Zulfa M.A.I

2311104010

TP

Project ini merupakan hasil refactoring dari project Modul 13 yang menerapkan **Observer Pattern** menggunakan JavaScript. Dalam kode ini, terdapat tiga kelas utama, yaitu WeatherStation, TemperatureDisplay, dan TemperatureLogger.

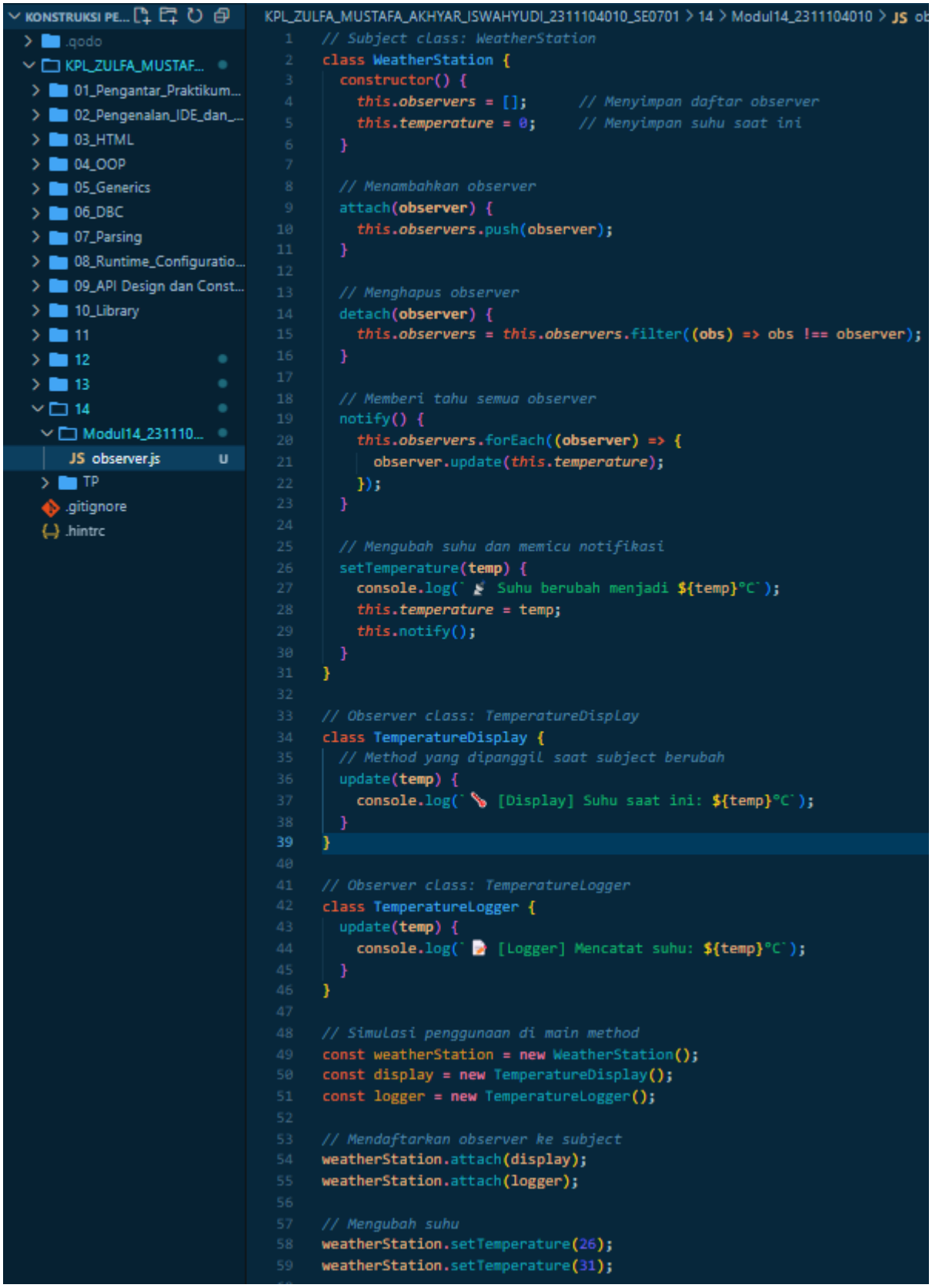
Disini saya ambil dari code Observer dari TP ke-13 saja, berikut refactoring yang dilakukan:

