Collection

Iterable
Superclass semua collection di java yang bukan turunan dari map
iterable digunakan untuk foreach collection, jadi semua collection yang turunan iterable pasti bisa di foreach

Collection Interface
Digunakan untuk memanipulasi data dari collection nya seperti tambah, hapus, edit
Collection dibagi lagi menjadi list, set, dan queue
Contoh method collection :
- size
- isEmpty
- contains
- toArray
- clear
- remove

List Interface
Struktur data collection yang punya sifat :
- element bisa duplikat
- data berurutan dengan kapan kita masukin
- list pake index default number yang mirip kayak array
List memiliki beberapa method kayak :
- add
- sort
- remove
- get
- indexOf
- set
List dibagi menjadi 2:
- ArrayList, implementasi dari list menggunakan array
- LinkedList, implementasi dari linkedlist class dengan data yang disimpan dalam node dengan tersedianya next dan previous. gada info array nya
List bisa diubah sifatnya menjadi mutable maupun immutable
Mutable : datanya tidak bisa diubah lagi isi elemennya
Immutable : datanya masih bisa diubah-ubah
Ada beberapa beberapa method untuk membuat immutable list
- Collections.emptyList() = list kosong
- Collections.singletonList(e) = immutable 1 element
- Collections.unmodifiableList(list) = konversi mutable ke immutable
- List.of(e..) = immutabli dari element-element

Set Interface
Set merupakan collections yang berisikan element uniq yang berarti tidak duplikat
Set tidak punya index kayak list, jadi gabakal urut
Karena ga punya index, untuk ambil datanya harus di iterate satu persatu
Hashtable, menyimpan hashcode
Set memiliki 3 turunan:
- EnumSet, tapi set ini jarang digunakan
- HashSet, data tidak terurut dengan waktu kapan memasukan data. Kalo ga butuh terurut bisa pake ini
- LinkedHashSet, data terurut dengan waktu kapan memasukan data. Kalo butuh urut pake ini
Dalam set juga bisa diubah menjadi immutable kayak list, bedanya kalo elements pakenya Set.of bukan List.of

SortedSet Interface
Ini kayak set biasa tapi pas dimasukkin nanti otomatis secara otomatis

Kalo elementnya bukan turunan dari comparable maka harus dibuat comparator untuk melakukan sorting nya

NavigableSet Interface
Turunan dari SortedSet
Menambahkan method untuk beberapa operasi seperti :
- lower
- floor
- ceiling
- higher
- poolFirst
- poolLast
Bisa immutable :
- Collections.emptyNavigableSet()
- Collections.unmodifiableNavigableSet(set)

Queue Interface
Collections antrian FIFO (First In First Out)
Ada beberapa method tambahan di queue :
- add, kalo udah mencapai kapasitas bakal throw error
- offer, kalo udah sampe kapasitasnya cuma bakal return false
- remove, mengambil dan menghapus data yang pertama kali dimasukkan, kalo gada data lagi bakal throw error
- poll, sama kayak remove tapi return null
- element, ambil data pertama tapi tidak dihapus, kalo data kosong bakal throw exception
- peek, kayak element tapi return null
Queue mempunyai turunan :
- Dequeue
- ArrayDequeue, menggunakan array sebagai implementasi queue nya, kayak arrayList
- PriorityQueue, menggunakan array tapi diurutkan menggunakan comparabel atau comparator
- LinkedList, menggunakan List dan gabakal ada batasnya

Dequeue Interface
Bisa dari depan maupun belakang, implementasi FIFO dan LIFO (Last In First Out)
Seperti antrian dan tumpukan
Deque bisa menggunakan LinkedList dan ArrayDequeue

