**VERSI** 0.1

JANUARI 2025


# PEMROGRAMAN LANJUT

***MODUL 4 - API***

**DISUSUN OLEH**:

Ir. Wildan Suharso, M.Kom.

Muhammad Ega Faiz Fadlillah

M. Ramadhan Titan Dwi .C

**TIM LABORATORIUM INFORMATIKA**

**UNIVERSITAS MUHAMMADIYAH MALANG**

# PENDAHULUAN

## *TUJUAN*

1. Mahasiswa mampu memahami konsep Java API dan mengenali kegunaan serta manfaatnya.
2. Mahasiswa mampu menggunakan kelas-kelas dari Java API untuk memperluas fungsionalitas program yang dibangun.
3. Mahasiswa mampu membuat program Java dengan API

## *TARGET MODUL*

1. Mahasiswa mampu memahami API.
2. Mahasiswa mampu membuat program Java dengan API.

## *PERSIAPAN*

1. Java Development Kit
2. Text Editor / IDE (**Preferably** Intellij IDEA, Visual Studio Code, Netbeans, etc).

## *KEYWORDS*

API, Java API

## *TABLE OF CONTENTS*

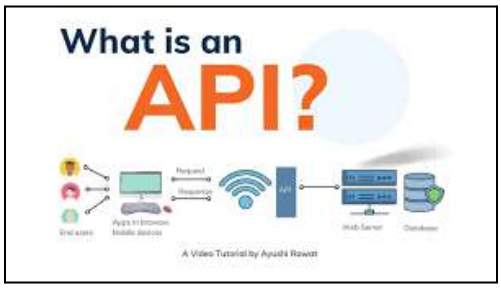

**PENDAHULUAN** ..... **2**
- TUJUAN ..... 2
- TARGET MODUL ..... 2
- PERSIAPAN ..... 2
- KEYWORDS ..... 2
- TABLE OF CONTENTS ..... 2

TEORI ..... 4
- A. APA ITU JAVA API ..... 4
- B. MEMAHAMAI API DI JAVA ..... 5
- C. SIAPA YANG MENGGUNAKAN JAVA API ..... 6
- D. MENGAPA KITA BUTUH API DI JAVA ..... 7
- E. KEUNTUNGAN DAN KERUGIAN PENGGUNAAN API ..... 8
- F. BAGIAN UTAMA JAVA API ..... 9
- G. FITUR JAVA API ..... 9
- H. CARA KERJA JAVA API ..... 10
- I. DASAR-DASAR API ..... 10
- J. TIPE JAVA API ..... 11

K. CONTOH JAVA API....12
CONTOH IMPLEMENTASI....23
REFRENSI....25
**CODELAB....26**
CODELAB 1....26
CODELAB 2....27
**TUGAS....28**
TUGAS 1....28
TUGAS 2....29
TUGAS 3....29
**KRITERIA & DETAIL PENILAIAN....30**

## TEORI

### A. APA ITU JAVA API



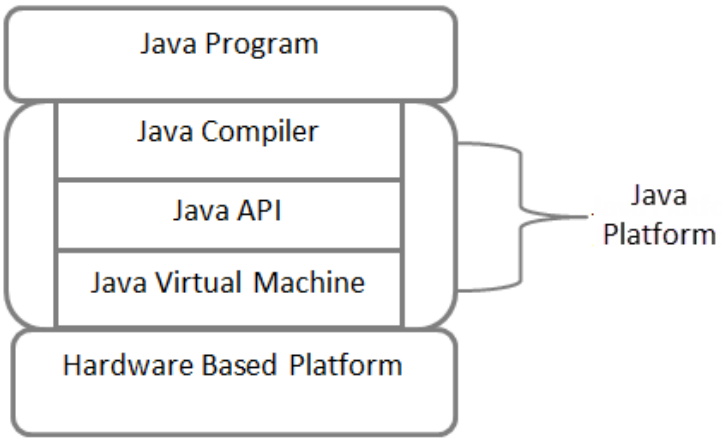

API (Application Programming Interface) adalah sebuah perantara atau penghubung dalam menyatukan data dari dua aplikasi berbeda pada waktu yang bersamaan. Dalam Java, API merupakan bagian penting yang sudah termasuk di dalam JDK (Java Development Kit). API di Java terdiri atas sekumpulan class, interface, dan library yang bisa langsung dipakai. Dengan adanya API ini, programmer dapat mengintegrasikan aplikasi atau website, sekaligus menyajikan informasi secara real-time.

