VERSION 1.2

SEPTEMBER 9, 2021

[STRUKTUR DATA]

MODUL 3, HASHMAP

DISUSUN OLEH: - BELLA DWI MARDIANA
- LIDYA FANKKY OKTAVIA P.

DIAUDIT OLEH: - IR. GITA INDAH MARTHASARI, ST., M.KOM.
- DIDIH RIZKI CHANDRANEGARA, S.KOM., M.KOM.

PRESENTED BY: TIM LAB-IT

UNIVERSITAS MUHAMMADIYAH MALANG

[STRUKTUR DATA]

PERSIAPAN MATERI

Mahasiswa diharapkan mempelajari materi sebelum mengerjakan tugas, materi yang tercakup antara lain:

1. Array List
2. Linked List

TUJUAN

Mahasiswa mampu menguasai & menjelaskan konsep dari Struktur data HashMap

TARGET MODUL

Mahasiswa mampu memahami & menerapkan HashMap

PERSIAPAN SOFTWARE/APLIKASI

1. Java Development Kit
2. Java Runtime Environment
3. IDE (Intellij IDEA, Eclipse, Netbeans, dll)

REFERENSI MATERI

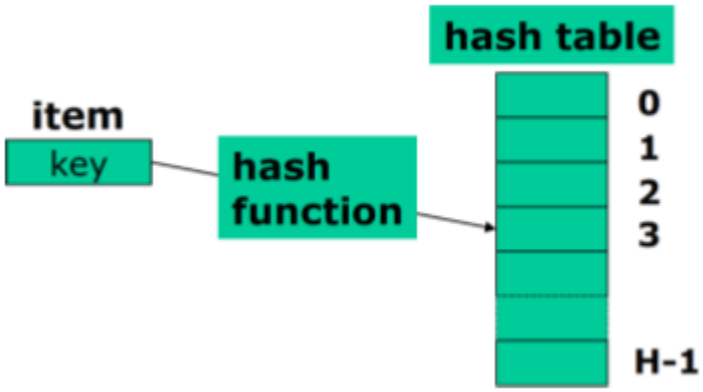
Oracle iLearning Java Programming section 6-3 Collections, sub section HashMap

MATERI POKOK

Pengertian 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 adalah 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 sepanjanag (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 telah dikonversi kemudian disimpan di suatu struktur data yang disebut Hash Table. Dasar dari Hashing adalah mendistribusikan record secara seragam ke seluruh array dengan menggunakan komputasi pada index untuk memberikan informasi dimana record yang dimaksudkan dapat ditemukan atau disisipkan.



Gambar 1. Ilustrasi Hashing

Hashing 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, dimana 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 Addressing**

   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 mendapatkan lokasi kosong pada index 9. Bila penelusuran telah mencapai posisi terakhir, maka pindah ke posisi pertama seperti berikut:

	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 HashTable Menggunakan Linear Probing

2. Chaining

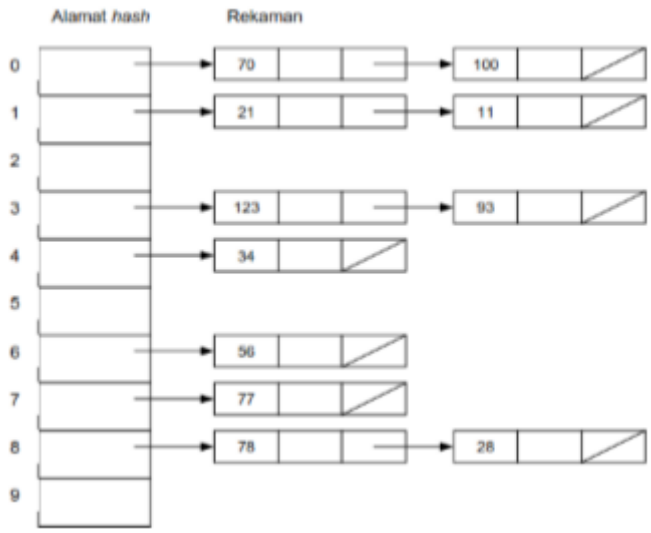
Pada metode Chaining, Hash Table bukan lagi menjadi array of records, tetapi menjadi array of pointers. Setiap pointers menunjuk ke LinkedList berisikan record yang menghasilkan nilai Hash yang sama.

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 recordnya saja, tidak menghapus satu LinkedList penuh.

Kelebihan dari metode Chaining ini adalah proses penghapusan yang relatif mudah dan penambahan ukuran 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.

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 HashTable Menggunakan Chaining

Pengertian 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 data null bagi key dan value. Class HashMap mirip seperti Hashtable kecuali dalam HashMap tidak tersinkronasi dan mengijinkan nilai null. Class HashMap tidak menjamin sebuah data yang terurut dalam map; khususnya, itu tidak menjamin 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 diantara bucket (<>). Iterasi atas koleksi membutuhkan waktu instance HashMap sebanding dengan “kapasitas” (jumlah pemetaan key-value). Dengan demikian, sangat penting untuk tidak menetapkan kapasitas awal terlalu tinggi jika kinerja iterasi penting.

HashMap dapat diinisialisasi menggunakan cara HashMap<Keytype, Valuetype> mapName = new HashMap<Keytype, Valuetype>();

Contoh inisialisasi sebuah HashMap:

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

Kenapa Menggunakan HashMap?

Singkatnya, permasalahan dilatarbelakangi oleh algoritma untuk mendapatkan value yang dimiliki key dalam sebuah tabel besar. Jika proses pencarian dilakukan menggunakan perulangan untuk mendapatkan value tersebut akan memakan waktu yang sangat lama untuk menemukan value yang cocok. Sehingga muncullah Hashmap sebuah struktur data menggunakan Hash dari sebuah key untuk menyimpan value.

