**LAPORAN PRAKTIKUM**

**MATA KULIAH PEMROGRAMAN BERBASIS OBJEK**

Dosen Pengampu : Vit Zuraida, S.Kom, M.Kom

**JOBSHEET - 3**

****

**Nama : M. Zidna Billah Faza**

**NIM : 2341760030**

**Prodi : D-IV Sistem Informasi Bisnis**

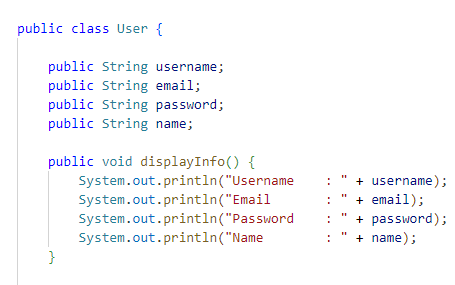
**JURUSAN TEKNOLOGI INFORMASI**

**POLITEKNIK NEGERI MALANG**

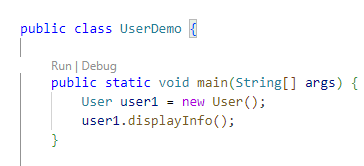
**2024**

**ENKAPSULASI**

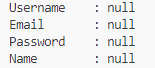
1. **Percobaan 1**
2. Buat folder baru dengan nama Praktikum03.
3. Misalkan di sebuah sistem informasi, terdapat class User yang memiliki atribut username, password, email, dan name. Class User dapat diimplementasikan sebagai berikut.



1. Buat class UserDemo kemudian cobalah membuat objek baru dengan nama user1. Objek user1 diinstansiasi dengan cara memanggil constructor User(). Meskipun constructor User() tidak secara eksplisit dituliskan pada class User, method ini tetap dapat dipanggil karena telah disediakan secara default oleh Java compiler.

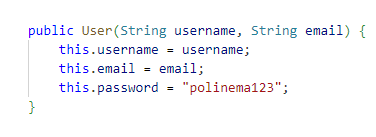


1. Run UserDemo.java maka output yang tampil adalah sebagai berikut.

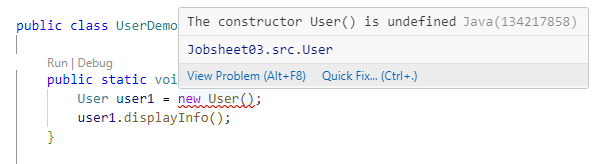


Proses instansiasi adalah pembuatan objek baru dan inisialisasi nilai atribut dengan nilai default (null untuk atribut bertipe data String, 0 untuk atribut bertipe data numerik, dan false untuk atribut bertipe data boolean).

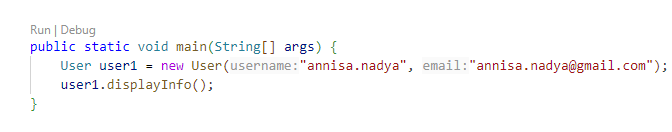
1. Misalnya terdapat requirement bahwa saat suatu objek user dibuat, user tersebut harus sudah memiliki nilai username dan email. Di samping itu, atribut password diberi nilai default, misalnya “polinema123”. Dengan kebutuhan tersebut, kita harus membuat sebuah constructor baru sebagai berikut :



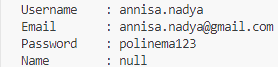
1. Setelah disediakan constructor secara eksplisit, maka constructor default yaitu User() tidak dapat digunakan lagi kecuali dituliskan juga pada class User. (Multiple constructor akan dibahas pada materi overloading & overriding).



1. Instansiasi objek user baru dengan constructor yang telah dibuat pada langkah no 4 bisa dilakukan dengan cara berikut :



1. Output program sebagai berikut:



**Pertanyaan Percobaan 1**

1. Apa fungsi dari constructor?

Fungsi constructor adalah sebuah wadah template ketika akan membuat objek baru. Atau sebuah method khusus yang akan dipanggil ketika membuat objek baru.

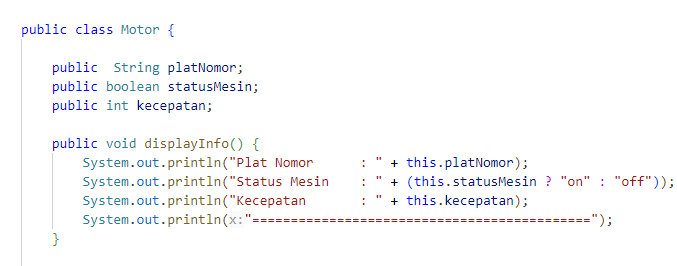
1. Sebutkan keistimewaan constructor dibanding method lain pada suatu class.

