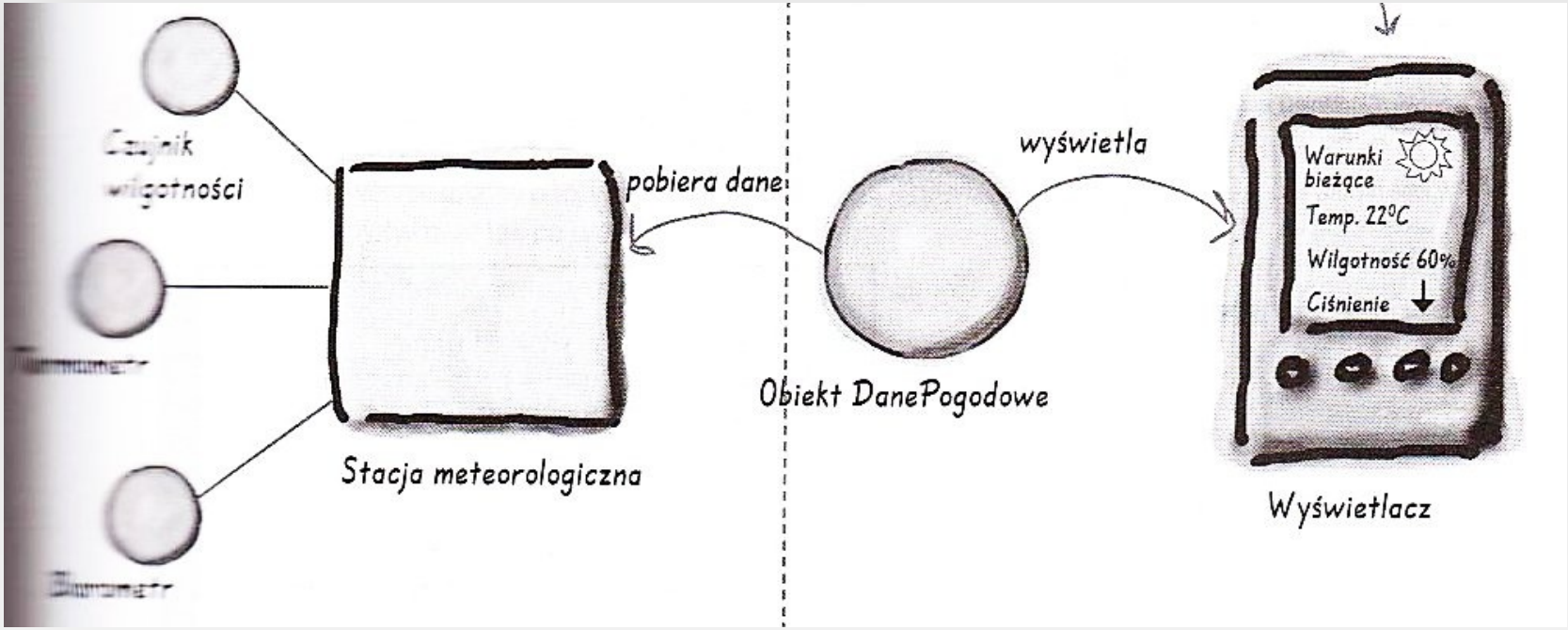
# Nowoczesne języki programowania obiektowego

