HASH TABLE

PRAKTIKUM

TK13024 - STRUKTUR DATA

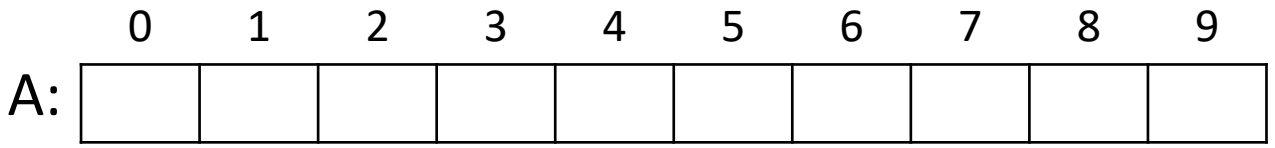
Materi Praktikum

Pertemuan 1	Review Dasar Program Java dan Membuat program Rekursif
Pertemuan 2	Membuat program dengan Array dan Linked List
Pertemuan 3	Membuat program Stack dan Queue
Pertemuan 4	Membuat program Hash Tables
Pertemuan 5	Membuat program Binary Search Tree
Pertemuan 6	Membuat program Heap dan Heapsort
Pertemuan 7	Membuat program Huffman Coding
Pertemuan 8	UTS (tidak ada praktikum)
Pertemuan 9	Membuat program 2-3 Tree
Pertemuan 10	Membuat program DFS dan BFS
Pertemuan 11	Membuat program Shortest Path
Pertemuan 12	Membuat program Minimum Spanning Tree
Pertemuan 13	Membuat program Topological Sort
Pertemuan 14	Membuat program untuk tugas akhir
Pertemuan 15	Membuat program untuk tugas akhir
Pertemuan 16	UAS (tidak ada praktikum)

List dengan Array

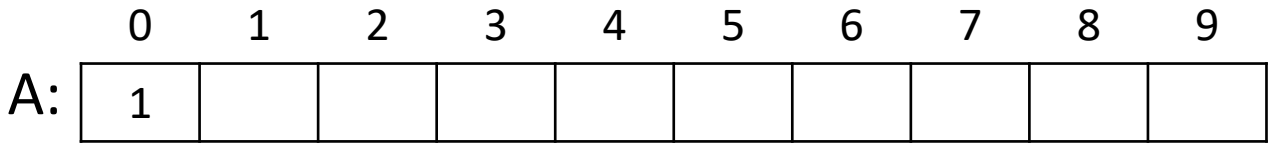
List dengan Array (1)

Array A dengan jumlah komponen maksimal 10



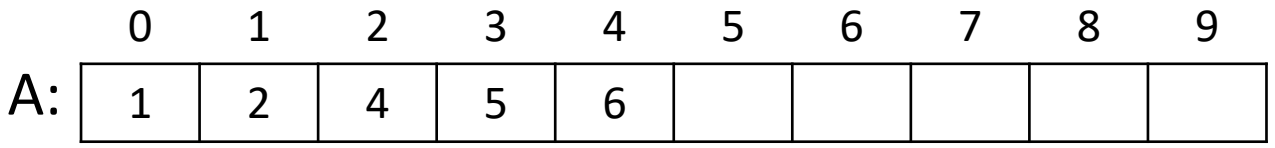
Add(1**)**

Size() ➔ 1



Add(2**)**, **Add(**4**)**, **Add(**5**)**, **Add(**6**)**

Size() ➔ 5

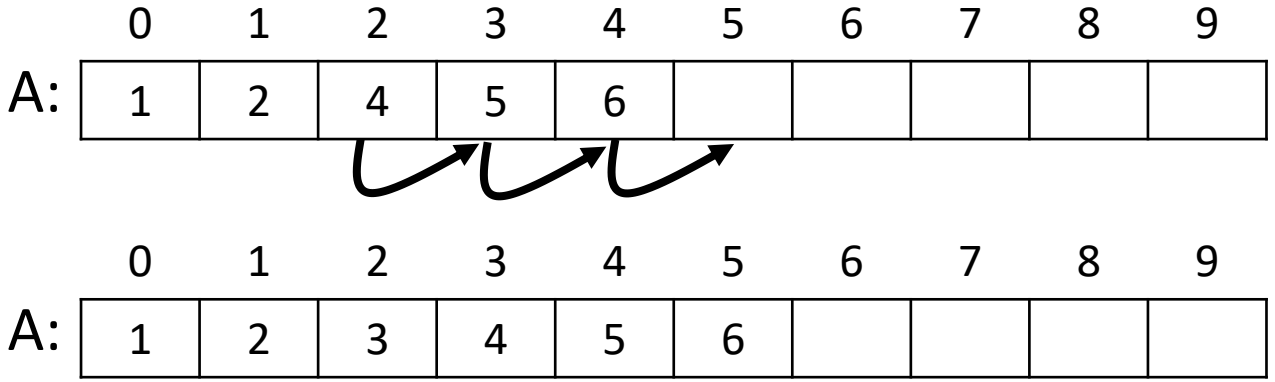


List dengan Array (2)

