DAFTAR ISI

MODUL I : ARRAY

1. Deklarasi Array

Array biasanya disebut juga sebagai tabel, vektor, atau larik. *Array* (larik) merupakan koleksi data, dimana **setiap elemen memakai nama dan tipe yang sama**, serta setiap elemen diakses dengan membedakan indeks *array*-nya. Array adalah bentukan yang menyediakan penyimpanan untuk sejumlah item dengan tipe sama. Item-item array dapat berupa data sederhana atau komposit. Array di Java dideklarasikan dengan kurung siku: `[...]`. Array merupakan cara nyaman untuk pengelompokkan informasi yang berhubungan. Sebuah array dapat mempunyai sejumlah nilai, sedangkan sebuah varibel biasa hanya dihubungkan dengan sebuah nilai saja

Sintaks umum deklarasi array adalah sebagai berikut:

<type> var-name;

atau

<type> var-name;

Bentuk umum lainnya adalah: <tipedata>NamaArray[n] = {nilai_0, nilai_1, …, nilai_n};

Type adalah mendeklarasikan tipe basis dari array. Tipe basis menentukan tipe data bagi masing-masing elemen yang membentuk array. Dengan demikian, tipe basis untuk array menentukan tipe data yang dimuat oleh array.

Berikut adalah contoh deklarasi array di Java:

int numbers;

char letters;

long grid;

Pada Java, kita tidak perlu menspesifikasikan ukuran array kosong saat mendeklarasikan *array*. Kita harus selalu menyatakan ukuran array secara eksplisit saat melakukan operasi penciptaan menggunakan operator **new()** atau dengan mendaftarkan item-item untuk array pada saat penciptaan.

Kelebihan lain di Java adalah penempatan kurung siku pada deklarasi array. Kita dapat menempatkan kurung siku setelah tipe variabel atau setelah identifier.

Berikut ini adalah contoh pendeklarasian array dengan menspesifikasikan ukuran spesifiknya menggunakan operator **new()** dan dengan memberikan daftar item yang termasuk di dalam array.
char alphabet = new() **char**[26];
int primes = {7, 11, 13};

2. Array Satu Dimensi

Array satu dimensi adalah kumpulan dari variabel bertipe serupa. Kita lebih dulu harus menciptakan variabel array dari tipe yang dikehendaki. Elemen dari variabel tersebut memiliki tipe data sama, tetapi isi data elemen dibedakan menggunakan indeks.

Nilai	100	90	87	65	29	65	19	65	35
Indeks	0	1	2	3	4	5	6	7	8

Berikut contoh potongan bagian source code untuk inisialisasi array dengan nilai awal 0:

…;

int[] angka = new int angka[100];

for (i = 0; i < 100; i++) {

angka[i] = 0; … }

…;

Contoh : **int** monthDays[];

Meskipun deklarasi telah menetapkan bahwa **monthDays** adalah variabel array, namun sesungguhnya tidak ada array yang diciptakan pada saat itu. Nilai dari **monthDays** adalah nilai **null** yang merepresentasikan array tanpa nilai. Untuk merangkai **monthDays** dengan array bilangan bulat aktual dan fisik, kita harus mengalokasikan menggunakan operator **new()**. Operator **new()** merupakan operator khusus untuk mengalokasikan memori. Kita akan membahas lebih rinci mengenai operator **new()**, yang perlu kita ketahui saat ini adalah bahwa operator **new()** untuk mengalokasikan memori array. Bentuk umum **new()** untuk array 1 dimensi adalah sebagai berikut:

array-var = new() type[size]

Disini tipe menspesifikasikan tipe data yang dialokasikan, size menspesifikasikan jumlah elemen array, dan array-var adalah variabel array yang dirangkai ke array. Dengan demikian, untuk menggunakan **new()**, kita harus menspesifikasikan tipe dan jumlah elemen untuk dialokasikan. Elemen-elemen di array yang dialokasikan oleh **new()** akan secara otomatis diinisialisasi ke nol. Perhatikan contoh berikut:

```
monthDays = new() int[12];
```

Setelah pernyataan di atas dieksekusi, **monthDays** akan mengacu ke array berisi 12 bilangan int, kedua belas angka tersebut diinisialisasi dengan nol. Jadi, proses untuk memperoleh array ada dua langkah, yaitu:

a. Deklarasi variabel dari tipe array yang dikehendaki.

b. Alokasi memori yang akan menyimpan array menggunakan operator **new()** dan memberikannya ke variabel array.

Dengan cara tersebut, semua array adalah dialokasikan secara dinamis. Jika konsep alokasi dinamis ini tidak akrab bagi Anda, maka jangan kawatir. Konsep array dinamis ini lebih mudah dan ampuh dalam memperlakukan array. Begitu kita telah mengalokasikan array, kita dapat mengakses elemen spesifik di array dengan menspesifikasikan indeks dengan menspesifikasikannya di pasangan kurung siku. Semua array dimulai dengan indeks nol. Sebagai contoh, untuk memberikan nilai 29 ke elemen kedua dari **monthDays**, kita meng-gunakan perintah berikut:

```
monthDays[1] = 29;
```

Berikut ini adalah menampilkan nilai yang disimpan di indeks 3, yaitu:

```
System.out.println(monthDays[3]);
```

Dimungkinkan menggabungkan deklarasi dan penciptaan sebagai berikut:

```
int monthDays[] = new() int[12];
```

Penulisan ini lebih ringkas, dan gaya ini yang sering tampil pada program-program Java para profesional yaitu menggabungkan antara deklarasi variabel dan penciptaan ruang memori untuk menampungnya. Array dapat juga diinisialisasi saat array dideklarasikan. Proses ini sangat banyak kesamaannya dengan inisialisasi tipe sederhana. Inisialisasi array adalah daftar ekspresi yang dipisahkan dengan koma diapit oleh kurung kurawal. Koma memisahkan nilai elemen-

elemen array. Array akan secara otomatis diciptakan dengan besar yang mencukupi untuk menampung jumlah yang elemen yang dispesifikasikan di inisialisasi.

Berikut ini adalah penciptaan untuk *array* yang berisi jumlah hari maksimum di masing-masing bulan.

Dengan demikian tidak diperlukan operator **new()**. Berikut ini adalah revisi program **ArraySatuDimensi.java** sehingga lebih ringkas yang dapat dilihat pada praktik 1.

```
class ArrayNextSatuDimensi {
    public static void main(String[] args) {
    // Langkah : deklarasi dan inisialisasi sekaligus juga dilakukan penciptaan
    int monthDays[] = {31,29,31,30,31,30,31,31,30,31,30,31};
    System.out.println("Agustus mempunyai "+monthDays[7]+" hari.");
   }
}
```

Kedua program diatas menghasilkan hasil yang sama. Selanjutnya perhatikan contoh program untuk mencari rerata.

3. Array Dua Dimensi

Array dua dimensi dapat juga disebut sebagai sebuah matriks. Jika array satu dimensi hanya terdiri dari sebuah baris dan beberapa kolom elemen, maka array dua dimensi terdiri dari beberapa baris dan beberapa kolom elemen yang bertipe sama sehingga dapat digambarkan sebagai berikut:

	Column 0	Column 1	Column 2	Column 3
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

Bentuk umum:

<**tipedata**>NamaArray [m][n];

Pendeklarasian array dua dimensi mirip dengan pendeklarasian array satu dimensi, kecuali bahwa array dua dimensi terdapat dua jumlah elemen yang terdapat di dalam kurung siku.

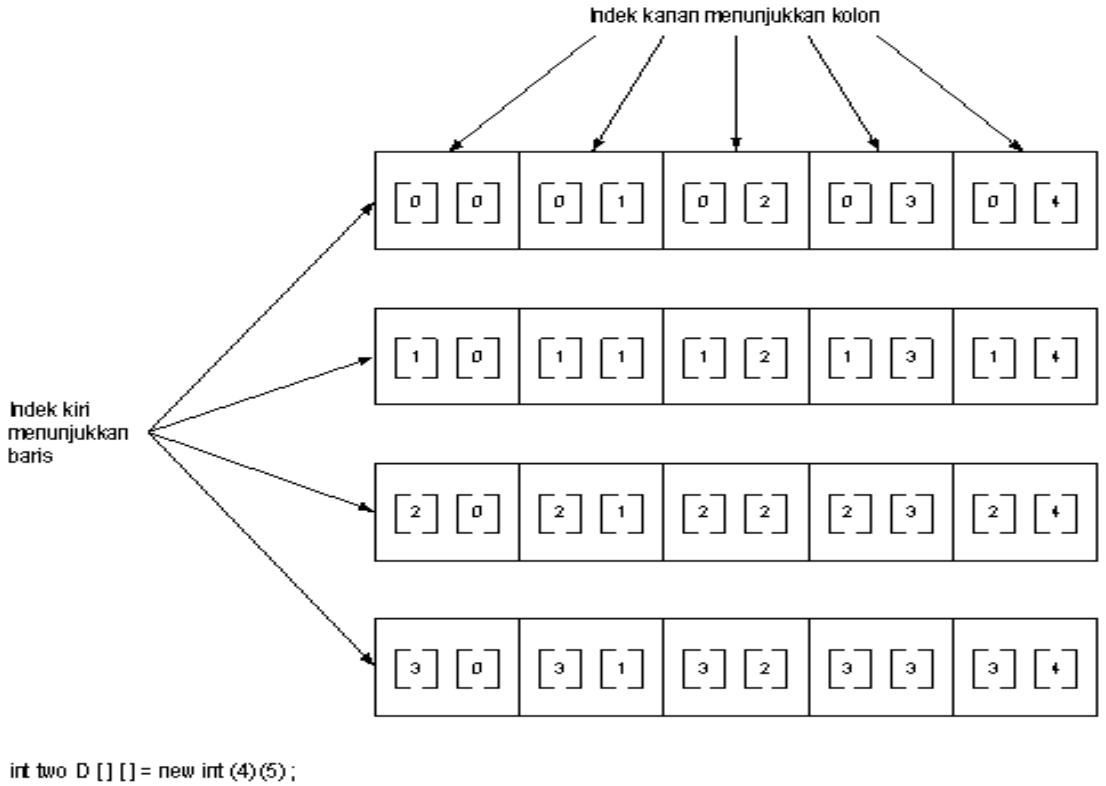
4. Array Multi Dimensi

Di Java, array multidimensi sesungguhnya array dari array. Pendekatan ini terlihat dan bertindak seperti array multidimensi biasa, namun terdapat beberapa perbedaan yang mendasar. Untuk mendeklarasikan variabel array, men-spesifikasikan masing-masing indeks menggunakan himpunan kurung siku yang lain.

Contoh: Array dua dimensi bertipe int bernama intArr dideklarasikan dan diciptakan dengan perintah sebagai berikut:

int DuaDimArr[][] = new() int[4][5];

Perintah ini mendeklarasikan dan mengalokasikan array 4 kali 5 dan memberikan ke variabel DuaDimArr. Secara internal matriks ini diimplementasikan sebagai array dari array bertipe int. Secara konseptual array ini dapat dilihat pada gambar 1.



Gambar 1 Array Multidimensi

Program berikut memberi angka ke masing-masing elemen di array dari kiri ke kanan, puncak ke bawah, kemudian menampilkan masing-masing elemen ini.

