



POLITEKNIK MASAMY INTERNASIONAL

SK Menristekdikti RI Nomor: 731/KPT/I/2018

Jalan Ikan Paus No.10-15 Kertosari Banyuwangi - 68411
Telp (0333) 3384593 – <http://polmain.info>

PROGRAM STUDI D3 TEKNIK KOMPUTER

Form:
B.Ak/eva/04/20

Nama Dosen : *Arif Fahmi, S.T.,M.T.*

Mata Kuliah : Pemrograman Berorientasi Object

Semester : 4 (Empat)

Kode Mata Kuliah : TKV4044

Th. Akdm : 2019/2020.

BAB

CLASS DAN OBJECT JAVA BAGIAN I

SUB BAB :

1. Pengertian Class dan Object
2. Variabel Instanisasi Class dan Object
3. Method

TUJUAN MATERI

1. Menjelaskan mengenai Pemrograman berorientasi Obyek dan beberapa konsepnya
2. Perbedaan antara class dan obyek
3. Perbedaan antara variabel/method yang diturunkan dan variable/method class (static)
4. Menjelaskan mengenai method, serta cara pemanggilan dan pemberian parameter ke dalam method
5. Mengidentifikasi beberapa jangkauan dari sebuah variabel

REFERENSI

1. J.Eck, D. (2006). *Introduction to Programming Using Java*. Geneva.
2. An Object-Oriented Approach to Programming Logic and Design, Joyce Farrel, USA, 2013
3. An Introduction to Object Oriented Programming with Java, C. Thomas Wu, McGraw-Hill, New York, 2010.

CLASS DAN OBJECT JAVA BAGIAN I

Pada dasarnya bahasa pemrograman Java merupakan bahasa pemrograman yang berorientasi pada objek, Sebagai bahasa yang memiliki fitur object oriented, Java mendukung konsep dasar berikut ini ; (*Classes, Objects, Method, Instance, Abstraction, Polymorphism, Inheritance, Encapsulation, Message Parsing*). Pada bab ini kita akan membahas mengenai class dan object serta method.

1. Pegertian Class dan Object

Pada pemrograman berorientasi object, Fungsi dan variabel **berada** dalam sebuah **objek** atau *class* yang dapat saling brinteraksi, sehingga membentuk sebuah program.

A. Class

Dalam dunia nyata, kita sering berinteraksi dengan banyak object. Kita tinggal di rumah, rumah adalah suatu object, dalam terminology OOP rumah kita adalah instance dari suatu class rumah. Misal kita tinggal dalam suatu komplek perumahan, sebelum membangun rumah, developer akan berpanduan pada rancang bangun rumah (blue print) yang telah dibuat seorang arsitek. ***Blueprint dari rumah adalah class, sedang rumah yang kita tinggal (rumah-rumah dalam komplek) disebut instance.***

Manusia adalah sebuah class ; anda, saya, kita adalah instance dari class manusia.

B. Object

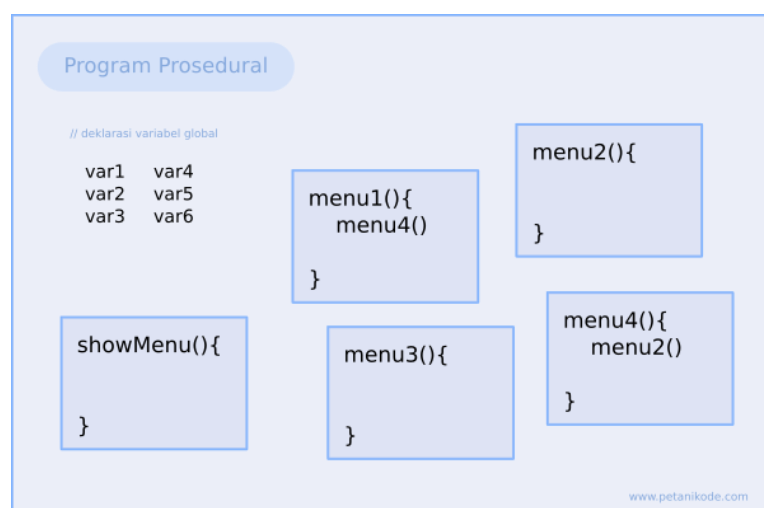
Object adalah instance dari class. Jika class secara umum merepresentasikan (template) sebuah object, ***sebuah instance adalah representasi nyata dari class itu sendiri.***

Catatan :

1. *Class* merupakan rancangan atau *blue print* dari sebuah objek.
2. *Object* adalah sebuah variabel yang merupakan *instance* atau perwujudan dari *Class*.
3. *Instance* bisa diartikan sebagai wujud dari *class*.

Berikut pemaparan konsep class dan object dalam pemrograman Java,

Kita ketahui pada pemrograman prosedural, kita biasanya memecah program menjadi beberapa prosedur. Lalu membuat variabel global dan lokal untuk menyimpan data.

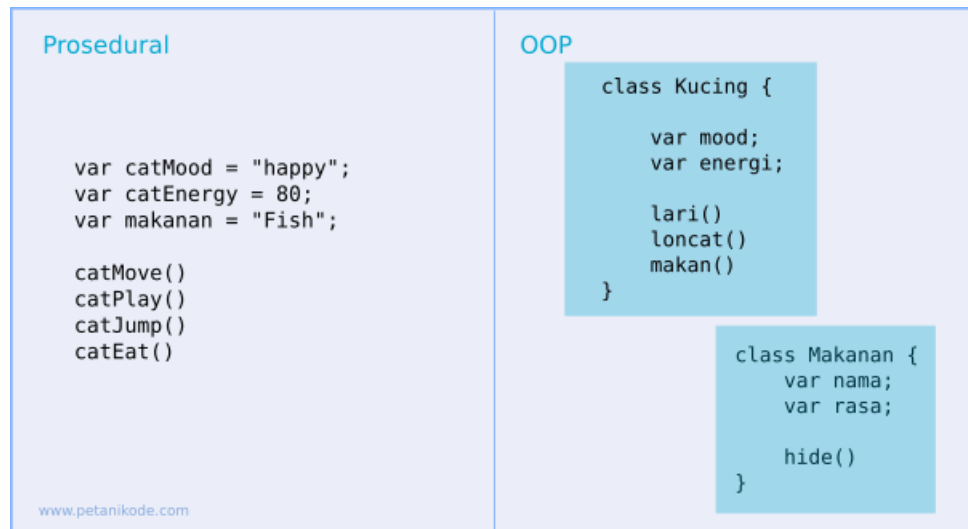


Semakin besar programnya, semakin banyak pula variabel dan prosedur yang harus dibuat. Lama-kelamaan, kode program akan sulit dimodifikasi, karena sudah terlalu kompleks.

