Nama : Naufal Aflakh Wijayanto

NIM : 2211104073

Kelas : SE063

Menjelaskan Design Pattern “Observer”

1. **Contoh Penggunaan Design Pattern Observer**:
    - **Contoh 1**: Penggunaan pada sistem notifikasi di aplikasi. Ketika ada update atau perubahan (misalnya update berita atau status), maka Observer yang berlangganan akan diberitahu secara otomatis untuk menampilkan informasi terbaru tersebut.
2. **Langkah-langkah Mengimplementasikan Design Pattern Observer**:
    - **Langkah 1**: Membuat objek Subject yang memiliki daftar observer yang terdaftar.
    - **Langkah 2**: Observer akan mendaftar (subscribe) ke Subject agar dapat menerima pemberitahuan.
    - **Langkah 3**: Ketika ada perubahan dalam Subject, objek ini akan memanggil metode notifyObservers(), yang akan memberitahukan semua observer yang terdaftar mengenai perubahan tersebut.
3. **Kelebihan dan Kekurangan Design Pattern Observer**:
    - **Kelebihan**:
        - Memungkinkan komunikasi yang longgar antara objek (tanpa perlu ketergantungan yang ketat).
        - Menjaga konsistensi antara objek yang terhubung, karena observer secara otomatis mendapat informasi terbaru.
    - **Kekurangan**:
        - Jika terlalu banyak observer, dapat menyebabkan masalah kinerja.
        - Dapat menyebabkan kerumitan dalam pengelolaan hubungan antara objek, terutama ketika banyak observer harus diperbarui.

Implementasi dan Pemahaman Design Pattern Observer

HTML yang berfungsi sebagai antarmuka pengguna (GUI) untuk tugas ini

Index.html

```
<!DOCTYPE html>
<html lang="id">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Observer Pattern Implementation</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <div class="container">
    <h1>Implementasi Design Pattern Observer</h1>
    <button onclick="addObserver()">Tambah Observer</button>
    <button onclick="notifyObservers()">Notifikasi ke Observer</button>
    <div id="output"></div> <!-- Tempat untuk menampilkan hasil -->
  </div>
  <script src="script.js"></script>
</body>
</html>
```

- Menyediakan antarmuka dengan tombol untuk menambah observer dan memberitahu observer.
- id="output" digunakan untuk menampilkan hasil notifikasi yang diterima oleh observer di halaman web.

Styles.css

```
/* Styling untuk seluruh halaman */
body {
  font-family: Arial, sans-serif;
  margin: 0;
  padding: 0;
  background-color: #f4f4f9;
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
}

/* Styling untuk container utama */
.container {
  text-align: center;
  background-color: #fff;
  padding: 30px;
  border-radius: 10px;
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
  width: 400px;
}

/* Styling untuk heading */
h1 {
  color: #333;
  font-size: 1.8em;
  margin-bottom: 20px;
}

/* Styling untuk tombol */
button {
  padding: 10px 20px;
  font-size: 1em;
  border: none;
  background-color: #007bff;
  color: white;
  border-radius: 5px;
  cursor: pointer;
  margin: 10px 5px;
  transition: background-color 0.3s ease;
}
```

```
/* Styling untuk tombol saat hover */
button:hover {
  background-color: #0056b3;
}

/* Styling untuk hasil data yang dicetak */
#output {
  margin-top: 20px;
  text-align: left;
  color: #333;
  font-size: 1em;
  font-family: 'Courier New', Courier, monospace;
  white-space: pre-wrap; /* Membiarkan teks untuk wrap sesuai lebar */
  margin-bottom: 20px;
}

/* Styling untuk setiap elemen data */
#output p {
  margin: 5px 0;
  padding: 5px;
  background-color: #f8f9fa;
  border-radius: 5px;
  border: 1px solid #ddd;
}
```

- **body dan .container memberikan styling untuk seluruh halaman agar terlihat rapi dan responsif.**
- **Tombol diberi warna biru dan efek hover untuk tampilan yang lebih interaktif.**
- **#output digunakan untuk menampilkan data hasil dari notifyObservers().**

Membuat Implementasi Observer Pattern di script.js:
Di bawah ini adalah contoh implementasi Observer Pattern menggunakan JavaScript. Ini mengikuti konsep yang sama seperti yang diberikan di contoh C# pada halaman referensi.