## Wzorce projektowe: OBSERWATOR

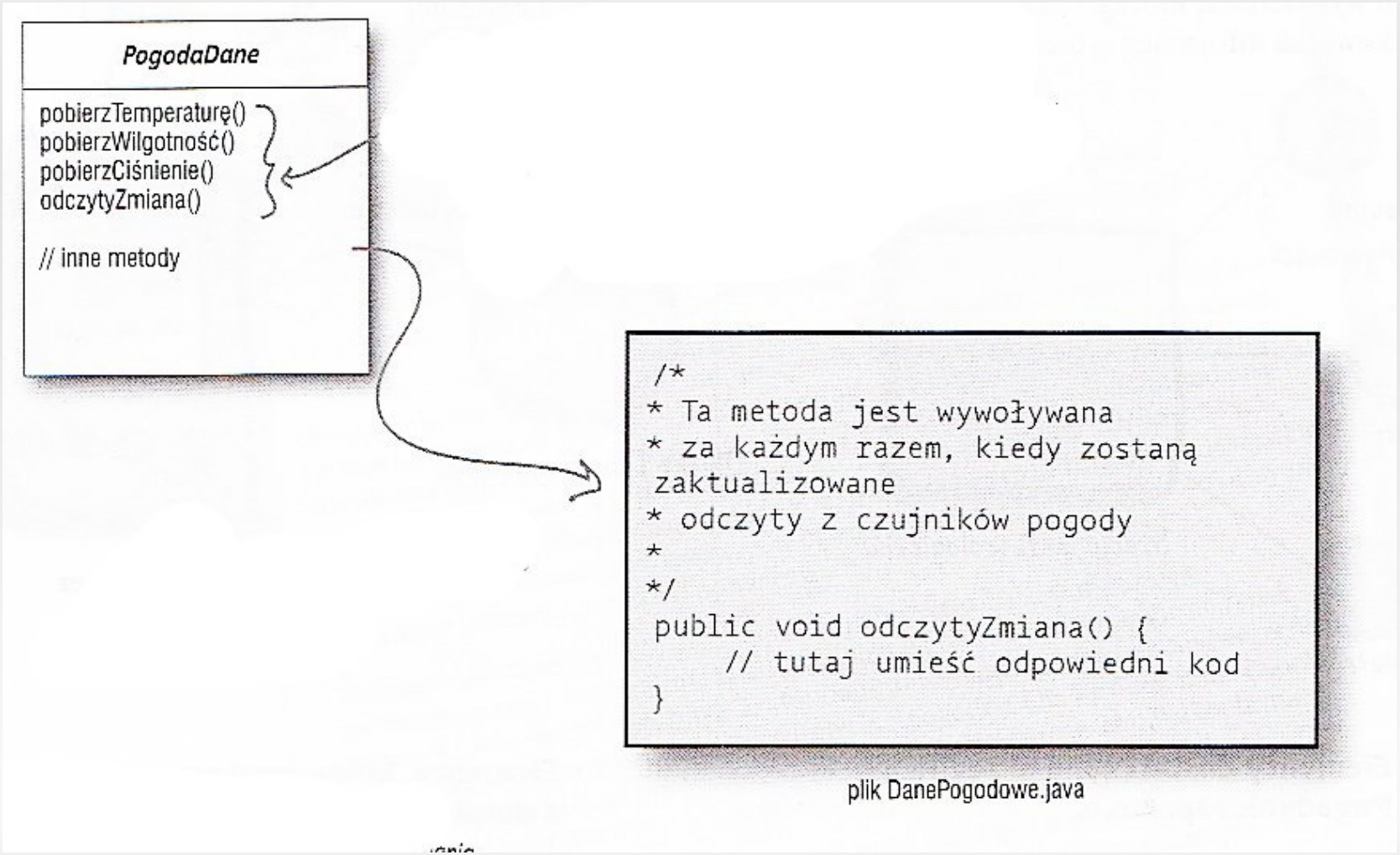
dr Aleksander Lamża

Wydział Informatyki i Nauki o Materiałach
Instytut Informatyki, Uniwersytet Śląski

# Problem


Czujnik wilgotności
pobiera dane
wyświetla
Warunki bieżące
Temp. 22°C
Wilgotność 60%
Ciśnienie
Obiekt DanePogodowe
Stacja meteorologiczna
Wyświetlacz

