LAPORAN PRAKTIKUM ALGORITMA DAN PEMROGRAMAN

METHOD DAN STRING PADA JAVA

PEMROGRAMAN JAVA

disusun Oleh:

Nama: Amiratul Fadhilah

NIM: 2511532023

Dosen Pengampu: Dr. Wahyudi, S.T, M.T

Asisten Praktikum: Jovantri Immanuel Gulo

DEPARTEMEN INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS ANDALAS

PADANG

2025

KATA PENGANTAR

Puji syukur ke hadirat Allah Yang Maha Esa atas rahmat dan karunia-Nya, sehingga laporan praktikum dengan judul "*Method* dan *String* pada Java" ini dapat diselesaikan dengan baik dan tepat waktu.

Laporan ini disusun sebagai salah satu syarat untuk memenuhi tugas mata kuliah Praktikum Algoritma dan Pemrograman. Tujuan utama dari penyusunan laporan ini adalah untuk menguji pemahaman dan implementasi dari *method* sebagai dasar pemrograman modular, serta penggunaan *class String* sebagai objek untuk manipulasi data teks dalam bahasa Java.

Penulis menyadari bahwa laporan ini masih memiliki banyak kekurangan, baik dari segi materi, analisis, mau. Oleh karena itu, kritik dan saran yang membangun sangat diharapkan untuk perbaikan di masa mendatang. Semoga laporan ini dapat memberikan manfaat serta pemahaman yang lebih mendalam bagi pembaca.

Padang, 13 November 2025

Penulis

DAFTAR ISI

BAB I

PENDAHULUAN

1.1 Latar Belakang

Dalam pengembangan perangkat lunak, efisiensi dan keterbacaan kode adalah dua prinsip utama. Seiring dengan meningkatnya kompleksitas program, penulisan seluruh logika dalam satu blok menjadi tidak praktis. Hal ini akan menghasilkan kode yang sulit dikelola, diperbarui, dan di-*st*. Untuk mengatasi masalah ini, bahasa pemrograman Java menerapkan konsep *method*. *Method* memungkinkan pemrogram untuk memecah program besar menjadi bagian-bagian yang lebih kecil dan de atau terstruktur, di mana setiap *method* bertanggung jawab untuk satu tugas spesifik. Prinsip ini dikenal sebagai *modularity* dan merupakan inti dari penulisan kode yang bersih dan dapat digunakan kembali.

Selain logika, program juga hampir selalu berinteraksi dengan data teks. Dalam Java, manipulasi teks tidak ditangani oleh tipe data primitif, melainkan oleh sebuah *mo* khusus bernama *String*. Berbeda dengan tipe data primitif, *String* adalah sebuah objek yang memiliki serangkaian *method* bawaan yang kuat untuk melakukan berbagai operasi, seperti pencarian, pemotongan, perbandingan, dan konversi format. Praktikum ini berfokus pada dua konsep fundamental tersebut: bagaimana membuat dan menggunakan *method* untuk membangun program yang terstruktur, serta bagaimana memanfaatkan *method-method* bawaan dari *class String* untuk mengelola dan memanipulasi data teks secara efektif.

1.2 Tujuan Praktikum

Pelaksanaan praktikum ini memiliki beberapa tujuan sebagai berikut:

1) Memahami konsep dasar, sintaksis, dan mekanisme pemanggilan *method* dalam bahasa Java, baik *method* yang mengembalikan nilai maupun yang tidak.
2) Membedakan dan mengimplementasikan *static method* dan *non-static method*.

3) Memahami bahwa *String* di Java adalah sebuah objek yang memiliki *method* bawaan.
4) Menerapkan berbagai *method* bawaan *class String* untuk manipulasi teks, seperti length(), toUpperCase(), toLowerCase(), indexOf(), concat(), startsWith(), dan contains().
5) Mengintegrasikan penggunaan *method* buatan sendiri dengan *method String* untuk menyelesaikan studi kasus sederhana.

