unindra
universitas indraprasta pgri

PERTEMUAN 3

Access Spesifiers

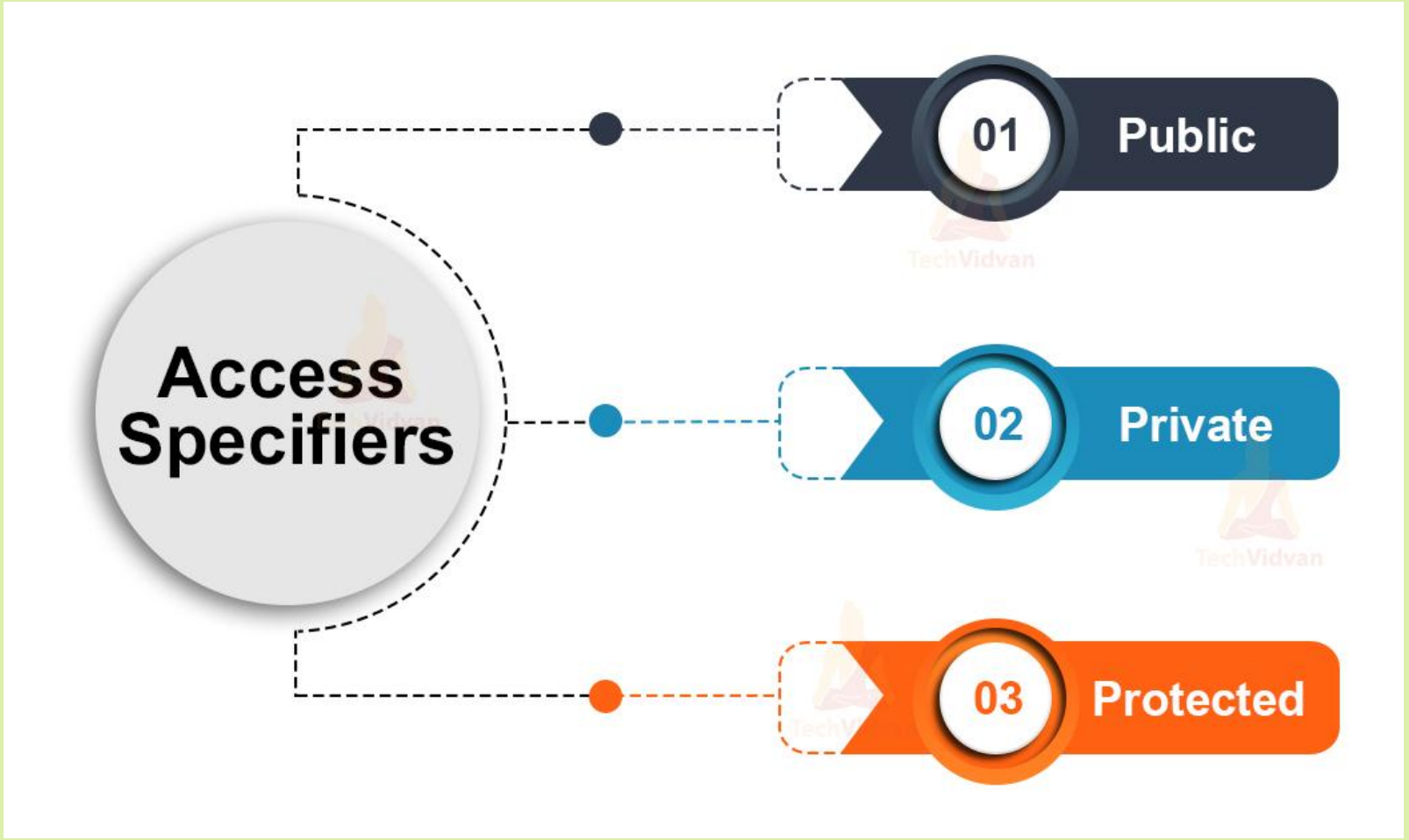
Access Modifiers

Access Control

Access Control dapat mengatur class mana saja yang dapat mengakses atau mengubah nilai suatu member dari objek. Dengan demikian, anda dapat mencegah kemungkinan penyalahgunaan fungsi objek itu sendiri.

