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

A. Percobaan 1

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

```
public class User {

    public String username;
    public String email;
    public String password;
    public String name;

    public void displayInfo() {
        System.out.println("Username    : " + username);
        System.out.println("Email       : " + email);
        System.out.println("Password    : " + password);
        System.out.println("Name        : " + name);
    }
```

3) 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.

```
public class UserDemo {

    Run | Debug
    public static void main(String[] args) {
        User user1 = new User();
        user1.displayInfo();
    }
```

4) Run UserDemo.java maka output yang tampil adalah sebagai berikut.

```
Username    : null
Email       : null
Password    : null
Name        : null
```

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).

5) 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 :

```
public User(String username, String email) {
    this.username = username;
    this.email = email;
    this.password = "polinema123";
}
```

6) 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).

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

```
Run | Debug
public static void main(String[] args) {
    User user1 = new User(username:"annisa.nadya", email:"annisa.nadya@gmail.com");
    user1.displayInfo();
}
```

8) Output program sebagai berikut:

```
Username    : annisa.nadya
Email       : annisa.nadya@gmail.com
Password    : polinema123
Name        : null
```

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.

2) 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.

3) 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.

B. Percobaan 2

1) Buat class Motor.

```
public class Motor {

    public  String platNomor;
    public boolean statusMesin;
    public int kecepatan;

    public void displayInfo() {
        System.out.println("Plat Nomor      : " + this.platNomor);
        System.out.println("Status Mesin    : " + (this.statusMesin ? "on" : "off"));
        System.out.println("Kecepatan       : " + this.kecepatan);
        System.out.println(x:"=========================================");
    }
```

2) Kemudian buatlah class MotorDemo.

```
public class MotorDemo {
    Run | Debug
    public static void main(String[] args) {
        Motor motor1 = new  Motor();
        motor1.platNomor = "B 0838 XZ";
        motor1.kecepatan = 50;
        motor1.displayInfo();
```

3) Run MotorDemo.java maka outputnya adalah sebagai berikut :

```
Plat Nomor      : B 0838 XZ
Status Mesin    : off
Kecepatan       : 0
=========================================
```

4) Selanjutnya instansiasi 2 objek motor lagi di class MotorDemo.java

```
Motor motor2 = new  Motor();
motor2.platNomor = "N 9840 AB";
motor2.statusMesin = true;
motor2.kecepatan = 40;
motor2.displayInfo();

Motor motor3 = new  Motor();
motor3.platNomor = "D 8343 CV";
motor3.kecepatan = 60;
motor3.displayInfo();
```

5) Hasilnya sebagai berikut :

```
Plat Nomor      : B 0838 XZ
Status Mesin    : off
Kecepatan       : 50
=============================================
Plat Nomor      : N 9840 AB
Status Mesin    : on
Kecepatan       : 40
=============================================
Plat Nomor      : D 8343 CV
Status Mesin    : off
Kecepatan       : 60
=============================================
```

6) 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".

7) Untuk mengatasi hal tersebut, nilai kecepatan baru perlu dicek terlebih dahulu sebelum di assign ke atribut kecepatan.

```
public class MotorDemo {
    Run | Debug
    public static void main(String[] args) {
        Motor motor1 = new  Motor();
        motor1.platNomor = "B 0838 XZ";
        int kecepatanBaru = 50;

        if (!motor1.statusMesin && kecepatanBaru > 0) {
            System.out.println(x:"Kecepatan tidak boleh lebih dari 0 jika mesin off");
        } else {
            motor1.kecepatan = kecepatanBaru;
        }

        motor1.displayInfo();
```

8) Lakukan pengecekan yang sama untuk motor2 dan motor3.

```
Motor motor2 = new  Motor();
motor2.platNomor = "N 9840 AB";
motor2.statusMesin = true;
kecepatanBaru = 40;
motor2.displayInfo();

if (!motor2.statusMesin && kecepatanBaru > 0) {
    System.out.println(x:"Kecepatan tidak boleh lebih dari 0 jika mesin off");
} else {
    motor2.kecepatan = kecepatanBaru;
}

Motor motor3 = new  Motor();
motor3.platNomor = "D 8343 CV";
kecepatanBaru = 60;

if (!motor3.statusMesin && kecepatanBaru > 0) {
    System.out.println(x:"Kecepatan tidak boleh lebih dari 0 jika mesin off");
} else {
    motor3.kecepatan = kecepatanBaru;
}

motor3.displayInfo();
```

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

```
Kecepatan tidak boleh lebih dari 0 jika mesin off
Plat Nomor      : B 0838 XZ
Status Mesin    : off
Kecepatan       : 0
=========================================
Plat Nomor      : N 9840 AB
Status Mesin    : on
Kecepatan       : 0
=========================================
Kecepatan tidak boleh lebih dari 0 jika mesin off
Plat Nomor      : D 8343 CV
Status Mesin    : off
Kecepatan       : 0
=========================================
```

