**LAPORAN PRAKTIKUM**

**ALGORITMA DAN PEMROGRAMAN**

**Pertemuan Ke – 11**



**DISUSUN OLEH :**

**HELDA LUDYA SAFITRI**

**175410186**

**TEKNIK INFORMATIKA**

**STMIK AKAKOM**

**YOGYAKARTA**

**2017**

**PERTEMUAN KE-11**

**CONSTRUKTOR, METODE OVERLOADING DAN MODIFIER**

1. **TUJUAN**
2. Mahasiswa dapat menjelaskan mengenai konstruktor.
3. Mahasiswa dapat membuat dan menjelaskan mengenai overloading pada konstruktor dan metode.
4. Mahasiswa dapat menjelaskan mengenai akses modifier.
5. Mahasiswa dapat menjelaskan mengenai variabel dan metode static.
6. **DASAR TEORI**

Constructor merupakan sebuah method khusus yang digunakan untuk menginisialisasi object dan tiap class boleh memiliki lebih dari satu konstruktor.

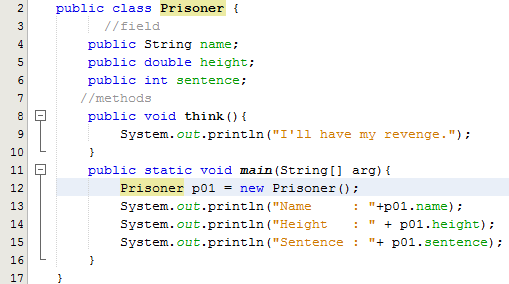
Method overloading merupakan pembuatan method lebih dari dengan nama yang sama namun dengan syarat harus menggunakan parameter yang berbeda,karena java akan melakukan eksekusi pada method tersebut berdardasarkan parameter-parameter tersebut.

Modifier berfungsi untuk melakukan enkapsulasi(membungkus data) pada object,dimana dengan menggunakan modifier kita bisa menentukan dimana saja member yang kita buat bisa diakses.

1. **PEMBAHASAN PRAKTIK**

**PRAKTIK 1 : Konstruktor**

1. Understand default value



Baris 3-5digunakan untuk mendeklarasikan variable dengan scope global.

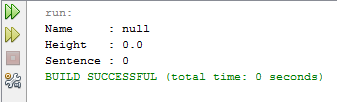
Scope global merupakan sebuah pendeklarasian variable dimana bisa di akses di dalam class itu sendiri maupun di class lain karena menggunakan modifier public. **Public** merupakan salah satu bentuk dari modifier apabila sebuah class menggunakan modifier ini maka class tersebut harus di simpan dengan nama file yang sama dengan nama classnya.

Kemudian pada baris 8 merupakan sebuah method **Public void think()** merupakan sebuah method tanpa adanya pengembalian nilai, karena pendeklarasian methodnya tanpa menggunakan tipe data. Baris 12 merupakan sebuah pembuatan instance object di dalam class baru dengan menggunakan konstruktor default.

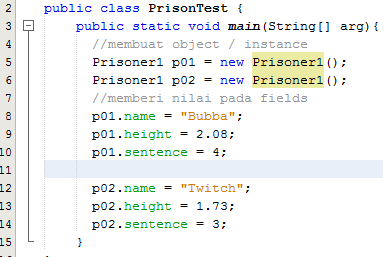
Konstruktor merupakan sebuah method khusus yang digunakan untuk menginisialisasikan object, dan penulisan nama konstruktor harus sama dengan nama classnya, apabila tidak ada konstuktor yang dideklarasikan maka secara otomatis java akan menambakan konstuktor default ke dalam class yang kita buat.

, sebuah konstruktor akan dijalankan pada saat sebuah object diinisialisasikan menggunakan keyword new. Baris 13-15 digunakan untuk mendeklarasikan field atau property yang kita buat di dalam class menggunakan object baru.

