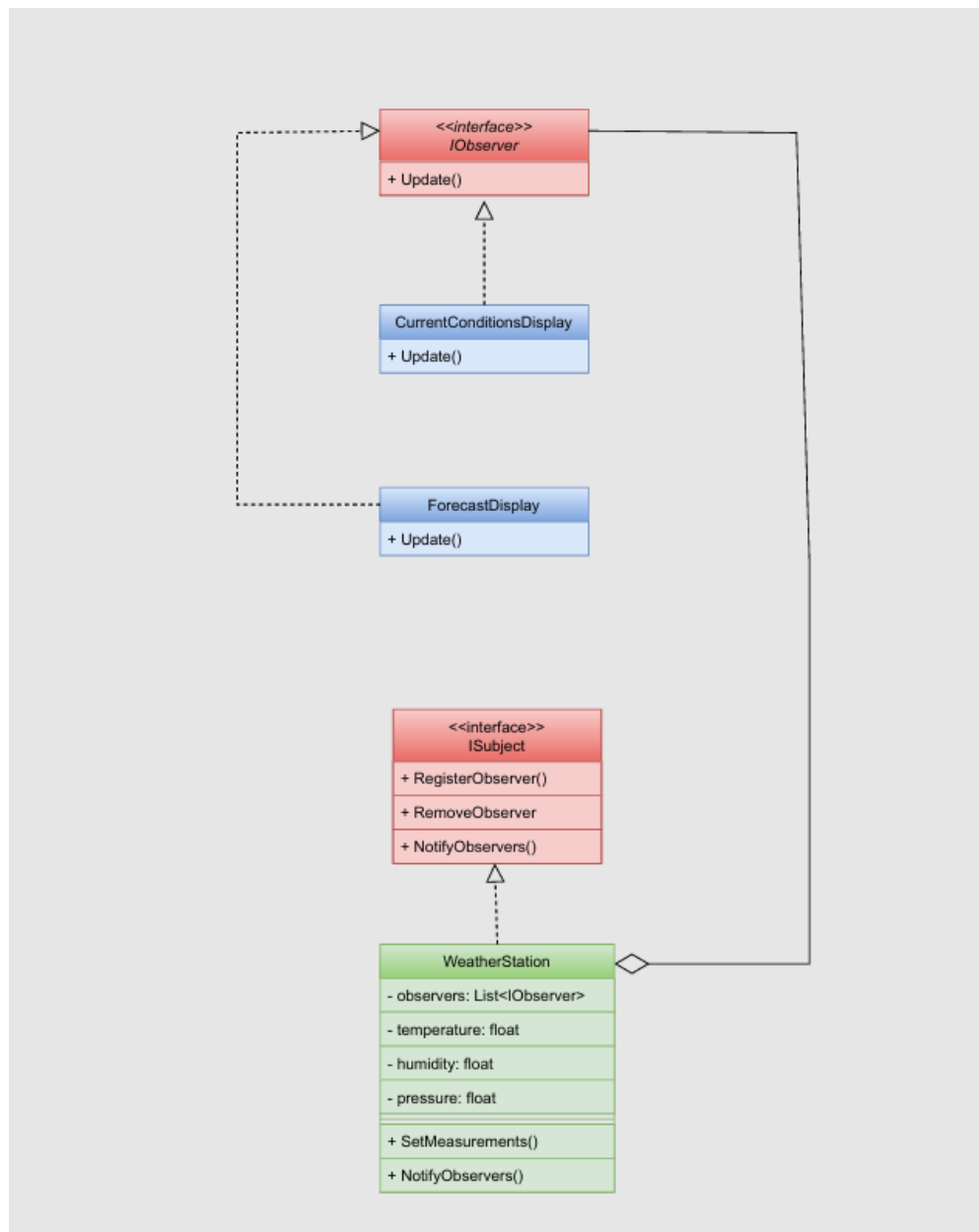
## 2. Opis rozwiązania

Rozwiązanie opiera się na utworzeniu dwóch interfejsów: **ISubject** (podmiot) oraz **IObserver** (obserwator). Podmiotem jest **klasa WeatherStation**, która przechowuje dane pogodowe i informuje o ich zmianie wszystkich zarejestrowanych obserwatorów.

