

**Modul 1**  
**Object Oriented Programming**  
**2022**



# **MODUL 1**

**KONSEP OOP & PROCEDURAL  
LANGUAGE DI PYTHON DAN  
JAVA BASIC**

## ASISTEN PRAKTIKUM



**Abiyoga Dhaniswara  
ABII**



**Luh Komang Devi  
DEVS**



**Ahmad Syahid Danu  
ASDW**



**Maulana Rakha R  
MARK**



**Amilia Agata  
MILY**



**Metta Triana Asri  
JMET**



**Dharu Widhyanugrah  
DINO**



**Mohamad Fahrudin  
LFTG**



**Erina Zahira Nurhasan  
RYIN**



**M. Nanda Nugraha  
NANZ**



**Fadli Kurniawan  
YORU**



**M. Haris Sitompul  
RTSU**



**Farel Baihaky  
YNWA**



**Nabila Melsyana  
LALA**



**Fazar Arya Suwandi  
RJEP**



**Rafidah Jasmine L  
MINT**



**Gede Dipta Narayana  
GING**



**Windy Kurniawan  
WNDD**



**Hudzaifah Afif  
GGEZ**



**Yoga Raditya N  
GOYS**



**Indira Agustia Garini  
DIRA**

## Daftar Isi

|   |           |
|---|-----------|
| <b>Daftar isi</b>                                   | <b>I</b>  |
| <b>Peraturan Praktikum</b>                          | <b>II</b> |
| <b>Konsep OOP</b>                                   | <b>1</b>  |
| Perbedaan Reusability Class dan Procedural Language | 1         |
| Keunggulan Java dalam Object Oriented Programming   | 2         |
| <b>Object Dan Class</b>                             | <b>3</b>  |
| Deklarasi Class                                     | 3         |
| Deklarasi Variable                                  | 3         |
| <b>Method</b>                                       | <b>5</b>  |
| Method Void   | 5         |
| Method Return                                       | 6         |
| Method Static                                       | 7         |
| Metode Setter dan Getter                            | 8         |
| <b>Scanner pada Java</b>                            | <b>9</b>  |
| <b>Create Object (Constructor)</b>                  | <b>10</b> |
| <b>Array List</b>                                   | <b>11</b> |
| <b>Loop</b>   | <b>12</b> |
| For Loop  | 12        |
| While Loop  | 13        |
| <b>Array List Loop</b>                              | <b>14</b> |
| For-Each Loop                                       | 14        |
| For Index Loop                                      | 15        |

## Peraturan Praktikum

1. Setiap peserta praktikum harus datang tepat waktu sesuai dengan jadwal. Toleransi keterlambatan hadir 15 menit. Jika melebihi batas waktu tersebut diperkenankan mengikuti praktikum namun tidak mendapatkan tambahan waktu dan/atau nilai.
2. Perizinan Praktikum :
  - a. Izin berkaitan dengan Sakit atau Kemalangan , maka praktikan dapat memberikan surat perizinan kepada pihak Komisi Disiplin maksimal 3 hari setelah jadwal (shift) praktikum.
  - b. Izin lomba atau penugasan institusi tidak berlaku apabila tidak terdapat bukti dispensasi dari Igracias. NB : screenshot dispensasi dari igracias wajib dilampirkan.
3. Seragam Praktikum :
  - a. Mahasiswa wajib menggunakan celana bahan hitam (bukan chino atau jeans) pada saat praktikum.
  - b. Mahasiswi wajib menggunakan rok hitam/biru gelap panjang tidak ketat pada saat praktikum.
  - c. Dresscode praktikum (Mengikuti Peraturan Telkom)
    - i. Senin : Menggunakan kemeja merah telkom atau kemeja putih polos
    - ii. Selasa s/d Rabu : Menggunakan kemeja putih polos
    - iii. Kamis s/d Sabtu : Menggunakan kemeja formal berkerah (bukan kerah sanghai dan bukan polo)
    - iv. Jumat : Menggunakan kemeja/ baju batik bukan outer.
  - d. Jika terdapat kendala dalam baju seragam maka praktikan diperbolehkan mengganti menggunakan kemeja putih Telkom atau polos..
  - e. Membuka sepatu saat memasuki ruangan lab.
4. Peraturan Pengerjaan
  - a. Studi Kasus dikerjakan secara individu.
  - b. Jawaban tidak boleh sama dengan setiap individu.
  - c. Hasil pengerjaan di push ke repository github tiap individu.
  - d. Submit hasil pengerjaan berupa file format PDF dan link github ke LMS tiap individu.
  - e. Format Pengumpulan Hasil Pengerjaan
    - **Format penamaan file yang dikumpulkan di LMS**
      - Format nama file Jurnal PDF:
        - **OOP\_KODEASISTEN\_NAMAPENDEK\_NIM\_SSMODULX**
        - Contoh: **OOP\_MILY\_AMILIA\_1202200000\_SSMODUL1**

