



MODUL PERKULIAHAN #7 **CLASS, OBJECT DAN PACKAGE**

Capaian Pembelajaran	:	Mahasiswa Mengerti dan Mampu: Menuliskan penggunaan class dan object serta manage class kedalam package dalam Penulisan kode program dengan Bahasa Java
Sub Pokok Bahasan	:	<ol style="list-style-type: none">1. Deklarasi Class, Atribut Class, Deklarasi Method, Pembuatan Objek, Akses Anggota Class, Keyword This2. Package, Pengaturan Class dalam Package, Hak Akses

Daftar Pustaka	:	<ol style="list-style-type: none"> 1. Farrell. Joyce, An Object-Oriented Approach to Programming Logic and Design 3rd Edition, Course Technology, 2009 2. Thomas Wu. C, An Introduction to Object-Oriented Programming with Java™ 4th Edition, McGraw Hill, 2006 3. Deitel, Java How To Program, Deitel, Prentice Hall, 2008 4. Anif, M, Seri Aplikasi : Aplikasi Penjualan (Pemesanan Barang) dengan Java, Mitra Wacana Media, 2008
----------------	---	--

PRAKTIKUM 7

CLASS, OBJECT DAN PACKAGE

7.1 Teori Class, Object dan Package

Class merupakan suatu *blueprint* atau cetakan untuk menciptakan suatu instan dari objek. *Class* juga merupakan grup suatu objek dengan kemiripan *attributes* atau *properties*, behavior dan relasi ke objek lain.

Bentuk Umum:

```
[modifier 1] class namaKelas [modifier 2] {  
    // body class  
}
```

Class dan **Method** terdiri dari:

a. **Class dan Method Standar**

Kelas dan method Standar merupakan *predefined class*, yaitu *class* yang disediakan oleh Java dan menjadi referensi library. Banyak *class* standar yang dapat digunakan untuk aplikasi yang spesifik. Kumpulan class ini sering dikenal dengan istilah API (*Application Programming Interface*).

b. **Class dan Method yang didefinisikan Sendiri *User Defined Function* (UDF)**

Selain kelas dan method standar, terkadang kita juga perlu memodelkan suatu objek kedalam kelas dan mendefenisikan data serta method yang dimilikinya. Bila aplikasi tersebut besar dan kompleks, kita dapat membaginya menjadi beberapa package yang didalamnya terdapat beberapa kelas.

Deklarasi *class*

a. Deklarasi *class* Sederhana

```
class namaClass{  
    // body class  
}
```

b. Deklarasi *class* Lengkap

```
modifier1 tipeData namaClass modifier2 [namaClass/namaInterface...]{  
    // body class  
}
```

c. *Modifier* Pada Kelas

Untuk menentukan sifat dari suatu kelas dan menentukan *previlage* (hak akses) dari kelas lain.

1) Modifier 1

public : Modifier ini dapat diakses dari kelas lain, baik dalam *package* yang sama maupun berbeda.

private : Modifier ini tidak dapat diakses sama sekali dari kelas lain, baik dari *package* yang sama maupun berbeda. (**hanya untuk kelas dan *package* yang sama**)

protected : Modifier ini membatasi akses kelas yang dilakukan oleh subkelas turunannya dan kelas lain yang terletak dalam *package* yang sama.

abstract : Dalam modifier ini, kelas tersebut tidak dapat diinstankan langsung menjadi objek. Dipakai pada Hirarki kelas tertinggi yang hanya mungkin dilakukan dengan cara inheritance. Atau dengan kata lain, kelas murni tanpa objek dan tidak boleh memiliki objek

final : Dalam modifier ini, kelas tersebut tidak dapat diturunkan menjadi subkelas.

Tabel Akses:

No	Modifier	Class Sama	Paket Sama		Paket Berbeda	
			Extends	Instan	Extends	Instan
1	public	✓	✓	✓	✓	✓
2	protected	✓	✓	✓	✓	
3	default	✓	✓	✓		
4	private	✓				

2) Modifier 2

Untuk menentukan relasi (extend atau implement) dengan kelas lainnya.

extends : Modifier ini digunakan untuk inheritance/pewarisan.