Praktik 1: Menggunakan Array Satu Dimensi

```
package halo;

class ArraySatuDimensi {
public static void main(String[] args) {
    // Langkah 1: deklarasi variabel array
    int monthDays[];
    // Langkah 2: penciptaan array bilangan int dengan elemen 12
    monthDays = new int[12];
    monthDays[0] = 31;
    monthDays[1] = 29;
    monthDays[2] = 31;
    monthDays[3] = 30;
    monthDays[4] = 31;
    monthDays[5] = 30;
    monthDays[6] = 31;
    monthDays[7] = 31;
    monthDays[8] = 30;
    monthDays[9] = 31;
    monthDays[10] = 30;
    monthDays[11] = 31;
    System.out.println("Agustus mempunyai "+monthDays[7]+" hari.");
   }
}
```

Praktik 2: Menggunakan Array Satu Dimensi Mencari Rerata

```
package halo;

class ArrayMencariRataan {
public static void main(String[] args) {
        // Langkah 1: deklarasi, penciptaan, inisialisasi variabel array
         double nilai_ujian[] = {20,30,20,30,20};
         double hasil = 0;
         for (int i=0;i<5;i++)
         hasil = hasil + nilai_ujian[i];

         System.out.println("Rataan adalah " +hasil/5);
    }
}
```

Praktik 3: Array matrix 2x2

```
class DemoSimpleMatrix {
    public static void main (String[] args){
        int matrix[][] = new int[2][2];
        matrix[0][0] = 5;
        matrix[0][1] = 0;
        matrix[1][0] = 2;
        matrix[1][1] = 6;

        for (int i = 0; i < matrix.length; i++){
            for (int j = 0; j < matrix[0].length; j++){
                System.out.print(matrix[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

Praktik 4: Menggunakan Array Multidimensi

```
package halo;

class ArrayDuaDimensi {
public static void main(String[] args) {
    int DuaDimArr[][] = new int[4][5];
    int k=0;

    for (int i=0;i<4;i++) {
    for (int j=0;j<5;j++){
    DuaDimArr[i][j] = k++;
          }
      }
    for (int i=0;i<4;i++) {
    for (int j=0;j<5;j++){
    System.out.print(DuaDimArr[i][j] + " ");
         }
    System.out.println();
        }
    }
}
```

Praktik 5: Lengkapi Program Array

Lengkapi potongan source code untuk mengisi array dengan nilai dari user berikut dan buat dengan Java hingga dapan dijalankan

```
….
int[] numbers = new int[100];
   for (int i = 0; i < 3; i++) {
      System.out.print("Angka:");
      numbers[i] = sc.nextInt();
      System.out.println(numbers[i]);
}
…….
```

Tugas :

- Buat program menghitung penjumlahan matrik 2x2.
- Buat program menghitung perkalian matrik 2x2.

MODUL 2: STACK

1. Pengertian Stack

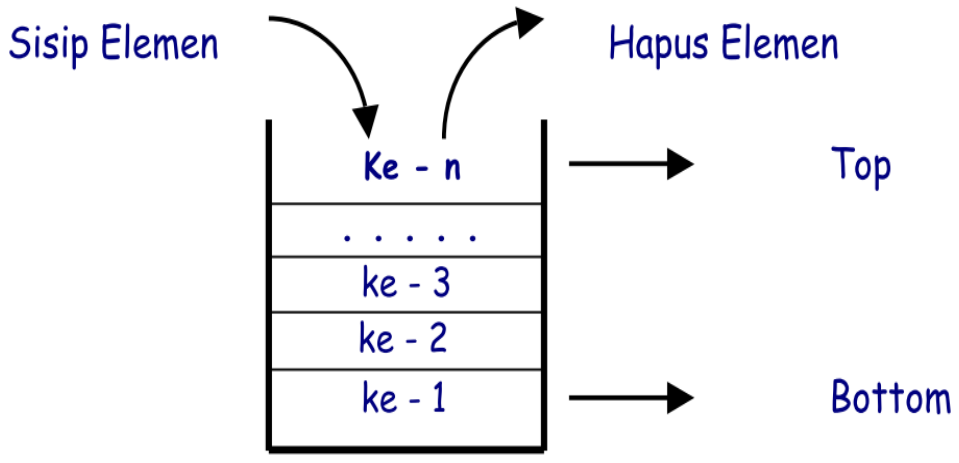
Suatu susunan koleksi data. Data dapat ditambahkan & dihapus selalu dilakukan pada bagian akhir data, yang disebut dengan **top** of stack. Struktur data linier dimana hanya bagian TOP-nya saja yang bisa diakses. Stack bersifat dinamis, artinya item/ data yang tidak terpakai bisa dihapus.

- Top : Ujung list data
- Stack bersifat LIFO = **Last In First Out**
- "Benda yang **terakhir masuk** ke dalam stack akan menjadi yang **pertama keluar** dari stack

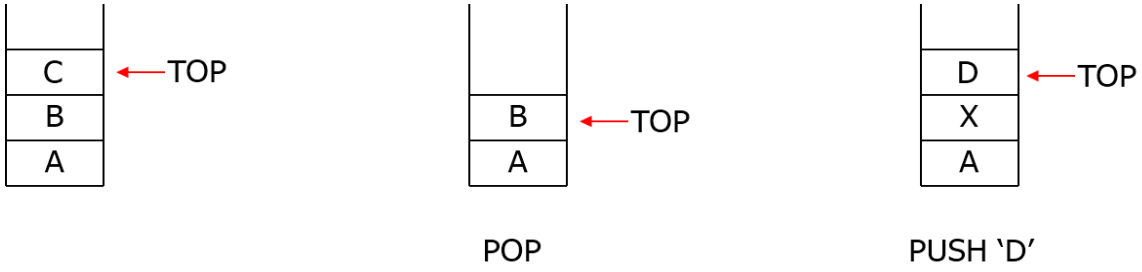
Contoh Stack:

- Piring kotor terakhir diletakkan diposisi paling atas tumpukan. Piring paling atas yang pertama kali dicuci.
- Naik lift

Berikut ilustrasi dari suatu stack!



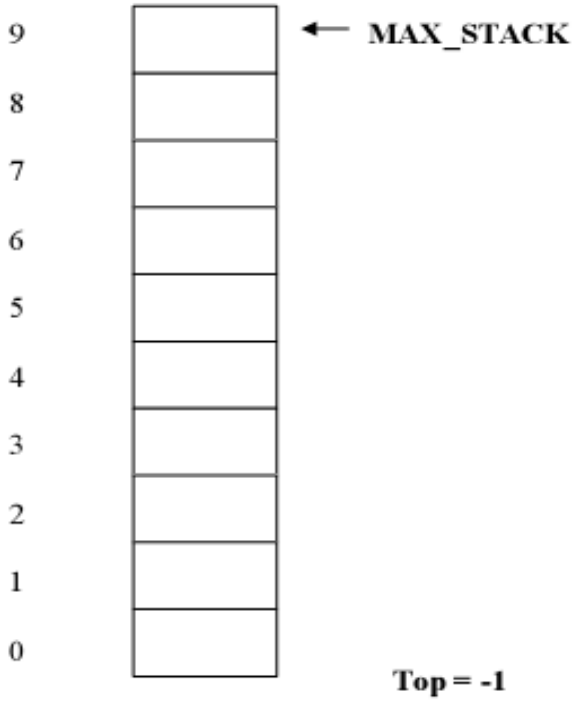
Untuk menambahkan data dalam stak digunakan perintah PUSH, sedangkan untuk mengeluarkan data digunakan POP. Berikut ilustrasi dari PUSH dan POP!



2. Fungsi dalam Stack

- Inisialisasi Stack

  Pada awalnya isi **top** dengan -1, array dimulai dari 0, yang berarti bahwa data stack adalah KOSONG!. **Top** adalah suatu variabel penanda dalam Stack yang menunjukkan elemen teratas data Stack sekarang. **Top Of Stack** akan selalu bergerak hingga mencapai MAX of STACK yang menyebabkan stack PENUH!




```
void inisialisasi(){
     tumpuk.top = -1;
}
```

- **Push :** digunakan untuk menambah item pada stack pada tumpukan paling atas

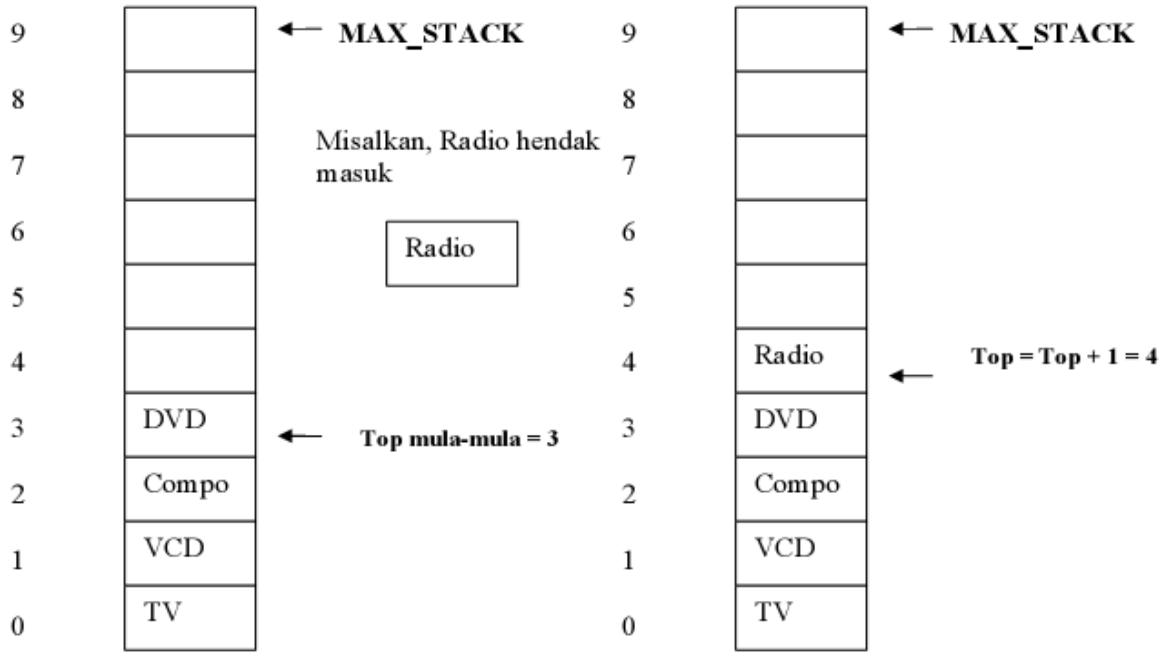
  Fungsi Push berguna untuk memasukkan elemen ke dalam data Stack.

  Data yang diinputkan **selalu** menjadi elemen teratas Stack (yang ditunjuk oleh ToS)

  Jika **data belum penuh**,

- Tambah satu (increment) nilai **top of stack** lebih dahulu setiap kali ada penambahan ke dalam array data Stack.
- Isikan data baru ke stack berdasarkan indeks top of stack yang telah di-increment sebelumnya.

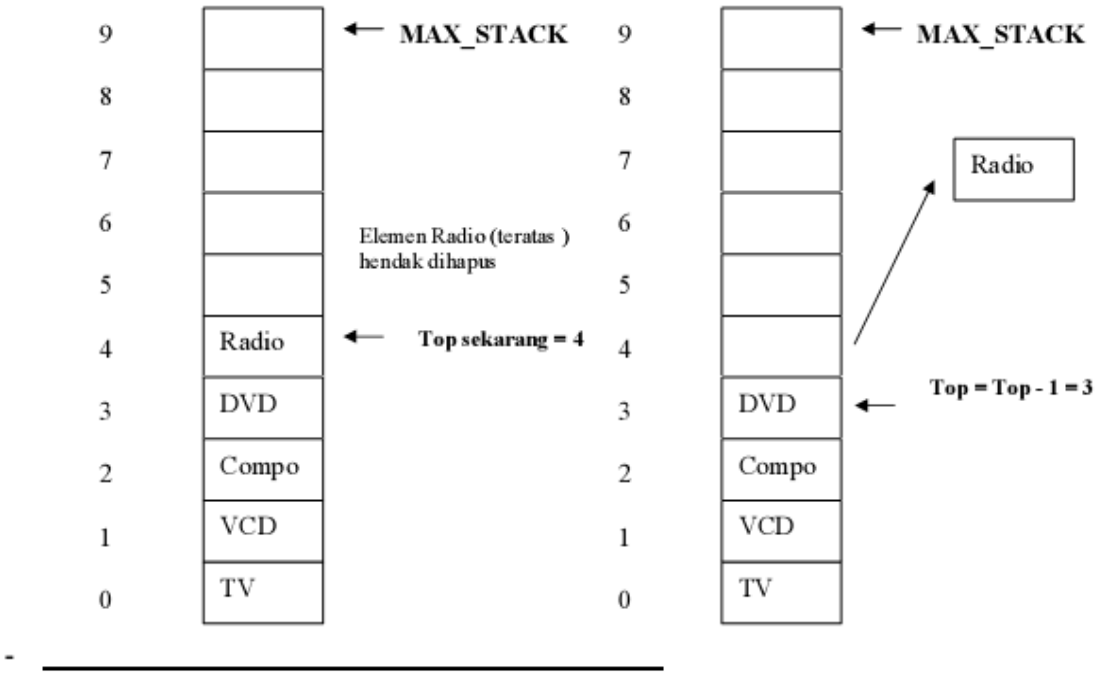
Jika tidak, outputkan "Penuh"




```
void Push(char d[10]){
  tumpuk.top++;
  strcpy(tumpuk.data[tumpuk.top],d);
}
```

- **Pop :** digunakan untuk mengambil item pada stack pada tumpukan paling atas

Fungsi Pop berguna untuk mengambil data Stack yang terletak paling atas (data yang ditunjuk oleh TOS). **Tampilkan terlebih dahulu** nilai elemen teratas stack dengan mengakses indeksnya sesuai dengan top of stacknya, baru dilakukan di-decrement nilai top of stacknya sehingga jumlah elemen stack berkurang.



Programnya:

```
void Pop(){
   printf("Data yang terambil = %s\n",tumpuk.data[tumpuk.top]);
   tumpuk.top--;
}
```

- **Clear :** digunakan untuk mengosongkan stack
- **IsEmpty :** fungsi yang digunakan untuk mengecek apakah stack sudah kosong

  Fungsi IsEmpty berguna untuk memeriksa apakah data Stack masih kosong? Dengan cara memeriksa **top of stack**, jika masih -1 maka berarti data Stack masih kosong!

