

LAPORAN RESMI
MODUL II
(CONSTRUCTOR DAN KEYWORD STATIC)
PEMROGRAMAN BERBASIS OBJEK



NAMA	: CYNDIFANITA B'THARI MARSHA
N.R.P	: 230441100109
DOSEN	: AHMAD ZAIN NUR S.Kom M.Kom
ASISTEN	: SYAHRUL RAMADHANI
TGL PRAKTIKUM	: 26 MARET 2023

Disetujui :... 2023
Asisten

SYAHRUL RAMADHANI
NIM. 22.04.411.00128



LABORATORIUM BISNIS INTELIJEN SISTEM
PRODI SISTEM INFORMASI
JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS TRUNOJOYO MADURA

BAB I

PENDAHULUAN

1.1 Latar Belakang

Constructor adalah metode khusus dalam sebuah kelas yang secara otomatis dipanggil saat objek dari kelas tersebut dibuat. Constructor biasanya digunakan untuk melakukan inisialisasi awal objek, seperti menetapkan nilai awal untuk atribut-atribut objek. Kata kunci static digunakan untuk menyatakan bahwa suatu metode, variabel, atau blok kode merupakan milik kelas (bukan objek) dan dapat diakses langsung dari kelas tersebut tanpa perlu membuat objek dari kelas tersebut.

Kedua konsep ini sangat penting dalam pemrograman berorientasi objek, karena constructor digunakan untuk menginisialisasi objek saat pembuatannya, sementara variabel atau metode static digunakan untuk menyimpan informasi bersama atau menyediakan perilaku yang terkait dengan kelas itu sendiri, bukan dengan setiap objek individu yang dibuat dari kelas tersebut.

1.2 Tujuan

- Mahasiswa mampu memahami konsep Constructor dalam Pemrograman Berorientasi Objek serta mampu mengimplementasikannya.
- Mahasiswa mampu memahami penggunaan keyword static dalam Pemrograman Berorientasi Objek serta mampu mengimplementasikannya.

BAB II

DASAR TEORI

2.1 Constructor

Constructor adalah method yang secara otomatis dipanggil/dijalankan pada saat `new` dipakai untuk menciptakan instan kelas. Atau dengan kata lain constructor adalah method yang pertama kali dijalankan pada saat sebuah objek pertama kali diciptakan. Jika dalam sebuah class tidak terdapat constructor, maka secara otomatis Java akan membuatkan sebuah default constructor. Sama halnya dengan method, constructor dapat memiliki satu atau banyak parameter maupun tanpa parameter. Hal mendasar yang perlu diperhatikan, yaitu :

- a. Nama Constructor sama dengan nama Class.
- b. Tidak ada return type yang diberikan kedalam Constructor Signature.
- c. Tidak ada return statement, di dalam tubuh constructor

a. Contoh ConstructorMotor.java

```
1 public class ConstructorKendaraan {
2     private String merk; // ini adalah instant variable
3     private static String pemilik; // ini adalah static variable
4
5     // ini adalah constructor tanpa parameter
6     protected ConstructorKendaraan() {
7         merk = null;
8     }
9
10    // overloading constructor
11    // ini adalah constructor dengan parameter merk
12    protected ConstructorKendaraan(String merk) {
13        this.merk = merk;
14        merk = null;
15    }
16
17    protected void setMerk(String merk) {
18        this.merk = merk;
19    }
20
21    protected String getMerk() {
22        return merk;
23    }
24
25    // ini adalah static method
26    protected static void setPemilik(String pemilik) {
27        ConstructorKendaraan.pemilik = pemilik;
28    }
29
30    // ini adalah static method
31    protected static String getPemilik() {
32        return ConstructorKendaraan.pemilik;
33    }
34
35    protected void tampil(String a)
36    { System.out.println(a);
37      a = null;
38    }
39
40    protected void hapus()
41    { // menghapus variable private dari memory
42      merk = null;
43      pemilik = null;
44    }
45 }
```

```
1 // ConstructorMotor turunan dari class ConstructorKendaraan
2 public class ConstructorMotor extends ConstructorKendaraan {
3     private String warna; // ini adalah instant variable
4
5     // ini adalah constructor dengan parameter merk & warna
6     protected ConstructorMotor(String merk, String warna) {
7         // memanggil constructor parent (class ConstructorKendaraan)
8         // dengan keyword super dan parameter merk
9         super(merk);
10
11        this.warna = warna;
12
13        // menghapus variable parameter dari memory
14        merk = null;
15        warna = null;
16    }
17
18    protected String getWarna() {
19        return warna;
20    }
21
22    protected void hapus()
23    { // menghapus variable private dari memory
24      warna = null;
25      // memanggil method hapus class parent (class ConstructorKendaraan)
26      // dengan keyword super
27      super.hapus();
28    }
29 }
```