SuperClass Bila terjadi pewarisan, kelas yang mewariskan method dan atributnya disebut kelas super, sedangkan yang diwariskan disebut subkelas

implement : Modifier ini digunakan bila kelas meng-implementasikan
Interface satu atau lebih interface

Deklarasi Method

Method merupakan fungsi yang diciptakan untuk melakukan tugas tertentu.

Bentuk umum:

```
modifier tipeNilaiKembalian namaMethod(parameter input) throws exception{  
    // body dari method  
}
```

Modifier (optional) yang digunakan sama dengan *modifier* untuk mendeklarasikan kelas karena pada prinsipnya modifier dapat diletakkan pada kelas, data dan method. Namun implikasi dari penggunaan modifier pada ketiganya belum tentu sama persis antara satu dengan yang lainnya.

Sebagai contoh:

1. **final** pada kelas berarti bahwa kelas itu tidak dapat diturunkan menjadi subkelas yang lain.
2. **final** pada method, berarti method tersebut tidak dapat dioverride oleh subkelas yang lain.
3. **final** pada variabel akan mengubah variabel tersebut menjadi sebuah konstanta.

Memanggil Method

Sama seperti ketika mengakses variabel, kita juga menggunakan notasi titik/dot (.) untuk memanggil *method*.

Bentuk umum pemanggilan *method* sbb:

```
namaObjek.namaMethod(argument);
```

Contoh:

```
objMahasiswa.isiMataKuliah();
```

Melewatkan Argumen ke Method

Dua cara melewati argument ke method, yaitu:

1. Melewatkan secara Nilai (*Pass by Value*)

Diterapkan pada *argument* bertipe data primitif (tipe data standar yang tidak diturunkan dari objek manapun). Prosesnya adalah compiler hanya menyalin/mengcopy isi memori (yang telah dialokasikan untuk suatu variabel) dan kemudian menyampaikan salinan tersebut kepada *method*. oleh karena itu, maka perubahan yang terjadi pada variabel akibat proses tidak akan berpengaruh pada nilai variabel asalnya

2. Melewatkan secara Referensi (*Pass by Reference*)

Diterapkan pada array dan objek. Variabel array dan objek menyimpan alamat memori bukan isi memori. Akibatnya setiap perubahan variabel didalam method akan mempengaruhi nilai pada variabel asalnya.

Method Overloading

Java memperbolehkan kita membuat dua atau lebih method dengan nama yang sama dalam suatu kelas. Meskipun memiliki nama sama namun jumlah dan tipe data argument masing-masing method haruslah berbeda satu dengan lainnya. Inilah yang disebut dengan Overloading Method.

Contoh:

```
class mencetakArgument{
    void Cetak(int bil){ //Memiliki Nama Sama
        System.out.println("Nilai yang dikirim adalah "+bil);
    }

    void Cetak(String nama){ //Memiliki Nama Sama
        System.out.println("Mana yang dikirim adalah "+nama);
    }
}
```

Method Konstrutor

Konstrutor adalah method yang berfungsi untuk menginisialisasi variabel-variabel instant yang akan dimiliki objek. Konstrutor ini dipanggil pada saat proses instansiasi kelas menjadi objek.