```
int IsEmpty(){
  if(tumpuk.top == -1)
      return 1;
  else
      return 0;
}
```

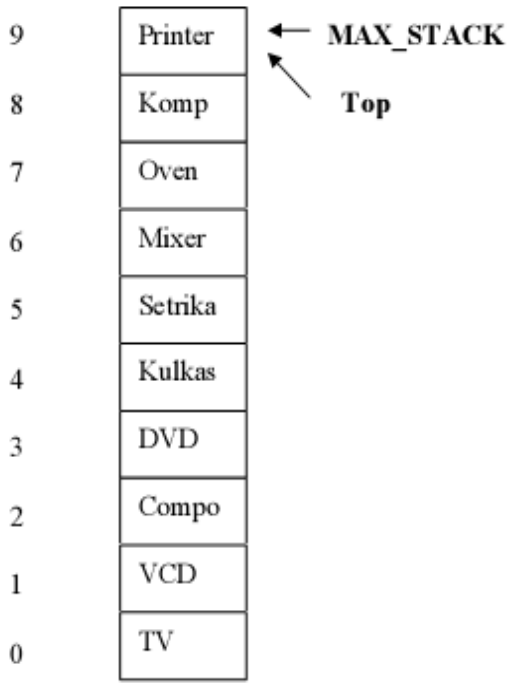
- **IsFull :** fungsi yang digunakan untuk mengecek apakah stack sudah penuh

  Fungsi IsFull berguna untuk mengetahui apakah suatu stack sudah terisi penuh?

  Dengan cara memeriksa **top of stack**,

  jika sudah sama dengan MAX_STACK-1 maka **full**,

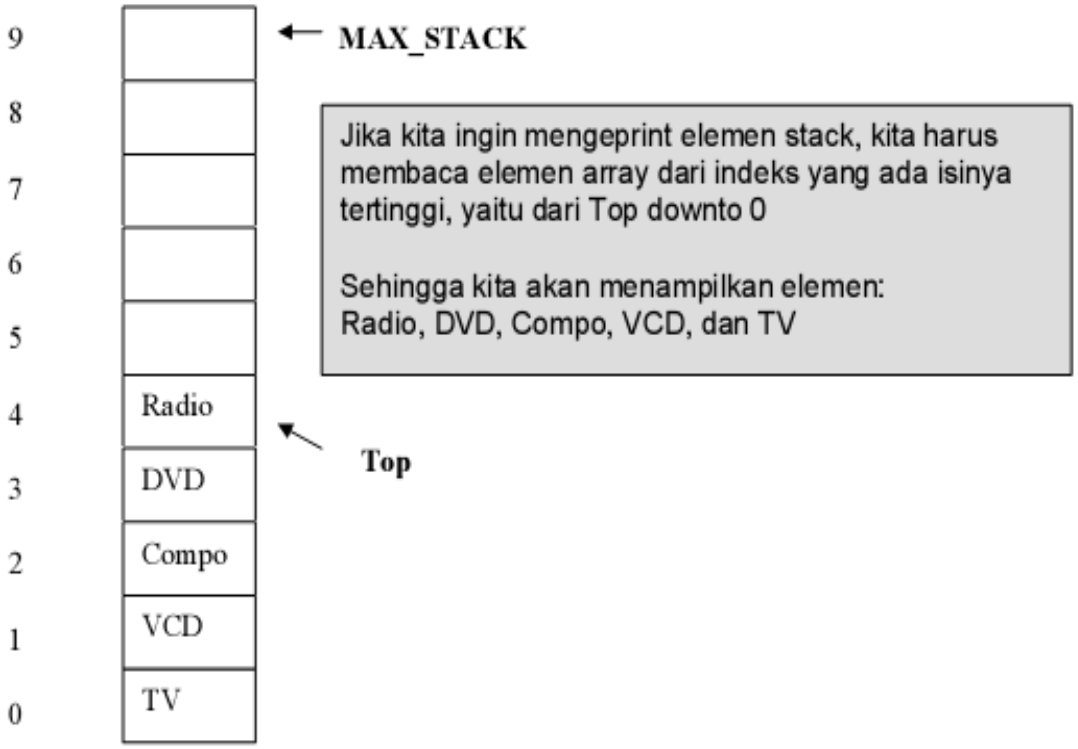
  jika belum (masih **lebih kecil** dari MAX_STACK-1) maka belum full




```
int IsFull(){
      if(tumpuk.top == MAX_STACK-1)
            return 1;
      else
            return 0;
}
```

- **Print:**

  Untuk menampilkan semua elemen-elemen data Stack. Dengan cara me-loop semua nilai array secara **terbalik**, karena kita harus mengakses dari indeks array tertinggi terlebih dahulu baru ke indeks yang lebih kecil!



- **Peek:**

  Peek, digunakan untuk melihat data yang berada di posisi paling atas

Praktik 6: Stack-Push (Menambahkan String)

```
package demostack_push;
import java.util.*;
public class DemoStack_push {
   public static void main(String[] args)  {
      // Creating an empty Stack
      Stack<String> STACK = new Stack<String>();

      // Use push() to add elements into the Stack
      STACK.push("Minggu");
      STACK.push("Senin");
      STACK.push("Selasa");
      STACK.push("Rabu");
      STACK.push("Kamis");

      // Displaying the Stack
      System.out.println("Stack Awal: " + STACK);

      // Pushing elements into the stack
      STACK.push("Jumat");
      STACK.push("Sabtu");

      // Displaying the final Stack
      System.out.println("Updata Stack: " + STACK);
   }
}
//source: https://www.geeksforgeeks.org/
```

Praktik 7: Stack-Push (Menambahkan Integer)

```
package demostackpushint;
import java.util.*;
public class DemoStackPushInt {
   public static void main(String args[])
    {
        // Creating an empty Stack
        Stack<Integer> STACK = new Stack<Integer>();

        // Use push() to add elements into the Stack
        STACK.push(10);
        STACK.push(15);
        STACK.push(30);
        STACK.push(20);
        STACK.push(5);

        // Displaying the Stack
        System.out.println("Stack Awal: " + STACK);

        // Pushing elements into the Stack
        STACK.push(1254);
        STACK.push(4521);

        // Displaying the final Stack
        System.out.println("Update Stack: " + STACK);
    }
}
//source:https://www.geeksforgeeks.org/
```

Praktik 8: Stack-Pop (Mengeluakan String)

```
package demostackpopstring;
import java.util.*;
public class DemoStackPopString {
   public static void main(String[] args) {
       // Creating an empty Stack
      Stack<String> STACK = new Stack<String>();

      STACK.push("Minggu");
      STACK.push("Senin");
      STACK.push("Selasa");
      STACK.push("Rabu");
      STACK.push("Kamis");

      // Displaying the Stack
      System.out.println("Stack Awal: " + STACK);

      // Removing elements using pop() method
      System.out.println("Pop Element: " +
                                 STACK.pop());
      System.out.println("Pop Element: " +
                                 STACK.pop());

      // Displaying the Stack after pop operation
      System.out.println("Update Stack "
                                    + STACK);
   }

}
//source: https://www.geeksforgeeks.org
```

Praktik 9: Stack-Pop (Mengeluakan Intenger)

```
package demostackpopint;
import java.util.*;
public class DemoStackPopInt {
   public static void main(String[] args) {
     // Creating an empty Stack
      Stack<Integer> STACK = new Stack<Integer>();

      STACK.push(10);
      STACK.push(15);
      STACK.push(30);
      STACK.push(20);
      STACK.push(5);

      // Displaying the Stack
      System.out.println("Initial Stack: " + STACK);

      // Removing elements using pop() method
      System.out.println("Pop elemen: " +
                                 STACK.pop());
      System.out.println("Pop elemen: " +
                                 STACK.pop());

      // Displaying the Stack after pop operation
      System.out.println("Update Stack "
                                    + STACK);
   }

}
//source:www.geeksforgeeks.org
```

Latihan Modul Stack

Ketik program berikut ini, amati dan pahami program dan outputnya.

```
package stackimplement;
import java.util.*;

/*  Class arrayStack  */
class arrayStack
{
   protected int arr[];
   protected int top, size, len;
   /*  Constructor for arrayStack */
   public arrayStack(int n)
   {
      size = n;
      len = 0;
      arr = new int[size];
      top = -1;
   }
   /*  Function to check if stack is empty */
   public boolean isEmpty()
   {
      return top == -1;
   }
   /*  Function to check if stack is full */
   public boolean isFull()
   {
      return top == size -1 ;
   }
   /*  Function to get the size of the stack */
   public int getSize()
   {
      return len ;
   }
   /*  Function to check the top element of the stack */
   public int peek()
   {
      if( isEmpty() )
         throw new NoSuchElementException("Underflow Exception");
      return arr[top];
   }
   /*  Function to add an element to the stack */
   public void push(int i)
   {
      if(top + 1 >= size)
         throw new IndexOutOfBoundsException("Overflow Exception");
```

```
        if(top + 1 < size )
            arr[++top] = i;
        len++ ;
    }
    /*  Function to delete an element from the stack */
    public int pop()
    {
        if( isEmpty() )
            throw new NoSuchElementException("Underflow Exception");
        len-- ;
        return arr[top--];
    }
    /*  Function to display the status of the stack */
    public void display()
    {
        System.out.print("\nStack = ");
        if (len == 0)
        {
            System.out.print("Empty\n");
            return ;
        }
        for (int i = top; i >= 0; i--)
            System.out.print(arr[i]+" ");
        System.out.println();
    }
}

/*  Class StackImplement  */
public class StackImplement
{
    public static void main(String[] args)
    {
        Scanner scan = new Scanner(System.in);
        System.out.println("Stack Test\n");
        System.out.println("Enter Size of Integer Stack ");
        int n = scan.nextInt();
        /* Creating object of class arrayStack */
        arrayStack stk = new arrayStack(n);
        /* Perform Stack Operations */
        char ch;
        do{
            System.out.println("\nStack Operations");
            System.out.println("1. push");
            System.out.println("2. pop");
            System.out.println("3. peek");
            System.out.println("4. check empty");
```

```
        System.out.println("5. check full");
        System.out.println("6. size");
        int choice = scan.nextInt();
        switch (choice)
        {
        case 1 :
           System.out.println("Enter integer element to push");
           try
           {
              stk.push( scan.nextInt() );
           }
           catch (Exception e)
           {
              System.out.println("Error : " + e.getMessage());
           }
           break;
        case 2 :
           try
           {
              System.out.println("Popped Element = " + stk.pop());
           }
           catch (Exception e)
           {
              System.out.println("Error : " + e.getMessage());
           }
           break;
        case 3 :
           try
           {
              System.out.println("Peek Element = " + stk.peek());
           }
           catch (Exception e)
           {
              System.out.println("Error : " + e.getMessage());
           }
           break;
        case 4 :
           System.out.println("Empty status = " + stk.isEmpty());
           break;
        case 5 :
           System.out.println("Full status = " + stk.isFull());
           break;
        case 6 :
           System.out.println("Size = " + stk.getSize());
           break;
        default :
```