b. MainConstructorMotor.java

```
1 public class MainConstructorMotor {
2     public static void main(String args[])
3     {
4         String pemilik = "Ahmad Afif";
5         String merk = "Honda";
6         String warna = "Merah";
7
8         // cara akses static variable dan static method dapat dengan
9         // memanggil class (ConstructorKendaraan) secara langsung
10        ConstructorKendaraan.setPemilik(pemilik);
11        System.out.println("Pemilik Kendaraan = "+ConstructorKendaraan.getPemilik());
12        System.out.println("=====");
13
14        // variable merk menjadi parameter Constructor pada saat
15        // instansiasi/membuat objek baru (ob)
16        ConstructorKendaraan ob = new ConstructorKendaraan(merk);
17        ob.tampil("Merk Kendaraan = "+ob.getMerk());
18        // cara akses static variable dan static method dapat juga dengan
19        // objek (ob) pada method getPemilik()
20        ob.tampil("Pemilik Kendaraan = "+ob.getPemilik());
21        System.out.println("=====");
22
23        // instansiasi/membuat objek baru (ob2) tanpa parameter
24
25        ConstructorKendaraan ob2 = new ConstructorKendaraan();
26        // bandingkan akses untuk menampilkan nilai pada instant variable (merk)
27        // dan static variable (pemilik) melalui method getter setelah membuat
28        // objek baru "ob2", dimana sebelumnya juga sudah membuat objek "ob".
29        // instant variable (merk) nilainya akan hilang,
30        // sedangkan static variable (pemilik) nilainya tidak hilang/berubah
31        ob2.tampil("Merk Kendaraan (instant variable) = "+ob2.getMerk());
32        ob2.tampil("Pemilik Kendaraan (static variable) = "+ob2.getPemilik());
33        System.out.println("=====");
34
35        // variable merk dan warna menjadi parameter Constructor pada saat
36        // instansiasi/membuat objek baru (ob3)
37        ConstructorMotor ob3 = new ConstructorMotor(merk, warna);
38        ob3.tampil("Merk Motor = "+ob3.getMerk());
39        ob3.tampil("Warna Motor = "+ob3.getWarna());
40        ob3.tampil("Pemilik Motor = "+ob3.getPemilik());
41
42        pemilik = null;
43        merk = null;
44        warna = null;
45        ob.hapus();
46        ob = null;
47        ob2 = null;
48        ob3 = null;
49    }
50 }
```

2.2 Keyword Static

Dalam pemrograman java, keyword static digunakan untuk mengakses variable ataupun method (prosedur atau fungsi) pada class tertentu tanpa harus membuat suatu objek dari class itu. Umumnya untuk mengakses member dari kelas lain kita harus membuat objek kelas, tapi dengan menggunakan keyword static kita dapat langsung menggunakan member kelas lain. Keyword static bisa digunakan untuk variable ataupun method.

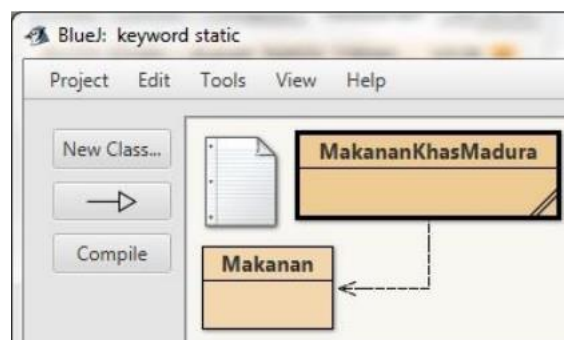
Static member adalah variable atau method yang bukan milik dari suatu object, melainkan milik dari class.

Bentuk Umum:

```
static void namaMethodStatic(){
```

Method yang dideklarasikan sebagai static memiliki aturan sebagai berikut :

- Hanya dapat dipanggil oleh method lain yang juga adalah static method.
- Hanya dapat mengakses atribut static.
- Tidak dapat menggunakan keyword this dan super, karena kedua keyword inimenunjuk ke suatu instance tertentu, bukan pada sebuah object.