Keistimewaan dari construktor adalah karena construktor tidak memerlukan tipe pengembalian (return), construktor juga otomatis dipanggil ketika membuat objek baru.

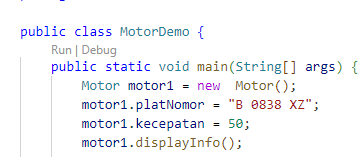
1. Lakukan analisa dan buatlah kesimpulan, apakah constructor bisa memiliki access level modifier private?

Construktor dapat memiliki access level private namun karena private maka kita tidak dapat membbuat objek diluar class tersebut.

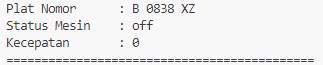
1. **Percobaan 2**
2. Buat class Motor.



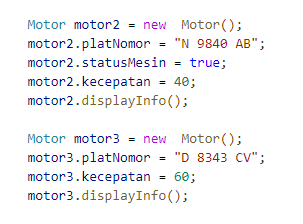
1. Kemudian buatlah class MotorDemo.



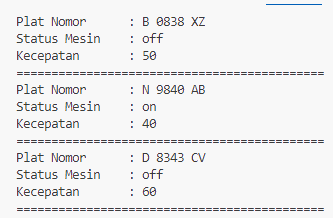
1. Run MotorDemo.java maka outputnya adalah sebagai berikut :



1. Selanjutnya instansiasi 2 objek motor lagi di class MotorDemo.java



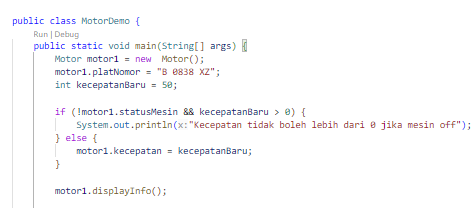
1. Hasilnya sebagai berikut :



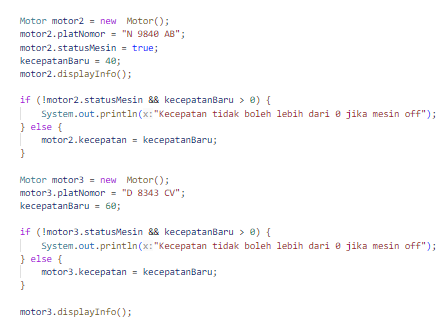
1. Dari hasil di atas, adakah yang janggal?

Pada motor1 dengan plat “B 0838 XZ”, kecepatannya dapat diset dengan nilai 50 padahal status mesin Off. Hal ini karena belum tersedia kontrol/batasan terhadap atribut kecepatan. Padahal, objek di dunia nyata selalu memiliki batasan/aturan dalam memberi nilai atribut serta mekanisme bagaimana objek tersebut dapat berfungsi/melakukan sesuatu. Misalnya motor yang harus dalam keadaan menyala ketika kecepatan lebih dari 0. Kejanggalan ini juga terjadi pada motor ketiga dengan plat nomor "D 8343 CV".

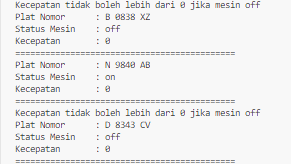
1. Untuk mengatasi hal tersebut, nilai kecepatan baru perlu dicek terlebih dahulu sebelum diassign ke atribut kecepatan.



1. Lakukan pengecekan yang sama untuk motor2 dan motor3.



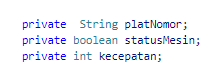
1. Run MotorDemo.java dan perhatikan bahwa sudah terdapat validasi nilai kecepatan terhadap status mesin untuk setiap objek motor.



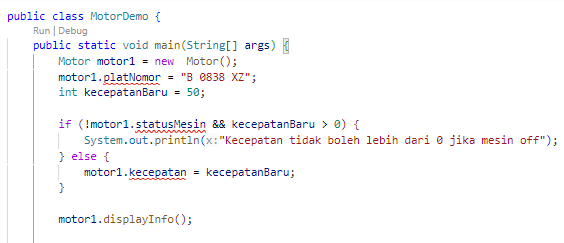
1. **Percobaan 3**
2. Bayangkan bahwa developer baru saja ingat bahwa seharusnya kecepatan tidak boleh lebih dari 0 jika status mesin off setelah membuat 20 objek motor di MotorDemo.java, 10 objek motor di MotorDemo2.java, 25 objek MotorDemo3.java. Pengecekan harus dilakukan 55 kali.
3. Konsep enkapsulasi penting untuk diimplementasikan dalam pemrograman berorientasi objek sehingga perubahan requirement hanya perlu dihandle pada “gerbang” yang dikelola di dalam class tersebut.