Output :



1. Memberi nilai pada fields

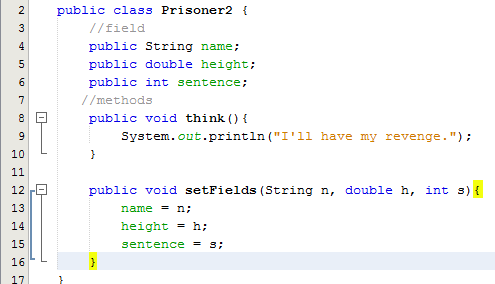


Baris 4 digunakan untuk membuat sebuah instance objet di dalam class baru atau memanggil/mendeklarasikan fungsi class lain di dalam class utama. Baris 8 akhir digunakan untuk memberikan nilai pada fields atau variable yang telah di deklarasikan sebelumnya pada class yang kita panggil, contohnya :

P01.name=”Bubba”; artinya fields/variable name akan kita isikan nilainya dengan kata “Bubba”.karena sebelumnya menggunakan modifier public maka fields tersebut bisa kita akses atau diberikan nilai walaupun di dalam class yang berbeda.

Praktik ini digunakan untuk memasukkan nilai pada fields yang telah kita deklarasikan pada class sebelumnya yaitu Prisoner dengan syarat fields termasuk scope global atau bisa di akses di dalam class yang berbeda.

1. Metode untuk setting nilai field



Pada praktik ini terdapat sebuah method yang digunakan untuk melakukan setting nilai dari fields yang kita deklarasikan dengan kata lain kita akan membuat sebuah method dengan dengan parameter dimana parameter ini nantinya akan digunakan untuk setting nilai dari fileds yang sudah dideklarasikan sebelumnya.

**Public void setFields (String n,double h,int s){**

**name =n;**

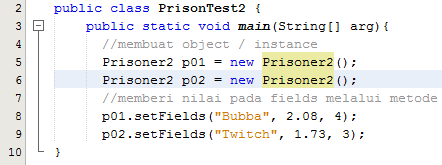
**height=h;**

**sentence=s;**

**}**

artinya nilai dari fields/variable name,height,dan sentence akan kita isi dengan nilai variable n,h dan s yang ada di dalam parameter method setFields, jadi untuk memberikan nilai pada fields/variable tersebut bisa dilakukan melalui method setFields.

1. Memberi nilai lewat metode setFields

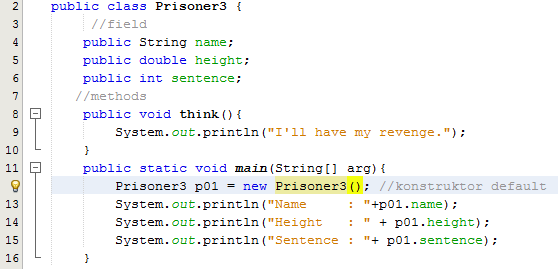


Pada praktik ini merupakan bagian dimana kita akan memberi nilai lewat method yang sebelumnya sudah kita setting untuk mengisi nilai dari dari fields yaitu method setFields.

* **p01.setFields(“Bubba”,2.08,4)** digunakan untuk mengisi nilai dari fields yang telah di setting sebelumnya, dengan cara cukup dengan mengganti nilai parameter dari method tersebut.
* **p01.** merupakan nama instance object yang digunakan untuk mendeklarasikan class lain di dalam class utama atau memaggil fungsi class lain di dalam class baru.

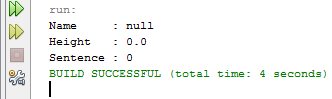
**Praktik 2 : Konstruktor**

1. Konstruktor Default

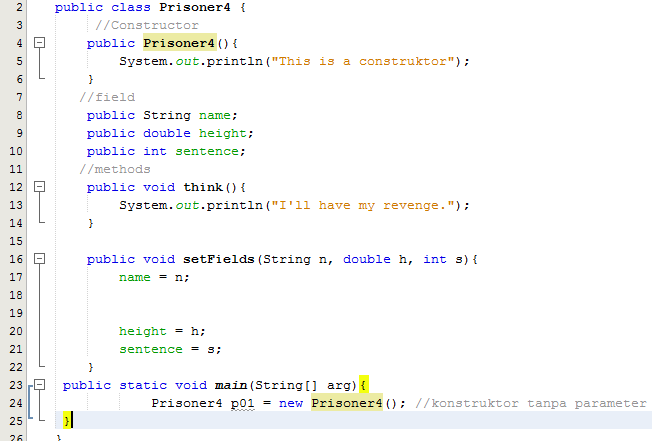


Penjelasan pada praktik ini sama dengan praktik 1a yaitu menggunakan konstruktor. Konstruktor merupakan sebuah method khusus yang digunakan untuk menginisialisasikan object. Karena pada bagian ini tidak ada pendeklarasian sebuah konstruktor maka secara otomatis yang akan digunakan adalah konstruktor default yaitu : **Prisioner2a p01=new Prisioner2a();**.

