

LAPORAN PRAKTIKUM MINGGU 1

PENGENALAN PARADIGMA (PROCEDURAL, OOP, dan FUNCTIONAL)

NAMA : ALVIRA LIBRA RAMADHANI

NIM : 240202851

KELAS : 3IKRA

1. TUJUAN

1. Mengetahui tiga paradigma pemrograman: Prosedural, OOP, dan Fungsional.
2. Menerapkan masing-masing paradigma dalam program sederhana menggunakan Java.
3. Menjalankan program menggunakan terminal (javac dan java).
4. Membandingkan perbedaan dari setiap paradigma.

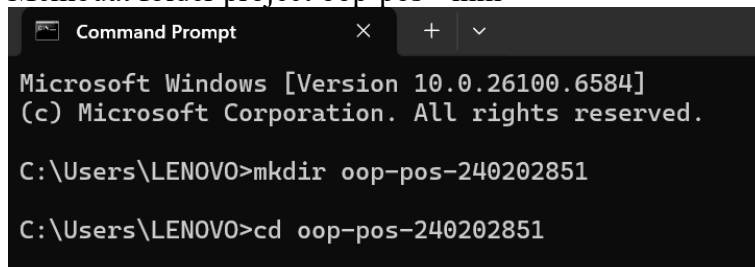
2. DASAR TEORI

1. Prosedural: paradigma yang menekankan eksekusi langkah demi langkah dalam bentuk prosedur/fungsi.
2. OOP (Object-Oriented Programming): paradigma berbasis objek dengan konsep class, object, method, constructor, encapsulation.
3. Fungsional: paradigma yang berorientasi pada fungsi, dengan fitur lambda expression dan functional interface.
4. Java mendukung ketiga paradigma ini sekaligus karena merupakan bahasa multiparadigma.

3. LANGKAH PRAKTIKUM

A. Setup Project

1. Menyiapkan dan menginstall **Java JDK** di laptop
2. Membuat folder project oop-pos-<nim>

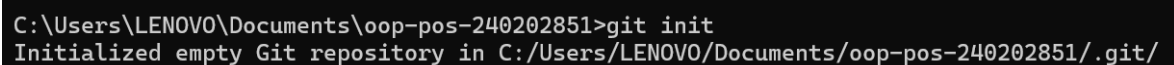


```
Microsoft Windows [Version 10.0.26100.6584]
(c) Microsoft Corporation. All rights reserved.

C:\Users\LENOVO>mkdir oop-pos-240202851

C:\Users\LENOVO>cd oop-pos-240202851
```

3. Inisialisasi repositori Git



```
C:\Users\LENOVO\Documents\oop-pos-240202851>git init
Initialized empty Git repository in C:/Users/LENOVO/Documents/oop-pos-240202851/.git/
```

4. Buat struktur awal src/main/java/com/upb/agripos/

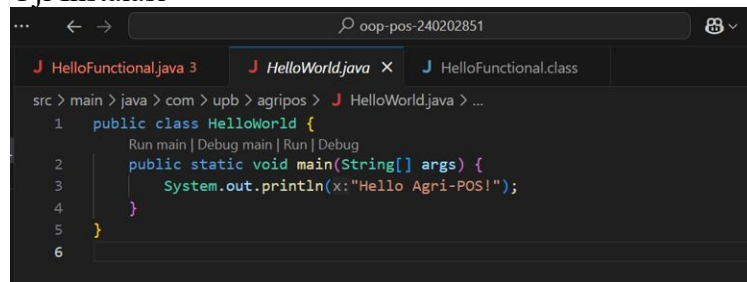
```

C:\Users\LENOVO\oop-pos-240202851>mkdir src
C:\Users\LENOVO\oop-pos-240202851>mkdir src\main
C:\Users\LENOVO\oop-pos-240202851>mkdir src\main\java
C:\Users\LENOVO\oop-pos-240202851>mkdir src\main\java\com
C:\Users\LENOVO\oop-pos-240202851>mkdir src\main\java\com\upb
C:\Users\LENOVO\oop-pos-240202851>mkdir src\main\java\com\upb\agripes
C:\Users\LENOVO\oop-pos-240202851>

```

5. Pastikan semua tools dapat berjalan (uji dengan membuat dan menjalankan program Java sederhana).

- Uji Instalasi



```

src > main > java > com > upb > agripes > J HelloWorld.java > ...
1 public class HelloWorld {
2     public static void main(String[] args) {
3         System.out.println("Hello Agri-POS!");
4     }
5 }
6