1.3 Manfaat Praktikum

Manfaat yang diharapkan dari praktikum ini adalah:

1) Meningkatkan kemampuan dalam merancang program Java yang lebih terstruktur, efisien, dan mudah dikelola dengan memecah logika ke dalam *method-method* fungsional.
2) Mampu melakukan berbagai operasi dan manipulasi data teks secara efektif menggunakan *method* yang tersedia pada *class String*.
3) Menjadi dasar yang kuat untuk memahami konsep pemrograman berorientasi objek yang lebih lanjut, terutama terkait enkapsulasi (getter/setter) dan perilaku objek.

BAB II

PEMBAHASAN

2.1 Dasar Teori

2.1.1 Pengertian *Method*

Sebuah *method* dideklarasikan di dalam *class* dan umumnya terdiri dari beberapa komponen utama. Deklarasi diawali dengan modifier, seperti public, private, atau *static*, yang menentukan tingkat akses dan perilakunya. Selanjutnya adalah return type, yang menunjukkan tipe data dari nilai yang akan dikembalikan oleh *method*. Jika *method* tidak mengembalikan nilai apa pun, digunakan kata kunci void.

Setelah itu, dituliskan nama *method* yang berfungsi sebagai pengenalnya. Terakhir, terdapat daftar parameter, yaitu masukan yang diterima oleh *method*, yang diletakkan di dalam tanda kurung (). Parameter ini ditulis sebagai pasangan tipe data dan nama variabel, namun jika *method* tidak menerima masukan, tanda kurung tetap ditulis dalam keadaan kosong.

2.1.2 Karakteristik *Method* Statis dan Non-Statis

Terdapat perbedaan mendasar antara *method static* dan *non-static* di Java. *Static method* adalah *method* yang menjadi milik *class*, bukan objek individual. Karena sifatnya ini, kita tidak perlu membuat instansiasi dari *class* tersebut untuk memanggilnya. Pemanggilan dapat dilakukan secara langsung menggunakan nama *class* diikuti tanda titik dan nama *method*-nya. Namun, *static method* memiliki batasan, yakni hanya dapat mengakses anggota *static* lain dari *class*-nya. Sebaliknya, *non-static method* adalah *method* yang menjadi milik objek. Untuk memanggilnya, kita wajib membuat objek dari *class* tersebut terlebih dahulu menggunakan kata kunci *new*. *Method non-static* lebih fleksibel karena dapat mengakses anggota *static* maupun *non-static* lain di dalam *class*-nya.

2.1.3 Pengertian *String*

Dalam Java, *String* bukanlah tipe data primitif seperti int atau double. String adalah sebuah *class* yang merepresentasikan urutan

karakter. Ketika kita membuat sebuah variabel String, kita sebenarnya sedang membuat sebuah objek dari *class String*.

Sebagai objek, *String* memiliki *method* dan properti. Salah satu karakteristik uatama *String* di Java adalah *immutable* atau tidak dapat diubah. Artinya, sekali sebuah objek *String* dibuat, nilainya tidak dapat diubah. Setiap operasi yang terlihat seperti memodifikasi *String* seperti toUpperCase() sebenarnya sedang membuat objek *String* baru di memori.

2.1.4 Operasi dan *Method* pada *String*

Karena *String* adalah *class*, ia menyediakan banyak *method* yang berguna untuk manipulasi teks. Beberapa di antaranya yang relevan dengan praktikum adalah:

- Length(): Mengembalikan jumlah karakter dari *String* sebagai int.
- toUpperCase(): Mengembalikan *String* baru dalam format huruf kapital
- toLowerCase(): Mengembalikan *String* baru dalam format huruf kecil.
- indexOf(String str): Mengembalikan indeks pertama dari karakter atau *String* yang mengembalikan.
- concat(String str): Menggabungkan *String* lain ke akhir *String* saat ini. Mirip dengan operator.
- startsWith(String prefix): Memeriksa apakah *String* dimulai dengan awalan.
- contains(CharSequence s): Memeriksa apakah *String* mengandung urutan karakter yang ditentukan.
- equals(On an0object): Membandingkan isi dari dua *String*. Ini adalah cara yang benar untuk membandingkan kesamaan *String*, berbeda dengan operator == yang membandingkan referensi objek.