Secara garis besar, API adalah antarmuka perangkat lunak yang memungkinkan dua aplikasi berinteraksi dan bertukar data tanpa harus melibatkan pengguna secara langsung. API mencakup serangkaian program dan operasi yang mendukung komunikasi antara dua perangkat lunak. API dapat dipandang sebagai aturan komunikasi dan fungsi bawaan yang memungkinkan aplikasi saling bekerja sama. Dengan memanfaatkan API, pengembang bisa membangun aplikasi lebih cepat karena tinggal menggunakan fungsi yang sudah tersedia tanpa menulis ulang kode yang rumit.

Selain itu, API sering dipakai dalam pengembangan aplikasi web, di mana aplikasi berbasis web maupun mobile harus berkomunikasi dengan server backend untuk mengirim dan menerima data. API juga punya peranan besar dalam ekosistem pengembangan modern, sebab bisa digunakan untuk terhubung dengan layanan pihak ketiga seperti sistem pembayaran, media sosial, hingga analitik. Hal ini membuat aplikasi menjadi lebih fungsional dan kaya fitur.

API bukan hanya mempermudah pengembangan perangkat lunak agar lebih cepat dan konsisten, tapi juga mendorong kolaborasi serta interoperabilitas antara berbagai sistem. Dengan API, pengembang bisa memanfaatkan layanan pihak ketiga, menghemat waktu dalam proses coding, serta memastikan aplikasi yang dibuat bisa berinteraksi dengan ekosistem perangkat lunak yang lebih luas.

## B. MEMAHAMAI API DI JAVA




**Java Development Kit (JDK)** terdiri dari 3 komponen utama, yaitu :

1. **Java Compiler**

   Java Compiler adalah sebuah program yang berfungsi untuk menerjemahkan kode sumber Java yang ditulis oleh programmer ke dalam bentuk bytecode (.class file). Bytecode adalah kode perantara yang dapat dijalankan oleh Java Virtual Machine (JVM) di berbagai sistem operasi. Proses kompilasi ini penting karena komputer tidak bisa langsung memahami bahasa pemrograman tingkat tinggi seperti Java, sehingga perlu diubah menjadi bahasa yang lebih rendah agar dapat dieksekusi. Compiler juga bertugas melakukan pengecekan kesalahan (error detection), seperti kesalahan sintaks, dan hanya akan menghasilkan bytecode jika semua error telah diperbaiki. Hasil kompilasi berupa bytecode ini kemudian akan dijalankan, diinterpretasikan, atau dioptimalkan oleh JVM menggunakan Just-In-Time (JIT) Compiler agar program berjalan lebih cepat di berbagai arsitektur perangkat keras.

2. **Java API (Application Programming Interface)**

   Java API yang terdapat dalam JDK menyediakan berbagai elemen yang bisa digunakan dalam pemrograman Java. Banyak komponen yang sudah disediakan secara bawaan dan sering dipakai, sehingga programmer tidak perlu menulis kode dari awal. Dengan memanfaatkan Java API, kita dapat menggunakan kembali kode atau fungsi yang sudah ada, sehingga proses pengembangan aplikasi menjadi lebih cepat dan efisien.

   Java API menjadi bagian penting dalam JDK karena berisi kumpulan komponen yang telah diproduksi dan siap digunakan. Saat menulis program dengan bahasa Java, kita cukup mendeklarasikan kelas serta paket yang dibutuhkan, kemudian langsung dapat memanfaatkan fungsi dari program yang sudah tersedia. Dengan cara ini, Java API membantu mempermudah, mengefisienkan, dan mempercepat pembuatan aplikasi.

3. **Java Virtual Machine**

Java Virtual Machine (JVM) adalah sebuah mesin abstrak yang berfungsi sebagai lingkungan runtime untuk menjalankan bytecode Java yang dihasilkan oleh Java Compiler. JVM memungkinkan program Java bersifat platform-independen, karena bytecode yang sama dapat dijalankan di berbagai sistem operasi dan perangkat keras tanpa perlu diubah. Proses kerjanya meliputi pemuatan kelas melalui Class Loader, pemeriksaan keamanan, serta eksekusi bytecode yang diterjemahkan menjadi instruksi mesin. Dengan cara ini, JVM memastikan program Java dapat berjalan secara efisien, aman, dan konsisten di berbagai platform sesuai prinsip Write Once, Run Anywhere.