Add(2,3**)**

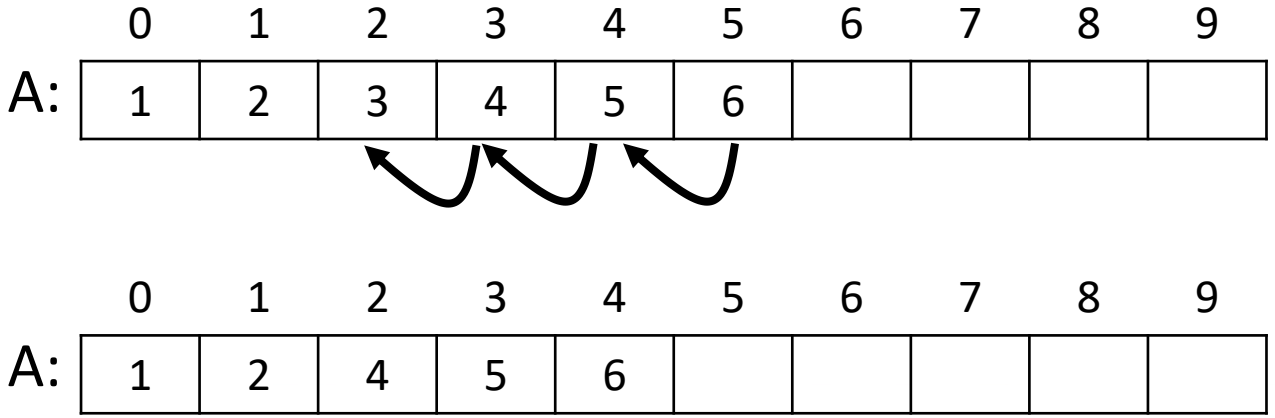
posisi / indeks untuk menyisipkan angkat 3

angka yang akan disisipkan



remove(2)

posisi / indeks elemen di array yang akan dihapus



List dengan Array (3)

- **add(***E e***)**: appends the specified element to the end of this list
- **add(***int index, E element***)**: Inserts the specified element at the specified position in this list
- **remove(***int index***)**: removes the element at the specified position in this list
- **get(***int index***)**: returns the element at the specified position in this list
- **isEmpty()**: returns true if this list contains no elements
- **clear()**: removes all of the elements from this list
- **set(***int index, E element***)**: replaces the element at the specified position in this list with the specified element
- **size()**: returns the number of elements in this list

Membuat Sendiri Generic Array List (1)

```
/*
* Generic Array List: Creating Our own version of Java's ArrayList
* Asumsi: ArrayList dibuat dengan ukuran maximum yang fixed
* Generic Array List ini unchecked (weak typing) generic data type
*/

public class TheArrayList <E> {
        /*
        * Prinsip Encapsulation dari OOP:
        * 1. semua variabel dari class HARUS private
        *    (hanya dapat diakses oleh classnya sendiri)
        * 2. akses variabel (set dan get) melalui public method
        *    yang dibuat di class ini.
        */
        Object[] thelist;
        private int n;
        private int max_size;

//slide berikutnya…
```

- Java **tidak bisa** mengalokasikan ukuran array untuk generic element E
  - E[] thelist = new E[max_size] (**error**)
  - Object[] thelist = new object[max_size] (**OK**)

Membuat Sendiri Generic Array List (2)

```
    //Constructor
    public TheArrayList(int max_size) {
        thelist = new Object[max_size];
        n = 0;
        this.max_size = max_size;
    }
    //mengembalikan ukuran maksimum array
    public int maxSize() {
        return max_size;
    }
    //mengembalikan jumlah elemen array
    //saat ini
    public int size() {
        return n;
    }

//slide berikutnya…
```

Membuat Sendiri Generic Array List (3)

```
        //mengembalikan true jika jumlah elemen
        //sudah mencapai kapasitas (maximum size)
        //dari list
        private boolean isFull() {
                if(n == max_size) return true;
                else return false;
        }

        public boolean isEmpty() {
                if(n == 0) return true;
                else return false;
        }

//slide berikutnya…
```

Membuat Sendiri Generic Array List (3)

```
    public void add(E value) {
        if(!isFull()) {
            thelist[n] = value;
            n = n + 1;
        }
        else System.out.println("List sudah penuh!");
    }
    public void add(int index, E value) {
        if(index >= 0 && !isFull()) {
            n = n + 1;
            int i = n;
            do {
                thelist[i] = thelist[i-1];
                i = i - 1;
            }while(i > index);
            thelist[index] = value;
        }
        else System.out.println("List sudah penuh!");
    }
//slide berikutnya…
```

