MODUL 8
HASHMAP

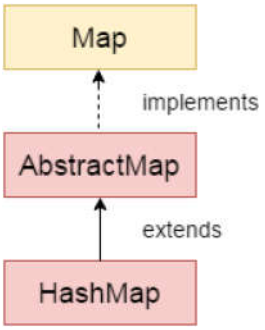
TUJUAN PEMBELAJARAN:

1. Mampu mengimplementasikan struktur data HashMap
2. Mampu memanfaatkan struktur data HashMap untuk menyelesaikan permasalahan.

PENGANTAR:

HashMap

Class HashMap merupakan class turunan dari class AbstractMap dan implementasi dari interface Map.



Gambar 1. Arsitektur HashMap

HashMap adalah sebuah class yang berisi sekumpulan pasangan nilai (value) dan kunci (key). Nilai bisa dalam bentuk string, integer, boolean, float, double, dan objek. Sedangkan untuk key biasanya dalam bentuk string dan integer. HashMap bisa dibilang seperti Array asosiatif dalam Java. Tabel 1 terdiri dari pasangan key dan value, seperti inilah isi dari class atau objek HashMap.

Tabel 1. Contoh HashMap

Key	Value
“name”	“Rekayasa Perangkat Lunak”
“url”	“https://se.ittelkom-pwt.ac.id”
“email”	“se@ittelkom-pwt.ac.id”
“isActive”	true

Method HashMap

```
size()
isEmpty()
containsKey(Object key)
containsValue(Object value)
get(K key)
put(K key, V val)
remove(K key)
```

PRAKTEK:
Pada percobaan ini, kita akan mendefinisikan sendiri method-method HashMap.

Percobaan 1: Membuat kelas Entry

```
public class Entry<K, V> {
    K key;
    V val;

    public K getKey() {
        return key;
    }

    public void setKey(K key) {
        this.key = key;
    }

    public V getVal() {
        return val;
    }

    public void setVal(V val) {
        this.val = val;
    }

    public int hashCode() {
        int prime = 13;
        int mul = 11;
        if (key != null) {
            int hashCode = prime * mul + key.hashCode();
            return hashCode;
        }
        return 0;
    }

    public boolean equals(Object o) {
        if (this == o) {
            return true;
        }
        if (o == null || this.getClass().getName() !=
o.getClass().getName()){
            return false;
        }
        Entry e = (Entry) o;
        if (this.key == e.key) {
            return true;
        }
        return false;
    }
}
```

Percobaan 2: Membuat method-method HashMap

Method Hashing

```
private int Hashing(int hashCode) {
    int location = hashCode % capacity;
    System.out.println("Location:" + location);
    return location;
}
public int size() {
    return this.size;
}
```

Method isEmpty, containsKey, containsValue

```
public boolean isEmpty() {
    if (this.size == 0) {
        return true;
    }
    return false;
}

public boolean containsKey(Object key) {
    if (key == null) {
        if (table[0].getKey() == null) {
            return true;
        }
    }
    int location = Hashing(key.hashCode());
    Entry e = null;
    try {
        e = table[location];
    } catch (NullPointerException ex) {

    }
    if (e != null && e.getKey() == key) {
        return true;
    }
    return false;
}
public boolean containsValue(Object value) {
    for (int i = 0; i < table.length; i++) {
        if (table[i] != null && table[i].getVal() == value) {
            return true;
        }
    }
    return false;
}
```

Method get

```
public V get(K key) {
    V ret = null;
    if (key == null) {
        Entry<K, V> e = null;
        try {
            e = table[0];
        } catch (NullPointerException ex) {

        }
        if (e != null) {
            return (V) e.getVal();
        }
    } else {
        int location = Hashing(key.hashCode());
        Entry<K, V> e = null;
        try {
            e = table[location];
        } catch (NullPointerException ex) {

        }
        if (e != null && e.getKey() == key) {
            return (V) e.getVal();
        }
    }
    return ret;
}
```

Method put

```
public V put(K key, V val) {
    V ret = null;
    if (key == null) {
        ret = putForNullKey(val);
        return ret;
    } else {
        int location = Hashing(key.hashCode());
        if (location >= capacity) {
            System.out.println("Rehashing required");
            return null;
        }
        Entry<K, V> e = null;
        try {
            e = table[location];
        } catch (NullPointerException ex) {

        }
        if (e != null && e.getKey() == key) {
            ret = (V) e.getVal();
        } else {
            Entry<K, V> eNew = new Entry<K, V>();
            eNew.setKey(key);
            eNew.setVal(val);
            table[location] = eNew;
            size++;
        }
    }
    return ret;
}
```

Method putForNullKey

```
private V putForNullKey(V val) {
    Entry<K, V> e = null;
    try {
        e = table[0];
    } catch (NullPointerException ex) {

    }
    V ret = null;
    if (e != null && e.getKey() == null) {
        ret = (V) e.getVal();
        e.setVal(val);
        return ret;
    } else {
        Entry<K, V> eNew = new Entry<K, V>();
        eNew.setKey(null);
        eNew.setVal(val);
        table[0] = eNew;
        size++;
    }
    return ret;
}
```

LAPORAN PRAKTIKUM:

Soal 1: Buatlah method main untuk menjalankan method-method HashMap!
Soal 2: Jalankan masing-masing method dan tuliskan hasilnya!
Soal 3: Analisis masing-masing kode pada method HashMap!
Soal 4: Buatlah method remove untuk menghapus nilai berdasarkan input kunci!