Pada OOP, konsep enkapsulasi diimplementasikan dengan cara :

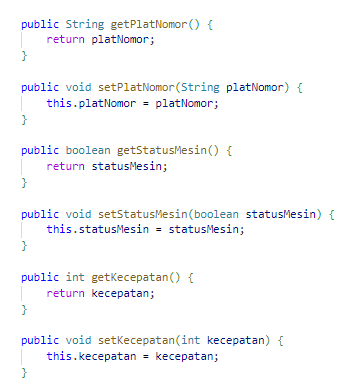
1. Menyembunyikan atribut internal (platNomor, statusMesin, dan kecepatan) dari luar/class lain dengan mengubah access level modifier menjadi private
2. Menyediakan setter dan getter untuk memanipulasi dan mengakses nilai atribut tersebut
3. Ubah access level modifier atribut menjadi private.



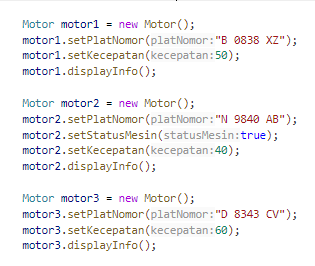
1. Setelah berubah menjadi private, atribut platNomor, statusMesin, dan kecepatan tidak bisa diakses dari luar class (muncul compile error).

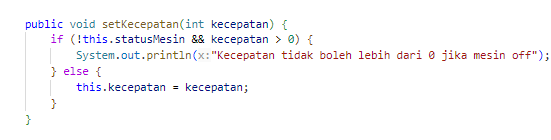


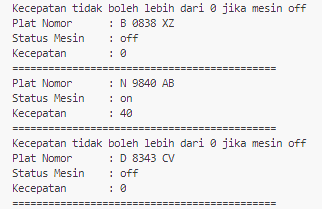
1. Selanjutnya perlu dibuat setter dan getter untuk setiap atribut (Jika terdapat requirement suatu atribut read-only, maka tidak perlu dibuat setternya dan jika suatu atribut write-only, maka tidak perlu dibuat getternya)



1. Dengan enkapsulasi, nilai atribut diakses menggunakan getter dan dimanipulasi menggunakan setter sebagai berikut (masih belum ada validasi nilai kecepatan terhadap status mesin).



1. Dengan menerapkan enkapsulasi, perubahan requirement di tengah implementasi program dapat dilakukan dengan lebih mudah. Pada setter kecepatan, dilakukan validasi nilai kecepatan terhadap status mesin sebagai berikut :
2. Run MotorDemo.java. Outputnya sebagai berikut :



1. Setter dan getter dipakai sebagai “gerbang” untuk mengakses atau memodifikasi atribut yang bernilai private. Hal ini akan membuat kontrol atau validasi atribut lebih mudah dilakukan. Jika ada perubahan requirement di kemudian hari, maka hanya perlu dilakukan modifikasi pada method setKecepatan() tanpa perlu melakukan perubahan berulang kali di seluruh program yang melakukan assignment nilai atribut kecepatan motor. Pengujian terhadap perubahan requirement juga dapat dilakukan pada scope yang lebih kecil, tanpa harus menguji keseluruhan sistem.

**Pertanyaan Perecobaan 3**

1. Pada class MotorDemo, saat kita menambah kecepatan untuk pertama kalinya, mengapa muncul peringatan “Kecepatan tidak boleh lebih dari 0 jika mesin off”?

Karena program menjalankan memthod setKecepatan() terlebih dahulu kemudian menjalankan method displayInfo().

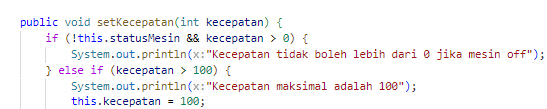
1. Mengapa atribut merek, kecepatan, dan statusMesin sebaiknya diset private?

Agar nilai atribute tidak diubah secara sembarangan, namun hanya bisa melalui setter dan getter

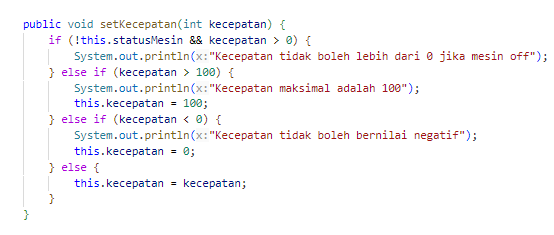
1. Apa fungsi dari setter dan getter?

Setter berfungsi untuk mengisi nilai atribute (write), sedangkan Getter berfungsi untuk mengambil atau membaca nilai atribute (read).