2.2 Langkah Praktikum

Langkah-langkah yang dilakukan dalam pelaksanaan praktikum ini adalah sebagai berikut:

1) Menyiapkan perangkat yang telah dilengkapi dengan Java Development Kit (JDK) dan sebuah Integrated Development Environment (IDE) seperti Eclipse.
2) Membuka IDE dan membuat sebuah project Java baru.
3) Di dalam project, membuat beberapa *class* Java baru sesuai dengan nama berkas kode program yang akan diuji (misalnya, String1_2511532023.java, Mahasiswa_2511532023.java, dst.).
4) Menuliskan kode program yang telah disediakan pada modul praktikum.
5) Melakukan kompilasi (*compile*) dan eksekusi (*run*) pada setiap *class* program yang memiliki *method* main.
6) Mengamati dan mencatat keluaran (*output*) yang dihasilkan oleh setiap program.
7) Menganalisis alur eksekusi kode, membandingkan *output* yang dihasilkan dengan teori, dan memahami bagaimana setiap *method* berkontribusi pada hasil akhir.
8) Menyusun laporan praktikum sesuai dengan format yang telah ditetapkan.

2.3 Kode Program

2.3.1 String1_2511532023

```java
package pekan7_2511532023;

public class String1_2511532023 {
    public static void main(String[] args) {
        String salam = "Assalamualaikum";
        System.out.println("panjang salam adalah: " + salam.length());
        System.out.println(salam.toUpperCase()); // Outputs "ASSALAMUALAIKUM"
        System.out.println(salam.toLowerCase()); // Outputs "assalamualaikum"
        System.out.println(salam.indexOf("salam")); // Outputs 2
    }
}
```

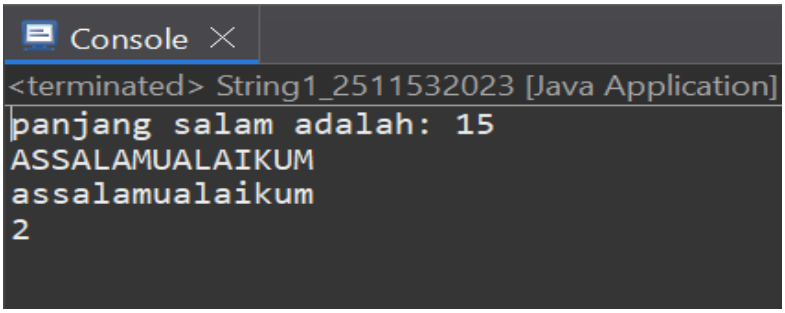
Kode Program 2.1

Program ini mendemonstrasikan beberapa *method* dasar yang dimiliki oleh objek *String*.

Struktur Pemrograman:

1) *String* salam = "Assalamualaikum";: Mendeklarasikan sebuah objek *String* bernama salam dan menginisialisasinya dengan nilai "Assalamualaikum".
2) System.out.println("panjang salam adalah: " + salam.length());: Memanggil *method* length() dari objek salam. *Method* ini mengembalikan jumlah karakter dalam *String* (yaitu 15) sebagai tipe int. Hasilnya kemudian digabungkan (konkatenasi) dengan String "panjang salam adalah: ".
3) System.out.println(salam.toUpperCase());: Memanggil *method* toUpperCase() dari objek salam. *Method* ini mengembalikan sebuah objek *String* baru yang berisi versi huruf kapital dari *String* aslinya ("ASSALAMUALAIKUM").
4) System.out.println(salam.toLowerCase());: Serupa dengan toUpperCase(), *method* ini mengembalikan objek *String* baru dalam versi huruf kecil ("assalamualaikum").
5) System.out.println(salam.indexOf("salam"));: Memanggil *method* indexOf() untuk mencari sub*string* "salam" di dalam *String* "Assalamualaikum". *Method* ini bersifat case-sensitive. Pencarian menemukan "salam" dimulai dari indeks ke-2 (indeks dimulai dari 0). Jika "salam" diubah menjadi "Salam" (dengan 'S' kapital), hasilnya akan -1 (tidak ditemukan).