Dari latar belakang tersebut terciptalah paradigma (cara berfikir) dalam membuat suatu program yang menitikberatkan pada identifikasi object-object yang terlibat dalam sebuah program.

Contoh studi kasus ;

Kita ingin membuat game sederhana, di dalamnya ada kucing dan makanan.



Dengan class, kita bisa menentukan mana variabel dan prosedur untuk kucing dan makanan. Class sebenarnya bertugas untuk mengumpulkan prosedur/fungsi dan variabel dalam satu tempat. Class ini nanti yang akan kita pakai untuk membuat objek. *Class* berisi definisi variabel dan fungsi yang menggambarkan sebuah objek.

Dalam OOP:

- Variabel disebut atribut atau properti;
- Fungsi disebut method.

Contoh *Class*:

```
class NamaClass {
    String atribut1;
    String atribut2;

    void namaMethod() { ... }
    void namaMethodLain() { ... }
}
```

Lalu, Kita biasanya membuat objek (*instance*) seperti ini:

```
NamaClass namaObj = new NamaClass();
```

Kata kunci “ `new` ” berfungsi untuk membuat objek baru dari *class* tertentu.

Setelah membuat objek, kita bisa mengakses atribut dan method dari objek tersebut.

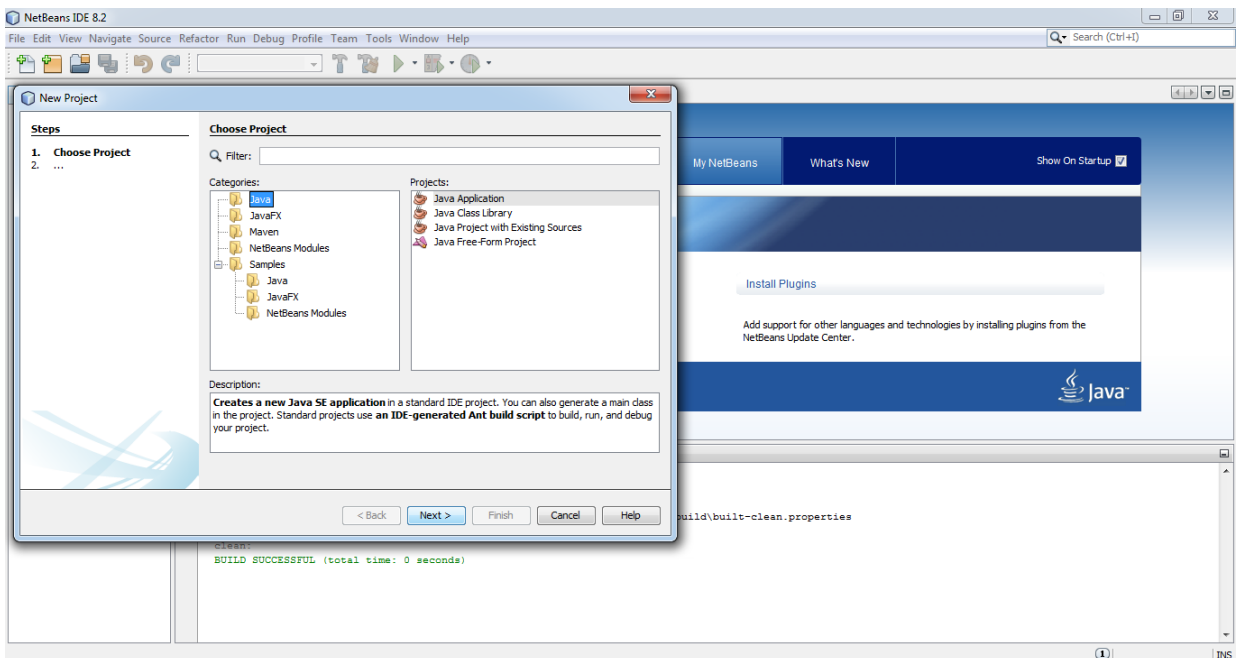
Contoh ,

```
namaObj.namaMethod();
namaObj.atribut1;
```

