

**LAPORAN PRAKTIKUM**  
**ALGORITMA DAN PEMROGRAMAN**

**“laporan Praktikum Pekan 7”**

Disusun Oleh:

Faiz Fikri Satria

2511533026

Dosen Pengampu : Dr. Wahyudi, S.T, M.T.  
Asisten Praktikum : Rahmad Dwirizki Olders



DAPARTEMEN INFORMATIKA  
FAKULTAS TEKNOLOGI INFORMASI  
UNIVERSITAS ANDALAS

2025

## **KATA PENGANTAR**

Puji syukur saya panjatkan ke hadirat Tuhan Yang Maha Esa atas rahmat dan karunia-Nya, sehingga dapat menyelesaikan laporan praktikum ini dengan baik. Tidak lupa juga saya ucapkan terima kasih kepada bapak Wahyudi. Dr. S.T.M.T sebagai dosen pembimbing dan Rahmad Dwirizki Olders yang telah membantu dalam pelaksanaan praktikum ini. Laporan ini saya susun untuk memenuhi tugas praktikum Algoritma dan Pemrograman dari praktikum pertemuan ke enam mengenai pemahaman dasar pemrograman bahasa Java menggunakan IDE Eclipse. Praktikum ini bertema Penggabungan dari Class yang Berbeda. Saya berharap laporan ini dapat memberikan manfaat bagi pembaca dalam memahami konsep dasar pemrograman Java.

Laporan ini dibuat dengan harapan dapat memberikan pemahaman dasar mengenai penggunaan Bahasa pemrograman Java, khususnya mengenai hal struktur dasar program dan fungsi.

Saya menyadari bahwa laporan ini masih jauh dari sempurna, oleh sebab itu, kritik dan saran sangat diharapkan demi penyempurnaan laporan di masa mendatang.

Padang 15 November, 2025

Penulis

## DAFTAR ISI

KATA PENGANTAR .....	ii
DAFTAR ISI .....	iii
BAB I PENDAHULUAN .....	1
1.1 Latar Belakang .....	1
1.2 Tujuan .....	1
1.3 Manfaat .....	1
BAB II PEMBAHASAN .....	2
<b>2.1 Class Akun dengan Encapsulation</b> .....	2
2.1.1 Langkah-langkah .....	2
2.1.2 Contoh Pemrograman .....	3
2.1.3 Hasil Output .....	3
2.1.4 Analisis .....	3
2.1.5 Teori Mengenai Kode .....	3
<b>2.2 Class Mahasiswa dengan Getter/Setter</b> .....	4
2.2.1 Langkah-langkah .....	4
2.2.2 Contoh Pemrograman .....	5
2.2.3 Analisis .....	5
2.2.4 Teori Mengenai Kode .....	5
<b>2.3 Penggabungan dengan Class Main Statis (Kode 3)</b> .....	6
2.3.1 Langkah-langkah .....	6
2.3.2 Contoh Pemrograman .....	7

2.3.3 Hasil Output .....	7
2.3.4 Analisis .....	7
2.3.5 Teori Mengenai Kode .....	7
<b>2.4 Penggabungan dengan Input dan Kondisi (Kode 4) .....</b>	<b>7</b>
2.4.1 Langkah-langkah .....	8
2.4.2 Contoh Pemrograman .....	8
2.4.3 Analisis .....	8
2.4.4 Teori Mengenai Kode .....	8
<b>2.5 Penggabungan dengan Input Langsung (Kode 5) .....</b>	<b>9</b>
2.5.1 Langkah-langkah .....	9
2.5.2 Contoh Pemrograman .....	9
2.5.3 Hasil Output .....	9
2.5.4 Analisis .....	10
2.5.5 Teori Mengenai Kode .....	10
<b>2.6 Manipulasi String Dasar (Kode 6) .....</b>	<b>10</b>
2.6.1 Langkah-langkah .....	10
2.6.2 Contoh Pemrograman .....	11
2.6.3 Analisis .....	11
2.6.4 Teori Mengenai Kode .....	11
<b>2.7 Manipulasi String dengan Input (Kode 7) .....</b>	<b>10</b>
2.7.1 Langkah-langkah .....	12
2.7.2 Contoh Pemrograman .....	12
2.5.3 Hasil Output .....	12

2.7.4 Analisis .....	13
2.7.5 Teori Mengenai Kode .....	13
BAB III KESIMPULAN .....	14
DAFTAR PUSTAKA .....	15

# **BAB I**

## **PENDAHULUAN**

### **1.1 Tujuan**

1. Memahami dan menerapkan konsep encapsulation dalam class dengan getter dan setter.
2. Menguasai penggabungan class yang berbeda, seperti penggunaan class model di class main dengan input statis atau dinamis.
3. Melatih kemampuan dalam menulis, menjalankan, dan menganalisis kode Java menggunakan Eclipse, termasuk manipulasi string.
4. Mengidentifikasi aplikasi praktis seperti validasi data dan pengecekan.