## Peraturan Praktikum

- Format nama file Tugas Pendahuluan PDF:
  - **OOP\_KODEASISTEN\_NAMAPENDEK\_NIM\_SSTPX**
  - Contoh: **OOP\_MILY\_AMILIA\_1202200000\_SSTPI**
- **Format penamaan folder yang di push ke repository github**
  - Alur penamaan folder:
    - Folder Modul "**MODUL X**"
  - Create Project untuk TP dan Jurnal :
    - Folder Project TP: "**TPMODULX\_NAMAPENDEK**" (Di dalam Folder Modul)
    - Folder Project Jurnal: "**MODULX\_NAMAPENDEK**" (Di dalam Folder Modul)
  - Format nama repository
    - **OOP-KODE ASISTEN-NAMA PENDEK PRAKTIKAN-NIM**
    - Contoh : **OOP-MILY-AMILIA-1202200000**
  - Format nama folder Jurnal:
    - **MODULX\_NAMAPENDEK**
    - Contoh: **MODUL1\_AMILIA**
  - Format nama folder Tugas Pendahuluan:
    - **TPMODULX\_NAMAPENDEK**
    - Contoh: **TPMODUL1\_AMILIA**
- f. Salah penamaan pada file pengerjaan nilai modul akan dipotong sebesar **10%**
- g. Terlambat mengumpulkan file pengerjaan nilai modul akan dipotong sebesar **15%**.
- h. Terlambat commit pengerjaan nilai modul akan dipotong sebesar **15%**.
- i. Segala alat komunikasi dikumpulkan di loker yang tersedia dalam ruangan.
- j. Jika ada perangkat praktikum yang bermasalah dapat menghubungi asprak yang bertugas.
- k. Segala bentuk kecurangan dan plagiarisme akan diproses ke komisi disiplin dan nilai akhir modul menjadi 0

## Konsep OOP

Konsep OOP (Object-Oriented Programming) adalah paradigma pemrograman yang berfokus pada pemodelan dunia nyata dengan menggunakan objek dan interaksi antara objek-objek tersebut. Dalam OOP, program dibangun berdasarkan kelas dan objek. Kelas adalah blueprint atau template yang mendefinisikan atribut dan metode yang dimiliki oleh objek, sedangkan objek adalah instansi konkret dari sebuah kelas yang dapat berinteraksi satu sama lain.

### Perbedaan Reusability Class dan Procedural Language

Perbedaan antara reusability class (OOP) dan penggunaan method biasa (procedural language) adalah bahwa reusability class memungkinkan pembuatan objek yang dapat digunakan kembali dengan mudah dan efisien. Dalam OOP, kita dapat membuat kelas yang berisi atribut dan metode yang dapat digunakan oleh objek-objek lain, sehingga memungkinkan pembuatan objek baru dengan menggunakan blueprint yang sama. Sebaliknya, dalam pemrograman Procedural Language, kita mungkin perlu menulis ulang kode atau menggunakan fungsi yang memproses data secara eksplisit untuk mencapai reusabilitas.

