Nama : Salman Alfarisi

Kelas : S1SE-07-01

NIM : 2311104036

2.A Contoh Penggunaan Observer

Salah satu contoh nyata adalah sistem notifikasi media sosial.
Ketika pengguna (subjek) memposting sesuatu, semua pengikut (observer) akan menerima pemberitahuan secara otomatis.

2.B **Langkah-langkah Implementasi Observer**

1. **Buat class Subject (subjek utama)**:
    - Menyimpan daftar observer.
    - Menyediakan method untuk menambahkan, menghapus, dan memberi notifikasi ke observer.
2. **Buat interface atau class Observer**:
    - Memiliki method seperti update() yang akan dipanggil oleh subject saat terjadi perubahan.
3. **Saat terjadi perubahan pada subject**, maka subject akan memanggil method update() untuk setiap observer yang terdaftar.

2. C Kelebihan dan Kekurangan Observer

Kelebihan:

1. Memisahkan subject dan observer → lebih fleksibel dan modular.
2. Mendukung pola event-driven (mirip publish-subscribe).
3. Observer dapat ditambahkan atau dihapus tanpa mengubah subject.

Kekurangan:

1. Sulit untuk melacak alur program jika terlalu banyak observer.
2. Kinerja menurun jika jumlah observer besar (banyak notifikasi).
3. Risiko referensi sirkular/memory leak jika observer tidak dilepas dengan benar.

```
// Subject (Publisher)
class NewsAgency {
  constructor() {
    this.observers = [];
    this.latestNews = "";
  }

  // Menambahkan observer
  addObserver(observer) {
    this.observers.push(observer);
  }

  // Menghapus observer
  removeObserver(observer) {
    this.observers = this.observers.filter(obs => obs !== observer);
  }

  // Memberi tahu semua observer
  notifyObservers() {
    this.observers.forEach(observer => observer.update(this.latestNews));
  }

  // Mengatur berita baru dan memberi tahu observer
  setNews(news) {
    this.latestNews = news;
    this.notifyObservers();
  }
}

// Observer
class NewsReader {
  constructor(name) {
    this.name = name;
  }

  update(news) {
    console.log(`${this.name} menerima berita baru: "${news}"`);
  }
}

// Export untuk digunakan di main.js
module.exports = { NewsAgency, NewsReader };
```

```
const { NewsAgency, NewsReader } = require('./ObserverPattern');

function main() {
  // Membuat instance Subject
  const agency = new NewsAgency();

  // Membuat beberapa observer
  const reader1 = new NewsReader("Andi");
  const reader2 = new NewsReader("Budi");
  const reader3 = new NewsReader("Cici");

  // Menambahkan observer ke agency
  agency.addObserver(reader1);
  agency.addObserver(reader2);
  agency.addObserver(reader3);

  // Mengirim berita baru
  agency.setNews("Berita hari ini: Design Pattern Observer dipelajari!");
}

main();
```

Kode ini merupakan implementasi dari pola desain Observer Pattern, yang memungkinkan suatu objek (dalam hal ini NewsAgency sebagai subject) untuk memberi tahu objek lain (dalam hal ini NewsReader sebagai observer) ketika terjadi perubahan data. Sebuah instance NewsAgency dibuat dan tiga pembaca berita (NewsReader) yaitu Andi, Budi, dan Cici juga diinstansiasi. Ketiganya didaftarkan sebagai observer melalui metode addObserver. Ketika agency.setNews(...) dipanggil dengan berita baru, method tersebut mengubah berita terakhir dan langsung memanggil notifyObservers(), yang secara otomatis memberi tahu semua observer dengan memanggil method update() mereka. Hasilnya, ketiga pembaca akan menerima dan mencetak berita baru secara bersamaan ke konsol. Ini menunjukkan bagaimana subject dapat secara otomatis menyebarkan perubahan ke semua observer yang terdaftar, sesuai prinsip Observer Pattern.