

**LAPORAN PRAKTIKUM
KONSTRUKSI PERANGKAT LUNAK**

**MODUL V
GENERICS**



Disusun Oleh:

Dewi Atika Muthi / 2211104042

SE-06-02

Asisten Praktikum:

Muhamad Taufiq Hidayat

Dosen Pengampu:

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

PROGRAM STUDI S1 SOFTWARE ENGINEERING

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

BAB I

PENDAHULUAN

A. DASAR TEORI

Generics adalah konsep pemrograman yang **memungkinkan penulisan kode** dalam bentuk yang dapat digunakan **dengan berbagai tipe data**, tanpa perlu mendefinisikan ulang kode tersebut untuk setiap tipe data yang berbeda. Generics hadir dalam berbagai bahasa pemrograman seperti C#, Java, TypeScript, dan beberapa implementasi konsep serupa dalam JavaScript.

Generics memungkinkan programmer untuk mendesain class, interface, method, dan struktur data lainnya yang bekerja dengan berbagai tipe data. Pada bahasa dengan static typing seperti C#, saat deklarasi, tipe data spesifik akan disediakan sebagai parameter sehingga kode menjadi type-safe. Sementara di JavaScript, yang merupakan bahasa dengan dynamic typing, konsep generics diimplementasikan melalui fleksibilitas tipe data bawaan.

Sebagai contoh, penggunaan struktur data yang dapat menampung berbagai tipe:

```
const mixedArray = [1, "hello", true, {name: "John"}];
```

Keuntungan penggunaan generics dan konsep serupa adalah :

1. **Fleksibilitas** - Kode dapat bekerja dengan berbagai tipe data.
2. **Reusability** - Kode dapat digunakan kembali untuk berbagai tipe data.
3. **Readability** - Membuat kode lebih mudah dibaca dan dipahami.
4. **DRY (Don't Repeat Yourself)** - Mengurangi duplikasi kode untuk operasi yang sama pada tipe data berbeda.

JavaScript, sebagai bahasa dengan dynamic typing, tidak memiliki generics dalam sintaksis seperti bahasa static typing. Namun, JavaScript secara inheren memungkinkan fleksibilitas tipe data yang mirip dengan konsep generics:

1. **Arrays & Collections** - JavaScript arrays dapat menyimpan berbagai tipe data secara alami.
2. **Functions** - Fungsi JavaScript dapat menerima dan mengembalikan berbagai tipe data.
3. **Callbacks & Higher Order Functions** - Fungsi dapat diteruskan sebagai parameter, mirip dengan delegate generics.

Meskipun JavaScript tidak memiliki generics eksplisit, penting untuk memahami implementasinya di bahasa lain untuk apresiasi konsep:

1. **C#** - Menggunakan sintaksis `<T>` dengan constraint yang ketat.
2. **Java** - Serupa dengan C#, dengan beberapa perbedaan dalam implementasi.
3. **TypeScript** - Menambahkan generics ke JavaScript untuk type checking saat kompilasi.

B. MAKSUD DAN TUJUAN

Tujuan dari praktikum ini adalah:

1. Memahami konsep generics dalam pemrograman dan implementasi konsep serupa di JavaScript.
2. Mempelajari cara membuat struktur data dan fungsi yang fleksibel yang dapat bekerja dengan berbagai tipe data.
3. Mengimplementasikan pattern yang mirip dengan generics dalam JavaScript.
4. Mengaplikasikan pengetahuan tentang fleksibilitas tipe dalam pengembangan kode yang lebih reusable.
5. Memahami perbedaan implementasi generics di bahasa static typing dan pendekatan yang dilakukan di bahasa dynamic typing seperti JavaScript.

BAB II IMPLEMENTASI

A. PRAKTIKUM (GUIDED)

1. Implementasi Generic Class

Soal Studi Case

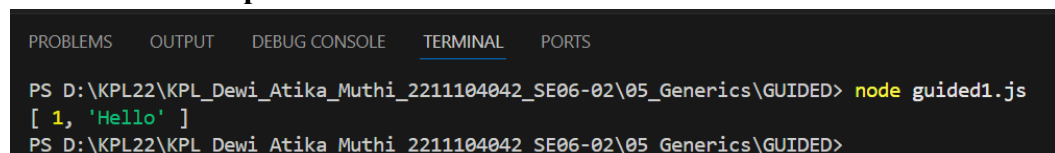
Dalam JavaScript, kita dapat membuat class yang bekerja dengan berbagai tipe data, mirip dengan konsep generic class di bahasa static typing:

Sourcecode

```
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());
```

Screenshot Output



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\KPL22\KPL_Dewi_Atika_Muthi_2211104042_SE06-02\05_Generics\GUIDED> node guided1.js
[ 1, 'Hello' ]
PS D:\KPL22\KPL_Dewi_Atika_Muthi_2211104042_SE06-02\05_Generics\GUIDED>
```

Deskripsi Program

- Program di atas mendefinisikan kelas `GenericList` yang dapat menyimpan berbagai tipe data dalam satu koleksi.
- JavaScript secara inherent mendukung konsep ini karena sifat dynamic typing-nya.
- Kelas ini memiliki metode `add()` untuk menambahkan item dan `getAll()` untuk mengambil semua item.
- Dalam contoh penggunaan, kita menambahkan integer `1` dan string `"Hello"` ke dalam list yang sama.
- Output program akan menampilkan array `[1, "Hello"]`, menunjukkan bahwa list dapat menyimpan tipe data yang berbeda.

Ini mendemonstrasikan fleksibilitas koleksi JavaScript yang dapat menyimpan berbagai tipe data tanpa definisi eksplisit seperti pada generic class bahasa static typing.

2. Implementasi Generic Function

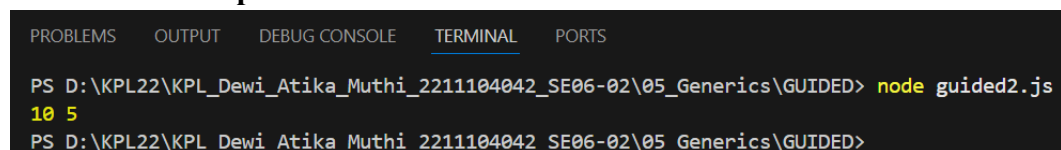
Soal Studi Case

JavaScript memungkinkan pembuatan fungsi yang dapat menerima dan mengembalikan berbagai tipe data:

Sourcecode

```
function swap(a, b) {  
    return [b, a];  
}  
  
let x = 5, y = 10;  
[x, y] = swap(x, y);  
console.log(x, y);
```

Screenshoot Output



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  
PS D:\KPL22\KPL_Dewi_Atika_Muthi_2211104042_SE06-02\05_Generics\GUIDED> node guided2.js  
10 5  
PS D:\KPL22\KPL_Dewi_Atika_Muthi_2211104042_SE06-02\05_Generics\GUIDED>
```

Deskripsi Program

- Program di atas mendefinisikan fungsi `swap` yang menukar dua nilai.
- Fungsi ini dapat menerima nilai dari berbagai tipe data (integer, string, objek, dll.) tanpa perlu mendefinisikan tipe secara eksplisit.
- Nilai-nilai yang diberikan dikembalikan dalam urutan terbalik sebagai array.
- Dalam contoh penggunaan, kita menukar nilai integer `x` dan `y` menggunakan destructuring assignment.
- Output menunjukkan bahwa nilai telah ditukar dengan sukses, dari `x=5, y=10` menjadi `x=10, y=5`.

3. Implementasi Function Delegate

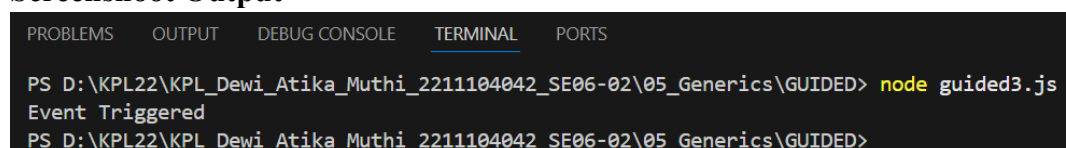
Soal Studi Case

JavaScript memungkinkan fungsi untuk diteruskan sebagai parameter, mirip dengan konsep delegate dalam bahasa seperti C#:

Sourcecode

```
function genericDelegate(callback, value) {  
    callback(value);  
}  
  
genericDelegate(console.log, "Event Triggered");
```