Contoh Procedural Language pada Python:

```
1 # Contoh fungsi prosedural untuk menghitung luas lingkaran
2 def hitung_luas_lingkaran(jari_jari):
3     luas = 3.14 * (jari_jari ** 2)
4     return luas
5
6 # Contoh penggunaan fungsi hitung_luas_lingkaran
7 jari_jari = 5
8 luas = hitung_luas_lingkaran(jari_jari)
9 print("Luas lingkaran dengan jari-jari ", jari_jari, " adalah ", luas)
```

Output:

```
Luas lingkaran dengan jari-jari 5 adalah 78.5
```

## Keunggulan Java dalam Object Oriented Programming

Java dipilih sebagai bahasa pemrograman yang umum digunakan untuk mempelajari konsep OOP karena memiliki dukungan yang kuat untuk OOP dan enkapsulasi. Java menyediakan fitur-fitur seperti class, objek, inheritance, dan access modifiers yang memungkinkan implementasi yang jelas dan terstruktur dari konsep OOP. Di sisi lain, Python, meskipun juga mendukung OOP, memiliki pendekatan yang lebih fleksibel dan kurang ketat terhadap enkapsulasi. Python tidak memiliki modifier private secara eksplisit, yang berarti tidak ada pembatasan akses ke atribut atau metode tertentu. Oleh karena itu, Java sering dipilih sebagai bahasa pemrograman yang lebih cocok untuk mempelajari OOP secara mendalam.

Contoh Object Oriented Programming dengan Python:

```
1 class Lingkaran:
2     def __init__(self, jari_jari):
3         self.jari_jari = jari_jari
4
5     def hitung_luas(self):
6         luas = 3.14 * (self.jari_jari ** 2)
7         return luas
8
9     def hitung_keliling(self):
10        keliling = 2 * 3.14 * self.jari_jari
11        return keliling
12
13 # Contoh penggunaan kelas Lingkaran
14 lingkaran1 = Lingkaran(5)
15 luas = lingkaran1.hitung_luas()
16 keliling = lingkaran1.hitung_keliling()
17 print("Luas lingkaran: ", luas)
18 print("Keliling lingkaran: ", keliling)
```

Output:

```
Luas lingkaran: 78.5
Keliling lingkaran: 31.4000000
00000002
```

## Object Dan Class

Dua konsep terpenting dalam pemrograman berorientasi objek adalah kelas dan objek. Dalam arti luas, objek adalah sesuatu, baik yang berwujud maupun tidak berwujud, yang dapat kita bayangkan. Sebuah program yang ditulis dalam gaya berorientasi objek akan terdiri dari objek-objek yang saling berinteraksi. Contoh dari Objek itu sendiri seperti : Motor, Mobil, dll.

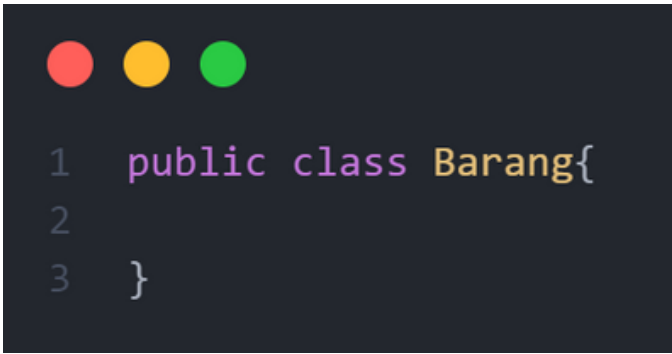
Di dalam program kita menulis instruksi untuk membuat objek. Agar komputer dapat membuat objek, kita harus memberikan definisi, yang disebut kelas. Kelas adalah sejenis cetakan atau template yang menentukan apa yang bisa dan tidak bisa dilakukan oleh objek. Sebuah objek disebut instance dari kelas.

### Deklarasi Class

Pada pemrograman Java, pendeklarasian kelas penamaan kelas harus sama dengan nama file java dan diawali dengan huruf kapital, dapat dilakukan dengan menggunakan syntax sebagai berikut:

**[ modifier ] class Class\_identifier**

Contoh Pendeklarasian Class:



```
1  public class Barang{  
2  
3  }
```



## Deklarasi Variable

Deklarasi variabel dilakukan di dalam kelas. Variabel yang didefinisikan di kelas merupakan atribut dari kelas tersebut (yang otomatis merupakan atribut dari objek). Penugasan terhadap variabel merupakan pemberian nilai kepada nilai. Karena dalam deklarasi kelas, penugasan variabel merupakan penugasan pertama kali, maka penugasan ini dapat juga disebut inisialisasi variabel. Syntax umum deklarasi dan inisialisasi variabel adalah sebagai berikut:

**[ modifier ] data\_type identifier [ =value ];**

Keterangan:

- **Modifier** : Merupakan kata kunci untuk mendefinisikan makna dari suatu variable, method, atau class. Seperti private, protected, public, dll.
  - **Private**: Hanya dapat diakses di dalam kelas yang sama. Anggota tersebut tidak dapat dilihat oleh kelas lain, termasuk subclass (kelas turunan).
  - **Protected**: Dapat diakses di dalam kelas yang sama, serta di dalam subclass (kelas turunan) dari kelas tersebut. Namun, anggota tersebut tidak dapat diakses di luar hierarki kelas tersebut.
  - **Public**: Dapat diakses dan dimodifikasi dari mana saja, baik di dalam kelas maupun di luar kelas.
- **Data\_type** : Merupakan kata kunci tipe data dari variabel, misalnya: int, float, double, string, dll.
- **Identifier** : Merupakan nama variabel (Nama variable tidak boleh ada spasi).
- **Value** : Merupakan nilai awal dari variabel, bersifat opsional.

Contoh Pendeklarasian Variable:

```
public class Barang {  
    public int jumlah;  
    public int hargaBeli;  
    public Date tanggalKadaluarsa;  
    public int hargaJual;  
    public String idBarang;  
    public double diskon=0.0;  
}
```

## Method

*Method* adalah satu container pada kelas yang memuat baris-baris kode. *Method* biasanya digunakan untuk mengenkapsulasi proses-proses yang diperlukan untuk membentuk suatu fungsi atau tugas tertentu. Misalnya ada *object* kucing, dan kucing tersebut dapat makan. *Method* secara singkat dibagi menjadi dua yaitu **method void** dan **method return**.

### Method Void

*Method void* adalah metode yang digunakan untuk memberikan nilai pada suatu *attribute*, *object*, dan lain - lain. *Method void* memiliki ciri ciri memiliki kata "void" dan "this".

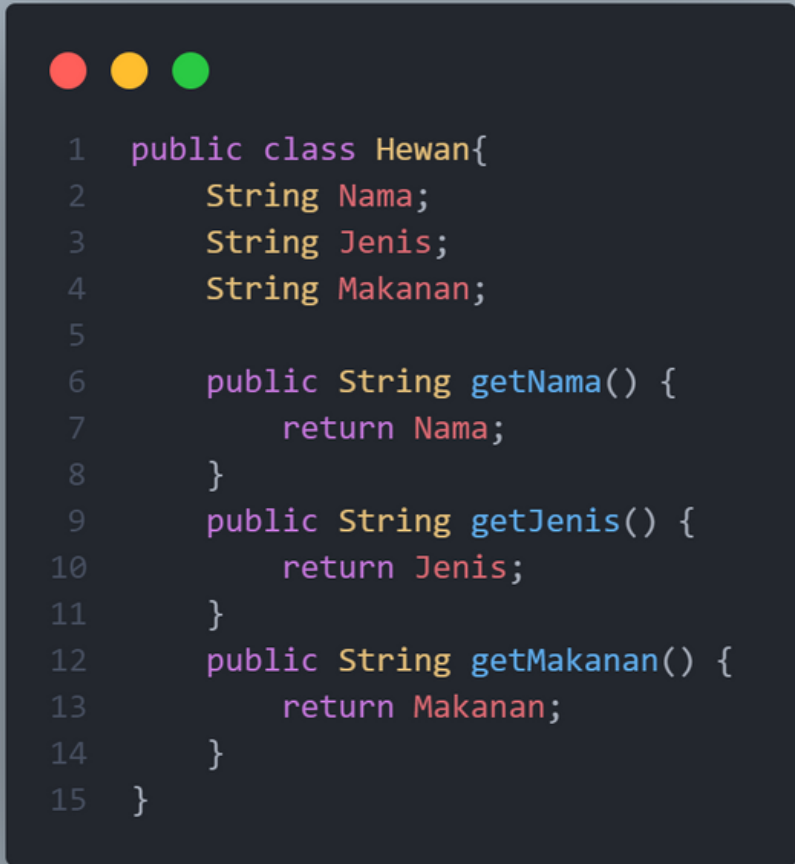
*Contoh Method Void:*

```
1  public class Hewan{
2      String Nama;
3      String Jenis;
4      String Makanan;
5
6      public void setNama(String nama) {
7          this.Nama = nama;
8      }
9
10     public void setJenis(String jenis) {
11         this.Jenis = jenis;
12     }
13
14     public void setMakanan(String makanan) {
15         this.Makanan = makanan;
16     }
17 }
```

## Method Return

Method Return adalah metode yang pengembalian nilai dari suatu object atau attribute yang sudah berisi nilai. Method return memakai tipe data dan tidak menggunakan "void" melainkan memiliki kata kunci "return".

