VERSI 2.2

SEPTEMBER, 2020

[STRUKTUR DATA]

MODUL 3, HASHMAP

TIM PENYUSUN: - DOSEN
- DICKY PRABOWO OCTIANTO

PRESENTED BY: LAB. TEKNIK INFORMATIKA

UNIVERSITAS MUHAMMADIYAH MALANG

[STRUKTUR DATA]

CAPAIAN PEMBELAJARAN MATA KULIAH

Mahasiswa mampu menguasai & menjelaskan konsep dari struktur data Hashmap

SUB CAPAIAN PEMBELAJARAN MATA KULIAH

Mahasiswa mampu memahami dan menerapkan Hashmap

PERSYARATAN PEMAHAMAN

1. Array List
2. Linked List

KEBUTUHAN HARDWARE & SOFTWARE

- Java Development Kit
- Java Runtime Environment
- IDE (Intellij IDEA, Eclipse, Netbeans, dll.)

REFERENSI MATERI

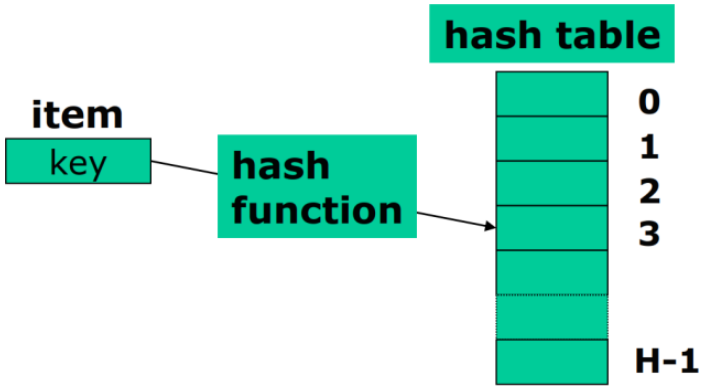
Oracle iLearning Java Programming section 6-3 Collections, sub section Hashmap

MATERI POKOK

Hash:

Hashing digunakan sebagai metode untuk menyimpan data dalam sebuah array agar penyimpanan data, pencarian data, penambahan data, dan penghapusan data dapat dilakukan dengan cepat. Fungsi Hash mentransformasi sebuah item kedalam index sesuai dengan ukuran array yang telah dibuat. h(k) -> {0,1,2,…,m-1} dimana m adalah ukuran array. Sifat-sifat yang diharapkan dari h(k) adalah mudah dalam komputasi dan menghasilkan kunci distribusi yang uniform sepanjang (0,1,2,…,m1), diharapkan h(k1) ≠ h(k2) apabila k1 ≠ k2 (tidak terjadi collision).

Dalam Hashing, key berukuran besar akan dikonversi menjadi lebih kecil menggunakan Hash Functions. Key index yang tekah dikonversi kemudian disimpan di suatu struktur data yang disebut Hash Table. Dasar dari Hashing adalah mendistribusikan record secara seragam ke seluruh array dengan melakukan komputasi pada index untuk memberikan informasi di mana record yang dimaksudkan dapat ditemukan atau disisipkan.



Gambar 1. Ilustrasi Hashing

Hashing selalu merupakan fungsi satu arah. Fungsi Hash yang ideal tidak bisa diperoleh dengan melakukan reverse engineering dengan menganalisa nilai Hash. Hash Function ideal memiliki kompleksitas waktu (T(n)) = O(1), untuk mencapainya setiap record membutuhkan key index yang unik, di mana kompleksitas waktu tersebut tidak ditemukan pada struktur data model lain. Ada beberapa macam Hash Function yang relatif sederhana yang dapat digunakan diantaranya (1) Modulo Division, (2) Midsquare, (3) Digit Summation, (4) Folding, (5) Truncation dan (6) Multiplication. Hash Function bukan merupakan fungsi one-to-one, artinya beberapa record yang berbeda dapat menghasilkan nilai Hash yang sama yang mengakibatkan collision. Dengan Hash Function yang baik, hal seperti ini akan sangat jarang terjadi, tapi pasti akan terjadi. Collision berarti ada lebih dari satu record yang memiliki nilai Hash atau key index yang sama. Ada dua strategi untuk mengatasi Collision diantaranya adalah Open Addressing dan Chaining.