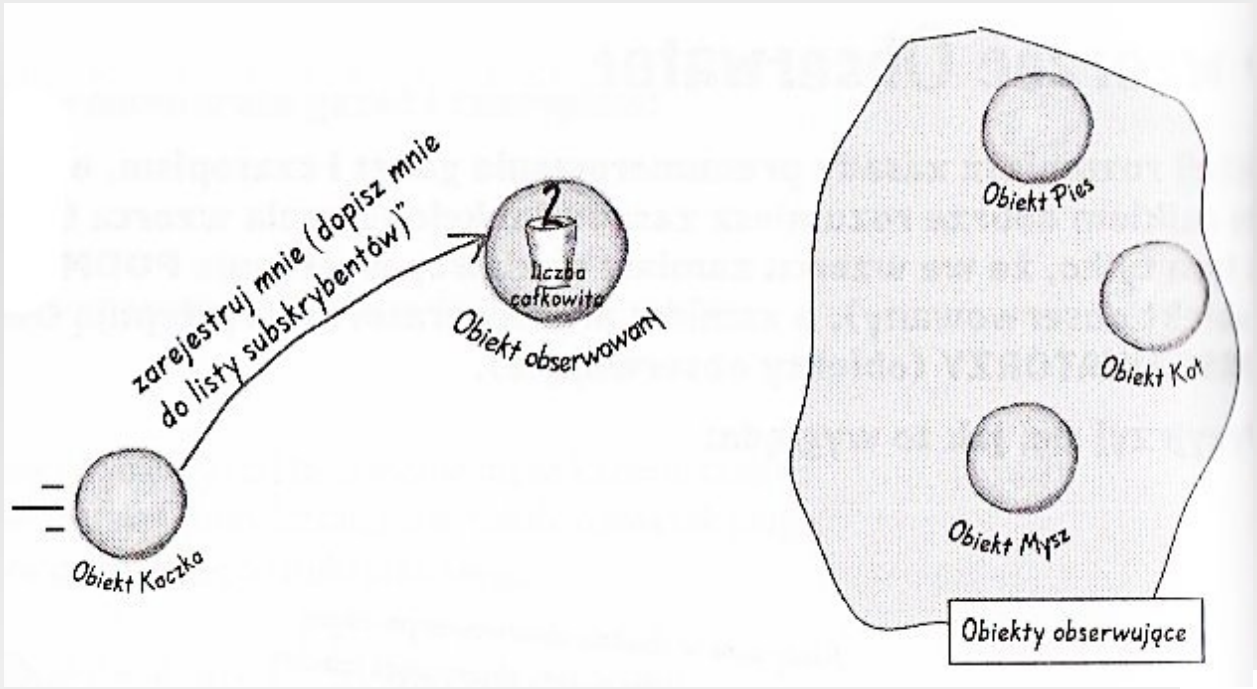
# Złe rozwiązanie problemu

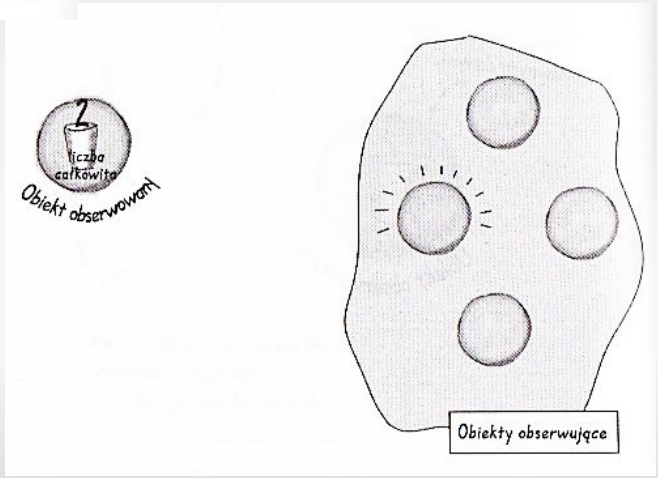


# Złe rozwiązanie problemu cd.

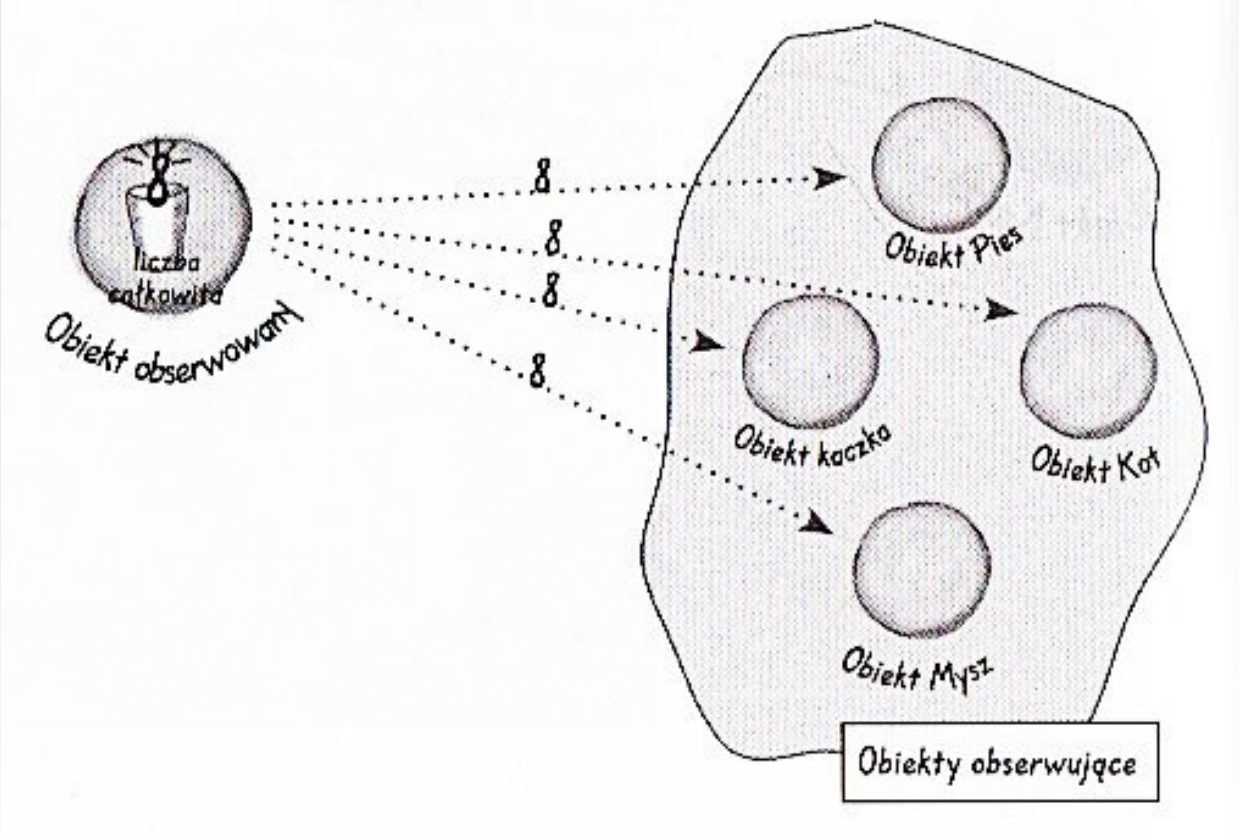
```
public class DanePogodowe {
    // deklaracje zmiennych obiektowych
    public void odczytyZmiana() {

        float temp = pobierzTemperaturę();
        float wilgotność = pobierzWilgotność();
        float pressure = pobierzCiśnienie();

        warunkiBieżąceWyświetl.aktualizacja(temp, wilgotność, ciśnienie);
        statystykaWyświetl.aktualizacja(temp, wilgotność, ciśnienie);
        prognozaWyświetl.aktualizacja(temp, wilgotność, ciśnienie);
    }
    // w tym miejscu można wstawić inne metody obiektu PogodaDane
}
```