Ketentuan sbb:

1. Namanya sama dengan nama Kelasnya.
2. Tidak mengembalikan suatu nilai
3. Satu kelas bisa memiliki lebih dari satu konstrutor, konstrutor satu dapat memanggil konstrutor lain.
4. Dapat dibubuhi modifier public, private, protected

Contoh:

```
class Cetak{ //Memiliki Nama Sama
    public Cetak(int bil){ //Memiliki Nama Sama
        System.out.println("Nilai yang dikirim adalah "+bil);
    }

    public Cetak(String nama){ //Memiliki Nama Sama
        System.out.println("Mana yang dikirim adalah "+Nama);
    }
}
```

Method Finalizer

Objek adalah dalam program yang sedang dieksekusi mempunyai waktu hidup (*life time*). Objek tercipta pada saat kita menginstankan suatu kelas dengan operator new dan akan dihapus pada saat objek dikumpulkan oleh *Garbage Collection* atau bila memori yang ditempatinya telah diklaim oleh objek/bagian program lain. Method yang dibubuhi *modifier finalize* ini dapat dikatakan juga sebagai lawan/kebalikan dari method konstrutor. Method ini dipanggil sesaat sebelum objek dihancurkan.

Bentuk Umum:

```
protected void finalize() throw Throwable{
    super.finalize();
}
```

Method Main

Setelah selesai mengetik source code, langkah berikutnya adalah mengkompilasi dan menjalankan program tersebut. Pada saat kompilasi, pertama-tama kompiler akan mencari bagian program yang disebut sebagai method utama (main method).

Bentuk Umum:

```
public static void main(String [] arguments){  
    // statement body dari main method  
}
```

1. **Public**, main method harus dapat dilihat atau visible oleh kelas manapun.
2. **Void**, tipe yg digunakan untuk mendeklarasikan sebuah method dan tidak mngembalikan nilai.
3. **Main(String[] arguments)**, main method ini dapat mengambil suatu parameter input yang merupakan array dari string yang diberi nama argument.

7.2 Objek

Untuk membuat sebuah objek dari sebuah *class* dibutuhkan Operator **new**
Operator ini digunakan untuk membentuk (menginstankan) objek dari kelas.

Bentuk umumnya sbb:

```
namaKelas namaObjek = new namaKelas();
```

Pada saat menggunakan **new**, terjadi beberapa proses internal sebagai berikut:

- a. Objek baru tercipta
- b. Memory dialokasikan untuk objek tersebut
- c. Method konstruktor dipanggil untuk menginisialisasi objek

Untuk mengakses data/variabel dan method dalam sebuah class.

```
namaObjek.namaVariabel;  
namaObjek.namaMethod();
```


Contoh:

```
Mahasiswa objMhs = new Mahasiswa();  
objMhs.NIM;  
objMhs.inputNilai();
```

7.3 Keyword This

Keyword this berfungsi sebagai referensi dari variabel *instance*, yang mengacu pada objek saat ini. *Keyword this* juga digunakan untuk membedakan variabel *instance* dengan variabel atribut. Jika diartikan kata *this* ke dalam Bahasa Indonesia yaitu berarti "ini", dan *keyword this* ini digunakan di dalam pemrograman yaitu biasanya digunakan menunjukkan bahwa atribut *class* yang saat ini di akses atau yang sedang digunakan. *Keyword "this"* digunakan untuk mengakses kelas itu sendiri, biasanya digunakan untuk mengakses atribut pada kelas.

Jika kita tidak menggunakan *keyword this* pada nama variabel *instance* dan atribut yang sama maka nilai/value akan menjadi 0 pada **integer** dan null pada **string**. Selain itu juga, *keyword this* digunakan untuk memanggil *Constructor* milik class yang sedang di gunakan.

Jika kita **tidak ingin menggunakan** *keyword this* tapi nilai/value tetap dapat ditampilkan tanpa nilai 0 atau null, dengan cara membedakan nama variabel *instance* dengan variabel atribut.

7.4 Package

Package adalah sebuah upaya untuk mengelompokkan bagian-bagian program java menjadi satu. Sebuah package dalam java terdiri dari sekumpulan *class* dan/atau interface. Didalam sebuah package juga dimungkinkan mempunyai sub-package.

Package bisa berupa package yang sudah dimiliki oleh Java, dan ada pula package yang dibuat oleh user. Package yang dibuat oleh user ini sering disebut dengan folder. Atau sebuah lokasi didalam media penyimpanan yang kita miliki.