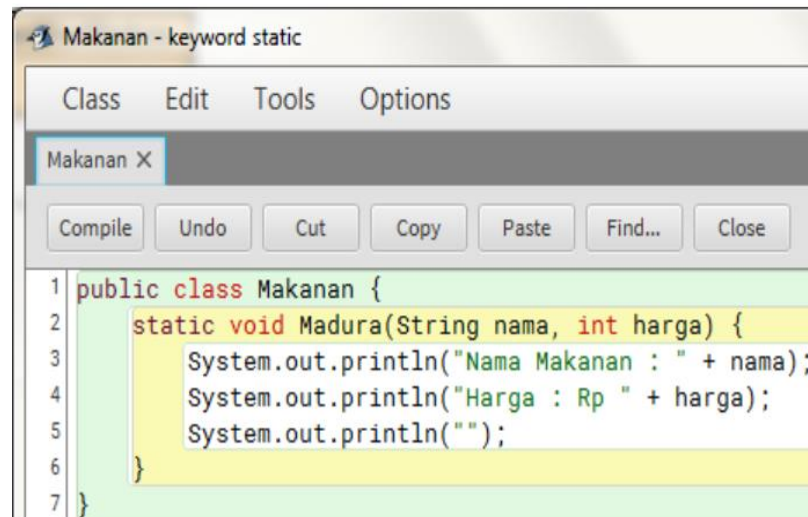


- Class Makanan
- Class MakananKhasMadura

Tampilan Class

Class Makanan

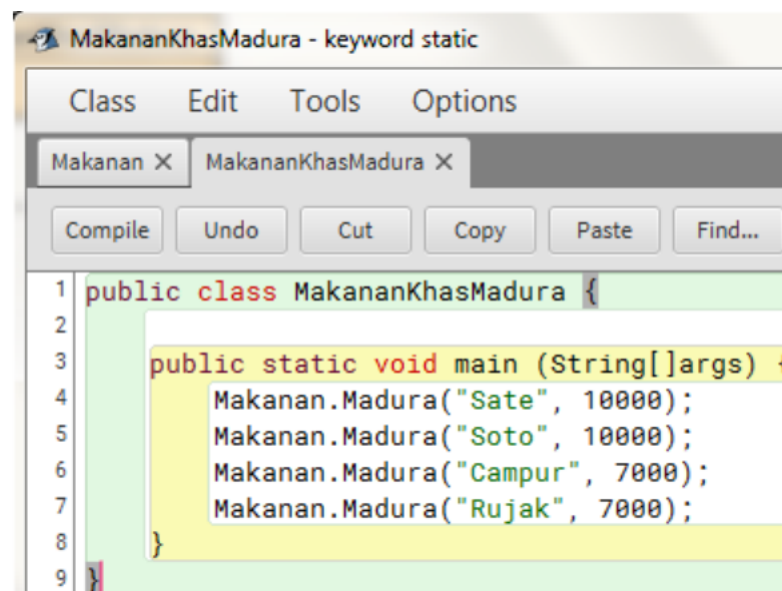
Pada class Makanan, method di set sebagai static dengan parameter makanan dan harga.



```
1 public class Makanan {
2     static void Madura(String nama, int harga) {
3         System.out.println("Nama Makanan : " + nama);
4         System.out.println("Harga : Rp " + harga);
5         System.out.println("");
6     }
7 }
```

Class Makanan Khas Madura

Pada class Makanan Khas Madura, method juga bersifat static tanpa membuat instance objek dari kelas Makanan.



```
1 public class MakananKhasMadura {
2
3     public static void main (String[] args) {
4         Makanan.Madura("Sate", 10000);
5         Makanan.Madura("Soto", 10000);
6         Makanan.Madura("Campur", 7000);
7         Makanan.Madura("Rujak", 7000);
8     }
9 }
```

Running Program

Pada saat program di eksekusi, maka akan menampilkan daftar makanan dan harga yang telah diset nilainya. Berikut tampilannya!



```
BlueJ: Terminal Window - keyword static
Options
Nama Makanan : Sate
Harga : Rp 10000

Nama Makanan : Soto
Harga : Rp 10000

Nama Makanan : Campur
Harga : Rp 7000

Nama Makanan : Rujak
Harga : Rp 7000
```

Penjelasan Program

Class Makanan, Pada class Makanan, terdapat method static dengan parameter makanan dan harga.

- i. Baris 2, keyword static yang berupa method
- ii. Baris 3-4, mencetak nama makanan dan harga.

Class Makanan Khas Madura

Pada class Makanan Khas Madura, method juga bersifat static tanpa membuat instance objek dari kelas Makanan.

- i. Baris 3, method static untuk menjalankan program.
- ii. Baris 4-7, nilai dari parameter yang terdapat pada class Makanan.