# Obserwator: zasada działania

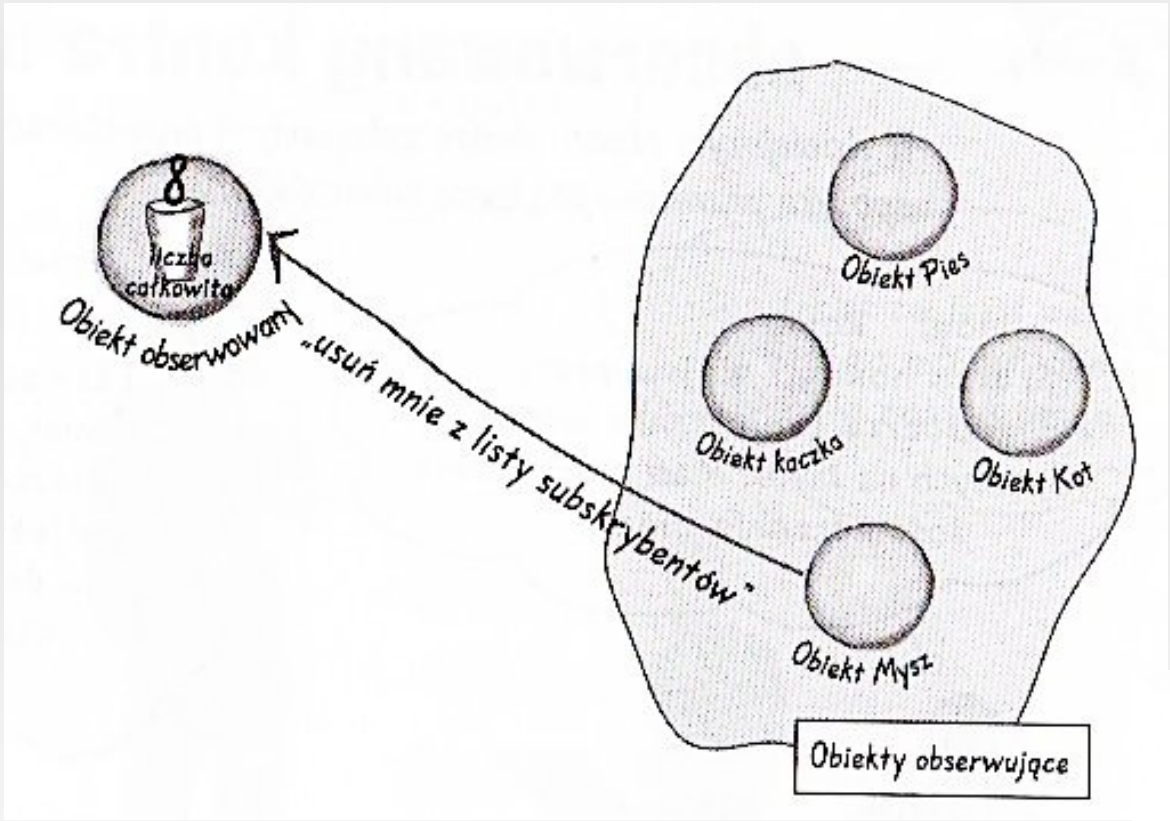


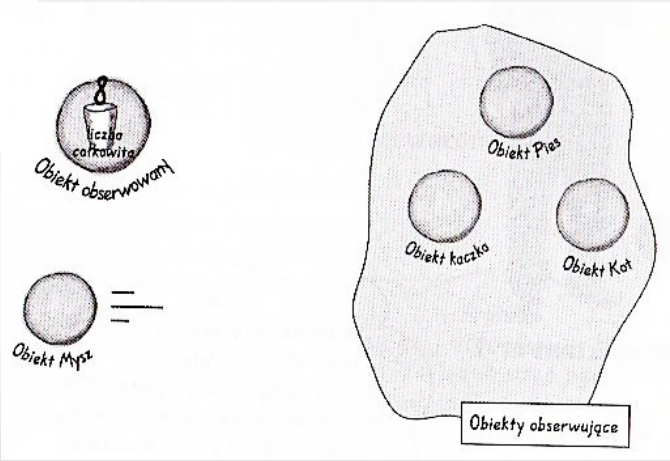



# Obserwator: zasada działania

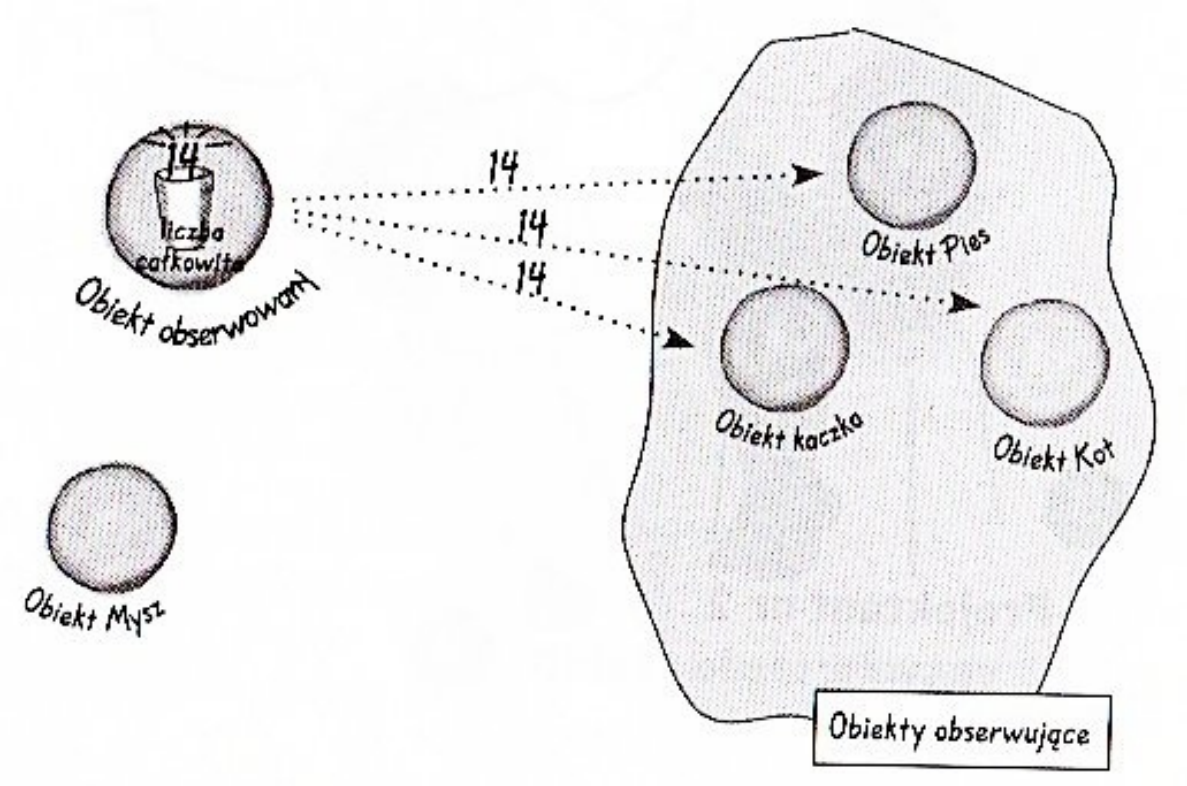