Screenshoot Output



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  
PS D:\KPL22\KPL_Dewi_Atika_Muthi_2211104042_SE06-02\05_Generics\GUIDED> node guided3.js  
Event Triggered  
PS D:\KPL22\KPL_Dewi_Atika_Muthi_2211104042_SE06-02\05_Generics\GUIDED>
```

Deskripsi Program

- Program di atas mendefinisikan fungsi `genericDelegate` yang menerima dua parameter: `callback` (fungsi yang akan dipanggil) dan `value` (nilai yang akan diteruskan ke fungsi `callback`).
- Fungsi ini menjalankan `callback` dengan nilai yang diberikan.
- Dalam contoh penggunaan, kita meneruskan `console.log` sebagai `callback` dan string `"Event Triggered"` sebagai nilai.
- Output akan menampilkan string `"Event Triggered"` di konsol.

B. TUGAS PENDAHULUAN (UNGUIDED)

Link TP: https://github.com/tikature/tpmodul5_2211104042

Link Repo Praktikan:

https://github.com/tikature/KPL_Dewi_Atika_Muthi_2211104042_SE06-02

1. Tugas 1 (dalam branch generic-method)

Soal Studi Case

Membuat class HaloGeneric yang memiliki method SapaUser dengan parameter generic. Method tersebut mencetak “Halo user X”, di mana X adalah input (nama panggilan praktikan), lalu method dipanggil di dalam fungsi utama dengan input berupa String.

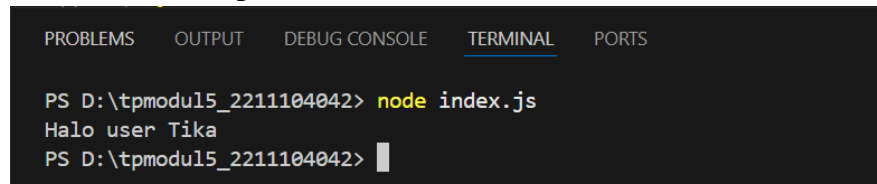
Sourcecode

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

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

main();
```

Screenshot Output



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\tpmodul5_2211104042> node index.js
Halo user Tika
PS D:\tpmodul5_2211104042> |
```

Deskripsi Program

- Class HaloGeneric memiliki method SapaUser yang menerima parameter user (string).
- Method ini mencetak string "Halo user X" di mana X adalah nama yang diinput.
- Dalam implementasi kita, nama yang digunakan adalah "Tika".

Method SapaUser pada class HaloGeneric dapat menerima parameter dengan tipe data apapun, demonstrasi dari flexible type parameter.

2. Tugas 2 (dalam branch generic-class)

Soal Studi Case

Membuat class `DataGeneric` dengan property data bertipe generic T.

Class memiliki konstruktor untuk mengisi data dan method `PrintData()` untuk mencetak "Data yang tersimpan adalah: Y". Method dipanggil di fungsi utama dengan mengisi data dengan NIM praktikan.

Sourcecode

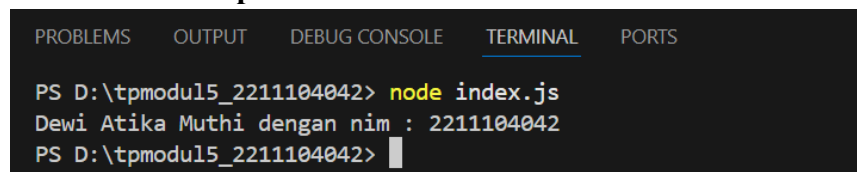
```
class DataGeneric {
  constructor(data) {
    this.data = data;
  }
  PrintData() {
    console.log(`${this.data}`);
  }
}

function main() {
  const nama = "Dewi Atika Muthi";
  const nim = "2211104042"; // Ganti dengan NIM praktikan
  const data = new DataGeneric(`${nama} dengan nim : ${nim}`);

  data.PrintData();
}

main();
```

Screenshoot Output



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\tpmodul5_2211104042> node index.js
Dewi Atika Muthi dengan nim : 2211104042
PS D:\tpmodul5_2211104042>
```

Deskripsi Program

- Membuat class `DataGeneric` dengan property data bertipe generic T.
- Class memiliki konstruktor untuk mengisi data dan method `PrintData()` untuk mencetak "Data yang tersimpan adalah: Y".
- Method dipanggil di fungsi utama dengan mengisi data dengan NIM praktikan.

Class `DataGeneric` dapat menyimpan data dengan tipe apapun dalam property data, menunjukkan konsep generic storage.

BAB III

KESIMPULAN DAN SARAN

A. KESIMPULAN

Penerapan generics dalam JavaScript membantu dalam membangun kode yang lebih efisien, 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.

B. REFERENSI

Modul 5 Generics, Konstruksi Perangkat Lunak.

MDN Web Docs. "JavaScript Guide". <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide>

Simpson, K. (2019). You Don't Know JS Yet: Get Started. O'Reilly Media.

Flanagan, D. (2020). JavaScript: The Definitive Guide. O'Reilly Media.

TypeScript Documentation. "Generics".

<https://www.typescriptlang.org/docs/handbook/2/generics.html>

Haverbeke, M. (2018). Eloquent JavaScript: A Modern Introduction to Programming. No Starch Press.