## *C. SIAPA YANG MENGGUNAKAN JAVA API*

Terdapat tiga jenis pengembang yang menggunakan API Java berdasarkan pekerjaan atau proyek mereka :

1. **Internal Developers**, pengembang yang bekerja di dalam suatu organisasi atau perusahaan dan memanfaatkan API Java untuk membangun aplikasi internal yang mendukung operasional bisnis. Aplikasi yang mereka kembangkan biasanya bersifat tertutup, hanya digunakan oleh karyawan atau sistem internal perusahaan, serta tidak dapat diakses oleh publik.

2. **Partner Developers**, yaitu adalah pengembang yang bekerja bersama mitra bisnis perusahaan untuk mengintegrasikan atau memperluas aplikasi perusahaan dengan layanan atau produk pihak ketiga. Mereka sering kali berasal dari vendor perangkat lunak, penyedia layanan, atau perusahaan teknologi yang berkolaborasi dengan perusahaan utama.

3. **Open Developers**, merupakan pengembang independen atau bagian dari komunitas pengembang terbuka yang membuat aplikasi maupun layanan untuk publik. Mereka biasanya mengembangkan perangkat lunak open-source, proyek berbasis komunitas, atau bahkan aplikasi komersial yang dapat diakses oleh siapa saja. Peran mereka penting dalam mendorong inovasi dan memperluas ekosistem Java ke khalayak yang lebih luas.

## D. MENGAPA KITA BUTUH API DI JAVA

Dalam bahasa pemrograman Java, API (Application Programming Interface) sangat dibutuhkan karena memberikan berbagai manfaat dan kemudahan bagi pengembang antara lain :

1. **Efisiensi**

   API memberikan efisiensi karena sudah menyediakan fungsionalitas siap pakai dalam bentuk paket, kelas, dan antarmuka yang bisa langsung digunakan. Dengan begitu, pengembang tidak perlu menulis ulang kode dari awal untuk tugas-tugas umum seperti koneksi basis data, interaksi jaringan, atau manipulasi data. Hal ini bukan hanya mempercepat proses pengembangan, tetapi juga menghemat biaya serta sumber daya. Bahkan, dengan adanya API dari pihak ketiga, perusahaan dapat melengkapi aplikasi mereka tanpa harus membangun semua fitur secara mandiri.

2. **Penyederhanaan**

   API juga berperan penting dalam menyederhanakan logika program yang kompleks dengan cara memecahnya menjadi bagian-bagian kecil yang lebih mudah dikelola. Selain itu, API memberikan metode yang sederhana untuk mengakses data sesuai kebutuhan pengguna, tanpa harus melakukan penelitian atau manipulasi tambahan. Hal ini membantu pengembang lebih fokus pada inti logika bisnis aplikasi, sementara detail teknis yang rumit dapat ditangani oleh API.

3. **Integrasi**

   Salah satu kekuatan utama API adalah kemampuannya dalam mendukung integrasi. API memungkinkan aplikasi Java berkomunikasi secara mulus dengan aplikasi atau layanan lain, sehingga dapat memperluas fungsi aplikasi. Misalnya, sebuah aplikasi pengiriman makanan bisa mengintegrasikan API Google Maps untuk menampilkan lokasi pesanan secara real-time. Dengan kemampuan ini, API menjadikan ekosistem Java lebih kaya, karena aplikasi tidak hanya berdiri sendiri, tetapi bisa terhubung dengan berbagai layanan eksternal untuk memberikan pengalaman yang lebih lengkap kepada pengguna.