Java menyediakan **Access Specifiers** dan **Access Modifiers** untuk menentukan bagian dari class yang boleh diakses oleh class lain dan bagaimana penggunaan atribut dan method tersebut.

Perbedaan utama Access Specifiers dan Access Modifiers adalah, Access Specifiers mendefinisikan hak akses anggota class, sedangkan Access Modifiers digunakan untuk mendefinisikan bagaimana suatu method atau atribut dapat dimodifikasi oleh class yang lain.


Access Specifiers
01
Public
02
Private
03
Protected

Access Spesifiers

Jenis access ini, digunakan untuk mengontrol pengaksesan bagian kelas oleh object yang lain. Dalam arti, yang memanagement, bagaimana suatu class dapat diakses oleh object, object siapa saja yang dapat mengakses, sampai batas mana class tersebut diakses oleh object dan lain sebagainya diatur atau dikontrol oleh access specifiers ini.

Java menyediakan tiga buah access spesifier yang dapat kita gunakan. Tiga access spesifier tersebut diantaranya :

1. Public (Bisa juga disebut Friendly/Default)
2. Private
3. Protected

Access Specifiers in Java

		public	private	protected	default
Same Package	Class	YES	YES	YES	YES
	Sub class	YES	NO	YES	YES
	Non sub class	YES	NO	YES	YES
Different Package	Sub class	YES	NO	YES	NO
	Non sub class	YES	NO	NO	NO

Public adalah kode akses yang bersifat umum. Dengan kata lain, data maupun method dalam suatu kodingan tersebut dapat diakses oleh semua bagian di dalam program, baik di dalam atau di luar class/package.

Private adalah kode yang sesuai dengan namanya, akses ini bersifat private. Dengan kata lain data maupun method hanya dapat diakses oleh kelas yang dimilikinya saja.

Protected adalah kode akses yang membuat suatu data atau method yang didefinisikan dengan tingkatan akses ini dapat diakses oleh kelas yang memilikinya saja dan juga kelas-kelas yang masih memiliki keturunan.

Contoh syntax public, private dan protected

public<data type><variable name>;

public String nama ;

Atau

<data type><variable name>;

int umur ;

private<data type><variable name>;

private int gaji;

protected<data type><variable name>;

protected String nama;

Noted :

Untuk access menggunakan private akan di bahas selanjutnya di materi Encapsulation

Untuk access menggunakan protected akan di bahas selanjutnya di materi pewarisan dan polymorphisme

Contoh Coding public dan private

Class Pertama :

```
class belajarpublik {
   public String nama;
}
```

Class Kedua :

```
public class belajar_publik1{
   public static void main(String[] args){
      belajarpublik publik1 = new
belajarpublik();
      publik1.nama = "Andika";
      System.out.println ( "Nama Anda : "
+publik1.nama );
   }
}
```

Dari koding di atas menjelaskan bahwa variabel nama bisa di akses di class lain karena bersifat public

Class Pertama :

```
class belajarprivate {
   private String nama;
}
```

Class Kedua :

```
public class belajar_private1 {
   public static void main(String[] args){
      belajarprivate private1 = new
belajarprivate();
      private1.nama = "Andika";
      System.out.println ( "Nama Anda : "
+private1.nama );
   }
}
```

Dari koding di atas menjelaskan bahwa variabel nama tidak bisa di akses di class lain karena bersifat private. Karena private hanya bisa di akses di class itu sendiri

Contoh Coding protected

Class Pertama :

```
class belajarprotec {
   protected String nama;
}
```

Class Kedua :

```
public class belajar_protect1 {
   public static void main(String[] args){
      belajarprotec protec1 = new
belajarprotec();
      protec1.nama = "Andika";
      System.out.println ( "Nama Anda : "
+protec1.nama );
   }
}
```

Dari koding di samping menjelaskan bahwa variabel nama bisa di akses di class lain walau bersifat protected. Karena class belajar_protec1 sudah menginstace dari class belajarprotec. Namun jika di access dari package lain maka tidak bisa.

