

**LAPORAN HASIL PRAKTIKUM**

**MATA KULIAH :**

**TEKNIK PEMROGRAMAN**



Disusun Oleh :

<b>NIM</b>	<b>: 221524061</b>
<b>NAMA</b>	<b>: Thoriq Muhammad Fadhli</b>
<b>KELAS</b>	<b>: 1BD4</b>

**Disusun pada : Minggu, 5 Februari 2023**

## DAFTAR ISI

Kasus 1.....	3
Kasus 2.....	5
Kasus 3.....	6

**Link GitHub :** <https://github.com/tmfadhi12/tekpro/tree/master/Pertemuan3>

## Kasus 1

Hasil Akhir Program :

### barang.java

```
terminal  Help  barang.java - Pertemuan3 - Visual Studio Code
J inventori.java  J Upinpin.java 1  J Item.java  J KelasSatu.java
J barang.java > ? barang > ? tambahStok(int)
1  public class barang {
2      String kode_barang;
3      String nama_barang;
4      private int stok;
5
6      public barang(String kode, String nama, int stk) {
7          kode_barang = kode;
8          nama_barang = nama;
9          stok = stk;
10     }
11
12     public int getStock() {
13         return stok;
14     }
15
16     public void tambahStok(int add) {
17         stok += add;
18     }
19 }
```

### inventori.java

```
inventori.java X  J Upinpin.java 1  J Item.java  J KelasSatu.java  J KelasDua.java 2  J barang.java
inventori.java > ? inventori
public class inventori {
    barang[] barangs;

    void initBarang() {
        barangs = new barang[2];
        barangs[0] = new barang(kode: "001", nama: "Baju", stk: 10);
        barangs[1] = new barang(kode: "002", nama: "Celana", stk: 20);
    }

    void showBarang() {
        System.out.println(barangs[0].nama_barang + "(" + barangs[0].getStock() + ")");
        System.out.println(barangs[1].nama_barang + "(" + barangs[1].getStock() + ")");
    }

    void pengadaan() {
        initBarang();
        barangs[0].tambahStok(add: 10);
        barangs[1].tambahStok(add: 16);
        showBarang();
    }

    Run | Debug
    public static void main(String[] args) {
        inventori beli = new inventori();
        beli.pengadaan();
    }
}
```

**Permasalahan Kasus :**

Carilah Solusi Agar variable stok dilindungi sehingga tidak bisa dilakukan operasi aritmatika selain hanya tambah saja.

**Solusi :**

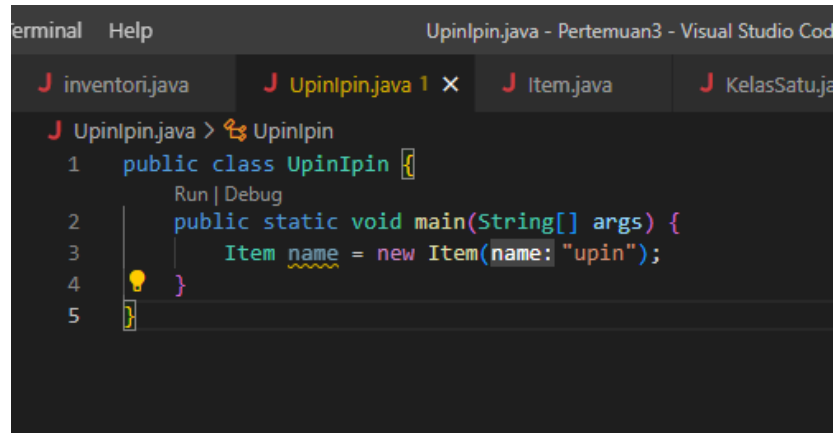
Menggunakan metode *encapsulation*, sehingga stok merupakan private variable dan hanya bisa diubah melalui fungsi tambahStok(), dan hanya melakukan operasi tambah saja.

