**LAPORAN PRAKTIKUM**

**PEMOGRAMAN BERORIENTASI OBJECT**

**“Fundamental Programming Stuctures in Java”**

****

Disusun oleh :

Alifah Fisalsabilawati

201511035

2B – D3 Teknik Informatika

**Jurusan Teknik Komputer dan Informatika**

**Program Studi D3 Teknik Informatika**

**Politeknik Negeri Bandung**

**Kasus 2**

* Amati desain setiap class, Apakah desain class tersebut sudah memenuhi konsep OOP yang benar?

Menurut buku Java Fundamental Core 10 th Edition disebutkan ada 7 konsep yang sesuai dengan konsep OOP yaitu :

1. *Always keep data private*

Ketika data dirubah menjadi private, perubahan dalam representasi tidak akan mempengaruhi pengguna kelas, selain itu kita juga bisa lebih mudah mendeteksi bug.

1. *Always initialize data.*

Karena Java tidak akan menginisialisasi variabel lokal untuk kita, tetapi akan menginisialisasi block instance dari objek. Maka dari itu inisialisasi semua variabel secara eksplisit, baik sesuai dengan default atau dengan menyetel default di semua konstruktor.

1. *Don’t use too many basic types in a class.*

Buatlah syntax sesederhana mungkin, hal ini bertujuan untuk membuat kelas Anda lebih mudah dipahami dan diubah.

1. *Not all fields need individual field accessors and mutators.*

untuk mencegah data agar tidak mudah berubah, maka dilakukanlah enkapsulasi.

1. *Break up classes that have too many responsibilities.*

Jangan terlalu banyak menuliskan syntax yang memiliki banyak responbility, maka dari itu kita harus kelompokkan mana saja method yang sesuai dengan class nya.

1. *Make the names of your classes and methods reflect their responsibilities.*

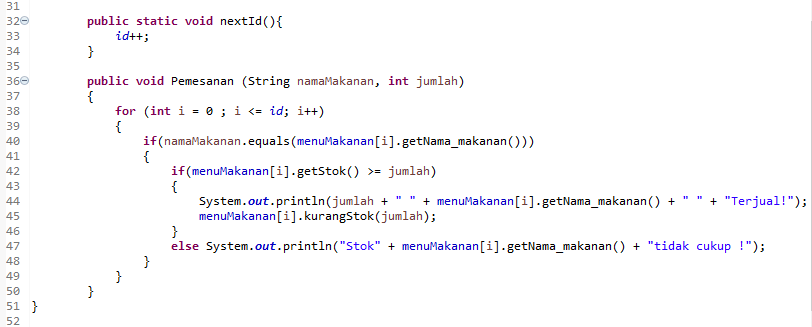
Semua variabel harus memiliki nama yang bermakna dengan apa yang mereka wakili. Penamaan class yang baik biasanya menggunakan kata benda. Selain class method juga harus memiliki nama yang bermakna agar tidak tertukar dengan nama class. Biasanya nama method diawali dengan get dan dilanjutkan dengan nama class nya, sebagai contoh getMakanan

1. *Prefer immutable classes*

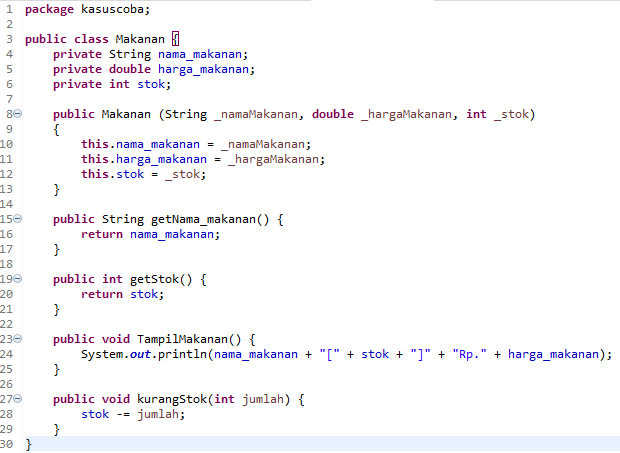
Dan program yang ada pada kasus 2 belum lah sesuai dengan konsep OOP maka dari itu saya membuat class baru yaitu Makanan.java karena pada Restaurant.java terlalu banyak method yang membahas makanan dan merubah data yang awalnya public menjadi private. Berikut merupakan program yang sudah di ubah menggunakan konsep OOP