Contoh Method Return:



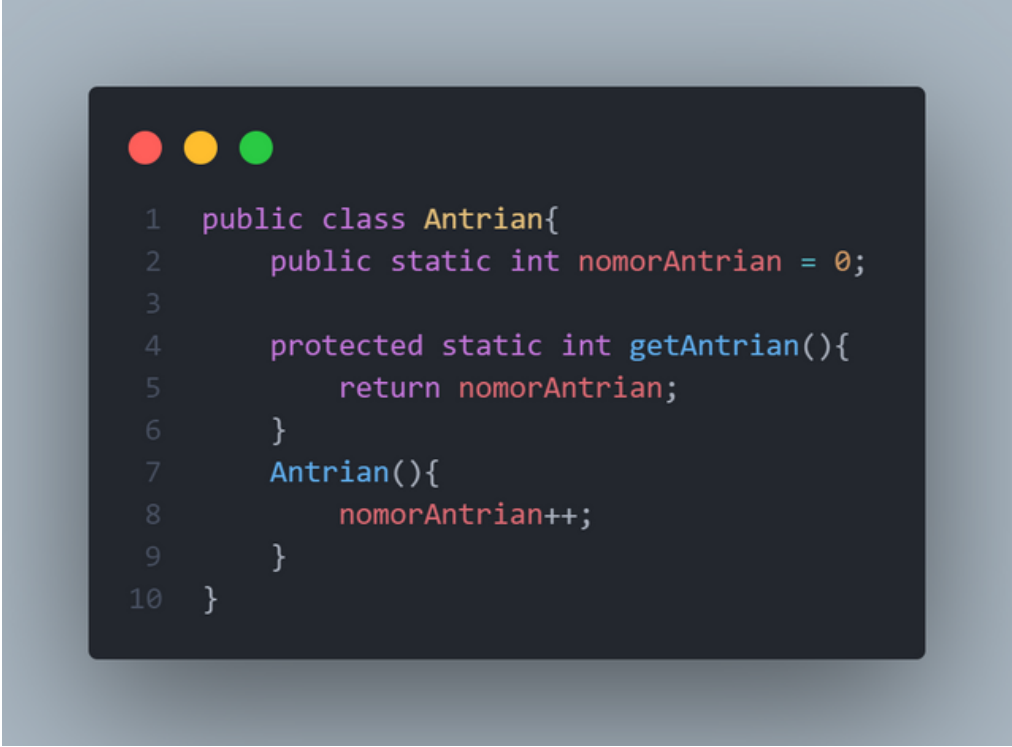
```
1  public class Hewan{
2      String Nama;
3      String Jenis;
4      String Makanan;
5
6      public String getNama() {
7          return Nama;
8      }
9      public String getJenis() {
10         return Jenis;
11     }
12     public String getMakanan() {
13         return Makanan;
14     }
15 }
```

## Method Static

Method Static merupakan metode yang dapat berdiri sendiri tanpa perlu instansiasi dari kelas. Dapat dilakukan dengan menggunakan syntax sebagai berikut:

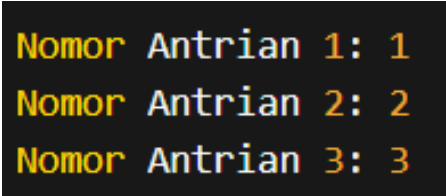
**[modifiers] static return\_type method\_identifier ( parameter )  
{method\_body; }**

Contoh Method Static:



```
1 public class Antrian{
2     public static int nomorAntrian = 0;
3
4     protected static int getAntrian(){
5         return nomorAntrian;
6     }
7     Antrian(){
8         nomorAntrian++;
9     }
10 }
```

Output:




```
Nomor Antrian 1: 1
Nomor Antrian 2: 2
Nomor Antrian 3: 3
```

## Metode Setter dan Getter

Setter dan Getter adalah metode dalam pemrograman yang digunakan untuk mengatur (set) dan mengambil (get) nilai dari suatu variabel dalam sebuah objek. Metode setter digunakan untuk mengubah nilai variabel, sedangkan metode getter digunakan untuk mengambil nilai variabel.