C. Percobaan 3

1) 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.
2) 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 :

   a. Menyembunyikan atribut internal (platNomor, statusMesin, 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 atribut menjadi private.

```
private  String platNomor;
private boolean statusMesin;
private int kecepatan;
```

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

```
public class MotorDemo {
    Run | Debug
    public static void main(String[] args) {
        Motor motor1 = new  Motor();
        motor1.platNomor = "B 0838 XZ";
        int kecepatanBaru = 50;

        if (!motor1.statusMesin && kecepatanBaru > 0) {
            System.out.println(x:"Kecepatan tidak boleh lebih dari 0 jika mesin off");
        } else {
            motor1.kecepatan = kecepatanBaru;
        }

        motor1.displayInfo();
```

5) 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)

```
public String getPlatNomor() {
    return platNomor;
}

public void setPlatNomor(String platNomor) {
    this.platNomor = platNomor;
}

public boolean getStatusMesin() {
    return statusMesin;
}

public void setStatusMesin(boolean statusMesin) {
    this.statusMesin = statusMesin;
}

public int getKecepatan() {
    return kecepatan;
}

public void setKecepatan(int kecepatan) {
    this.kecepatan = kecepatan;
}
```

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

```
Motor motor1 = new Motor();
motor1.setPlatNomor(platNomor:"B 0838 XZ");
motor1.setKecepatan(kecepatan:50);
motor1.displayInfo();

Motor motor2 = new Motor();
motor2.setPlatNomor(platNomor:"N 9840 AB");
motor2.setStatusMesin(statusMesin:true);
motor2.setKecepatan(kecepatan:40);
motor2.displayInfo();

Motor motor3 = new Motor();
motor3.setPlatNomor(platNomor:"D 8343 CV");
motor3.setKecepatan(kecepatan:60);
motor3.displayInfo();
```

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.statusMesin && kecepatan > 0) {
        System.out.println(x:"Kecepatan tidak boleh lebih dari 0 jika mesin off");
    } else {
        this.kecepatan = kecepatan;
    }
}
```

8) Run MotorDemo.java. Outputnya sebagai berikut :

```
Kecepatan tidak boleh lebih dari 0 jika mesin off
Plat Nomor      : B 0838 XZ
Status Mesin    : off
Kecepatan       : 0
=========================================
Plat Nomor      : N 9840 AB
Status Mesin    : on
Kecepatan       : 40
=========================================
Kecepatan tidak boleh lebih dari 0 jika mesin off
Plat Nomor      : D 8343 CV
Status Mesin    : off
Kecepatan       : 0
=========================================
```

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, 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().
2) Mengapa atribut merek, kecepatan, dan statusMesin sebaiknya diset private?
   Agar nilai atribute tidak diubah secara sembarangan, namun hanya bisa melalui setter dan getter
3) Apa fungsi dari setter dan getter?
   Setter berfungsi untuk mengisi nilai atribute (write), sedangkan Getter berfungsi untuk mengambil atau membaca nilai atribute (read).
4) Modifikasi class Motor sehingga kecepatan maksimalnya adalah 100.

```
public void setKecepatan(int kecepatan) {
    if (!this.statusMesin && kecepatan > 0) {
        System.out.println(x:"Kecepatan tidak boleh lebih dari 0 jika mesin off");
    } else if (kecepatan > 100) {
        System.out.println(x:"Kecepatan maksimal adalah 100");
        this.kecepatan = 100;
```

5) Modifikasi class Motor sehingga kecepatan nya tidak boleh bernilai negatif

```
public void setKecepatan(int kecepatan) {
    if (!this.statusMesin && kecepatan > 0) {
        System.out.println(x:"Kecepatan tidak boleh lebih dari 0 jika mesin off");
    } else if (kecepatan > 100) {
        System.out.println(x:"Kecepatan maksimal adalah 100");
        this.kecepatan = 100;
    } else if (kecepatan < 0) {
        System.out.println(x:"Kecepatan tidak boleh bernilai negatif");
        this.kecepatan = 0;
    } else {
        this.kecepatan = kecepatan;
    }
}
```

D. Tugas

1) 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().

2) 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”

E. Hasil Tugas (Program dan Output)

1) Program Pada class Anggota sebelum terdapat minimal angsur