   Output:



Gambar 2.1

2.3.2 String2_2511532023

```java
package pekan7_2511532023;

public class String2_2511532023 {
    public static void main(String[] args) {
        String firstName = "Syifa";
        String lastName = "Muhassanah";
        String txtl = "Dosen \"intelektual\" kampus";
        System.out.println("Nama Lengkap: "+firstName + " " + lastName);
        System.out.println("Nama Lengkap: "+firstName.concat (lastName));
        System.out.println(txtl);
        int x = 10;
        int y = 20;
        int z = x + y;
        System.out.println("x + y = "+z);
        String a = "10";
        String b = "20";
        String c = a + b;
        System.out.println("String a + string b = "+c);
        String v = a + y;
        System.out.println("String a + integer y = "+v);
    }
}
```

Kode Program 2.2

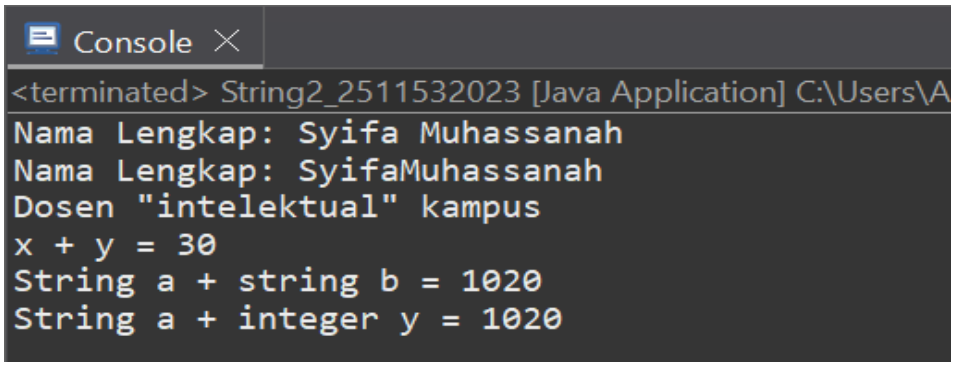
Program ini menunjukkan perbedaan penting antara operator + pada tipe numerik dan tipe *String*, serta penggunaan *method* .concat()

Struktur Program:

1) System.out.println("Nama Lengkap: "+firstName + " " + lastName);: Menggunakan operator + untuk konkatenasi (penggabungan) *String*. firstName, spasi " ", dan lastName digabung menjadi satu *String* "Syifa Muhassanah".
2) System.out.println("Nama Lengkap: "+firstName.concat (lastName));: Menggunakan *method* concat() untuk menggabungkan lastName ke firstName. Perlu dicatat bahwa concat() tidak secara otomatis menambahkan spasi, sehingga hasilnya adalah "SyifaMuhassanah".
3) *String* txtl = "Dosen \"intelektual\" kampus";: Mendemonstrasikan penggunaan escape character \ (backslash) untuk memasukkan karakter spesial seperti tanda kutip ganda (") ke dalam sebuah *String* literal.
4) int z = x + y;: Operator + di antara dua variabel int (10 dan 20) berlaku sebagai operator aritmatika (penjumlahan). Hasilnya adalah 30 (tipe int).

5) *String* c = a + b;: Operator + yang melibatkan setidaknya satu *String* akan selalu berlaku sebagai operator konkatenasi. Di sini, String "10" dan *String* "20" digabung menjadi String "1020".
6) *String* v = a + y;: Ini adalah konkatenasi tipe campuran. Karena a adalah *String*, Java secara otomatis mengkonversi int y (nilai 20) menjadi *String* "20", lalu menggabungkannya. Hasilnya adalah *String* "1020".