```
                System.out.println("Wrong Entry \n ");
                break;
            }
            /* display stack */
            stk.display();
            System.out.println("\nDo you want to continue (Type y or n) \n");
            ch = scan.next().charAt(0);

        } while (ch == 'Y'|| ch == 'y');
    }
}
//source: https://www.sanfoundry.com/
```

MODUL 3: QUEUE

1. Pengertian Queue

Queue atau bisa disebut juga Antrian adalah list linier yang berfungsi untuk menyimpan informasi seperti stack. Karakteristiknya bersifat FIFO (First In First Out). Elemen yang **pertama masuk** ke antrian akan **keluar pertama** kalinya.

Contohnya:

- Antrian dalam dunia nyata
- Antrian SPBU.
- Antrian pembelian tiket.
- Antrian nasabah bank
- Antrian pengembalian formulir

Konsep dasar dari Queue adalah terdapat satu buah pintu masuk di suatu ujung dan satu buah pintu keluar di ujung satunya. Sehingga membutuhkan 2 elemen: Head dan Tail.

Queue memiliki 2 element utama:

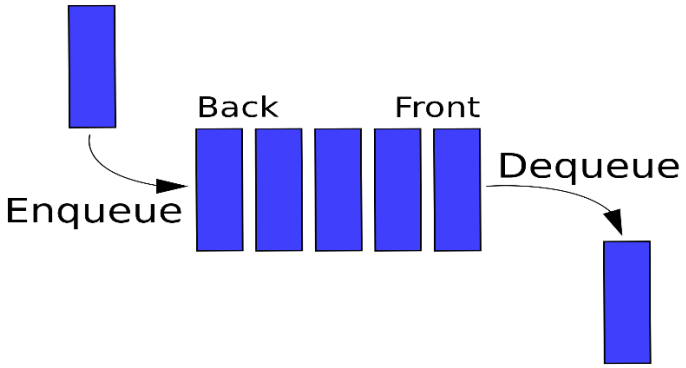
- Head element : elemen pertama
- Tail element : elemen terakhir

Queue memiliki 2 fungsi utama:

- Enqueue : menambah item/data pada tail elemet
- Dequeue : menghapus data pada head element

Aturan penyisipan dan penghapusan elemennya disefinisikan sebagai berikut :

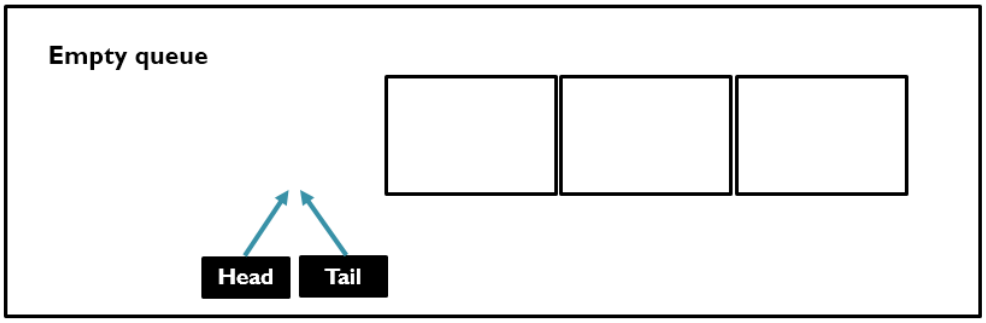
- Penambahan selalu dilakukan setelah elemen terakhir
- Penghapusan selalu dilakukan pada elemen pertama



2. Fungsi dalam Queue

Untuk menciptakan dan menginisialisasi Queue dengan cara membuat Head dan Tail = -1. Berikut algoritma dari inisialisasi queue awal.

```
void initial(){
   queue.head  = -1;
   queue.tail    = -1;
}
```



Menambah Elemen queue

- **enqueue()**
    - Untuk menambahkan elemen ke dalam queue
    - penambahan elemen selalu ditambahkan di elemen paling **belakang**

Penambahan elemen selalu menggerakan variabel Tail dengan cara increment counter Tail terlebih dahulu

Kode:

```
void enqueue(int data){
  if(isEmpty()==1){
     queue.head=queue.tail=0;
     queue.data[queue.tail]=data;
     printf("%d success enqueu !\n", queue.data[queue.tail]);
  }
else{
     if(isFull()==0){
```

```
        queue.tail++;
        queue.data[queue.tail]=data;
        printf("%d success enqueue !\n", queue.data[queue.tail]);
    }
    else{
        printf("Queue is full\n");
    }
  }
}
```

Menghapus Elemen Queue

- **dequeue()**
    - Digunakan untuk menghapus elemen terdepan/pertama (head) dari queue
    - Dengan cara menggeser semua elemen queue kedepan dan mengurangi Tail dgn 1
    - Penggeseran dilakukan dengan menggunakan looping

Kode:

```
int dequeue(){
        int e;
        if(isEmpty()==0){
                e=queue.data[queue.head];
                queue.head++;
                return e;
        }
        else{
                printf(“Queue is empty\n”);
        }
}
```

Clear Queue

- **Clear()**
    - Untuk menghapus elemen-elemen queue dengan cara membuat Tail dan Head = -1
    - Penghapusan elemen-elemen queue sebenarnya tidak menghapus arraynya, namun hanya mengeset indeks pengaksesan-nya ke nilai -1 sehingga elemen-elemen queue tidak lagi terbaca

Kode:

```
void clear(){
   if(isEmpty()==0){
      queue.head=-1;
         queue.tail=-1;
      printf("Now queue is empty");
   }
         else{
                  printf("Queue was empty");
         }
```

Check Queue Empty or not

- **IsEmpty()**
    - Untuk memeriksa apakah Antrian sudah penuh atau belum
    - Dengan cara memeriksa nilai head atau tail,
        - jika Head = -1 maka empty
        - jika Tail = -1 maka empty
    - Kita tidak memeriksa Head, karena Head adalah tanda untuk kepala antrian (elemen pertama dalam antrian) yang tidak akan berubah-ubah
    - Pergerakan pada Antrian terjadi dengan penambahan elemen Antrian kebelakang, yaitu menggunakan nilai Tail

Kode dari isEmpty() dapat berupa:

```
int isEmpty(){
   if(queue.head==-1)
      return 1;
   else
      return 0;
}
```

Check Queue Full or not

- **IsFull()**
    - Untuk mengecek apakah queue sudah penuh atau belum
    - Dengan cara mengecek nilai Tail,
        - jika Tail = MAX-1 (karena MAX-1 adalah batas elemen array pada C) berarti sudah penuh

Kode isFull():

```
int isFull(){    if(queue.tail==MAX-1)
      return 1;
   else
      return 0;
}
```

Print Elemen Queue

- **print()**
    - Untuk menampilkan nilai-nilai elemen queue
    - Menggunakan looping dari head s/d tail

Kode:

```
void show(){
   int i;
```

```
        if(isEmpty()==0){
                for(i=queue.head;i<=queue.tail;i++){
                printf("%d",queue.data[i]);
                }printf("\n");
        }
        else{
                printf("Queue is still empty");
        }
}
```

Praktik 10 : Antrian

```
package queue;
public class Queue {

 // Variabel Internal
 private int head = 0;
 private int tail = 0;
 private int[] isi = new int[4];

 // Kontruktor
 public Queue() {}

 public int GetIsiByIndex(int index)
 {
  return isi[index];
 }

 // Enqueue
 public void Enqueue(int x)
 {
  if(tail < 4) {
   isi[tail] = x;
   tail++;
  } else {
   System.out.println("Penuh");
  }
 }

 // Dequeue
 public int Dequeue()
```

```
{
 if(tail != head) {
  int temp = isi[head];
  for(int a = head; a < tail; a++) {
   if(a == tail-1) {
    isi[a] = 0;
   } else {
    isi[a] = isi[a+1];
   }
  }
  tail--;
  return temp;
 } else {
  System.out.println("Kosong");
  return 0;
 }
}

// Print
public void Print()
{
 String teks = "";
 for(int a=0; a<4; a++) {
  teks += isi[a]+",";
 }
 System.out.println(teks);
}

public static void main(String[] args) {

 // Buat queue
 Queue q1 = new Queue();

 // Enqueue
 q1.Enqueue(31);
 q1.Enqueue(32);
 q1.Enqueue(33);
 q1.Enqueue(34);
 q1.Enqueue(35);
 q1.Enqueue(36);
 q1.Print();

 System.out.println(q1.Dequeue());
 q1.Print();
 System.out.println(q1.Dequeue());
 q1.Print();
```

```
  System.out.println(q1.Dequeue());
  q1.Print();
  System.out.println(q1.Dequeue());
  q1.Print();
  System.out.println(q1.Dequeue());
  q1.Print();
 }

}
//source: http://bobby-syahronanda.blogspot.com/
```

Praktik 11: Antrian2

```
public class Antrian2 {

 static int ekor = -1;
 static int maxEkor = 5;
 public static void addQueue(String antrian[], String data)
 { if (ekor >= maxEkor)
   System.out.println("Maaf, antrian penuh");
  else
  { //System.out.println("add Queue: "+data);
   ekor++;
   antrian[ekor]=data;
  }
 }
 public static String deQueue(String antrian[])
 { String hasil="";
  if (ekor<0)
   hasil="Maaf, antrian kosong";
  else
  { hasil = antrian[0];
   antrian[0]=null;
   for (int i=1;i<=ekor;i++)
   {
    antrian[i-1]=antrian[i];
    antrian[i]=null;
   }
   ekor--;
  }
  return (hasil);
 }

 public static void bacaAntrian(String antrian[])
 { System.out.println("Kondisi Antrian : ");
  for (int i=0;i<=maxEkor;i++)
  {
   if (i==ekor)
    System.out.println(i+". "+ antrian[i]+" ekor");
   else if (antrian[i]==null)
    System.out.println(i+". ");
   else
    System.out.println(i+". "+antrian[i]);
  }
 }
 public static void main(String []wawan)
 { String antrian[]=new String [16];
     System.out.println("Antrian Mobil");
```

```
        addQueue (antrian, "Mobil A");
        addQueue (antrian, "Mobil B");
        addQueue (antrian, "Mobil C");
        addQueue (antrian, "Mobil D");
        addQueue (antrian, "Mobil E");
        addQueue (antrian, "Mobil F");
     System.out.println("\n");
    System.out.println(" ");
  bacaAntrian(antrian);
    System.out.println("\n\nURUTAN KELUAR\nQueue : "+deQueue (antrian));
    System.out.println("Queue : "+deQueue (antrian));
    System.out.println("Queue : "+deQueue (antrian));
    System.out.println("Queue : "+deQueue (antrian));
    System.out.println("Queue : "+deQueue (antrian));
    System.out.println ("\n\nSedangkan Mobil F masih mengantri");
  }
//source:http://jemepatas.blogspot.com
```

Latihan Modul Queue: Antrian Bank

Ketik program berikut ini, amati dan pahami program dan outputnya.