```
public class Anggota {
    private String noKtp;
    private String nama;
    private int limitPinjaman;
    private int jumlahPinjaman;

    public Anggota(String noKtp, String nama, int limitPinjaman) {
        this.noKtp = noKtp;
        this.nama = nama;
        this.limitPinjaman = limitPinjaman;
    }

    public String getNoKtp() {
        return noKtp;
    }

    public String getNama() {
        return nama;
    }

    public int getLimitPinjaman() {
        return limitPinjaman;
    }

    public int getJumlahPinjaman() {
        return jumlahPinjaman;
    }

    public int pinjam(int nominalPinjam) {
        if (nominalPinjam > limitPinjaman) {
            System.out.println(x:"Maaf, jumlah pinjaman melebihi limit.");
        } else {
            jumlahPinjaman = jumlahPinjaman + nominalPinjam;
        }
        return jumlahPinjaman;
    }

    public int angsur(int nominalAngsur) {

        if (nominalAngsur > limitPinjaman) {
            System.out.println(x:"Maaf, uang kamu kelebihan");
        } else {
            jumlahPinjaman = jumlahPinjaman - nominalAngsur;
        }
        return jumlahPinjaman;
    }
```

2) Program Pada class TestKoperasi

```
public class TestKoperasi {
    Run | Debug
    public static void main(String[] args) {
        Anggota anggota1 = new Anggota(noKtp:"111333444", nama:"Donny", limitPinjaman:5000000);

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

        System.out.println(x:"\nMeminjam uang 10.000.000");
        anggota1.pinjam(nominalPinjam:10000000);
        System.out.println("Jumlah pinjaman saat ini : " + anggota1.getJumlahPinjaman());

        System.out.println(x:"\nMeminjam uang 4.000.000");
        anggota1.pinjam(nominalPinjam:4000000);
        System.out.println("Jumlah pinjaman saat ini : " + anggota1.getJumlahPinjaman());

        System.out.println(x:"\nMembayar angsuran 1.000.000");
        anggota1.angsur(nominalAngsur:100000);
        System.out.println("Jumlah pinjaman saat ini : " + anggota1.getJumlahPinjaman());

        System.out.println(x:"\nMembayar angsuran 3.000.000");
        anggota1.angsur(nominalAngsur:3000000);
        System.out.println("Jumlah pinjaman saat ini : " + anggota1.getJumlahPinjaman());

    }
}
```

3) Output

```
Nama Anggota    : Donny
Limit Pinjaman  : 5000000

Meminjam uang 10.000.000
Maaf, jumlah pinjaman melebihi limit.
Jumlah pinjaman saat ini : 0

Meminjam uang 4.000.000
Jumlah pinjaman saat ini : 4000000

Membayar angsuran 1.000.000
Jumlah pinjaman saat ini : 3000000

Membayar angsuran 3.000.000
Jumlah pinjaman saat ini : 0
```

4) Program pada class Anggota setelah terdapat minimal angsur

```java
public class Anggota {
    private String noKtp;
    private String nama;
    private int limitPinjaman;
    private int jumlahPinjaman;

    public Anggota(String noKtp, String nama, int limitPinjaman) {
        this.noKtp = noKtp;
        this.nama = nama;
        this.limitPinjaman = limitPinjaman;
    }

    public String getNoKtp() {
        return noKtp;
    }

    public String getNama() {
        return nama;
    }

    public int getLimitPinjaman() {
        return limitPinjaman;
    }

    public int getJumlahPinjaman() {
        return jumlahPinjaman;
    }

    public int pinjam(int nominalPinjam) {
        if (nominalPinjam > limitPinjaman) {
            System.out.println(x:"Maaf, jumlah pinjaman melebihi limit.");
        } else {
            jumlahPinjaman = jumlahPinjaman + nominalPinjam;
        }
        return jumlahPinjaman;
    }

    public int angsur(int nominalAngsur) {
        double minimalAngsur = 0.1 * jumlahPinjaman;

        if (nominalAngsur > limitPinjaman) {
            System.out.println(x:"Maaf, uang kamu kelebihan");
        } else if (nominalAngsur < minimalAngsur) {
            System.out.println(x:"Maaf, angsuran harus 10% dari jumlah pinjaman");
        } else {
            jumlahPinjaman = jumlahPinjaman - nominalAngsur;
        }
        return jumlahPinjaman;
    }
```

5) Output dengan minimal angsur

```
Nama Anggota    : Donny
Limit Pinjaman  : 5000000

Meminjam uang 10.000.000
Maaf, jumlah pinjaman melebihi limit.
Jumlah pinjaman saat ini : 0

Meminjam uang 4.000.000
Jumlah pinjaman saat ini : 4000000

Membayar angsuran 100.000
Maaf, angsuran harus 10% dari jumlah pinjaman
Jumlah pinjaman saat ini : 4000000

Membayar angsuran 3.000.000
Jumlah pinjaman saat ini : 1000000
```