```

- Jalankan dan muncul output

```

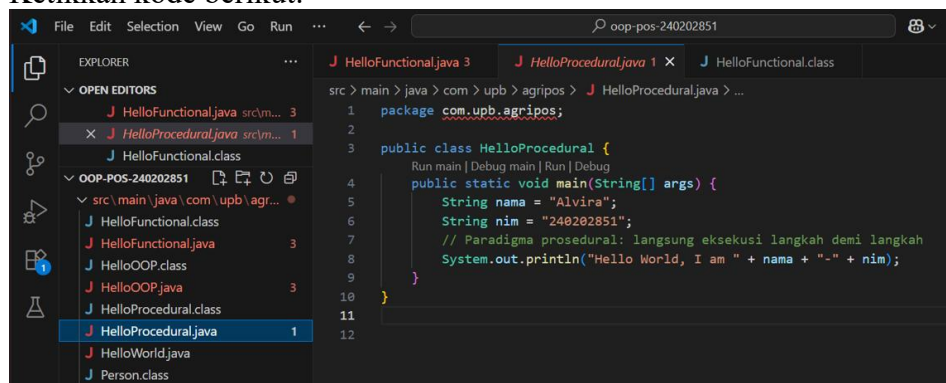
PS C:\Users\LENOVO\Documents\oop-pos-240202851\src\main\java> javac com\upb\agripes\HelloWorld.java
PS C:\Users\LENOVO\Documents\oop-pos-240202851\src\main\java> java com.upb.agripes.HelloWorld
Hello Agri-POS!
PS C:\Users\LENOVO\Documents\oop-pos-240202851\src\main\java>

```

- B. Membuat program 'Hello World, I am <nama> - <nim>' dalam 3 paradigma [nama = Alvira, nim = 240202851]

1. Paradigma Prosedural

- a. Buat file baru dengan nama HelloProcedural.java di dalam com/upb/agripes
- b. Ketikkan kode berikut:



```

src > main > java > com > upb > agripes > J HelloProcedural.java > ...
1 package com.upb.agripes;
2
3 public class HelloProcedural {
4     public static void main(String[] args) {
5         String nama = "Alvira";
6         String nim = "240202851";
7         // Paradigma prosedural: langsung eksekusi langkah demi langkah
8         System.out.println("Hello World, I am " + nama + "-" + nim);
9     }
10 }
11
12

```

- c. Lakukan compile dan jalankan program, lalu output akan muncul:

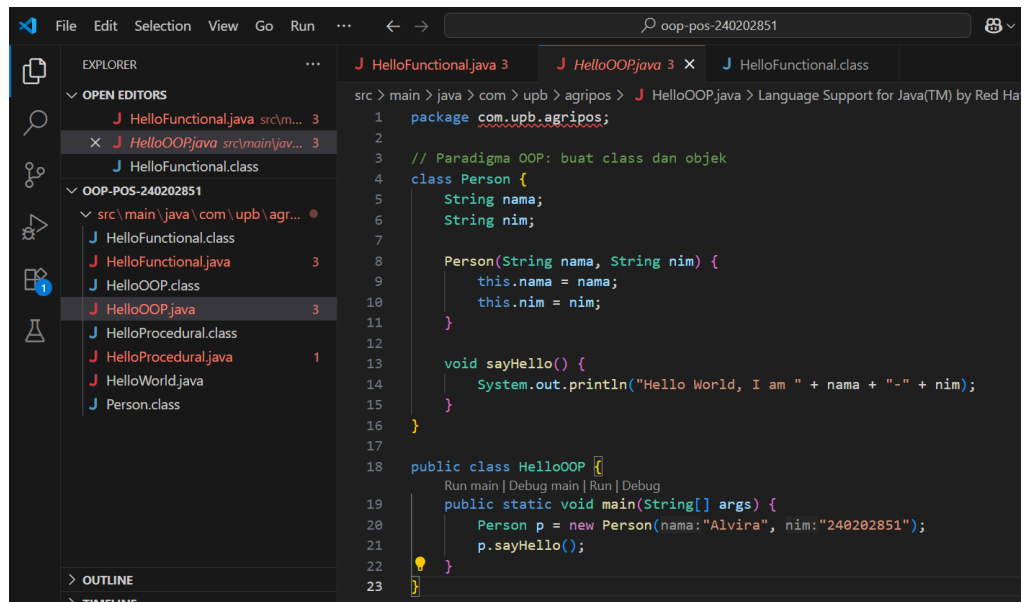
```

PS C:\Users\LENOVO\Documents\oop-pos-240202851\src\main\java> javac com\upb\agripes\HelloProcedural.java
PS C:\Users\LENOVO\Documents\oop-pos-240202851\src\main\java> java com.upb.agripes.HelloProcedural
Hello World, I am Alvira-240202851