# Obserwator: zasada działania

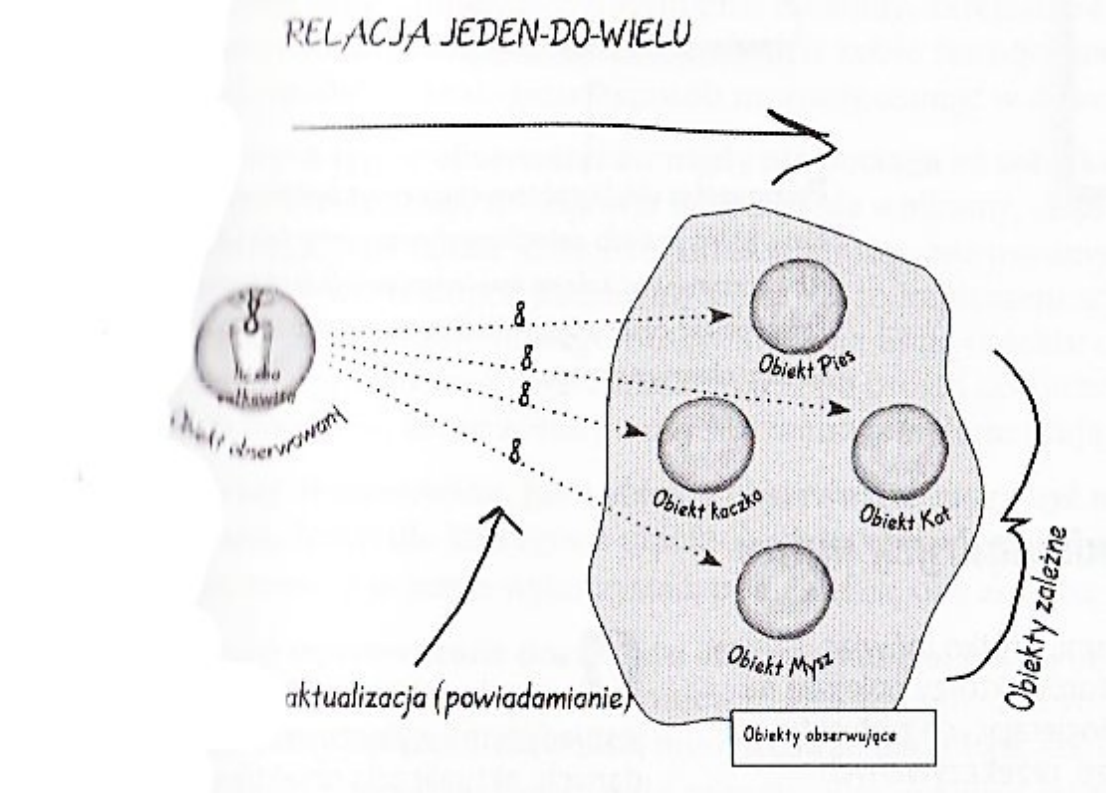


# Obserwator: zasada działania

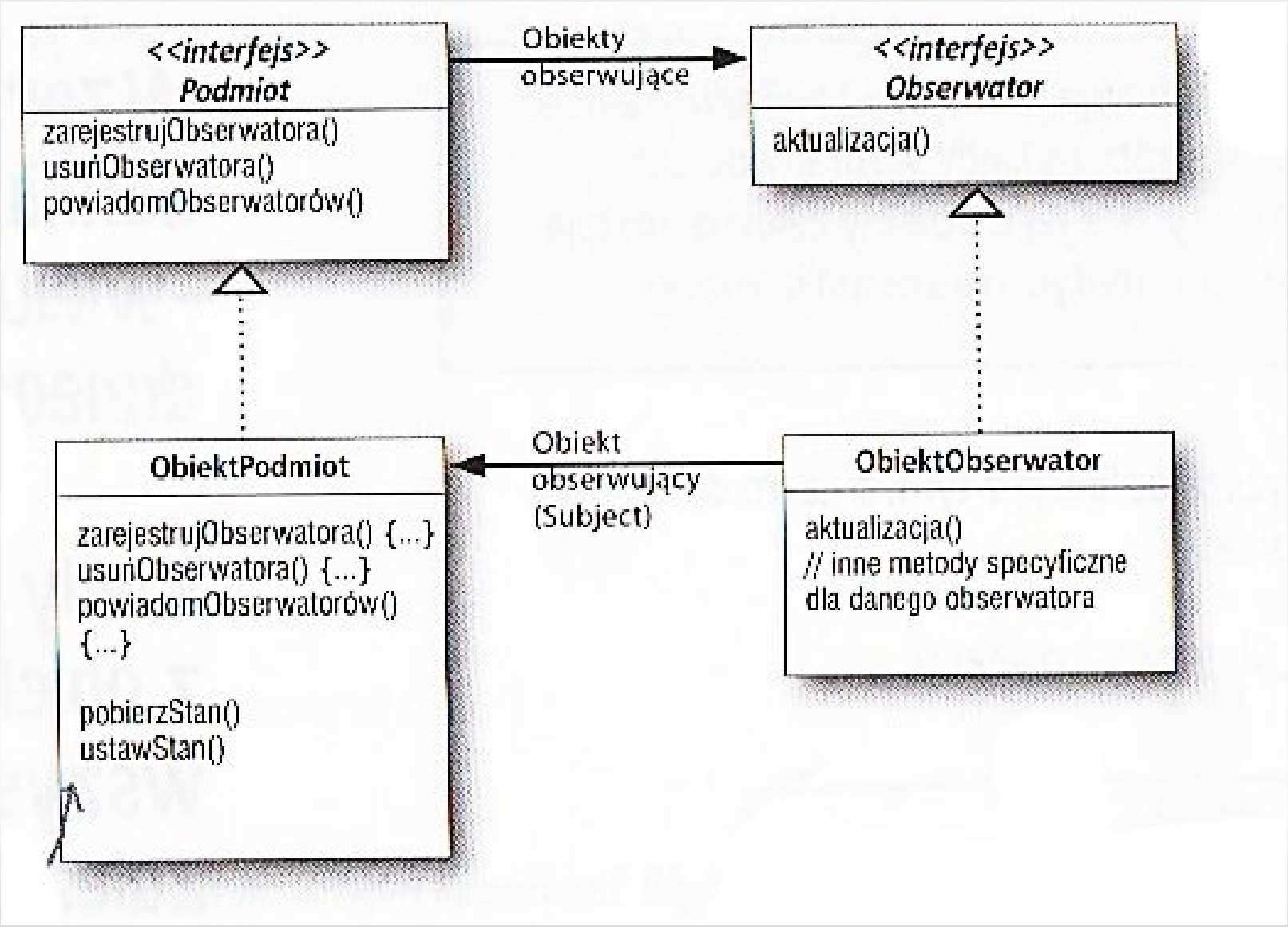


