LAPORAN PRAKTIKUM
KONSTRUKSI PERANGKAT LUNAK

MODUL V

GENERICS

Disusun Oleh :

Satria Ariq Adelard

Dompas/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 pemrograman yang memungkinkan pembuatan kode yang lebih fleksibel, dapat digunakan kembali dan efisien dengan dukungan dari berbagai jenis data. JavaScript tidak memiliki dukungan umum untuk TypeScript atau C#, tetapi dapat diimplementasikan menggunakan pendekatan berbasis kelas dan fungsi.

Konsep ini sangat berguna untuk pengembangan perangkat lunak karena memungkinkan untuk menggunakan kembali kode tanpa fungsi penulisan ulang atau kelas untuk setiap tipe data yang berbeda. Generik memungkinkan Anda untuk menghindari duplikasi kode, meningkatkan skala aplikasi, membuat struktur kode yang mudah dipahami, dan mengelola

.

5.2 Generic Class

Kelas tipikal dapat membuat struktur data yang dapat menyimpan elemen dari berbagai jenis di kelas yang sama. Ini sangat berguna untuk mengembangkan aplikasi yang membutuhkan fleksibilitas tanpa membuat banyak kelas khusus untuk setiap tipe data yang berbeda.

Keuntungan utama menggunakan

kelas umum adalah efisiensi manajemen data. Di kelas umum, tidak perlu membuat kelas terpisah untuk setiap tipe data yang berbeda. Misalnya, Anda dapat memiliki kelas yang secara bersamaan dapat memproses tipe atau string data

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

Menggunakan obat generik dalam JavaScript sangat berguna untuk mengembangkan aplikasi database yang kompleks. Misalnya, membuat sistem manajemen inventaris, sistem pemrosesan transaksi, atau kerangka kerja yang mendukung berbagai jenis data.

Saat mengelola data pengguna, Anda dapat menggunakan kelas umum untuk menyimpan dan memproses informasi pengguna..

5.6 Kesimpulan

Menggunakan obat generik di JavaScript membantu Anda membangun kode yang lebih efisien dan lebih dapat digunakan kembali. Dengan menggunakan kelas dan fitur umum, Anda dapat menghindari pengulangan kode yang tidak perlu dan membuat pengembangan aplikasi lebih diskalakan dan sederhana. Penduduk asli JavaScript tidak mendukung generik eksplisit seperti TypeScript, tetapi Anda dapat menggunakan prinsip ini dengan pendekatan berbasis desain yang baik.

Dengan memahami dan mengimplementasikan konsep generik JavaScript, pengembang dapat meningkatkan kualitas kode mereka dan menciptakan solusi perangkat lunak yang lebih kuat dan efisien di lingkungan yang diikat. React JS.

BAB II
TUGAS PENDAHULUAN

Hasil Program:

1. haloGeneric.js

```
class HaloGeneric {
    SapaUser(user) {
      console.log(`Halo user ${user}`);
    }
  }


  function main() {
    const halo = new HaloGeneric();
    const nama = "SatriaAriq";
    halo.SapaUser(nama);

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

  main();
```

Output:

```
PS D:\PraktikumKPL\KPL_Satria_Ariq_Adelard_Dompas_2211104033_SE062\05_Generic> node HaloGeneric.js
Halo user SatriaAriq
=== Code Execution Successful ===
PS D:\PraktikumKPL\KPL_Satria_Ariq_Adelard_Dompas_2211104033_SE062\05_Generic>
```

Penjelasan Kode:

Class **HaloGeneric** dalam JavaScript merupakan class sederhana yang dirancang untuk memberikan sapaan kepada pengguna. Di dalam class ini terdapat method bernama **SapaUser(user)** yang menerima satu parameter, lalu mencetak pesan "Halo user \<nama\>". Pada fungsi utama **main()**, dibuat sebuah instance dari class tersebut. Selanjutnya, sebuah variabel bernama **nama** dideklarasikan dan nilainya digunakan sebagai argumen saat memanggil method **SapaUser()**. Ketika program dijalankan, akan muncul output di konsol berupa tulisan: **"Halo user SatriaAriq"**..

2. dataGeneric.js

```
class DataGeneric {
    constructor(data) {
      this.data = data;
    }

    PrintData() {
      console.log(`${this.data}`);
    }
  }


  function main() {
    const nama = "Satria Ariq Adelard Dompas";
    const nim = "2211104033";
    const data = new DataGeneric(`${nama} dengan nim : ${nim}`);

    data.PrintData();

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

  main();
```

Output:

```
PS D:\PraktikumKPL\KPL_Satria_Ariq_Adelard_Dompas_2211104033_SE062\05_Generic> node DataGeneric.js
Satria Ariq Adelard Dompas dengan nim : 2211104033
=== Code Execution Successful ===
PS D:\PraktikumKPL\KPL_Satria_Ariq_Adelard_Dompas_2211104033_SE062\05_Generic>
```

Penjelasan Kode:

Class DataGeneric dalam JavaScript digunakan untuk menyimpan data yang diberikan saat pembuatan objek, serta menampilkannya. Di dalam class ini terdapat konstruktor yang menerima parameter data dan menyimpannya dalam properti this.data. Untuk menampilkan data tersebut, tersedia method PrintData(). Dalam fungsi main(), dibuat objek dari class DataGeneric, di mana data yang dimasukkan merupakan gabungan nama dan NIM menggunakan template literal. Setelah itu, method PrintData() dipanggil untuk menampilkan data ke konsol.