Obserwatorami są **klasy CurrentConditionsDisplay** i **ForecastDisplay**, które implementują **metodę Update()** i reagują na zmiany danych.

Po każdej zmianie danych pogodowych, WeatherStation wywołuje Update() dla wszystkich obserwatorów, które następnie wyświetlają aktualne dane lub prognozę.

- Diagram klas programu



- Pseudokod klas i interfejsów programu

```
Pseudokod.txt
Plik  Edytuj  Wyświetl

// Interfejs obserwatora
interface IObserver {
    void Update(float temperature, float humidity, float pressure);
}

// Interfejs podmiotu (subject)
interface ISubject {
    void RegisterObserver(IObserver o);
    void RemoveObserver(IObserver o);
    void NotifyObservers();
}

// Klasa WeatherStation - podmiot
class WeatherStation : ISubject {
    private List<IObserver> observers = new List<IObserver>();
    private float temperature;
    private float humidity;
    private float pressure;

    public void RegisterObserver(IObserver o) {
        observers.Add(o);
    }

    public void RemoveObserver(IObserver o) {
        observers.Remove(o);
    }

    public void NotifyObservers() {
        foreach (var observer in observers) {
            observer.Update(temperature, humidity, pressure);
        }
    }

    public void SetMeasurements(float temp, float hum, float pres) {
        this.temperature = temp;
        this.humidity = hum;
        this.pressure = pres;
        NotifyObservers();
    }
}

// Obserwator: wyświetla aktualne warunki
class CurrentConditionsDisplay : IObserver {
    public void Update(float temperature, float humidity, float pressure) {
        Console.WriteLine($"Aktualne warunki: {temperature}°C, {humidity}% wilgotności");
    }
}

// Obserwator: wyświetla prognozę na podstawie ciśnienia
class ForecastDisplay : IObserver {
    public void Update(float temperature, float humidity, float pressure) {
        string forecast = pressure > 1013 ? "Słonecznie" : "Możliwy deszcz";
        Console.WriteLine($"Prognoza pogody: {forecast}");
    }
}

// Przykładowe użycie
class Program {
    static void Main() {
        WeatherStation station = new WeatherStation();

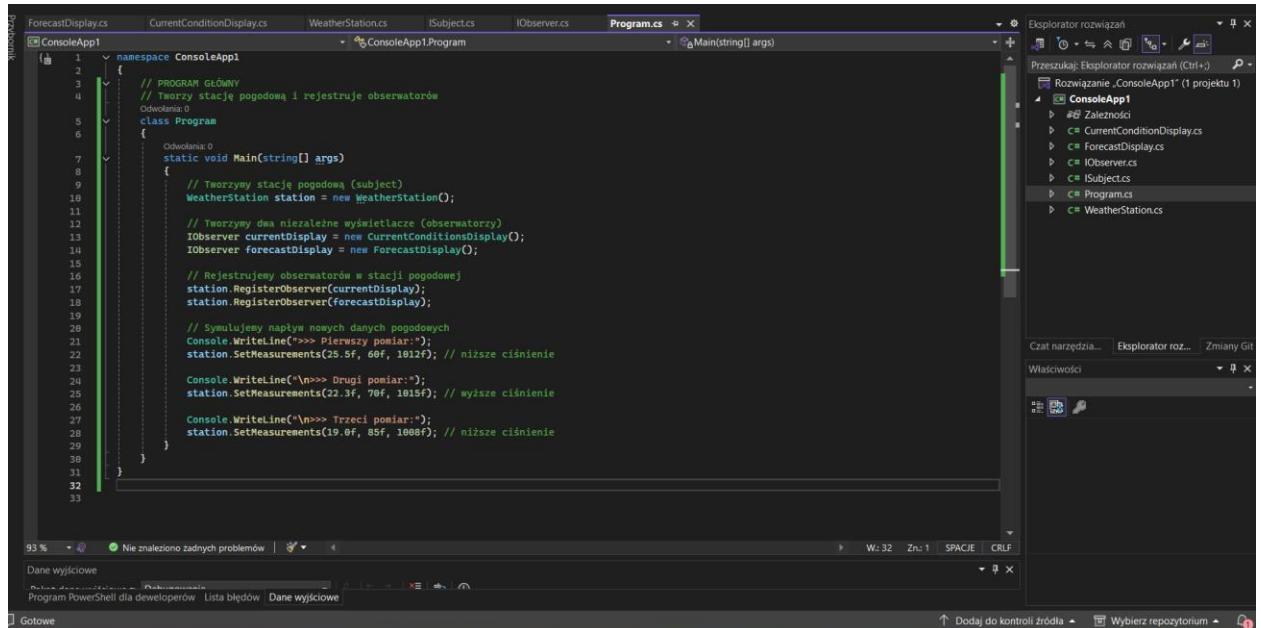
        CurrentConditionsDisplay currentDisplay = new CurrentConditionsDisplay();
        ForecastDisplay forecastDisplay = new ForecastDisplay();

        station.RegisterObserver(currentDisplay);
        station.RegisterObserver(forecastDisplay);

        station.SetMeasurements(25.5f, 60f, 1012f);
        station.SetMeasurements(22.3f, 70f, 1015f);
    }
}
```