# Relacja jeden-do-wielu

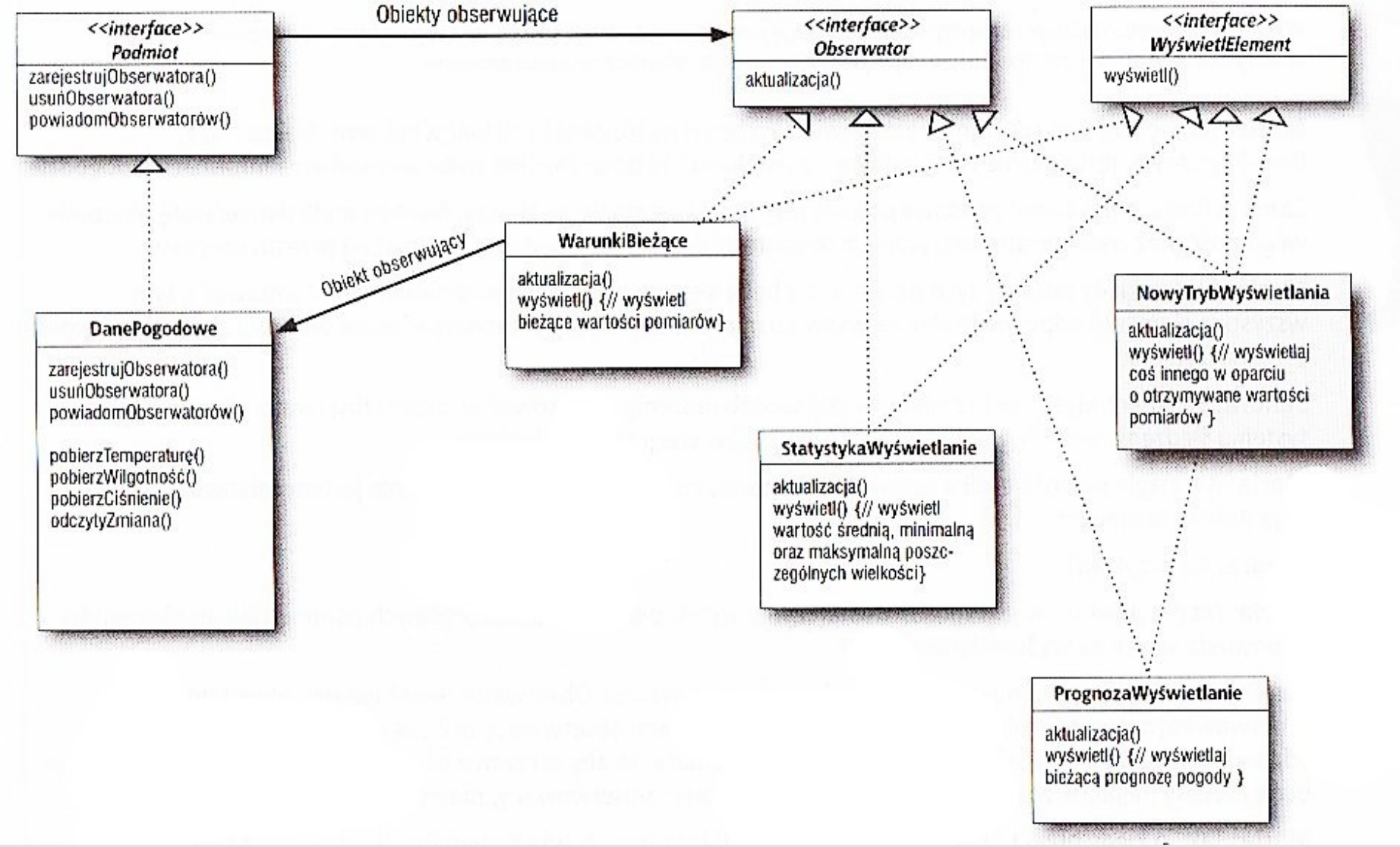



Wzorzec projektowy Obserwator definiuje między obiektami *relację jeden-do-wielu* w taki sposób, że kiedy **wybrany obiekt** (podmiot, obiekt obserwowany) **zmienia stan**, **wszystkie obiekty zależne** (obserwujące) zostają o tym **powiadomione i automatycznie zaktualizowane**.

