

LAPORAN PRATIKUM  
ALGORITMA DAN PEMROGRAMAN  
“ PENERAPAN DASAR *OBJECT-ORIENTED PROGRAMMING* DAN  
MANIPULASI *STRING* DALAM *JAVA* ”

disusun Oleh:

Muhammad Yasin Habiburrahman

2511532016

Dosen Pengampu:

Dr. Wahyudi. S.T.M.T

Asisten Pratikum:

Muhammad Zaki Al Hafiz



DEPARTEMEN INFORMATIKA FAKULTAS TEKNOLOGI INFORMASI  
UNIVERSITAS ANDALAS  
TAHUN 2025

## KATA PENGANTAR

Puji dan syukur kami panjatkan ke hadirat Allah SWT atas segala rahmat dan karunia-Nya, sehingga kami dapat menyelesaikan tugas kelompok ini dengan baik. Shalawat serta salam semoga senantiasa tercurah kepada junjungan kita Nabi Muhammad SAW, keluarga, sahabat, serta para pengikutnya hingga akhir zaman.

Tugas kelompok ini kami susun dengan tujuan untuk memenuhi salah satu tugas mata kuliah Algoritma dan Pemrograman, dengan tema “Penerapan dasar *object-oriented programming* dan manipulasi *String* dalam Java”, yang mencakup bagaimana *class* dan *object* saling berinteraksi antar program dan sub-program.

Kami menyadari bahwa penyusunan tugas ini masih jauh dari sempurna, baik dari segi materi maupun penyajiannya. Oleh karena itu, kami sangat mengharapkan kritik dan saran yang membangun dari semua pihak demi kesempurnaan tugas kami di masa mendatang.

Akhir kata, kami mengucapkan terima kasih kepada dosen pembimbing yang telah memberikan arahan, serta kepada seluruh anggota kelompok yang telah bekerja sama dengan baik sehingga tugas ini dapat terselesaikan tepat waktu. Semoga tugas ini dapat bermanfaat bagi kami khususnya, dan bagi pembaca pada umumnya.

Padang, 11 November 2025

Muhammad Yasin Habiburrahman

## DAFTAR ISI

<b>KATA PENGANTAR .....</b>	i
<b>DAFTAR ISI .....</b>	ii
<b>BAB I PENDAHULUAN .....</b>	1
1.1 Latar Belakang .....	1
1.2 Tujuan .....	1
1.3 Manfaat Praktikum.....	2
<b>BAB II PEMBAHASAN .....</b>	3
2.1 Mengecek Kebenaran Prima Suatu Bilangan .....	3
2.2 Penerapan <i>Class</i> dan <i>Method</i> pada Pemrograman Berorientasi Objek .....	4
2.3 Menggunakan <i>Object</i> dengan <i>Input User</i> dan Operasi pada <i>String</i> .....	7
2.4 Pengenalan Kelas <i>String</i> dan Beberapa <i>Method</i> Dasarnya .....	9
2.5 Operasi Penggabungan (Concatenation) dan Manipulasi String.....	10
<b>BAB III KESIMPULAN.....</b>	13

## BAB I

# PENDAHULUAN

### 1.1 Latar Belakang

Pada pemrograman modern, pendekatan Object-Oriented Programming (OOP) menjadi dasar utama dalam pengembangan perangkat lunak karena mampu mengorganisasi kode secara lebih terstruktur, efisien, dan mudah dikembangkan. Java sebagai salah satu bahasa pemrograman berorientasi objek menyediakan fitur-fitur penting seperti class, object, method, dan encapsulation yang memungkinkan programmer memisahkan antara data dan perilakunya.

Melalui konsep OOP, sebuah program tidak hanya terdiri dari sekumpulan perintah, tetapi juga kumpulan object yang saling berinteraksi. Setiap object memiliki atribut dan perilaku yang didefinisikan oleh class tempatnya berasal. Pemahaman mengenai class, object, method, dan penggunaan kata kunci *this* menjadi landasan penting sebelum mempelajari konsep OOP yang lebih kompleks seperti inheritance dan polymorphism.