### 3. Implementacja

- Kod klasy **Program.cs**



The screenshot shows the Visual Studio IDE with a C# console application named 'ConsoleApp1'. The 'Program.cs' file is open, showing the following code:

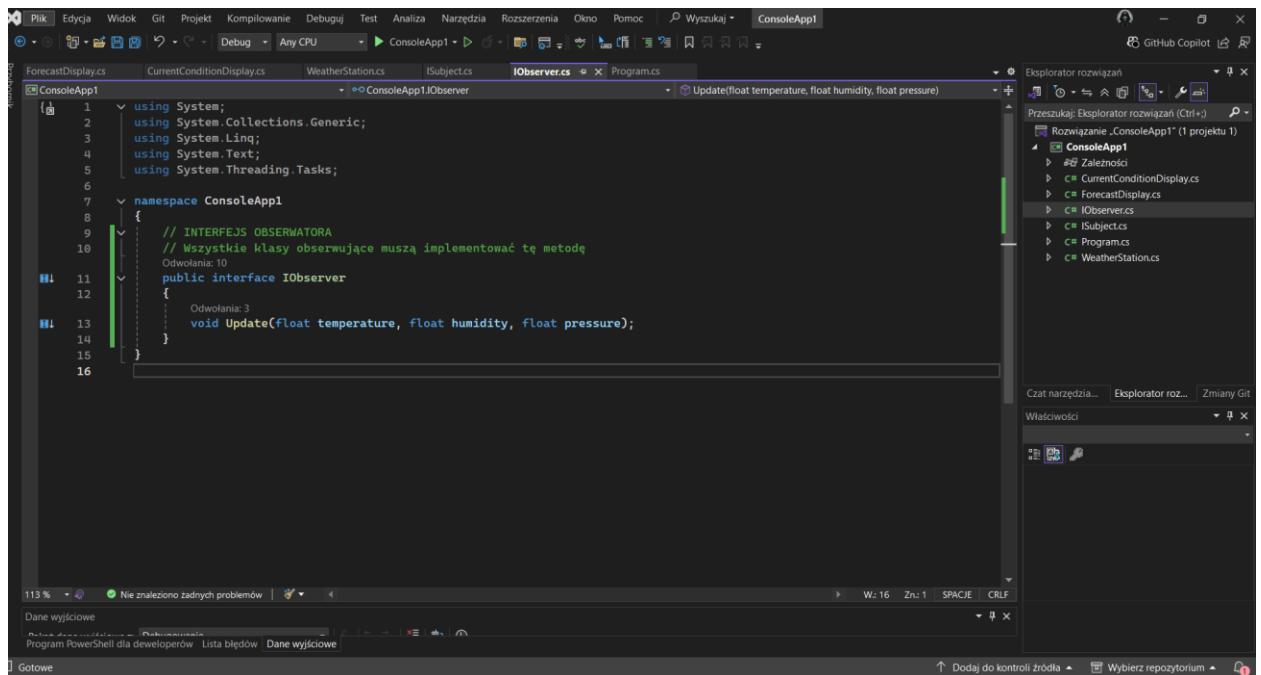
```
1 namespace ConsoleApp1
2 {
3     // PROGRAM GŁÓWNY
4     // Tworzy stację pogodową i rejestruje obserwatorów
5     class Program
6     {
7         Odwołania: 0
8         static void Main(string[] args)
9         {
10             // Tworzymy stację pogodową (subject)
11             WeatherStation station = new WeatherStation();
12
13             // Tworzymy dwa niezależne wyświetlacze (obserwatory)
14             IObservable currentDisplay = new CurrentConditionsDisplay();
15             IObservable forecastDisplay = new ForecastDisplay();
16
17             // Rejestrujemy obserwatorów w stacji pogodowej
18             station.RegisterObserver(currentDisplay);
19             station.RegisterObserver(forecastDisplay);
20
21             // Symulujemy napływ nowych danych pogodowych
22             Console.WriteLine(">>> Pierwszy pomiar:");
23             station.SetMeasurements(25.5f, 60f, 1012f); // niższe ciśnienie
24
25             Console.WriteLine("\n>>> Drugi pomiar:");
26             station.SetMeasurements(22.3f, 70f, 1015f); // wyższe ciśnienie
27
28             Console.WriteLine("\n>>> Trzeci pomiar:");
29             station.SetMeasurements(19.0f, 85f, 1008f); // niższe ciśnienie
30
31         }
32     }
33 }
```