Access Modifiers in Java

static
final
abstract

Access Modifiers

Jenis access ini digunakan untuk mendefinisikan bagaimana suatu atribut danmethod digunakan oleh class dan object yang lain. Modifier menyediakan tiga buah access modifier yang dapat kita gunakan. Tiga access tersebut diantaranya :

1. Static
2. Final
3. Abstract

Noted :

Static, Final akan di pelajari setelah masuk materi Method

Abstract akan di pelajari setelah masuk materi kelas turunan/inheritance

Static

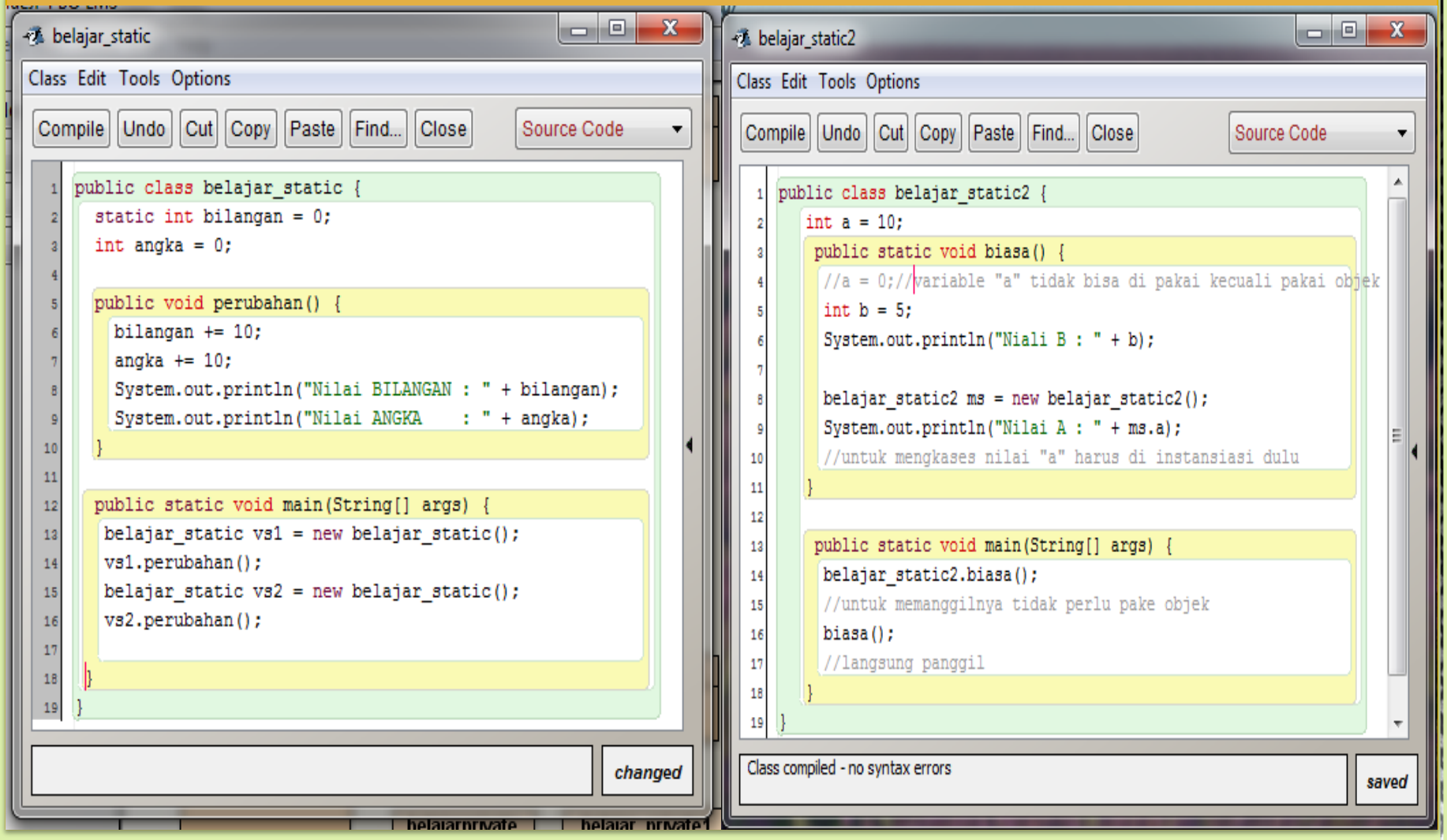
1.Dalam Variable

Jika sebuah variable menggunakan static maka variable tersebut akan menjadi variable kelas. Nilainya akan selalu berubah jika ada perubahan di objeknya.

2.Dalam Method

Jika sebuah method menggunakan static maka untuk mengaksesnya tidak perlu menggunakan objek / instansiasi kelas nya dulu cukup NamaKelas.NamaMethode. Dan variable yang ada didalamnya haruslah static tidak boleh menggunakan variable biasa, kecuali variable dari instansiasi dan variable yang baru dideklarasikan dimethod tersebut.

Contoh Static di Variabel dan di Method



belajar_static

Class Edit Tools Options

Compile Undo Cut Copy Paste Find... Close Source Code

```