1. Modifikasi class Motor sehingga kecepatan maksimalnya adalah 100.



1. Modifikasi class Motor sehingga kecepatan nya tidak boleh bernilai negatif

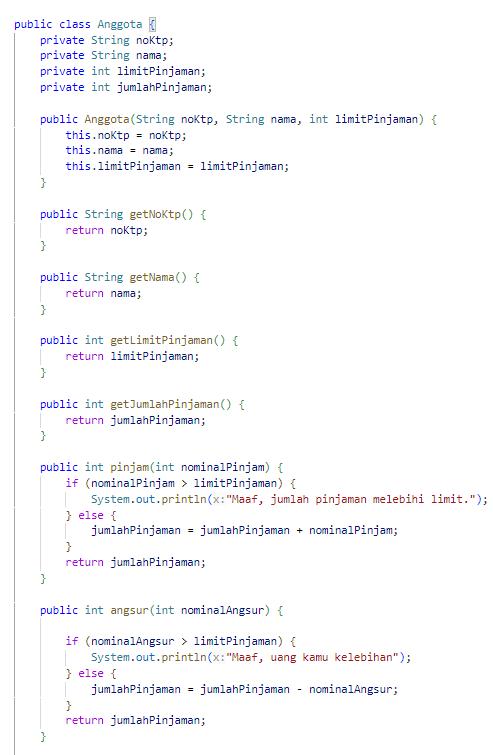


1. **Tugas**
2. Pada sebuah sistem informasi koperasi simpan pinjam, terdapat class Anggota yang memiliki atribut nomor KTP, nama, limit peminjaman, dan jumlah pinjaman. Anggota dapat meminjam uang dengan limit peminjaman yang ditentukan. Anggota juga dapat mengangsur pinjaman. Ketika anggota tersebut mengangsur pinjaman, maka jumlah pinjaman akan berkurang sesuai dengan nominal yang diangsur

Buatlah class Anggota tersebut dengan konsep enkapsulasi. Berikan atribut, method, dan constructor sesuai dengan kebutuhan. Uji dengan TestKoperasi.java berikut ini untuk memeriksa apakah class Anggota yang anda buat telah sesuai dengan yang diharapkan.

Perhatikan bahwa nilai atribut pinjaman tidak dapat diubah secara random dari luar class, tetapi hanya dapat diubah melalui method pinjam() dan angsur().

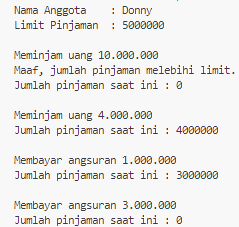
1. Modifikasi class Anggota agar nominal yang dapat diangsur minimal adalah 10% dari jumlah pinjaman saat ini. Jika mengangsur kurang dari 10%, maka muncul peringatan “Maaf, angsuran harus 10% dari jumlah pinjaman”
2. **Hasil Tugas (Program dan Output)**
3. Program Pada class Anggota sebelum terdapat minimal angsur



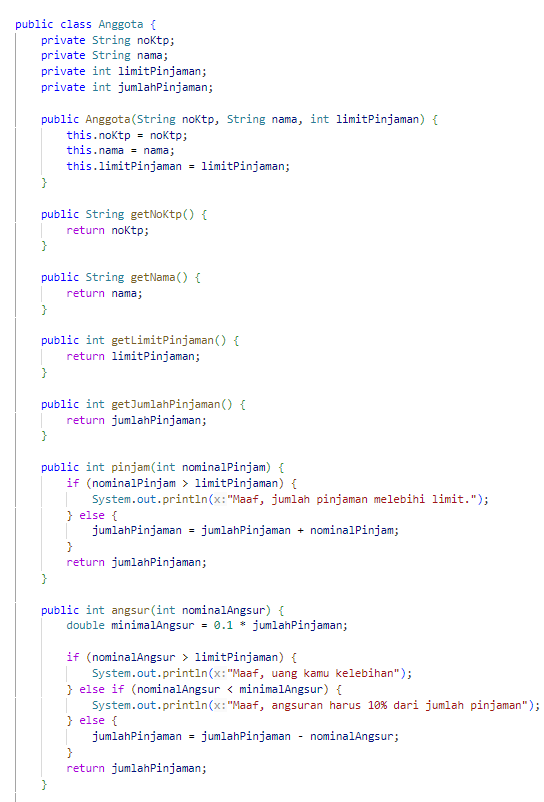
1. Program Pada class TestKoperasi



1. Output



1. Program pada class Anggota setelah terdapat minimal angsur



1. Output dengan minimal angsur