The right sidebar shows the 'Eksplorator rozwiązań' (Solution Explorer) with the project structure:

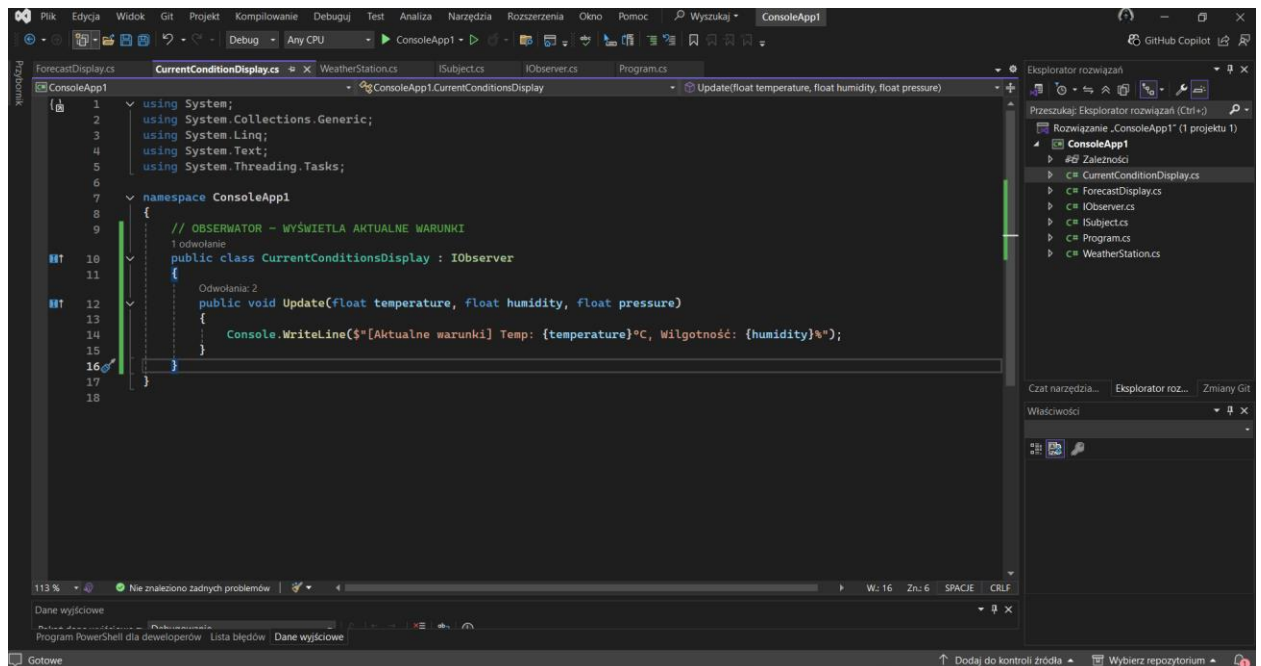
- ConsoleApp1 (1 projektu 1)
- ## Zależności
- CurrentConditionDisplay.cs
- ForecastDisplay.cs
- IObservable.cs
- ISubject.cs
- Program.cs
- WeatherStation.cs

The status bar at the bottom indicates '93 %' completion, 'Nie znaleziono żadnych problemów' (No problems found), and 'W: 32 Zn: 1 SPACJE CRLF'.