### **1.2 Manfaat**

1. Meningkatkan kemampuan logika pemrograman mahasiswa melalui latihan pembuatan class modular dan integrasinya.
2. Memberikan pengalaman praktis dalam menggunakan Eclipse, yang berguna untuk proyek pemrograman di masa depan.
3. Membantu mahasiswa dalam memahami aplikasi nyata dari penggabungan class, misalnya untuk model data seperti akun atau mahasiswa, dan manipulasi string untuk validasi.
4. Sebagai bahan referensi untuk pengembangan kemampuan troubleshooting kode, terutama pada akses atribut privat dan handling input.

## **BAB II**

### **PEMBAHASAN**

#### **2.1 Class Akun dengan Encapsulation**

Kode ini memperkenalkan konsep encapsulation dalam pemrograman berorientasi objek menggunakan Java, di mana class Akun memiliki atribut privat seperti username, password, email, dan pinAngka, yang hanya dapat diakses melalui getter dan setter. Constructor disediakan untuk inisialisasi objek, sementara method validasi untuk password dan email menambah fungsi logika sederhana. Class ini dirancang sebagai model data yang bisa digunakan di class lain, menunjukkan pentingnya pengendalian akses untuk menjaga integritas data, seperti memastikan password minimal 8 karakter dan email mengandung tanda "@" dan ".". Praktikum ini membantu mahasiswa memahami bagaimana membuat class yang reusable, yang merupakan dasar untuk pengembangan aplikasi seperti sistem autentikasi atau manajemen pengguna.

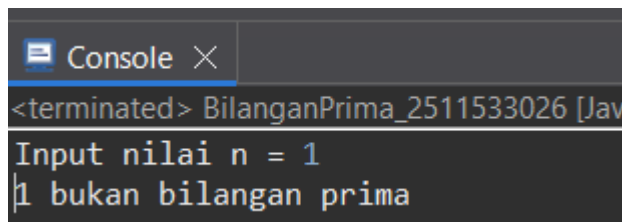
##### **2.1.1 Langkah-langkah**

1. Buka Eclipse dan buat project baru dengan package Tugas2.
2. Buat class Akun dengan atribut privat (username, password, email, pinAngka).
3. Tambahkan constructor untuk inisialisasi.
4. Buat getter dan setter untuk setiap atribut.
5. Tambahkan method validasi untuk password dan email.
6. Simpan dan compile tanpa main untuk penggunaan di class lain.

### 2.1.2 Contoh Pemrograman

```
1 package Pekan_Tujuh_2511533026;
2
3 import java.util.Scanner;
4
5 public class BilanganPrima_2511533026 {
6     public static boolean isPrime (int n) {
7         int factors = 0;
8         for (int i = 1; i <= n; i++) {
9             if (n % i == 0) {
10                 factors++;
11             }
12         }
13         return (factors == 2);
14     }
15
16     public static void main (String[] args) {
17         Scanner input = new Scanner (System.in);
18         System.out.print("Input nilai n = ");
19         int a= input.nextInt();
20         if (isPrime(a)) {
21             System.out.println(a+" bilangan prima");
22         } else {
23             System.out.println(a+" bukan bilangan prima");
24         }
25     }
26 }
```

### 2.1.3 Hasil Output



```
<terminated> BilanganPrima_2511533026 [Java...]  
Input nilai n = 1  
1 bukan bilangan prima
```

### 2.1.4 Analisis

Class ini menyimpan data akun dengan akses terkontrol melalui getter dan setter. Validasi sederhana memeriksa panjang password dan format email, tapi tidak menangani edge case seperti null. Analisis: Encapsulation mencegah akses langsung, meningkatkan keamanan data, namun validasi email bisa diperkuat dengan regex. Constructor memastikan inisialisasi penuh, tapi bisa error jika parameter tidak valid.