Dalam package yang kita buat dapat berisi sejumlah kelas dan didalam *package* tersebut juga bisa berupa *sub-package*.

1. Deklarasi *Package*

Deklarasi sebuah package diawali dengan nama package pada bagian teratas sebuah source program.

Bentuk Deklarasi *Package*:

```
package [namaPackage];
```

Contoh:

```
package kampus;  
class unit1{  
    // ... body kelas  
    // ... metode-metode  
    // ... field-field  
    // ...  
}
```

2. Mengakses Kelas dalam *Package*

Mengakses Kelas dengan Kata Kunci import:


```
import namaPackage.namaKelas;
```

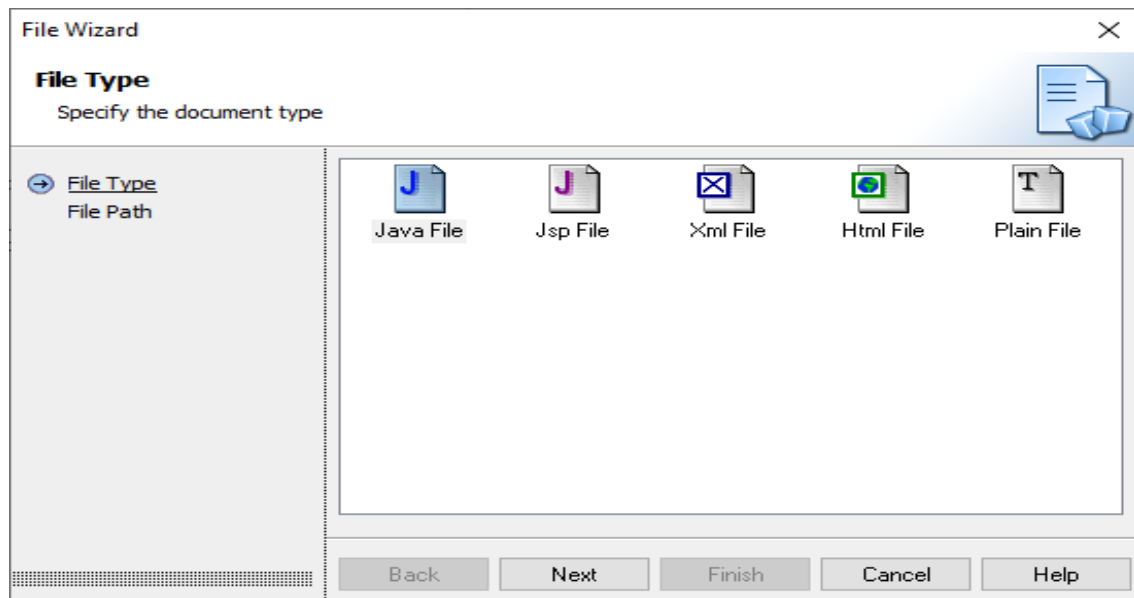
Contoh:

```
package demo;  
import kampus.*; // mengenalkan class yang terdapat dalam package kampus  
import java.awt.Color;  
class demo1{  
    public static void main(String[] args){  
        unit1 gedung1 = new unit1(); // class unit1 ada di package kampus  
        Color merah = new Color(Color.red);  
    }  
}
```

