Laporan Jobsheet 3 - Enkapsulasi pada Pemrograman Berorientasi Objek

Nama : Rizqi Rohmatul Huda

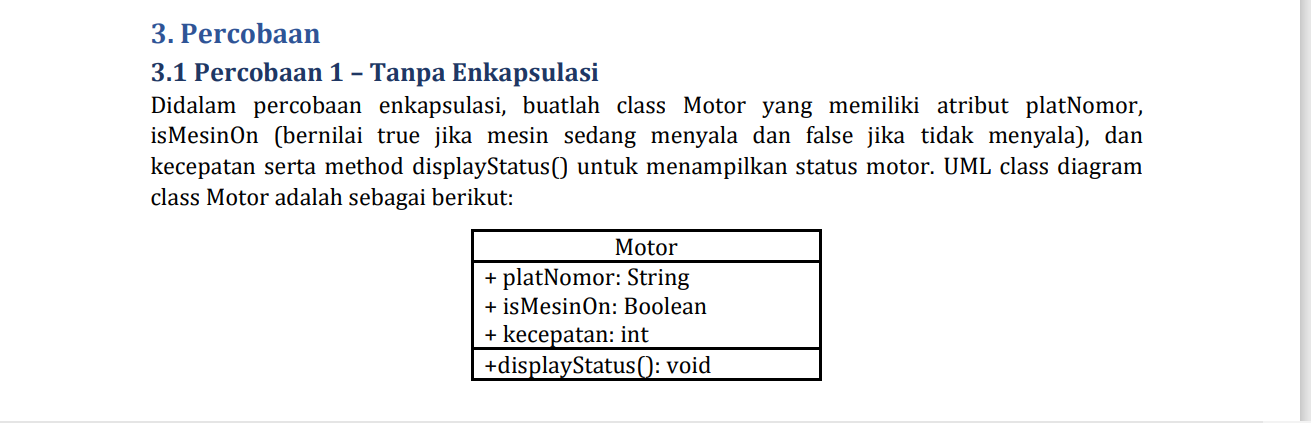
Kelas : 2G – TI

No.Absen : 26

NIM : 2141720264

# 3. Percobaan

# 3.1 Percobaan 1 – Tanpa Enkapsulasi



1. Buka Netbeans, buat project Jobsheet03.

2. Buat class Motor. Klik kanan pada package jobsheet03 – New – Java Class.

3. Ketikkan kode class Motor dibawah ini

package pemrogramanberbasisobjek.pertemuan3;

 public class Motor {

    public String platNomor;

    public *boolean* isMesinOn;

    public *int* kecepatan;

    public *void* displayStatus(){

        System.out.println("Plat Nomor: " + *this*.platNomor);

        if(isMesinOn) {

            System.out.println("Mesin On");

        }

        else {

            System.out.println("Mesin Off");

        }

        System.out.println("Kecepatan: " + *this*.kecepatan);

        System.out.println("=====================");

    }

}

4. Kemudian buat class MotorDemo, ketikkan kode berikut ini.

package pemrogramanberbasisobjek.pertemuan3;

 public class MotorDemo {

    public static *void* main(String[] *args*) {

        Motor motor1 = **new** Motor();

        motor1.displayStatus();

        motor1.platNomor = "B 0838 XZ";

        motor1.kecepatan = 50;

        motor1.displayStatus();

      }

}

5. Hasilnya adalah sebagai berikut:



6. Selanjutnya buatlah 2 objek motor lagi di class MotorDemo.java

 Motor motor2 = **new** Motor();

        motor2.platNomor = "N 9840 AB";

        motor2.isMesinOn = true;

        motor2.kecepatan = 40;

        motor2.displayStatus();

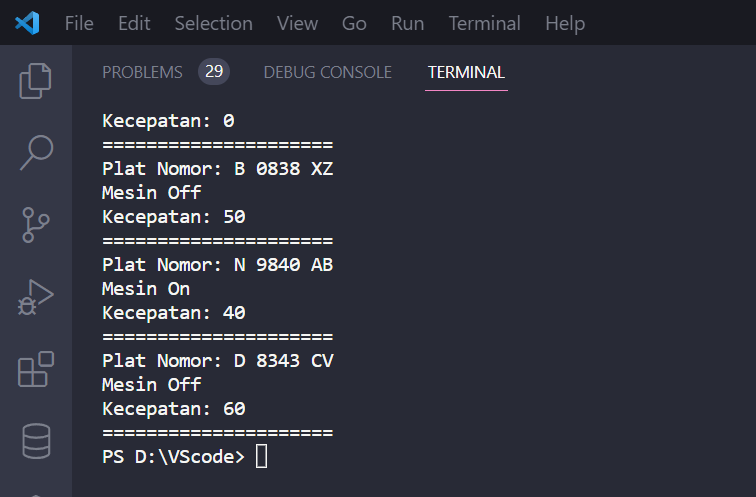
        Motor motor3 = **new** Motor();

        motor3.platNomor = "D 8343 CV";

        motor3.kecepatan = 60;

        motor3.displayStatus();

7. Hasilnya sebagai berikut :



8. Dari hasil di atas, adakah yang janggal?

Pada motor1 dengan plat “B 0838 XZ”, kecepatannya dapat berubah dari 0 ke 50 padahal

mesin motor masih dalam kondisi Off. Bagaimana mungkin atribut kecepatan bernilai 50

padahal mesin masih Off? Hal ini karena belum tersedia kontrol/batasan terhadap atribut

kecepatan. Padahal, objek di dunia nyata selalu memiliki batasan dan mekanisme

bagaimana objek tersebut dapat digunakan. 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".