Selain itu, pada praktikum ini juga dipelajari pengolahan teks menggunakan kelas String. Karena String di Java merupakan sebuah object, maka manipulasi teks — seperti penggabungan, perubahan huruf besar-kecil, dan pencarian karakter — dapat dilakukan dengan memanfaatkan berbagai method bawaan. Pemahaman terhadap kelas String sangat penting karena data teks merupakan komponen umum dalam berbagai aplikasi, mulai dari formulir input hingga pengolahan data pengguna.

### 1.2 Tujuan

1. Memahami konsep dasar Object-Oriented Programming (OOP) dalam bahasa Java.
2. Menjelaskan fungsi dan cara kerja class, object, method, serta penerapan *encapsulation*.
3. Menjelaskan penggunaan kata kunci *this* dalam membedakan variabel lokal dan atribut object.
4. Menunjukkan cara melakukan instansiasi object menggunakan kata kunci *new*.
5. Menerapkan manipulasi teks menggunakan kelas *String* dan berbagai method bawaannya.
6. Menjelaskan perbedaan antara operasi aritmetika dan penggabungan teks (*concatenation*) di Java.

### **1.3 Manfaat Praktikum**

1. Memahami dan menerapkan dasar pemrograman berorientasi objek dalam *Java*.
2. Mampu membuat, mengakses, serta memodifikasi data menggunakan *class* dan *object*.
3. Menyadari pentingnya prinsip *encapsulation* dalam menjaga keamanan data program.
4. Menguasai penggunaan *method* bawaan dari kelas *String* untuk manipulasi teks.
5. Membedakan konteks penggunaan operator + sebagai operasi aritmetika maupun penggabungan teks.
6. Menyiapkan dasar pemahaman yang kuat untuk mempelajari konsep lanjutan seperti *inheritance* dan *polymorphism*.

## BAB II

### PEMBAHASAN

#### 2.1 Mengecek Kebenaran Prima Suatu Bilangan

Bilangan prima adalah suatu bilangan bulat positif yang memiliki tepat 2 faktor perkalian, yaitu dirinya sendiri, dan 1. Misalkan suatu bilangan bulat positif  $n$ , kita bisa mengecek apakah suatu bilangan prima atau tidak dengan mengecek dari 1-. Kemudian mengecek apakah ada suatu bilangan  $k$  yang habis membagi  $n$ . Jika terdapat tepat 2 bilangan yang habis membagi dari 1 sampai  $n$ , maka bilangan tersebut adalah prima. Program Jawa yang mengecek suatu bilangan prima atau tidak tertera pada gambar di bawah.

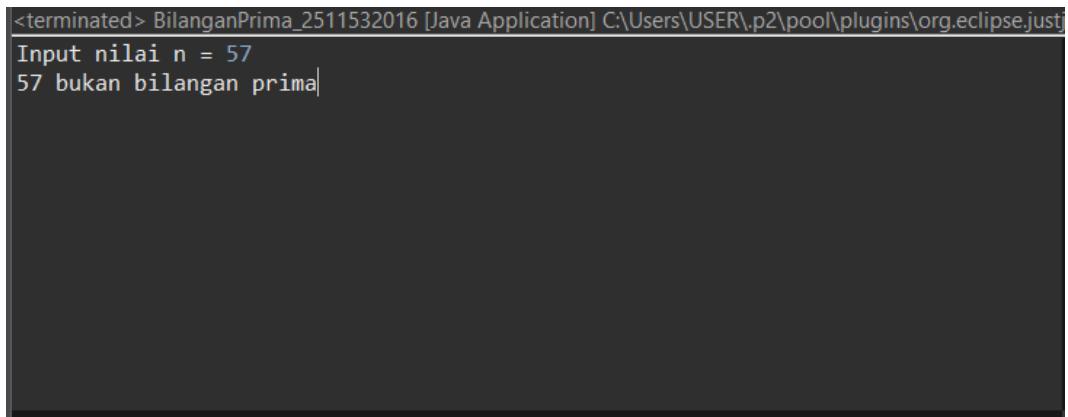
```

1 package pekan7_2511532016;
2 import java.util.Scanner;
3
4 public class BilanganPrima_2511532016 {
5     public static boolean isPrime(int n) {
6         int factors = 0;
7         for(int i=1; i<= n; i++) {
8             if (n %i == 0) {
9                 factors++;
10            }
11        }
12        return(factors ==2);
13    }
14
15    public static void main(String[] args) {
16        Scanner input = new Scanner(System.in);
17        System.out.print("Input nilai n = ");
18        int a = input.nextInt();
19        input.close();
20        if(isPrime(a)) System.out.print(a + " bilangan prima");
21        else System.out.print(a + " bukan bilangan prima" );
22    }
23}
```