### 2.1.5 Teori Mengenai Kode

Encapsulation dalam OOP Java melibatkan atribut privat dan akses melalui getter/setter untuk menjaga integritas data. Constructor digunakan untuk menginisialisasi objek dengan nilai awal, sementara method validasi seperti `isPasswordValid()` dan `isEmailValid()` menambah logika bisnis. Teori ini

mendukung modularitas dan keamanan, memungkinkan class Akun digunakan dalam sistem yang lebih besar seperti autentikasi pengguna.

## **2.2 Class Mahasiswa dengan Getter/Setter**

Kode ini mendefinisikan class Mahasiswa\_2511533026 sebagai model data dengan atribut privat untuk nim (integer), nama, dan nim2 (string), yang diakses melalui getter dan setter. Dua method Cetak() dan Cetak2() disediakan untuk menampilkan data dengan format berbeda, menunjukkan fleksibilitas dalam presentasi output. Class ini dirancang untuk digunakan bersama class main lain, menunjukkan bagaimana class model dapat menyimpan data mahasiswa dengan tipe data berbeda (int dan string untuk NIM). Praktikum ini membantu mahasiswa memahami pentingnya encapsulation untuk mencegah modifikasi langsung atribut dan bagaimana method tambahan dapat meningkatkan fungsionalitas class, yang merupakan dasar untuk pengembangan aplikasi manajemen data akademik.

### **2.2.1 Langkah-langkah**

1. Buat package Pekan\_Tujuh\_2511533026.
2. Buat class Mahasiswa\_2511533026 dengan atribut privat (nim int, nama String, nim2 String).
3. Tambahkan setter untuk nim, nim2, nama.
4. Tambahkan getter untuk nim, nim2, nama.
5. Buat method Cetak dan Cetak2 untuk output.
6. Compile untuk penggunaan di class main.

### 2.2.2 Contoh Pemrograman

```
1 package Pekan_Tujuh_2511533026;
2
3 public class Mahasiswa_2511533026 {
4     // variabel global
5     private int nim;
6     private String nama,nim2;
7     // membuat mutator
8     public void setNim (int nim) {
9         this.nim=nim;
10    }
11    public void setNim2 (String nim2) {
12        this.nim2=nim2;
13    }
14    public void setNama (String nama) {
15        this.nama=nama;
16    }
17    //membuat aksesori (getter)
18    public int getNim() {
19        return nim;
20    }
21    public String getNim2() {
22        return nim2;
23    }
24    public String getNama() {
25        return nama;
26    }
27    //metode lain
28    public void Cetak() {
29        System.out.println("Nim : "+nim);
30        System.out.println("Nama : "+nama);
31    }
32    public void Cetak2() {
33        System.out.println("Nim : "+nim2);
34        System.out.println("Nama : "+nama);
35    }
36
37 }
```

### 2.2.3 Analisis

Class ini menyimpan data mahasiswa dengan dual representasi NIM (int dan string), meningkatkan fleksibilitas. Method Cetak dan Cetak2 redundan dalam struktur tapi menunjukkan variasi output. Analisis: Desain ini efisien untuk data sederhana, tapi redundansi nim/nim2 bisa dihilangkan dengan validasi lebih ketat di setter untuk konsistensi.

### 2.2.4 Teori Mengenai Kode

Getter (accessor) dan setter (mutator) adalah bagian dari encapsulation dalam OOP, memungkinkan akses terkontrol ke atribut privat. Method tambahan

seperti `Cetak()` meningkatkan fungsi class tanpa memodifikasi data langsung. Teori ini mendukung modularitas, memungkinkan class Mahasiswa digunakan dalam aplikasi seperti sistem informasi akademik.

### **2.3 Penggabungan dengan Class Main Statis (Kode 3)**

Kode ini menunjukkan penggabungan class Mahasiswa\_2511533026 dengan class main PanggilMahasiswa\_2511533026, di mana data mahasiswa diatur secara statis menggunakan setter dan ditampilkan melalui getter serta method `Cetak()`. Pendekatan ini memperkenalkan konsep instantiation objek dan pemanggilan method dalam Java, menyoroti bagaimana class model dapat digunakan dalam class pengontrol (main) untuk menghasilkan output. Dengan nilai hardcoded, kode ini sederhana dan cocok untuk demonstrasi dasar OOP, membantu mahasiswa memahami alur dari pembuatan objek hingga penggunaan data dalam program. Praktikum ini menekankan pentingnya struktur modular, di mana class terpisah memiliki tanggung jawab berbeda, yang merupakan prinsip inti dalam pengembangan software skala besar.

