

TUGAS TP MODUL 13



Disusun Oleh :

Pradana Argo Pangestu

2311104079

Kelas :SE-07-02

Dosen :

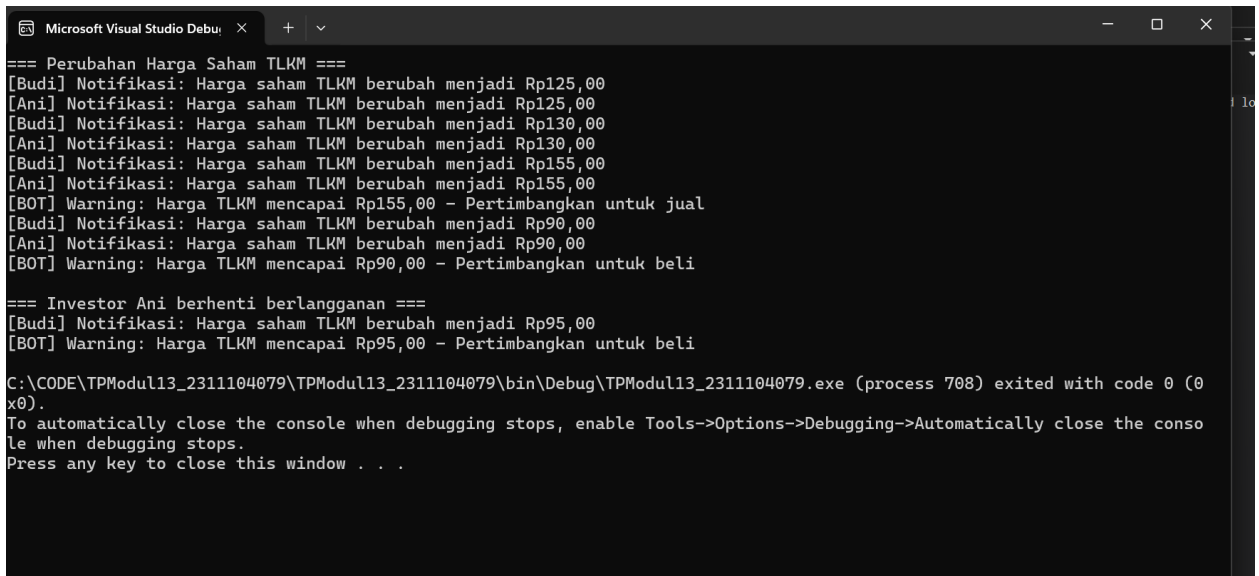
Yudha Islami Sulistya

**PROGRAM STUDI SOFTWARE ENGINEERING
DIREKTORAT KAMPUS PURWOKERTO
TELKOM UNIVERSITY PURWOKERTO
2025**

I. Link Github

[https://github.com/Pradana123/KPL_PradanaAP_2311104079/tree/main/12 Performance Analysis Unit Testing dan Debbing](https://github.com/Pradana123/KPL_PradanaAP_2311104079/tree/main/12%20Performance%20Analysis%20Unit%20Testing%20dan%20Debbing)

I. Ss Output :



```
Microsoft Visual Studio Debug Console
=== Perubahan Harga Saham TLKM ===
[Budi] Notifikasi: Harga saham TLKM berubah menjadi Rp125,00
[Ani] Notifikasi: Harga saham TLKM berubah menjadi Rp125,00
[Budi] Notifikasi: Harga saham TLKM berubah menjadi Rp130,00
[Ani] Notifikasi: Harga saham TLKM berubah menjadi Rp130,00
[Budi] Notifikasi: Harga saham TLKM berubah menjadi Rp155,00
[Ani] Notifikasi: Harga saham TLKM berubah menjadi Rp155,00
[BOT] Warning: Harga TLKM mencapai Rp155,00 - Pertimbangkan untuk jual
[Budi] Notifikasi: Harga saham TLKM berubah menjadi Rp90,00
[Ani] Notifikasi: Harga saham TLKM berubah menjadi Rp90,00
[BOT] Warning: Harga TLKM mencapai Rp90,00 - Pertimbangkan untuk beli

=== Investor Ani berhenti berlangganan ===
[Budi] Notifikasi: Harga saham TLKM berubah menjadi Rp95,00
[BOT] Warning: Harga TLKM mencapai Rp95,00 - Pertimbangkan untuk beli

C:\CODE\TPModul13_2311104079\TPModul13_2311104079\bin\Debug\TPModul13_2311104079.exe (process 708) exited with code 0 (0x0).
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

Penjelasan Design Pattern Observer

A. Sistem Trading Saham

Dalam sistem trading saham, Observer Pattern dapat digunakan untuk:

- **Notifikasi Perubahan Harga Saham:** Ketika harga saham tertentu berubah, semua investor/trader yang berlangganan (subscribe) saham tersebut akan mendapatkan notifikasi secara otomatis.
- **Portofolio Monitoring:** Investor bisa memantau beberapa saham sekaligus dalam portofolionya dan mendapatkan update real-time ketika ada perubahan.
- **Trading Automation:** Sistem trading otomatis bisa bereaksi langsung ketika kondisi tertentu terpenuhi (contoh: harga mencapai level tertentu).

B. Implementasi Observer Pattern

1. Buat Subject (Publisher):

- o Kelas `StockMarket` yang menyimpan daftar saham dan harganya
- o Memiliki method untuk mengupdate harga saham

2. Buat Observer (Subscriber):

- o Interface `IStockObserver` dengan method `Update()`
- o Implementasi konkret seperti `Investor` atau `TradingBot`

3. Mekanisme Subscribe/Unsubscribe:

- o Method `Register()` dan `Unregister()` di Subject
- o Method `NotifyObservers()` untuk mengirim update ke semua observer

4. Trigger Update:

- o Ketika harga saham berubah, panggil `NotifyObservers()`

C. Kelebihan dan Kekurangan

Kelebihan:

- Investor bisa memilih saham mana saja yang ingin dipantau
- Perubahan harga bisa di-broadcast ke ribuan investor sekaligus
- Mudah menambah jenis observer baru (email notifikasi, SMS, dll)

Kekurangan:

- Jika terlalu banyak observer, bisa mempengaruhi performa
- Urutan notifikasi tidak bisa dijamin
- Potensi memory leak jika observer tidak di-unregister dengan benar

3. Implementasi Observer Pattern untuk Sistem Trading Saham

Kode Program.cs