```
package antrianbank;
import java.util.Scanner;

public class AntrianBank {
    int id;
    String nama,perlu;
    AntrianBank next;
    static Scanner in=new Scanner(System.in);
    static Scanner str=new Scanner(System.in);
    public void input(){
        System.out.print("Masukkan id (angka): ");
        id=in.nextInt();
        System.out.print("Masukkan nama      : ");
        nama=str.nextLine();
        System.out.print("Masukkan keperluan : ");
        perlu=str.nextLine();
        next=null;
    }

    public void read(){
        System.out.println("|| "+id+" \t|| "+nama+" \t|| "+perlu+" \t||");
    }

    public static void main(String[] args) {
        int menu=0;
        linked que=new linked();
        while(menu!=4){
            System.out.print("1.Enqueue\n2.Dequeue\n3.View\n4.Exit\n : ");
            menu=in.nextInt();
            if(menu==1)que.enque();
            else if(menu==2)que.deque();
            else if(menu==3)que.view();
            else if(menu==4)System.out.println("- keluar -");
            else System.out.println("- Salah -");
            System.out.println("");
        }
    }
    }
}

class linked{
    AntrianBank head,tail;
    public linked(){
```

```
        head=null;
        tail=null;
    }
    public void enque(){
        AntrianBank baru=new AntrianBank();
        baru.input();
        if(head==null)head=baru;
        else tail.next=baru;
        tail=baru;
    }
    public void deque(){
        if(head==null)System.out.println("- Kosong -");
        else{
            System.out.println("Keluar Data Dengan Id : "+head.id);
            head=head.next;
        }
    }
    public void view(){
        if(head==null)System.out.println("- Kosong -");
        else{
            System.out.println("|| Id \t|| Nama \t|| Keperluan \t||");
            for(AntrianBank a=head; a!=null; a=a.next) a.read();
        }
    }
}
//source: https://safriansah.blogspot.com
```

MODUL 4: LINKED LIST

Misalkan ada 2500 item yang merupakan elemen dari suatu linear list. Jika elemen ke-56 akan kita keluarkan, maka elemen ke-1 s/d elemen ke-55 tidak akan berubah posisinya pada linear list tersebut. Tetapi elemen ke-57 akan menjadi elemen ke-56, elemen ke-58 akan menjadi elemen ke-57 dst.

1	2	3	4	5	...	...	...	2499	2500

Selanjutnya, jika kita sisipkan satu elemen pada posisi setelah elemen ke-41, maka elemen ke-42 s/d elemen ke-2500 akan berubah posisinya.

Untuk menyatakan keadaan diatas diperlukan suatu konsep yang berbeda dengan konsep sekuensial sebelumnya.

Linked list merupakan suatu cara non-sekuensial yang digunakan untuk merepresentasikan suatu data.

Mengapa menggunakan linked list?

1. Mudah dalam menambahkan dan menghapus elemen（pada array tidak mungkin menambahkan elemen, karena banyaknya elemen sudah ditentukan dari awal）
2. Panjang list bisa diubah dengan bebas (panjang array fixed)
3. Mudah untuk menyambungkan beberapa list, maupun memutuskannya (array tidak bisa)
4. Memungkinkan user mendesain struktur data yang kompleks

Apa itu linked list?

Struktur data yang terdiri dari beberapa simpul (node) dimana setiap simpulnya saling terhubung (linked). Untuk menghubungkan node satu dg lainnya digunakan ***pointer***.

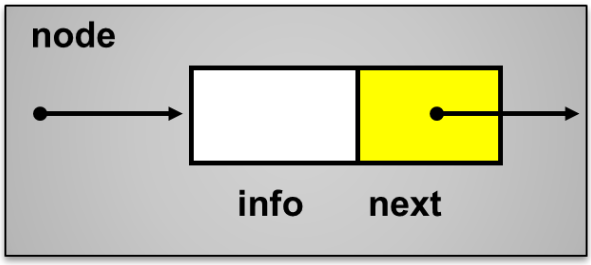
- Menyimpan koleksi elemen secara non contiguously

Elemen dapat terletak pada lokasi memory yang saling berjauhan. Sedangkan array dimana tiap-tiap elemen akan terletak pada lokasi memory yang berurutan.

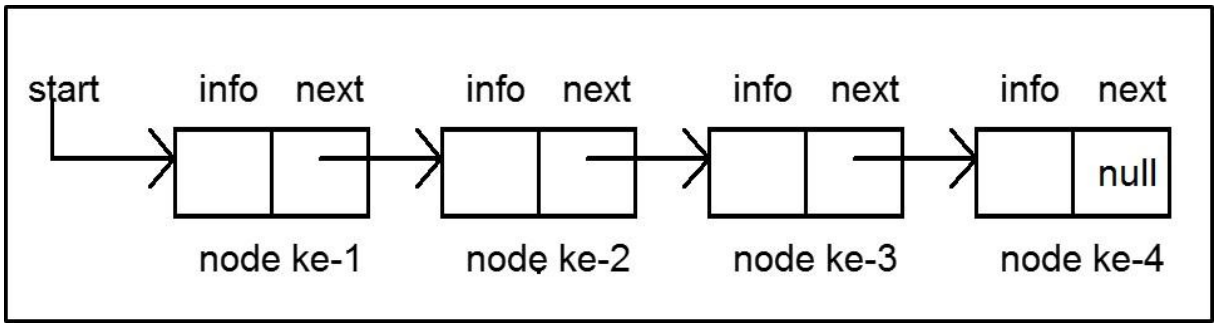
Struktur linked list?

- Setiap elemen (node) dari suatu linked list terdiri atas dua bagian, yaitu :
    - INFO berisi informasi tentang elemen data yang bersangkutan.

- NEXT (link field/next pointer field) berisi alamat dari elemen (node) selanjutnya yang dituju.



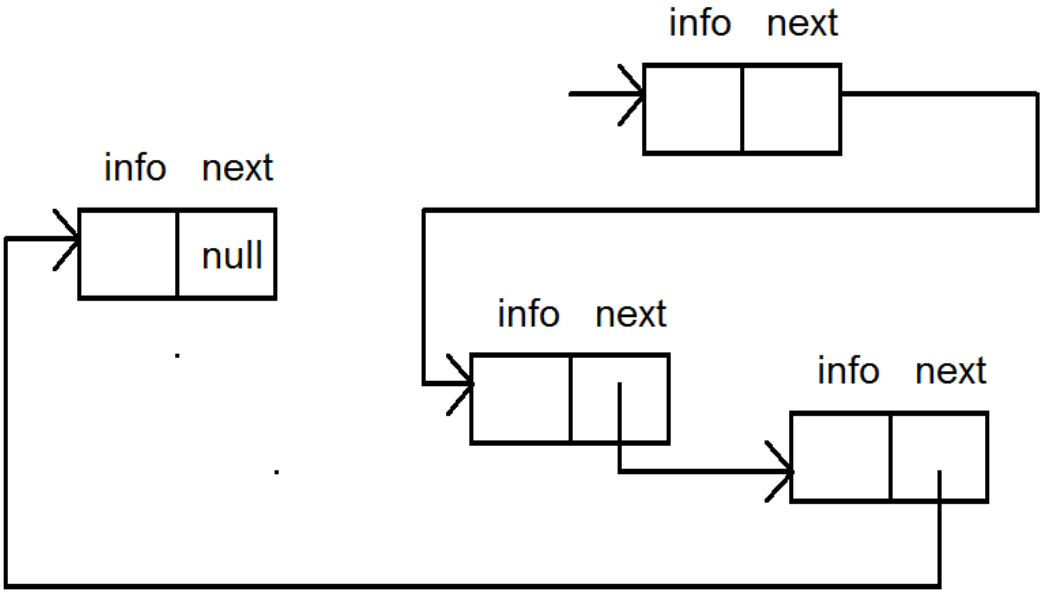
1. Elemen pertamanya, First (head)
2. Alamat element berikutnya yang dapat diakses melalui info NEXT.
3. Setiap elemen memiliki alamat(address)
4. Elemen terakhirnya



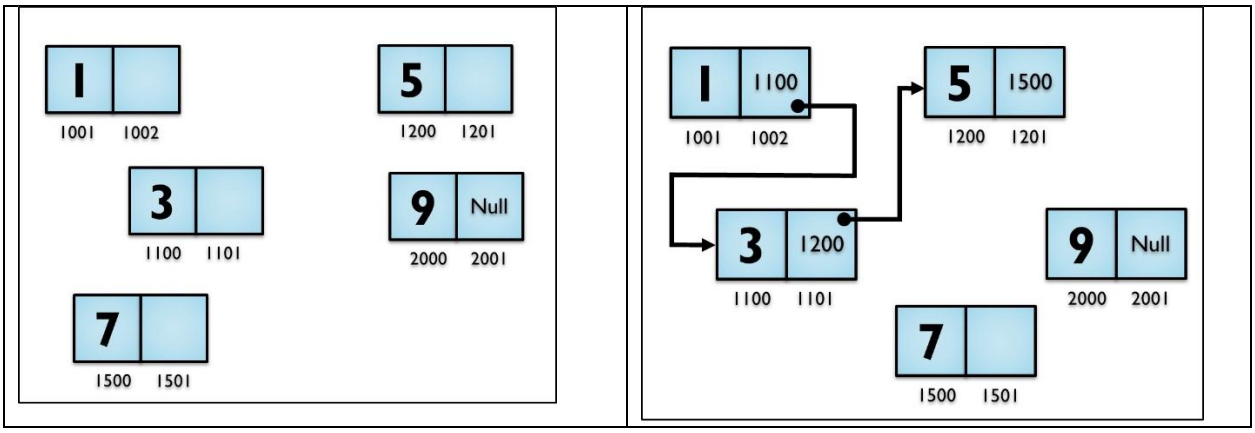
- Pada node ke-4 field NEXT-nya berisi NULL, artinya node ke-4 tsb. adalah node terakhir.
- Node-node dalam linked list tidak harus selalu digambarkan paralel seperti pada gambar diatas

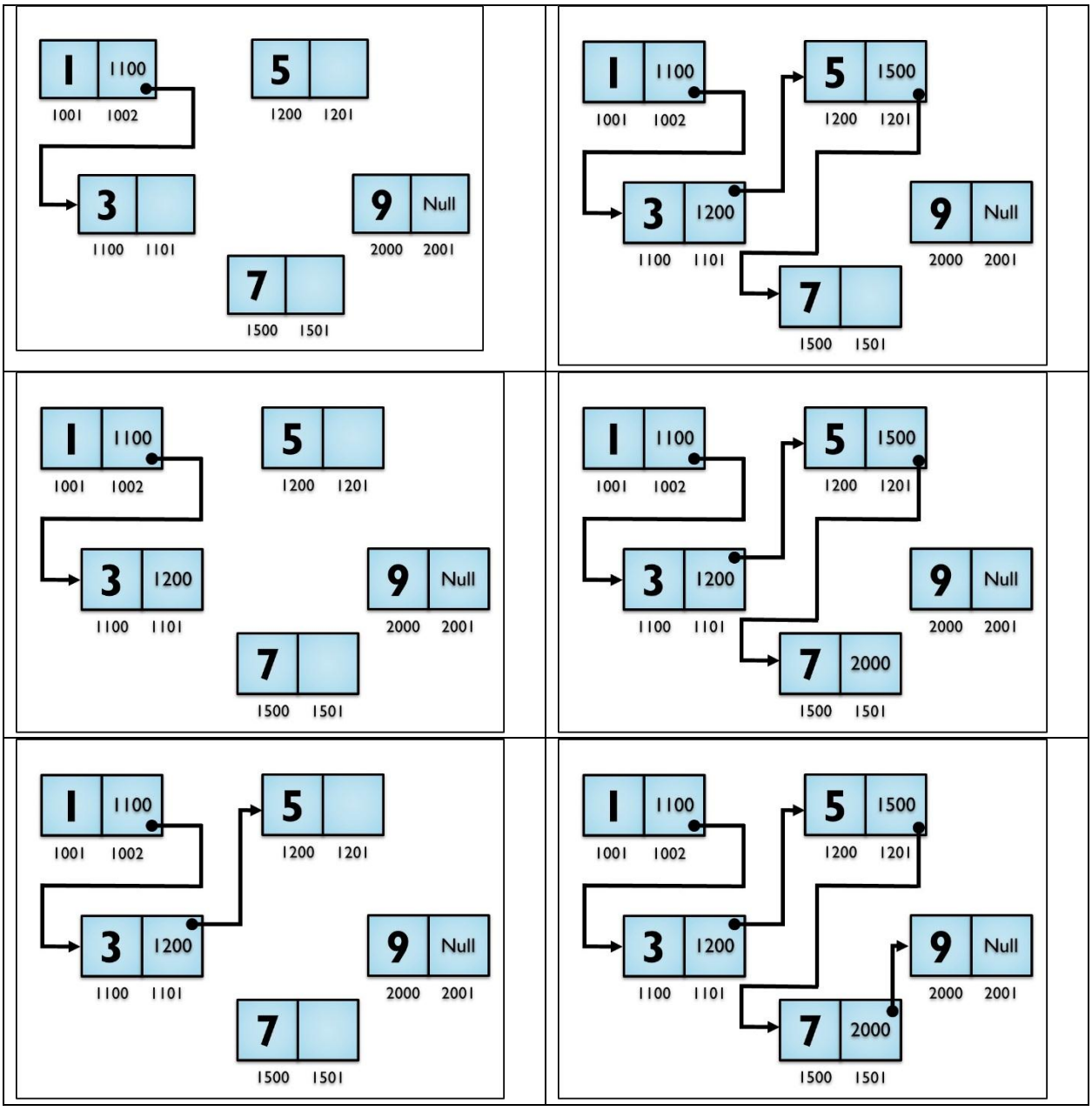
- Ada 2 hal yang menjadi kerugian representasi suatu data dengan linked list ini, yaitu :
  - Diperlukan ruang tambahan untuk menyatakan/ tempat field pointer.
  - Diperlukan waktu yang lebih banyak untuk mencari suatu node dalam linked list.

- Sedangkan keuntungannya adalah :
- Jenis data yang berbeda dapat di-link.
- Operasi REMOVE atau INSERT hanya dilakukan dengan mengubah pointer-nya saja.



2. Contoh Linked List




1
1100
1001
1002
5
1200
1201
3
1100
1101
9
Null
2000
2001
7
1500
1501
1
1100
1001
1002
5
1500
1200
1201
3
1200
1100
1101
9
Null
2000
2001
7
1500
1501
1
1100
1001
1002
5
1200
1201
3
1200
1100
1101
9
Null
2000
2001
7
1500
1501
1
1100
1001
1002
5
1500
1200
1201
3
1200
1100
1101
9
Null
2000
2001
7
2000
1500
1501
1
1100
1001
1002
5
1200
1201
3
1200
1100
1101
9
Null
2000
2001
7
1500
1501
1
1100
1001
1002
5
1500
1200
1201
3
1200
1100
1101
9
Null
2000
2001
7
2000
1500
1501

Praktik 12 : Linked List 01

