Nama : Salman Alfarisi

Kelas : S1SE-07-01

NIM : 231104036

```
/**
 * Class NewsReader adalah Observer yang menerima update dari NewsAgency.
 */
class NewsReader {
  /**
   * @param {string} name - Nama pembaca berita
   */
  constructor(name) {
    this.name = name;
  }

  /**
   * Method ini dipanggil saat ada update berita dari agency.
   * @param {string} news - Isi berita
   */
  update(news) {
    console.log(`${this.name} menerima berita baru: "${news}"`);
  }
}

module.exports = NewsReader;
```

```
/**
 * Class NewsAgency berperan sebagai Subject (Publisher)
 * yang menyimpan dan mengelola daftar observer.
 */
class NewsAgency {
  constructor() {
    this.observers = [];
    this.latestNews = '';
  }

  /**
   * Menambahkan observer baru ke daftar.
   * @param {object} observer - Observer yang akan menerima update
   */
  addObserver(observer) {
    this.observers.push(observer);
  }

  /**
   * Menghapus observer dari daftar.
   * @param {object} observer - Observer yang ingin dihapus
   */
  removeObserver(observer) {
    this.observers = this.observers.filter((obs) => obs !== observer);
  }

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

  /**
   * Mengatur berita terbaru dan memberi tahu observer.
   * @param {string} news - Berita baru yang akan dikirim
   */
  setNews(news) {
    this.latestNews = news;
    this.notifyObservers();
  }
}

module.exports = NewsAgency;
```

```
const NewsAgency = require('./NewsAgency');
const NewsReader = require('./NewsReader');

/**
 * Fungsi utama menjalankan alur Observer Pattern:
 * membuat agency, menambahkan observer, dan mengirim berita.
 */
function main() {
  const agency = new NewsAgency();

  const reader1 = new NewsReader('Andi');
  const reader2 = new NewsReader('Budi');
  const reader3 = new NewsReader('Cici');

  agency.addObserver(reader1);
  agency.addObserver(reader2);
  agency.addObserver(reader3);

  agency.setNews('Berita hari ini: Design Pattern Observer dipelajari!');
}

main();
```

Kode ini mengimplementasikan Observer Pattern, sebuah pola desain perangkat lunak di mana satu objek (subject atau publisher) mengelola dan memberi tahu sekumpulan objek lainnya (observers) ketika terjadi perubahan. Dalam kasus ini, NewsAgency berperan sebagai subject, sedangkan NewsReader adalah observer yang akan menerima pembaruan berita.

Fungsi main menunjukkan alur kerja pola ini. Pertama, sebuah instance dari NewsAgency dibuat sebagai sumber berita. Lalu, tiga pembaca berita (Andi, Budi, dan Cici) dibuat sebagai instance dari NewsReader. Ketiga pembaca ini didaftarkan ke agency menggunakan method addObserver, yang artinya mereka ingin menerima update setiap kali ada berita baru. Ketika method setNews dipanggil dengan berita baru, NewsAgency menyimpan berita tersebut lalu segera memanggil notifyObservers, yang pada gilirannya akan memanggil method update dari masing-masing observer untuk menyampaikan isi berita. Output-nya akan berupa pesan di konsol yang menunjukkan bahwa setiap pembaca menerima berita terbaru.