Script.js

```
// Subject (yang mempublikasikan perubahan)
class Subject {
  constructor() {
    this.observers = []; // Daftar observer yang terdaftar
  }

  // Method untuk mendaftar observer
  addObserver(observer) {
    this.observers.push(observer);
  }

  // Method untuk memberitahu observer jika ada perubahan
  notifyObservers(message) {
    this.observers.forEach(observer => observer.update(message));
  }
}

// Observer (yang menerima notifikasi dari Subject)
class Observer {
  constructor(name) {
    this.name = name;
  }

  // Method yang dipanggil oleh Subject untuk mengirim notifikasi
  update(message) {
    console.log(`${this.name} menerima pesan: ${message}`);
    document.getElementById('output').innerHTML += `${this.name} menerima pesan:
${message}<br>`;
  }
}

// Inisialisasi Subject dan Observer
const subject = new Subject();
const observer1 = new Observer("Observer 1");
const observer2 = new Observer("Observer 2");

// Menambahkan observer ke subject
subject.addObserver(observer1);
```

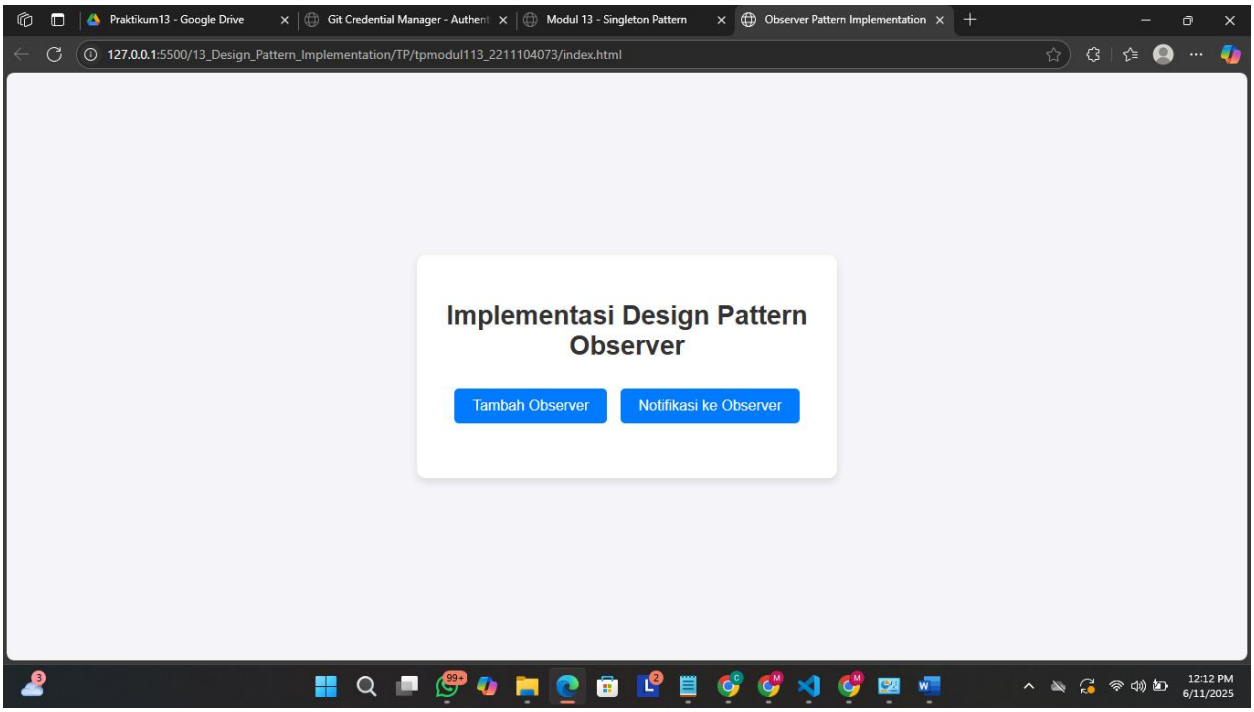
```
subject.addObserver(observer2);

// Fungsi untuk menambahkan observer baru
function addObserver() {
  const observer3 = new Observer("Observer 3");
  subject.addObserver(observer3);
  document.getElementById('output').innerHTML += "Observer 3 ditambahkan.<br>";
}

// Fungsi untuk memberitahu semua observer
function notifyObservers() {
  subject.notifyObservers("Update terbaru telah tersedia!");
}
```