1. Open Adressing

Pada Open Addressing, record baru disimpan di dalam Hash Table, yang akan bertambah terus menerus. Jika suatu record dimasukkan ke dalam Hash Table pada lokasi sesuai nilai Hash-nya dan ternyata lokasi tersebut sudah diisi dengan record lain maka harus dicari lokasi alternatif yang masih belum terisi.

Misalnya record tambahan dengan nilai “26, James Gray, DB & trans processing “. Hash Function memberikan nilai 3, yang ternyata telah ditempati record lain sebelumnya. Penelusuran mendapatkan lokasi kosong pada index 6 sehingga data ini ditempatkan pada index 6. Record tambahan lainnya dengan nilai ”54, Manuel Blum, Computational complexity“ dengan
Hash Function 54 % 3 = 8, yang ternyata telah ditempati record lain sebelumnya. Penelusuran selanjutnya mendapat lokasi kosong pada index 9.

Bila penelusuran telah mencapai posisi terakhir maka pindah ke posisi pertama

	SEMULA				MENJADI		
Index	Kode	Nama			Kode	Nama	
[0]	46	John McCarthy	...		46	John McCarthy	...
[1]							
[2]	25	Donald E. Knuth	...		25	Donald E. Knuth	...
[3]	**49**	**CAR Hoare**	...		49	CAR Hoare	...
[4]	**50**	**Raj Reddy**	...		50	Raj Reddy	...
[5]	**5**	**John Hopcroft**	...		5	John Hopcroft	...
[6]					**26**	**James Gray**	...
[7]	30	Dennis M. Ritchie	...		30	Dennis M. Ritchie	...
[8]	**8**	**Marvin Minsky**	...		8	Marvin Minsky	...
[9]					**54**	**Manuel Blum**	...
[10]	33	Niklaus Wirth	...		33	Niklaus Wirth	...
[11]							
[12]	35	E.W. Dijkstra	...		35	E.W. Dijkstra	...

Gambar 2. Penanganan collision pada Hash Table menggunakan Linear Probing

2. Chaining

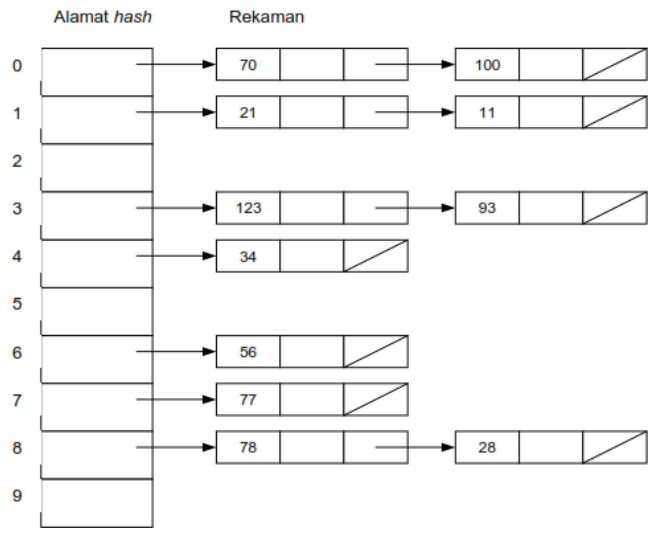
Pada metode chaining, Hash Table bukan lagi menjadi array of records, tetapi menjadi array of pointers. Setiap pointer menunjuk ke LinkedList berisikan record yang menghasilkan nilai Hash yang sama.

Penambahan record dapat dilakukan dengan menambah Node LinkedList berisi record baru. Untuk langkah pencarian record pada Hash Table, pertama-tama dicari nilai Hash terlebih dahulu, kemudian dilakukan pencarian dalam LinkedList yang bersangkutan. Untuk menghapus record, hanya menghapus record-nya saja, tidak menghapus satu LinkedList penuh.

Kelebihan dari metode chaining ini adalah proses penghapusan yang relarif mudah dan penambahan ukuran Hash Table bisa ditunda untuk waktu yang lebih lama meskipun seluruh lokasi pada Hash Table sudah penuh. Bahkan, penambahan ukuran Hash table bisa saja tidak perlu dilakukan sama sekali.

Struktur data lain dapat digunakan sebagai pengganti LinkedList. Misalnya dengan tree, kompleksitas waktu terburuk bisa diturunkan menjadi O(log n) dari yang sebelumnya O(n). Namun demikian, struktur data tree kurang efisien kecuali Hash Table memang didesain untuk jumlah record yang banyak atau kemungkinan terjadi collision sangat besar yang mungkin terjadi.