Membuat Sendiri Generic Array List (4)

```
    public void remove(int index) {
        if(index >= 0 && !isEmpty()) {
            for(int i = index; i < n-1; i++)
            thelist[i] = thelist[i+1];
            thelist[n-1] = null;
            n = n - 1;
        }
    }
    public E get(int i) {
        //compile will ignore type casting dari
        // unchecked data type dari elemen thelist
        @SuppressWarnings("unchecked")
        final E e = (E) thelist[i];
        return e;
    }
    public void set(int index, E value) {
        thelist[index] = value;
    }
```

//slide berikutnya…

Membuat Sendiri Generic Array List (5)

```
    public void clear() {
        if(!isEmpty()) {
            for(int i = 0; i < n; i++) thelist[i] = null;
            n = 0;
        }
    }

    public void cetakList() {
        //jika list kosong, tampilkan pesan list kosong
        if(isEmpty()) System.out.println("List kosong!");
        // jika list tidak kosong, maka cetak elemen pada list
        else {
            System.out.print("[ ");
            for(int i = 0; i < n; i++)
                System.out.print(thelist[i].toString() + " ");
            System.out.println("]");
        }
    }
}
```

Membuat Package strukdat

- Membuat **package strukdat** untuk koleksi semua generic class yang dibuat sendiri, seperti SingleList.java dan TheArrayList.java:
  - Untuk praktikum ini diasumsikan satu folder PraktikumDS digunakan untuk menyimpan semua program praktikum sebagai berikut:

```
C:
 |_PraktikumDS
     |_Prak1.java
     |_Prak2.java
     |_Prak3.java
     |_SingleList.java
```

  - Misalkan TheArrayList.java disimpan di folder PraktikumDS:

```
C:
 |_PraktikumDS
     |_Prak1.java
     |_Prak2.java
     |_Prak3.java
     |_SingleList.java
     |_TheArrayList.java
```

Membuat Package strukdat (2)

- Tambahkan **package strukdat** ke SingleList.java (nama package harus huruf kecil semua):

```
package strukdat;

class Node<T> {
    T data;
    Node<T> next;

    //constructor
    Node(T value) {
        data = value;
        next = null;
    }
}

public class SingleList<T> {...}
```

Membuat Package strukdat (3)

- Tambahkan **package strukdat** ke TheArrayList.java (nama package harus huruf kecil semua):

```
package strukdat;
/*
* Generic Array List: Creating Our own version of Java's
ArrayList
* Generic Array List ini unchecked (weak typing) generic data
type
*/

public class TheArrayList <E> {...}
```

Membuat Package strukdat (4)

- Compile:
  - javac –d . SingleList.java
  - javac –d . TheArrayList.java
- Hasilnya terbuat direktori strukdat:

C:
|_PraktikumDS
|_Prak1.java
|_Prak2.java
|_Prak3.java
|_SingleList.java
|_TheArrayList.java
|_strukdat
|_Node.class
|_SingleList.class
|_ TheArrayList.class

Lakukan ini untuk generic data structure lain di praktikum berikutnya!

Menggunakan ArrayList

```
import strukdat.TheArrayList;

//Main Method
public class MainProgram {
	public static void main(String[] args) {
	TheArrayList<Integer> myList1 = new TheArrayList<Integer>(10);
	myList1.add(1);
	myList1.add(2);
	myList1.add(4);
	myList1.add(5);
	myList1.add(6);
	myList1.add(2,3); //insert 3 ke index 2
	myList1.cetakList();
	myList1.remove(3); //remove item di index 3
	myList1.cetakList();
}
```

Output:
[1 2 3 4 5 6]
[1 2 3 5 6]

Menggunakan ArrayList

Item adalah Object dari class Student

```
import strukdat.TheArrayList;

class Student {
	private int nim;
	private String nama;

	Student(int nim, String nama) {
		this.nim = nim;
		this.nama = nama;
	}

	@Override //toString dari class String
	public String toString() {
		return(Integer.toString(nim) + " - " + nama + " ");
	}
}
//Lanjut dislide berikutnya...
```

Menggunakan ArrayList

Item adalah Object dari class Student

```
//Main Method
public class MainProgram {
	public static void main(String[] args) {
		TheArrayList<Student> myList3 = new TheArrayList<Student>(10);
		myList3.add(new Student(535230001, "Dodi Lim"));
		myList3.add(new Student(535230001, "Didi Ciang"));
		myList3.add(new Student(535230001, "Dedi Koh"));
		myList3.add(new Student(535230001, "Dudi Liang"));
		myList3.cetakList();
	}
}
```

Output:
[535230001 - Dodi Lim 535230001 - Didi Ciang 535230001 - Dedi Koh 535230001 - Dudi Liang]