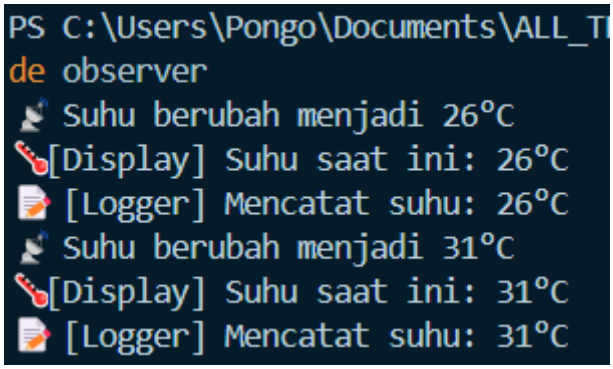
Aspek	Deskripsi Refactoring
Naming	Menggunakan camelCase untuk method dan atribut
Indentation	Konsisten 2 spasi, rapi
Whitespace	Ada jarak antar method untuk keterbacaan
Comments	Komentar singkat menjelaskan fungsi tiap bagian
Emoji opsional	Ditambahkan untuk visual feedback saat simulasi dijalankan

Kelas WeatherStation bertugas sebagai **subject**, yaitu objek pusat yang menyimpan data suhu. Ketika suhu diperbarui melalui method setTemperature(), objek ini akan memanggil notify() yang bertugas memberi tahu semua observer tentang perubahan tersebut.

Kelas TemperatureDisplay dan TemperatureLogger adalah dua **observer** yang masing-masing menampilkan dan mencatat suhu terbaru. Keduanya memiliki method update() yang akan dipanggil oleh subject saat ada perubahan.

Struktur ini memungkinkan pengembangan lebih lanjut yang mudah, seperti menambahkan observer lain tanpa harus mengubah class WeatherStation.


KONSTRUKSI PE...
.qodo
KPL_ZULFA_MUSTAF...
01_Pengantar_Praktikum...
02_Pengenalan_IDE_dan_...
03_HTML
04_OOP
05_Generics
06_DBC
07_Parsing
08_Runtime_Configuratio...
09_API Design dan Const...
10_Library
11
12
13
14
Modul14_231110...
observer.js
U
TP
.gitignore
.hintrc
KPL_ZULFA_MUSTAFA_AKHYAR_ISWAHYUDI_2311104010_SE0701 > 14 > Modul14_2311104010 > JS ob
// Subject class: WeatherStation
class WeatherStation {
constructor() {
this.observers = []; // Menyimpan daftar observer
this.temperature = 0; // Menyimpan suhu saat ini
}
// Menambahkan observer
attach(observer) {
this.observers.push(observer);
}
// Menghapus observer
detach(observer) {
this.observers = this.observers.filter((obs) => obs !== observer);
}
// Memberi tahu semua observer
notify() {
this.observers.forEach((observer) => {
observer.update(this.temperature);
});
}
// Mengubah suhu dan memicu notifikasi
setTemperature(temp) {
console.log(` Suhu berubah menjadi ${temp}°C`);
this.temperature = temp;
this.notify();
}
}
// Observer class: TemperatureDisplay
class TemperatureDisplay {
// Method yang dipanggil saat subject berubah
update(temp) {
console.log(` [Display] Suhu saat ini: ${temp}°C`);
}
}
// Observer class: TemperatureLogger
class TemperatureLogger {
update(temp) {
console.log(` [Logger] Mencatat suhu: ${temp}°C`);
}
}
// Simulasi penggunaan di main method
const weatherStation = new WeatherStation();
const display = new TemperatureDisplay();
const logger = new TemperatureLogger();
// Mendaftarkan observer ke subject
weatherStation.attach(display);
weatherStation.attach(logger);
// Mengubah suhu
weatherStation.setTemperature(26);
weatherStation.setTemperature(31);



TP

Pada tugas ini, saya menggunakan proyek kalkulator sederhana berbasis HTML dan JavaScript sebagai bahan refactoring. Kalkulator ini dirancang untuk menerima input angka, menampilkan ekspresi matematika, serta melakukan perhitungan dasar seperti penjumlahan.