```
package contohlinkedlist;
/**
  * sumber: https://www.geeksforgeeks.org/linked-list-in-java/
 */

import java.util.*;

public class ContohLinkedList {
    public static void main(String[] args){
// Membuat objek yang bernama daftar pada class LinkedList
        LinkedList<String> daftar = new LinkedList<String>();
         // Menambahkan elemen pada linked list
        daftar.add("A");
        daftar.add("B");
        daftar.addLast("C");
        daftar.addFirst("D");
        daftar.add(3, "B");
        daftar.add("F");
        daftar.add("G");
        System.out.println("Linked list : " + daftar);
         // Menghapus elemen dari linked list
        daftar.remove("B");
        daftar.remove(3);
        daftar.removeFirst();
        daftar.removeLast();
        System.out.println("Linked list setelah dihapus: " + daftar);
         // Menemukan elemen pada linked list
        boolean status = daftar.contains("E");
         if(status)
            System.out.println("Di List terdapat elemen 'E' ");
        else
            System.out.println("Di List tidak terdapat elemen 'E'");

        // ukuran dari elemen pada linked list
        int size = daftar.size();
        System.out.println("Ukuran dari linked list = " + size);

        // Menunjuk dan memberi nilai elemen dari linked list
        Object element = daftar.get(2);
        System.out.println("elemen yang ditunjuk oleh get() : " + element);
        daftar.set(2, "Y");
        System.out.println("Linked list pasca perubahan: " + daftar);
    }
}
```

Praktik 13 : Linked List 02

```
import java.util.LinkedList;
public class ContohSenarai2 {
   public static void main(String[] args) {
      LinkedList<Simpul> senarai = new LinkedList<Simpul>();
      Simpul simpul;

      // Masukkan 5 nama
      senarai.addFirst(new Simpul("AMN", "Aminudin"));
      senarai.addFirst(new Simpul("ZAS", "Zaskia"));
      senarai.addFirst(new Simpul("RIN", "Rina Melati"));
      senarai.addFirst(new Simpul("FAR", "Farhan"));
      senarai.addFirst(new Simpul("AGN", "Agnes Monica"));

      System.out.println("Keadaan awal:");
      for (int i = 0; i < senarai.size(); i++) {
         simpul = senarai.get(i);
         simpul.display();
      }
      hapus(senarai, "RIN");
      System.out.println();
      System.out.println("Setelah RIN dihapus:");
      for (int i = 0; i < senarai.size(); i++) {
         simpul = senarai.get(i);
         simpul.display();
      }
      // Cari RIN
      String dicari = "RIN";
      System.out.println();
      System.out.println("Pencarian " + dicari);
      Simpul posisiData =  cari(senarai, dicari);
      if (posisiData == null)
         System.out.println(dicari + " tidak ditemukan.");
      else {
         System.out.println("Hasil pencarian:");
         posisiData.display();
      }
      // Cari FAR
      dicari = "FAR";
      System.out.println();
      System.out.println("Pencarian " + dicari);
      posisiData =  cari(senarai, dicari);
      if (posisiData == null)
         System.out.println(dicari + " tidak ditemukan.");
      else {
         System.out.println("Hasil pencarian:");
```

```
            posisiData.display();
        }
    }
        // Untuk menghapus data
    public static void hapus(LinkedList<Simpul> senarai,
                             String x) {
        int posisi = -1;
        for (int i = 0; i < senarai.size(); i++) {
            Simpul simpul = senarai.get(i);
            if (simpul.kode.compareTo(x) == 0) {
                posisi = i;
                break;
            }
        }
      if (posisi != -1)
            senarai.remove(posisi);
        else
            System.out.println(x + " tidak dapat dihapus.");
    }
        // Untuk mencari data berdasarkan kode
    public static Simpul cari(LinkedList<Simpul> senarai,
            String x) {
        int posisi = -1;
        for (int i = 0; i < senarai.size(); i++) {
            Simpul simpul = senarai.get(i);
            if (simpul.kode.compareTo(x) == 0) {
                posisi = i;
                break;
            }
        }
        if (posisi == -1)
            return null;
        else
            return senarai.get(posisi);
    }
}

class Simpul {
    public String kode;
    public String nama;

    public Simpul(String xkode, String xnama) {
        kode = xkode;
        nama = xnama;
    }
```

MODUL 5: BUBBLE SORT

1. Pengertian Sorting

Sorting adalah proses menyusun kembali data yang sebelumnya telah disusun dengan suatu pola tertentu, sehingga tersusun secara teratur menurut aturan tertentu.

Ada 2 macam pengurutan, yaitu:

- Pengurutan secara ascending (urut naik).
- Pengurutan secara descending (urut turun).

2. Jenis Sorting

Beberapa algoritma dari jenis sorting diantaranya adalah

- Pengurutan berdasarkan perbandingan (*comparison-based sorting**)***
    - Bubble sort, exchange sort
- Pengurutan berdasarkan prioritas (*priority queue sorting method**)***
    - Selection sort, heap sort (menggunakan tree)
- Pengurutan berdasarkan penyisipan dan penjagaan terurut (*insert and keep sorted method*)
    - Insertion sort, tree sort
- Pengurutan berdasarkan pembagian dan penguasaan (*devide and conquer method*)
    - Quick sort, merge sort
- Pengurutan berkurang menurun (*diminishing increment sort method*)
    - Shell sort (pengembangan insertion)

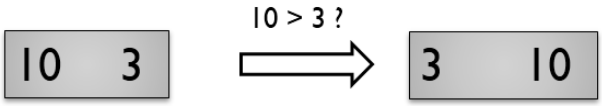
3. Konsep Bubble Sort

Metode sorting paling mudah, namun paling lambat dibandingkan dengan yang lain. Bubble sort mengurutkan data dengan cara membandingkan elemen sekarang dengan elemen berikutnya. Bisa dilakukan baik dari kepala array maupun ekor array.

Proses yang berlangsung, jika:

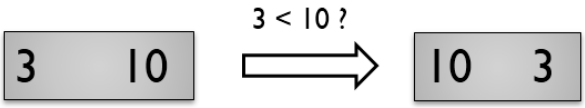
- Ascending: jika elemen sekarang lebih besar daripada elemen berikutnya, maka kedua elemen tersebut ditukar.

  Jika elemen sekarang **lebih besar** dari elemen berikutnya maka kedua elemen tersebut **ditukar**



- Descending: jika elemen sekarang lebih kecil daripada elemen berikutnya, maka kedua elemen tersebut ditukar.

  Jika elemen sekarang **lebih kecil** dari elemen berikutnya, maka kedua elemen tersebut **ditukar.**



. Contoh:

Data Acak	: 21	11	13	3	7	1
Ascending	: 1	3	7	11	13	21
Descending	: 21	13	11	7	3	1

Hal ini akan terlihat seperti penggeseran angka satu per satu elemen dari kanan ke kiri atau kiri ke kanan, tergantung jenis pengurutannya, asc atau desc. Proses penukaran ini akan terus dilakukan hingga seluruh array telah diperiksa. Dalam proses pengurutan data secara sederhana bisa diibaratkan seperti halnya gelembung udara (***bubble***). Bubble sort akan menggeser nilai yang terkecil atau terbesar ke posisi ujung dari daftar. Hingga semua daftar dalam keadaan terurut. Proses dasar yang terjadi dalam bubble sort adalah pertukaran nilai (*swapping*).

Contoh lainnnya dari bubble sort dapat dilihat pada tabel berikut ini.

44	**55**	**12**	**42**	**94**	**18**	**06**	**67**	**Data Awal**
44	12	42	55	18	06	67	94	Bandingkan data ke 1 dengan data ke 2, bila data ke 1 lebih besar tukar, seterusnya sampai data yang terbesar terletak di data ke 8
12	42	44	18	06	55	67	94	Bandingkan data ke 1 dengan data ke 2, bila data ke 1 lebih besar tukar, seterusnya sampai data yang terbesar terletak di data ke 7
12	42	18	06	44	55	67	94	Bandingkan data ke 1 dengan data ke 2, bila data ke 1 lebih besar tukar, seterusnya sampai data yang terbesar terletak di data ke 6
12	18	06	42	44	55	67	94	Bandingkan data ke 1 dengan data ke 2, bila data ke 1 lebih besar tukar, seterusnya sampai data yang terbesar terletak di data ke 5
12	06	18	42	44	55	67	94	Bandingkan data ke 1 dengan data ke 2, bila data ke 1 lebih besar tukar, seterusnya sampai data yang terbesar terletak di data ke 4
06	12	18	42	44	55	67	94	Bandingkan data ke 1 dengan data ke 2, bila data ke 1 lebih besar tukar, seterusnya sampai data yang terbesar terletak di data ke 3
06	12	18	42	44	55	67	94	Bandingkan data ke 1 dengan data ke 2, bila data ke 1 lebih besar tukar, seterusnya sampai data yang terbesar terletak di data ke 2
06	**12**	**18**	**42**	**44**	**55**	**67**	**94**	**Data setelah terurut**

4. Algoritma Bubble Sort:

```
void bubble ( int X [ ], int n )
{
        int hold, j, pass;
        for ( pass = 0; pass < n-1; pass++)
                for ( j = 0; j < n-pass-1; j++)
                        if ( X[j] > X[j+1] )
                        {
                                hold = X[j];
                                X[j] = X[j+1];
                                X[j+1] = hold;
                        }
}
```

Praktik 14 : Simple Bubble Sort:

```
package bubblesort;
public class BubbleSort {

   void bubbleSort(int arr[])
   {
      int n = arr.length;
      for (int i = 0; i < n-1; i++)
         for (int j = 0; j < n-i-1; j++)
            if (arr[j] > arr[j+1])
            {
               // swap arr[j+1] and arr[i]
               int temp = arr[j];
               arr[j] = arr[j+1];
               arr[j+1] = temp;
            }
   }

   /* Prints the array */
   void printArray(int arr[])
   {
      int n = arr.length;
      for (int i=0; i<n; ++i)
```

```
            System.out.print(arr[i] + " ");
        System.out.println();
    }