Map Interface
Collections yang berisikan mapping key dan value
1 key cuma boleh ke 1 value
Kayak array tapi index nya bebas ga cuma angka
Beberapa method dari map :
- size
- isEmpty
- containsKey
- containsValue
- get, mengambil data berdasarkan key
- put, nyimpen data
- remove
- keySet, ambil semua key
- values, ambil semua value
- entrySet, ambil key dan value
Ada 5 implementasi Map :
- HashMap
impelmentasi dari map dengan menggunakan hashCode function dengan mengecek kesamaan menggunakan equals
- WeakHashMap

sama kayak HashMap tapi menggunakan weak key dimana jika tidak digunakan lagi maka datanya akan dihapus key nya.
Cocok diganakan pada cache
- IdentityHashMap
Kayak HashMap tapi berbeda cara mengecek kesamaan datanya menggunakan method ==
- LinkedHashMap
Menggunakan LinkedList, datanya juga berurutan sesuai dengan kapan dimasukkan
Proses get nya akan semakin lambat karena harus di loop satu persatu
Kalo ambil data pake key lebih baik menggunakan HashMap biasa
- EnumMap
keynya menggunakan ENUM karena ENUM pasti uniq dan bakal lebih baik dari menggunakan Hash

Immutable Map
Map bisa dikonversi menjadi immutable seperti list dan set
Method :
- Collections.emptyMap()
- Collections.unmodifiableMap(map)
- Collections.singletonMap(key, value)
- Map.of

SortedMap Interface
Impelmentasi Map dengan data key nya yang diurutkan menggunakan comparable maupun dibuat comparatornya
SortedMap bisa mneggunakan semua method yang ada di Map
Ada beberapa method tambahan seperti :
- comparator(), memasukkan comparatornya
- subMap(), memotong dari posisi key awal sampai terakhir
- headMap(), ambil dari awal
- tailMap(), ambil dari belakang
Immutable SortedMap :
- emptySortedMap
- unmodifiableSortedMap

NavigableMap
Turunan dari SortedMap
Bisa melakukan operasi kayak NavigableSet tapi isinya key dan value

Enrtry
Data dalam map disimpan dalam pair (key-value)
Entry ini interface sederhana untuk mengambil key dan value

LegacyCollections
Terdiri dari :
- Vector Class yang mirip sekali dengan ArrayList tapi methodnya menggunakan kata kunci synchronized yang thread safe
- HashTable Class yang mirip dengan HashMap
- Stack Class yang mirip dequeue yang LIFO

Sorting
Berbagai algoritma sorting sudah disediakan oleh java. sorting ini hanya bisa digunakan pada List karena pada collection lainnya sudah disediakan
Method :
- sort(list)
- sort(list, comparator)

BinarySerach
Algoritma pencarian yang lebih cepat dari search bawaan java di list yang menggunakan sequential serach yang mencari satu persatu

Penggunaan binary search ini harus menggunakan list yang sudah di sorting
Menggunakan method :
- Collectons.binarySearch(list, value)
- Collectons.binarySearch(list, value, comparator)

Collections Class
Utility static method collection untuk manipulasi data collection
Contoh :
- copy(listTo, listFrom)
- frequency(collection, object)
- reverse(list)
- suffle(list)
- swap(list, from, to)

Abstract Collection
Semua algoritma dasar dari collection sudah ada abstract class nya karena algoritma tidak akan berubah
Jika ingin membuat collection sendiri bisa implements abstract nya bukan interface nya
Kumpulan abstract class :
- AbstractCollection : Collection
- AbstractList : List
- AbstractMap : Map
- AbstractQueue : Queue
- AbstractSet : Set

Default Method
Berada pada interface baik dari List maupun Map, dengan penggunaan default method ini dapat mempersingkat penulisan kode jika sudah menggunakan lambda

Spliterator Interface
Interface untuk melakukan partisi data yang ada di collection dalam jumlah besar
Nantinya akan di split dan diproses secara paralel nantinya pada multithread

Konversi Array
Untuk melakukan konversi ke array menggunakan toArray()
Method :
- Object[] toArray()
- T[] toArray(new T[])

Lambda

Merupakan anonymous function, tapi di java itu anonymous class atau versi sederhana dari anonymous class
Syarat lambda :
- Berupa interface
- memiliki 1 method abstract
- menambahkan annotation @FunctionalInterface

Cara penulisan lambda
- TipeData namaVariable = (TipeData parameter) -> {return;};
- TipeData namaVariable = (parameter) -> {return;};
- TipeData namaVariable = parameter -> balikan;
Menggunakan method reference
- TipeData<TipeDataParameter, TipeDataKembalian> namaVariable = TipeData::methodTipeData;

