



Collections

Tim Ajar Algoritma dan Struktur Data
Genap 2024/2025

Tujuan

- Memahami jenis-jenis Collection dan contoh penggunaannya
- Dapat mengimplementasikan Collection pada library Java untuk menyelesaikan studi kasus yang sesuai

Outline

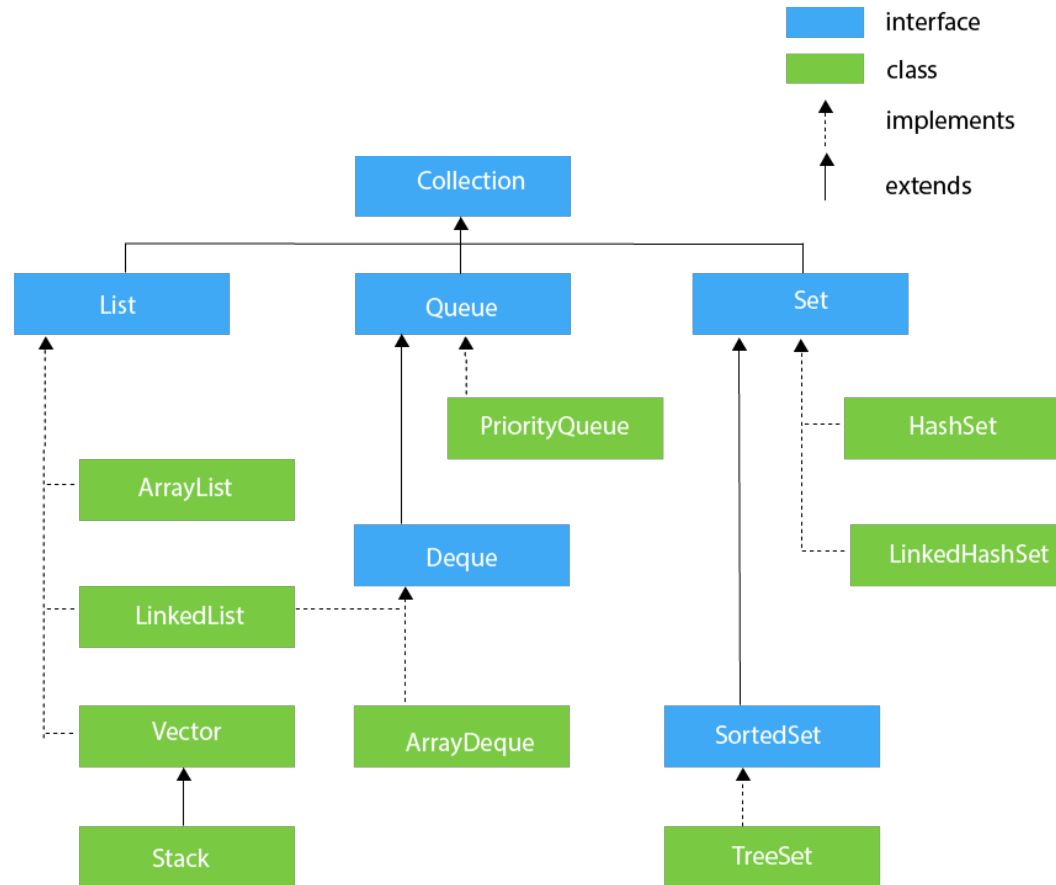
- Java Collection Framework
- Sorting dan Searching
- Subinterface Collection

Pendahuluan

- Struktur data merupakan format tertentu untuk mengatur, memproses, mengambil, dan menyimpan data.
- Terdapat struktur data linear (Stack, Queue, LinkedList) dan nonlinear (Tree, Graph)
- Struktur data linear merupakan struktur data yang terurut secara serial. Sedangkan struktur data non linear berarti data tersebut disajikan dalam bentuk yang tidak serial (hierarki ataupun tidak teratur)

Java Collection Framework

- Java platform memiliki collection framework untuk merepresentasikan dan memanipulasi collection
- Collection adalah suatu object yang terdiri dari sekumpulan objek-objek
- Java collection framework menyediakan sekumpulan interface dan class yang digunakan untuk mengimplementasikan suatu struktur data
- Manfaat:
 - ❖ Tidak perlu membuat struktur data dan algoritma sendiri
 - ❖ Struktur data dan algoritma sudah efisien



<https://www.javatpoint.com/collections-in-java>

Interface

- Class merupakan template/blueprint untuk membuat objek
- Interface menentukan/memberi panduan apa saja yang harus dapat dilakukan oleh sebuah class
- Object tidak dapat diinstansiasi dari interface
- Yang dapat dilakukan adalah instansiasi object dari class yang mengimplement interface