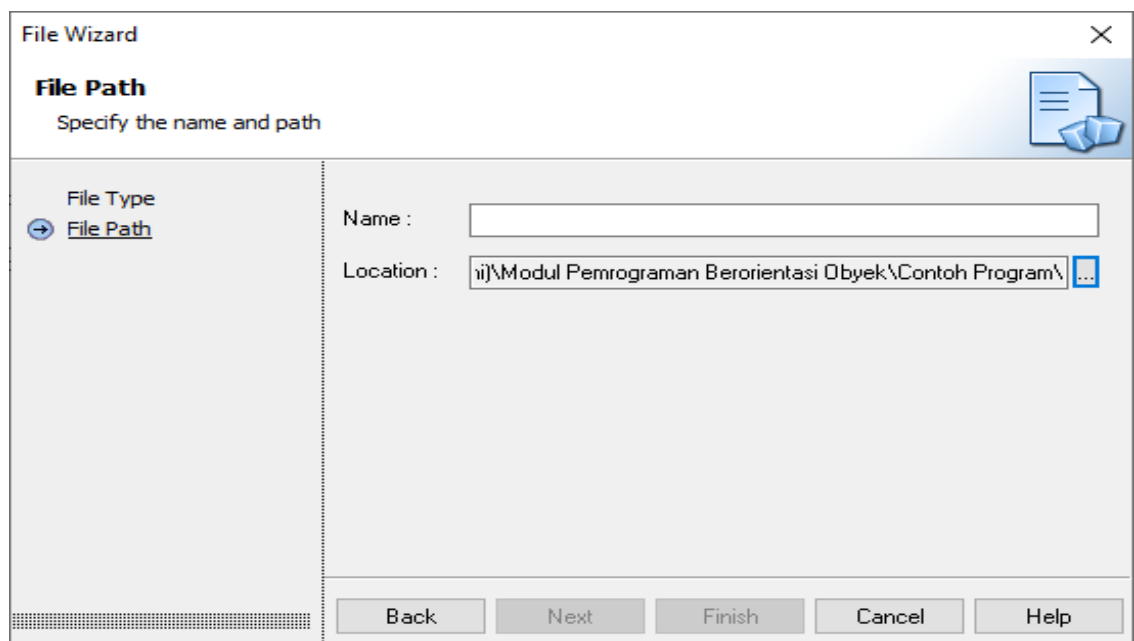
7.5 Latihan Program: Praktikum

Langkah-langkah Praktikum

21. Buka Editor JCreator
22. Buatlah file baru dengan membuka menu File > New > File atau dengan Shortcut Ctrl + N atau dengan klik icon  pada ribbon menu, kemudian pilih > Java File dan Klik tombol Next



23. Buat Nama File sesuai dengan nama file Java masing-masing Contoh program, isikan pada bagian > Name, pilih lokasi tempat penyimpanan file pada pojok kanan Bagian Location kemudian Klik tombol Finish.



24. Lanjutkan dengan menuliskan program pada layar editor JCreator

Program 7.1: Kotak.java

Tuliskan program 7.1 berikut pada editor JCreator

```
Kotak.java *
1  /*
2   * Contoh class sederhana
3   * Disini kita akan membuat kelas kotak.
4   * Untuk saat ini kita belum perlu menambahkan method ke dalam kelas tersebut.
5   */
6
7  class Kotak {
8      double panjang;
9      double lebar;
10     double tinggi;
11 }
12
13 /*
14 * Melalui kode diatas,
15 * berarti kita telah mendefinisikan sebuah tipe data baru dengan nama Kotak.
16 * Penting untuk diingat bahwa pendefinisian kelas hanya akan membuat sebuah pola atau template,
17 * bukan membuat objek. Objek actual dari kelas tersebut harus dibuat sendiri.
18 */
19
```

Lakukan Kompilasi dan Jalankan program 7.1 diatas dengan membuka menu Build

>Compile File  dan > Execute File 

Program 7.2: DemoKotak.java

Tuliskan program 7.2 berikut pada editor JCreator

```
DemoKotak.java *
1  /*
2   * untuk mengakses data/variabel yang terdapat pada kelas kotak kelas kotak,
3   * perlu dilakukan pendeklarasian kelas kotak menjadi objek.
4   * untuk mengaksesnya menggunakan (.) titik melalui objek yang sudah di buat.
5   */
6
7  class DemoKotak {
8      // method main method yang di akses pertama kali saat program di jalankan
9      public static void main (String[]args){
10         double volume;
11
12         // membuat objek dengan nama k
13         Kotak k = new Kotak ();
14
15         // mengisi nilai ke dalam data-data kelas Kotak
16         k.panjang = 4;
17         k.lebar = 3;
18         k.tinggi = 2;
19
20         // menghitung isi/volume kotak
21         volume = k.panjang * k.tinggi * k.lebar;
22
23         // menampilkan nilai volume ke layar monitor
24         System.out.println("volume kotak = " +volume);
25     }
26 }
```