Gambar 2.1 – Kode Program Bilangan Prima

Program di atas menggunakan suatu sub-program. Sub-program dikenal dengan sebutan *method* atau fungsi. Pada bagian awal, *method* `isPrime(int n)` dideklarasikan dengan tipe kembalian *boolean*, yang berarti nilai yang dikembalikan hanya berupa *true* atau *false*. Di dalam *method* ini, dilakukan proses perulangan menggunakan *for loop* dari 1 sampai  $n$  untuk menghitung banyaknya faktor pembagi bilangan tersebut. Setiap kali  $n$  habis dibagi oleh  $i$ , nilai *factors* akan bertambah satu. Setelah perulangan selesai, *method* mengembalikan nilai *true* jika `factors == 2`, yang berarti bilangan hanya memiliki dua pembagi (1 dan dirinya sendiri), sesuai definisi bilangan prima.

Bagian *main method* berfungsi sebagai titik awal program. Dengan menggunakan kelas *Scanner*, program meminta pengguna memasukkan sebuah bilangan bulat melalui perintah `input.nextInt()`. Nilai ini kemudian disimpan ke dalam variabel *a* dan dikirim sebagai argumen ke *method* `isPrime(a)`. Hasil dari pemanggilan *method* tersebut kemudian digunakan dalam struktur *if-else* untuk menampilkan hasil ke layar.

A screenshot of a terminal window titled "BilanganPrima\_2511532016 [Java Application] C:\Users\USER\p2\pool\plugins\org.eclipse.jdt.core\src\BilanganPrima\BilanganPrima.java". The window shows the following text:

```
<terminated> BilanganPrima_2511532016 [Java Application] C:\Users\USER\p2\pool\plugins\org.eclipse.jdt.core\src\BilanganPrima\BilanganPrima.java
Input nilai n = 57
57 bukan bilangan prima
```

The window has a dark background and white text.

Gambar 2.2 – Contoh Keluaran Program Bilangan Prima

Pada contoh di atas, pengguna memasukkan 57 sebagai angka yang akan dicek nilai kebenaran primanya. Berdasarkan program, 57 bukanlah bilangan prima karena 57 memiliki faktor 1, 3, 19, dan 57 itu sendiri.

## 2.2 Penerapan *Class* dan *Method* pada Pemrograman Berorientasi Objek

Java merupakan salah satu bahasa pemrograman yang populer. Bahasa ini dikenal sebagai bahasa pemrograman yang berorientasi pada objek (*Object-Oriented Programming / OOP*).

Program disusun berdasarkan konsep *class* dan *object*.

- *Class*: Dapat diartikan sebagai cetak biru (*blueprint*) yang mendefinisikan struktur dan perilaku suatu objek.
- *Object* adalah wujud nyata dari sebuah *class*.

Setiap *class* dapat memiliki:

1. Atribut (*attribute*) → variabel yang menyimpan data atau keadaan objek.
2. Metode (*method*) → fungsi yang mendefinisikan perilaku atau aksi objek.

```

1 package pekan7_2511532016;
2
3 public class Mahasiswa_2511532016 {
4     //Variabel global
5     private int nim;
6     private String nama, nim2;
7     //Membuat mutator (setter)
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
18    public static void main(String[] args) {
19    }
20
21    public int getNim() {
22        return nim;
23    }
24    public String getNim2() {
25        return nim2;
26    }
27    public String getNama() {
28        return nama;
29    }
30
31    //Metode lain
32    public void Cetak() {
33        System.out.println("Nim : " + nim);
34        System.out.println("Nama : " + nama);
35    }
36    public void Cetak2() {
37        System.out.println("Nim : " + nim2);
38        System.out.println("Nama : " + nama);
39    }
40 }
```

Gambar 2.3 – Kode Program *class* Mahasiswa

Di sini muncul kata kunci *this*. *this.nim* berarti atribut nim milik *object* yang sedang aktif. Ketika parameter *method* memiliki nama yang sama dengan atribut kelas, penulisan *this.nim = nim;* diperlukan untuk membedakan atribut kelas (kiri) dengan parameter lokal (kanan). Tanpa *this*, penulisan *nim = nim;* hanya akan meng-*assign* parameter ke dirinya sendiri dan atribut kelas tidak berubah.