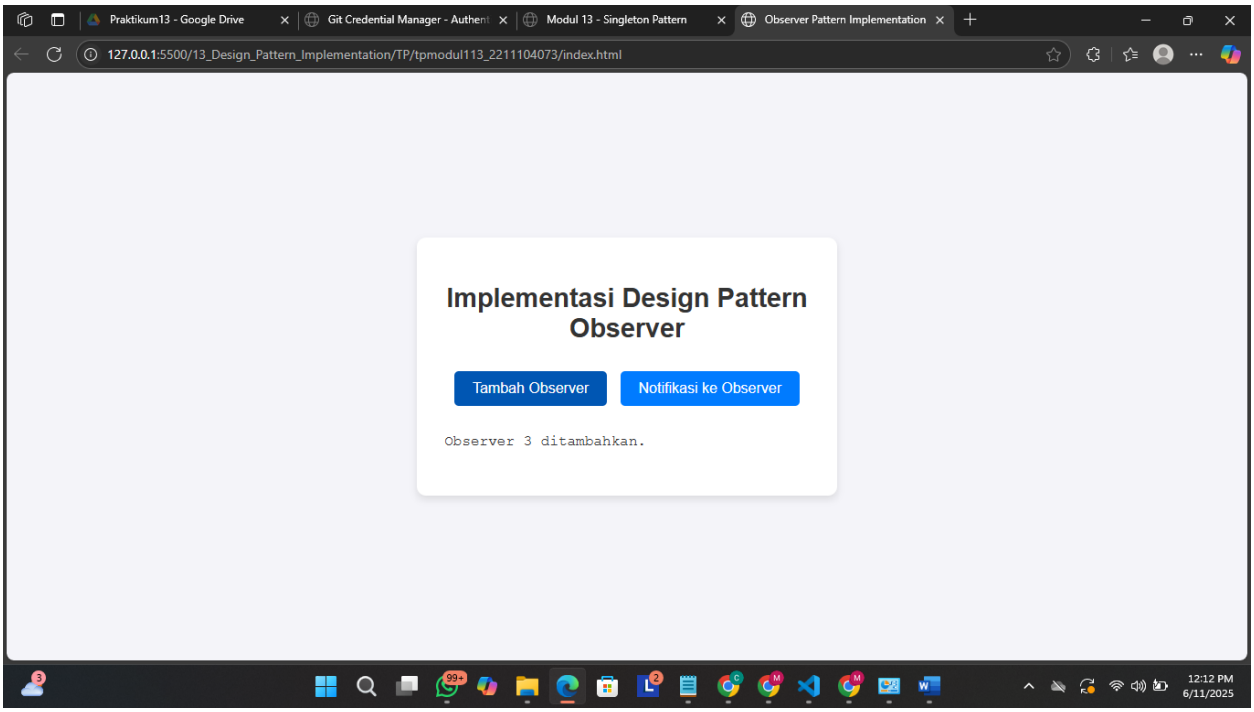
- **Class Subject**: Menyimpan daftar observer dan memiliki metode addObserver() untuk menambahkan observer dan notifyObservers() untuk memberitahukan observer jika ada perubahan.
- **Class Observer**: Menerima pemberitahuan dari Subject dan menampilkan pesan yang diterima di halaman web.
- **Metode addObserver()**: Digunakan untuk menambahkan observer baru.
- **Metode notifyObservers()**: Memanggil metode update() pada setiap observer untuk memberi notifikasi.

