Review & Latihan – Konsep Objects & Classes

Perhatikan deklarasi sebuah kelas berikut:

```
public class Counter {

    private int value;

    public Counter() {
        this.value = 0;
    }

    public Counter(int value) {
        this.value = value;
    }

    public void count() {
        this.value = this.value + 1;
    }

    public int getValue() {
        return this.value;
    }

    public void setValue(int value) {
        this.value = value;
    }
}
```

a) Silakan Anda beri tanda (dengan lingkaran atau sejenisnya) bagian-bagian berikut: deklarasi **Instance Variables**, **Contructors**, dan **Methods**.
b) Method apa saja yang termasuk **Setter/Getter**?
c) Mengapa instance variables wajib bersifat ***private***? Kaitkan dengan salah satu prinsip OOP yaitu ***Encapsulation***!
d) Yang mana yang merupakan method **mutator**? Dan yang mana yang merupakan **accessor**?
e) Bagaimana caranya menghidupkan sebuah object dari kelas Counter dengan
    a. nilai awal 0?
    b. nilai awal 10?
f) Jika <u>Constructor **tidak** dideklarasikan secara eksplisit</u>, apakah akan menyebabkan compile error?
g) Apakah output dari potongan program berikut? Gambarkan yang terjadi di memori pada kolom sebelah kanan!

`Counter counter1 = new Counter();`
`counter1.count();`
`counter1.count();`
`Counter counter2 = counter1;`
`counter2.count();`
`System.out.println(counter1.getValue());`
`System.out.println(counter2.getValue());`
`counter2 = new Counter(5);`
`System.out.println(counter1.getValue());`
`System.out.println(counter2.getValue());`	

`int number1 = 10;`
`int number2 = number1;`
`number1 = 5;`	

```
System.out.println(number1);
System.out.println(number2);
```

h) Misal, kita tambah sebuah method baru yang bernama `reset()` di kelas Counter. Perhatikan ada berapa parameter di method `setValue()` yang dipanggil (call)?

```
public void reset() {
    setValue(0);
}
```

i) Apakah bermasalah jika method `count()` dideklarasikan ulang seperti hal berikut (tanpa **`this`**):

```
public void count() {
    value = value + 1;
}
```

j) Apakah perbedaan antara **Local & Parameter variables** dengan **Instance Variables**?

k) Buatlah sebuah kelas tester untuk menguji kelas **Counter** tersebut!

B. Kelas BankAccount

Implementasikan kelas **BankAccount** yang merepresentasikan sebuah rekening bank seseorang. Seseorang bisa menabung sejumlah uang, dan juga bisa mengambil sejumlah uang.

Instance Variables

- **`balance`** yang bertipe **`double`**. **`balance`** merepresentasikan jumlah uang saat ini di bank.

Constructor

- Konstruktor Pertama: tanpa parameter yang melakukan inisialisasi **`balance`** dengan **`0`**.
- Konstruktor Kedua: dengan sebuah parameter yang melakukan inisialisasi terhadap **`balance`**.

Method setter/getter

- **`int getBalance()`** : yang mengembalikan nilai uang saat ini di bank.

Method yang lain

- **`void deposit(double amount)`** : menyimpan uang ke bank sejumlah **`amount`**.
- **`void withdraw(double amount)`** : mengambil uang dari bank sejumlah **`amount`**.

C. Kelas Time

Implementasikan sebuah kelas **`Time`** yang merepresentasikan waktu dalam **hh:mm:ss**. Jam bernilai diantara 0 – 23, menit diantara 0 – 59, dan detik diantara 0 – 59.

Instance Variables

- **`jam`** yang bertipe **`int (0 - 23)`**.
- **`menit`** yang bertipe **`int (0 - 59)`**.
- **`detik`** yang bertipe **`int (0 - 59)`**.

Constructor

- Konstruktor Pertama: tanpa parameter yang melakukan inisialisasi **`jam`**, **`tinggi`**, dan **`detik`** masing-masing dengan **0**.
- Konstruktor Kedua: dengan 3 buah parameter yang melakukan inisialisasi **`jam`**, **`tinggi`**, dan **`detik`**. Masukan pasti valid.

Method setter/getter

- **`int getJam()`**
- **`int getMenit()`**
- **`int getDetik()`**
- **`void setJam(int jam)`**. Nilai jam yang dimasukkan pasti valid 0 – 23.
- **`void setMenit(int menit)`** . Nilai menit yang dimasukkan pasti valid 0 – 59.
- **`void setDetik(int detik)`** . Nilai detik yang dimasukkan pasti valid 0 – 59.

Method yang lain

- **`void nextDetik()`** : set kondisi waktu saat ini ke satu detik ke depan
- **`void prevDetik()`** : set kondisi waktu saat ini ke satu detik ke belakang
- **`void nextMenit()`** : set kondisi waktu saat ini ke satu menit ke depan
- **`void prevMenit()`** : set kondisi waktu saat ini ke satu menit ke belakang
- **`String toString()`** : mengembalikan sebuah String yang merepresentasikan waktu saat ini dengan format **"jam:menit:detik"**.