    // Driver method to test above
    public static void main(String args[])
    {
        BubbleSort ob = new BubbleSort();
        int arr[] = {5, 3, 1, 9, 8, 2, 4, 7};
        ob.bubbleSort(arr);
        System.out.println("Sorted array");
        ob.printArray(arr);
    }
}
/* Source: https://www.geeksforgeeks.org/ */
```

Ilustrasi Praktik Simple Bubble Sort:

i = 0	j	0	1	2	3	4	5	6	7
	0	5	3	1	9	8	2	4	7
	1	3	5	1	9	8	2	4	7
	2	3	1	5	9	8	2	4	7
	3	3	1	5	9	8	2	4	7
	4	3	1	5	8	9	2	4	7
	5	3	1	5	8	2	9	4	7
	6	3	1	5	8	2	4	9	7
i = 1	0	3	1	5	8	2	4	7	**9**
	1	1	3	5	8	2	4	7	
	2	1	3	5	8	2	4	7	
	3	1	3	5	8	2	4	7	
	4	1	3	5	2	8	4	7	
	5	1	3	5	2	4	8	7	
i = 2	0	1	3	5	2	4	7	**8**	
	1	1	3	5	2	4	7		
	2	1	3	5	2	4	7		
	3	1	3	2	5	4	7		
	4	1	3	2	4	5	7		
i = 3	0	1	3	2	4	5	**7**		
	1	1	3	2	4	5			
	2	1	2	3	4	5			
	3	1	2	3	4	5			
i = 4	0	1	2	3	4	**5**			
	1	1	2	3	4				
	2	1	2	3	4				
i = 5	0	1	2	3	**4**				
	1	1	2	3					
i = 6	0	1	2	**3**					
		1	**2**						

Praktik 15: Detail Simple Bubble Sort:

```
package bubblesortdetail;
public class BubbleSortDetail {
   public static void main(String[] args){
        int arr[] ={5, 3, 1, 9, 8, 2, 4, 7};
        System.out.println("---Array BEFORE Bubble Sort---");
        printArray(arr);
        BubbleSortDetail(arr);//sorting array elements using bubble sort
        System.out.println("---Array AFTER Bubble Sort---");
        printArray(arr);
        }
   static void BubbleSortDetail(int[] array){
        int n = array.length;
        int temp = 0;
        for(int i=0; i < n; i++) // Looping through the array length
        {  System.out.println("Sort yang ke "+(i+1));
           for(int j=1; j < (n-i); j++){
              System.out.println("Bandingkan "+ array[j-1]+ " dan " + array[j]);
                 if(array[j-1] > array[j]){
                    //swap elements
                    temp = array[j-1];
                    array[j-1] = array[j];
                    array[j] = temp;
                    System.out.println(array[j]  + " lebih besar dari pada " + array[j-1]);
                    System.out.println("Tukar elemen: Array baru setelah tukar");
                    printArray(array);
                          }
              }
                 }
        }
   static void printArray(int[] array){
        for(int i=0; i < array.length; i++){
        System.out.print(array[i] + " ");
        }
        System.out.println();
   }
}
//source: https://www.guru99.com
```

Latihan Modul Bubble Sort

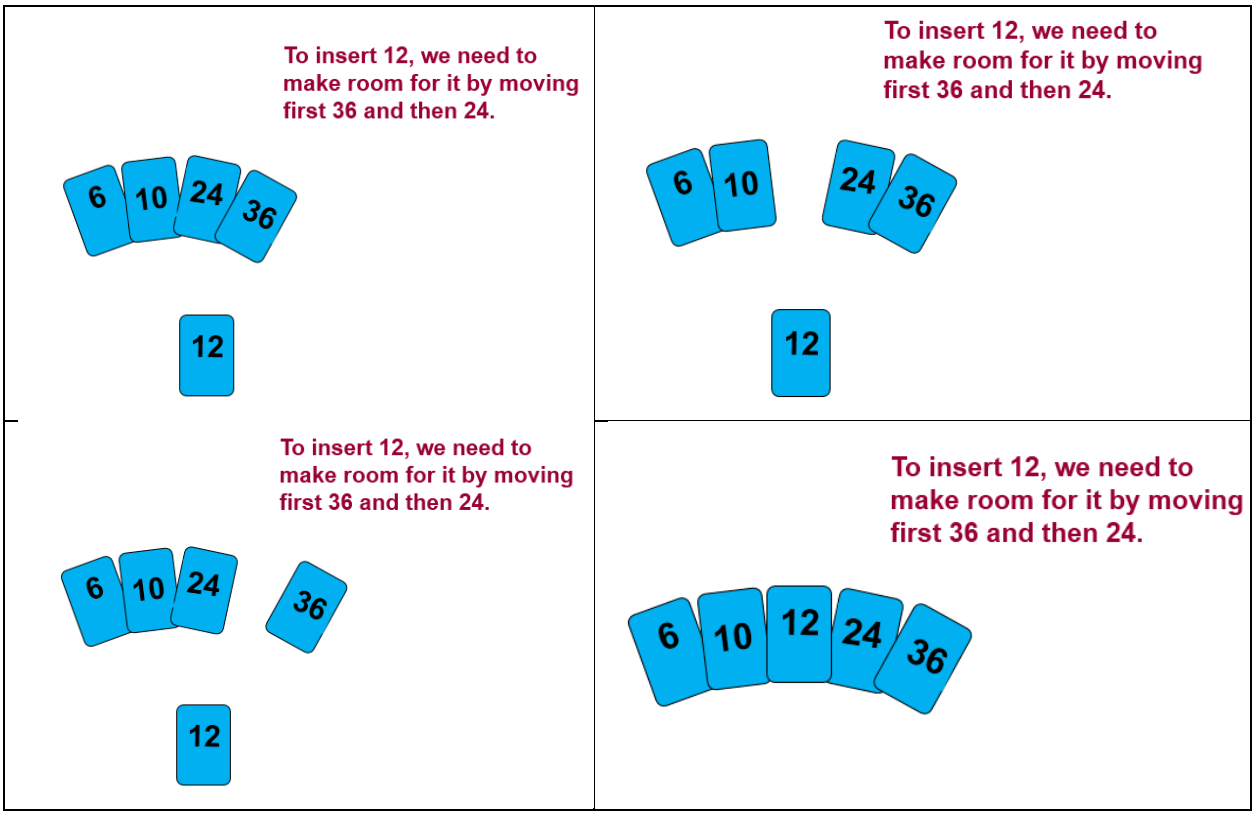
Ketik program berikut ini, amati dan pahami program dan outputnya.

```
package bubblesort2;
public class BubbleSort2 {
   static void bubbleSort(int[] arr) {
      int n = arr.length;
      int temp = 0;

      for(int i = 0; i < n; i++) {
         for(int j=1; j < (n-i); j++) {
            if(arr[j-1] > arr[j]) {
               temp = arr[j-1];
               arr[j-1] = arr[j];
               arr[j] = temp;
            }
         }
      }
   }
   public static void main(String[] args) {
      int arr[] = { 2, 5, -2, 6, -3, 8, 0, -7, -9, 4 };
      System.out.println("Array Before Bubble Sort");

      for(int i = 0; i < arr.length; i++) {
         System.out.print(arr[i] + " ");
      }
      System.out.println();
      bubbleSort(arr);
      System.out.println("Array After Bubble Sort");

      for(int i = 0; i < arr.length; i++) {
         System.out.print(arr[i] + " ");
      }
   }
}
//source : https://www.tutorialspoint.com/
```

MODUL 6: INSERTION SORT

1. Konsep Insertion Sort

Sebelum melihat konsep dari insertion sort ini, kita liat analogi pengurutan kartu. Proses dilakukan dengan membandingkan data ke-I dengan data yang sebelum-sebelumnya. Misal ascending: pengurutan dimulai dari data ke-2 sampai dengan data terakhir, jika ditemukan data yang lebih kecil, maka akan dimasukkan di posisi yang seharusnya. Pada penyisipan elemen, maka elemen-elemen lain akan bergeser ke belakang. Metode pengurutan langsung yang mengambil sebuah data sisip, dimana data sisip tersebut akan diurutkan dan menggeser data yang lebih besar dari data sisip agar data sisip dapat ditempatkan di tempat yang benar.



2. Algoritma dari Insertion Sort

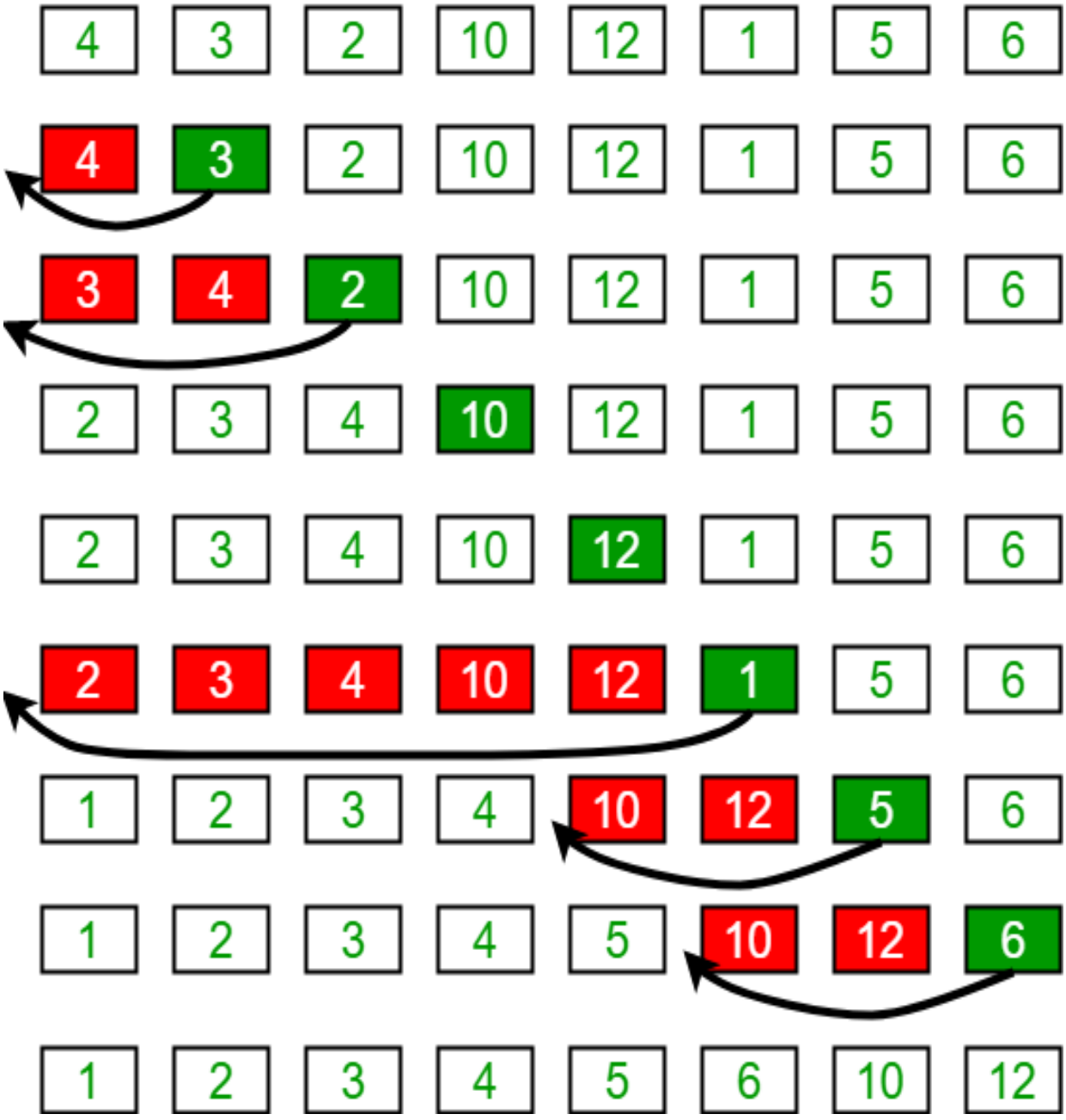
```
void insertion_sort(int data[]){
        int i, n, temp;
        for(i=1;i<n;i++){
                temp=data[i];
                j=i-1;
                while(data[j]>temp && j>=0){
                        data[j+1]=data[j];
                        j--;
                }data[j+1]=temp;
        }
}
```

Praktik 16: Insertion Sort Simple

```
package demoinsertionsort;
public class DemoInsertionSort {
   /*Function to sort array using insertion sort*/
   void sort(int arr[])
   {
      int n = arr.length;
      for (int i = 1; i < n; ++i) {
         int key = arr[i];
         int j = i - 1;