Contoh Metode Setter dan Getter:



```
1  public class Hewan{
2      String name;
3      String type;
4
5      // Setter untuk variabel name dan type
6      public void setName(String name) {
7          this.name = name;
8      }
9      public void setType(String type) {
10         this.type = type;
11     }
12
13     // Getter untuk variabel name dan type
14     public String getName() {
15         return name;
16     }
17     public String getType() {
18         return type;
19     }
20 }
```

## Scanner pada Java

Scanner adalah kelas yang disediakan oleh Java untuk membaca input dari pengguna. Dengan menggunakan kelas Scanner, Anda dapat membaca input dari keyboard atau dari sumber lainnya dalam program Java Anda. Untuk menggunakan Scanner, Anda perlu mengimpor paket `java.util.Scanner`.

Contoh penggunaan Scanner :

```
1  import java.util.Scanner;
2
3  public class Main {
4      public static void main(String[] args) {
5          // Membuat objek scanner
6          Scanner scanner = new Scanner(System.in);
7          System.out.print("Masukkan sebuah angka: ");
8          int angka = scanner.nextInt();
9          System.out.println("Angka yang dimasukkan: " + angka);
10
11         System.out.print("Masukkan sebuah teks: ");
12         String teks = scanner.next();
13         System.out.println("Teks yang dimasukkan: " + teks);
14
15         // Menutup objek Scanner setelah selesai digunakan
16         scanner.close();
17     }
18 }
```

Output:

```
Masukkan sebuah angka: 7
Angka yang dimasukkan: 7
Masukkan sebuah teks: Hello
Teks yang dimasukkan: Hello
```

## Create Object (Constructor)

Constructor adalah metode khusus yang digunakan untuk membuat dan menginisialisasi objek dari kelas tertentu. Constructor memiliki nama yang sama dengan nama kelasnya dan tidak mengembalikan nilai. Constructor ini dipanggil saat objek baru dibuat dari kelas yang bersangkutan. Constructor digunakan untuk menginisialisasi atribut (variabel anggota) objek dan melakukan pekerjaan awal yang diperlukan saat objek dibuat.

Contoh deklarasi constructor :

```
1 public class Mobil {
2     String merek;
3     int tahunProduksi;
4
5     // Constructor dengan parameter
6     public Mobil(String merek, int tahunProduksi) {
7         // Menginisialisasi atribut merek dengan nilai dari parameter merek
8         this.merek = merek;
9         // Menginisialisasi atribut tahunProduksi dengan nilai dari parameter tahunProduksi
10        this.tahunProduksi = tahunProduksi;
11    }
12
13    public void infoMobil() {
14
15        System.out.println("Merek: " + merek);
16        System.out.println("Tahun Produksi: " + tahunProduksi);
17    }
18
19    public static void main(String[] args) {
20        // Membuat objek mobil1 dengan merek "Toyota" dan tahun produksi 2020
21        Mobil mobil1 = new Mobil("Toyota", 2020);
22        // Membuat objek mobil2 dengan merek "Honda" dan tahun produksi 2019
23        Mobil mobil2 = new Mobil("Honda", 2019);
24
25        // Memanggil metode infoMobil() untuk objek mobil1
26        mobil1.infoMobil();
27        // Memanggil metode infoMobil() untuk objek mobil2
28        mobil2.infoMobil();
29    }
30 }
31
```

Output:

```
Merek: Toyota
Tahun Produksi: 2020
Merek: Honda
Tahun Produksi: 2019
```

## Array List

Array list pada Java adalah kelas yang digunakan untuk menyimpan data berupa list objek berbentuk array yang ukurannya dapat berubah secara dinamis sesuai dengan jumlah data yang kita masukkan.

Array list bersifat dinamis dan dapat dimodifikasi, berbeda dengan array biasa yang ukurannya tetap dan tidak dapat diubah.

