Monitorizarea Fluctuațiilor Criptomonedelor folosind Design Pattern-ul Observer

1. Introducere

Această aplicație ilustrează utilizarea design pattern-ului Observer pentru a monitoriza fluctuațiile prețurilor unei criptomonede. Scopul este de a urmări și afișa în timp real datele legate de prețuri, utilizând mai multe componente care reacționează automat la schimbările prețului (subiect).

Design pattern-ul Observer este utilizat pentru a permite unui subiect (CryptoSubject) să notifice o listă de observatori atunci când starea sa se modifică.

2. Descrierea Aplicației

Aplicația oferă următoarele funcționalități:

- Monitorizarea prețurilor criptomonedelor.
- Afișarea prețului curent.
- Alarme pentru valori care depășesc sau scad sub anumite praguri.
- Istoric detaliat al fluctuațiilor prețurilor.
- Un grafic în timp real care afișează evoluția prețurilor.

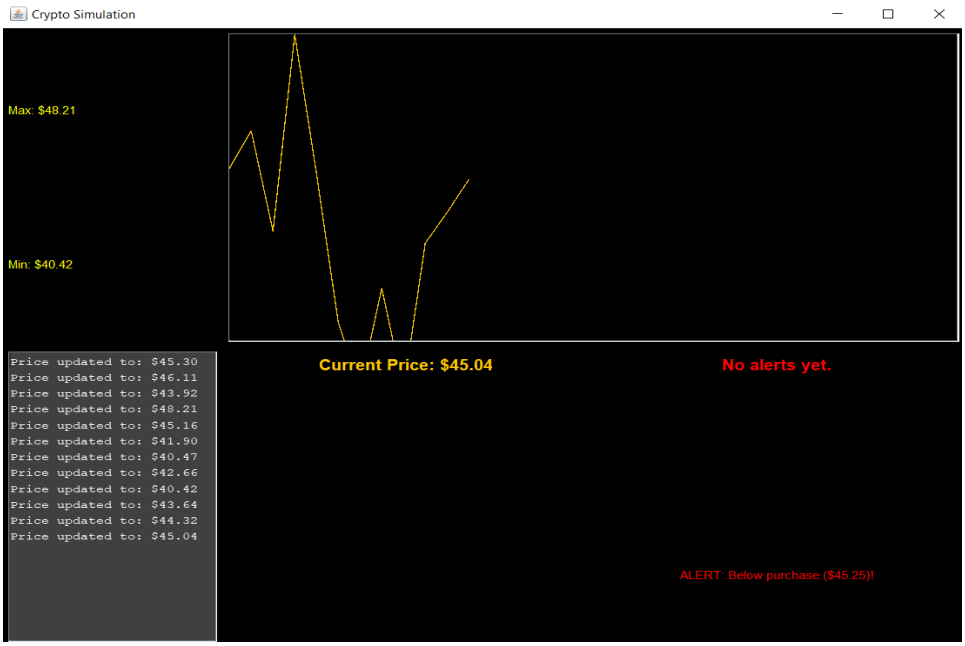
3. Design Pattern-ul Utilizat: Observer

Design pattern-ul Observer este utilizat pentru a permite actualizarea în timp real a mai multor componente (Observers) atunci când CryptoSubject (subiectul) își schimbă starea.

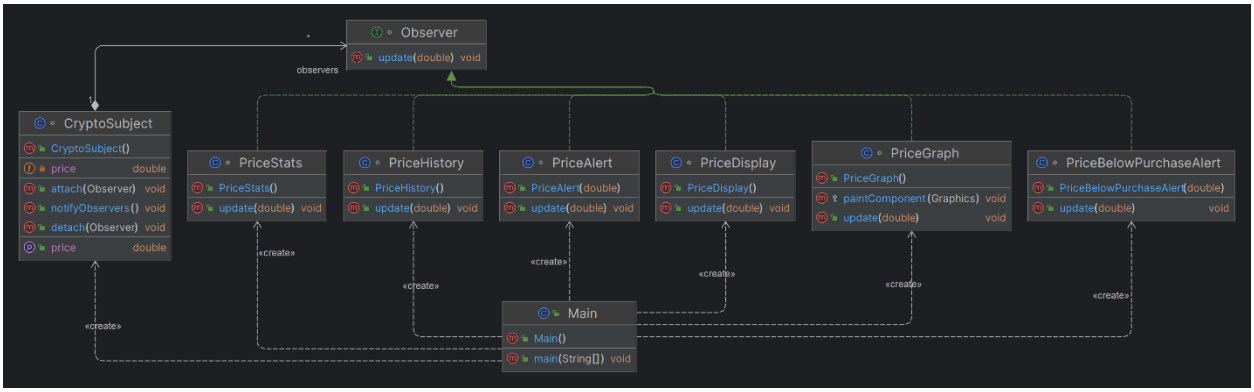
Avantaje:

- Separarea clară a logicii între subiect și observatori.
- Scalabilitate, prin adăugarea de noi observatori fără modificări majore

4 Interfata Gui



4.1 Diagrama UML



5. Implementare

5.1. Clasa Principală: CryptoSimulationApp

```
public class CryptoSimulationApp {
    public static void main(String[] args) {
```

```
        CryptoSubject crypto = new CryptoSubject();
        // Inițializare interfață și observatori
        PriceDisplay display = new PriceDisplay();
        PriceAlert alert = new PriceAlert(100.0);
        PriceHistory history = new PriceHistory();
        JScrollPane graphScrollPane = new JScrollPane();
        PriceGraph graph = new PriceGraph(graphScrollPane);
        PriceBelowPurchaseAlert belowPurchaseAlert = new PriceBelowPurchaseAlert(50.0);

        // Atașare observatori
        crypto.attach(display);
        crypto.attach(alert);
        crypto.attach(history);
        crypto.attach(graph);
        crypto.attach(belowPurchaseAlert);

        // Simulare fluctuații prețuri
        simulatePriceFluctuations(crypto);

        // Configurare UI și afișare
        configureUI(display, alert, history, graphScrollPane, belowPurchaseAlert);
    }
}
```

5.2. Clasa Subiect: CryptoSubject

```
class CryptoSubject {
    private List<Observer> observers = new ArrayList<>();
    private double price;
```

```
    public void attach(Observer observer) { observers.add(observer); }
    public void detach(Observer observer) { observers.remove(observer); }
    public void notifyObservers() {
        for (Observer observer : observers) {
            observer.update(price);
        }
    }
    public void setPrice(double price) { this.price = price; notifyObservers(); }
}
```

5.3. Interfața Observer

```
interface Observer {
    void update(double price);
}
```

5.4. Observatori

1. PriceDisplay:
    - Afișează prețul curent.
2. PriceAlert:
    - Notifică dacă prețul depășește pragul specificat.
3. PriceHistory:
    - Salvează și afișează istoricul prețurilor.
4. PriceGraph:
    - Grafic în timp real al evoluției prețurilor.
5. PriceBelowPurchaseAlert:
    - Notifică dacă prețul este mai mic decât prețul de cumpărare.

6. Instrucțiuni de Utilizare

1. Rulați programul folosind un IDE Java (ex. IntelliJ IDEA, Eclipse) sau din linia de comandă.
2. Aplicația simulează fluctuațiile prețurilor criptomonedelor la fiecare secundă.
3. Observați alertele și graficele generate în timp real.

7. Concluzii

Această aplicație demonstrează utilizarea eficientă a design pattern-ului Observer pentru a construi un sistem scalabil și reactiv. Funcționalitățile sunt organizate modular, ceea ce permite adăugarea ușoară a noilor observatori.