Program di atas mendefinisikan sebuah *class* dengan nama Mahasiswa\_2511532016 yang merepresentasikan data seorang mahasiswa. Di dalam *class* ini terdapat tiga atribut utama, yaitu nim, nim2, dan nama.

Atribut-atribut tersebut dideklarasikan dengan *access modifier* private agar tidak dapat diakses langsung dari luar *class*.

Untuk mengatur dan mengambil nilai atribut tersebut, digunakan *method* khusus, yaitu:

- setNim(), setNim2(), dan setNama() sebagai *setter*, yang berfungsi memberi nilai pada atribut.
- getNim(), getNim2(), dan getNama() sebagai *getter*, yang berfungsi mengembalikan nilai dari atribut.

Selain itu, terdapat dua *method* tambahan yaitu Cetak() dan Cetak2() yang berfungsi menampilkan data mahasiswa ke layar. Perbedaan keduanya hanya terletak pada jenis NIM yang digunakan (int atau String).

Struktur ini belum membuat objek secara langsung, karena *method main()* masih kosong. Namun, *class* ini sudah siap digunakan sebagai cetak biru untuk membuat objek mahasiswa di program lain.

Dalam pemrograman berorientasi objek, *class* tidak akan berguna sampai kita membuat *object*-nya. *Object* adalah wujud nyata/instansi dari sebuah *class*, yaitu entitas yang menyimpan data dan memiliki perilaku sesuai dengan rancangan *class*-nya.

```
1 package pekan7_2511532016;
2
3 public class PanggilMahasiswa_2511532016 {
4     public static void main(String[] args) {
5         Mahasiswa_2511532016 a = new Mahasiswa_2511532016();
6         a.setNim(2016);
7         a.setNama("Muhammad Yasin Habiburrahman");
8         System.out.println(a.getNim());
9         System.out.println(a.getNama());
10        a.Cetak();
11    }
12 }
```

Gambar 2.4 – Kode Program Panggil Mahasiswa

Untuk membuat *object* baru dari sebuah *class*, digunakan kata kunci new, diikuti nama *class* dan tanda kurung seperti memanggil *constructor*. Misalnya seperti contoh di atas:

Mahasiswa\_2511532016 a = new Mahasiswa\_2511532016();

Pernyataan di atas membuat sebuah *object* bernama *a* dari *class Mahasiswa\_2511532016*. Setelah objek dibuat, seluruh *method* dan *attribute* dalam kelas tersebut dapat diakses melalui notasi titik (.)

Program di atas menunjukkan cara membuat dan menggunakan *object* dari *class Mahasiswa\_2511532016*. Pada baris *Mahasiswa\_2511532016 a = new Mahasiswa\_2511532016();*, sebuah *object* bernama *a* dibuat dengan memanggil kelas *Mahasiswa\_2511532016*.

Setelah *object* *a* terbentuk, program memanfaatkan *setNim()* dan *setNama()* untuk memberikan nilai pada atribut nim dan nama. Proses ini merupakan penerapan prinsip *encapsulation*, di mana data tidak dimodifikasi langsung, melainkan melalui *method* pengatur nilai.

Selanjutnya, *method getter* *getNim()* dan *getNama()* digunakan untuk mengambil dan menampilkan nilai yang telah disimpan di dalam atribut. Dua perintah berikut:

```
System.out.println(a.getNim());  
System.out.println(a.getNama());
```

akan menampilkan nilai nim dan nama secara terpisah.