Output :

****

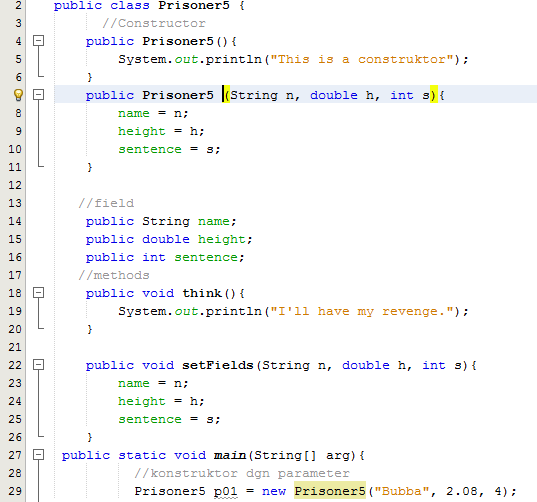
1. Menulis metode Konstruktor



Pada praktik ini merupakan bagian untuk mendeklarasikan sebuah konstruktor. Pada bagian ini penulisan konstruktor terdapat pada baris 4yaitu:

**Public Prisoner4()** seperti yang dijelaskan sebelumnya bahwa penulisan nama konstruktor harus sama dengan nama class yang mendeklarasikannya dan konstruktor juga tidak memiliki nilai pengembalian (return) atau tidak memiliki tipe data. Dari contoh di atas konstruktor tersebut merupakan bentuk dari konstruktor tanpa parameter.

1. Konstruktor dengan parameter untuk setting nilai awal field

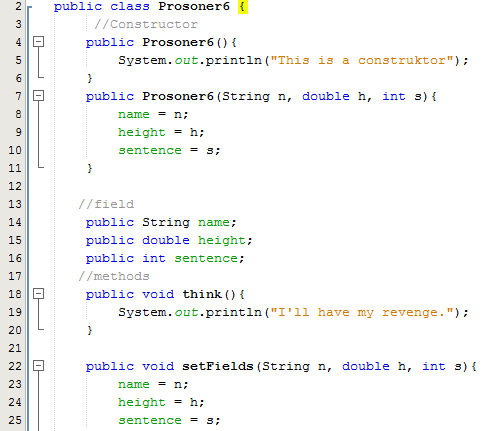
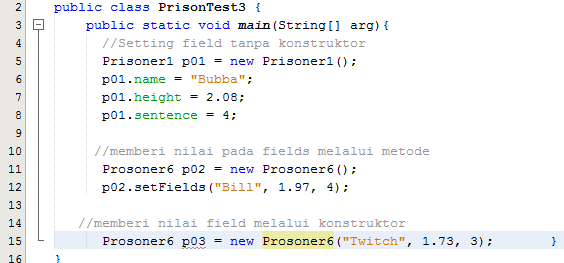


Pada bagian ini setting nilai awal pada fields menggunakan konstruktor, dimana konstruktor harus menggunakan parameter sama seperti method biasa.seperti pada contoh di atas yaitu :

**Public prisoner5(String n,double h,int s);** artinya konstruktor tersebut memiliki 3 parameter dan parameter ini yang akan di gunakan untuk melakukan setting nilai dari fields/atribut yang sudah kita deklarasikan sebelumnya. Parameter bersifat local scope artinya parameter tersebut hanya bisa diakses di dalam method yang mendeklarasikannya saja. Contoh pengisian nilai fields menggunakan parameter konstruktor yaitu :

**Prisoner2c p01=new Prisoner2c(“Bubba”,2.08,4);** artinya nilai yang di inputkan harus urut berdasarkan urutan dari tipe parameter yang dideklarasikan, jika tidak maka nilai tersebut tidak akan bisa di kenali oleh java karena salah tipe data yang mendeklarasikannya.