Sebagai contoh, record bernilai 34, 56, 123, 78, 93, 70, 100, 21, 11, 77, 28 dan Hash Function yang dipilih adalah k mod 10. Dengan demikian, alamat Hash akan terdiri dari 10 buah alamat yang bernomor 0 sampai 9.



Gambar 3. Penanganan collision pada Hash Table menggunakan Chaining

Hashmap:

Hashmap adalah sebuah struktur data table yang mengimplementasikan hash dari Map Interface di java. Implementasi ini menyediakan semua operasi optional dari map. Dan mengijinkan null data bagi key dan values. Kelas Hashmap mirip seperti Hashtable kecuali dalam Hashmap tidak tersinkronasi dan mengjinkan nilai null. Kelas Hashmap tidak menjamin sebuah data yang terurut dalam map; khususnya, itu tidak menjamin bahwa pesanan akan tetap konstan seiring waktu. Implementasi ini memberikan kinerja waktu-konstan untuk operasi dasar (get danset), dengan asumsi fungsi hash menyebarkan elemen-elemen dengan baik di antara bucket. Iterasi atas koleksi membutuhkan waktu instance HashMap sebanding dengan "kapasitas" (jumlah pemetaan nilai-kunci). Dengan demikian, sangat penting untuk tidak menetapkan kapasitas awal terlalu tinggi jika kinerja iterasi penting.

Hashmap dapat dinisialisasi menggunakan cara HashMap<KeyType,ValueType> mapName = new HashMap<KeyType,ValueType>();

Contoh inisialisasi sebuah Hashmap:

```
HashMap<String,String> mangkokBuah = new HashMap<String,String>();
```

Kenapa memakai Hashmap?:

Singkatnya, permasalahan dilatar belakangi oleh algoritma untuk mendapatkan value yang dimiliki key dalam sebuah tabel besar. Jika proses pencarian dilakukan menggunakan perulangan untuk mendapatkan value tesebut akan memakan waktu yang sangat lama untuk menemukan value yang cocok. Sehingga munculah Hashmap sebuah struktur data menggunakan perhitungan hash dari sebuah key untuk menyimpan value.

Daftar method yang dapat dipanggil dalam HashMap:

Method	Deskripsi
boolean containsKey(Object Key)	Mengembalikan true jika sudah terdapat key yang sama dalam sebuah HashMap
Boolean containsValue(Object Value)	Mengembalikan true jika dalam map terdapat value yang sama dengan parameter value yang sudah dimasukkan
Set<K> keySet()	Mengembalikan kumpulan key yang ada dalam HashMap
Collection<V> values()	Mengembalikan sebuah koleksi dari nilai yang ada dalam HashMap
V remove(Object key)	Menghapus nilai yang ada dalam map jika terdapat key yang sama dengan parameter
Int size()	Mengembalikan jumlah key-value yang terdapat dalam sebuah HashMap

MATERI PRAKTIKUM

LATIHAN 1.1 HASHMAP NAMA FAKULTAS YANG ADA DI UMM

Biasakan untuk mengetik secara manual tidak menggunakan copy paste sehingga mengerti alur proses dari kode.

```
package Modul3;

import java.util.HashMap;

public class LatihanHashMap {
    public static void main(String[] args){
        //inisialisasi data
        String[][] fakultasUMM = new String[][]{
                {"FT","FK","FIKES","FEB","FAI","FPP"},
                {"Fakultas Teknik","Fakultas Kedokteran","Fakultas Ilmu
Kesehatan","Fakultas Ekonomi dan Bisnis","Fakultas Agama Islam","Fakultas
Peternakan dan Pertanian"}};
        HashMap<String, String> listFakultas = new HashMap<>();
        int barisFakultas = 0;
        //jika panjang kolom pertama sama dengan kolom kedua, pengecekan agar
tidak terjadi null pointer
        if(fakultasUMM[0].length==fakultasUMM[1].length){
```

```
            //looping data dimulai dari baris pertama hingga selesai untuk
dimasukkan dalam HashMap
            while (barisFakultas<fakultasUMM[0].length){

listFakultas.put(fakultasUMM[0][barisFakultas],fakultasUMM[1][barisFakultas]);
                barisFakultas++;
            }
        }else{
            System.out.println("Inisialisasi data panjang antar kolom tidak
sama");
        }
        //
        System.out.print("List fakultas:");
        //mendapatkan semua value tanpa memerlukan key
        System.out.println(listFakultas.values());
        //uji coba dapatkan data dari HashMap jika key adalah "FT"
        System.out.print("Kepanjangan dari FT adalah : "+ listFakultas.get("FT")
);
        //uji coba dapatkan data dengan key yang tidak diketahui misal "FTI"
        System.out.print("Kepanjangan dari FTI adalah : "+
listFakultas.get("FTI") );
    }
}
```