## E. KEUNTUNGAN DAN KERUGIAN PENGGUNAAN API

| KEUNTUNGAN | KEKURANGAN |
|---|---|
| **Efisiensi,** Java API membantu pengembang menghemat waktu karena tidak perlu menulis kode dari awal. Dengan adanya API, fungsi-fungsi tertentu sudah tersedia dan siap digunakan. Hal ini membuat pengembangan lebih cepat dan lebih efisien | **Biaya,** walaupun penggunaan API dapat mempercepat proses pengembangan, biaya yang diperlukan kadang cukup besar. Beberapa API mungkin bersifat berbayar, terutama jika digunakan dalam skala besar atau untuk layanan premium. Selain itu, proses pengembangan, pengujian, serta pemeliharaan API juga membutuhkan tenaga ahli, sehingga menambah biaya operasional. |
| **Fleksibilitas Layanan**, API memberikan kebebasan bagi pengembang untuk membangun layanan yang sesuai dengan kebutuhan aplikasi maupun pengguna. Misalnya, sebuah aplikasi dapat dengan mudah menambahkan fitur login menggunakan akun media sosial hanya dengan memanfaatkan API yang sudah ada, tanpa perlu membangun sistem autentikasi dari nol. | **Keamanan,** setiap kali sebuah aplikasi membuka akses API, maka terbentuk jalur komunikasi baru antar sistem. Jika jalur ini tidak dijaga dengan baik, bisa menjadi celah bagi peretasan, pencurian data, atau serangan siber lainnya. Oleh karena itu, penggunaan API harus disertai dengan penerapan standar keamanan yang ketat, seperti enkripsi data dan autentikasi yang kuat, agar tidak menimbulkan risiko bagi pengguna maupun sistem. |
| **Integrasi**, salah satu keunggulan utama API adalah kemampuannya dalam menghubungkan berbagai aplikasi atau platform. Dengan API, data dapat berpindah dan digunakan secara lintas sistem. | |
| **Otomatisasi**, Java API memungkinkan banyak proses dilakukan secara otomatis oleh sistem. Misalnya, API dapat digunakan untuk memproses transaksi pembayaran, mengirim notifikasi, atau melakukan update data secara berkala tanpa harus ada campur tangan manual dari pengguna. | |

## F. BAGIAN UTAMA JAVA API

Java API terbagi menjadi 3 bagian utama yaitu :

1. **Java Micro Edition (Java ME)** adalah bagian dari Java API yang digunakan untuk merancang aplikasi pada perangkat kecil seperti telepon genggam, perangkat embedded, dan perangkat IoT. Java ME menyediakan API serta runtime environment yang dioptimalkan agar bisa berjalan pada perangkat dengan keterbatasan memori, prosesor, dan tampilan. Contoh penggunaannya misalnya aplikasi pada feature phone, perangkat sensor pintar, atau alat elektronik rumah tangga yang terhubung.

2. **Java Standard Edition (Java SE),** merupakan standar API yang digunakan untuk mengembangkan aplikasi desktop dan aplikasi dasar lainnya. API ini mendukung berbagai fitur penting seperti grafis, keamanan, konektivitas jaringan, serta akses basis data. Java SE biasanya digunakan untuk membangun aplikasi desktop, tool pengembangan, maupun aplikasi server sederhana. Bisa dibilang, Java SE adalah fondasi utama dari pemrograman Java karena menyediakan bahasa inti dan pustaka dasar yang dipakai juga oleh edisi Java lainnya.

3. **Java Enterprise Edition (Java EE)** adalah pengembangan lebih lanjut dari Java SE yang dirancang untuk aplikasi server berskala besar dan kompleks. Java EE menyediakan API tambahan untuk mendukung pengembangan aplikasi enterprise, seperti pengelolaan transaksi, keamanan tingkat lanjut, skalabilitas, serta integrasi dengan berbagai sistem dan layanan. Contoh aplikasinya adalah sistem bisnis perusahaan, aplikasi perbankan, layanan web, dan aplikasi berbasis cloud.

## G. FITUR JAVA API

1. **Java Server Pages (JSP)** : Java Server Pages adalah teknologi web berbasis Java untuk membuat halaman web dinamis. JSP memungkinkan penyisipan kode Java ke dalam HTML sehingga konten web bisa menyesuaikan kebutuhan pengguna.

2. **Java Database Connectivity (JDBC)** : JDBC adalah API Java yang digunakan untuk mengakses database dengan bahasa SQL. Dengan JDBC, aplikasi dapat berkomunikasi dengan berbagai DBMS seperti MySQL atau Oracle.