```
1. using System;
2. using System.Collections.Generic;
3.
4. namespace TModul13_2311104079
5. {
6.     public class StockMarket
7.     {
8.         private string _symbol;
9.         private decimal _price;
```

```
10.     private List<Istockobserver> _observers = new
List<Istockobserver>();
11.
12.     public StockMarket(string symbol, decimal price)
13.     {
14.         _symbol = symbol;
15.         _price = price;
16.     }
17.
18.     public decimal Price
19.     {
20.         get { return _price; }
21.         set
22.         {
23.             if (_price != value)
24.             {
25.                 _price = value;
26.                 NotifyObservers();
27.             }
28.         }
29.     }
30.
31.     public void RegisterObserver(IstockObserver observer)
32.     {
33.         _observers.Add(observer);
34.     }
35.
36.     public void UnregisterObserver(IstockObserver observer)
37.     {
38.         _observers.Remove(observer);
39.     }
40.
41.     private void NotifyObservers()
42.     {
43.         foreach (var observer in _observers)
44.         {
45.             observer.Update(_symbol, _price);
46.         }
47.     }
48. }
49.
50. // Observer Interface
51. public interface IstockObserver
52. {
53.     void Update(string symbol, decimal price);
54. }
55.
56. // Concrete Observer 1: Investor
57. public class Investor : IstockObserver
58. {
59.     private string _name;
60.
61.     public Investor(string name)
62.     {
63.         _name = name;
64.     }
65.
66.     public void Update(string symbol, decimal price)
67.     {
68.         Console.WriteLine($"[{_name}] Notifikasi: Harga saham
{symbol} berubah menjadi {price:C}");
69.     }
70. }
71.
```

```
72. // Concrete Observer 2: Trading Bot
73. public class TradingBot : IStockObserver
74. {
75.     public void Update(string symbol, decimal price)
76.     {
77.         if (price > 150m)
78.         {
79.             Console.WriteLine($"[BOT] Warning: Harga {symbol}
mencapai {price:C} - Pertimbangkan untuk jual");
80.         }
81.         else if (price < 100m)
82.         {
83.             Console.WriteLine($"[BOT] Warning: Harga {symbol}
mencapai {price:C} - Pertimbangkan untuk beli");
84.         }
85.     }
86. }
87.
88. class Program
89. {
90.     static void Main(string[] args)
91.     {
92.         // Buat subject (stock market)
93.         var telkomStock = new StockMarket("TLKM", 120m);
94.
95.         // Buat observer (investor dan bot)
96.         var investor1 = new Investor("Budi");
97.         var investor2 = new Investor("Ani");
98.         var tradingBot = new TradingBot();
99.
100.        // Daftarkan observer
101.        telkomStock.RegisterObserver(investor1);
102.        telkomStock.RegisterObserver(investor2);
103.        telkomStock.RegisterObserver(tradingBot);
104.
105.        // Simulasikan perubahan harga
106.        Console.WriteLine("=== Perubahan Harga Saham TLKM
===");
107.        telkomStock.Price = 125m;
108.        telkomStock.Price = 130m;
109.        telkomStock.Price = 155m; // Bot akan memberi warning
110.        telkomStock.Price = 90m; // Bot akan memberi warning
111.
112.        // Unregister salah satu investor
113.        telkomStock.UnregisterObserver(investor2);
114.
115.        Console.WriteLine("\n=== Investor Ani berhenti
berlangganan ===");
116.        telkomStock.Price = 95m;
117.    }
118. }
119. }
```

Kesimpulan

Kode ini adalah implementasi yang sangat bagus dari **Observer Pattern**:

- **Subject (StockMarket)** menyimpan status (harga) dan memberitahu semua **observers** saat berubah.
- **Observers (Investor dan TradingBot)** melakukan tindakan berbeda saat diberi tahu.

- Cocok untuk skenario seperti **notifikasi harga saham, sensor data, atau sistem event-driven.**