LAPORAN PRAKTIKUM
KONSTRUKSI PERANGKAT LUNAK

MODUL V

GENERICS

Disusun Oleh :

Ade Fatkhul Anam

S1SE-06-02

Asisten Praktikum :

Muhamad Taufiq Hidayat

Dosen Pengampu :

Riyan Dwi Yulian Prakoso, S.Kom., M.Kom.

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

DIREKTORAT TELKOM KAMPUS PURWOKERTO

2025

BAB I
PENDAHULUAN

A. DASAR TEORI

5.1 Generics

Generics adalah teknik dalam pemrograman yang memungkinkan pembuatan kode yang lebih fleksibel, reusable, dan efisien dengan mendukung berbagai tipe data. Dalam JavaScript, meskipun tidak memiliki dukungan generics seperti TypeScript atau C#, kita masih dapat mengimplementasikannya dengan pendekatan berbasis class dan function.

Konsep ini sangat berguna dalam pengembangan perangkat lunak karena memungkinkan penggunaan kembali kode tanpa harus menuliskan ulang fungsi atau class untuk setiap tipe data yang berbeda. Dengan generics, kita dapat menghindari duplikasi kode, meningkatkan skalabilitas aplikasi, dan membuat struktur kode yang lebih mudah dipahami serta dikelola

5.2 Generic Class

Generic Class memungkinkan kita membuat struktur data yang dapat menyimpan berbagai tipe elemen dalam satu class yang sama. Ini sangat berguna dalam pengembangan aplikasi yang membutuhkan fleksibilitas tanpa harus membuat banyak class khusus untuk tiap tipe data yang berbeda.

Keuntungan utama dari penggunaan Generic Class adalah efisiensi dalam pengelolaan data. Dengan class generik, kita tidak perlu membuat class terpisah untuk setiap tipe data yang berbeda. Misalnya, kita bisa memiliki satu class yang bisa menangani data bertipe angka maupun string secara bersamaan.

```
class GenericList {
    constructor() {
        this.items = [];
    }

    add(item) {
        this.items.push(item);
    }

    getAll() {
        return this.items;
    }
}

const list = new GenericList();
list.add(1);
list.add("Hello");
console.log(list.getAll());
```

5.3 Generic Function

Generic Function memungkinkan kita menulis fungsi yang dapat bekerja dengan berbagai jenis tipe data tanpa harus menduplikasi kode.

Dalam banyak kasus, fungsi generik dapat digunakan untuk meningkatkan fleksibilitas kode, misalnya dalam operasi pertukaran nilai atau pemrosesan koleksi data yang berbeda jenis.

```
function swap(a, b) {
    return [b, a];
}

let x = 5, y = 10;
[x, y] = swap(x, y);
console.log(x, y); // Output: 10, 5
```

5.4 Generic Delegate

Generic Delegate adalah konsep dalam generics yang memungkinkan kita meneruskan fungsi sebagai parameter, sehingga meningkatkan modularitas dan fleksibilitas kode. Delegate sering digunakan dalam pengembangan perangkat lunak untuk mendukung desain berbasis event-driven, seperti callback function dalam JavaScript.

```
function genericDelegate(callback, value) {
    callback(value);
}

genericDelegate(console.log, "Event Triggered");
```

5.5 Implementasi dalam Aplikasi Nyata

Penerapan generics dalam JavaScript sangat bermanfaat untuk pengembangan aplikasi berbasis data yang kompleks. Contohnya dalam pembuatan sistem manajemen inventaris, sistem pemrosesan transaksi, atau framework yang mendukung berbagai tipe data.

Misalnya, dalam pengelolaan data pengguna, kita bisa menggunakan Generic Class untuk menyimpan dan memproses informasi pengguna tanpa harus menulis ulang kode untuk setiap jenis data yang berbeda.

5.6 Kesimpulan

Penerapan generics dalam JavaScript membantu dalam membangun kode yang lebihefisien, reusable, dan mudah dipelihara. Dengan menggunakan class dan function generik, kita dapat menghindari pengulangan kode yang tidak perlu serta memastikan aplikasi lebih scalable dan mudah dikembangkan. Meskipun JavaScript secara native tidak mendukung generics secara eksplisit seperti TypeScript, kita tetap dapat menerapkan prinsip ini dengan pendekatan berbasis pola desain yang tepat.