3. **Java Card** : Java Card adalah platform dari Sun Microsystems untuk menjalankan aplikasi Java pada kartu pintar (smart card). Teknologi ini digunakan di kartu SIM, kartu kredit pintar, atau kartu identitas elektronik

4. **Applet** : Applet adalah program kecil berbasis Java yang dapat dijalankan melalui halaman web. Program ini diunduh oleh browser dan berjalan dengan bantuan Java Virtual Machine (JVM). Walaupun dulu populer, Applet kini jarang dipakai karena faktor keamanan dan hadirnya teknologi web modern.

5. **Java Networking** : Java Networking adalah API untuk menghubungkan komputer melalui jaringan menggunakan protokol seperti TCP/IP. Dengan fitur ini, aplikasi Java bisa bertukar data atau membuat sistem client-server. Contoh penerapannya adalah aplikasi chatting atau layanan berbasis web.

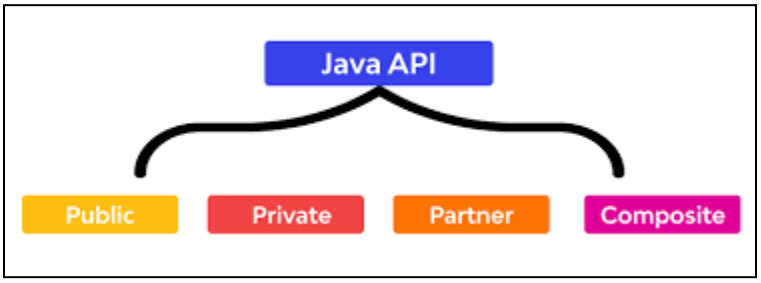
## *H. CARA KERJA JAVA API*

1. Aplikasi klien mengirimkan permintaan API, yang disebut API call, untuk memperoleh informasi. Permintaan ini berisi metode (kata kerja) permintaan, header, dan kadang-kadang juga badan pesan, yang kemudian diarahkan melalui URI (Uniform Resource Identifier) API.
2. Setelah permintaan tersebut diterima dan dianggap valid, API akan meneruskannya ke program eksternal atau server web.
3. Server kemudian mengirimkan respons berisi data yang diminta kembali ke API.
4. Data tersebut akhirnya diteruskan API kepada aplikasi klien yang awalnya melakukan permintaan.

## *I. DASAR-DASAR API*

1. Membuat API yang mampu memberikan respons OK sekaligus menyediakan string atau konten web.
2. Menangani serta memproses data dalam format JSON/XML.
3. Mendukung pengiriman data melalui formulir (form submissions).
4. Mengolah data yang diterima (baik berupa form, JSON, maupun XML), melakukan validasi, lalu menyimpannya ke dalam database.
5. Melakukan integrasi dengan API eksternal lainnya.
6. Menyimpan data pada berbagai jenis penyimpanan, baik SQL maupun NoSQL.
7. Melakukan pembaruan atau modifikasi data yang ada di dalam database.
8. Menghapus data dari database.
9. Menjamin keamanan API agar tetap aman digunakan.

## J. TIPE JAVA API




1. **Public API**

   Public API adalah API terbuka yang sudah tersedia di dalam JDK dan bisa digunakan oleh semua pengembang tanpa batasan yang ketat. Contohnya adalah paket java.lang, java.util, dan java.io yang berisi fungsi dasar seperti manipulasi string, struktur data, dan operasi input-output.

2. **Private API**

   Private API bersifat internal, artinya hanya dibuat dan digunakan dalam lingkup organisasi tertentu. API ini tidak tersedia untuk publik dan biasanya mendukung kebutuhan aplikasi atau sistem internal perusahaan.

3. **Partner API**

   Partner API dikembangkan khusus untuk bekerja sama dengan mitra bisnis atau pihak ketiga. Jenis API ini memungkinkan integrasi antar perusahaan, misalnya untuk pertukaran data atau pengembangan solusi bisnis bersama.

4. **Composite API**

   Composite API merupakan gabungan dari beberapa layanan API yang digabungkan menjadi satu titik akses tunggal. Konsep ini sering digunakan dalam arsitektur microservices agar pengembang bisa mengakses berbagai layanan sekaligus melalui satu permintaan saja.