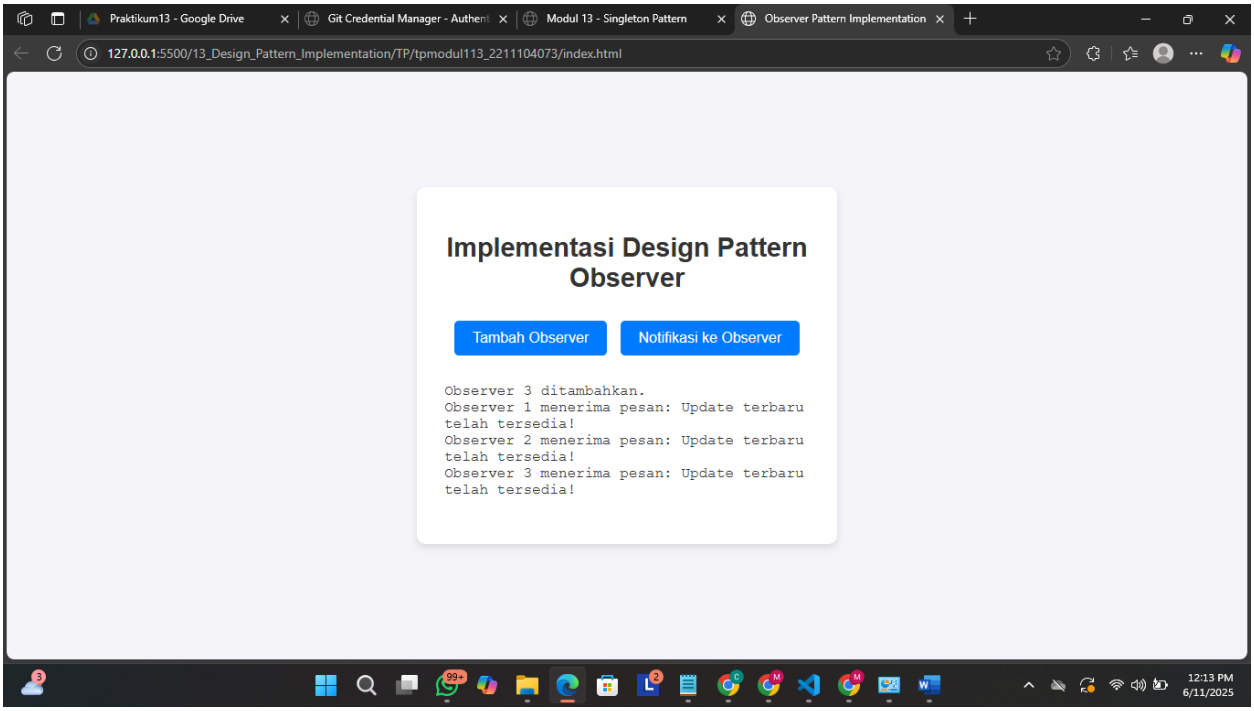
Fungsi utama:

- **addObserver()**: Menambahkan observer baru (Observer 3) ke daftar observer dan menampilkan pesan di halaman web.
- **notifyObservers()**: Memberitahu semua observer mengenai update terbaru dan menampilkan pesan yang diterima di halaman web.

Output:


Praktikum13 - Google Drive
Git Credential Manager - Authent
Modul 13 - Singleton Pattern
Observer Pattern Implementation
127.0.0.1:5500/13_Design_Pattern_Implementation/TP/tpmodul113_2211104073/index.html
Implementasi Design Pattern Observer
Tambah Observer
Notifikasi ke Observer
12:12 PM
6/11/2025


Praktikum13 - Google Drive
Git Credential Manager - Authent
Modul 13 - Singleton Pattern
Observer Pattern Implementation
127.0.0.1:5500/13_Design_Pattern_Implementation/TP/tpmodul113_2211104073/index.html
Implementasi Design Pattern Observer
Tambah Observer
Notifikasi ke Observer
Observer 3 ditambahkan.
12:12 PM
6/11/2025


Praktikum13 - Google Drive
Git Credential Manager - Authent
Modul 13 - Singleton Pattern
Observer Pattern Implementation
127.0.0.1:5500/13_Design_Pattern_Implementation/TP/tpmodul113_2211104073/index.html
Implementasi Design Pattern Observer
Tambah Observer
Notifikasi ke Observer
Observer 3 ditambahkan.
Observer 1 menerima pesan: Update terbaru telah tersedia!
Observer 2 menerima pesan: Update terbaru telah tersedia!
Observer 3 menerima pesan: Update terbaru telah tersedia!
12:13 PM
6/11/2025