**LAPORAN PRAKTIKUM TEKNIK PEMROGRAMAN MINGGU 5**

Disusun Oleh :

|  |  |
| --- | --- |
| Radhitya Maulana Arrafi | 241524025 |
| 1A D4 Teknik Informatika |

Daftar Isi

[**1** **LIST, SET AND MAP** 3](#_Toc192962499)

[**1.1** **Definisi** 3](#_Toc192962500)

[**1.2** **Keperluan** 3](#_Toc192962501)

[**1.3** **Penerapan di dunia nyata** 4](#_Toc192962502)

[**2** **RECORD** 5](#_Toc192962503)

[**2.1** **Definisi** 5](#_Toc192962504)

[**2.2** **Keperluan** 6](#_Toc192962505)

[**2.3** **Penerapan di dunia nyata** 6](#_Toc192962506)

[**3** **OPTIONAL** 6](#_Toc192962507)

[**3.1** **Definisi** 6](#_Toc192962508)

[**3.2** **Keperluan** 7](#_Toc192962509)

[**3.3** **Penerapan di dunia nyata** 7](#_Toc192962510)

[**4** **CONCURRENT COLLECTION** 8](#_Toc192962511)

[**4.1** **Definisi** 8](#_Toc192962512)

[**4.2** **Keperluan** 8](#_Toc192962513)

[**4.3** **Penerapan di dunia nyata** 8](#_Toc192962514)

[**5** **QUEUE AND DEQUEUE** 9](#_Toc192962515)

[**5.1** **Definisi** 9](#_Toc192962516)

[**5.2** **Keperluan** 9](#_Toc192962517)

[**5.3** **Penerapan di dunia nyata** 9](#_Toc192962518)

[**6** **IMMUTABLE COLLECTION** 10](#_Toc192962519)

[**6.1** **Definisi** 10](#_Toc192962520)

[**6.2** **Keperluan** 10](#_Toc192962521)

[**6.3** **Penerapan di dunia nyata** 10](#_Toc192962522)

# **LIST, SET AND MAP**

## **Definisi**

* **List** :

Sebuah koleksi yang berurutan yang memungkinkan kita untuk menyimpan nilai yang sama. **List** juga menyimpan urutan masuk dan menggunakan indeks untuk mengakses elemennya sehingga mirip dengan **array**. Perbedaannya terletak dimana **List** memungkinkan kita untuk menggunakan operasi dinamis dan pengelolaan yang lebih baik sehingga menjadikan **List** opsi yang lebih baik dibandingkan dengan **array**.

* **Set** :

**Set** merupakan sebuah koleksi tidak berurutan yang tidak mengizinkan nilai yang sama. Hal ini memodelkan abstraksi himpunan matematika. Jika terdapat elemen yang sama ketika memasukkan nilai, maka elemen yang lama akan digantikan dengan nilai yang baru. Penggunaan **Set** dapat dipertimbangkan bila kita ingin mempertahankan keunikan dari data yang digunakan untuk dicari.

* **Map** :

**Map** merupakan sebuah objek yang memetakan **Key** dengan sebuah nilai, sehingga memungkinkan kita untuk mencari berdasarkan **Key**. **Map** juga dapat berupa koleksi yang menyimpan data lebih dari satu. **Key** yang digunakan di dalam **Map** dapat disesuaikan dengan kebutuhan misalkan **String** yang memungkinkan kita untuk menggabungkan huruf dan angka.

## **Keperluan**

* **List** :
  + Digunakan ketika data perlu disimpan dalam **urutan tertentu**.
  + Memungkinkan penyimpanan **nilai yang duplikat**.
  + Cocok untuk **pengolahan data sekuensial**, seperti antrian tugas atau daftar elemen yang sering diakses berdasarkan indeks.
* **Set** :
  + Digunakan saat kita ingin menyimpan elemen unik, tanpa ada duplikasi.
  + Tidak menggunakan indeks seperti List, sehingga elemen disimpan berdasarkan aturan internal masing-masing implementasi.
  + Cocok untuk mengecek keberadaan data tanpa perlu melakukan pencarian manual.
* **Map** :
  + Digunakan ketika kita ingin menyimpan **pasangan kunci-nilai** (**key-value pair**), di mana setiap kunci harus **unik**.
  + Berbeda dari List dan Set, Map tidak hanya menyimpan elemen, tetapi juga menghubungkan **setiap kunci dengan satu nilai tertentu**.
  + Cocok untuk **mendapatkan data secara cepat berdasarkan kunci**, tanpa harus melakukan pencarian satu per satu.

## **Penerapan di dunia nyata**

* **List** :

A computer screen shot of code

AI-generated content may be incorrect.Menyimpan data **nilai siswa**

Hasil keluaran program :

A screen shot of a computer

AI-generated content may be incorrect.

* **Set** :

Menyimpan **daftar Tag** pada artikel

A computer screen shot of code

AI-generated content may be incorrect.

Hasil keluaran program :

A black background with white text

AI-generated content may be incorrect.

* **Map** :

A computer screen shot of a program code

AI-generated content may be incorrect.Menyimpan **Hasil Voting Kandidat**

Hasil keluaran program :

A screenshot of a computer screen

AI-generated content may be incorrect.