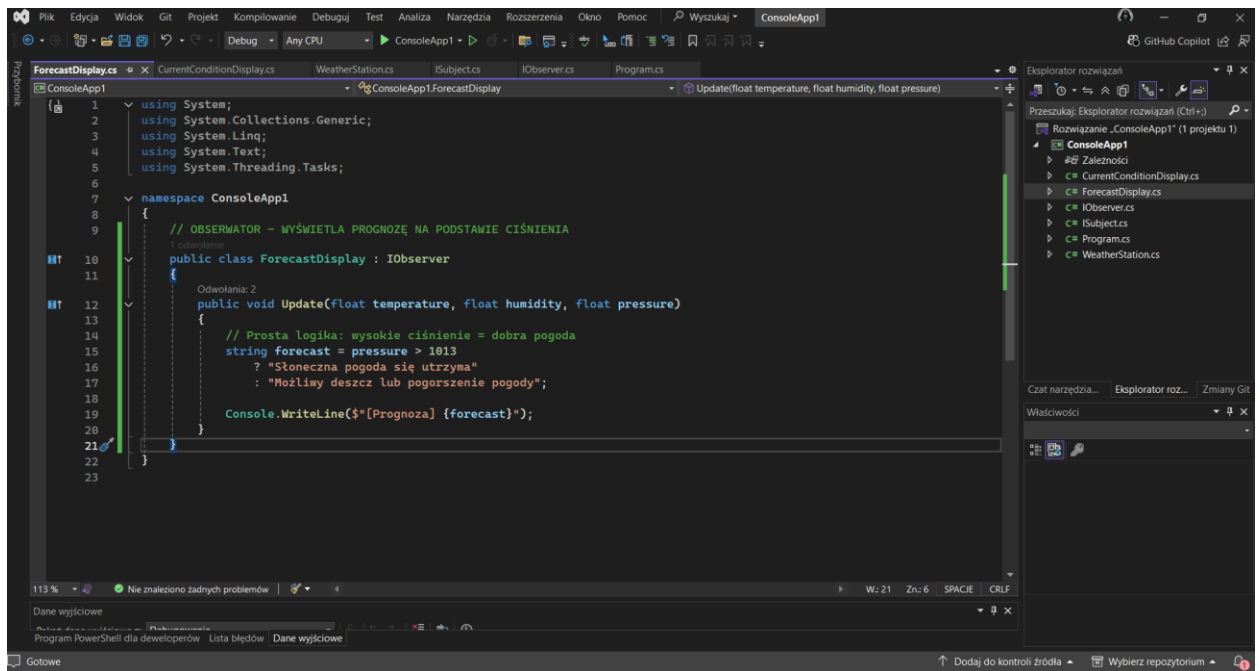
- Kod interfejsu **IObserver.cs**



- Kod klasy **CurrentConditionDisplay.cs**

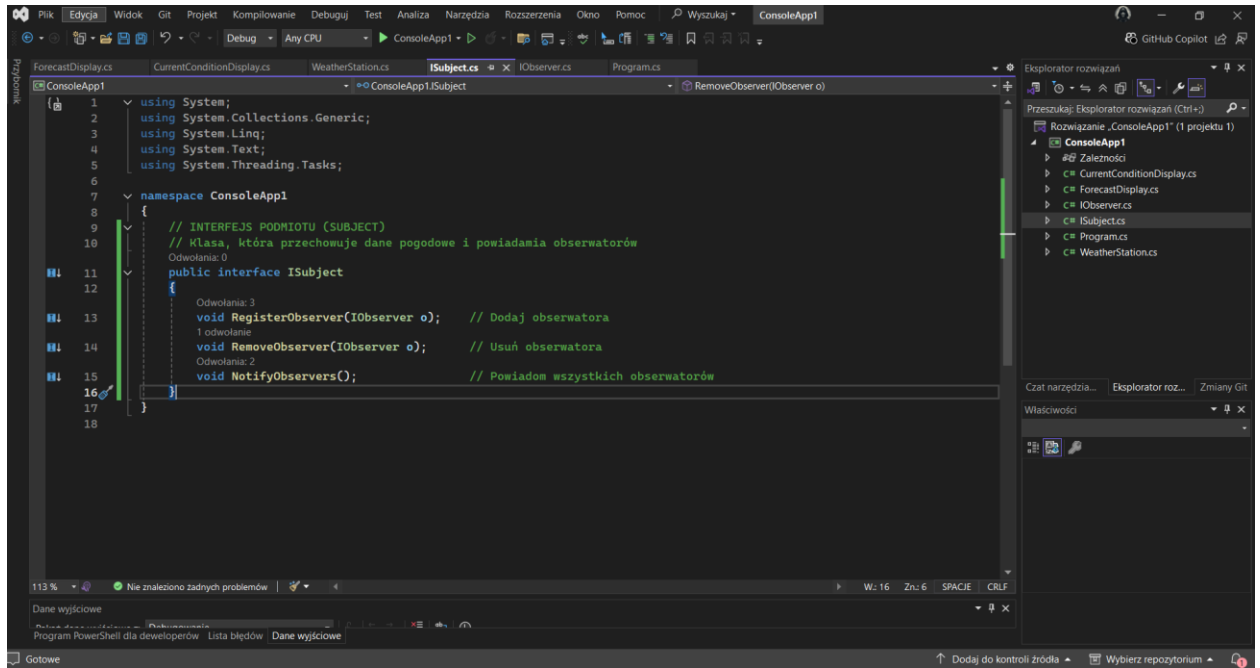


- Kod klasy **ForecastDisplay.cs**





- Kod interfejsu **ISubject.cs**



- Kod klasy **WeatherStation.cs**



```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace ConsoleApp1
8 {
9     // KLASA STACJI POGODOWEJ
10    // Przechowuje dane pogodowe i informuje obserwatorów o zmianach
11    // Odwołania: 3
12    public class WeatherStation : ISubject
13    {
14        private List<IObserver> observers; // Lista obserwatorów
15        private float temperature;
16        private float humidity;
17        private float pressure;
18
19        // 1 odwołanie
20        public WeatherStation()
21        {
22            observers = new List<IObserver>(); // Inicjalizacja listy obserwatorów
23        }
24
25        // Dodaj obserwatora do listy
26        // Odwołania: 3
27        public void RegisterObserver(IObserver o)
28        {
29            observers.Add(o);
30        }
31
32        // Usuń obserwatora z listy
33        // 1 odwołanie
34        public void RemoveObserver(IObserver o)
35        {
36            observers.Remove(o);
37        }
38
39        // Powiadom wszystkich obserwatorów o nowych danych
40        // Odwołania: 2
41        public void NotifyObservers()
42        {
43            foreach (var observer in observers)
44            {
45                // Przekazujemy aktualne dane każdemu obserwatorowi
46                observer.Update(temperature, humidity, pressure);
47            }
48        }
49
50        // Ustaw nowe dane pogodowe i powiadom obserwatorów
51        // Odwołania: 3
52        public void SetMeasurements(float temp, float hum, float pres)
53        {
54            temperature = temp;
55            humidity = hum;
56            pressure = pres;
57            NotifyObservers(); // Automatycznie wywołujemy informowanie obserwatorów
58        }
59    }
60 }
```