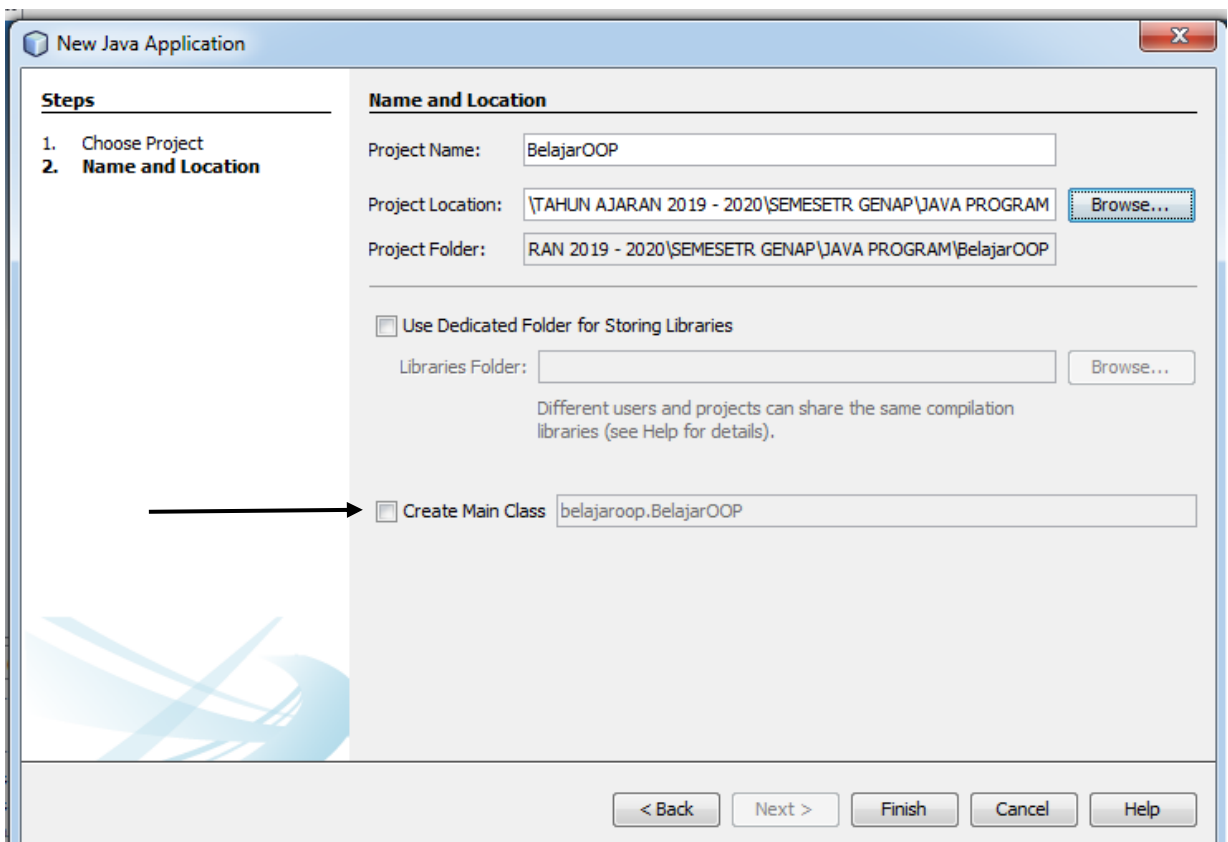
Tanda titik (.) berfungsi untuk mengakses atribut dan method.

Contoh penerapan Class pada pemrograman Berorientasi Object java menggunakan IDE Netbeans.

1. langkah pertama pilih software Netbeans (Run Administator), pilih menu file kemudian kilk folder NewProject, sehingga tampil seperti gambar berikut ini.



2. kemudian pilih categoris (Java) dan Peoject (Java Application), sehingga muncul tampilan berikut ini.



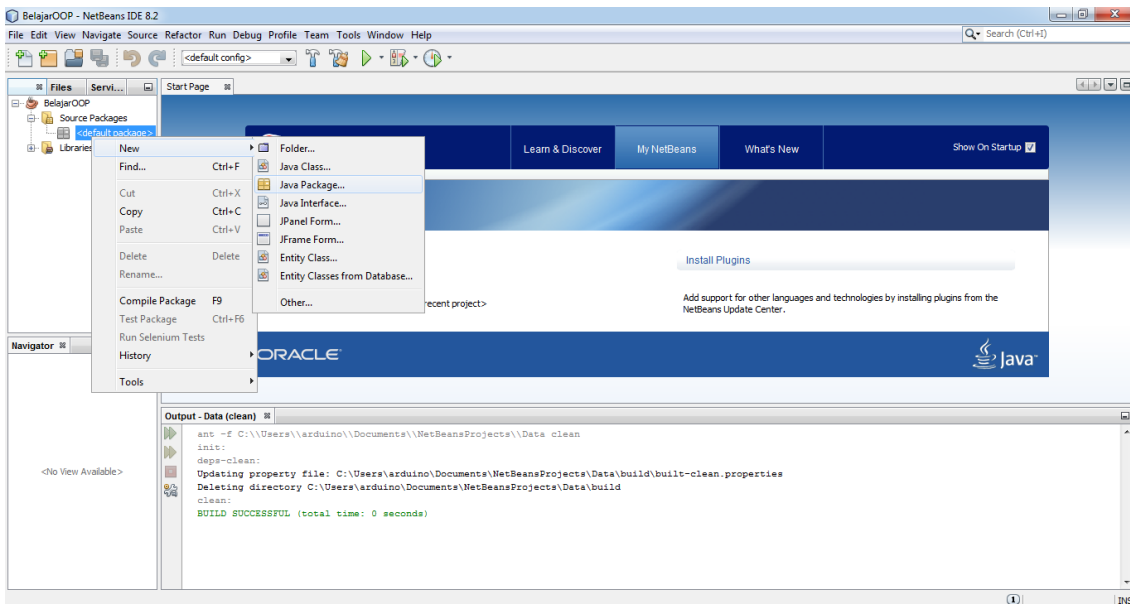
Isilah project name dengan “BelajarOOP”. Kemudian klik “Finish”

Catatan : Pada menu “Create Man class” jangan di centang tanda dipanahi.