Method Interface Collection

- add() – menambahkan elemen
- addAll() – menambahkan sekumpulan elemen
- contains() – return true jika elemen ada di collection
- clear() – menghapus semua elemen
- isEmpty() – return true jika collection kosong
- remove() – menghapus elemen
- removeAll() – menghapus semua elemen dari collection yang match dengan collection tertentu
- size() – mengembalikan ukuran collection
- toArray() – mengembalikan array yang berisi semua elemen pada collection

Sorting dan Searching Java Collection Framework

- Java Collection Framework menyediakan operasi standard seperti pengurutan dan pencarian data.
- Kedua fungsi tersebut terdapat di dalam package `java.util.Collections`.
- Untuk mengurutkan data menggunakan static function `sort()` → `Collections.sort()`.
- Fungsi `sort()` pada `Collections` menerapkan algoritma merge sort.
- Untuk mencari data bisa menggunakan `Collections.binarySearch()`

Subinterface Collection

Interface Collection memiliki beberapa subinterface yang dapat diimplementasikan oleh berbagai macam class di Java

- Interface List: collection dengan index-based method untuk insert, update, delete, dan search element. Dapat menyimpan duplicate element dan null
- Interface Set: collection tanpa duplicate element
- Interface Queue: collection dengan konsep FIFO

Interface List

- Interface List: collection dengan index-based method untuk insert, update, delete, dan search element. Dapat menyimpan duplicate element dan null
- Tidak dapat diinstansiasi
- Class yang mengimplement interface List: ArrayList, LinkedList, Vector, dan Stack.
- Untuk menggunakan List: `import java.util.List;`

Method Interface List

- `add(int index, E element)` – menambahkan elemen pada index tertentu
- `get(int index)` – mengembalikane elemen pada index tertentu
- `indexOf(E element)` – mengembalikan index dari elemen pertama yang dicari
- `remove(int index)` – menghapus elemen pada index tertentu
- `set(int index, E element)` – replace elemen pada index tertentu
- `subList(int fromIndex, int toIndex)` – mengembalikan sublist dengan range index tertentu
- `sort()` – mengurutkan elemen

Class ArrayList

ArrayList merupakan sebuah class dari Java Collection Framework yang menyediakan implementasi struktur data serupa dengan array tetapi dengan ukuran dinamis

Contoh Method Class ArrayList

- *Semua method Collection*
- *Semua method List*
- clone() – mengembalikan arraylist baru dengan elemen, ukuran, dan kapasitas yang sama
- trimToSize() – memangkas kapasitas arrayList sesuai jumlah elemen

Cara Menggunakan ArrayList

- Pertama yang harus dilakukan adalah import `java.util.ArrayList`
- Sintaks:

```
ArrayList<Type> arrayList= new ArrayList<>();
```

- Contoh:

```
ArrayList<Integer> arrayList1 = new ArrayList<>();
```

```
ArrayList<String> arrayList2 = new ArrayList<>();
```

- `arrayList1` hanya dapat menyimpan data bertipe integer
- `arrayList2` hanya dapat menyimpan data bertipe string

Class Stack

- Stack merupakan sebuah class dari Java Collection Framework yang menyediakan implementasi struktur data tumpukan
- Pada sebuah stack, elemen dapat ditambahkan atau diakses menggunakan konsep Last In First Out (LIFO)
- Elemen ditambahkan pada top of stack dan diambil dari top of stack.

Method Class Stack

- `push()` – menambahkan elemen pada stack
- `pop()` – mengeluarkan elemen pada stack
- `peek()` – mengembalikan elemen yang terdapat pada top stack
- `search()` – mengembalikan posisi elemen pada sebuah stack
- `empty()` – mengecek apakah sebuah stack kosong

Cara Menggunakan Stack

- Pertama yang harus dilakukan adalah import `java.util.Stack`
- Sintaks:

```
Stack<Type> stacks = new Stack<>();
```

- Contoh:

```
Stack<Integer> numbers = new Stack<>();
```

```
Stack<Student> students = new Stack<>();
```

- `numbers` hanya dapat menyimpan data tipe integer
- `students` hanya dapat menyimpan objek bertipe `Student`

Interface Queue

- Queue merupakan sebuah interface dari Java Collection Framework yang menyediakan implementasi struktur data antrian dengan konsep FIFO
- Queue adalah sebuah interface sehingga tidak bisa langsung diimplementasikan
- Instansiasi objek hanya dapat dilakukan dari class yang mengimplements Queue, yaitu LinkedList, PriorityQueue, dan ArrayDeque