         /* Move elements of arr[0..i-1], that are  greater than key, to one position ahead
            of their current position */
         while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j = j - 1;
         }
         arr[j + 1] = key;
      }
   }
 /* A utility function to print array of size n*/
   static void printArray(int arr[])
   {
      int n = arr.length;
      for (int i = 0; i < n; ++i)
```

```
            System.out.print(arr[i] + " ");

        System.out.println();
    }

    // Driver method
    public static void main(String args[])
    {
        int arr[] = { 4, 3, 2, 10, 12, 1, 5, 6};

        DemoInsertionSort ob = new DemoInsertionSort();
        ob.sort(arr);

        printArray(arr);
    }
} /* This code is contributed by Rajat Mishra. https://www.geeksforgeeks.org/insertion-sort/*/
```

Insertion Sort Execution Example



Praktik 17 : Detail Insertion Sort Simple

```
package insertionsortdetail;
public class InsertionSortDetail {
public static void main(String a[]) {
        int[] myArray  = {4, 3, 2, 10, 12, 1, 5, 6};
        System.out.println("Before Insertion Sort");
        printArray(myArray);
        insertionSort(myArray);//sorting array using insertion sort
        System.out.println("After Insertion Sort");
        printArray(myArray);
    }
    public static void insertionSort(int arr[]) {
        int n = arr.length;
        for (int i = 1; i < n; i++)
        {   System.out.println("Sorting yang ke:  "+(i));
            int key = arr[i];
            int j = i-1;

            while ( (j > -1) && ( arr [j] > key ) )
            {
            System.out.println("Bandingkan "+ key  + " dan " + arr [j]);
                arr [j+1] = arr [j];
                j--;
            }
            arr[j+1] = key;
            System.out.println("Tukarkan elemen: Array baru setelah tukar");
            printArray(arr);
        }
    }
 static void printArray(int[] array){

                for(int i=0; i < array.length; i++)
                        {
                                System.out.print(array[i] + " ");
                        }
                System.out.println();

            }
}
//source: https://www.guru99.com/
```

Latihan Modul Insertion Sort

Ketik program berikut ini, amati dan pahami program dan outputnya.

```
package insertionsortexample;
public class InsertionSortExample {
    public static void insertionSort(int array[]) {
        int n = array.length;
        for (int j = 1; j < n; j++) {
            int key = array[j];
            int i = j-1;
            while ( (i > -1) && ( array [i] > key ) ) {
                array [i+1] = array [i];
                i--;
            }
            array[i+1] = key;
        }
    }   public static void main(String a[]){
        int[] arr1 = {9,14,3,2,43,11,58,22};
        System.out.println("Before Insertion Sort");
        for(int i:arr1){
            System.out.print(i+" ");
        }
        System.out.println();

        insertionSort(arr1);//sorting array using insertion sort

        System.out.println("After Insertion Sort");
        for(int i:arr1){
            System.out.print(i+" ");
        }
        System.out.println();
    }

}
//source: https://www.javatpoint.com/insertion-sort-in-java
```

MODUL 7: SELECTION SORT

1. Konsep Selection Sort

Pada dasarnya, selection sort merupakan kombinasi dari sorting dan searching. Untuk setiap proses, dilakukan dengan mencari elemen dari posisi yang belum diurutkan dan kemudian memilih elemen yang memiliki nilai terkecil atau terbesar yang akan ditukarkan ke posisi yang tepat di dalam array. Misalnya untuk putaran pertama, akan dicari data dengan nilai terkecil dan data ini akan ditempatkan pada indeks terkecil, pada putaran kedua akan dicari data kedua terkecil, dan akan ditempatkan di indeks kedua, begitu seterusnya hingga tidak ada data yang dicari lagi. Selama proses, pembandingan dan pengubahan hanya dilakukan pada indeks pembanding saja, **pertukaran data secara fisik terjadi pada akhir proses.**

Ide utama dari algoritma selection sort adalah memilih elemen dengan nilai paling rendah dan menukar elemen yang terpilih dengan elemen ke-i. Nilai dari i dimulai dari 1 ke n, dimana n adalah jumlah total elemen dikurangi 1.

2. Algoritma dari Selection Sort

```
void selection ( int X [ ], int n ) {
        int i, j, index, large;
        for ( i = n-1; i > 0; i--)  {
                large = X[0];
                index = 0;
                for ( j = 1; j <= i; j++){
                        if ( X[j] > large ){
                                large = X[j];
                                index = j;
                        }
                }
                X[index] = X[i];
                X[i] = large;
        }
}
```

Praktik 18: Selection Sort-1

```
package selectionsort;
public class SelectionSort {
 void sort(int arr[])
    {
        int n = arr.length;

        // One by one move boundary of unsorted subarray
        for (int i = 0; i < n-1; i++)
        {
            // Find the minimum element in unsorted array
            int min_idx = i;
            for (int j = i+1; j < n; j++)
                if (arr[j] < arr[min_idx])
                    min_idx = j;

            // Swap the found minimum element with the first
            // element
            int temp = arr[min_idx];
            arr[min_idx] = arr[i];
            arr[i] = temp;
        }
    }

   // Prints the array
    void printArray(int arr[])
    {
        int n = arr.length;
        for (int i=0; i<n; ++i)
            System.out.print(arr[i]+" ");
        System.out.println();
    }

    // Driver code to test above
    public static void main(String args[])
    {
        SelectionSort ob = new SelectionSort();
        int arr[] = { 7,4,6,3,0,5,1,2};
        ob.sort(arr);
        System.out.println("Sorted array");
        ob.printArray(arr);
    }
}
/* Source: https://www.geeksforgeeks.org/selection-sort/
   This code is contributed by Rajat Mishra*/
```

Latihan Modul Selection Sort

Ketik program berikut ini, amati dan pahami program dan outputnya.

```
package selectionsort2;
public class SelectionSort2 {
public static void main(String a[]){
   int[] myArray = {7,4,6,3,0,5,1,2};
   System.out.println("------Before Selection Sort-----");
   printArray(myArray);
   selection(myArray);//sorting array using selection sort
   System.out.println("-----After Selection Sort-----");
      printArray(myArray);
   }
public static void selection(int[] array){
   for (int i = 0; i < array.length - 1; i++) {
      System.out.println("Sort yang ke "+(i+1));
      int index = i;
      for (int j = i + 1; j < array.length; j++){
         System.out.println("Bandingkan "+ array[index]  + " dan " + array[j]);
         if (array[j] < array[index]){
                  System.out.println(array[index]  + " lebih besar dari pada " + array[j] );
                  index = j;
         }
         }
      int smallerNumber = array[index];
         array[index] = array[i];
         array[i] = smallerNumber;
         System.out.println("Tukar elemen: Array Baru setelah tukar");
         printArray(array);
   }
}
   static void printArray(int[] array){
      for(int i=0; i < array.length; i++){
         System.out.print(array[i] + " ");
      }
      System.out.println();
   }

}
//source: https://www.guru99.com/
```

MODUL 8: SEQUANTIAL SEARCH

1. Konsep Sequential Search

Searching adalah pencarian data dengan cara menelusuri data-data tersebut. Tempat pencarian data dapat berupa array dalam memori, bisa juga pada file pada external storage. Pada suatu data seringkali dibutuhkan pembacaan kembali informasi (retrieval information) dengan cara searching.

Sequential search adalah suatu teknik pencarian data dalam array (1 dimensi) yang akan menelusuri semua elemen-elemen array dari awal sampai akhir, dimana data-data **tidak perlu** diurutkan terlebih dahulu.

Kemungkinan terbaik adalah jika data yang dicari terletak di indeks array terdepan (elemen array pertama) sehingga waktu yang dibutuhkan untuk pencarian data sangat sebentar (minimal). Kemungkinan terburuk adalah jika data yang dicari terletak di indeks array terakhir (elemen array terakhir) sehingga waktu yang dibutuhkan untuk pencarian data sangat lama (maksimal).

2. Algoritma sequential search

```
for (i = 0; i < n; i++)
        {
                if (data[i]==key)
                        return (i);         //return the index
        }
        return (-1);     //key not found
```

Ilustrasi Sequential Search

- Misalnya terdapat array satu dimensi sebagai berikut:

0	1	2	3	4	5	6	7	indeks
8	10	6	-2	11	7	1	100	value

- Kemudian program akan meminta data yang akan dicari, misalnya **6 (x = 6)**.
- Iterasi :

  6 = 8 (tidak!)

  6 = 10 (tidak!)

  6 = 6 (Ya!) => output : "Ada" pada index ke-2
- Jika sampai data terakhir tidak ditemukan data yang sama maka output : " data yang dicari tidak ada".

Praktik 19 : Sequential Searching-1

```
package sequentialsearching.pkg1;
public class SequentialSearching1{

public static int search(int arr[], int x)
{
   int n = arr.length;
   for(int i = 0; i < n; i++)
   {
      if(arr[i] == x)
         return i;
   }
   return -1;
}
   public static void main(String args[]){
      int arr[] = { 2, 3, 4, 10, 40 };
      int x = 10;

      int result = search(arr, x);
      if(result == -1)
         System.out.print("Element is not present in array");
      else
         System.out.print("Element is present at index " + result);

    System.out.println();
   }
}
//source: https://www.geeksforgeeks.org/
```

DAFTAR PUSTAKA

1. http://www.unsri.ac.id/upload/arsip/prak_strukturdata.pdf
2. http://elearning.upnjatim.ac.id/courses/MODULPRAKTIKUMJURUSANSISTEMINFORMASI/document/Modul_Praktikum_Struktur_Data.pdf?cidReq=MODULPRAKTIKUMJURUSANSISTEMINFORMASI
3. https://www.codepolitan.com/mengenal-matriks-dan-operasinya-di-java
4. http://elektro.um.ac.id/wp-content/uploads/2016/09/Modul-1-Struct-Aray-Pointer-2015.pdf
5. https://www.geeksforgeeks.org/stack-push-method-in-java/
6. https://agung-setiawan.com
7. https://co-code.blogspot.com/2016/11/queue-dequeue-enqueue.html
8. https://safriansah.blogspot.com/2018/08/modul-java-queue-program-antrian-bank.html
9. http://bobby-syahronanda.blogspot.com/2014/01/contoh-program-queue-dengan-java.html
10. http://sisinform-aaf1231072.blogspot.com/2013/02/bubble-sort.html
11. https://www.tutorialspoint.com/java-program-to-implement-bubble-sort
12. https://www.slideshare.net/DurgaDevi72/unit-ivdsc
13. https://cdncontribute.geeksforgeeks.org/wp-content/uploads/insertionsort.png
14. https://www.javatpoint.com/insertion-sort-in-java
15. https://www.guru99.com/insertion-sort-java.html
16. https://lunarphue.wordpress.com/information-technology/asd/macam-macam-sorting/
17. https://www.guru99.com/selection-sorting-java.html
18. https://slideplayer.info/slide/3958960/12/images/9/Ilustrasi+Sequential+Search.jpg

Referensi Gambar

- http://whitecatcoaching.com.au/wp-content/uploads/2015/04/White-Cat-Coaching-not-problem.jpg
- http://3.bp.blogspot.com/-p6BPt-ggtiQ/Vf0KWECcJPI/AAAAAAAANDM/MpcS4ysSKuA/s1600/IMG_20150908_075647.jpg
- https://www.google.co.id/url?sa=i&rct=j&q=&esrc=s&source=imgres&cd=&cad=rja&uact=8&ved=0ahUKEwj-te-8icDOAhUdTo8KHbnBBEIQjRwIBw&url=http%3A%2F%2Fwww.twitrcovers.com%2Ftwitter-covers%2Fprogramming-code%2F&psig=AFQjCNE_jfQ0lYXvdV8O67YZx5og1kQSjQ&ust=1471235500926867
- http://afathoni25.blog.unsoed.ac.id/files/2011/10/al-khawarizmi.jpg
- https://www.codepolitan.com/mengenal-dan-memulai-pemrograman-java-belajar-java
- https://www.okedroid.com/2016/04/belajar-cara-memahami-konsep-dasar-bahasa-program-java.html