LEMBAR KERJA

Buat class bernama **Auth** kemudian tambahkan properti dari object **tabelAkun** dan **tabelSesiLogin** sehingga kedua object tersebut nantinya akan ada dalam satu class untuk method selengkapnya ada penjelasan dibawah berikut.

KEGIATAN 1 TABEL DATA AKUN

Buat object HashMap bernama **tabelAkun**. Key, dan Value dari HashMap berupa tipe string, key dapat diisi dengan variable email sedangkan value dapat diisi dengan variable password.
Berikut beberapa method yang harus ada untuk mengelola data dalam object tersebut.

Tipe	Nama Method	Deskripsi
Boolean	registerAkun(String email, String password)	Memasukkan data ke dalam object **tabelAkun** dengan syarat belum ada email yang sama sebelumnya dan **hanya menerima register akun email berdomain @umm.ac.id**. True jika berhasil, false jika gagal memasukkan

Boolean	hapusAkun(String email, String konfirmasiPassword)	Menghapus akun yang terdapat dalam object **tabelAkun** jika berhasil true sebaliknya false
Int	totalEmail()	Total data dengan email yang ada dalam HashMap

Isi minimal 3 data dalam **tabelAkun** tersebut sehingga jika ditampilkan akan terdapat 3 instances (data) dari object tersebut.

Email	Password
labit@umm.ac.id	labitUwoke
laboranlab@umm.ac.id	Lab_oyisam
instrukturlab@umm.ac.id	instrukturkece

Praktikkan semua method diatas dalam driver Class (Main method).

KEGIATAN 2 TABEL SESI LOGIN AKUN

Buat object HashMap bernama **tabelSesiLogin**. Key dan Value berupa tipe string, key dapat diisi dengan variable email sedangkan value dapat diisi dengan variable password. TabelSesi ini sebagai tanda akun berhasil melakukan metode loginAkun() dengan mengambil data dari object **tabelAkun** di kegiatan 1. Berikut beberapa method yang harus ada untuk mengelola data dalam object tersebut.

Tipe	Nama Method	Deskripsi
Boolean	loginAkun(String email, String password)	Cek ke dalam object **tabelAkun** jika terdapat data dengan key dan value yang sama maka return true dan simpan data akun dalam object **tabelSesiLogin** jika tidak false
Boolean	logoutAkun(String email)	Menghapus data akun yang terdapat dalam object **tabelSesiLogin** jika berhasil true sebaliknya false.
Int	totalLogout()	Total data yang sudah melakukan method logoutAkun()
Int	totalLogin()	Total data yang sudah melakukan method loginAkun()
Int	totalAuth()	Total data yang sedang login (data yang berada dalam **tabelSesiLogin)**

Praktikkan semua method diatas dalam driver Class (Main method).

CATATAN

aturan umum penulisan bahasa JAVA agar mudah di koreksi oleh asisten:

1. Untuk nama kelas,interface,enum, dan yang lainnya biasakan menggunakan gaya CamelCase (diawali dengan huruf besar pada tiap kata untuk mengganti spasi) seperti: **K**ursi , **J**alan**R**aya, **P**arkiran**G**edung, dan lain seterusnya.
2. Untuk penulisan nama method, dan attribute diawali dengan huruf kecil di awal katadan menggunakan huruf besar untuk kata setelahnya, seperti: **g**et**N**ama**J**alan, **n**ama**J**alan, **h**arga, **s**et**N**ama**J**alan, dan lain seterusnya.
3. Jika menggunakan IDE Intellij jangan lupa untuk memformat penulisan kode agar terlihat rapi menggunakan menu code -> show reformat file dialog -> centang semua field dan klik ok.

Silahkan dikerjakan tanpa copy – paste

RUBRIK PENILAIAN

Soal	Nilai
Kegiatan 1:	40%
Kegiatan 2:	60%