1  public class belajar_static {
2    static int bilangan = 0;
3    int angka = 0;
4
5    public void perubahan() {
6      bilangan += 10;
7      angka += 10;
8      System.out.println("Nilai BILANGAN : " + bilangan);
9      System.out.println("Nilai ANGKA    : " + angka);
10   }
11
12   public static void main(String[] args) {
13     belajar_static vs1 = new belajar_static();
14     vs1.perubahan();
15     belajar_static vs2 = new belajar_static();
16     vs2.perubahan();
17
18   }
19 }
```

changed

belajar_static2

Class Edit Tools Options

Compile Undo Cut Copy Paste Find... Close Source Code

```
1  public class belajar_static2 {
2    int a = 10;
3    public static void biasa() {
4      //a = 0;//variable "a" tidak bisa di pakai kecuali pakai objek
5      int b = 5;
6      System.out.println("Niali B : " + b);
7
8      belajar_static2 ms = new belajar_static2();
9      System.out.println("Nilai A : " + ms.a);
10     //untuk mengkases nilai "a" harus di instansiasi dulu
11   }
12
13   public static void main(String[] args) {
14     belajar_static2.biasa();
15     //untuk memanggilnya tidak perlu pake objek
16     biasa();
17     //langsung panggil
18   }
19 }
```