Saya pilih Tugas Jurnal minggu ke-4 tentang HTML + JS pada kalkulator sederhana:
Penjelasan refactor:

- **Naming convention**:
    - Variabel dan fungsi memakai camelCase, konstan memakai PascalCase (tidak ada di sini).
    - Fungsi diberi nama deskriptif seperti displayNumber, addOperator, calculateResult, resetCalculator.
- **Indentasi dan whitespace**:
    - Konsisten menggunakan 2 spasi untuk indentasi (bisa juga 4 spasi kalau kamu mau, yang penting konsisten).
    - Baris kosong yang cukup untuk memisahkan fungsi dan blok kode agar mudah dibaca.
- **Komentar**:
    - Tambahkan komentar fungsi dan penjelasan variabel global supaya mudah dipahami.
    - Hindari komentar berlebihan yang hanya mengulang kode.
- **Deklarasi variabel**:
    - Deklarasi di awal, dengan konstanta (const) dan variabel (let) sudah tepat.
- **Perbaikan kecil**:
    - Cek agar tidak bisa memasukkan operator dua kali berturut-turut (misal 1++2) dengan fungsi isOperatorLastChar().
    - Hindari error dengan cek kondisi sebelum evaluasi.

Contoh refactoring file gui kalkulator yagn sudah di clean-code:

```
<script>
  /**
   * Kalkulator Sederhana
   * Variabel global untuk menyimpan ekspresi dan status input baru
   */
  let currentExpression = "";
  let isNewInput = true;

  // Elemen output
  const outputElement = document.getElementById("output");

  /**
   * Menampilkan angka atau tanda desimal ke output
   * @param {string} number - Angka atau tanda desimal yang akan ditampilkan
   */
  function displayNumber(number) {
    if (isNewInput) {
      currentExpression = number;
      isNewInput = false;
    } else {
      currentExpression += number;
    }
    outputElement.textContent = currentExpression;
  }

  /**
   * Menambahkan operator ke ekspresi jika ekspresi tidak kosong
   * @param {string} operator - Operator matematika (+, -, *, /)
   */
  function addOperator(operator) {
    if (currentExpression !== "" && !isOperatorLastChar()) {
      currentExpression += operator;
      outputElement.textContent = currentExpression;
      isNewInput = false;
    }
  }

  /**
   * Mengecek apakah karakter terakhir pada ekspresi adalah operator
   * @returns {boolean} true jika karakter terakhir adalah operator
   */
  function isOperatorLastChar() {
    return /[+\-*/]$/.test(currentExpression);
  }

  /**
   * Menghitung hasil dari ekspresi matematika yang ada
   */
  function calculateResult() {
    if (currentExpression === "" || isOperatorLastChar()) return;

    try {
      // Ganti koma menjadi titik (untuk desimal)
      const expressionForEval = currentExpression.replace(/,/g, ".");

      // Evaluasi ekspresi
      const result = eval(expressionForEval);

      // Format hasil agar tidak dalam notasi ilmiah dan hapus trailing zero
      const formattedResult = Number.isInteger(result)
        ? result.toString()
        : result.toFixed(8).replace(/\.?0+$/, "");

      outputElement.textContent = formattedResult;
      currentExpression = formattedResult;
      isNewInput = true;
    } catch {
      outputElement.textContent = "Error";
      resetCalculator();
    }
  }
```

```
  /**
   * Reset kalkulator ke kondisi awal
   */
  function resetCalculator() {
    currentExpression = "";
    outputElement.textContent = "0";
    isNewInput = true;
  }

  /**
   * Menangani klik tombol angka dan desimal
   */
  function handleNumberButtonClick(event) {
    const buttonText = event.target.textContent;

    if (buttonText === ".") {
      // Cek apakah sudah ada titik desimal pada angka terakhir
      const lastNumber = currentExpression.split(/[+\-*/]/).pop();
      if (!lastNumber.includes(".")) {
        displayNumber(".");
      }
    } else {
      displayNumber(buttonText);
    }
  }

  // Inisialisasi event listener

  // Tombol angka dan desimal (kecuali operator, equals, dan clear)
  document.querySelectorAll(".button:not(.operator):not(.equals):not(.clear)").forEach(button => {
    button.addEventListener("click", handleNumberButtonClick);
  });

  // Tombol operator
  document.getElementById("btnAdd").addEventListener("click", () => addOperator("+"));

  // Tombol sama dengan (=)
  document.getElementById("btnEquals").addEventListener("click", calculateResult);

  // Tombol AC (reset)
  document.getElementById("btnAC").addEventListener("click", resetCalculator);
</script>
```