## K. CONTOH JAVA API

Berikut adalah beberapa contoh Java API yang umum digunakan :

1. **Java API untuk Manajemen Koleksi**
   a. **java.util.List** : Merupakan struktur data yang menyimpan elemen dalam urutan tertentu, memungkinkan elemen-elemen tersebut diakses berdasarkan posisi atau indeksnya.

```java
import java.util.ArrayList;
import java.util.List;

public class AttendanceExample {
    public static void main(String[] args) {
        // Create a List to store the names of students who are present
        List<String> attendees = new ArrayList<>();

        // Add student names to the list
        attendees.add("Andi");
        attendees.add("Budi");
        attendees.add("Citra");
        attendees.add("Dewi");

        // Display the attendance list
        System.out.println("List of Present Students:");
        for (String name : attendees) {
            System.out.println("- " + name);
        }
    }
}

/*
List of Present Students:
- Andi
- Budi
- Citra
- Dewi
*/
```

   b. **java.util.set** : Merupakan koleksi yang dirancang untuk menyimpan elemen-elemen tanpa duplikasi, memastikan setiap item dalam kumpulan bersifat unik.

```java
import java.util.HashSet;
import java.util.Set;

public class AttendanceSetExample {
    public static void main(String[] args) {
        // Create a Set to store unique student names
        Set<String> attendees = new HashSet<>();

        // Add student names to the attendance list
        attendees.add("Andi");
        attendees.add("Budi");
        attendees.add("Citra");
        attendees.add("Dewi");

        // Duplicate name will be ignored
        attendees.add("Andi");

        // Display the list of present students
        System.out.println("List of Present Students:");
        for (String name : attendees) {
            System.out.println("- " + name);
        }
    }
}

/*
List of Present Students:
- Dewi
- Budi
- Citra
- Andi
*/
```

c. **java.util.Map** : Digunakan untuk menyimpan data dalam bentuk pasangan kunci-nilai.

```java
import java.util.HashMap;
import java.util.Map;

public class MapExample {
    public static void main(String[] args) {
        // Create a Map with String as key and Integer as value
        Map<String, Integer> vegetableQuantities = new HashMap<>();

        // Add elements to the Map
        vegetableQuantities.put("Carrot", 12);
        vegetableQuantities.put("Broccoli", 8);
        vegetableQuantities.put("Spinach", 20);
        vegetableQuantities.put("Tomato", 15);

        // Display all elements in the Map
        System.out.println("Vegetable Quantities:");
        for (Map.Entry<String, Integer> entry : vegetableQuantities.entrySet()) {
            System.out.println(entry.getKey() + " : " + entry.getValue());
        }
    }
}

/*
Vegetable Quantities:
Carrot : 12
Broccoli : 8
Spinach : 20
Tomato : 15
*/
```

2. **Java API untuk Manajemen I/O (Input / Output)**
    a. **java.io.InputStream** : Kelas yang digunakan untuk membaca data mentah (byte) dari suatu sumber, misalnya file atau jaringan.

```java
package Modul4.Example.IO;

import java.io.FileInputStream;
import java.io.InputStream;
import java.io.IOException;

public class InputStreamExample {
    public static void main(String[] args) {
        // Base path and file name
        String basePath = "src/Modul4/Example/IO/";
        String fileName = "data.txt";

        // Combine base path and file name
        String fullPath = basePath + fileName;

        // Create InputStream to read the file
        try (InputStream inputStream = new FileInputStream(fullPath)) {
            int data;
            System.out.println("File content:");

            // Read data from the file
            while ((data = inputStream.read()) != -1) {
                // Display the character read
                System.out.print((char) data);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

/*
        Expected output if the file data.txt contains :
        Hello, World!

        The output will be :
        File content:
        Hello, World!
*/
```

b. **java.io.OutputStream** : Kelas yang digunakan untuk menulis data mentah (byte) ke suatu tujuan, misalnya file atau jaringan.

```java
package Modul4.Example.IO;

import java.io.FileOutputStream;
import java.io.OutputStream;
import java.io.IOException;

public class OutputStreamExample {
    public static void main(String[] args) {
        // Base path and file name
        String basePath = "src/Modul4/Example/IO/";
        String fileName = "data.txt";

        // Combine base path and file name
        String fullPath = basePath + fileName;

        String data = "Hello, World!";

        try (OutputStream outputStream = new FileOutputStream(fullPath)) {
            outputStream.write(data.getBytes());
            System.out.println("Data has been successfully written to file: " + fullPath);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

/*
        The output will be :
        Data has been successfully written to file: src/Modul4/Example/IO/data.txt

        The contents of the file data.txt :
        Hello, World!
*/
```