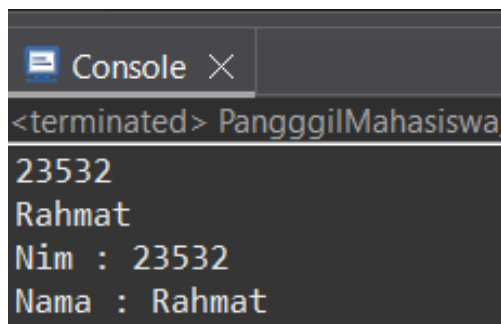
#### **2.3.1 Langkah-langkah**

1. Buat class PanggilMahasiswa\_2511533026 di package sama.
2. Instansiasi objek Mahasiswa\_2511533026.
3. Set nim dan nama dengan nilai hardcoded.
4. Cetak getter dan panggil `Cetak`.
5. Jalankan main.

### 2.3.2 Contoh Pemrograman

```
1 package Pekan_Tujuh_2511533026;  
2  
3 public class PanggilMahasiswa_2511533026 {  
4     public static void main(String[] args) {  
5         Mahasiswa_2511533026 a= new Mahasiswa_2511533026();  
6         a.setNim(23532);  
7         a.setNama("Rahmat");  
8         System.out.println(a.getNim());  
9         System.out.println(a.getNama());  
10        a.Cetak();  
11    }  
12 }
```

### 2.3.3 Hasil Output



```
<terminated> PanggilMahasiswa_2511533026  
23532  
Rahmat  
Nim : 23532  
Nama : Rahmat
```

### 2.3.4 Analisis

Kode ini menggabungkan class Mahasiswa dengan main statis, menghasilkan output duplikat dari getter dan Cetak. Analisis: Demonstrasi instantiation dan akses berhasil, tapi duplikasi output bisa dihindari dengan desain output lebih terfokus. Tidak ada error, menunjukkan pemahaman dasar OOP.

### 2.3.5 Teori Mengenai Kode

Penggabungan class melalui instantiation di method main melibatkan pembuatan objek dengan keyword new, diikuti pemanggilan method getter/setter. Teori ini mendukung pemisahan tanggung jawab antara class model dan pengontrol, memungkinkan kode yang lebih terorganisir dan mudah di-maintain.

## 2.4 Penggabungan dengan Input dan Kondisi (Kode 4)

Kode ini memperluas penggabungan class Mahasiswa\_2511533026 dengan class main PanggilMahasiswa2\_2511533026, yang menggunakan input pengguna melalui Scanner untuk mengatur data mahasiswa dan memeriksa kondisi pada NIM menggunakan method string seperti startsWith() dan contains(). Pendekatan ini

menambah interaktivitas, memungkinkan program menyesuaikan output berdasarkan input, seperti menentukan angkatan atau jurusan berdasarkan NIM. Praktikum ini membantu mahasiswa memahami bagaimana mengintegrasikan input dinamis dengan logika kondisional dalam OOP, yang relevan untuk aplikasi seperti sistem pendaftaran mahasiswa. Kode ini juga menunjukkan pentingnya menutup Scanner untuk mencegah resource leak, memperkenalkan praktik pemrograman yang baik.

#### 2.4.1 Langkah-langkah

1. Buat class PanggilMahasiswa2\_2511533026.
2. Import Scanner.
3. Ambil input nim (string) dan nama.
4. Set ke objek, cek startsWith dan contains untuk kondisi.
5. Panggil Cetak2.

