

Pemrograman Desktop - Tugas 2

Nama : Tito Pandu Brahmanto

NIM : 042065142

Soal No 1

```
class Encapsulation {
    private String nama;
    private ... nim;
    public String getNama() {
        return this.nama;
    }
    public void ModifNama(String nama) {
        this.nama = nama;
    }

    public int ...() {
        return this.nim;
    }
    public void ModifNim(... nim) {
        this.nim = ...;
    }
}

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

        Encapsulation objek = new Encapsulation();
        .....(1420123);
        objek.ModifNama("Mahasiswa");
        System.out.println("Nim : " + objek....());
        System.out.println("... : " + objek.getNama());
    }
}
```

Lengkapi kode program di atas sehingga menghasilkan output:

Nim : 1420123

Nama : Mahasiswa

Jawab

```
class Encapsulation {
    private String nama;
    private int nim;
    public String getNama() {
        return this.nama;
    }
    public void ModifNama(String nama) {
        this.nama = nama;
    }

    public int getNim() {
        return this.nim;
    }
}
```

```

    public void ModifNim(int nim) {
        this.nim = nim;
    }
}

public class main {
    public static void main(String[] args) {
        Encapsulation objek = new Encapsulation();
        objek.ModifNim(1420123);
        objek.ModifNama("Mahasiswa");
        System.out.println("Nim : " + objek.getNim());
        System.out.println("Nama : " + objek.getNama());
    }
}

```

The screenshot shows an IDE with a Java file named 'main.java' and a task named 'tugas-3.rmd'. The code defines an 'Encapsulation' class with private attributes 'nama' and 'nim', and methods 'getNama()', 'ModifNama()', 'getNim()', and 'ModifNim()'. A 'main' class contains a 'main' method that creates an 'Encapsulation' object, modifies its 'nim' and 'nama' attributes, and prints them out. The output console shows the execution of the 'main' method, displaying 'Nim : 1420123' and 'Nama : Mahasiswa'. The build is successful, taking 266ms and executing 2 tasks.

```

1 class Encapsulation {
2     private String nama;
3     private int nim;
4     public String getNama() {
5         return this.nama;
6     }
7     public void ModifNama(String nama) {
8         this.nama = nama;
9     }
10
11     public int getNim() {
12         return this.nim;
13     }
14     public void ModifNim(int nim) {
15         this.nim = nim;
16     }
17 }
18
19 public class main {
20     public static void main(String[] args) {
21         Encapsulation objek = new Encapsulation();
22         objek.ModifNim(1420123);
23         objek.ModifNama("Mahasiswa");
24         System.out.println("Nim : " + objek.getNim());
25         System.out.println("Nama : " + objek.getNama());
26     }
27 }
28

```

Run: desktop_programming [main.main()]

desktop 365 ms 28.41.23: Executing 'main.main()'

> Task :compileJava
> Task :processResources NO-SOURCE
> Task :classes

> Task :main.main()
Nim : 1420123
Nama : Mahasiswa

Deprecated Gradle features were used in this build, making it incompatible with Gradle 8.0.

You can use '--warning-mode all' to show the individual deprecation warnings and determine if they come from your own code or generated code.

See https://docs.gradle.org/7.4/userguide/command_line_interface.html#sec:command_line_warnings for more details.

BUILD SUCCESSFUL in 266ms
2 actionable tasks: 2 executed
28.41.23: Execution finished 'main.main()'.

Figure 1: soal 1

Soal No 2

Perhatikan contoh kode program di bawah ini:

```
public class overloading{
    public void segitiga(){
        int alas=5, tinggi=10;
        System.out.println("Luas segitiga satu = "+(alas*tinggi)/2);
    }

    public void segitiga2(int x, int y){
        System.out.println("Luas segitiga dua = "+(x*y)/2);
    }

    public static void main(String [] args){
        System.out.println("Contoh Overloading");
        System.out.println("");
        overloading s3;
        s3 = new overloading();
        s3.segitiga();
        s3.segitiga2(20,8);
    }
}
```

Kode program di atas adalah Polymorphism. Jawablah pertanyaan berikut ini: a. Sebutkan dan jelaskan jenis dari Polymorphism dari program di atas b. Jelaskan baris kode pada no 1,2, dan 3.