Lakukan Kompilasi dan Jalankan program 7.2 diatas dengan membuka menu Build

>Compile File  dan > Execute File 

Program 7.3: Kotak2.java

Tuliskan program 7.3 berikut pada editor JCreator

```
Kotak2.java
1 class Kotak2{
2     int panjang;
3     int lebar;
4     int tinggi;
5     int volume;
6
7     // pembuatan method hitung volume dengan jenis non void/ mengembalikan nilai.
8     public int HitungVolume(){
9         volume = panjang * lebar * tinggi;
10        return volume;
11    }
12
13    public void SetData(int p, int l, int t){
14        panjang = p;
15        lebar = l;
16        tinggi = t;
17    }
18
19    public static void main(String[] args){
20        Kotak2 obj = new Kotak2();
21        obj.SetData(10,20,5); // melewati data (pass by value)
22        System.out.println("Volume Balok adalah " + obj.HitungVolume());
23    }
24 }
```

Lakukan Kompilasi dan Jalankan program 7.3 diatas dengan membuka menu Build

>Compile File  dan > Execute File 

Program 7.4: MethodOverLoading.java

Tuliskan program 7.4 berikut pada editor JCreator

```
MethodOverLoading.java *
1 /*
2  * method overloading setNilai().
3  * memiliki nama yang sama dengan parameter yang berbeda
4  */
5
6 class Manusia{
7     String nama;
8     String jenkel;
9
10    void setNilai(String param1){
11        nama = param1;
12    }
13    void setNilai(String param1,String param2){
14        nama = param1;
15        jenkel = param2;
16    }
17    void cetak(){
18        System.out.println(nama+" adalah "+jenkel);
19    }
20 }
21
22 class MethodOverLoading{
23     public static void main(String args[]){
24         Manusia m1,m2;
25         m1 = new Manusia();
26         m2 = new Manusia();
27
28         m1.setNilai("Zamzam");
29         m2.setNilai("Zamzam","Laki-laki");
30
31         m1.cetak();
32         m2.cetak();
33     }
34 }
35 }
```

Lakukan Kompilasi dan Jalankan program 7.4 diatas dengan membuka menu Build

>Compile File  dan > Execute File 

Program 7.5: Mahasiswa.java

Tuliskan program 7.5 berikut pada editor JCreator

```
Mahasiswa.java
1  /*
2   * contoh konstruktor,
3   * nama method sama dengan nama class
4   */
5
6  class Mahasiswa {
7      //deklarasi variabel
8      String nim;
9      String nama;
10
11     //default constructor
12     public Mahasiswa(){
13
14     }
15
16     //constructor perparameter
17     public Mahasiswa(String nim, String nama){
18         this.nim = nim;
19         this.nama= nama;
20     }
21
22     public String getNim(){
23         return nim;
24     }
25
26     public String getNama(){
27         return nama;
28     }
29 }
```

Lakukan Kompilasi dan Jalankan program 7.5 diatas dengan membuka menu Build

>Compile File  dan > Execute File 

Program 7.6: DemoMahasiswa.java

Tuliskan program 7.6 berikut pada editor JCreator

```
DemoMahasiswa.java
1  /*
2   * method konstruktor adalah method yang di jalankan saat objek mau dibuat
3   */
4
5  public class DemoMahasiswa {
6      public static void main(String[] args) {
7          Mahasiswa m= new Mahasiswa("11630441","Ria"); // akses method konstruktor
8
9          System.out.println("Nim :"+ m.getNim());
10         System.out.println("Nama :"+ m.getNama());
11     }
12 }
```

