

VERSION 1.0

February 6, 2024



[STRUKTUR DATA]

MODUL 4, HASHMAP

DISUSUN OLEH:

IZZA IHSAN FATONY

MOCH IQBAL ARIZKI WIDYANSYAKH

DIAUDIT OLEH:

MUHAMMAD ILHAM PERDANA. S.TR.T., M.T.

PRESENTED BY: TIM LAB-IT

UNIVERSITAS MUHAMMADIYAH MALANG

PETUNJUK Pengerjaan Modul

Perhatikan petunjuk praktikum dibawah ini:

1. Wajib membaca materi modul, sewaktu – waktu dapat direview oleh asisten saat demo
2. Gunakan referensi yang disediakan modul dan referensi lain pada google (yang kredibel)
3. Latihan praktikum wajib dikerjakan pada praktikum minggu pertama secara bersama - sama di laboratorium dan tidak boleh dijadikan pekerjaan rumah
4. Tugas praktikum boleh dijadikan pekerjaan rumah dan di demokan kepada asisten pada praktikum minggu kedua
5. Memperhatikan kerapian *source code* termasuk aturan penamaan Class, Method, Variable, File, dan lain - lainnya.
6. Segera lapor kepada asisten jika ada kesalahan pada modul praktikum.

PERSIAPAN MATERI

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

1. Array List
2. Linked List

TUJUAN

Mahasiswa mampu menguasai dan 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 (Intelij IDEA, Eclipse, Netbeans, dll)

REFERENSI MATERI

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

Artikel

https://www.w3schools.com/java/java_hashmap.asp

<https://www.javatpoint.com/java-hashmap>

<https://www.geeksforgeeks.org/java-util-hashmap-in-java-with-examples/>

<https://www.petanikode.com/java-hashmap/>

Youtube

https://www.youtube.com/watch?v=H62Jfv1DJIU&ab_channel=CodingwithJohn

https://www.youtube.com/watch?v=gGSv7TUeFQg&t=166s&ab_channel=brongo-ding

https://www.youtube.com/watch?v=-DVT3SvyXcA&t=43s&ab_channel=SekolahKoding

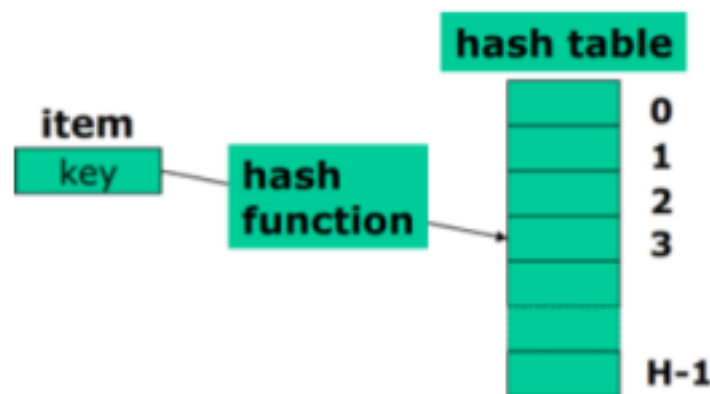
https://www.youtube.com/watch?v=Tp0nma5LIZY&ab_channel=RhioSutoyo

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. $h(k) \rightarrow \{0,1,2,\dots,m-1\}$ dimana m adalah ukuran array. Sifat – sifat yang diharapkan dari $h(k)$ adalah mudah dalam komputasi dan menghasilkan kunci distribusi yang uniform sepanjang $(0,1,2,\dots,m-1)$, diharapkan $h(k_1) \neq h(k_2)$ apabila $k_1 \neq k_2$ (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

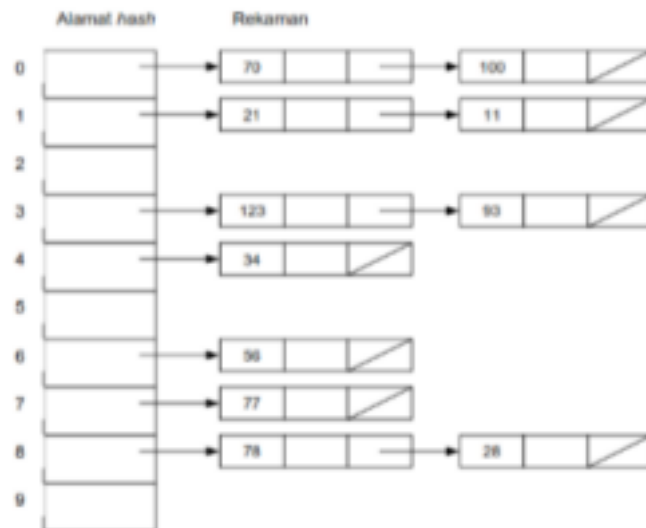
2. Chaining

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 $k \bmod 10$. 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 mengizinkan data null bagi key dan value. Class HashMap mirip seperti Hashtable kecuali dalam HashMap tidak tersinkronasi dan mengizinkan 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 dan set), 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:

```
// Inisiasi 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:

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

CODELAB

LATIHAN 1

Berikut adalah contoh pemanfaatan HashMap pada data gudang toko alat-alat perkantoran:

```
import java.util.HashMap;
import java.util.Map;

public class Inventory{
    public static void main(String[] args) {
        HashMap<String, Integer> inventori = new HashMap<>();

        inventori.put("Pensil", 50);
        inventori.put("Buku", 35);
        inventori.put("Penghapus", 25);

        System.out.println("Inventori awal: " + inventori);

        inventori.put("Pensil", inventori.get("Pensil") + 20);

        inventori.remove("Penghapus");

        System.out.println("Inventori setelah update: " + inventori);

        Scanner scanner = new Scanner(System.in);
        System.out.println("Masukkan nama barang yang ingin dicari: ");
        String barang = scanner.nextLine();

        cariBarang(inventori, barang);
    }

    public static void cariBarang(HashMap<String, Integer> inventori, String barang) {
        if (inventori.containsKey(barang)) {
            System.out.println("Stok " + barang + ": " + inventori.get(barang) + " unit.");
        } else {
            System.out.println("Barang " + barang + " tidak tersedia di inventori.");
        }
    }
}
```

Penjelasan Code

- **HashMap<String,Integer> inventori = new HashMap<>()** berfungsi untuk membuat variabel inventori bertipe hashmap untuk menyimpan inventori toko yang memiliki key bertipe String dan value bertipe Integer.
- **Inventori.put("Pensil", 50)** yaitu menambah barang ke dalam variabel inventori yang bertipe hashmap yang mempunyai key Pensil(string) dan value 50(integer).
- **Inventori.get("Pensil")** memiliki arti mengambil data dari hashmap inventori yang memiliki key "Pensil"
- **Inventori.put("Pensil", inventori.get("Pensil") + 20)** yaitu merubah value dari hashmap inventori yang memiliki key "Pensil" menjadi value dari key "Pensil" ditambahkan dengan 20, sekarang value dari key

“Pensil” adalah $50 + 20$ yaitu 70

- **Inventori.remove(“Penghapus”)** artinya adalah menghapus data dari hashmap inventori yang mempunyai key “Penghapus”
- **cariBarang(inventori, barang)** adalah melakukan pemanggilan function cariBarang yang diisi dengan hashmap inventori dan barang dimana barang adalah inputan dari user
- **If(Inventori.containsKey(barang))** yaitu melakukan pengecekan apakah di dalam hashmap inventori memiliki key dari variabel barang yang dimana barang didapatkan dari parameter function cariBarang

TUGAS PRAKTIKUM

KEGIATAN 1

Buatlah sebuah source code bertemakan **sistem voting online** yang memiliki atribut bernama **candidates** bertipe **HashMap**. Dalam kegiatan 1, sistem ini bisa memasukkan **voting lebih dari satu**. Manfaatkan library **HashMap** untuk mempermudah pembuatan kelas.

Key dan value dari atribut **candidates** berupa nama kandidat yang bertipe string dan jumlah voting yang bertipe integer. Berikut adalah contoh output yang dihasilkan.

```
Selamat datang di Sistem Voting Online

Pilih kandidat yang ingin Anda dukung:
- Kandidat B
- Kandidat C
- Kandidat A
Masukkan nama kandidat (atau ketik 'selesai' untuk keluar): Kandidat B
Terima kasih, suara Anda telah direkam.

Pilih kandidat yang ingin Anda dukung:
- Kandidat B
- Kandidat C
- Kandidat A
Masukkan nama kandidat (atau ketik 'selesai' untuk keluar): Kandidat B
Terima kasih, suara Anda telah direkam.

Pilih kandidat yang ingin Anda dukung:
- Kandidat B
- Kandidat C
- Kandidat A
Masukkan nama kandidat (atau ketik 'selesai' untuk keluar): Kandidat A
Terima kasih, suara Anda telah direkam.