# Ogólny diagram klas



# Diagram klas dla pogodynki



# Implementacja

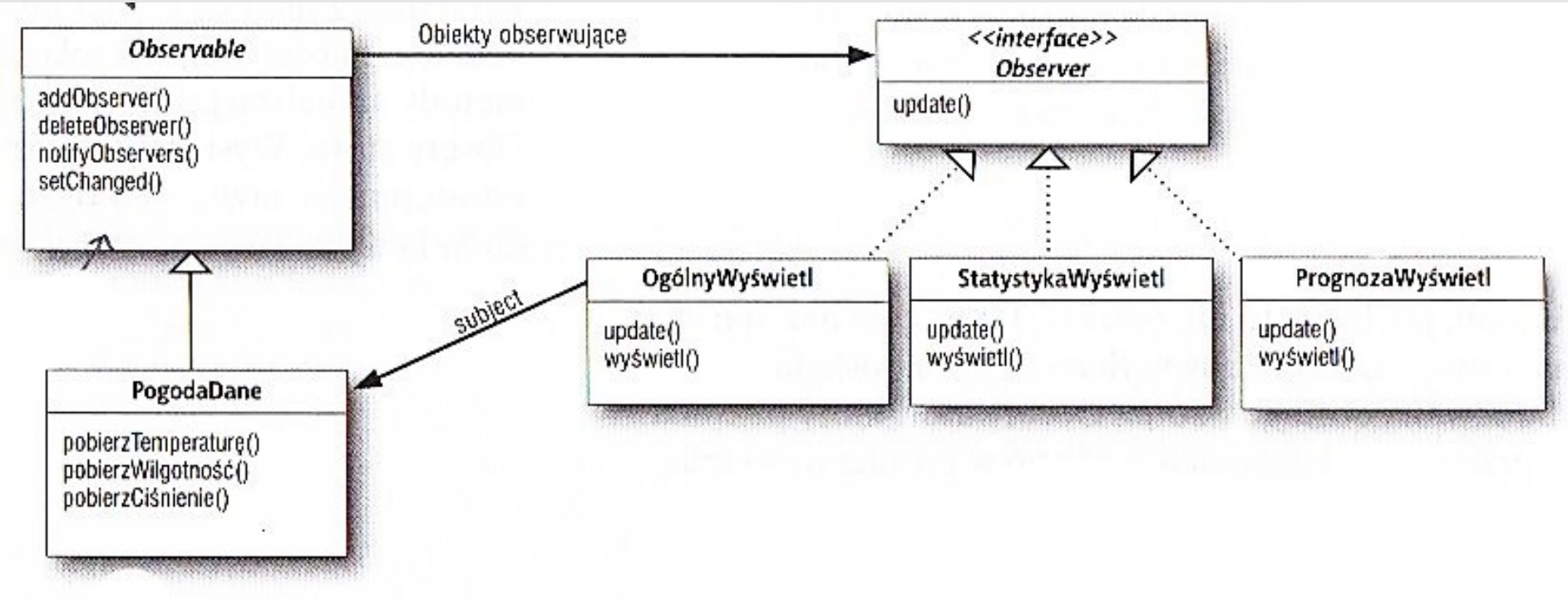
```
public void zarejestrujObserwatora(Obserwator o) {
    obserwatorzy.add(o);
}

public void usuńObserwatora(Obserwator o) {
    int i = obserwatorzy.indexOf(o);
    if (i >= 0) {
        obserwatorzy.remove(i);
    }
}

public void powiadomObserwatorów() {
    for (int i = 0; i < obserwatorzy.size(); i++) {
        Obserwator Obs = (Obserwator)obserwatorzy.get(i);
        Obs.aktualizacja(temperatura, wilgotność, ciśnienie);
    }
}

public void odczytyZmiana() {
    powiadomObserwatorów();
}

public void ustawOdczyty(float temperatura, float wilgotność, float ciśnienie) {
    this.temperatura = temperatura;
    this.wilgotność = wilgotność;
    this.ciśnienie = ciśnienie;
    odczytyZmiana();
}
```

# Implementacja

```
public class WarunkiBieżąceWyświetl implements Obserwator, WyświetlElement {
    private float temperatura;
    private float wilgotność;
    private Podmiot DanePogodowe;

    public WarunkiBieżąceWyświetl(Podmiot DanePogodowe) {
        this.DanePogodowe = DanePogodowe;
        DanePogodowe.zarejestrujObserwatora(this);
    }

    public void aktualizacja(float temperatura, float wilgotność, float ciśnienie) {
        this.temperatura = temperatura;
        this.wilgotność = wilgotność;
        wyświetl();
    }

    public void wyświetl() {
        System.out.println("Warunki bieżące " + temperatura
            + " stopni C oraz " + wilgotność + "% wilgotność");
    }
}
```

# Obserwator w Javie

- klasa **java.util.Observable**
- interfejs **java.util.Observer**



# Obserwator w Javie

```
setChanged() {
    changed = true
}

notifyObservers(Object arg){
    if (changed) {
        dla każdego obserwatora na liście {
            call update(this, arg)
        }
        changed = false
    }
}

notifyObservers(){
    notifyObservers(null)
}
```

# Obserwator w Javie

```
import java.util.Observable;
import java.util.Observer;

public class DanePogodowe extends Observable {
    private float temperatura;
    private float wilgotność;
    private float ciśnienie;

    public DanePogodowe() { }

    public void odczytyZmiania() {
        setChanged();
        notifyObservers();*
    }

    public void ustawOdczyty(float temperatura, float wilgotność, float ciśnienie) {
        this.temperatura = temperatura;
        this.wilgotność = wilgotność;
        this.ciśnienie = ciśnienie;
        odczytyZmiana();
    }

    public float pobierzTemperaturę() {
        return temperatura;
    }

    public float pobierzWilgotność() {
        return wilgotność;
    }

    public float pobierzCiśnienie() {
        return ciśnienie;
    }
}
```

# Obserwator w Javie

```
import java.util.Observable;
import java.util.Observer;

public class WarunkiBieżąceWyświetl implements Observer, WyświetlElement {
    Observable observable;
    private float temperatura;
    private float wilgotność;

    public WarunkiBieżąceWyświetl(Observable observable) {
        this.observable = observable;
        observable.addObserver(this);
    }

    public void update(Observable obs, Object arg) {
        if (obs instanceof DanePogodowe) {
            DanePogodowe danePogodowe = (DanePogodowe)obs;
            this.temperatura = danePogodowe.pobierzTemperaturę();
            this.wilgotność = danePogodowe. pobierzWilgotność();
            wyświetl();
        }
    }

    public void wyświetl() {
        System.out.println("Warunki bieżące " + temperatura
            + "stopni C oraz " + wilgotność + v% wilgotność");
    }
}
```