Class compiled - no syntax errors

saved

Final

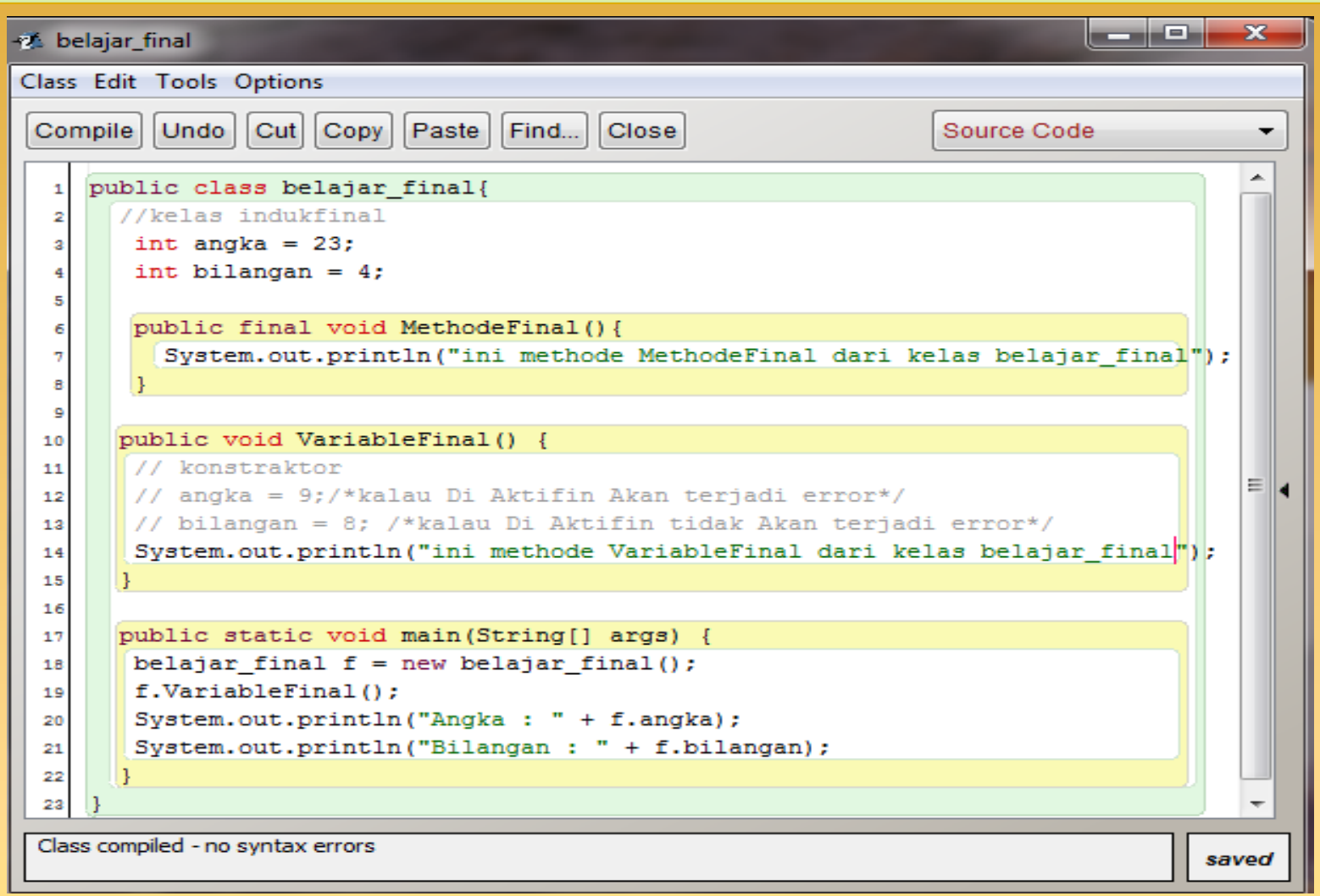
1. Dalam Variable

Jika sebuah variable menggunakan kata final maka variable tersebut akan berubah menjadi sebuah konstantadan tidak dapat di set / di isi dengan nilai yang baru.

2. Dalam Method

Jika sebuah methode menggunakan akses final maka method tersebut tidak bisa di override methodnya oleh kelas anaknya / subkelas

Contoh Final



Abstract

Abstract Class adalah sebuah class yang tidak bisa di-*instansiasi* (tidak bisa dibuat menjadi objek) dan berperan sebagai '*kerangka dasar*' bagi class turunannya. Di dalam *abstract class* umumnya akan memiliki *abstract method*.

Abstract Method adalah sebuah '*method dasar*' yang harus diimplementasikan ulang di dalam class anak (*child class*). *Abstract method* ditulis tanpa isi dari *method*, melainkan hanya '**signature**'-nya saja. **Signature** dari sebuah *method* adalah bagian method yang terdiri dari nama method dan parameternya (jika ada).