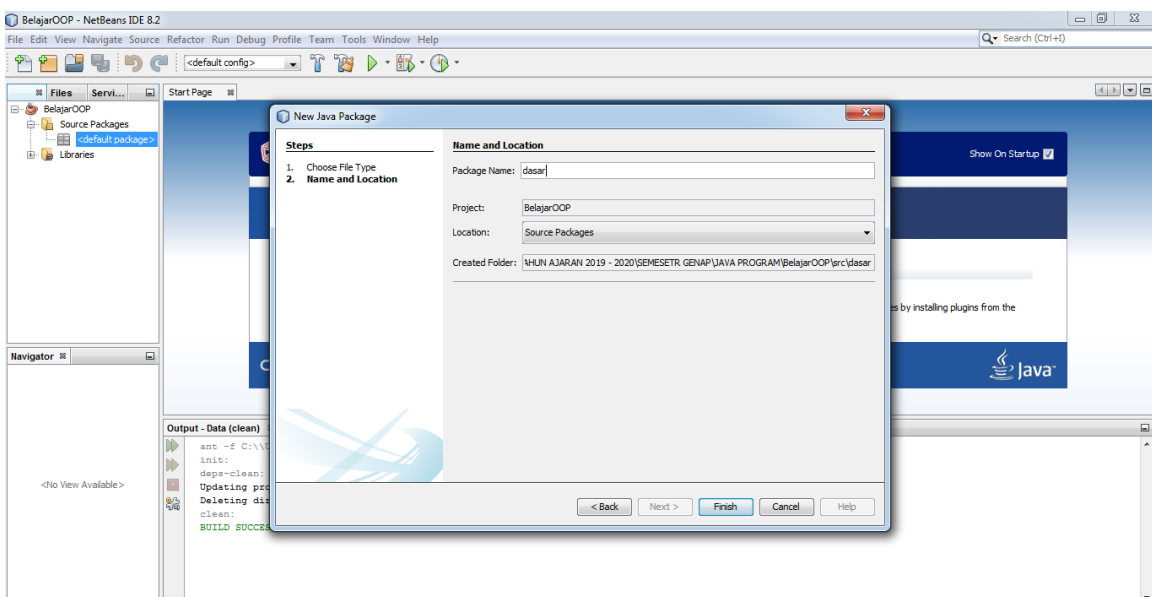


Java Programming

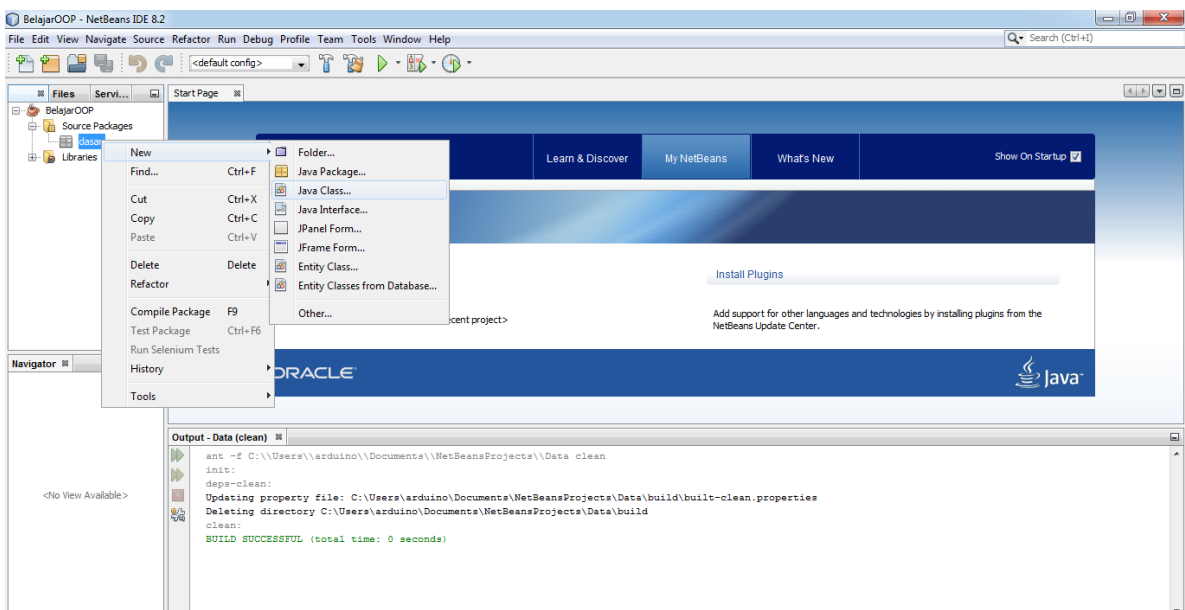
- Setelah itu, silahkan buat *package* baru. Klik kanan pada **<default package>**, lalu pilih **New->Java Package**.



- Isi nama *package* dengan “dasar”.