Kemudian, pemanggilan *a.Cetak();* menjalankan *method* yang sudah didefinisikan di dalam kelas *Mahasiswa\_2511532016*, yang secara otomatis menampilkan informasi mahasiswa.

```
<terminated> PanggilMahasiswa_2511532016 [Java Application] C:\Users\USER\p2\pool\plugins\org.eclipse.  
2016  
Muhammad Yasin Habiburrahman  
Nim : 2016  
Nama : Muhammad Yasin Habiburrahman
```

Gambar 2.5 – Keluaran dari Kode Program Panggil Mahasiswa

### 2.3 Menggunakan *Object* dengan *Input User* dan *Operasi pada String*

Dalam pemrograman berorientasi objek, *object* dapat dikombinasikan dengan interaksi pengguna untuk membuat program yang lebih dinamis dan fleksibel. Salah satu contohnya adalah ketika data untuk suatu *object* tidak ditulis langsung di dalam program, tetapi dimasukkan oleh pengguna melalui *input*.

```

1 package pekan7_2511532016;
2 import java.util.Scanner;
3
4 public class PanggilMahasiswa2_25115532016 {
5     public static void main(String[] args) {
6         Scanner input = new Scanner(System.in);
7         System.out.print("NIM: ");
8         String x = input.nextLine();
9         System.out.print("Nama: ");
10        String y = input.nextLine();
11        Mahasiswa_2511532016 a = new Mahasiswa_2511532016();
12        a.setNim2(x);
13        a.setNama(y);
14        if(x.startsWith("25")) {
15            System.out.println(y + " anda angkatan 2025");
16        }
17        if(x.contains("1153")) {
18            System.out.println("Anda Mahasiswa Informatika");
19        }
20        a.Cetak2();
21        input.close();
22    }
23}
24 }
```

Gambar 2.6 – Kode Program Panggil Mahasiswa 2

Program ini merupakan versi interaktif dari program sebelumnya. Kali ini, data mahasiswa tidak diatur langsung di dalam kode, tetapi dimasukkan oleh pengguna melalui keyboard menggunakan kelas Scanner.

Pertama, program meminta pengguna untuk mengisi NIM dan nama. Nilai untuk NIM dimasukkan dan disimpan dalam variabel x, dan y untuk nama. Selanjutnya, dibuat sebuah *object* a dari kelas Mahasiswa\_2511532016. Nilai NIM dan nama yang diperoleh dari pengguna disimpan ke dalam atribut *object* melalui *method setter* setNim2() dan setNama().

Selain itu, bahasa Java menyediakan berbagai *method* bawaan pada kelas *String* untuk memeriksa dan memanipulasi teks. Beberapa di antaranya adalah:

- .startsWith() : mengecek apakah sebuah teks dimulai dengan huruf atau angka tertentu.
- .contains() : memeriksa apakah sebuah teks mengandung potongan teks tertentu di dalamnya.

Selanjutnya, digunakan dua *if statement* yang memanfaatkan *method* bawaan dari kelas *String*:

- x.startsWith("25") → memeriksa apakah NIM diawali dengan angka “25”. Jika benar, program menampilkan pesan bahwa mahasiswa tersebut merupakan angkatan 2025.

- `x.contains("1153")` → memeriksa apakah NIM mengandung kode “1153”, yang dalam konteks ini dianggap menunjukkan mahasiswa Program Studi Informatika.

Setelah pemeriksaan selesai, *method* `a.Cetak2()` dipanggil untuk menampilkan data mahasiswa yang telah dimasukkan.

```
<terminated> PanggilMahasiswa2_25115532016 [Java Application] C:\Users\USER\p2\pool\plugins\org.eclipse.jdt.core\src\String1_2511532016.java
NIM: 2511532016
Nama: Muhammad Yasin Habiburrahman
Muhammad Yasin Habiburrahman anda angkatan 2025
Anda Mahasiswa Informatika
Nim : 2511532016
Nama : Muhammad Yasin Habiburrahman
```

Gambar 2.7 – Contoh Keluaran Kode Program Panggil Mahasiswa 2

## 2.4 Pengenalan Kelas *String* dan Beberapa *Method* Dasarnya

Dalam bahasa pemrograman *Java*, teks atau kumpulan karakter direpresentasikan menggunakan kelas *String*. Berbeda dengan tipe data primitif seperti `int` atau `char`.