Lakukan Kompilasi dan Jalankan program 7.5 diatas dengan membuka menu Build

>Compile File  dan > Execute File 

Program 7.7: DemoTanpaThis.java

Tuliskan program 7.7 berikut pada editor JCreator

```
DemoTanpaThis.java *
1 class DemoTanpaThis {
2     String nama;
3     int id;
4
5     void setSiswa(int id, String nama){
6         id = id;
7         nama = nama;
8     }
9
10    void tampil(){
11        System.out.println("ID : " + id);
12        System.out.println("Nama : " + nama);
13    }
14
15    public static void main(String[] args) {
16        DemoTanpaThis demoThis = new DemoTanpaThis();
17        demoThis.setSiswa(1, "Ucup");
18        demoThis.tampil();
19    }
20 }
21
22
23
24
```

Lakukan Kompilasi dan Jalankan program 7.7 diatas dengan membuka menu Build

>Compile File  dan > Execute File 

Program 7.8: TestPackage01.java

Tuliskan program 7.8 berikut pada editor JCreator

```
1 /*
2  * Ini kelas yang hanya bisa diakses melalui kelas lain
3  * dengan kata lain, tidak dapat berdiri sendiri, karena tidak punya main
4  */
5
6 package Family;
7
8 class TestPackage01
9 {
10     static String nama    = "Tariq Athar Kisan";
11     static String rambut  = "Sawo Matang";
12     static int  tinggi    = 120;
13     static int  berat     = 25;
14     static int  umur      = 9;
15
16     public static void biodata()
17     {
18         System.out.println("Nama lengkap : " +nama);
19         System.out.println("Warna rambut : " +rambut);
20         System.out.println("Tinggi badan : " +tinggi);
21         System.out.println("Berat badan : " +berat);
22         System.out.println("Umur      : " +umur);
23     }
24 }
```

Lakukan Kompilasi dan Jalankan program 7.7 diatas dengan membuka menu Build

>Compile File  dan > Execute File 

Program 7.9: TestPackage02.java

Tuliskan program 7.9 berikut pada editor JCreator

```
1  /*
2   * akses kelas TestPackage01 di package yang sama dengan kelas TestPackage02
3   * tidak menggunakan import karena class berada di tempat yang sama
4   * kelas 1 bukan public karena berada di package yang sama
5   */
6
7  package Family;
8
9  class TestPackage02
10 {
11     public static void main(String[] args)
12     {
13         TestPackage01 objClass2 = new TestPackage01();
14         objClass2.biodata();
15     }
16 }
```

Lakukan Kompilasi dan Jalankan program 7.9 diatas dengan membuka menu Build

>Compile File  dan > Execute File 

Program 7.10: TestPackage03.java

Tuliskan program 7.10 berikut pada editor JCreator

```
1  /*
2   * tidak dapat mengakses kelas TestPackage01
3   * karena terletak di package yang berbeda dengan TestPackage03
4   */
5
6  package notFamily;
7
8  class TestPackage03
9  {
10     public static void main(String[] args)
11     {
12         TestPackage01 objClass3 = new TestPackage01();
13         objClass3.biodata();
14     }
15 }
```

Lakukan Kompilasi dan Jalankan program 7.10 diatas dengan membuka menu Build

>Compile File  dan > Execute File 

Program 7.11: TestPackage04.java

Tuliskan program 7.11 berikut pada editor JCreator

```
1  /*
2   * Solusi dari permasalahan di kelas TestPackage03
3   * Menambahkan sintak "import" dengan diikuti memanggil nama package Family
4   * Namun muncul permasalahan lagi karena kelas TestPackage01 yang mau kita akses bukan public
5   */
6
7  package notFamily;
8
9  import Family.*;
10
11 class TestPackage04
12 {
13     public static void main(String[] args)
14     {
15         TestPackage01 objClass4 = new TestPackage01();
16         objClass4.biodata();
17     }
18 }
```