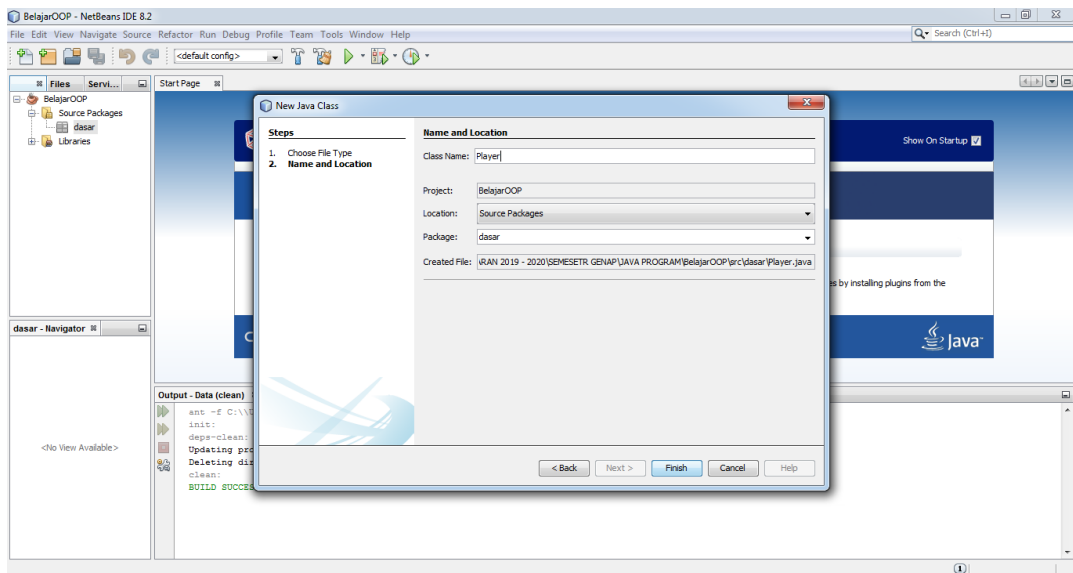


- Setelah itu, di dalam package dasar, silahkan buat *class* baru

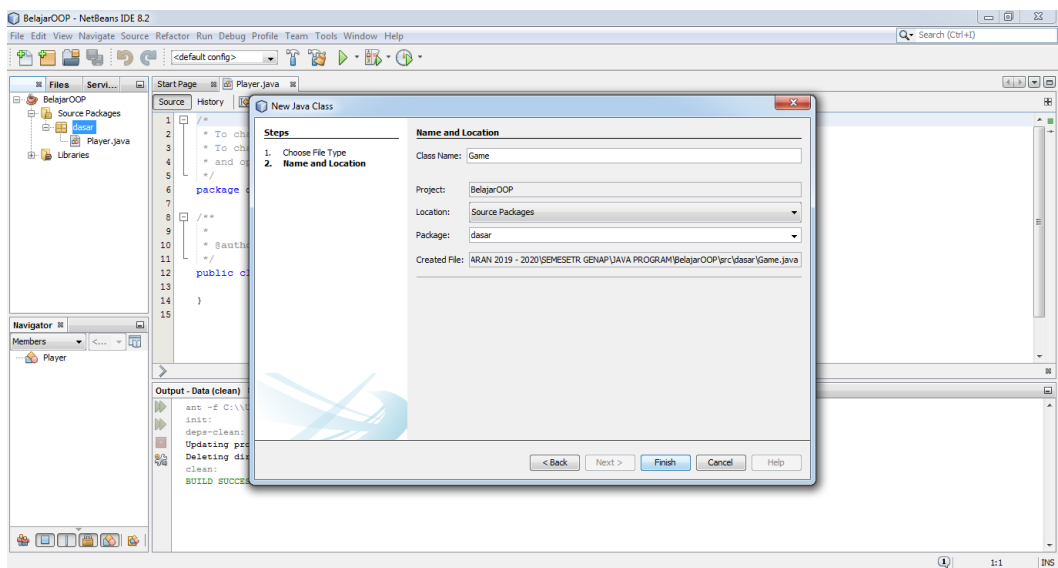




6. Isi dengan nama“Player”.



7. Berikutnya, buat *class* baru lagi bernama “Game”



Berikut contoh script Class Player

```
package dasar;

public class Player {

    // definisi atribut
    String name;
    int speed;
    int healthPoin;

    // definisi method run
    void run(){
        System.out.println(name + " is running...");
        System.out.println("Speed: " + speed);
    }

    // definisi method isDead untuk mengecek nilai
    kesehatan (healthPoin)
    boolean isDead(){
        if(healthPoin <= 0) return true;
        return false;
    }

}
```



```
Start Page  Player.java  Game.java
Source  History  [Icons]
1  package dasar;
2  /**
3   *
4   * @author arduino
5   */
6  public class Player {
7      // definisi atribut
8      String name;
9      int speed;
10     int healthPoin;
11     // definisi method run
12     void run(){
13         System.out.println(name + " is running...");
14         System.out.println("Speed: " + speed);
15     }
16     // definisi method isDead untuk mengecek nilai kesehatan (healthPoin)
17     boolean isDead(){
18         if(healthPoin <= 0) return true;
19         return false;
20     }
21 }
```

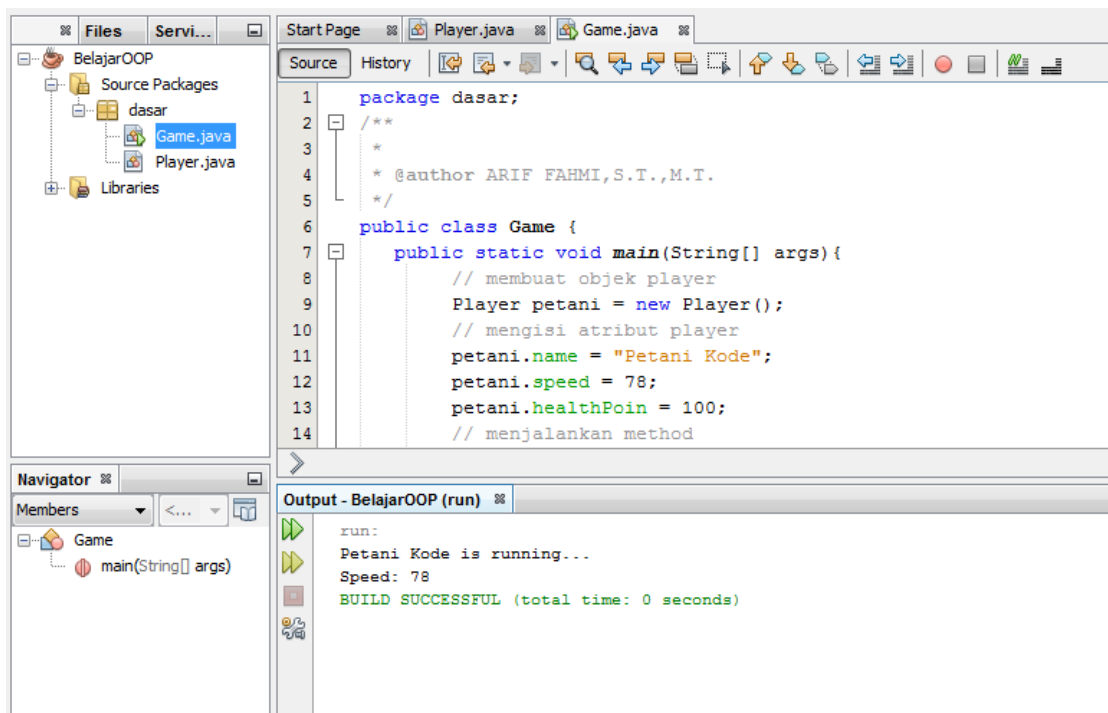
Berikut contoh *script Class Game*

```
package dasar;

public class Game {
    public static void main(String[] args){
        // membuat objek player
        Player petani = new Player();
        // mengisi atribut player
        petani.name = "Petani Kode";
        petani.speed = 78;
        petani.healthPoin = 100;
        // menjalankan method
        petani.run();
        if(petani.isDead()){
            System.out.println("Game Over!");
        }
    }
}
```

```
Start Page  Player.java  Game.java
Source  History  [Icons]
1  package dasar;
2  /**
3   *
4   * @author ARIF FAHMI, S.T., M.T.
5   */
6  public class Game {
7      public static void main(String[] args){
8          // membuat objek player
9          Player petani = new Player();
10         // mengisi atribut player
11         petani.name = "Petani Kode";
12         petani.speed = 78;
13         petani.healthPoin = 100;
14         // menjalankan method
15         petani.run();
16         if(petani.isDead()){
17             System.out.println("Game Over!");
18         }
19     }
20 }
```