*String* adalah objek dari kelas `java.lang.String`. Karena merupakan kelas, *String* memiliki banyak *method* bawaan yang dapat digunakan untuk memanipulasi teks, seperti menghitung panjang, mengubah huruf menjadi besar atau kecil, mencari potongan teks tertentu, dan lain-lain.

```
1 package pekan7_2511532016;
2
3 public class String1_2511532016 {
4
5     public static void main(String[] args) {
6         String salam = "Assalamualaikum";
7         System.out.println("Panjang salam adalah " + salam.length());
8         System.out.println(salam.toUpperCase());
9         System.out.println(salam.toLowerCase());
10        System.out.println(salam.indexOf("salam"));
11
12    }
13
14 }
```

Gambar 2.8 – Kode Program String 1

Program di atas memperlihatkan penggunaan beberapa *method* bawaan dari kelas *String*. Pertama, variabel `salam` dideklarasikan dan diisi dengan teks "Assalamualaikum". Karena *String* adalah objek, maka variabel `salam` memiliki akses ke berbagai *method* untuk memanipulasi teks tersebut.

- a. salam.length()  
Menghitung jumlah seluruh karakter di dalam teks, termasuk huruf besar dan kecil. Dalam hal ini, hasilnya adalah **15**, karena “Assalamualaikum” memiliki 15 karakter.
  
- b. salam.toUpperCase()  
Mengubah seluruh huruf menjadi huruf besar
  
- c. salam.toLowerCase()  
Mengubah seluruh huruf menjadi huruf kecil
  
- d. salam.indexOf("salam")  
Mencari posisi awal kata “salam” di dalam teks. Karena posisi indeks di Java dimulai dari 0, maka hasilnya adalah **2** (huruf ‘s’ pertama berada di posisi ke-2).

```
panjang salam adalah 15
ASSALAMUALAIKUM
assalamualaikum
2
```

Gambar 2.8 – Keluaran Kode Program String 1

## 2.5 Operasi Penggabungan (Concatenation) dan Manipulasi String

Dalam bahasa pemrograman *Java*, tanda “+” tidak hanya berfungsi untuk operasi aritmetika, tetapi juga digunakan untuk menggabungkan teks atau *string* — proses ini disebut *concatenation*.

Ketika salah satu operand bertipe *String*, Java akan mengonversi operand lainnya (misalnya bilangan) menjadi *String* secara otomatis, lalu menggabungkannya.

Selain itu, kelas *String* menyediakan *method* *.concat()* untuk melakukan hal yang sama dengan operator +, tetapi secara eksplisit menyatakan bahwa yang dilakukan adalah penggabungan teks.

Java juga mendukung penggunaan tanda “\” di dalam teks untuk menampilkan tanda kutip ganda tanpa menimbulkan kesalahan sintaks.

```

1 package pekan7_2511532016;
2
3 public class String2_2511532016 {
4
5     public static void main(String[] args) {
6         String firstName = "Syifa";
7         String lastName = "Muhasannah";
8         String txt1 = "Dosen\"intelektual\" kampus";
9         System.out.println("Nama Lengkap: " + firstName + " " + lastName);
10        System.out.println("Nama Lengkap: " + firstName.concat(lastName));
11        System.out.println(txt1);
12        int x = 10;
13        int y = 20;
14        int z = x+y;
15
16        System.out.println("x + y= " + z);
17        String a = "10";
18        String b = "20";
19        String c = a + b;
20        System.out.println("String a + string b = " +c);
21        String v = a + y;
22        System.out.println("String a + Integer y = " + v);
23
24    }
25
26 }

```

Gambar 2.9 – Kode Program String 2

Program di atas mendemonstrasikan berbagai cara untuk menggabungkan *string* dan memanipulasi nilai teks di Java.

Pertama-tama, dideklarasikan 3 variabel string dengan nama fisrtName, lastName, dan txt1 yang memiliki nilai “Syifa”, “Muhasannah”, dan “Dosen\\”intelektual\\” kampus” berturut-turut. Kemudian, dilakukan penggabungan antar *String*,yaitu antara variabel firstName dan lastName yang dipisahkan dengan spasi menggunakan “ “.