```

2. Paradigma OOP

- a. Buat nama file baru dengan nama HelloOOP.java
- b. Ketikkan kode berikut:



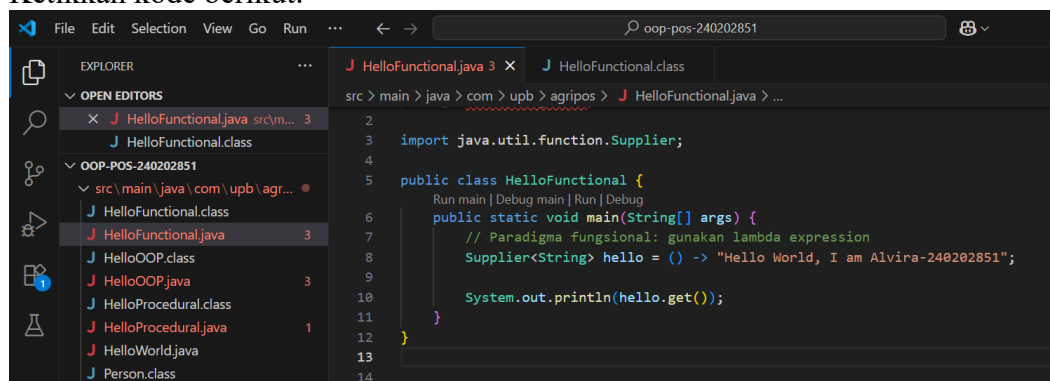
```
1 package com.upb.agripos;
2
3 // Paradigma OOP: buat class dan objek
4 class Person {
5     String nama;
6     String nim;
7
8     Person(String nama, String nim) {
9         this.nama = nama;
10        this.nim = nim;
11    }
12
13    void sayHello() {
14        System.out.println("Hello World, I am " + nama + "-" + nim);
15    }
16 }
17
18 public class HelloOOP {
19     Run main | Debug main | Run | Debug
20     public static void main(String[] args) {
21         Person p = new Person(nama:"Alvira", nim:"240202851");
22         p.sayHello();
23     }
24 }
```

c. Lakukan compile dan jalankan program, lalu output akan muncul:

```
PS C:\Users\LENOVO\Documents\oop-pos-240202851\src\main\java> javac com\upb\agripos\HelloOOP.java
PS C:\Users\LENOVO\Documents\oop-pos-240202851\src\main\java> java com.upb.agripos.HelloOOP
Hello World, I am Alvira-240202851
```

3. Paradigma Functional

- Buat nama file baru dengan nama HelloFunctional.java
- Ketikkan kode berikut:



```
1 import java.util.function.Supplier;
2
3 public class HelloFunctional {
4     Run main | Debug main | Run | Debug
5     public static void main(String[] args) {
6         // Paradigma fungsional: gunakan lambda expression
7         Supplier<String> hello = () -> "Hello World, I am Alvira-240202851";
8
9         System.out.println(hello.get());
10    }
11 }
12
13
14
```

c. Lakukan compile dan jalankan program, lalu output akan muncul:

```
PS C:\Users\LENOVO\Documents\oop-pos-240202851\src\main\java> javac com\upb\agripos\HelloFunctional.java
PS C:\Users\LENOVO\Documents\oop-pos-240202851\src\main\java> java com.upb.agripos.HelloFunctional
Hello World, I am Alvira-240202851
```

➤ Kelebihan dan Kekurangan

a. Paradigma Prosedural

Kelebihan:

- Sederhana dan mudah terutama untuk program kecil.
- Eksekusi program berjalan step by step, jadi alurnya jelas.
- Cocok untuk masalah yang sifatnya linear dan tidak terlalu kompleks.

Kekurangan:

- Sulit dikelola saat program menjadi besar
- Reuse kode lebih sedikit, harus banyak copy-paste.
- Perubahan di satu fungsi bisa berdampak luas pada bagian lain.

b. Paradigma OOP

Kelebihan:

1. Membagi program ke dalam class & object, sehingga lebih terstruktur.
2. Lebih mudah diperluas (scalable) dan cocok untuk proyek besar.
3. Lebih dekat dengan pemodelan dunia nyata (objek & atribut).