Java Util FUnction
Package yang berisikan functional interface untuk membuat lambda

Dengan interface ini tidak perlu dibuat lagi manual dan sebagian besar menggunakan generic
Contoh :
- Consumer, method void accept(T t) yang biasa digunakan untuk perulangan forEach dan berfungsi unutk menerima data
Penggunaan :
Consumer<T> namaVariable = parameter -> doSomething;
namaVariable.accept(parameter);
- Function, method R apply(T t) dibuthkan jika butuh nilai kembalian dan parameter
- Predicate, method boolean test(T t) dengan return true or false
- Supplier, method T get(), digubakan untuk mendapatkan nilai

Method Reference
Saat hanya mengakses method yang ada pada parameternya dapat dipersingkat menggunakan method references

Lambda di Collection
Default method yang digunakan pada collection kebanyakan menggunakan parameter di java.util.function sehingga prameter tersebut dapat diubah jadi lambda
List<String> names = List.of("Ahamd", "Solikhin", "Gayuh", "Raharjo");
names.forEach(name -> System.out.println(name));
Diubah menjadi
names.forEach(System.out::println);

Lambda Sebagai Lazy Parameter
Sebenarnya java tidak memiliki fitur lazy parameter
Lazy parameter sendiri adalah parameter tersebut hanya akan dieksekusi jika diakses
Dengan memanfaatkan lambda java dapat parameter layaknya lazy parameter
Hal ini akan meningkatkan performa jika parameter itu tidak digunakan atau tidak dipanggil
Lazy Parameter di java bisa mrenggunakan Supplier<T> namaVariable
dan untuk mengakses nya tinggal menggunakan namaVariable.get()

Lambda di Optional Class
Class optional digunakan sebagai wrapper pada value yang dapat null maupun object yang null
Hal ini dikarenakan java dapat mengthrow NullPointerException

Apache Maven

Template maven yang biasa digunakan

cheat sheet :
- mvn archetype:generate
- maven-archetype-quickstart

Maven lifecycle :
- clean, menghapus target folder
- compile, mengcompile source code
- test, menjalankan unit test
- tets-compile, mengcompile unit test
- install, menginstall ke lokal repo
- deploy, mendeploy code ke remote server
- package, mempackage ke jar maupun war

Menjalankan jar : java -jar target/namaFileJar

Dependency
Merupakan pulgin maupun tools dari luar project
Maven mendukung dependency management dan akan mendownloadnya secara otomatis
Terdapat scope depedency, seperti :
- compile, digunakan pada source code dan unit test
- test, hanya dapat digunkan di test

Referensi mencari dependency di java:
- search.maven.org
- mvnrepository.com

Repository merupakan lokasi dimana jar itu tersedia
Jika ingin menambahkan repository baru dan menggunakannya bisa ditambahkan ditambah dengan url lokasi url nya

Maven Properties
Mendukung properties untuk menyimpan data konfigurasi ke dalam tag properties
lalu diakses dengan cara ${namaTag}

Distribusi File
Secara default membuat distribusi file secara default menggunakan package
Tapi depedencies dan properties yang digunakan tidak terinclude
Cara agar jar dimasukkan saat distrubisi file bisa menggunakan Assembly Plugin
Cara penggunaan : mvn package assembly:single

Multi Model Project
Saat aplikasi sangat besar ada baiknya dibuat aplikasi modular
Misal dipecah menjadi modul modelm controllerm view, service, dsb
Untuk membuat modul baru hanya cukup tambah folder dan tambahkan seting di pom xml
Modul harus memiliki parent yang merupakan project diatas modul tersebut
Selanjutnya di parent nya pun harus menginclude modulnya
Cara mengakses modul yang berbeda dengan menambahkan di dependency

Dpendecy Management
Jika telah menggunakan banyak dependency yang banyak namun berbeda modul akan menimbulkan masalah
Di maven mendukung dependecy management dimana bisa memasukannya di parent dan menambahkannya tanpa harus menggunakan versinya
Secara otomatis versi dependency akan sama dengan yang ada di parent modul
Dilakukan dengan cara memasukkan tag dependencies dalam dependencyManagement