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

package pemrogramanberbasisobjek.pertemuan3;

 public class MotorDemo {

    public static *void* main(String[] *args*) {

        Motor motor1 = **new** Motor();

        motor1.displayStatus();

        motor1.platNomor = "B 0838 XZ";

*int* kecepatanBaru = 50;

        if(!motor1.isMesinOn && kecepatanBaru > 50){

            System.out.println("Kecepatan tidak boleh dari 0 jika mesin off");

        }else {

            motor1.kecepatan = kecepatanBaru;

        }

        motor1.displayStatus();

    }

}

10. Lakukan pengecekan yang sama untuk motor2 dan motor3

Motor motor2 = **new** Motor();

        motor2.platNomor = "N 9840 AB";

        motor2.isMesinOn = true;

        motor2.kecepatan = 40;

        if(!motor2.isMesinOn && kecepatanBaru > 0){

            System.out.println("Kecepatan tidak boleh lebih dari 0 jika mesin off");

        }

        else {

            motor2.kecepatan = kecepatanBaru;

        }

        motor2.displayStatus();

        Motor motor3 = **new** Motor();

        motor3.platNomor = "D 8343 CV";

        motor3.kecepatan = 60;

        if(!motor3.isMesinOn && kecepatanBaru > 0){

          System.out.println("Kecepatan tidak boleh lebih dari 0 jika mesin off");

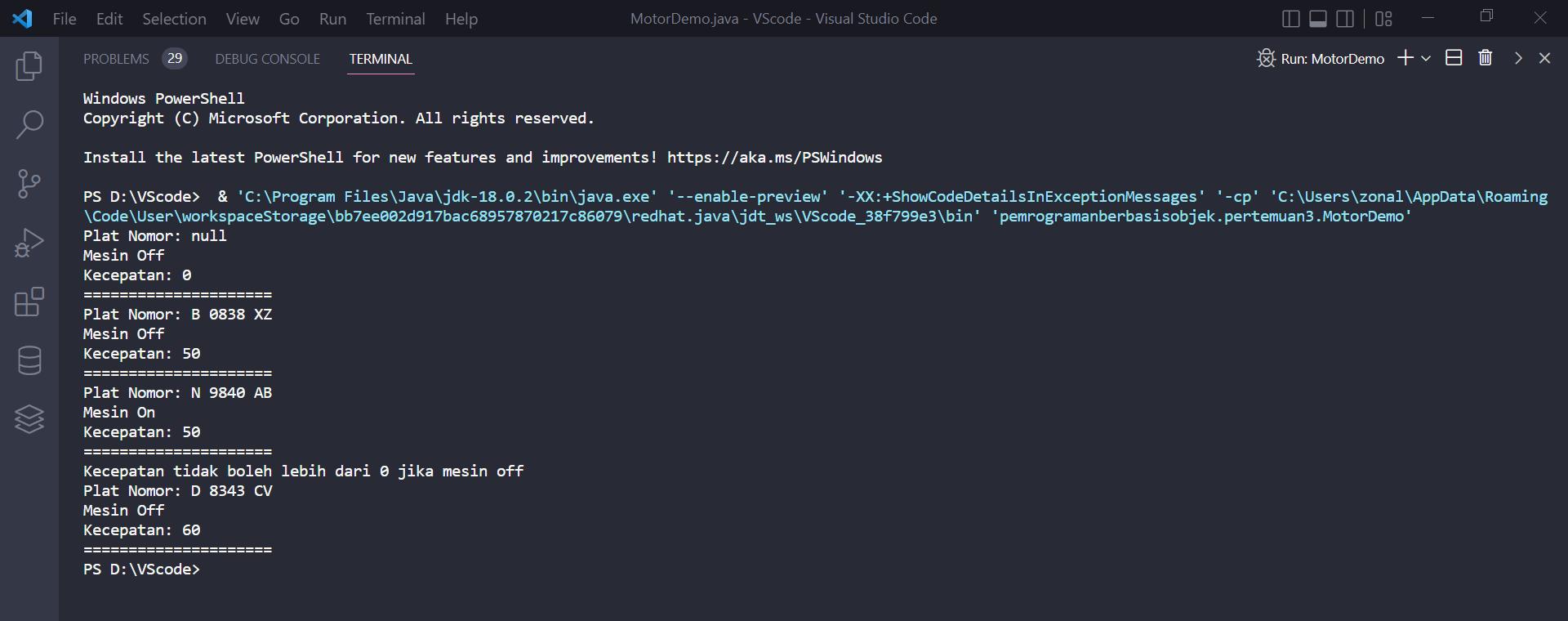
        } else {

            motor3.kecepatan = kecepatanBaru;

        }

        motor3.displayStatus();

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



# 3.2 Percobaan 2 – Enkapsulasi

1. Bayangkan bahwa developer baru ingat bahwa seharusnya kecepatan tidak boleh lebih dari

0 jika status mesin tidak menyala setelah membuat 20 objek motor di MotorDemo.java, 10

objek motor di MotorDemo2.java, 25 objek MotorDemo3.java? Pengecekan harus dilakukan

55 kali.

2. Lalu, bagaimana kita bisa memperbaiki class Motor diatas agar dapat digunakan dengan

baik? Di sinilah pentingnya melakukan enkapsulasi dalam pemrograman berorientasi objek.