Lakukan Kompilasi dan Jalankan program 7.11 diatas dengan membuka menu Build

>Compile File  dan > Execute File 

Program 7.12: TestPackage04.java

Tuliskan program 7.12 berikut pada editor JCreator

```
1  /*
2   * Solusi dari permasalahan di kelas TestPackage04
3   * Dengan mengubah kelas TestPackage01 menjadi TestPackage01a yang semula bukan public menjadi menjadi public
4   */
5
6  package notFamily;
7
8  import Family.*;
9
10 class TestPackage05
11 {
12     public static void main(String[] args)
13     {
14         TestPackage01a objClass5 = new TestPackage01a();
15         objClass5.biodata();
16     }
17 }
```

Lakukan Kompilasi dan Jalankan program 7.12 diatas dengan membuka menu Build

>Compile File  dan > Execute File 

7.6 Latihan Mandiri

Contoh lain:

Deklarasi: Class, method dan Object

```
1
2 public class Cetak {
3     //Main Class
4     int no;
5     String nama;
6     //deklarasi variable class
7     Cetak(int no,String nama){
8
9         this.no = no;
10        this.nama = nama;
11        //nama variable class dan variable atribut sama
12        //dengan keyword this
13    }
14
15    void tampil(){
16
17        System.out.println(no+" "+nama);
18        //menampilkan nilai/value yang di variable no dan nama
19    }
20
21    public static void main(String[] okedroid) { //method void main utama
22
23        Cetak c1 = new Cetak(43 ,"Fathurrahman");
24        //membuat obyek baru dari constructor Cetak
25        c1.tampil();
26
27        //menampilkan method tampil()
28
29    }
30
31 }
```

Penjelasan :

Baris 2 : Deklarasi class
Baris 7, 15 : Deklarasi method
Baris 23 : Deklarasi objek

Contoh: tanpa Keyword this

```
1 public class Cetak {
2     //Main Class
3     int no;
4     String nama;
5
6     //deklarasi variable class
7
8
9     Cetak(int no,String nama){
10        no = no;
11        nama = nama;
12        //nama variable class dan variable atribut sama
13        //tanpa keyword this
14    }
15
16    void tampil(){
17
18        System.out.println(no+" "+nama);
19        //menampilkan nilai/value yang di variable no dan nama
20    }
21
22    public static void main(String[] okedroid) { //method void main utama
23
24        Cetak c1 = new Cetak(43 ,"Fathurrahman");
25        //membuat obyek baru dari constructor Cetak
26        c1.tampil();
27
28        //menampilkan method tampil()
29
30    }
31
32 }
```

Contoh: dengan Keyword this

```
1
2 public class Cetak {
3     //Main Class
4     int no;
5     String nama;
6     //deklarasi variable class
7     Cetak(int no,String nama){
8
9         this.no = no;
10        this.nama = nama;
11        //nama variable class dan variable atribut sama
12        //dengan keyword this
13    }
14
15    void tampil(){
16
17        System.out.println(no+" "+nama);
18        //menampilkan nilai/value yang di variable no dan nama
19    }
20
21    public static void main(String[] okedroid) { //method void main utama
22
23        Cetak c1 = new Cetak(43 , "Fathurrahman");
24        //membuat obyek baru dari constructor Cetak
25        c1.tampil();
26
27        //menampilkan method tampil()
28
29    }
30
31 }
```

7.7 Kesimpulan

Pada Pertemuan ini dapat disimpulkan bahwa Mahasiswa dapat:

1. Menjelaskan dan menuliskan penggunaan class mulai deklarasi *class*, atribut class dan deklarasi *method*, serta pembuatan objek dengan bahasa pemrograman Java
2. Menjelaskan dan menuliskan penggunaan *keyword* this dengan bahasa pemrograman Java
3. Menjelaskan dan menuliskan penggunaan package mulai deklarasi package, pengaturan *class* dalam *package*, serta hak akses *class* dalam *package* dengan bahasa pemrograman Java