Kekurangan:

1. Membutuhkan pemahaman konsep yang lebih rumit
2. Overhead memori lebih besar karena harus membuat objek.
3. Kadang berlebihan untuk program sederhana.

c. Paradigma Functional

Kelebihan:

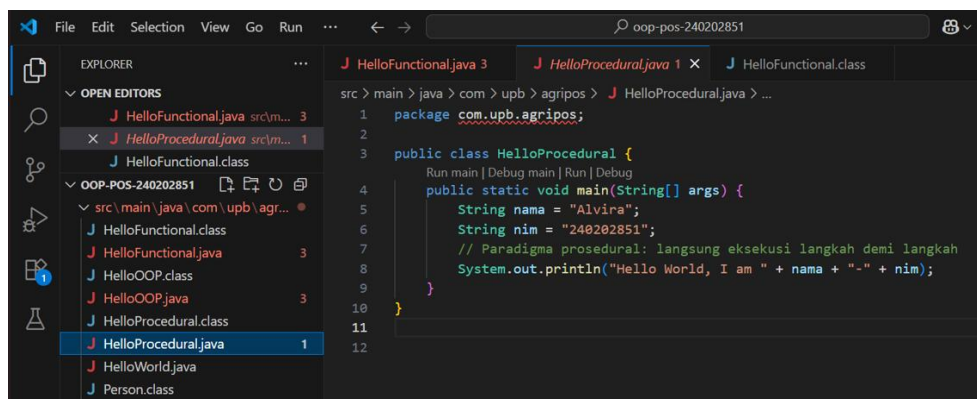
1. Mengutamakan fungsi murni → lebih mudah diuji dan diprediksi.
2. Menghindari efek samping, sehingga program lebih aman dan stabil.
3. Mendukung parallel & concurrent processing lebih mudah.

Kekurangan:

1. Kurang populer dibanding procedural dan OOP, jadi dukungan lebih sedikit.
2. Lebih sulit dipahami oleh pemula karena cara berpikirnya berbeda
3. Eksekusi bisa lebih lambat pada kasus tertentu dibanding prosedural.

4. KODE PROGRAM

1. HelloProcedural.java



```
1 package com.upb.agripis;
2
3 public class HelloProcedural {
4     Run main | Debug main | Run | Debug
5     public static void main(String[] args) {
6         String nama = "Alvira";
7         String nim = "240202851";
8         // Paradigma prosedural: langsung eksekusi langkah demi langkah
9         System.out.println("Hello World, I am " + nama + "-" + nim);
10    }
11
12 }
```

2. HelloOOP.java

```

1 package com.upb.agripis;
2
3 // Paradigma OOP: buat class dan objek
4 class Person {
5     String nama;
6     String nim;
7
8     Person(String nama, String nim) {
9         this.nama = nama;
10        this.nim = nim;
11    }
12
13    void sayHello() {
14        System.out.println("Hello World, I am " + nama + "-" + nim);
15    }
16 }
17
18 public class HelloOOP {
19     Run main | Debug main | Run | Debug
20     public static void main(String[] args) {
21         Person p = new Person(nama:"Alvira", nim:"240202851");
22         p.sayHello();
23     }
24 }

```

3. HelloFunctional.java

```

1
2
3 import java.util.function.Supplier;
4
5 public class HelloFunctional {
6     Run main | Debug main | Run | Debug
7     public static void main(String[] args) {
8         // Paradigma fungsional: gunakan lambda expression
9         Supplier<String> hello = () -> "Hello World, I am Alvira-240202851";
10
11         System.out.println(hello.get());
12     }
13 }
14

```

5. HASIL EKSEKUSI

1. Paradigma Pcedural

```

PS C:\Users\LENOVO\Documents\oop-pos-240202851\src\main\java> javac com\upb\agripis\HelloProcedural.java
PS C:\Users\LENOVO\Documents\oop-pos-240202851\src\main\java> java com.upb.agripis.HelloProcedural
Hello World, I am Alvira-240202851

```

2. Paradigma OOP

```

PS C:\Users\LENOVO\Documents\oop-pos-240202851\src\main\java> javac com\upb\agripis\HelloOOP.java
PS C:\Users\LENOVO\Documents\oop-pos-240202851\src\main\java> java com.upb.agripis.HelloOOP
Hello World, I am Alvira-240202851

```

3. Paradigma Functional

```

PS C:\Users\LENOVO\Documents\oop-pos-240202851\src\main\java> javac com\upb\agripis\HelloFunctional.java
PS C:\Users\LENOVO\Documents\oop-pos-240202851\src\main\java> java com.upb.agripis.HelloFunctional
Hello World, I am Alvira-240202851

```

➤ Analisis

1. Bagaimana kode berjalan

- Pada versi **Procedural**, eksekusi langsung dimulai dari main() yang berisi satu perintah System.out.println().