Output:



Gambar 2.2

2.3.3 BilanganPrima_2511532023

```java
package pekan7_2511532023;

import java.util.Scanner;

public class BilanganPrima_2511532023 {
    public static boolean isPrime(int n) {
        int factors = 0;
        for (int i = 1; i <= n; i++) {
            if (n % i == 0) {
                factors++;
            }
        }
        return (factors == 2);

    }
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Input nilai n = ");
        int a= input.nextInt();
        if (isPrime(a)) {
            System.out.println(a+" bilangan prima");
        } else {
            System.out.println(a+" bukan bilangan prima");
        }
    }
}
```

Kode Program 2.3

Program ini mendemonstrasikan pembuatan dan penggunaan *method static* buatan sendiri.

Struktur Program:

1) public static boolean isPrime(int n): Ini adalah deklarasi *method static*.
   - public static: Modifier yang berarti *method* ini dapat diakses dari mana saja dan merupakan milik *class* (tidak perlu objek).
   - boolean: Tipe kembalian. *Method* ini akan mengembalikan nilai true atau false.
   - isPrime: Nama method.
   - (int n): Parameter masukan, yaitu sebuah bilangan bulat n yang akan diperiksa.
2) Di dalam isPrime(n): Logika untuk menghitung jumlah faktor dari n. Sebuah bilangan prima adalah bilangan yang memiliki tepat dua faktor (1 dan dirinya sendiri).
3) return (factors == 2);: *Method* mengembalikan hasil dari evaluasi ekspresi factors == 2. Jika factors adalah 2, ia mengembalikan true, jika tidak, ia mengembalikan false.
4) public static void main(String[] args): *Method* utama, titik masuk program.
5) Scanner input = ...: Membuat objek Scanner untuk menerima masukan dari pengguna.
6) int a = input.nextInt();: Membaca masukan integer dari pengguna dan menyimpannya di variabel a.
7) if (isPrime(a)): Ini adalah pemanggilan *method*. *Method* main memanggil *method* isPrime dan mengirimkan nilai a sebagai argumen. Karena kedua *method* berada di *class* yang sama dan keduanya static, isPrime dapat dipanggil langsung.
8) Program kemudian mencetak hasil "bilangan prima" atau "bukan bilangan prima" berdasarkan nilai boolean yang dikembalikan oleh *method* isPrime.

2.3.4 Mahasiswa_2511532023, PanggilMahasiswa_2511532023, dan PanggilMahasiswa2_2511532023

```java
package pekan7_2511532023;

public class Mahasiswa_2511532023 {
    // variabel global
    private int nim;
    private String nama, nim2;
    // membuat mutator (setter)
    public void setNim (int nim) {
        this.nim=nim;
    }
    public void setNim2 (String nim2) {
        this.nim2=nim2;
    }
    public void setNama (String nama) {
        this.nama=nama;
    }
    // membuat accessor (getter)
    public int getNim() {
        return nim;
    }
    public String getNim2() {
        return nim2;
    }
    public String getNama() {
        return nama;
    }
    // metode lain
    public void Cetak() {
        System.out.println("Nim : "+nim);
        System.out.println("Nama : "+nama);
    }
    public void Cetak2() {
        System.out.println("Nim : "+nim2);
        System.out.println("Nama : "+nama);
    }
}
```

Kode Program 2.4

```java
package pekan7_2511532023;

public class PanggilMahasiswa_2511532023 {
    public static void main(String[] args) {
        Mahasiswa_2511532023 a= new Mahasiswa_2511532023();
        a.setNim(23532);
        a.setNama("Rahmat");
        System.out.println(a.getNim());
        System.out.println(a.getNama());
        a.Cetak();
    }
}
```

Kode Program 2.5

```java
package pekan7_2511532023;

import java.util.Scanner;

public class PanggilMahasiswa2_2511532023 {
    public static void main(String[] args) {
        Scanner input= new Scanner(System.in);
        System.out.print("NIM: ");
        String x= input.nextLine();
        System.out.print("Nama: ");
        String y= input.nextLine();
        Mahasiswa_2511532023 a= new Mahasiswa_2511532023();
        a.setNim2(x);
        a.setNama(y);
        if (x.startsWith("25")) {
            System.out.println(y + " anda angkatan 2025");
        }
        if (x.contains("1153")) {
            System.out.println("Anda Mahasiswa Informatika");
        }
        a.Cetak2();
        input.close();
    }

}
```

Kode Program 2.6

Program ini terdiri dari tiga berkas yang saling terkait: satu *class* yang mendefinisikan *method non-static*, dan dua *class* pemanggil yang menggunakan *method* tersebut.