1. Setting Field

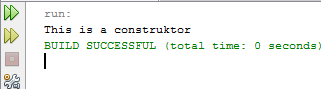
 

Dari script di atas bisa dilihat beberapa cara memberi nilai pada sebuah fields diantaranya :

* Menggunakan instance object yang kita buat di dalam class baru seperti : **p01.name=”Bubba”;** melalui sebuah method seperti :**p02.setFields(“Bill”,1.97,4);**
* Menggunakan sebuah konstruktor seperti :

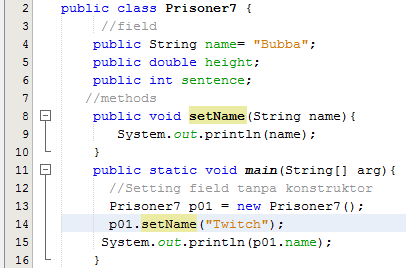
**Prisoner2d p03=new Prisoner(“Twitch”,1.73,3);**

Output :



**Praktik 3. Keyword this**

1. Nama parameter sama dengan nama field



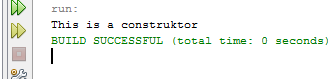
Praktik tersebut merupakan bentuk dimana nama parameter sama dengan nama dari fieldsnya. Dalam java diperbolehkan nama parameter dengan nama fields sama seperti contoh di atas :

**public String name=”Bubba”**

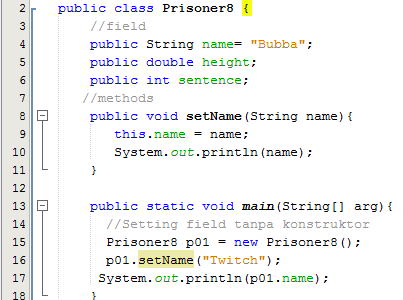
**public void setName(String name)**

Ketika mendeklarasikannya kedalam class utama maka kedua nilai tersebut akan tetep muncul,itu dikarenakan fields yang ada didalam parameter hanya bisa dia akses oleh methodnya saja. Karena nama methodnya berbeda dengan nama fieldsnya maka keduanya akan tetep dieksekusi meskipun nama dari parameter method tersebut sama.

Output :



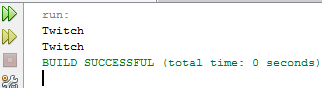
1. Merefensi field name dengan keyword this



Penjelasan Praktik 3b sama dengan Praktik 3a namun pada bagian ini menggunakan keyword this. This digunakan untuk menunjukan/membedakan suatu object atau fields di dalam sebuah method ketika nama fields dan parameter method tersebut sama seperti :

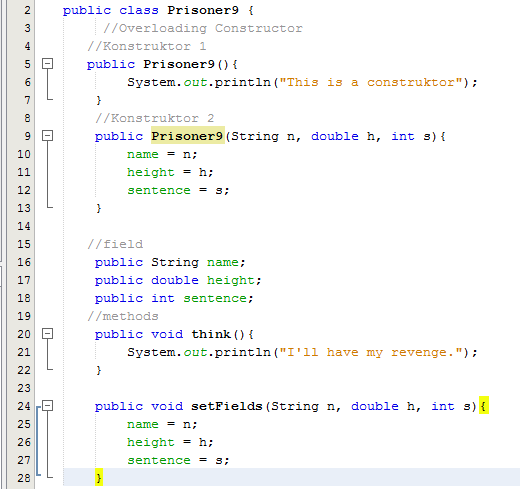
**this.name=name;** artinya bahwa fields/variable yang telah kita deklarasikan sebelumnya sama dengan parameter dari method tesebut. Maka nilai dari outputnya secara otomatis akan sama yaitu Twitch Twitch

Output :