Struktur internal class Motor harus disembunyikan dari class lain.

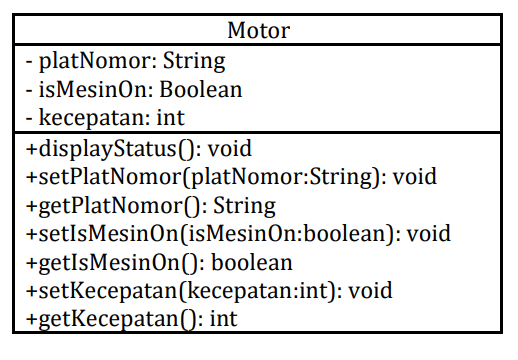
Pada OOP, konsep enkapsulasi diimplementasikan dengan cara:

a. Menyembunyikan atribut internal (platNomor, isMesinOn, dan kecepatan) dari

luar/class lain dengan mengubah access level modifier menjadi private

b. Menyediakan setter dan getter untuk memanipulasi dan mengakses nilai atribut

tersebut



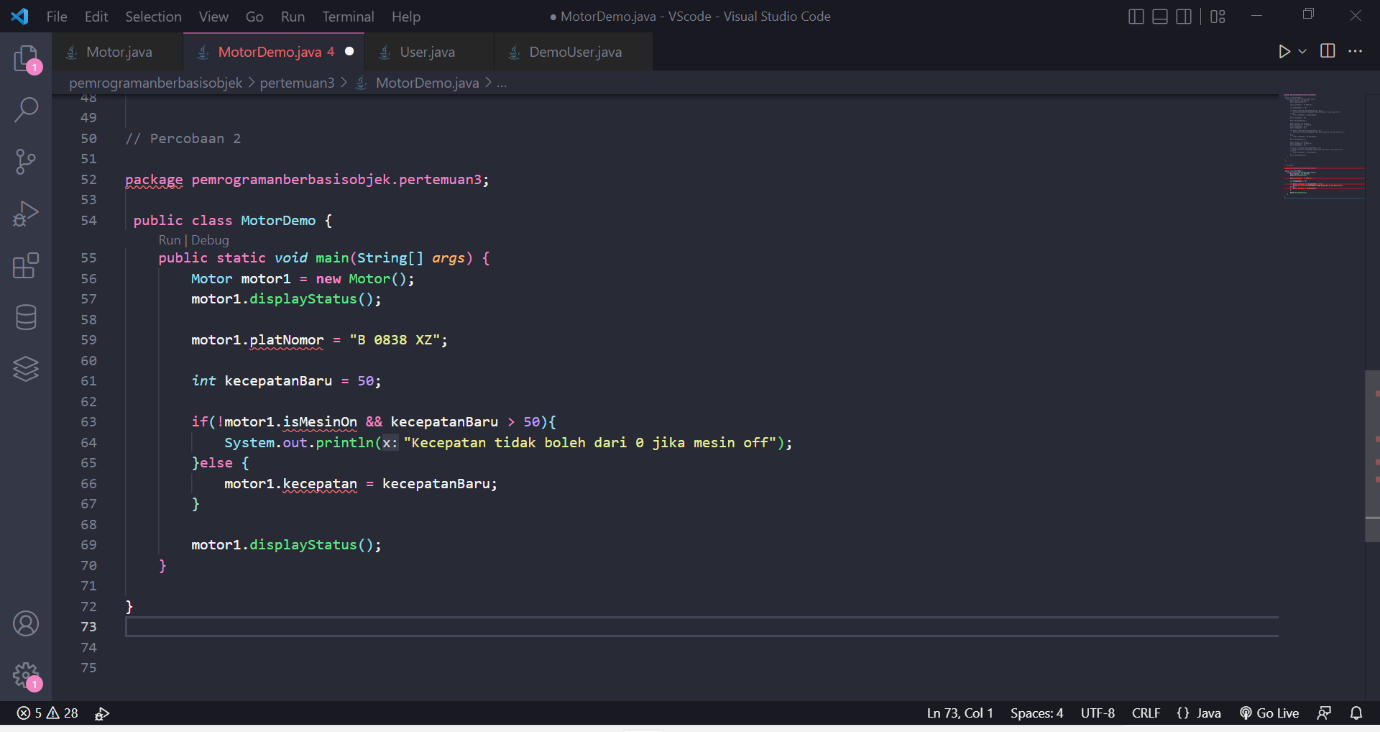
3. Ubah access level modifier menjadi private

  private String platNomor;

    private *boolean* isMesinOn;

    private *int* kecepatan;

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



5. Selanjutnya perlu di buat setter dan getter untuk setiap atribut.

public String getPlatNomor() {

        return *this*.platNomor;

    }

    public *void* setPlatNomor(String *platNomor*) {

*this*.platNomor = *platNomor*;

    }

    public *boolean* isIsMesinOn() {

        return *this*.isMesinOn;

    }

    public *void* setIsMesinOn(*boolean* *isMesinOn*) {

*this*.isMesinOn = *isMesinOn*;

    }

    public *int* getKecepatan() {

        return *this*.kecepatan;

    }

    public *void* setKecepatan(*int* *kecepatan*) {

*this*.kecepatan = *kecepatan*;

    }

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

Motor motor1 = **new** Motor();

        motor1.displayStatus();

        motor1.setPlatNomor("B 0838 XZ");

        motor1.setKecepatan(50);

        motor1.displayStatus();