**Teman yang membantu :** Faris Abulkhoir.

## Kasus 2

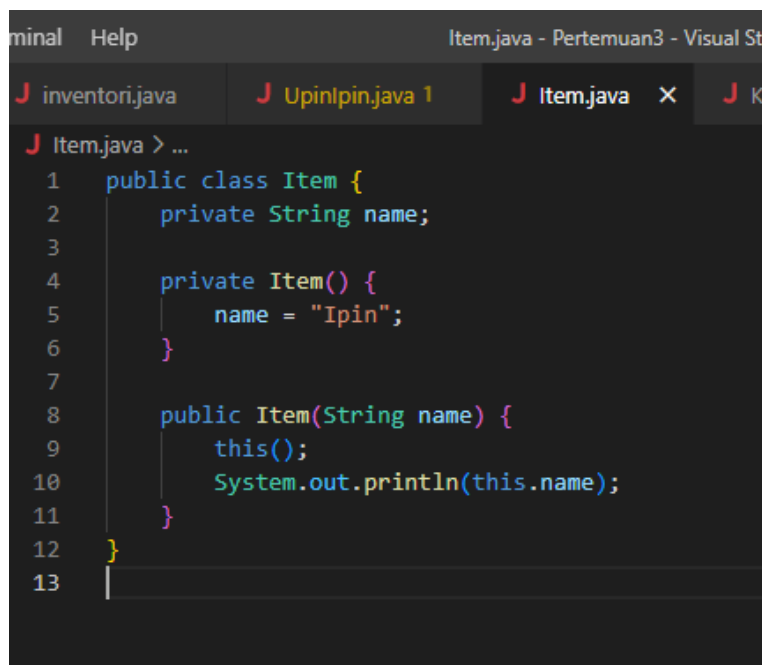
Hasil Akhir Program :

UpinIpin.java

A screenshot of a Visual Studio Code editor window showing the code for UpinIpin.java. The window title is 'UpinIpin.java - Pertemuan3 - Visual Studio Code'. The file explorer on the left shows 'inventori.java', 'UpinIpin.java 1 x', 'Item.java', and 'KelasSatu.java'. The code in the editor is as follows:

```
1 public class UpinIpin {  
2     public static void main(String[] args) {  
3         Item name = new Item(name: "upin");  
4     }  
5 }
```

Item.java

A screenshot of a Visual Studio Code editor window showing the code for Item.java. The window title is 'Item.java - Pertemuan3 - Visual Studio Code'. The file explorer on the left shows 'inventori.java', 'UpinIpin.java 1', 'Item.java x', and 'K'. The code in the editor is as follows:

```
1 public class Item {  
2     private String name;  
3  
4     private Item() {  
5         name = "Ipin";  
6     }  
7  
8     public Item(String name) {  
9         this();  
10        System.out.println(this.name);  
11    }  
12 }  
13
```

**Permasalahan Kasus :** Tampilan Null dan seharusnya menampilkan Ipin.

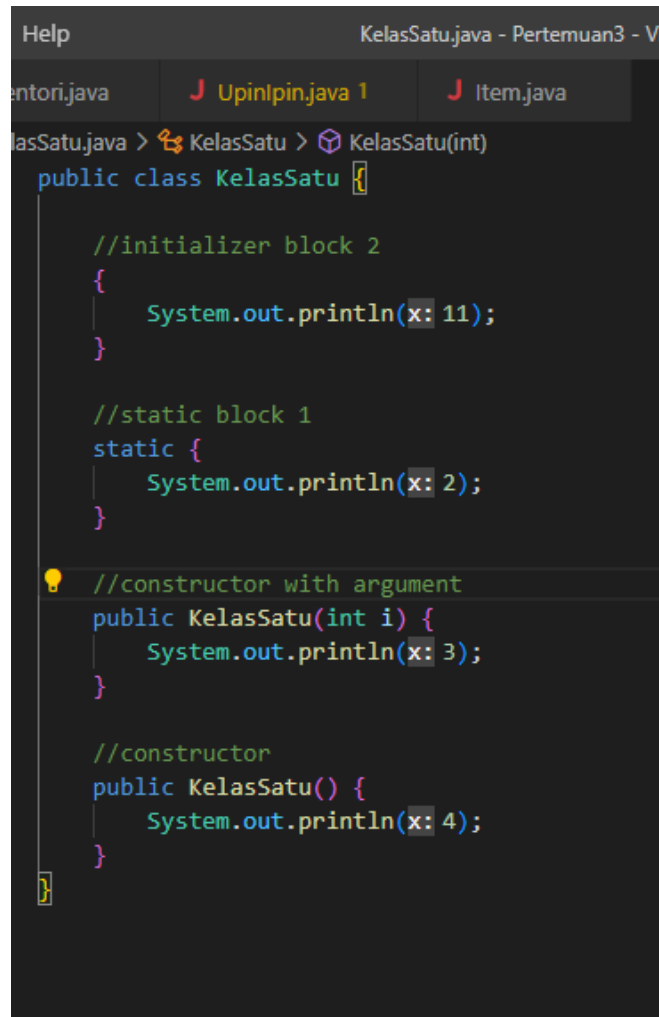
**Solusi :**

Menambahkan this() pada constructor public Item(). Sehingga constructor private Item terpanggil dikarenakan nama constructor private Item() sama dengan nama class Item().