Daftar method yang dapat dipanggil dalam HashMap:

Method	Deskripsi
Boolean containsKey(Objectkey)	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

LATIHAN PRAKTIKUM

LATIHAN 1

Berikut adalah contoh pemanfaatan HashMap pada data mahasiswa dengan value lebih dari 1:

```
public class Mahasiswa {
    String nama, kelas, matkulPraktikum;
    int nim;

    public Mahasiswa(String nm, String kl, String mat, int ni){
        nama = nm;
        kelas = kl;
        matkulPraktikum = mat;
        nim = ni;

    }
    public static void main(String[] args){
        Scanner in = new Scanner(System.in);
        HashMap<String, Mahasiswa> mhs = new HashMap<>();
        String input;
        Mahasiswa data;

        mhs.put("1", new Mahasiswa( nm: "Putri", kl: "3H", mat: "Struktur Data", ni: 2020000));
        mhs.put("2", new Mahasiswa( nm: "Agus", kl: "3A", mat: "Matematika", ni: 2020012));
```

```
        mhs.put("3", new Mahasiswa( nm: "Arro",  kl: "3D",  mat: "Pemrograman",  ni: 2020017));

        System.out.println("Masukkan ID: ");
        input = in.nextLine();
        data = mhs.get(input);
        if (data != null){
            System.out.println("Data Mahasiswa : " + data.nama + ", kelas : " +
                    data.kelas + ", mata kuliah praktikum : " + data.matkulPraktikum +
                    ", nim : " + data.nim);
        }
    }
}
```

TUGAS PRAKTIKUM

KEGIATAN 1 (DATA LIST PRAKTIKAN UNTUK PRAKTIKUM)

Buat kelas baru bernama **DataPraktikan** yang memiliki atribut bernama **tabelData** bertipe **HashMap** dan beberapa method (seperti pada tabel). Manfaatkan library HashMap untuk mempermudah pembuatan kelas.

Key dan value dari atribut dataPraktikan dapat berupa string yang akan menyimpan NIM praktikan dan nama asisten.

Tipe	**Nama Method**	**Deskripsi**

Boolean	tambahData(String nimPraktikan, String namaAsisten)	Menambah data baru ke dalam object **dataPraktikan** dengan syarat nimPraktikan belum terdaftar dan hanya menerima inputan nim dengan kombinasi kata **"IF"**.
		
		Berikut contoh output untuk **tambahData**:
		`--Tambah Data Baru--`
		`Masukkan Nim : IF2020123`
		`Masukkan namaAsisten : Putri`

Void	tampil()	Menampilkan semua data **nimPraktikan** beserta **namaAsisten** yang telah berhasil diinputkan.
		
		Contoh output tampilkan semua data yang ada:
		`Total Data Yang Tersimpan : 1`
		`Nim: IF2020123  Nama Asisten : Putri`

Void	listNimPraktikan()	Hanya menampilkan semua data nimPraktikan yang telah berhasil diinputkan.
Void	listNamaAsisten()	Hanya menampilkan semua data namaAsisten yang telah berhasil diinputkan.
Int	totalEmail()	Total data berhasil diinputkan yang ada dalam HashMap.
Boolean	hapusData(String nimPraktikan, String namaAsisten)	Menghapus data yang terdapat dalam object **dataPraktikan** dengan syarat nimPraktikan yang dihapus memang terdaftar sebelumnya.
Void	editData(String nimPraktikan, String namaAsisten)	Melakukan edit data yang terdapat dalam object **dataPraktikan** dengan syarat nimPraktikan yang akan di edit memang terdaftar sebelumnya.

* Optional : tambahkan fitur search untuk menampilkan listNimPraktikan yang memiliki namaAsisten dengan nama sama.

KEGIATAN 2

Setelah Anda berhasil menyelesaikan kegiatan 1, tambahkan 1 (satu) atribut baru pada kelas **DataPraktikan** yang bernama **tabelSesiLogin** yang bertipe **HashMap**. Variabel ini akan menyimpan pasangan email dan password. Kemudian, tambahkan method Login dan Logout pada kelas DataPraktikan dengan spesifikasi sebagai berikut:

Metode Login:

- User diminta memasukkan email dan password
- Sistem memeriksa apakah data email dan password sesuai dengan data yang tersimpan pada atribut tabelSesiLogin
- Jika ada, diperiksa lagi apakah email yang dimasukkan menggunakan domain “@umm.ac.id”
- Jika email dan password benar maka tampilkan daftar fitur/metode seperti pada tabel kegiatan 1
- Jika data tidak sesuai maka tampilkan “Gagal Login”

Metode Logout:

- User keluar dari program

CATATAN

Aturan umum penulisan JAVA agar mudah dikoreksi oleh asisten:

1. Untuk nama class, enum, dan yang lainnya biasakan menggunakan gaya CamelCase (diawali dengan huruf besar pada tiap kata untuk mengganti spasi) seperti: Kursi, JalanRaya, ParkiranGedung, dan lain seterusnya.
2. Untuk penulisan nama method dan attribute diawali dengan huruf kecil di awal kata dan menggunakan huruf besar untuk kata setelahnya, seperti: getNamaJalan, namaJalan, harga, setNamaJalan, dan lain seterusnya.
3. Jika menggunakan IDE Intellij jangan lupa untuk memformat kode agar terlihat rapi menggunakan menu code -> show reformat file dialog -> centang semua field dan klik ok.

DETAIL PENILAIAN TUGAS

Kriteria	Nilai
Kegiatan 1	45
Kegiatan 2	40
Pemahaman Materi Keseluruhan	15