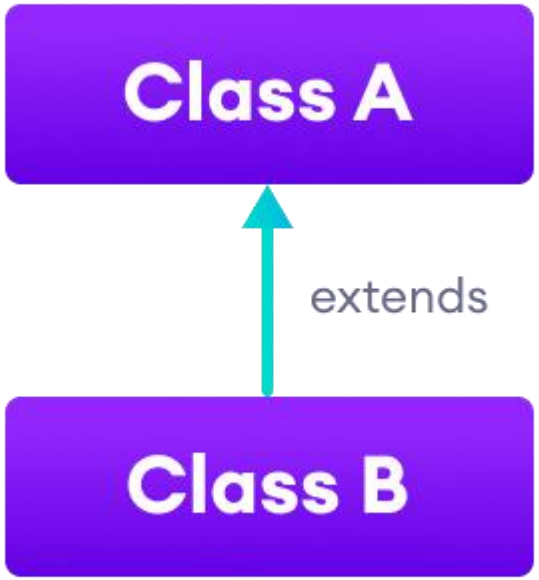
Inheritance & Polymorphism

Inheritance?

- Membuat class baru dari class yang sudah ada
  - Class baru disebut sebagai subclass, child class atau derived class
  - Class yang sudah ada disebut base class atau parent class
  - subclass dapat menggakses variables dan methods dari base class
  - subclass dapat mengcode ulang methods dari base class (method overriding)
- Mengapa Inheritance Di butuhkan?
  - Code reusability
  - Method Overriding
  - Abstraction

Lima Jenis Inheritance di Java

1. Single Inheritance

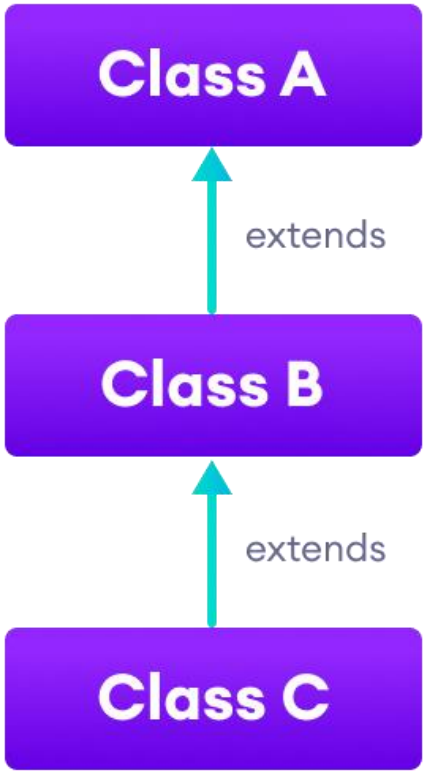


- Class A adalah **base class** dari Class B
- Class B adalah **subclass** dari class A
- Class B bisa mengakses variables, constructor dan methods dari Class A
  - **super()**: wajib ada jika constructor dari base class memerlukan inisialisasi untuk variabelnya.
  - **super.***namavariabel* = mengakses sebuah variable di base class, jika nama dari variabel tersebut sama dengan nama variable di subclass B.
  - **super.***namamethod()* = mengakses sebuah method di base class yang dioverridden di subclass B.

https://www.programiz.com/java-programming/inheritance#google_vignette

Lima Jenis Inheritance

2. Multilevel Inheritance

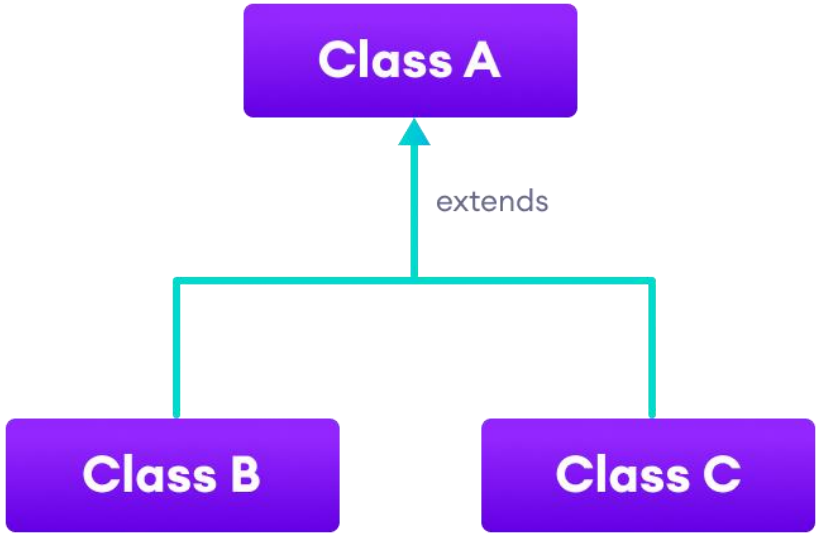


- Class C bisa mengakses variables, constructor dan methods class B (sebagai parent class), tetapi tidak bisa mengakses variables, constructor dan methods Class A (sebagai Grandparent class)