**Praktik 4 : Overloading methods**

1. Overloading Construktor



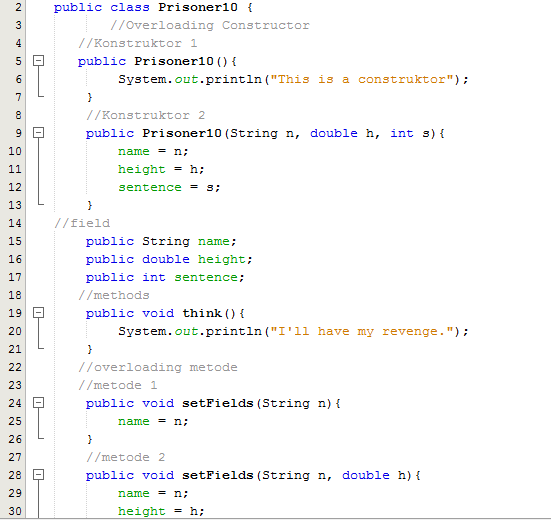
Pada bagian ini merupakan bentuk dari Overloading Konstruktor artinya di dalam java konstruktor boleh dideklarasikan lebih dari satu kali dengan syarat memiliki parameter yang berbeda-beda. Karena java akan menggunakan parameter-parameter tersebut untuk menentukan konstruktor mana yang akan di eksekusi seperti pada contoh di atas yaitu :

**Konstruktor 1 🡪public Prisoner9(){“steatment”}**

**Konstruktor 2 🡪public Prisoner9(String n,double h,int s){“steatment”}**

Dari contoh di atas bisa dilihat bahwa konstruktor 1tidak memiliki parameter dan konstruktor 2 memiliki parameter.

1. Overloading Metode



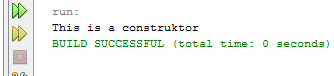
Sama dengan konstruktor, didalam java diperbolehkan memiliki lebih dari satu method dengan nama yang sama dengan syarat memiliki parameter yang berbeda. Karena parameter-parameter tersebut yang akan digunakan java untuk melakukan eksekusi pada method yang mana, seperti contoh di atas :

**Method 1 🡪public void setFields(String n){“Statement”}**

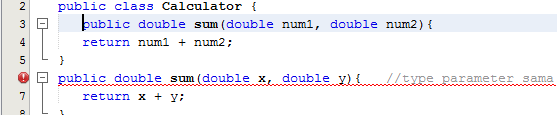
**Method 2 🡪public void setFields(String n,double h){“Statement”}**

**Method 3 🡪public void setFields(String n,double h,int s){“Statement”}**

Output:



1. Parameter yang tidak sesuai

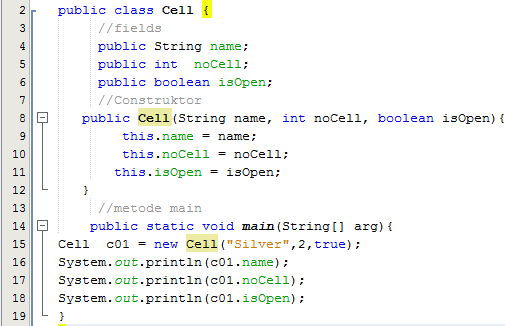


Praktik 4c menunjukan penerapan Overloading method yang salahkarena kedua method tersebut memiliki tipe parameter yang sama dengan nama method yang sama.Maka java tidak akan bisa membaca/mengeksekusi method tersebut karena kedua method tersebut sama meskipun nama variable di dalam parameter method tersebut berbeda.

Java tidak memperbolehkan pendeklarasian 2 variable yang sama,2 method atau konstruktor yang sama dengan nama yang sama dan tipe parameter yang sama, karena secara otomatis salah satunya tidak akan dikenali atau terdeklarasi.

**Praktik 5 : Access Modifier**

1. Akses public

****

Pada bagian ini menjelaskan tentang penggunaan modifier public. Public digunakan untuk mendeklarasikan class,method,fields agar visible/dapat dipanggil dalam package yang berbeda-beda atau yang menggunakan modifier ini bisa di akses di dalam class itu sendiri maupun di luar class tersebut.

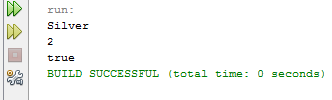
Dari contoh di atas terdapat semua fields,class,konstruktor dan method menggunakan modifier public seperti :

**public String name;** artinya fields/variable name ini bisa diakses di dalam class maupun di luar atau di dalam package yang berbeda-beda.

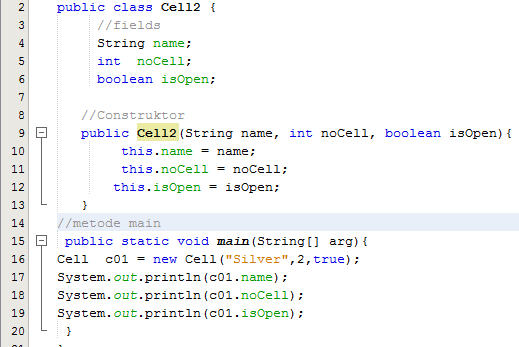
**public class cell{}** artinya class ini bisa di akses di dalam package yang berbeda-beda.