        Motor motor2 = **new** Motor();

        motor2.setPlatNomor("N 9840 AB");

        motor2.setIsMesinOn(true);

        motor2.setKecepatan(140);

        motor2.displayStatus();

        Motor motor3 = **new** Motor();

        motor3.setPlatNomor("D 8343 CV");

        motor3.setKecepatan(-1);

        motor3.setIsMesinOn(true);

        motor3.displayStatus();

7. 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:

public *void* setKecepatan(*int* *kecepatan*){

        if(!*this*.isMesinOn && *kecepatan* > 0){

            System.out.println("Kecepatan tidak boleh lebih dari 0 jika mesin off");

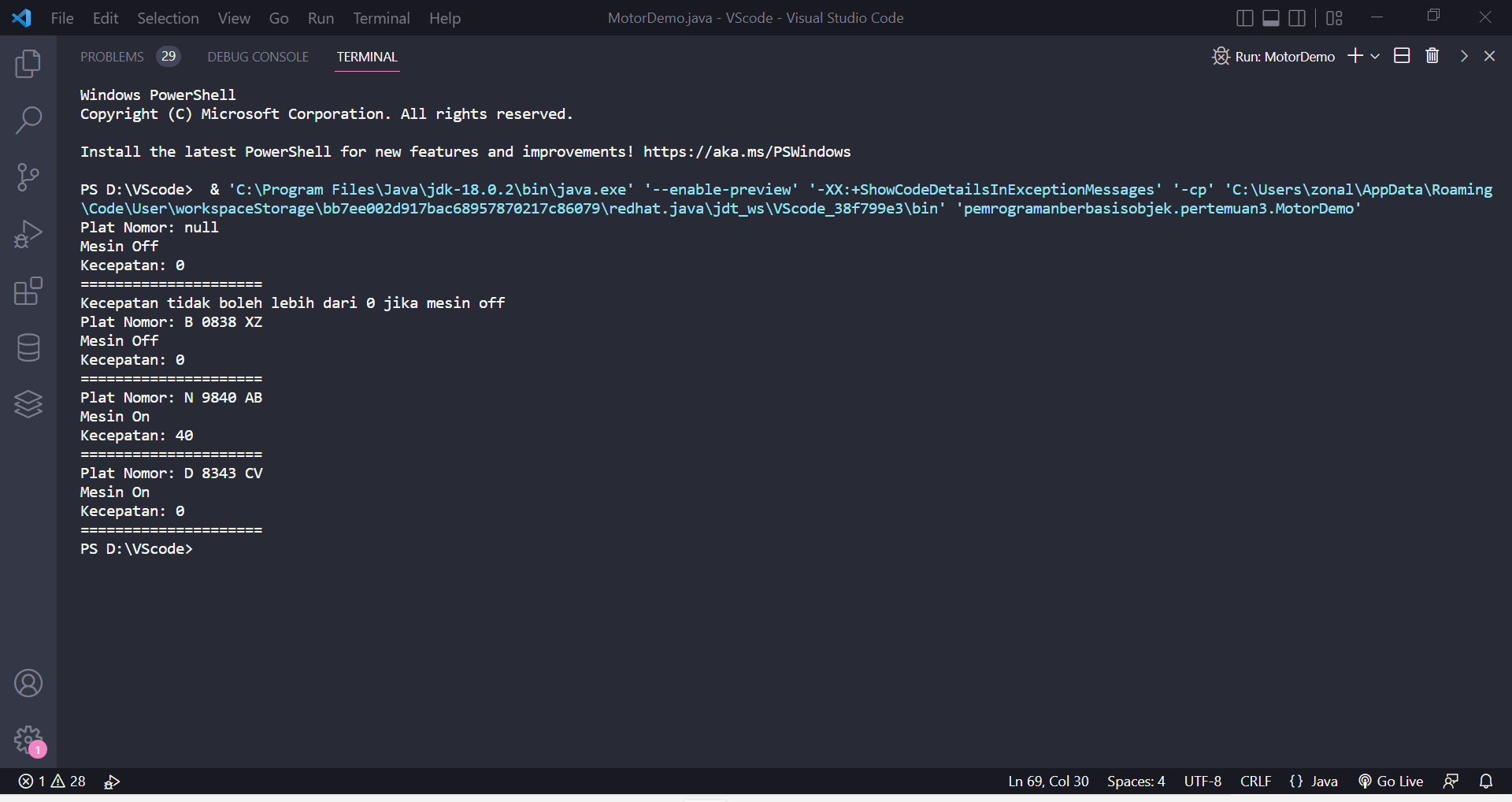
        }

        else {

*this*.kecepatan = *kecepatan*;

        }

8. Run MotorDemo.java. Hasilnya sebagai berikut:



9. 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, misalnya atribut kecepatan tidak boleh bernilai negatif, hanya perlu dilakukan modifikasi pada setKecepatan() tanpa perlu melakukan perubahan berulang kali di seluruh program yang melakukan assignment nilai kecepatan motor.

**3.3 Pertanyaan**

1. Pada class MotorDemo, saat kita menambah kecepatan untuk pertama kalinya, mengapa

muncul peringatan “Kecepatan tidak bisa bertambah karena Mesin Off!”?