8. Selanjutnya eksekusi Game.java dengan klik kanan pada Game.java lalu pilih Run File.
Maka hasilnya:

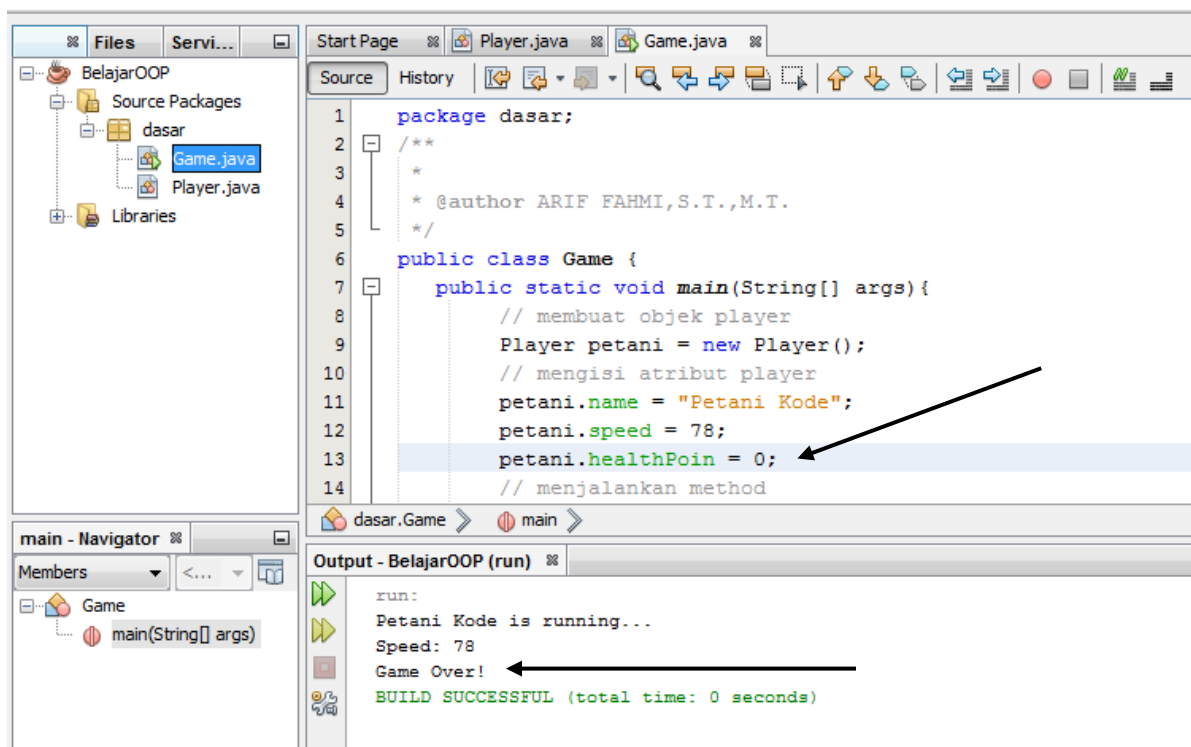


```

1 package dasar;
2 /**
3  *
4  * @author ARIF FAHMI, S.T., M.T.
5  */
6 public class Game {
7     public static void main(String[] args){
8         // membuat objek player
9         Player petani = new Player();
10        // mengisi atribut player
11        petani.name = "Petani Kode";
12        petani.speed = 78;
13        petani.healthPoin = 100;
14        // menjalankan method
    
```

run:
Petani Kode is running...
Speed: 78
BUILD SUCCESSFUL (total time: 0 seconds)

9. Coba ganti nilai healthPoin menjadi 0.



```

1 package dasar;
2 /**
3  *
4  * @author ARIF FAHMI, S.T., M.T.
5  */
6 public class Game {
7     public static void main(String[] args){
8         // membuat objek player
9         Player petani = new Player();
10        // mengisi atribut player
11        petani.name = "Petani Kode";
12        petani.speed = 78;
13        petani.healthPoin = 0;
14        // menjalankan method
    
```

run:
Petani Kode is running...
Speed: 78
Game Over!
BUILD SUCCESSFUL (total time: 0 seconds)

Pembahasan ;

Baik, kita mulai dari pembuatan *class*.

Pertama-tama, kita membuat sebuah *class* dengan nama Player. Class ini mendefinisikan objek Player dalam Game dengan atribut sebagai berikut:

- name adalah nama objek;
- speed adalah kecepatan;
- healthPoin adalah nilai kesehatan dari player, biasanya disingkat hp.