Dengan memahami dan mengimplementasikan konsep generics dalam JavaScript, pengembang dapat meningkatkan kualitas kode dan membuat solusi perangkat lunak yang lebih robust dan efisien dalam lingkungan Node.js.

BAB II
TUGAS PENDAHULUAN

Link Github:

github.com/falnam/KPL_Ade_Fatkhul_Anam_2211104051_SE02.git

Hasil Program:

1. haloGeneric.js

05_Generic > JS HaloGeneric.js > ...

```
Codeium: Refactor | Explain
class HaloGeneric {
    Codeium: Refactor | Explain | Generate JSDoc | ×
    SapaUser(user) {
      console.log(`Halo user ${user}`);
    }
}


Codeium: Refactor | Explain | Generate JSDoc | ×
function main() {
    const halo = new HaloGeneric();
    const nama = "Ade Fatkhul Anam";
    halo.SapaUser(nama);

    console.log("=== Code Execution Successful ===");
}

main();
```

Output:

```
PS D:\KPL\KPL_Ade_Fatkhul_Anam_2211104051_SE02\05_generic> node HaloGeneric.js
Halo user Ade Fatkhul Anam
=== Code Execution Successful ===
PS D:\KPL\KPL_Ade_Fatkhul_Anam_2211104051_SE02\05_generic>
```

Penjelasan Kode:

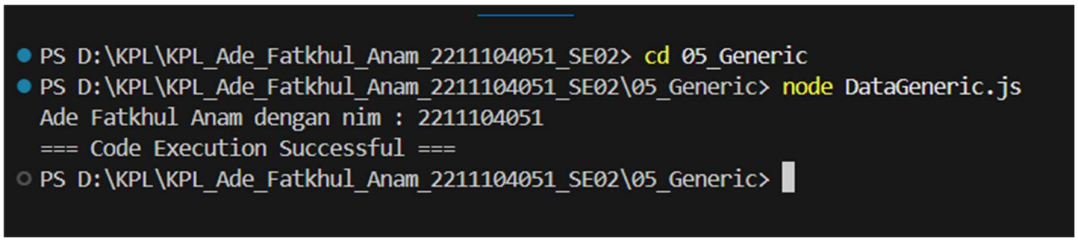
Class HaloGeneric dalam JavaScript adalah class sederhana yang berfungsi untuk menampilkan pesan sapaan kepada user. Di dalamnya terdapat method SapaUser(user) yang menerima sebuah parameter dan mencetak kalimat "Halo user <nama>". Pada fungsi utama main(), dibuat sebuah objek dari class tersebut, lalu didefinisikan variabel nama yang kemudian digunakan sebagai argumen saat memanggil method SapaUser(). Hasil dari program ini adalah menampilkan teks "Halo user Ade Fatkhul Anam" ke konsol.

2. DataGeneric.js

05_Generic > JS DataGeneric.js > main

```
Codeium: Refactor | Explain
class DataGeneric {
    Codeium: Refactor | Explain | Generate JSDoc | ×
    constructor(data) {
      this.data = data;
    }

    Codeium: Refactor | Explain | Generate JSDoc | ×
    PrintData() {
      console.log(`${this.data}`);
    }
}


Codeium: Refactor | Explain | Generate JSDoc | ×
function main() {
    const nama = "Ade Fatkhul Anam";
    const nim = "2211104051";
    const data = new DataGeneric(`${nama} dengan nim : ${nim}`);

    data.PrintData();

    console.log("=== Code Execution Successful ===");
}

main();
```

Output:



Penjelasan Kode:

Class DataGeneric dalam JavaScript berfungsi untuk menyimpan serta menampilkan data yang diberikan saat objek diinstansiasi. Class ini memiliki konstruktor yang menerima argumen data dan menyimpannya ke dalam properti this.data. Method PrintData() digunakan untuk menampilkan isi dari this.data. Pada fungsi main(), dibuat sebuah instance dari DataGeneric dengan data berupa gabungan nama dan NIM menggunakan template literal, kemudian method PrintData() dipanggil untuk mencetak hasilnya ke konsol.