Jawab :

*int* kecepatanBaru = 50;

        if(!motor1.isMesinOn && kecepatanBaru > 50){

            System.out.println("Kecepatan tidak boleh dari 0 jika mesin off");

        }else {

            motor1.kecepatan = kecepatanBaru;

        }

Karena jika Boolean isMesinOn bernilai false ataupun belum didefinisikan maka akan tetap bernilai false sehingga akan muncul peringatan "Kecepatan tidak boleh dari 0 jika mesin off".

1. Mengapat atribut merek, kecepatan, dan statusMesin diset private?

Jawab : Karena kalau diset private maka atribut merek, kecepatan, dan statusMesin tidak akan terubah-ubah secara random dan tidak dapat mengalami perubahan di class lain, selain itu juga memudahkan dalam melakukan control dalam perubahan atribut sehingga tidak perlu membuat kondisi satu per satu.

1. Apa fungsi dari setter dan getter?

Jawab : Fungsi dari setter dan getter

* Getter

Public method yang berfungsi mengembalikan nilai dari atribut private

* Setter

Public method yang berfungsi untuk memanipulasi nilai dari atribut private

1. Ubah class Motor sehingga kecepatan maksimalnya adalah 100!

Jawab :

Code pada class Motor

 public *void* setKecepatan(*int* *kecepatan*){

        if(!*this*.isMesinOn && *kecepatan* > 0){

            System.out.println("Kecepatan tidak boleh lebih dari 0 jika mesin off");

        }

        else {

            if(*kecepatan* > 100){

                System.out.println("Kecepatan tidak boleh lebih dari 100");

            }

*this*.kecepatan = *kecepatan*;

        }

    }

Code pada class MotorDemo

    Motor motor2 = **new** Motor();

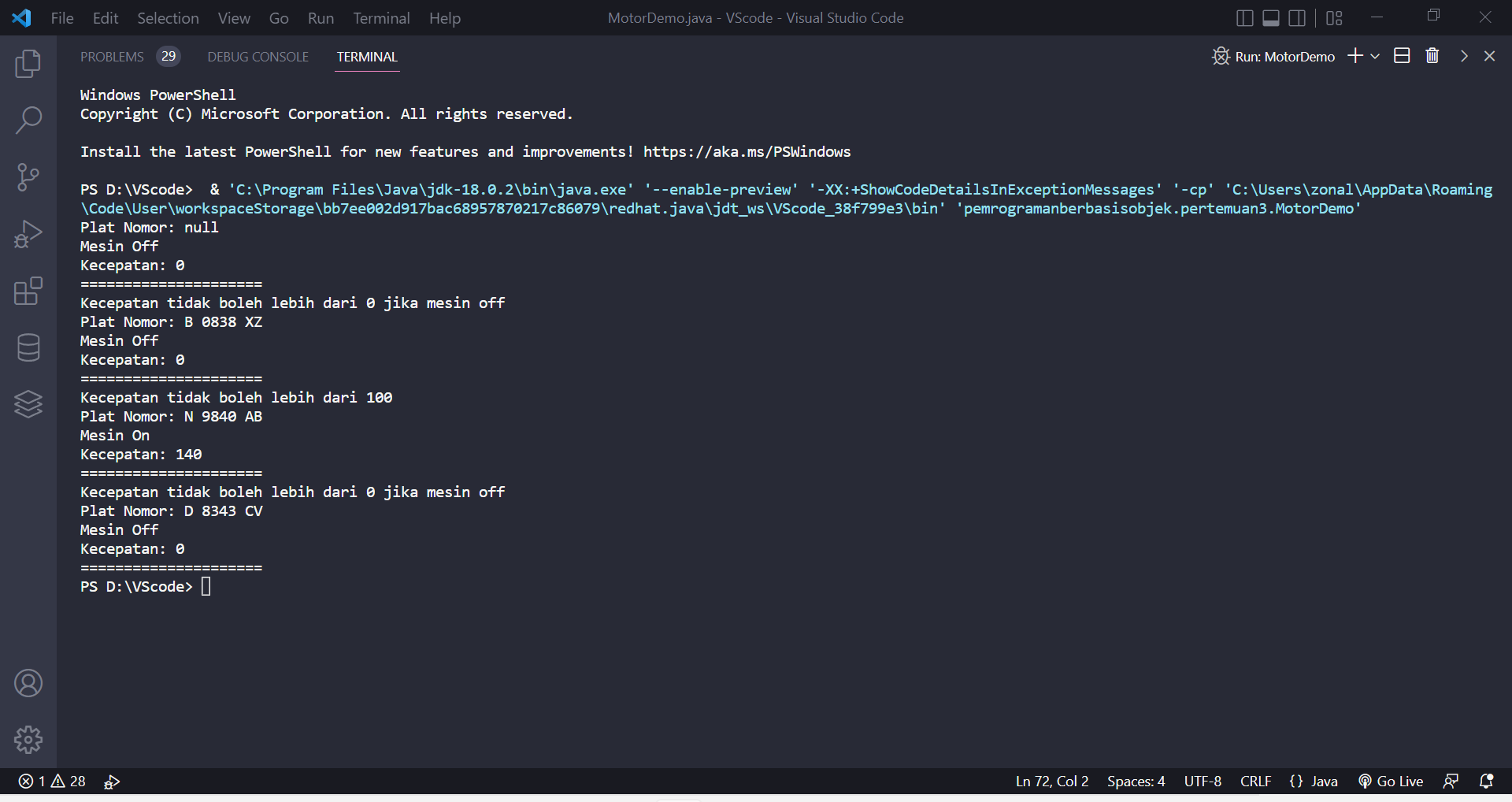
        motor2.setPlatNomor("N 9840 AB");

        motor2.setIsMesinOn(true);