- b. Pada versi **OOP**, objek Student dibuat, kemudian method sayHello() dipanggil untuk mencetak output.
 - c. Pada versi **Functional**, digunakan Supplier dengan lambda expression untuk menghasilkan string, lalu dipanggil dengan .get().
 - d. Semua file dikompilasi dengan javac, lalu class yang sudah ter-compile dijalankan dengan java sesuai package.
2. Kendala yang dihadapi dan cara mengatasinya
- a. **Kendala 1:** File .java awalnya tidak dikenali saat kompilasi.
Solusi: Memastikan path package (com.upb.agripos) sesuai dengan struktur folder.
 - b. **Kendala 2:** Saat menjalankan java, program tidak ditemukan.
Solusi: Menjalankan perintah java dengan fully-qualified name (com.upb.agripos.HelloOOP, bukan hanya HelloOOP).
 - c. **Kendala 3:** Error di awal karena salah ketik pada perintah terminal.
Solusi: Memperbaiki command sesuai sintaks Java.

6. KESIMPULAN

1. Praktikum minggu ini memperlihatkan bahwa Java mendukung multi-paradigma: Prosedural, OOP, dan Functional.
2. Dengan OOP, kode menjadi lebih terstruktur dan reusable.
3. Dengan Functional, kode menjadi lebih ringkas.
4. Mahasiswa jadi lebih memahami perbedaan gaya penulisan program di Java.

7. CHEKLIST KEBERHASILAN

- ☒ Lingkungan kerja sudah siap (JDK, IDE, Git, PostgreSQL, JavaFX).
- ☒ Repositori Git sudah dibuat dan commit awal berhasil (week1-setup-hello-pos).
- ☒ Program "Hello POS World" berjalan di tiga paradigma dan menampilkan NIM serta nama mahasiswa.
- ☒ Versi OOP dan fungsional menggunakan minimal tiga objek/entri produk.
- ☒ Tangkapan layar hasil eksekusi ketiga program telah disertakan.
- ☒ Laporan singkat telah dilampirkan.
- ☒ Perbedaan paradigma sudah dipahami dan dijelaskan pada laporan.

8. QUIZ

1. Apakah OOP selalu lebih baik dari prosedural?
Jawaban: Tidak selalu. OOP lebih baik ketika aplikasi kompleks, butuh banyak objek yang saling berinteraksi, dan perlu pengelolaan kode dalam jangka panjang. Namun untuk program kecil, sederhana, atau sekali pakai, pendekatan prosedural justru lebih cepat, ringkas, dan mudah dipahami.
2. Kapan functional programming lebih cocok digunakan dibanding OOP atau prosedural?
Jawaban: Functional programming lebih cocok ketika aplikasi membutuhkan pemrosesan data dalam jumlah besar, bersifat paralel, atau butuh kode yang ringkas tanpa banyak state. Contohnya pada aplikasi analisis data, stream processing, atau kasus yang banyak menggunakan operasi matematis/logika.

3. Bagaimana paradigma (prosedural, OOP, fungsional) memengaruhi maintainability dan scalability aplikasi?

Jawaban:

- d. **Prosedural** → maintainability rendah karena kode cenderung menumpuk dalam satu alur; scalability terbatas.
 - e. **OOP** → maintainability tinggi karena konsep class dan objek membuat kode modular; scalability lebih baik karena mudah menambah fitur dengan class baru.
 - f. **Fungsional** → maintainability cukup baik karena kode lebih ringkas dan menghindari state yang membingungkan; scalability bagus terutama dalam sistem yang butuh eksekusi paralel.
4. Mengapa OOP lebih cocok untuk mengembangkan aplikasi POS dibanding prosedural?
- Jawaban:** Karena aplikasi POS (Point of Sale) melibatkan banyak entitas nyata seperti Produk, Pelanggan, Transaksi, Kasir, dan lain-lain. Dengan OOP, setiap entitas dapat dimodelkan sebagai class dengan atribut dan metode. Hal ini membuat aplikasi lebih mudah dikembangkan, dipelihara, dan diperluas dibanding prosedural yang hanya berupa rangkaian instruksi.
5. Bagaimana paradigma fungsional dapat membantu mengurangi kode berulang (*boilerplate code*)?

Jawaban: Functional programming menggunakan konsep fungsi sebagai first-class citizen dan lambda expression, sehingga kode bisa ditulis lebih ringkas tanpa harus membuat banyak class/method tambahan. Misalnya, operasi filter, map, dan reduce bisa langsung digunakan tanpa harus membuat loop manual berulang-ulang, sehingga boilerplate code berkurang drastis.