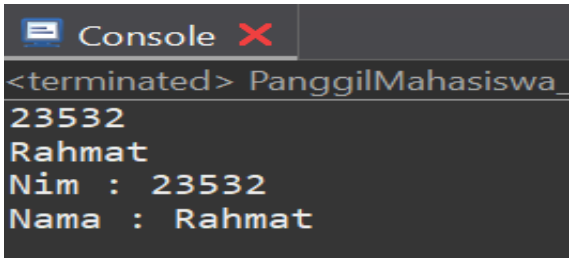
Struktur Program:

1) Mahasiswa_2511532023.java:
    a. *Class* ini bertindak sebagai blueprint (cetakan) untuk objek Mahasiswa.
    b. private int nim; ...: Mendeklarasikan fields (variabel) sebagai private. Ini berarti variabel-variabel ini tidak dapat diakses langsung dari luar *class*.
    c. public void setNim(int nim): Ini adalah *method* setter *non-static*. Kata kunci public membuatnya bisa diakses dari luar *class*. void berarti ia tidak mengembalikan nilai. Fungsinya adalah untuk mengatur nilai field private nim.
    d. this.nim = nim;: Kata kunci this merujuk pada objek saat ini. Ini digunakan untuk membedakan antara field nim (milik *class*) dan parameter nim (milik *method*).

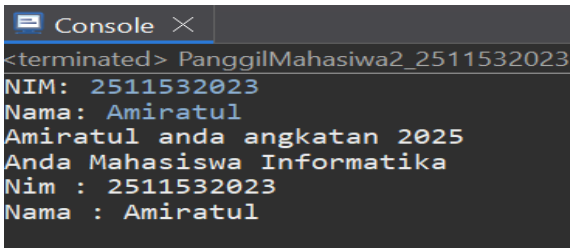
e. public int getNim(): Ini adalah *method* getter *non-static*. Fungsinya untuk mengambil (mengembalikan) nilai dari field private nim.
f. Semua *method* di *class* ini (setNim, getNim, Cetak, dll.) adalah *non-static*, yang berarti mereka hanya dapat dipanggil setelah sebuah objek Mahasiswa dibuat.

2) PanggilMahasiswa_2511532023.java:
   a. Mahasiswa_2511532023 a = new Mahasiswa_2511532023();: Ini adalah instansiasi. Kode ini membuat objek baru (a) dari blueprint Mahasiswa_2511532023.
   b. a.setNim(23532);: Ini adalah pemanggilan *method non-static*. *Method* setNim dipanggil pada objek a. Nilai 23532 disimpan ke dalam field nim milik objek a.
   c. a.getNim(): Memanggil *method* getNim pada objek a untuk mengambil nilai yang telah disimpan.
   d. a.Cetak(): Memanggil *method* Cetak pada objek a.
3) PanggilMahasiwa2_2511532023.java:
   a. Program ini mirip dengan pemanggil 1, namun menggunakan Scanner untuk mendapatkan data dari pengguna.
   b. *String* x = input.nextLine();: Membaca NIM sebagai *String*.
   c. a.setNim2(x);: Memanggil setter untuk menyimpan NIM String.
   d. if (x.startsWith("25")): Program ini menggunakan *method* bawaan *String* (startsWith) pada variabel x (NIM yang diinput) untuk melakukan pengecekan logika.
   e. if (x.contains("1153")): Menggunakan *method* contains() dari *class* untuk logika serupa.
   f. a.Cetak2();: Memanggil *method* Cetak2 dari objek a untuk menampilkan data.