Jawab

- a. Jenis dari polymorphism ini adalah Static Polymorphism atau trivial atau disebut juga function overloading (penggunaan kembali nama fungsi yang sama tapi dengan argumen yang berbeda). Seharusnya pada baris kode no 1 menggunakan nama fungsi yang sama, yaitu segitiga. Sehingga nama fungsi yang sama dapat digunakan dengan argumen yang berbeda.
- b. Berikut ini penjelasan beberapa baris kode
 1. Baris kode ini adalah pembuatan method overload. Seharusnya menggunakan nama `segitiga` sehingga dapat meng-overload method `segitiga` sebelumnya yang tanpa argumen. Baris kode ini menghitung perkalian argumen `x` dan `y`, lalu dibagi 2. Sesuai dengan rumus luas segitiga yaitu $\text{alas} * \text{tinggi} / 2$
 2. Baris kode ini adalah pemanggilan method `segitiga` yang tidak memiliki argumen. Alas dan tinggi sudah di-assign di dalam method ini dengan `alas = 5` dan `tinggi = 10`
 3. Baris kode ini adalah pemanggilan method `segitiga2` (seharusnya `segitiga`) yang memiliki argumen `x` dan `y`.

```
public class overloading{
    public void segitiga(){
        int alas=5, tinggi=10;
        System.out.println("Luas segitiga satu = "+(alas*tinggi)/2);
    }

    public void segitiga(int x, int y){
        System.out.println("Luas segitiga dua = "+(x*y)/2);
    }

    public static void main(String [] args){
        System.out.println("Contoh Overloading");
    }
}
```

```

        System.out.println("");
        overloading s3;
        s3 = new overloading();
        s3.segitiga();
        s3.segitiga(20,8);
    }
}

```

The screenshot shows an IDE with a Java file named `overloading.java`. The code defines a class `overloading` with two methods: `segitiga()` (no arguments) and `segitiga(int x, int y)` (two arguments). The `main` method creates an instance of `overloading`, calls `segitiga()`, and then `segitiga(20, 8)`. The output window shows the results of the execution: "Luas segitiga satu = 25" and "Luas segitiga dua = 80". The build is successful.

```

1 package org.example;
2
3 public class overloading{
4     public void segitiga(){
5         int alas=5, tinggi=10;
6         System.out.println("Luas segitiga satu = "+(alas*tinggi)/2);
7     }
8
9     public void segitiga(int x, int y){
10        System.out.println("Luas segitiga dua = "+(x*y)/2);
11    }
12
13    public static void main(String [] args){
14        System.out.println("Contoh Overloading");
15        System.out.println("");
16        overloading s3;
17        s3 = new overloading();
18        s3.segitiga();
19        s3.segitiga(20, 8);
20    }
21 }

```

```

Run: desktop_programming [:overloading.main()]
desktop 305 ms 20.53.42: Executing ':overloading...'
> Task :compileJava
> Task :processResources NO-SOURCE
> Task :classes

> Task :overloading.main()
Contoh Overloading

Luas segitiga satu = 25
Luas segitiga dua = 80

Deprecated Gradle features were used in this build, making it incompatible with Gradle 8.0.
You can use '--warning-mode all' to show the individual deprecation warnings and determine if they appear to be harmless.
See https://docs.gradle.org/7.4/userguide/command_line_interface.html#sec:command_line_warnings

BUILD SUCCESSFUL in 199ms
2 actionable tasks: 2 executed
20.53.42: Execution finished 'overloading.main()'

```

Figure 2: soal 2

Sumber: BMP Pemrograman Berbasis Desktop MSIM4301 - Modul 8