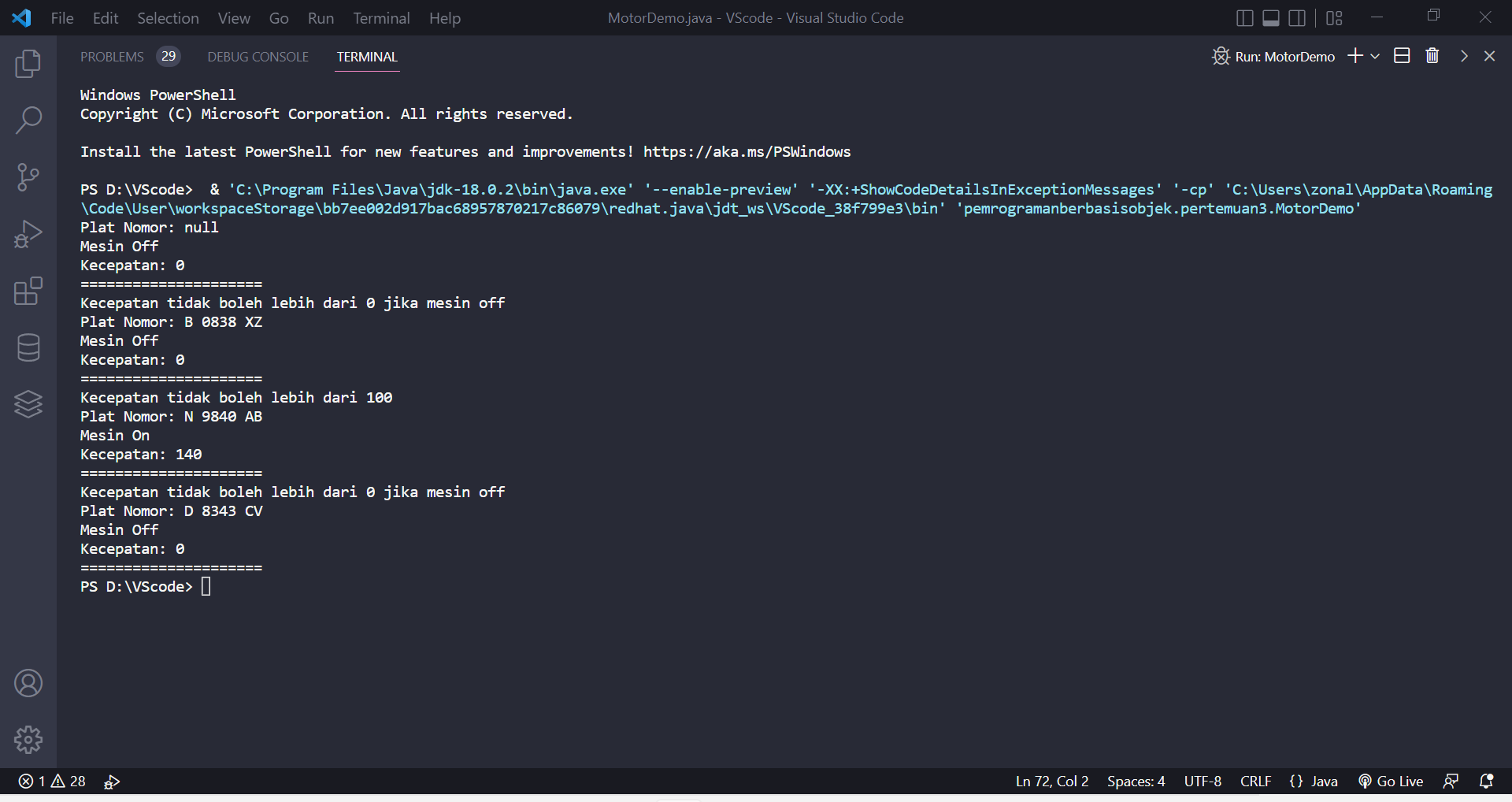
        motor2.setKecepatan(140);

        motor2.displayStatus();

Hasil running



Hasil running dengan kecepatan lebih dari 100



1. Ubah class Motor sehingga kecepatan nya tidak boleh nilai negative!

Jawab :

Code pada class Motor

    public *void* setKecepatan(*int* *kecepatan*){

        //5.  Ubah class Motor sehingga kecepatan nya tidak boleh nilai negative!

        if(*kecepatan* < 0){

            System.out.println("Kecepatan tidak boleh bernilai negatif");

            return;

        }

        if(!*this*.isMesinOn && *kecepatan* > 0){

            System.out.println("Kecepatan tidak boleh lebih dari 0 jika mesin off");

        }

        else {

            if(*kecepatan* > 100){

                System.out.println("Kecepatan tidak boleh lebih dari 100");

            }

*this*.kecepatan = *kecepatan*;

        }

    }

Code pada class MotorDemo

 Motor motor3 = **new** Motor();

        motor3.setPlatNomor("D 8343 CV");

        motor3.setKecepatan(-1);

        motor3.setIsMesinOn(true);

        motor3.displayStatus();

Hasil running



# 3.4 Percobaan 3 – Constructor

Pada pelajaran sebelumnya, instansiasi objek dari suatu class dilakukan dengan menggunakan

syntax new <NamaClass>(); misalnya motor1 = new Motor();

Dengan baris kode tersebut, kita telah menggunakan constructor default yaitu Motor() tanpa

parameter apapun. Oleh karena itu, setiap nilai atribut pada motor1 akan bernilai default.