BAB III

TUGAS PENDAHULUAN

3.1 Soal

1. Jelaskan perbedaan constructor dan keyword static!
2. Jelaskan setter dan getter menurut bahasa pemahamamu!

3.2 Jawaban

1. Konstruktor adalah method yang secara otomatis dipanggil atau dijalankan pada saat new dipakai untuk menciptakan instan kelas sedangkan keyword static digunakan untuk mengakses variable ataupun method pada class tersebut tanpa harus membuat suatu objek dari class itu.
2. Setter digunakan untuk mengubah nilai sedangkan getter digunakan untuk mengambil nilai.

BAB IV

IMPLEMENTASI

4.1 Source code

1. Program mahasiswa2

```
public class MahasiswaMODUL2 {

    private String Nama, NIM, Jurusan, Alamat;

    //Konstruktor
    public MahasiswaMODUL2(String Namaa, String NIMM, String
Jurusan, String Alamatt){
        this.Nama = Namaa;
        this.NIM = NIMM;
        this.Jurusan = Jurusan;
        this.Alatamat = Alamatt;
    }

    //Getter
    public String getNama(){
        return Nama;
    }

    public String getNIM(){
        return NIM;
    }

    public String getJurusan(){
        return Jurusan;
    }

    public String getAlamat(){
        return Alamat;
    }

    //Setter
    public void setNama(String Nama){
        this.Nama = Nama;
    }

    public void setNIM(String NIM){
        this.NIM = NIM;
    }
}
```

```

    public void setJurusan(String Jurusan){
        this.Jurusan = Jurusan;
    }

    public void setAlamat(String Alamat){
        this.Alatmat = Alamat;
    }

    public static void main(String[] args) {
        MahasiswaMODUL2 mhs = new MahasiswaMODUL2("Fanita",
"230441100109", "Sistem Informasi", "bojonegoro");
        System.out.println("Nama: " + mhs.getNama());
        System.out.println("NIM: " + mhs.getNIM());
        System.out.println("Jurusan: " + mhs.getJurusan());
        System.out.println("Alamat: " + mhs.getAlamat());
    }
}

```

2.Program MahasiswaSoal2

```

package MahasiswaModul2Soal2;

/**
 *
 * @author HP
 */
public class MahasiswaModul2Soal2 {
    String namaa;
    String niim;
    String jurusanProdi;
    String alamatt;
    String[] ukmm; // Atribut untuk menyimpan UKM

    static String universitas;

    public MahasiswaModul2Soal2(String nama, String nim, String
        jurusanProdi, String alamat) {
        namaa = nama;
        niim = nim;
        jurusanProdi = jurusanProdi;
        alamatt = alamat;
    }
}

```

```

// Getter dan setter untuk atribut non-static
public String getNama() {
    return namaa;
}

public void setNama(String nama) {
    namaa = nama;
}

public String getNim() {
    return niim;
}

public void setNim(String nim) {
    niim = nim;
}

public String getJurusanProdi() {
    return jurusanProdii;
}

public void setJurusanProdi(String jurusanProdi) {
    jurusanProdii = jurusanProdi;
}

public String getAlamat() {
    return alamatt;
}

public void setAlamat(String alamat) {
    alamatt = alamat;
}

// Getter dan setter untuk atribut static universitas
public static String getUniversitas() {
    return universitas;
}

public static void setUniversitas(String universitas) {
    MahasiswaModul2Soal2.universitas = universitas;
}

// Getter dan setter untuk atribut ukm
public String[] getUkm() {
    return ukmm;
}

```

```

    }

    public void setUkm(String[] ukm) {
        ukmm = ukm;
    }

    // Method static untuk menampilkan nilai atribut static universitas
    public static void displayUniversitas() {
        System.out.println("Universitas: " + universitas);
    }
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        // Pengaturan nilai atribut static universitas
        MahasiswaModul2Soal2.setUniversitas("Universitas ABC");

        System.out.println("Universitas: " + universitas);

        // Membuat objek mahasiswa
        MahasiswaModul2Soal2 mhs = new MahasiswaModul2Soal2("John
        Doe", "123456", "Teknik Informatika", "Jl. Contoh No. 123");

        // Menambahkan UKM
        String[] ukm = {"ITC", "pramuka"};
        mhs.setUkm(ukm);

        // Menampilkan informasi mahasiswa
        System.out.println("Nama: " + mhs.getNama());
        System.out.println("NIM: " + mhs.getNim());
        System.out.println("Jurusan/Prodi: " + mhs.getJurusanProdi());
        System.out.println("Alamat: " + mhs.getAlamat());

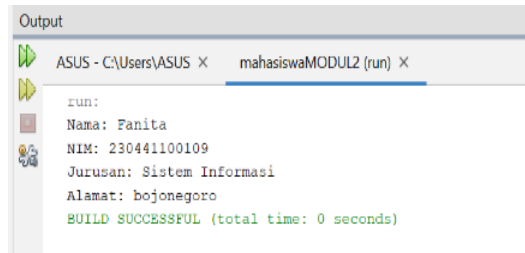
        // Menampilkan UKM yang diikuti mahasiswa
        System.out.println("UKM:");
        for (String namaUKM : mhs.getUkm()) {
            System.out.println("- " + namaUKM);
        }

        // Menampilkan nilai atribut static universitas
        MahasiswaModul2Soal2.displayUniversitas();
    }
}

