2311104025
Eduardo Bagus Prima Julian
SE0701

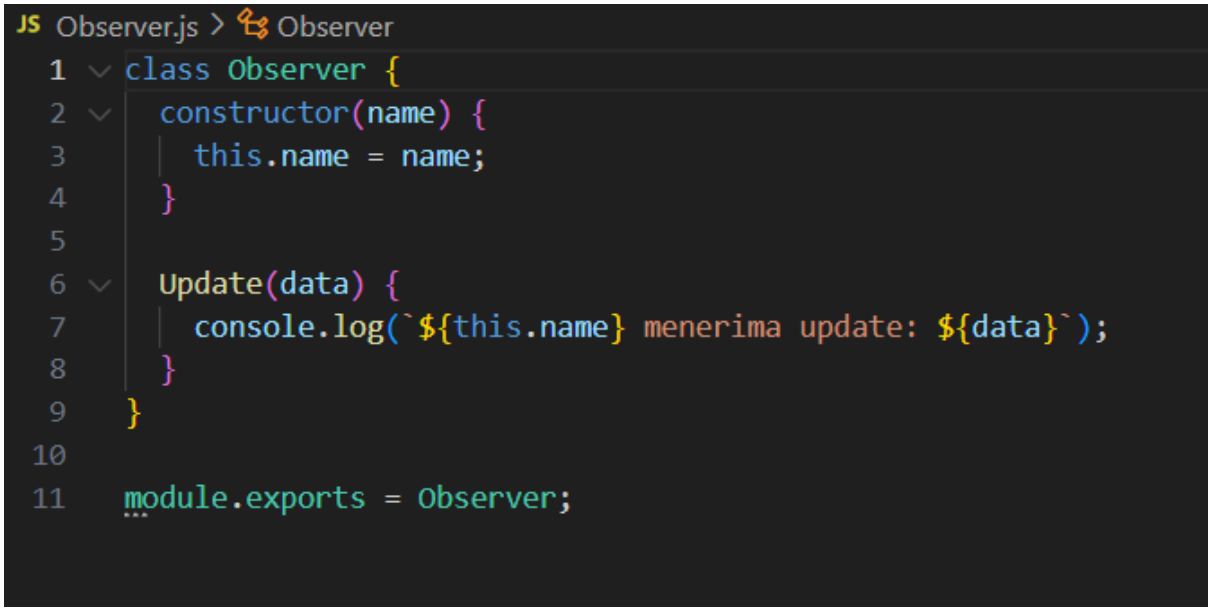
Subject.js

```
JS Subject.js > Subject
1   class Subject {
2     constructor() {
3       this.observers = [];
4     }
5
6     Attach(observer) {
7       this.observers.push(observer);
8     }
9
10    Detach(observer) {
11      this.observers = this.observers.filter((obs) => obs !== observer);
12    }
13
14    Notify(data) {
15      this.observers.forEach((observer) => {
16        observer.Update(data);
17      });
18    }
19  }
20
21  module.exports = Subject;
```

Penjelasan:

Class Subject merupakan inti dari Observer Design Pattern yang berperan sebagai objek yang diamati (publisher). Class ini memiliki properti observers, yaitu array yang menyimpan daftar objek-objek pengamat (subscribers). Method Attach(observer) digunakan untuk menambahkan observer ke daftar pengamat, sedangkan Detach(observer) untuk menghapus observer dari daftar tersebut. Method Notify(data) akan memanggil method Update(data) pada setiap observer yang terdaftar, dengan membawa informasi atau datayang ingin dikirimkan. Dengan demikian, setiap perubahan yang terjadi pada Subject dapat langsung diinformasikan ke seluruh observer secara otomatis.

Obeserver.js



Penjelasan:

Class Observer merepresentasikan objek pengamat dalam Observer Design Pattern, yaitu pihak yang menerima notifikasi dari objek yang diamatinya (Subject). Setiap instance Observer memiliki atribut name untuk mengidentifikasi observer tersebut. Method Update(data) akan dipanggil oleh Subject saat terjadi perubahan atau pembaruan data, dan fungsinya adalah mencetak pesan ke konsol bahwa observer bersangkutan telah menerima update tertentu. Dengan pendekatan ini, setiap observer dapat merespons perubahan dari subject secara mandiri dan terpisah.

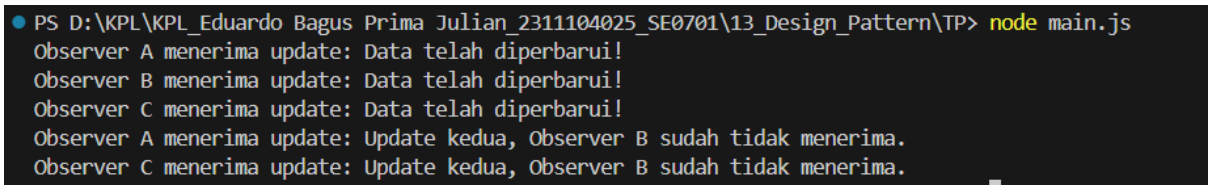
Main.js

```
JS main.js > ...
1   const Subject = require('./Subject');
2   const Observer = require('./Observer');
3
4   // Membuat subject utama (objek yang diamati)
5   const pusatData = new Subject();
6
7   // Membuat observer (pengamat)
8   const observer1 = new Observer("Observer A");
9   const observer2 = new Observer("Observer B");
10  const observer3 = new Observer("Observer C");
11
12  // Menambahkan observer ke subject
13  pusatData.Attach(observer1);
14  pusatData.Attach(observer2);
15  pusatData.Attach(observer3);
16
17  // Memberi notifikasi ke semua observer
18  pusatData.Notify("Data telah diperbarui!");
19
20  // Melepas salah satu observer
21  pusatData.Detach(observer2);
22
23  // Memberi notifikasi lagi
24  pusatData.Notify("Update kedua, Observer B sudah tidak menerima.");
```

Penjelasan:

Pertama, program meng-import class Subject dan Observer. Lalu dibuat sebuah instance pusatData dari class Subject, yang berperan sebagai objek yang diamati (publisher). Kemudian, tiga observer (observer1, observer2, dan observer3) dibuat dengan nama yang berbeda agar bisa dikenali saat menerima notifikasi. Ketiga observer tersebut ditambahkan ke dalam pusatData melalui method Attach, sehingga mereka akan menerima notifikasi setiap kali ada perubahan. Selanjutnya, pusatData.Notify("Data telah diperbarui!") digunakan untuk mengirim pemberitahuan ke semua observer yang terdaftar, dan setiap observer akan menjalankan method Update() untuk mencetak pesan ke konsol. Setelah itu, observer2 dihapus dari daftar pengamat dengan method Detach, sehingga saat pusatData mengirimkan notifikasi kedua (Notify("Update kedua, Observer B sudah tidak menerima.")), hanya observer1 dan observer3 yang menerima pesan. Ini menunjukkan bahwa perubahan pada Subject hanya dikirimkan ke observer yang masih aktif/terdaftar.

Output:


PS D:\KPL\KPL_Eduardo Bagus Prima Julian_2311104025_SE0701\13_Design_Pattern\TP> node main.js
Observer A menerima update: Data telah diperbarui!
Observer B menerima update: Data telah diperbarui!
Observer C menerima update: Data telah diperbarui!
Observer A menerima update: Update kedua, Observer B sudah tidak menerima.
Observer C menerima update: Update kedua, Observer B sudah tidak menerima.