Array list merupakan bagian dari Java Util, yang merupakan kumpulan kelas dan antarmuka yang menyediakan berbagai operasi untuk memanipulasi koleksi data

Contoh penggunaan Array List :

```
1 import java.util.ArrayList; // mengimpor kelas ArrayList
2
3 public class ContohArrayList {
4     public static void main(String[] args) {
5         // Membuat ArrayList untuk menyimpan nama-nama
6         ArrayList<String> daftarNama = new ArrayList<String>();
7
8         // Menambahkan nama-nama ke ArrayList
9         daftarNama.add("Fahru");
10        daftarNama.add("Adin");
11        daftarNama.add("Abi");
12        daftarNama.add("Yoga");
13
14        // Mengakses elemen-elemen dalam ArrayList
15        System.out.println("Nama di indeks 0: " + daftarNama.get(0));
16        System.out.println("Nama di indeks 2: " + daftarNama.get(2));
17
18        // Mengubah elemen dalam ArrayList
19        daftarNama.set(1, "Aldean");
20        System.out.println("Nama di indeks 1 setelah diubah: " + daftarNama.get(1));
21
22        // Menghapus elemen dari ArrayList
23        daftarNama.remove(3);
24        System.out.println("Setelah menghapus elemen di indeks 3:");
25        for (String nama : daftarNama) {
26            System.out.println(nama);
27        }
28    }
29 }
30
```

Output:

```
Nama di indeks 0: Fahru
Nama di indeks 2: Abi
Nama di indeks 1 setelah diubah: Aldean
Setelah menghapus elemen di indeks 3:
Fahru
Aldean
Abi
```



## Perulangan (Loop)

Looping (perulangan) adalah sebuah konsep dalam bahasa pemrograman Java (dan banyak bahasa pemrograman lainnya) yang memungkinkan Anda untuk menjalankan sekelompok pernyataan atau blok kode secara berulang berdasarkan kondisi tertentu. Looping digunakan untuk mengeksekusi perintah yang sama beberapa kali, sehingga menghemat waktu dan menghindari penulisan kode yang berulang.

Di Java, ada beberapa jenis loop yang sering digunakan, di antaranya adalah for, dan while loop. Berikut penjelasan singkat tentang masing-masing loop:

### For Loop

Loop for digunakan ketika Anda tahu berapa kali Anda ingin menjalankan blok kode. Biasanya terdiri dari tiga bagian: inisialisasi, kondisi, dan perubahan

Contoh penggunaan for loop :

```
public class ForLoopExample {  
    public static void main(String[] args) {  
  
        // Contoh for loop dengan increment  
        for (int i = 1; i ≤ 5; i++)  
            System.out.println("Iterasi ke-" + i);  
  
        // Contoh for loop dengan decrement  
        for (int j = 5; j ≥ 1; j--)  
            System.out.println("Iterasi ke-" + j);  
  
        // Contoh for loop untuk mengiterasi elemen dalam array  
        int[] numbers = {1, 2, 3, 4, 5};  
        for (int number : numbers)  
            System.out.println("Elemen: " + number);  
  
        // Contoh for loop dengan langkah penambahan yang berbeda  
        for (int m = 0; m ≤ 10; m += 2)  
            System.out.println(m);  
    }  
}
```

Output :

```
Iterasi ke-1  
Iterasi ke-2  
Iterasi ke-3  
Iterasi ke-4  
Iterasi ke-5  
Iterasi ke-5  
Iterasi ke-4  
Iterasi ke-3  
Iterasi ke-2  
Iterasi ke-1  
Elemen: 1  
Elemen: 2  
Elemen: 3  
Elemen: 4  
Elemen: 5  
0  
2  
4  
6  
8  
10
```

## While Loop

Loop while digunakan ketika Anda ingin menjalankan blok kode selama suatu kondisi tertentu terpenuhi. Kondisi diuji sebelum eksekusi blok kode, dan jika kondisi benar, blok kode akan diulang.