```

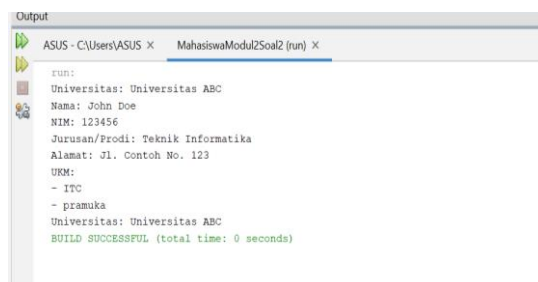
4.2 Hasil

1. Hasil mahasiswa2



```
Output
ASUS - C:\Users\ASUS X mahasiswaMODUL2 (run) X
run:
Nama: Fanita
NIM: 230441100109
Jurusan: Sistem Informasi
Alamat: bojonegoro
BUILD SUCCESSFUL (total time: 0 seconds)
```

2. Hasil mahasiswaSoal2



```
Output
ASUS - C:\Users\ASUS X MahasiswaModul2Soal2 (run) X
run:
Universitas: Universitas ABC
Nama: John Doe
NIM: 123456
Jurusan/Prodi: Teknik Informatika
Alamat: Jl. Contoh No. 123
URM:
- ITC
- pramuka
Universitas: Universitas ABC
BUILD SUCCESSFUL (total time: 0 seconds)
```

Penjelasan :

- Melalui implementasi kelas Mahasiswa2 dalam bahasa pemrograman Java, kita dapat memperoleh pemahaman yang lebih baik tentang konsep dasar pemrograman berorientasi objek.
- Penggunaan atribut, metode, dan konstruktor dalam kelas tersebut memungkinkan kita untuk merepresentasikan dan mengelola informasi terkait mahasiswa dengan lebih terstruktur. Dengan adanya getter dan setter, kita dapat mengakses dan mengubah nilai atribut secara terkontrol, sehingga memungkinkan fleksibilitas dalam manipulasi data.
- Selain itu, penggunaan atribut static universitas memperlihatkan bagaimana nilai yang bersifat universal dapat dibagikan di antara seluruh instance dari kelas yang sama.
- Keseluruhan implementasi ini memberikan gambaran tentang cara memodelkan dan mengorganisir entitas dalam program, serta bagaimana berbagai konsep dasar pemrograman objek, seperti enkapsulasi dan penggunaan array, dapat diterapkan dalam praktek.

BAB V

PENUTUP

5.1 Analisa

Analisis konstruktor dan kata kunci static dalam pemrograman berorientasi objek (OOP) memberikan pemahaman yang mendalam tentang bagaimana kelas bekerja, bagaimana objek dibuat, dan bagaimana data dan perilaku bersama dikelola. Konstruktor bertanggung jawab untuk menginisialisasi objek dari suatu kelas. Ini dilakukan saat objek pertama kali dibuat dengan menggunakan operator new. : Metode static sering digunakan untuk menyediakan fungsi utilitas yang dapat diakses tanpa perlu membuat objek baru. Contohnya adalah metode utilitas matematika, operasi string, dan sebagainya.

Variabel static digunakan untuk menyimpan data bersama yang dibagikan oleh semua objek dari kelas tersebut. Misalnya, hitungan jumlah objek yang telah dibuat dari kelas, atau konstanta yang sama untuk semua objek.

5.2 Kesimpulan

- 1 Konstruktor dan kata kunci static adalah fitur kunci dalam OOP yang menyediakan cara untuk mengelola inisialisasi objek dan data bersama di antara objek-objek dari kelas yang sama.
- 2 Konstruktor memungkinkan inisialisasi objek dengan nilai-nilai awal yang diperlukan, sedangkan static digunakan untuk menyediakan variabel dan metode yang bersama-sama digunakan oleh semua objek dari kelas.
- 3 Dengan menggunakan konstruktor dan kata kunci static dengan bijaksana, pengembang dapat membangun aplikasi yang lebih efisien dan mudah dimengerti dengan memanfaatkan paradigma pemrograman berorientasi objek secara penuh.