https://www.programiz.com/java-programming/inheritance#google_vignette

Lima Jenis Inheritance

3. Hierarchical Inheritance

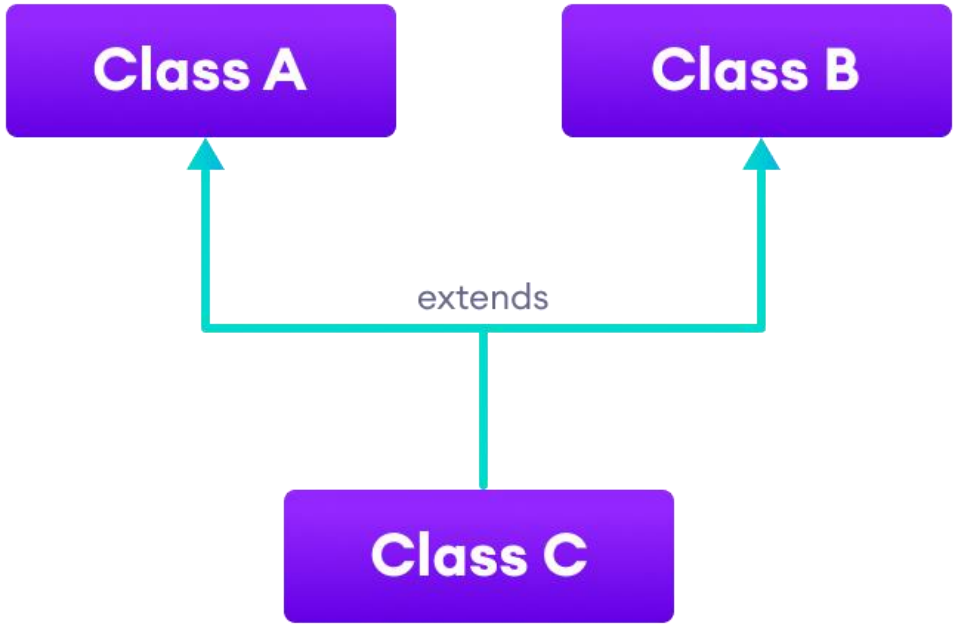


- Class B, C bisa mengakses langsung variables, constructor dan methods class A (sebagai parent class)

https://www.programiz.com/java-programming/inheritance#google_vignette

Lima Jenis Inheritance

4. Multiple Inheritance

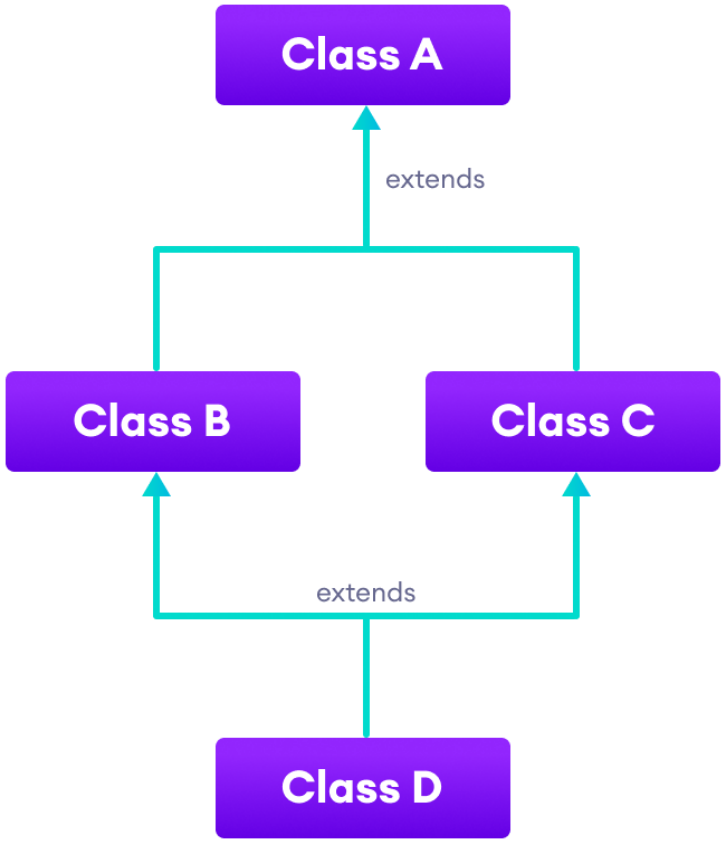


- Java **TIDAK** mensupport multiple inheritance.
- Java mengimplementasikan multiple inheritance menggunakan **interfaces**.

https://www.programiz.com/java-programming/inheritance#google_vignette

Lima Jenis Inheritance

5. Hybrid Inheritance



- Java **TIDAK** mensupport hybrid inheritance.
- Java mengimplementasikan hybrid inheritance menggunakan **interfaces**.