Pilih kandidat yang ingin Anda dukung:
- Kandidat B
- Kandidat C
- Kandidat A
Masukkan nama kandidat (atau ketik 'selesai' untuk keluar): selesai

Hasil Voting:
- Kandidat B: 2 suara
- Kandidat C: 0 suara
- Kandidat A: 1 suara
```


KEGIATAN 2

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

Metode Register

- User diminta memasukkan email, password, nama, dan nik
- Simpan email dan password dalam satu atribut users yang bertipe hashmap
- Simpan juga email, nama dan nik dalam satu atribut userDetails bertipe hashmap.
- Dalam atribut userDetails gunakan email sebagai key dan arraylist yang berisi nama dan nik sebagai value.
- Tidak boleh ada duplikat email dan nik dalam satu sistem.
- Format email harus menggunakan "@gmail.com"
- Jika semua ketentuan tersebut sudah terpenuhi maka tampilkan pilihan antara login atau daftar, juga tampilkan "Berhasil Mendaftar"
- Jika semua data rules tidak terpenuhi maka kembalikan ke menu pilihan login dan daftar dan tampilkan "Gagal Login"

Metode Login:

- User diminta memasukkan email dan password
- Sistem memeriksa apakah data email dan password sesuai dengan data yang tersimpan pada atribut users
- Jika ada, diperiksa lagi apakah email yang dimasukkan menggunakan domain "@gmail.com"
- Jika data tidak sesuai maka tampilkan "Gagal Login"
- Jika data sesuai maka tampilkan "Login Berhasil" dan tampilkan juga nama dan nik dari akun yang sudah login, kemudian tampilkan pilihan kandidat nya

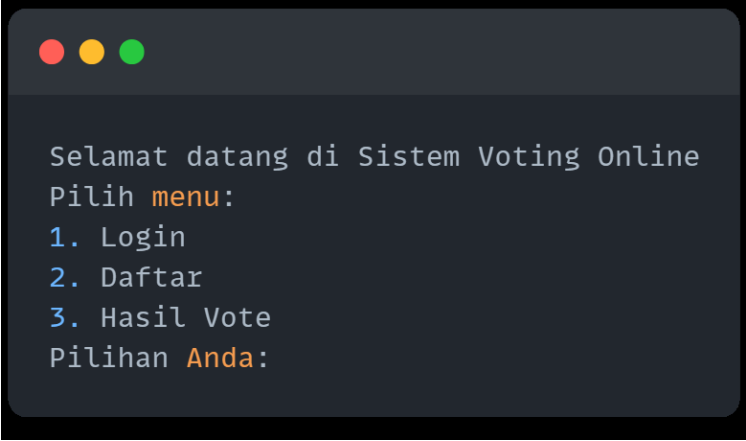
Metode Pemilihan:

- Satu user **maksimal hanya bisa memilih satu kandidat**

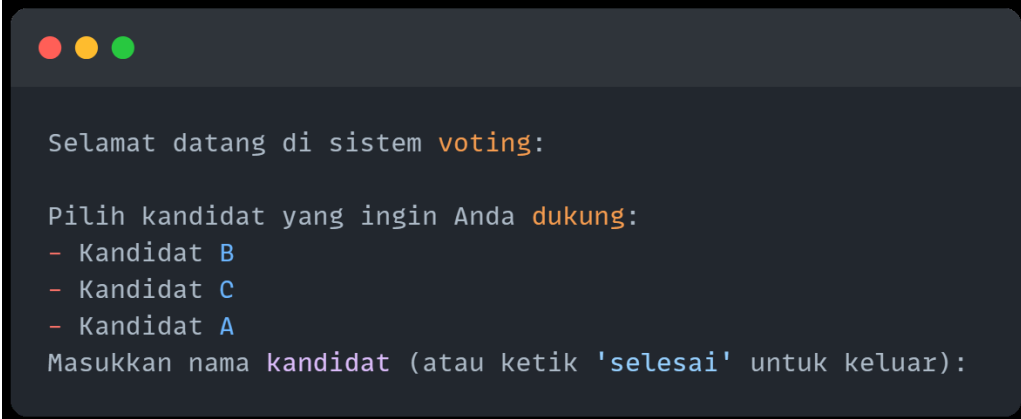
Metode Logout:

- User keluar dari program

Berikut adalah contoh output dari program



```
Selamat datang di Sistem Voting Online
Pilih menu:
1. Login
2. Daftar
3. Hasil Vote
Pilihan Anda:
```



```
Selamat datang di sistem voting:

Pilih kandidat yang ingin Anda dukung:
- Kandidat B
- Kandidat C
- Kandidat A
Masukkan nama kandidat (atau ketik 'selesai' untuk keluar):
```

CATATAN

Aturan umum penulisan JAVA agar mudah dikoreksi oleh asisten:

1. Untuk nama class, enum, dan yang lainnya biasanya 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.

KRITERIA & DETAIL PENILAIAN

Kriteria	Nilai
Codelab	20
Tugas 1	30
Tugas 2	30
Pemahaman	20
Total	100