# **RECORD**

## **Definisi**

**Record** adalah fitur baru java yang dikenalkan pada java 14 dan resmi dirilis pada java 16. **Record** digunakan untuk membuat kelas baru yag bersifat *immutable* (tidak bisa diubah / **final**) dengan cara yang lebih ringkas dan mudah untuk dibaca. Berikut adalah contoh pendeklarasian sebuah **Record** :



## **Keperluan**

Dalam Java Collection Framework, **Record** digunakan untuk menyimpan data secara **immutable** dengan sintaks yang ringkas, otomatis menyediakan equals(), hashCode(), dan toString(). Dalam **List**, record memungkinkan penyimpanan elemen berurutan tanpa modifikasi. Dalam **Set**, record memastikan **keunikan data** tanpa perlu override manual. Dalam **Map**, record dapat digunakan sebagai **kunci atau nilai**, memudahkan pencarian data. Dengan sifatnya yang aman dan efisien, record mengurangi kompleksitas kode dan meningkatkan performa dalam pengolahan koleksi data.

## **Penerapan di dunia nyata**

Membuat sebuah record Student dengan isi nim, nama, dan major untuk setiap class Student yang ingin dibuat

A computer screen shot of code

AI-generated content may be incorrect.

Hasil keluaran program :

A computer screen with numbers and symbols

AI-generated content may be incorrect.

# **OPTIONAL**

## **Definisi**

**Optional** adalah kelas pembungkus yang diperkenalkan di **Java 8** dalam package java.util, digunakan untuk **menghindari null secara eksplisit** dan mengurangi risiko **NullPointerException (NPE)**. Optional sering digunakan dalam koleksi seperti List, Set, dan Map untuk menangani data yang mungkin **tidak ada (absen)** atau bernilai **null** dengan cara yang lebih aman dan terstruktur. Penggunaan **Optional** membantu dalam **menulis kode yang lebih bersih, lebih aman, dan lebih mudah dibaca** dengan menghindari pengecekan null secara manual.

## **Keperluan**

**Optional** digunakan untuk menangani nilai yang mungkin null dengan cara yang lebih aman dan ekspresif. Dengan **Optional**, kita dapat menghindari **NullPointerException** dan membuat kode lebih mudah dibaca karena tidak perlu melakukan pengecekan **null** secara manual. Fitur ini juga mendorong praktik pemrograman yang lebih baik dengan memaksa developer untuk secara eksplisit menangani kasus di mana nilai mungkin tidak ada. Selain itu, **Optional** mendukung gaya pemrograman fungsional dengan menyediakan metode seperti **map(), filter(),** dan **orElse(),** sehingga memudahkan pengelolaan data yang mungkin kosong. Secara keseluruhan, **Optional** membantu meningkatkan keamanan, keterbacaan, dan kualitas kode.

## **Penerapan di dunia nyata**

A computer screen shot of code

AI-generated content may be incorrect.Membuat sebuah program untuk **validasi** setelah user menginput nama

Hasil keluaran program :

A black background with white text

AI-generated content may be incorrect.A black background with white text

AI-generated content may be incorrect. Kasus 1 : Kasus 2 :

# **CONCURRENT COLLECTION**

## **Definisi**

**Concurrent Collections** adalah versi **thread-safe** dari koleksi standar, dirancang untuk menangani akses simultan dalam lingkungan multi-threading tanpa perlu sinkronisasi manual. Contohnya termasuk **ConcurrentHashMap**, **CopyOnWriteArrayList**, dan **ConcurrentLinkedQueue**.

## **Keperluan**

Digunakan dalam **aplikasi multi-threading** seperti server web atau sistem pemrosesan data paralel, di mana beberapa **thread** harus membaca dan menulis data koleksi secara **bersamaan** tanpa menyebabkan konflik atau inkonsistensi.

## **Penerapan di dunia nyata**

Contoh sederhana penggunaan thread untuk mengakses data :

A computer screen shot of a program code

AI-generated content may be incorrect.

Hasil keluaran program :

A black background with white text

AI-generated content may be incorrect.

# **QUEUE AND DEQUEUE**

## **Definisi**

**Queue** adalah struktur data **FIFO (First-In-First-Out)**, sedangkan **Deque** adalah versi dua arah yang mendukung penambahan dan penghapusan dari kedua ujung. Contoh implementasi meliputi **LinkedList**, **PriorityQueue**, dan **ArrayDeque**.

## **Keperluan**

Digunakan untuk manajemen antrian tugas, seperti sistem antrean pelanggan, jadwal eksekusi tugas, **atau pemrosesan pesan dalam sistem antrian (message queue).** **PriorityQueue** berguna untuk pemrosesan berbasis prioritas, sementara **Deque** cocok untuk struktur seperti stack dan double-ended queue.

## **Penerapan di dunia nyata**

A screenshot of a computer program

AI-generated content may be incorrect.Membuat sebuah proses pelayanan pelanggan dalam sebuah restaurant.

A screenshot of a computer screen

AI-generated content may be incorrect. Hasil keluaran program :

# **IMMUTABLE COLLECTION**

## **Definisi**

**Immutable Collection** adalah koleksi yang **tidak dapat diubah** setelah dibuat. Java menyediakan metode seperti **List.of(), Set.of(),** dan **Map.of()** untuk membuat immutable collection dengan lebih mudah dibandingkan menggunakan **Collections.unmodifiableList()**.

## **Keperluan**

Digunakan ketika data harus tetap **konstan** dan tidak boleh diubah setelah inisialisasi, seperti **konfigurasi aplikasi**, **daftar status tetap**, atau **data default dalam suatu sistem**. Immutable collection meningkatkan **keamanan dan performa** dengan mencegah modifikasi yang tidak diinginkan.

## **Penerapan di dunia nyata**

A computer screen shot of a program code

AI-generated content may be incorrect.Membuat sebuah aplikasi yang perlu menyimpan konfigurasi default, seperti pengaturan tema, bahasa, dan ukuran font. Konfigurasi ini tidak boleh diubah oleh pengguna atau kode lain selama runtime

Hasil keluaran program :

A screenshot of a computer screen

AI-generated content may be incorrect.