**public cell(){}** artinya konstruktor ini bisa di akses di dalam class atau di luar class tersebut bahkan di dalam package yang berbeda-beda.

Output :



1. Akses Default



Pada bagian ini akan menggunakan modifier default atau tanpa modifier.

Dalam java diperbolehkan mendeklarasikan sebuah fields,class ataupun method tanpa modifier, maka secara otomatis java akan menggunakan modifier defaultnya walaupun tidak terlihat di dalam program. Modifier default merupakan modifier yang visible/dapat diakses hanya dalam package yang sama artinya dapat modifier ini dapat di akses di dalam di dalam class maupun di luar class dengan syarat class tersebut berada di dalam satu package.

Dari contoh di atas bisa dilihat pada pendeklarsian fields dimana :

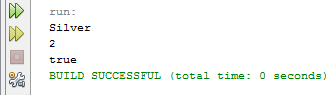
**String name;**

**int noCell**

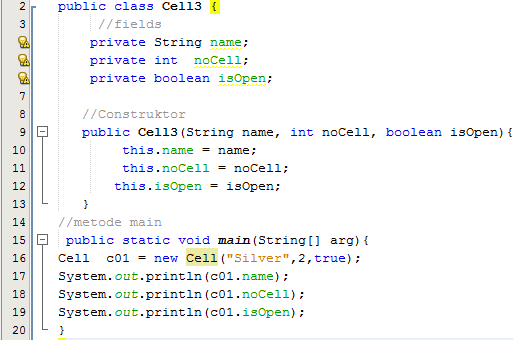
**Boolean isOpen**

Semua fields tersebut langsung dideklarasikan dengan type datanya masing-masing makan java pun secara otomatis akan membuat modifiernya menjadi default.

Output :



1. Akses Private



Praktik sama dengan praktik-praktik diatas,namun pada bagian ini membahas modifier private seperti :

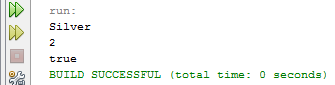
**Private String name;**

**Private int noCell;**

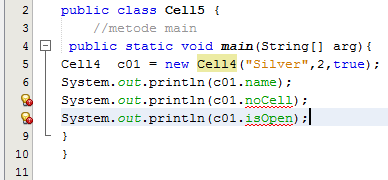
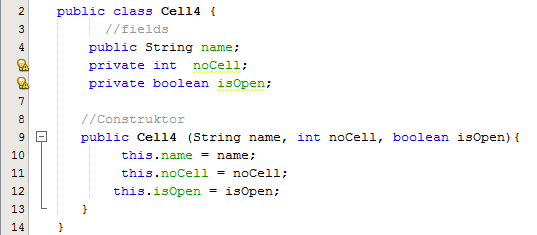
**Private Boolean isOpen;**

Modifier private merupakan bentuk modifier yang hanya bisa diakses di dalam class tersebut atau dengan kata lain tidak bisa diakses di luar class yang mendeklarasikannya. Modifier ini biasanya hanya digunakan untuk mendeklarasikan fields dan method saja.

Output :



1. Mengakses Member of Class



Pada praktik ini hanya digunakan untuk mendeklarasikan sebuah class dengan menggunakan modifier dimana fields dan methodnya menggunakan modifier yang berbeda-beda seperti:

**public String name;** artinya fields ini bisa di akses di dalam class atau diluar class tersebut meskipun dalam package yang berbeda.

**private int noCell;** artinya fields ini hanya bisa di akses di dalam class tersebut.

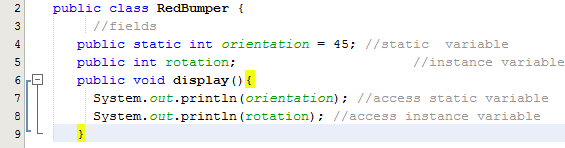
**private boolean isOpen;** artinya fields ini hanya bisa di akses di dalam class tersebut.