#### 2.4.2 Contoh Pemrograman

```
1 package Pekan_Tujuh_2511533026;
2
3 import java.util.Scanner;
4
5 public class PanggilMahasiswa2_2511533026 {
6     public static void main (String [] args) {
7         Scanner input= new Scanner (System.in);
8         System.out.print("Nim: ");
9         String x= input.nextLine();
10        System.out.print("Nama: ");
11        String y= input.nextLine();
12        Mahasiswa_2511533026 a= new Mahasiswa_2511533026();
13        a.setNim2(x);
14        a.setNama(y);
15        if(x.startsWith("25")) {
16            System.out.println(a.getNama()+ " anda angkatan 2025");
17        }
18        if (a.getNim2().contains("1153")) {
19            System.out.println(y+" Anda Mahasiswa Informatika");
20        }
21        a.Cetak2();
22        input.close();
23    }
24 }
```

#### 2.4.3 Analisis

Kode ini mengintegrasikan input pengguna dengan class Mahasiswa, dengan kondisi string untuk validasi NIM. Method startsWith dan contains efektif untuk pengecekan sederhana. Analisis: Interaktivitas meningkatkan fleksibilitas, tapi validasi NIM bisa diperluas untuk menangani input tidak valid seperti string kosong.

#### 2.4.4 Teori Mengenai Kode

Method string seperti startsWith() dan contains() memungkinkan manipulasi dan validasi teks dalam Java. Scanner digunakan untuk input dinamis, dan penutupan Scanner mencegah resource leak. Teori ini mendukung integrasi OOP dengan interaksi pengguna, relevan untuk aplikasi berbasis data.

## 2.5 Penggabungan dengan Input Langsung (Kode 5)

Kode ini adalah varian dari kode 4, menggunakan class main PanggilMahasiswa3\_2511533026 untuk menggabungkan class Mahasiswa\_2511533026 dengan input langsung ke setter tanpa variabel sementara. Pendekatan ini menunjukkan efisiensi dalam penulisan kode sambil mempertahankan fungsi yang sama, yaitu memvalidasi NIM dan menampilkan data mahasiswa. Dengan logika kondisional yang identik, kode ini memperkuat pemahaman tentang fleksibilitas dalam desain program, di mana struktur input dapat disesuaikan tanpa mengubah hasil. Praktikum ini membantu mahasiswa melihat bagaimana variasi kecil dalam kode dapat mempertahankan fungsionalitas, yang penting untuk optimasi kode dalam pengembangan software yang lebih besar seperti sistem informasi.

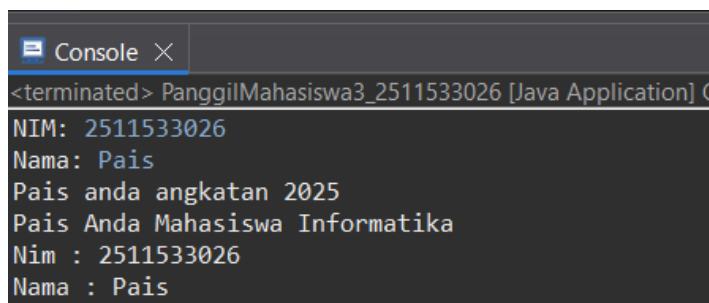
### 2.5.1 Langkah-langkah

1. Buat class PanggilMahasiswa3\_2511533026.
2. Import Scanner.
3. Instansiasi, set langsung dari input.
4. Cek kondisi, panggil Cetak2.

### 2.5.2 Contoh Pemrograman

```
1 package Pekan_Tujuh_2511533026;
2 import java.util.Scanner;
3
4 public class PanggilMahasiswa3_2511533026 {
5     public static void main (String [] args) {
6         Mahasiswa_2511533026 a= new Mahasiswa_2511533026();
7         Scanner input = new Scanner (System.in);
8         System.out.print("NIM: ");
9         a.setNim2(input.nextLine());
10        System.out.print("Nama: ");
11        a.setNama(input.nextLine());
12        if(a.getNim2().startsWith("25")) {
13            System.out.println(a.getNama()+ " anda angkatan 2025");
14        }
15        if(a.getNim2().contains("1153")) {
16            System.out.println(a.getNama()+ " Anda Mahasiswa Informatika");
17        }
18        a.Cetak2();
19        input.close();
20    }
21 }
22 }
```

### 2.5.3 Hasil Output



```
<terminated> PanggilMahasiswa3_2511533026 [Java Application] C
NIM: 2511533026
Nama: Pais
Pais anda angkatan 2025
Pais Anda Mahasiswa Informatika
Nim : 2511533026
Nama : Pais
```

#### **2.5.4 Analisis**

Varian dari kode 4 dengan input langsung ke setter, menghasilkan output sama. Analisis: Lebih ringkas, tapi fungsi identik, menunjukkan fleksibilitas desain. Tidak ada error, tapi rentan terhadap input tidak valid tanpa validasi tambahan.