Restaurant.java

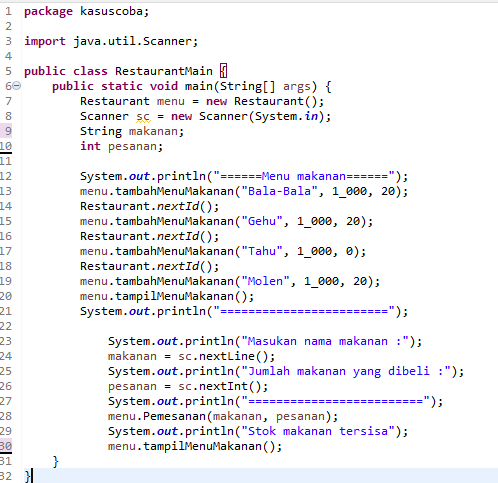




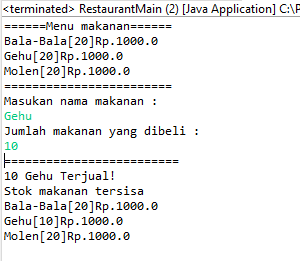
Makanan.java



RestaurantMain.java



**Output**

****

* Ada kebutuhan untuk mengembangkan aplikasi tersebut, dengan menambah fitur pemesanan dan mengurangi setiap stok yang ada. Apakah dengan desain program yang ada dapat dikembangkan?

Saya menambahkan fitur pemesanan yang berfungsi untuk memesan makanan dan mengurangi stok makanan ketika makanan berhasil di pesan.

* Permasalahan yang dihadapi

Jumlah stok yang tidak berkurang terus

* Solusi dari permasalahan yang dihadapi

Saat menuliskan program ternyata saya salah memasukkan variabel, yang harusnya saya menuliskan namaMakanan tapi saya malah menuliskan menuMakanan

**Kasus 3**

* Salin ulang baris kode dan lakukan eksekusi terhadap program tersebut
* Jika dirunning, akan menampilkan output awal sebagai berikut:

****

* Bagaimana cara agar Sasuke dapat menguasai Dojutsu & KekkeiGenkai (\*menampilkan output\*) sebagai berikut:

Rineggan

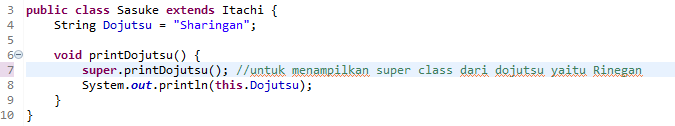
Mangekyou Sharingan

Sharingan

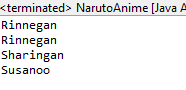
Susanoo

* Pelajari dan jelaskan bagaimana urutan dan cara kerjanya!

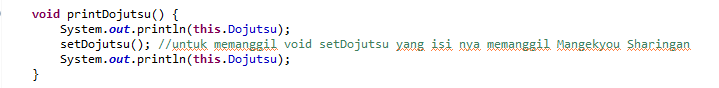
Awalnya output yang muncul hanyalah Sharingan dan Susanoo, untuk menampilkan Rineggan dan Mangekyou Sharingan hal yang pertama saya lakukan adalah menambahkan syntax *super.printDojutsu ()* di Sasuke.java seperti dibawah ini

****

Setelah saya lakukan pencarian ternyata *super.print* merupakan syntax untuk menampilkan atau print dari super class. Karena Sasuke.java merupakan extend dari Itachi.java dan Itachi.java juga ternyata extend dari Rikudo.java yang di dalam class nya berisi **protected** String Dojutsu = "Rinnegan"; dan di wariskan ke Itachi.java yang selanjutnya di Itachi.java di panggil dua kali pada *void printDojutsu ()* yang menyebabkan output program seperti berikut



Maka dari itu langkah selanjutnya saya menambahkan syntax *setDojutsu ()* di Itachi.java seperti program dibawah , hal ini bertujuan untuk menampilkan Mangekyou Sharingan yang ternyata ada pada dalam *void setDojutsu()*

**

Dan akhirnya output final dari program diatas adalah seperti berikut