Lalu class Player memiliki method:

- run() untuk menggerakkan player;
- isDead() untuk mengecek kondisi kesehatan player. Method ini akan mengembalikan nilai true apabila nilai hp lebih kecil atau sama dengan nol (0), sebaliknya akan mengembalikan nilai false.

Berikutnya kita membuat objek baru dari class Player pada class Game bernama petani.

```
// membuat objek player  
Player petani = new Player();
```

```
package dasar;  
  
/**  
 *  
 * @author ARIF FAHMI,S.T.,M.T.  
 */  
public class Game {  
    public static void main(String[] args){  
        // membuat objek player  
        Player petani = new Player();  
        // mengisi atribut player  
        petani.name = "Petani Kode";  
        petani.speed = 78;  
        petani.healthPoin = 0;  
        // menjalankan method  
        petani.run();  
        if(petani.isDead()){  
            System.out.println("Game Over!");  
        }  
    }  
}
```

Setelah itu mengisi atribut-atributny. Karena kalau tidak diisi akan bernilai *Null* dan bisa menyebabkan *NullPointerException*.



```
petani.name = "Petani Kode";  
petani.speed = 78;  
petani.healthPoin = 100;  
  
petani.isDead(); //-> false
```

Berikutnya kita coba modifikasi nilai hp menjadi nol.

Hasilnya, pesan Game Over! akan ditampilkan. Karena method isDead() akan mengembalikan nilai true jika nilai hp kurang atau sama dengan nol.

```
petani.name = "Petani Kode";  
petani.speed = 78;  
petani.healthPoin = 0;  
  
petani.isDead(); //-> true
```



2. Variabel Intanisasi Class dan Object

Untuk membuat sebuah objek atau sebuah instance pada sebuah class. Kita menggunakan operator `new`. Sebagai contoh, jika anda ingin membuat instance dari class string, kita menggunakan kode berikut :

```
String str2 = new String("Hello world!");
```

Ini juga sama dengan,

```
String str2 = "Hello";
```

Selain dari variabel instance, kita juga memungkinkan untuk mendefinisikan variabel dari class, yang nantinya variabel ini dimiliki oleh class. Ini berarti variabel ini dapat memiliki nilai yang sama untuk semua objek pada class yang sama. Mereka juga disebut *static member variables*.

3. Method

Apakah Method itu dan mengapa menggunakan Method?

Pada contoh yang telah kita diskusikan sebelumnya, kita hanya memiliki satu method, dan itu adalah method `main()`. Di dalam Java, kita dapat mendefinisikan banyak method yang akan kita panggil dari method yang berbeda.

Sebuah method adalah bagian-bagian kode yang dapat dipanggil oleh program utama atau dari method lainnya untuk menjalankan fungsi yang spesifik.

Berikut adalah karakteristik dari method :

- dapat mengembalikan satu nilai atau tidak sama sekali
- dapat diterima beberapa parameter yang dibutuhkan atau tidak ada parameter sama sekali. Parameter bisa juga disebut sebagai argumen dari fungsi
- setelah method telah selesai dieksekusi, dia akan kembali pada method yang memanggilnya.

Sekarang mengapa kita butuh untuk membuat banyak method? Mengapa kita tidak menuliskan semua kode pada sebuah method? Hal ini karena penyelesaian masalah yang sangat efektif adalah memecah masalah-masalah tersebut menjadi beberapa bagian. Kita juga dapat melakukan hal ini di Java dengan membuat method untuk mengatasi bagian tertentu dari masalah. Sebuah permasalahan dapat dipecah-pecah menjadi beberapa bagian kecil. Hal ini sangat baik sekali untuk membuat program yang sangat besar.

A. Memanggil Instance dan memberikan Variabel dari Method

Sekarang, untuk mengilustrasikan bagaimana memanggil method, mari kita menggunakan class string sebagai contoh. Anda dapat menggunakan the dokumentasi dari Java API untuk melihat semua method yang tersedia dalam class string. Selanjutnya, kita akan membuat method, kita sendiri. Tapi untuk saat ini, mari terlebih dahulu kita gunakan method yang sudah disediakan oleh Java.

Untuk memanggil sebuah instance method, kita dapat menuliskan :

```
nameOfObject.nameOfMethod( parameters );
```

mari kita mengambil dua contoh method yang ditemukan dalam class String.

Deklarasi method	Definisi
public char charAt (int index)	Mengambil karakter pada indeks tertentu.
public boolean equalsIgnoreCase (String anotherString)	Membandingkan antar String, tidak case sensitive.

Menggunakan Method;

```
String    str1 = "Hello";
charx = str2.charAt(0); //will return the
character H
//simpan pada variabel x

String    str2 = "hello";

//return boolean
Boolean result = str1.equalsIgnoreCase( str1 );
```

B. Pemberian Variabel Dalam Method

Pada contoh kita sebelumnya, kita sudah pernah mencoba melewati variabel pada method. Walaupun kita belum dapat membedakan antara perbedaan tipe variabel yang diberikan (*passing*) ke method dalam Java. Ada dua tipe data variabel *passing* pada method, yang pertama adalah *pass-by-value* dan yang kedua adalah *pass-by-reference*.