Atribut merek yang bertipe string bernilai default null, atribut isMesinOn yang bertipe boolean

bernilai default false, dan atribut kecepatan yang bertipe integer bernilai default 0.

Pada beberapa kasus, kita menginginkan suatu objek dari class tertentu sudah memiliki nilai

untuk beberapa (atau seluruh) atribut pada saat objek tersebut dibuat.

1. Misalkan di sebuah sistem informasi, terdapat class User yang memiliki atribut username,

nama, email, alamat, dan pekerjaan. Saat suatu objek user dibuat, user tersebut harus sudah

memiliki nilai username, nama, dan email. Dengan kebutuhan tersebut, kita harus membuat

sebuah constructor baru sebagai berikut:

package pemrogramanberbasisobjek.pertemuan3;

public class User {

    public String username;

    public String nama;

    public String email;

    public String alamat;

    public String pekerjaan;

    public User(String *username*, String *nama*, String *email*) {

*this*.username = *username*;

*this*.nama = *nama*;

*this*.email = *email*;

    }

    public *void* cetakInfo() {

        System.out.println("Username: " + username);

        System.out.println("Nama: " + nama);

        System.out.println("Alamat: " + alamat);

        System.out.println("Pekerjaan: " + pekerjaan);

        System.err.println("=========================");

    }

}

2. Setelah kita menyediakan constructor baru secara eksplisit, maka constructor default yaitu User() tidak bisa digunakan lagi kecuali kita buat juga. Multiple constructor akan dibahas pada materi overloading dan overriding.

package pemrogramanberbasisobjek.pertemuan3;

public class DemoUser {

    public static *void* main(String[] *args*) {

        User user1 = **new** User();

        user1.cetakInfo();

    }

}

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

package pemrogramanberbasisobjek.pertemuan3;

public class DemoUser {

    public static *void* main(String[] *args*) {

        User user1 = **new** User("annisa.nadya", "Annisa Nadya", "annisa.nadya@gmail.com");

        user1.cetakInfo();

    }

}

4. Hasilnya sebagai berikut:



**3.5 Pertanyaan**

1. Apa yand dimaksud constructor?

Jawab : Constructor adalah method khusus yang digunakan untuk menginisialisasi objek. Constructor dipanggil ketika objek kelas dibuat. Constructor dapat digunakan untuk menetapkan nilai awal untuk atribut objek.

1. Sebutkan aturan dalam membuat constructor?

Jawab :

Aturan pembuatan Constructor :

* + nama constructor harus sama dengan nama kelas
  + tidak memiliki tipe pengembalian (void)
  + constructor dipanggil saat objek dibuat

1. Lakukan analisa dan buat kesimpulan apakah constructor bisa bertipe private?

Jawab : Bisa, tetapi constructor tidak bisa diinstansiasi di class lain

**4. Tugas**

1. Pada sebuah sistem informasi koperasi simpan pinjam, terdapat class Anggota yang

memiliki atribut antara lain 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, berikan atribut, method dan constructor sesuai dengan

kebutuhan. Uji dengan TestKoperasi 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. Code class TestKoperasi

package pemrogramanberbasisobjek.pertemuan3;

public class TestKoperasi {

    public static *void* main(String[] *args*) {

        Anggota anggota1 = **new** Anggota("111333444", "Donny", 5000000);

        System.out.println("Nama Anggota: " + anggota1.getNama());

        System.out.println("Limit Pinjaman: " + anggota1.getLimitPinjam());

        System.out.println("\nMeminjam uang 10.000.000...");

        anggota1.pinjam(10000000);

        System.out.println("Jumlah pinjaman saat ini: " + anggota1.getJumlahPinjam());

        System.out.println("\nMeminjam uang 4.000.000");

        anggota1.pinjam(4000000);

        System.out.println("Jumlah pinjaman saat ini: " + anggota1.getJumlahPinjam());

        System.out.println("\nMembayar angsuran 1.000.000");

        anggota1.angsur(1000000);

        System.out.println("Jumlah pinjaman saat ini: " + anggota1.getJumlahPinjam());

        System.out.println("\nMembayar angsuran 3.000.000");

        anggota1.angsur(3000000);

        System.out.println("Jumlah pinjaman saat ini: " + anggota1.getJumlahPinjam());

    }

}

Code class Anggota

package pemrogramanberbasisobjek.pertemuan3;

public class Anggota {

    private String noKTP;

    private String nama;

    private *int* limitPinjam;

    private *int* jumlahPinjam;

    public Anggota(String *noKTP*, String *nama*, *int* *limitPinjam*){

*this*.noKTP = *noKTP*;

*this*.nama = *nama*;

*this*.limitPinjam = *limitPinjam*;

    }

    public String getNoKTP() {

        return *this*.noKTP;

    }

    public *void* setNoKTP(String *noKTP*) {

*this*.noKTP = *noKTP*;

    }

    public String getNama() {

        return *this*.nama;

    }

    public *void* setNama(String *nama*) {

*this*.nama = *nama*;

    }

    public *int* getLimitPinjam() {

        return limitPinjam;

    }

    public *void* setLimitPinjam(*int* *limitPinjam*) {

*this*.limitPinjam = *limitPinjam*;