#### **2.5.5 Teori Mengenai Kode**

Mengintegrasikan input langsung ke setter mengurangi variabel sementara, meningkatkan efisiensi kode. Teori ini mendukung prinsip minimalis dalam pemrograman, tetap memanfaatkan encapsulation dan logika kondisional untuk validasi data.

### **2.6 Manipulasi String Dasar (Kode 6)**

Kode ini menunjukkan manipulasi string dasar dalam Java menggunakan class `String1_2511533026`, dengan operasi seperti `length()`, `toUpperCase()`, `toLowerCase()`, dan `indexOf()` pada string statis "Assalamualaikum". Operasi ini memperkenalkan kemampuan dasar Java untuk memanipulasi teks, yang penting untuk pengolahan data seperti format output atau pencarian substring. Praktikum ini membantu mahasiswa memahami sifat immutable string di Java dan bagaimana method string mengembalikan nilai baru tanpa mengubah aslinya. Kode ini sederhana namun efektif untuk menunjukkan dasar-dasar manipulasi teks, yang merupakan keterampilan penting dalam pengembangan aplikasi seperti pengolahan form input atau analisis data teks.

#### **2.6.1 Langkah-langkah**

1. Buat class `String1_2511533026`.
2. Definisi string salam.
3. Cetak `length`, `upper`, `lower`, `indexOf`.
4. Jalankan main.

### 2.6.2 Contoh Pemrograman

```
1 package Pekan_Tujuh_2511533026;
2
3 public class String1_2511533026 {
4     public static void main(String[] args) {
5         String salam = "Assalamualaikum";
6         System.out.println("panjang salam adalah: " + salam.length());
7         System.out.println(salam.toUpperCase()); //Outputs "ASSALAMUALAIKUM"
8         System.out.println(salam.toLowerCase()); //Outputs "assalamualaikum"
9         System.out.println(salam.indexOf("salam")); //Outputs2
10    }
11 }
```

### 2.6.3 Analisis

Kode ini mendemonstrasikan method string dasar dengan output sesuai ekspektasi. IndexOf() menghitung dari indeks 0, case-sensitive. Analisis: Efektif untuk pengenalan, tapi tidak menangani kasus seperti substring tidak ditemukan (return -1).

### 2.6.4 Teori Mengenai Kode

String di Java bersifat immutable, sehingga method seperti toUpperCase() mengembalikan string baru. Method length(), indexOf(), dan lainnya mendukung manipulasi teks untuk pencarian atau format. Teori ini penting untuk aplikasi seperti validasi input atau pengolahan data teks.

## 2.7 Manipulasi String dengan Input (Kode 7)

Kode ini memperluas manipulasi string dengan class String2\_2511533026, menggabungkan input pengguna melalui Scanner untuk nama depan dan belakang, serta demonstrasi operasi seperti concat(), escape sequence, dan perbedaan antara penjumlahan string versus integer. Kode ini menyoroti bagaimana operator + berfungsi berbeda pada string (concatenation) dan integer (aritmatika), serta pentingnya escape character untuk menampilkan tanda kutip. Praktikum ini membantu mahasiswa memahami interaksi antara input dinamis dan manipulasi teks, yang relevan untuk aplikasi seperti pembuatan laporan atau format data pengguna. Kode ini juga memperkenalkan konsep tipe data campuran, menunjukkan bagaimana Java menangani string dan integer dalam operasi yang sama.

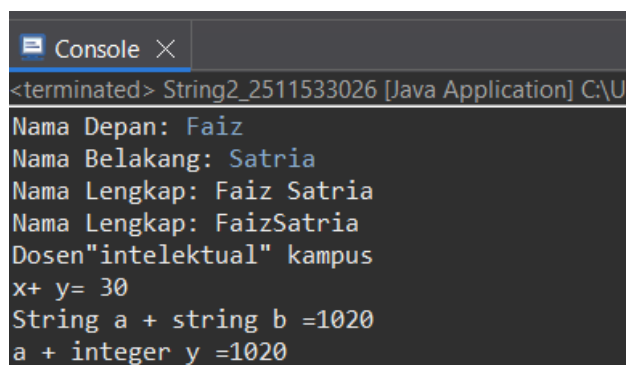
### 2.7.1 Langkah-langkah

1. Buat class String2\_2511533026.
2. Import Scanner.
3. Ambil input nama depan/belakang.
4. Demo concat, escape, penjumlahan string vs int.