Interface Queue

- Pertama yang harus dilakukan adalah import `java.util.Queue`
- Sintaks:

```
LinkedList<Type> namaQueue = new LinkedList<>();  
ArrayDeque<Type> namaQueue = new ArrayDeque<>();  
PriorityQueue<Type> namaQueue = new PriorityQueue<>();
```

- Contoh:

```
PriorityQueue<Integer> numbers = new PriorityQueue<>();  
LinkedList<Customer> customers = new LinkedList<>();
```

Method Queue

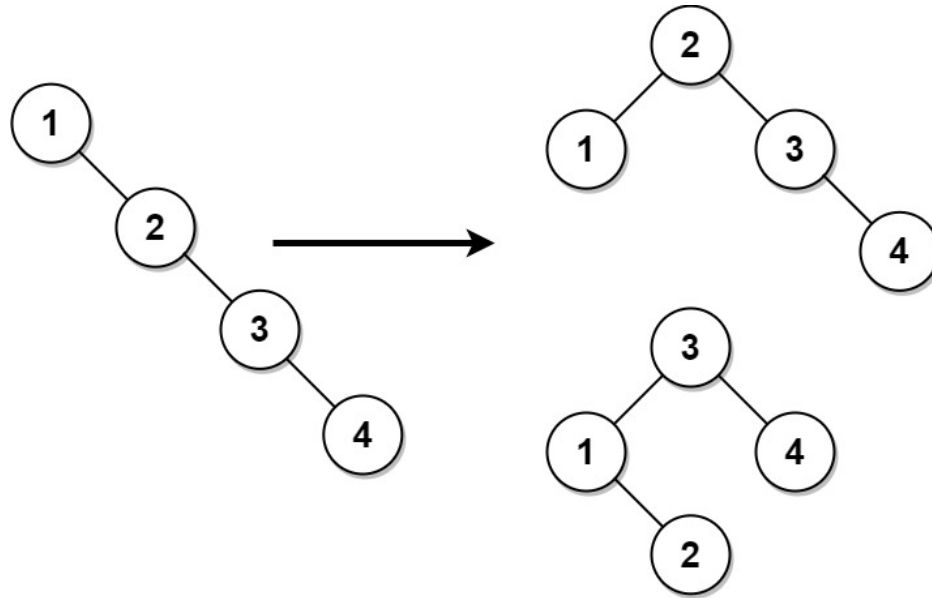
- add() – menambahkan elemen
- peek() – mengembalikan elemen head, null if empty
- poll() - mengembalikan dan menghapus head, null if empty

ArrayDeque & PriorityQueue

- PriorityQueue merupakan class yang mengimplement interface Queue. Struktur data ini memberikan prioritas dengan meletakkan elemen terkecil di sisi depan, walaupun elemen tersebut tidak ditambahkan pertama kali
- ArrayDeque merupakan sebuah class yang mengimplement interface Queue dan Deque. Pada struktur data ini, elemen dapat ditambahkan dan dihapus dari kedua sisi queue

Class TreeSet

- Sebuah class dari Java collection framework yang menyediakan implementasi struktur data tree
- Tidak mengizinkan duplikasi data
- Tree yang diimplementasikan merupakan self-balancing binary search tree
- Binary search tree adalah binary tree dengan aturan left-child < parent < right-child, sehingga data tidak disimpan sesuai urutan penambahan
- Self-balancing maksudnya tree secara otomatis menjaga bahwa maksimal height adalah $\log n$ setelah insert atau hapus elemen



<https://leetcode.com/problems/balance-a-binary-search-tree/description/>

Contoh Method Class TreeSet

Method yang dimiliki oleh interface Collection dan Set akan diimplementasikan oleh class TreeSet.

- `first()` – mengembalikan elemen terkecil
- `last()` – mengembalikan elemen terbesar
- `pollFirst()` & `pollLast()` – menghapus elemen terkecil dan terbesar

Instansiasi TreeSet

- Pertama yang harus dilakukan adalah import `java.util.TreeSet`
- Sintaks:

```
Treeset<Type> treeset = new TreeSet<>();
```

- Contoh:

```
TreeSet<Integer> numbers = new TreeSet<>();
```

Graph

- Java secara khusus tidak memiliki class yang mengimplementasikan struktur data graph
- Beberapa library yang bisa digunakan untuk mendukung operasi-operasi graph
 - JGraphT
 - JUNG — the Java Universal Network/Graph Framework
 - GraphStream