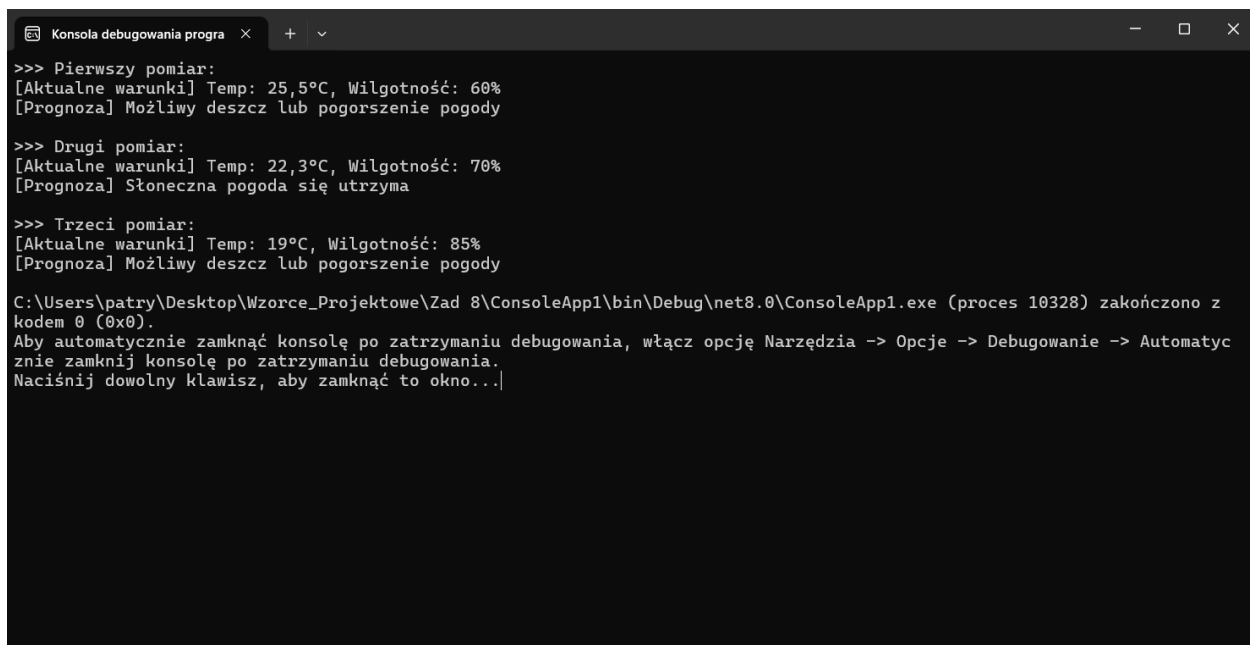
58 % Nie znaleziono żadnych problemów

## 4. Podsumowanie

Do realizacji zadania zastosowano wzorzec **Observer**, ponieważ:

- umożliwia niezależną reakcję wielu komponentów na zmianę stanu jednego obiektu,
- pozwala dynamicznie dodawać i usuwać obserwatorów bez modyfikacji klasy WeatherStation,
- ułatwia rozdzielenie odpowiedzialności (zasada SOLID – Single Responsibility).

Implementacja wzorca zakończyła się sukcesem – aplikacja działa zgodnie z założeniami i umożliwia łatwą rozbudowę (np. dodanie nowych typów wyświetlaczy).



```
>>> Pierwszy pomiar:
[Aktualne warunki] Temp: 25,5°C, Wilgotność: 60%
[Prognoza] Możliwy deszcz lub pogorszenie pogody

>>> Drugi pomiar:
[Aktualne warunki] Temp: 22,3°C, Wilgotność: 70%
[Prognoza] Słoneczna pogoda się utrzyma

>>> Trzeci pomiar:
[Aktualne warunki] Temp: 19°C, Wilgotność: 85%
[Prognoza] Możliwy deszcz lub pogorszenie pogody

C:\Users\patry\Desktop\Wzorce_Projektowe\Zad 8\ConsoleApp1\bin\Debug\net8.0\ConsoleApp1.exe (proces 10328) zakończono z
kodem 0 (0x0).
Aby automatycznie zamknąć konsolę po zatrzymaniu debugowania, włącz opcję Narzędzia -> Opcje -> Debugowanie -> Automatyc
znie zamknij konsolę po zatrzymaniu debugowania.
Naciśnij dowolny klawisz, aby zamknąć to okno...
```

Alternatywne wzorce (np. Mediator lub Publisher-Subscriber) mogłyby działać, ale **Observer jest najprostszy i najbardziej adekwatny** w przypadku bezpośrednich relacji między obiektami.

## Lista załączników

Repozytorium GITHUB z projektem:

<https://github.com/PatrykFigas/Wzorce-projektowe.git>