https://www.programiz.com/java-programming/inheritance#google_vignette

Contoh Inheritance (1)

```
// Base class
class Class_A {

        private char code;
        public Class_A(char kode) {
            code = kode;
        }

        public void print_A()
        {
            System.out.println("Parent " + code);
        }
}
```

Contoh Inheritance (2)

```
//B subclass dari Class_A
class Class_B extends Class_A {

        private int no;

        public Class_B(char code, int no) {
                super(code);
                this.no = no;
        }

        public void print_B() {
                System.out.print("Child " + no + " dari ");
                print_A();
        }
}
```

Contoh Inheritance (3)

```
//C subclass dari Class_B
class Class_C extends Class_B {

        private char id;

        public Class_C(char code, int no, char id) {
                super(code, no);
                this.id = id;
        }

        public void print_C() {
                System.out.print("Grandchild " + id + " dari ");
                print_B();
        }
}
```

Contoh Inheritance (4)

```
// Main program class
public class SingleInheritance {
	// Main function
	public static void main(String[] args)
	{
		Class_C c = new Class_C('A',1,'i’);
		c.print_A();
		c.print_B();
		c.print_C();
	}
}
```

Polymorphism

- Entitas yang sama dapat melakukan operasi yang berbeda
  - Constructor Overloading
  - Method
    - Method Overriding
    - Method Overloading
  - Operator
    - Operator ‘+’ : penjumlahan numerik dan penggabungan string
    - Operator ‘&’, ‘|’, and ‘!’ : untuk operasi logika dan operasi bit.
  - Variables:
    - Polymorphic Variables
    - Object variables (instance variables) mereferensi object dari class dan subclass (class turunannya).

Hash Table

Hashing.java – Base/Parent Class

LinearProbing.java – Subclass dari Hashing

Base Class: Hashing.java

```
package strukdat;
import strukdat.TheArrayList;

class HashNode<K,V> {...}

public class Hashing<K,V> {
        private int size;
        TheArrayList<HashNode<K,V>> table;
        // Constructor
        Hashing(int capacity) {
                table = new TheArrayList<HashNode<K,V>>(capacity);
                size = 0;
        }
        // Convert key to numbers
        int convertToNumber(K key) {...}
        // Memeriksa tabrakan
        boolean isCollision(int hashkey) {
        //Mengembalikan jumlah item di hash table
        int size() {...}
        // Menyisipkan value (tidak mengatasi collision)
        public void put(K key, V value) {...}
        //Menampilkan isi hash tabel
        public void displayHashTable() {...}
```

Base Class: Hashing.java

```
package strukdat;
import strukdat.TheArrayList;

class HashNode<K,V> {...}

public class Has
    private int
    TheArrayL
    // Constru
    Hashing(ir
        table
        size =
    }
    // Convert
    int conver
    // Memer
    boolean is
    //Mengen
    int size() {...}
    // Menyisipkan value (tidak mengatasi collision)
    public void put(K key, V value) {...}
    //Menampilkan isi hash tabel
    public void displayHashTable() {...}
```

```
class HashNode<K,V> {
    K key;
    V data;

    HashNode(K key, V data) {
        this.key = key;
        this.data = data;
    }
    @Override //toString dari class String
    public String toString() {
        return(key.toString() + " - " + data.toString() + " ");
    }
}
```

Base Class: Hashing.java

```
package strukdat;
import strukdat.TheArrayList;

class HashNode<K,V> {...}

public class Hashing<K,V> {
        private int size;
        TheArrayList<HashNode<K,
        // Constructor
        Hashing(int capacity) {
                table = new TheArrayL
                size = 0;
        }
        // Convert the key to numb
        int convertToNumber(K key
        // Memeriksa tabrakan
        boolean isCollision(int hash
        //Mengembalikan jumlah it
        int size() {...}
        // Menyisipkan value (tidak mengatasi collision)
        public void put(K key, V value) {...}
        //Menampilkan isi hash tabel
        public void displayHashTable() {...}
```

```
// Convert key to numbers
int convertToNumber(K key){
        String theKey = (key.toString()).toLowerCase();
        int k = 0, bil;
        int i = 0;
        int j = theKey.length() - 1;
        do {
                bil = theKey.charAt(i);
                if(bil >= 48 && bil <= 57) bil = bil - 48;
                else bil = bil - 97 + 1;
                k = k + (bil * (int) Math.pow(10,j));
                i++; --j;
        }while(i < theKey.length());
        return k;
}
```

convertToNumber(K key):

- Jika key berisi angka semua maka dikonversikan tetap ke angka (Integer), contoh:
  key = 123 atau key = “123”
  $k = 1*10^2 + 2*10^1 + 3*10^0 = 123$
- Jika key berisi alphanumeric maka, dikonversikan ke angka (Integer); setiap huruf dijadikan huruf kecil dan dipetakan ke angka 1 – 26. Contoh a => 1, b => 2, c => 3, ..., z => 26. contoh:
  key = “a23”
  $k = 1*10^2 + 2*10^1 + 3*10^0 = 123$