**Teman yang membantu :** Faris Abulkhoir

### Kasus 3

Hasil Akhir Program :



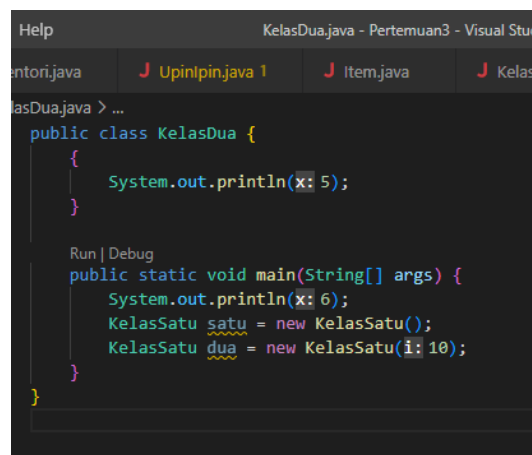
```
Help KelasSatu.java - Pertemuan3 - Vi
entori.java UpinIpin.java 1 Item.java
lasSatu.java > KelasSatu > KelasSatu(int)
public class KelasSatu {

    //initializer block 2
    {
        System.out.println(x: 11);
    }

    //static block 1
    static {
        System.out.println(x: 2);
    }

    //constructor with argument
    public KelasSatu(int i) {
        System.out.println(x: 3);
    }

    //constructor
    public KelasSatu() {
        System.out.println(x: 4);
    }
}
```



```
Help KelasDua.java - Pertemuan3 - Visual Stud
entori.java UpinIpin.java 1 Item.java Kelas
lasDua.java > ...
public class KelasDua {
    {
        System.out.println(x: 5);
    }

    Run | Debug
    public static void main(String[] args) {
        System.out.println(x: 6);
        KelasSatu satu = new KelasSatu();
        KelasSatu dua = new KelasSatu(10);
    }
}
```

## Penjelasan :

**Static Initialization Block** atau *Static Block* , contoh :

```
//static block 1
static {
    System.out.println(x: 2);
}
```

Merupakan block yang dijalankan/dieksekusi paling awal bahkan sebelum main() method pada java. Dan hanya dijalankan 1x, sehingga ketika dipanggil 2x static block tetap hanya mengeksekusi perintah didalamnya 1x.

**Instance Initialization Block**, contoh :

```
//initializer block 2
{
    System.out.println(x: 11);
}
```

Merupakan block yang dijalankan ke-2 jika ada *Static Block* di class yang sama. Dan *Instance Initialization Block* akan berjalan setiap kali ada memanggil block ini. Tidak seperti *Static Block* yang hanya 1x.

**Constructor Without an Argument**, contoh :

```
//constructor
public KelasSatu() {
    System.out.println(x: 4);
}
```

**Constructor With an Argument**, contoh :

```
💡 //constructor with argument
public KelasSatu(int i) {
    System.out.println(x: 3);
}
```

*Constructor* adalah merupakan suatu method yang biasa digunakan untuk menginisialisasi sebuah objek yang kemudian nanti dipanggil di main().

Apabila *constructor* tidak memiliki argument, maka ketika dipanggil (misal: `KelasSatu s = new KelasSatu()`) maka *constructor* `KelasSatu` didalam *class* `KelasSatu` yang tidak memiliki argument akan terpanggil dan sebaliknya jika cara memanggilnya diganti menjadi “`KelasSatu s = new KelasSatu(10)`” maka *constructor* `KelasSatu(int i)` akan terpanggil dikarenakan kita memasukkan sebuah argument ke dalam parameter, Sehingga *constructor* `KelasSatu` yang tidak memiliki argument tidak akan terpanggil.