### 2.7.2 Contoh Pemrograman

```
1 package Pekan_Tujuh_2511533026;
2
3 import java.util.Scanner;
4
5 public class String2_2511533026 {
6     public static void main(String[] args) {
7         Scanner input= new Scanner(System.in);
8         System.out.print("Nama Depan: ");
9         String firstName= input.nextLine();
10        System.out.print("Nama Belakang: ");
11        String lastName= input.nextLine();
12        String txt1 = "Dosen\"intelektual\" kampus";
13        System.out.println("Nama Lengkap: "+firstName + " " + lastName);
14        System.out.println("Nama Lengkap: "+firstName.concat(lastName));
15        System.out.println(txt1);
16        int x = 10;
17        int y = 20;
18        int z = x + y;
19        System.out.println("x+ y= "+z);
20        String a = "10";
21        String b = "20";
22        String c = a + b;
23        System.out.println("String a + string b =" +c);
24        String v = a + y;
25        System.out.println("a + integer y =" +v);
26    }
27 }
```

### 2.7.3 Hasil Output



```
<terminated> String2_2511533026 [Java Application] C:\U
Nama Depan: Faiz
Nama Belakang: Satria
Nama Lengkap: Faiz Satria
Nama Lengkap: FaizSatria
Dosen"intelektual" kampus
x+ y= 30
String a + string b =1020
a + integer y =1020
```

#### **2.7.4 Analisis**

Kode ini menggabungkan input pengguna dengan manipulasi string, menunjukkan perbedaan concatenation dan aritmatika. Escape sequence berfungsi baik. Analisis: Concat tanpa spasi menghasilkan teks gabung, dan penjumlahan string/int otomatis konversi ke string. Tidak ada error, tapi input kosong tidak ditangani.

#### **2.7.5 Teori Mengenai Kode**

Operator + di Java overload untuk concatenation pada string dan aritmatika pada integer. Escape sequence seperti " memungkinkan karakter khusus. Method concat() mirip + tapi lebih eksplisit. Teori ini mendukung manipulasi teks dinamis dalam aplikasi interaktif.

## **BAB III**

### **KESIMPULAN**

Praktikum pertemuan kedua ini berhasil memberikan pemahaman dasar mengenai penggabungan class yang berbeda dalam bahasa Java menggunakan Eclipse. Melalui pembuatan class model seperti Akun dan Mahasiswa, mahasiswa dapat melihat bagaimana encapsulation dengan getter/setter memungkinkan akses terkontrol, sementara integrasi di class main dengan input statis atau dinamis menunjukkan modularitas OOP. Manipulasi string melalui method seperti `startsWith()`, `contains()`, dan `concat()` melengkapi dengan kemampuan validasi dan pengolahan teks. Secara keseluruhan, praktikum ini memperkuat fondasi pengembangan program berbasis objek untuk aplikasi nyata seperti sistem manajemen data.

Dari analisis kode, terlihat bahwa kesalahan seperti akses langsung atribut privat dapat dihindari dengan getter/setter, dan input dinamis menambah interaktivitas tetapi memerlukan validasi tambahan untuk menangani edge case. Manfaat praktikum ini mencakup keterampilan dalam troubleshooting logika kondisional, manipulasi string, dan penggunaan package untuk organisasi kode. Penggunaan Scanner dan penutupan resource menunjukkan praktik pemrograman yang baik, yang penting untuk proyek skala besar.

Akhirnya, praktikum ini mendorong mahasiswa untuk bereksperimen lebih lanjut, seperti menambahkan inheritance atau exception handling untuk meningkatkan robustitas kode. Pemahaman ini dapat diterapkan pada pengembangan aplikasi seperti sistem pendaftaran atau pengolahan data pengguna. Saran untuk praktikum berikutnya adalah mengintegrasikan elemen GUI atau database sederhana untuk memperluas cakupan aplikasi dan memberikan pengalaman yang lebih komprehensif dalam pengembangan software.

## DAFTAR PUSTAKA

1. Schildt, Herbert. (2019). *Java: The Complete Reference*. McGraw-Hill Education.
2. Oracle. (2023). *Java Documentation*.
3. Deitel, Paul & Deitel, Harvey. (2020). *Java How to Program*. Pearson.
4. Tutorialspoint. (2024). *Java OOP Tutorial*.