    }

    public *int* getJumlahPinjam() {

        return *this*.jumlahPinjam;

    }

    public *void* pinjam(*int* *tambahanPinjaman*){

        if(*tambahanPinjaman* <= limitPinjam){

            jumlahPinjam = jumlahPinjam + *tambahanPinjaman*;

        }

       else {

        System.out.println("Maaf jumlah pinjaman melebihi limit");

       }

    }

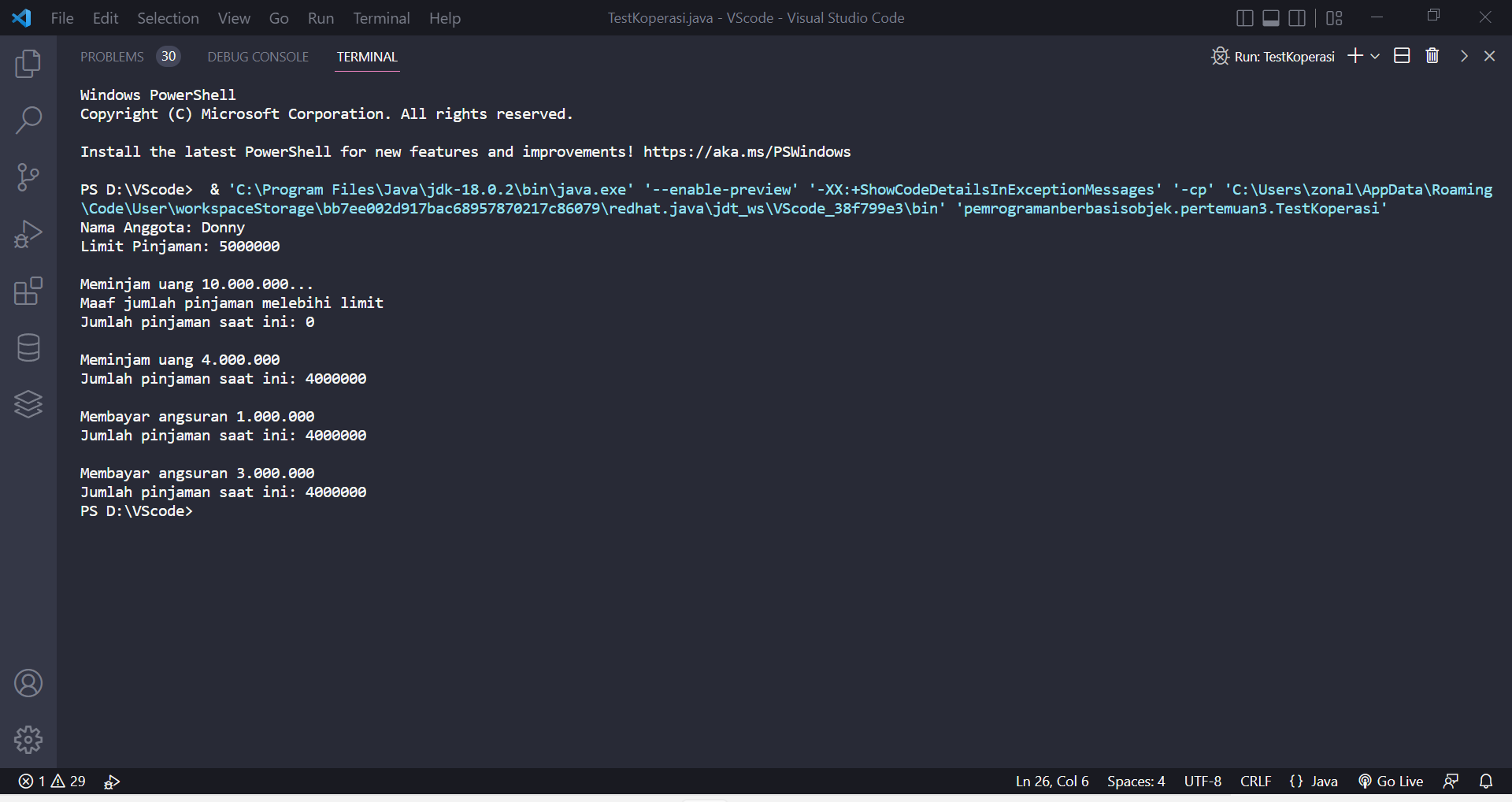
    public *int* angsur(*int* *angsuran*){

        return jumlahPinjam;

    }

}

Hasil running



2. Modifikasi class Anggota agar nominal yang dapat diangsur minimal adalah 10% dari jumlah pinjaman saat ini. Jika mengangsur kurang dari itu, maka muncul peringatan “Maaf, angsuran harus 10% dari jumlah pinjaman”.

Jawab :

Menambahkan code tersebut pada class Anggota bagian method public int angsur(int angsur)

 public *int* angsur(*int* *angsuran*){

        if(*angsuran*>=(jumlahPinjam\*0.1)){

            jumlahPinjam -= *angsuran*;

        }else{

            System.out.println("Maaf angsuran harus 10% dari jumlah pinjaman");

        }

        return jumlahPinjam;

    }

Mengubah nilai dari parameter angsuran dari 1.000.000 menjadi 10.000

 System.out.println("\nMembayar angsuran 1.000.000");

        anggota1.angsur(10000);

        System.out.println("Jumlah pinjaman saat ini: " + anggota1.getJumlahPinjam());

Hasil running