Base Class: Hashing.java

```
package strukdat;
import strukdat.TheArrayList;

class HashNode<K,V> {...}

public class Hashing<K,V> {
        private int size;
        TheArrayList<HashNode<K,V>> table;
        // Constructor
        Hashing(int capacity) {
                table = new TheArrayList<HashNode<K,V>>(capacity);
                size = 0;
        }
        // Convert the key to numbers
        int convertToNumber(K key) {...}
        // Memeriksa tabrakan
        boolean isCollision(int hashkey) {
        //Mengembalikan jumlah item di has
        int size() {...}
        // Menyisipkan value (tidak mengata
        public void put(K key, V value) {...}
        //Menampilkan isi hash tabel
        public void displayHashTable() {...}
```

```
// Memeriksa tabrakan dengan cara
// mengecek apakah tabel di index
// hashkey berisi data atau kosong (null)
boolean isCollision(int hashkey) {
        if(table.get(hashkey) != null) {
                return true;
        }
        return false;
}
```

Base Class: Hashing.java

```
package strukdat;
import strukdat.TheArrayList;

class HashNode<K,V> {...}

public class Hashing<K,V> {
        private int size;
        TheArrayList<HashNode
        // Constructor
        Hashing(int capacity) {
                table = new TheArr
                size = 0;
        }
        // Convert the key to nu
        int convertToNumber(K
        // Memeriksa tabrakan
        boolean isCollision(int h
        //Mengembalikan jumla
        int size() {...}
        // Menyisipkan value (ti
        public void put(K key, V
        //Menampilkan isi hash tabel
        public void displayHashTable() {...}
```

```
//Mengembalikan jumlah item di hash table
int size() {
        return size;
}

//menaikkan size karena ada penambahan
//item di hash table
void incSize() {
        ++size;
}

//mengurangi size karena ada penghapusan
//item di hash table
void decSize() {
        --size;
}
```

Base Class: Hashing.java

```
package strukdat;
import strukdat.TheArrayList;

class HashNode<K,V> {...}

public class Hashing<K,V> {
        private int size;
        TheArrayList<HashNode<K,V>> table;
        // Constructor
        Hashing(int capaci
                table = new T
                size = 0;
        }
        // Convert the key
        int convertToNum
        // Memeriksa tabr
        boolean isCollision
        //Mengembalikan
        int size() {...}
        // Menyisipkan va
        public void put(K key, V value) {...}
        //Menampilkan isi hash tabel
        public void displayHashTable() {...}
```

```
// Menyisipkan value (tidak mengatasi collision)
public void put(K key, V value) {
        HashNode<K,V> N = new HashNode<K,V>(key, value);
        int h = convertToNumber(key) % table.maxSize();
        if(!isCollision(h)) {
                table.set(h,N);
                incSize();
        }
        else System.out.println("Item exist or table is full!");
}
```

NESIA

Base Class: Hashing.java

```
package strukdat;
import strukdat.TheArrayList;

class HashNode<K,V> {...}

public class Hashing<K,V> {
        private int size;
        TheArrayList<HashNode<K,V>> table;
        // Constructor
        Hashing(int capacity) {
                table = new TheArrayList<HashNode<K,V>>(capacity);
                size = 0;
        }
        //
        in
        //
        b
        //
        in
        //
        p
        //Menampilkan isi hash tabel
        public void displayHashTable() {...}
```

```
//Menampilkan isi hash tabel
public void displayHashTable() {
        System.out.println("Hash table contains " + size() + " items");
        for(int i = 0; i < table.maxSize(); i++) {
                if(table.get(i) != null)
                        System.out.println(i + ": " + table.get(i).toString() + " ");
                else System.out.println(i + ": ");
        }
}
```

Subclass: LinearProbing.java

```
package strukdat;
/* Hashing dengan Linear Probing untuk mengatasi collision */

public class LinearProbing<K,V> extends Hashing<K,V>{
	public LinearProbing(int capacity) {
		super(capacity);
	}
	// Menyisipkan value dan mengatasi collision
	// dengan Linear Probing
	@Override  //method put() dari base class yaitu Hashing
	public void put(K key, V value) {...}
	// Mengembalikan data sesuai input key,
	// tapi tidak menghapusnya dari Hash Table
	public HashNode<K,V> get(K key) {...}
	// Mengembalikan data Sesuai input key,
	// dan menghapusnya dari Hash Table
	public HashNode<K,V> remove(K key) {...}
}
```