**public cell4 (){}** artinya method ini dapat di akses didalam ataupun diluar class tersebut.

Pada modifier private terjadi error saat ditampilkan karena tidak bisa diakses dalam class lain.

**Praktik 6 : Static Variables and Methods**

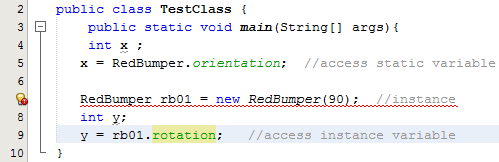
1. Static variable



Praktik ini membahas tentang penggunaan keyword static pada fields/variable dimana. Static yaitu ketika keyword ini digunakan untuk mendeklarasikan member (variable dan method) dari suatu class,maka member class tersebut dapat diakses tanpa menciptakan object class tersebut seperti :

**public static int orientation=45** ini merupakan bentuk dari static variable dan bisa langsung di akses tanpa membuat object class terlebih dahulu.

1. Accesing Static variable



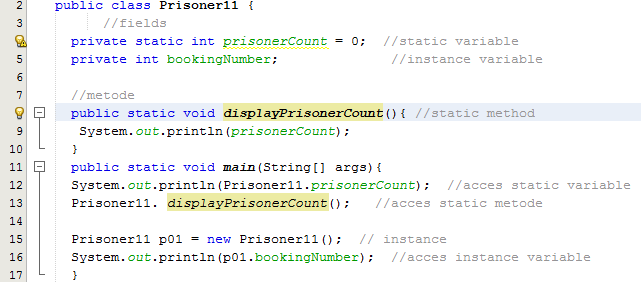
Pada bagian ini merupakan cara mengkases static variable

**RedBumper.orientation;** merupakan bentuk pendeklarasian atau akses dari static variable dimana variable tersebut bisa kita akses tanpa tanpa harus membuat instance object terlebih dahulu. Berbeda dengan instance variable untuk melakukan akses pada variable tersebut kita harus membuat instance object terlebih dahulu seperti : **RedBumper rb01=new RedBumper(90); rb01.rotation;**

Jika instance variable dideklarsikan langsung tanpa pembuatan instance object atau object class maka variable tersebut tidak akan bisa diakses oleh java.

Program ini defaultnya error karena disaat pembuatan instance object yaitu : **RedBumper rb01=new RedBumper(90);** artinya pada class sebelumnya tidak ada konstruktor yang dideklarasikan sehingga java menggunakan kosntruktor default artinya tanpa parameter, ketika pembuatan instance object konstruktor defaultnya diberikan parameter. Maka secara otomatis parameter tersebut tidak akan bisa terbaca oleh java karena tidak pernah di deklarasikan.

1. Static Method



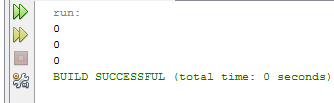
Praktik ini sama dengan praktik di atas hanya saja ini merupakan bentuk dari static method, contoh penulisannya seperti :

**public static int displayPrisonerCount(){}**

Dan bentuk akses dari method tersebut sama seperti akses static variable yaitu :

**Prisoner11.displayPrisonerCount();**

Output :

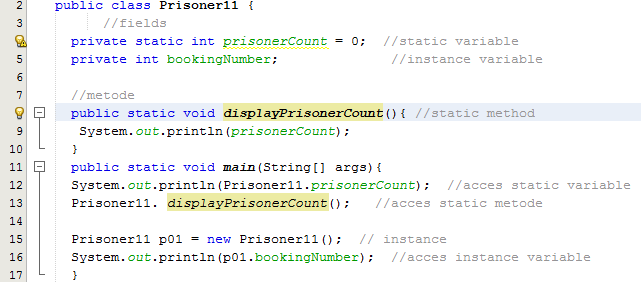


1. **TUGAS**
2. Tugas 1

**Setter :** merupakan sebuah method yang digunakan untuk mengeset nilai atribut ke dalam variable.

**Getter :** merupakan sebuah method yang digunakan untuk mengambil nilai atribut dan di return(dikembalikan).

1. Tugas 2



1. **KESIMPULAN**

Berdasarkan pembahasan praktikum diatas dapat disimpulkan bahwa ada cara atau metode tertentu dalam Java dengan aturan yang berbeda-beda dalam pembuatan program.

1. **LISTING**

Terlampir