Selanjutnya, kembali dilakukan penggabungan *String* kembali dengan variabel yang sama, tetapi kini menggunakan .concat(); ini digunakan untuk menyatukan teks. Perbedaan dengan yang sebelumnya hanya pada penambahan spasi.

Pada String txt1 = "Dosen\\\"intelektual\\\" kampus";, tanda \\\" berfungsi untuk menampilkan tanda kutip ganda di dalam teks tanpa menyebabkan error.

Perbedaan antara operasi numerik dan penggabungan teks dapat dilihat pada baris program selanjutnya. Dideklarasikan 3 buah variabel dengan tipe data *int* yaitu x, y, dan z yang memiliki nilai 10, 20, dan x+y berturut-turut. Kemudian program mengeluarkan nilai z dengan tambahan persamaan “x + y = “ di depannya. Hasil keluarannya adalah 30.

Kemudian, dideklarasikan 3 variabel dengan tipe data *String* yaitu a, b, dan c yang menyimpan nilai “10”, “20”, dan a+b berturut-turut. Lalu, ditampilkan nilai c ke layar. Hasilnya adalah “1020”. Java menganggap bahwa ini adalah kedua teks yang ditempelkan, bukan sebagai angka.

Terakhir, ketika *string* digabung dengan bilangan bulat. Dibuat suatu variabel baru dengan tipe data *String* yang menyimpan “penjumlahan” nilai *a* yang bertipe data *String* dan *y* yang bertipe data *int*. Hasil keluarannya adalah “1020”. Program otomatis mengubah bilangan bulat menjadi *String* ketika dilakukan penggabungan *String* dan *int*.

```
<terminated> String2_2511532016 [Java Application] C:\Users\USER\p2\pool\plugins\org.eclipse.justj.openj
Nama Lengkap: Syifa Muhasannah
Nama Lengkap: SyifaMuhasannah
Dosen "intelektual" kampus
x + y= 30
String a + string b = 1020
String a + Integer y = 1020
```

Gambar 2.10 – Keluaran Kode Program String 2

## BAB III

### KESIMPULAN

Pada praktikum minggu ketujuh ini, mahasiswa mempelajari dasar-dasar pemrograman berorientasi objek (*Object-Oriented Programming / OOP*) serta penerapannya dalam bahasa *Java*. Melalui serangkaian percobaan, mahasiswa memahami bahwa *class* berfungsi sebagai cetak biru (*blueprint*) untuk membentuk *object*, yang nantinya akan memiliki atribut dan perilaku tersendiri.

Konsep *encapsulation* menjadi hal penting yang diperkenalkan melalui penggunaan *method setter* dan *getter*, di mana data disimpan secara tertutup dan hanya dapat diakses atau diubah melalui *method* khusus. Penggunaan kata kunci *this* dijelaskan sebagai cara untuk membedakan antara variabel lokal dan atribut milik *object* yang sedang aktif. Mahasiswa juga mempelajari proses *instansiasi* — pembuatan *object* baru menggunakan kata kunci *new* — serta bagaimana *object* tersebut dapat digunakan untuk menyimpan dan menampilkan data.

Selain itu, praktikum ini juga memperkenalkan penggunaan kelas *String* beserta beberapa *method* bawaan seperti *.length()*, *.toUpperCase()*, *.toLowerCase()*, *.indexOf()*, dan *.concat()*. Melalui percobaan tersebut, mahasiswa dapat memahami bahwa *String* bukan hanya kumpulan karakter, tetapi merupakan *object* yang memiliki berbagai *method* untuk manipulasi teks.

Praktikum kemudian diakhiri dengan pengenalan konsep *concatenation*, yaitu proses penggabungan teks menggunakan operator *+* maupun *method .concat()*. Mahasiswa juga belajar membedakan antara operasi aritmetika dan penggabungan teks, serta bagaimana *Java* secara otomatis mengonversi tipe data bilangan menjadi *String* ketika digunakan bersama teks.

Secara keseluruhan, melalui praktikum ini mahasiswa memperoleh pemahaman mendasar tentang bagaimana *object*, *method*, dan *class* saling berinteraksi dalam paradigma *OOP*, serta bagaimana konsep tersebut diterapkan dalam pengelolaan dan manipulasi data berbasis teks menggunakan kelas *String* di *Java*.