Subclass: LinearProbing.java

```
package strukdat;
/* Hashing dengan Linear Pr

public class LinearProbing<K,
	public LinearProbing(int
		super(capacity);
	}
	// Menyisipkan value da
	// dengan Linear Probin
	@Override
	public void put(K key, V
	// Mengembalikan data
	// tapi tidak menghapus
	public HashNode<K,V> g
	// Mengembalikan data
	// dan menghapusnya d
	public HashNode<K,V> r
}
```

```
// Insert data dan mengatasi collision dengan Linear Probing
@Override
public void put(K key, V value) {
	HashNode<K,V> N = new HashNode<K,V>(key, value);
	int theKey = convertToNumber(key) % table.maxSize();
	int curKey = theKey;
	boolean firstScan = true;
	while (isCollision(curKey) && curKey >= 0) {
		curKey = (curKey+1) % table.maxSize();
		if(curKey == theKey && !firstScan) curKey = -1;
		firstScan = false;
	}
	if(curKey >= 0) {
		table.set(curKey,N);
		incSize();
	}
	else System.out.println("table is full!");
}
```

```java
// Mencari data dengan input key di Hash Table dengan Linear Probing
public HashNode<K,V> get(K key) {
        int theKey = convertToNumber(key) % table.maxSize();
        int curKey = theKey;
        boolean firstScan = true;
        while (isCollision(curKey) && convertToNumber(table.get(curKey).key) != convertToNumber(key) && curKey >= 0) {
                curKey = (curKey+1) % table.maxSize();
                if(curKey == theKey && !firstScan) curKey = -1;
                firstScan = false;
        }
        if(curKey >= 0) return table.get(curKey);
        else {
                System.out.println("Not found!");
                return null;
        }
}
```

```java
pac
/* H

pub
```

```java
    // Mengembalikan data sesuai input key,
    // tapi tidak menghapusnya dari Hash Table
    public HashNode<K,V> get(K key) {...}
    // Mengembalikan data Sesuai input key,
    // dan menghapusnya dari Hash Table
    public HashNode<K,V> remove(K key) {...}
}
```

```
pac
/* H

pub
```

```
public HashNode<K,V> remove(K key) {
        HashNode<K,V> N;
        int theKey = convertToNumber(key) % table.maxSize();
        int curKey = theKey;
        boolean firstScan = true;
        while (isCollision(curKey) && convertToNumber(table.get(curKey).key) != convertToNumber(key) && curKey >= 0) {
                curKey = (curKey+1) % table.maxSize();
                if(curKey == theKey && !firstScan) curKey = -1;
                firstScan = false;
        }
        if(curKey >= 0) {
                N = table.get(curKey);
                table.set(curKey,null);
                decSize();
                return N;
        }
        else {
                System.out.println("Not found!");
                return null;
        }
}
```

```
    // dan menghapusnya dari Hash Table
    public HashNode<K,V> remove(K key) {...}
}
```

Menggunakan Hashing.java dan LinearProbing.java

```
import strukdat.LinearProbing;

//Main Method
public class MainProgram {
	public static void main(String[] args) {
		LinearProbing<Object,String> T = new LinearProbing<Object,String>(7);
		T.put(2, "B");
		T.put(10, "J");
		T.put(14, "N");
		T.put(19, "S");
		T.put(24, "X");
		T.put(23, "W");
		T.displayHashTable();

		if (T.get(23) != null) System.out.println(("Data found: " + T.get(23));
		else System.out.println("data does not exist!");
	}
}
```

Soal Latihan no 1

20.5 Given the input {4371, 1323, 6173, 4199, 4344, 9679, 1989}, a fixed table size of 10, and a hash function $H(X) = X \bmod 10$, show the resulting

a. Linear probing hash table
b. Quadratic probing hash table
c. Separate chaining hash table

20.6 Show the result of rehashing the probing tables in Exercise 20.5. Rehash to a prime table size.

- Hitung jumlah probe yang diperlukan untuk memasukkan seluruh data ke table Hash pada soal 20. 5 dan 20.6 untuk tiap metode collision

Soal Latihan no 2

Diketahui data berikut ini:

261,381, 835, 195, 134, 477, 568, 99, 726, 139

a.Tentukan ukuran tabel Hash dan fungsi Hash sehingga tabrakan dapat dihindari seminim mungkin lalu petakan data di atas ke dalam tabel Hash tersebut

b.Jika terjadi tabrakan, atasi dengan Linear Probing, Quadratic Probing dan Double Hashing: h2(key) = 1 + key mod 7.

c.Hitunglah jumlah probe yang diperlukan untuk memetakan seluruh data ke ta.bel Hash pada masing-masing metode collision

Ketentuan

- Gunakan generic hashing (Hashing.java).
- Buatlah subclass dari Hashing.java untuk mengatasi masalah tabrakan dengan:
  - Quadratic probing
  - double hashing (H2(key) = 1 + key mod 7)
- Jalankan program dengan menggunakan data pada Latihan 1 dan 2

PR Praktikum

- Kelompok nomor ganjil: buat subclass untuk Quadratic probing
- Kelompok nomor genap: buat subclass untuk double hashing (H2(key) = 1 + key mod 7)