1. Pass-by-Value

Ketika *pass-by-values* terjadi, method membuat sebuah salinan dari nilai variable yang dikirimkan ke method. Walaupun demikian, method tidak dapat secara langsung memodifikasi nilai variabel pengirimnya meskipun parameter salinannya sudah dimodifikasi nilainya di dalam method.

```
public class TestPassByValue
{
    public static void main( String[] args ){
        int i = 10;
        //mencetak nilai i
        System.out.println( i );

        //memanggil method test
        //passing i pada method test
        test( i );

        //Mencetak nilai i
        System.out.println( i );
    }

    public static void test( int j ){
        //merubah nilai parameter j
        j = 33;
    }
}
```

Pada contoh diatas, kita memanggil method tes dan melewati nilai variabel i sebagai parameter. Nilai pada i disalinkan ke variable j pada method. Pada kondisi ini variabel j



Java Programming

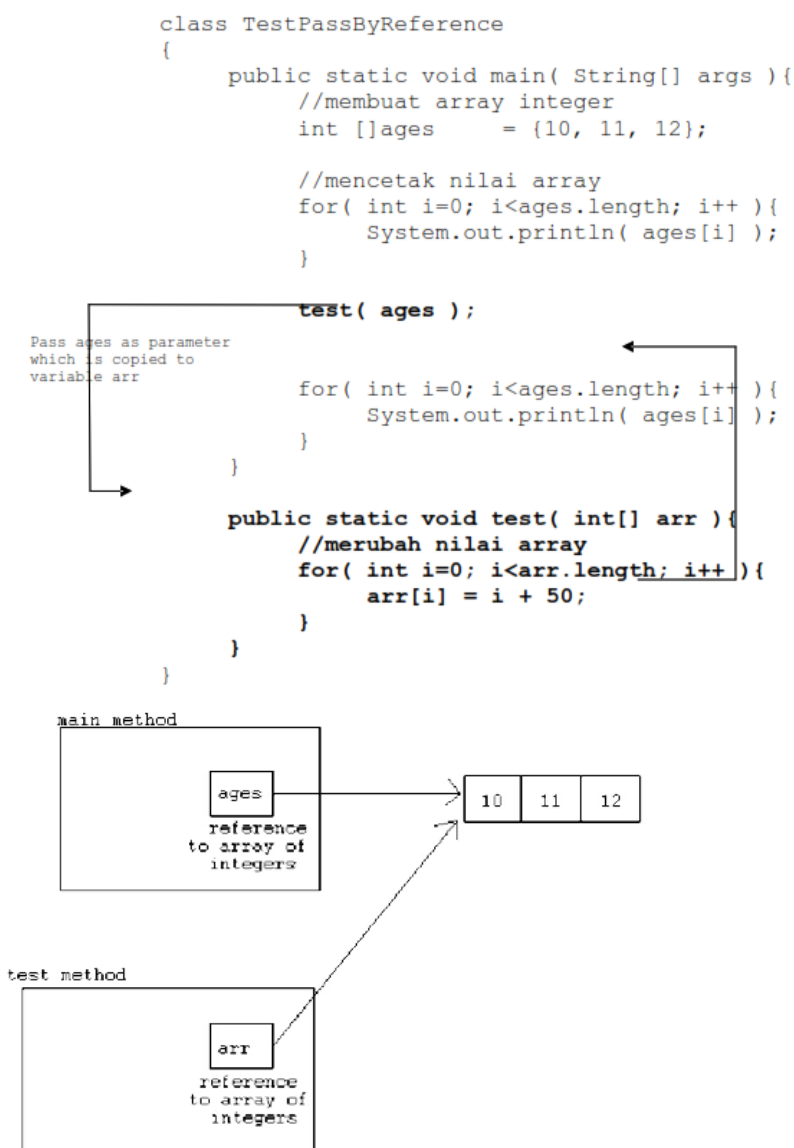
adalah merupakan variabel pengganti pada method tes, jika nilai j berubah maka nilai variabel i yang terletak pada main tidak akan ikut berubah walaupun awalnya variabel j merupakan salinan dari variabel i.

Secara default, semua tipe data primitif ketika dilewatkan pada sebuah method adalah *pass-by-value*

2. Pass-by-reference

Ketika sebuah pass-by-reference terjadi, alamat memori dari nilai pada sebuah variabel dilewatkan pada saat pemanggilan method. Hal ini berarti bahwa method menyalin alamat memori dari variabel yang dilewatkan pada method. Ini tidak seperti pada pass-by-value, method dapat memodifikasi variabel asli dengan menggunakan alamat memori tersebut, meskipun berbeda nama variabel yang digunakan dalam method dengan variabel aslinya, kedua variabel ini menunjukkan lokasi dari data yang sama.

contoh :



Petunjuk Penulisan Program :

Kesalahan konsep mengenai pass-by-reference pada Java adalah pada saat membuat method untuk menukar nilai variabel menggunakan pengalamatan java. Perlu diingat bahwa java memanipulasi obyek-obyek dengan cara 'by reference', akan tetapi java mengirimkan alamat obyek ke dalam method dengan cara 'by value'. Untuk itu, anda tidak dapat menuliskan method penukaran standar ke dalam obyek.



C. Memanggil Method Static

Method Static adalah method yang dapat dipakai tanpa harus menginisialisasi suatu class (maksudnya tanpa menggunakan variabel terlebih dahulu). Method static hanya dimiliki oleh class dan tidak dapat digunakan oleh instance (atau objek) dari suatu class. Method static dibedakan dari method yang dapat instance di dalam suatu class oleh kata kunci static.

Untuk memanggil method static, ketik :

```
Classname.staticMethodName(params);
```

Contoh dari static method yang digunakan :

```
//mencetak data pada layar
System.out.println("Hello world");
//convert string menjadi integer
int i = Integer.parseInt("10");

String hexEquivalent = Integer.toHexString( 10 );
```

Tugas

Catatan.

- Tugas dikumpulkan maks pada hari Kamis tgl 16-04-2020 pukul 16.00
- Tugas boleh dikumpulkan dalam bentuk (PDF,PPT,WORD)
- File dikirim ke email : fahmi03031995@gmail.com
- Tugas program dapat dikerjakan di aplikasi (Dcoder, java ide for android, maupun software Netbeans,atau Notepad)
- Format file dengan tata cara: NamaMhs_NIM_PBO

Soal

- Buatlah sebuah pemrograman Java dengan menerapkan konsep “ class dan object” sesuai kreatifitas kalian.

Catatan : setelah program dibuat jelaskan secara terperinci maksud dan fungsi dari program yang kamu buat