Contoh penggunaan while loop :

Output :

```
public class WhileLoopExample {
    public static void main(String[] args) {

        // Contoh while loop dengan kondisi
        int i = 1;
        while (i <= 5)
            System.out.println("Iterasi ke-" + i++);

        // Contoh while loop dengan kondisi yang menggunakan break
        int j = 10;
        while (true) {
            System.out.println("Iterasi ke-" + j);
            if (--j == 5)
                break;
        }

        // Contoh while loop dengan kondisi awal yang tidak terpenuhi
        int k = 1;
        while (k > 5) {
            System.out.println("Iterasi ke-" + k);
            k++;
        }

        // Contoh while loop untuk membaca input dari pengguna
        Scanner scanner = new Scanner(System.in);
        String input = "";
        while (!(input = scanner.nextLine()).equalsIgnoreCase("exit"))
            System.out.println("Anda memasukkan: " + input);
        scanner.close();
    }
}
```

```
Iterasi ke-1
Iterasi ke-2
Iterasi ke-3
Iterasi ke-4
Iterasi ke-5
Iterasi ke-10
Iterasi ke-9
Iterasi ke-8
Iterasi ke-7
Iterasi ke-6
Anda memasukkan: Hello
Anda memasukkan: World
Anda memasukkan: exit
```

## Arraylist Loop

Dalam Java, ArrayList adalah salah satu jenis dari banyak koleksi yang digunakan untuk menyimpan dan mengelola sejumlah objek. Anda dapat menggunakan loop, baik dengan for-each atau for dengan indeks, untuk mengakses elemen-elemen dalam ArrayList. Berikut adalah penjelasan tentang cara melakukannya:

### For-Each Loop

Loop for-each adalah cara yang nyaman untuk mengulangi semua elemen dalam ArrayList tanpa perlu mengkhawatirkan indeks atau ukuran. Ini digunakan ketika Anda hanya perlu mengakses nilai dari setiap elemen, bukan indeksnya.

Contoh penggunaan for-each loop :

```
public class ForEachLoopExample {  
    public static void main(String[] args) {  
        int[] numbers = {1, 2, 3, 4, 5};  
  
        // Mencetak elemen-elemen dalam array menggunakan foreach loop  
        for (int number : numbers) {  
            System.out.println(number);  
        }  
  
        // Contoh penggunaan foreach loop pada koleksi ArrayList  
        List<String> names = new ArrayList<>();  
        names.add("Farel");  
        names.add("Yoga");  
        names.add("Danu");  
  
        // Mencetak elemen-elemen dalam ArrayList menggunakan foreach loop  
        for (String name : names) {  
            System.out.println(name);  
        }  
    }  
}
```

Output :

```
1  
2  
3  
4  
5  
Farel  
Yoga  
Danu
```

## For Index Loop

Anda juga dapat menggunakan loop **for** dengan indeks untuk mengakses elemen dalam ArrayList. Ini berguna jika Anda perlu mengakses elemen berdasarkan indeksnya atau melakukan perubahan elemen.

Contoh penggunaan for index loop :

Output:

```
public class ForIndexLoopExample {
    public static void main(String[] args) {

        // Contoh penggunaan foreach loop pada array
        int[] numbers = {1, 2, 3, 4, 5};
        for (int number : numbers) {
            System.out.println("Elemen: " + number);
        }

        // Contoh penggunaan foreach loop pada string
        String message = "Welcome to EAD Lab!";
        for (char character : message.toCharArray()) {
            System.out.println("Karakter: " + character);
        }

        // Contoh penggunaan foreach loop pada koleksi ArrayList
        List<String> names = new ArrayList<>();
        names.add("Farel");
        names.add("Yoga");
        names.add("Danu");
        for (String name : names) {
            System.out.println("Nama: " + name);
        }
    }
}
```

```
Elemen ke-1: 1
Elemen ke-2: 2
Elemen ke-3: 3
Elemen ke-4: 4
Elemen ke-5: 5
Karakter ke-1: W
Karakter ke-2: e
Karakter ke-3: l
Karakter ke-4: c
Karakter ke-5: o
Karakter ke-6: m
Karakter ke-7: e
Karakter ke-8:
Karakter ke-9: t
Karakter ke-10: o
Karakter ke-11:
Karakter ke-12: E
Karakter ke-13: A
Karakter ke-14: D
Karakter ke-15:
Karakter ke-16: L
Karakter ke-17: a
Karakter ke-18: b
Karakter ke-19: !
Nama ke-1: Farel
Nama ke-2: Yoga
Nama ke-3: Danu
```

**“Never give up. Today is hard, tomorrow will be worse, but the day after tomorrow will be sunshine”**

find us on :

 @eadlaboratory

 @ozc7189g