Output:



Gambar 2.4



Gambar 2.5

BAB III

PENUTUP

3.1 Kesimpulan

Berdasarkan praktikum yang telah dilaksanakan, dapat dilerai beberapa kesimpulan. Pertama, *method* adalah blok kode fungsional yang digunakan untuk membungkus tugas spesifik, sehingga meningkatkan modularitas dan *reusability* kode. *Method* di Java dapat dibedakan menjadi *Static Method*, yang merupakan milik *class* dan dipanggil menggunakan nama *class*, serta *Non-Static Method*, yang merupakan milik objek dan dipanggil melalui instansiasi objek. Kedua, String dalam Java merupakan sebuah *class*, bukan tipe data primitif, yang memberikannya kemampuan untuk memiliki *method-method* bawaan untuk manipulasi data teks.

Operasi pada *String* ini, seperti `length()`, `toUpperCase()`, `toLowerCase()`, `indexOf()`, `startsWith()`, dan `contains()`, terbukti sangat esensial untuk validasi data, pemformatan, dan logika program. Terakhir, praktikum ini juga menunjukkan bahwa penggunaan operator `+` pada String berfungsi sebagai konkatenasi (penggabungan), yang fungsinya berbeda secara fundamental dengan operator `+` pada tipe data numerik yang berfungsi sebagai penjumlahan aritmatika.

3.2 Saran

Untuk pengembangan pemahaman lebih lanjut, disarankan untuk memperdalam tiga area utama. Pertama, sebaiknya dipelajari *method-method* bawaan String yang lebih kompleks, seperti substring(), split(), dan replace(), yang sangat berguna untuk parsing data. Kedua, penting untuk memperdalam pemahaman mengenai perbedaan fundamental perbandingan String menggunakan == (yang membandingkan alamat memori atau referensi objek) versus .equals() (yang membandingkan isi atau nilai String). Ketiga, disarankan pula untuk mengeksorasi konsep *method* lanjutan, seperti *method* overloading, yaitu membuat beberapa *method* dengan nama yang sama namun

parameter berbeda, dan constructor, yakni *method* khusus yang dipanggil saat objek dibuat.

DAFTAR PUSTAKA

[1] B. Hartono, *Pemrograman Java untuk Pemula*. Semarang: Yayasan Prima Agus Teknik, 2022.

[2] "Java String dan Operasinya: Panduan," *https://www.google.com/search?q=Sufyan97.com*, 2025. [Daring]. Tersedia pada: https://www.sufyan97.com/2025/03/java-string-dan-operasinya-panduan.html. [Diakses: 12-Nov-2025].

[3] "Bab 5 String Tutorial," Studocu. [Daring]. Tersedia pada: https://www.studocu.id/id/document/politeknik-elektronika-negeri-surabaya/teknik-elektro/bab-5-string-tutorial/34731686. [Diakses: 12-Nov-2025].

[4] "Laporan PBO Pratikum 2," *SlideShare*, 2014. [Daring]. Tersedia pada: https://www.slideshare.net/slideshow/laporan-pbo-pratikum-2/41005946. [Diakses: 12-Nov-2025].

[5] "Java String Methods," GeeksforGeeks. [Daring]. Tersedia pada: https://www.geeksforgeeks.org/java/java-string-methods/. [Diakses: 12-Nov-2025].

[6] "Method String Pada Pemrograman Java," *Ilmu-detil.blogspot.com*, 2018. [Daring]. Tersedia pada: https://ilmu-detil.blogspot.com/2018/01/method-string-pada-pemrograman-java.html. [Diakses: 13-Nov-2025].

[7] "String Manipulation in Java," Guru99. [Daring]. Tersedia pada: https://www.guru99.com/id/string-manipulation-in-java.html. [Diakses: 13-Nov-2025].

[8] "Tutorial Belajar Java: Tipe Data String Bahasa Pemrograman Java," Duniailkom. [Daring]. Tersedia pada: https://www.duniailkom.com/tutorial-belajar-java-tipe-data-string-bahasa-pemrograman-java/. [Diakses: 13-Nov-2025].