c. **java.io.File** : Kelas yang merepresentasikan file atau folder di komputer, sehingga program bisa memeriksa, membuat, atau mengubahnya

```java
package Modul4.Example.IO;

import java.io.File;

public class FileInfoExample {
    public static void main(String[] args) {
        // Base path and file name
        String basePath = "src/Modul4/Example/IO/";
        String fileName = "data.txt";

        // Combine base path and file name
        String fullPath = basePath + fileName;

        // Create a File object representing the file
        File file = new File(fullPath);

        // Display information about the file
        if (file.exists()) {
            System.out.println("File name: " + file.getName());
            System.out.println("Absolute path: " + file.getAbsolutePath());
            System.out.println("Writable: " + file.canWrite());
            System.out.println("Readable: " + file.canRead());
            System.out.println("File size in bytes: " + file.length());
        } else {
            System.out.println("File does not exist.");
        }
    }
}

/*
        The program output (assuming data.txt exists):
        File name: data.txt
        Absolute path: C:\Users\Titan\Documents\strukdat-modul\data.txt
        Writable: true
        Readable: true
        File size in bytes: 13

        If data.txt is missing, the program will output :
        File does not exist.
*/
```

3. **Java API untuk Manajemen Jaringan**
    a. **java.net.URL** : Digunakan untuk merepresentasikan alamat sebuah sumber daya di jaringan dan memungkinkan program membuka koneksi ke alamat tersebut

```java
package Modul4.Example.Network;

import java.net.URL;
import java.net.MalformedURLException;

public class URLExample {
    public static void main(String[] args) {
        try {
            // Create a URL object
            URL url = new URL("https://www.example.com:8080/docs/" +
                    "resource1.html?name=example#section1");

            // Display information about the URL
            System.out.println("Full URL: " + url.toString());
            System.out.println("Protocol: " + url.getProtocol());
            System.out.println("Host: " + url.getHost());
            System.out.println("Port: " + url.getPort());
            System.out.println("Path: " + url.getPath());
            System.out.println("Query: " + url.getQuery());
            System.out.println("File: " + url.getFile());
            System.out.println("Reference: " + url.getRef());
        } catch (MalformedURLException e) {
            System.out.println("The URL is not valid.");
            e.printStackTrace();
        }
    }
}

/*
Output :
Full URL: https://www.example.com:8080/docs/resource1.html?name=example#section1
Protocol: https
Host: www.example.com
Port: 8080
Path: /docs/resource1.html
Query: name=example
File: /docs/resource1.html?name=example
Reference: section1
*/
```

b. **java.net.HttpURLConnection** : Memungkinkan program membuat koneksi HTTP ke server, serta mengirim permintaan dan menerima respons dari server.

```java
import java.net.HttpURLConnection;
import java.net.URL;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;

public class HttpURLConnectionsExample {
    public static void main(String[] args) {
        try {
            // Create a URL and open a connection
            URL url = new URL("https://www.example.com");
            HttpURLConnection connection = (HttpURLConnection) url.openConnection();

            // Set the request method
            connection.setRequestMethod("GET");

            // Get the response code
            int responseCode = connection.getResponseCode();
            System.out.println("Response Code: " + responseCode);

            // Read the response from the input stream
            BufferedReader in = new BufferedReader(new InputStreamReader(connection.getInputStream()));
            String inputLine;
            StringBuffer response = new StringBuffer();

            while ((inputLine = in.readLine()) != null) {
                response.append(inputLine);
            }

            // Close the input stream
            in.close();

            // Display the response content
            System.out.println("Response Content: " + response.toString());
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

/*
        The program output depends on the server response of the accessed URI.
        For example, the output might be:

        Response Code: 200
        Response Content: <!DOCTYPE HTML><!-- NewPage --><html/ ... </html>
*/
```

4. **Java API untuk Manajemen Konkurensi**
    a. **java.util.concurrent.Executors** : Digunakan untuk menjalankan tugas-tugas pada thread terpisah secara otomatis.

```java
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class ExecutorExample {
    public static void main(String[] args) {
        // Create 3 tasks using a fixed thread pool
        ExecutorService executor = Executors.newFixedThreadPool(3);

        // Define tasks (Runnable) with different sleep durations
        Runnable task1 = () -> {
            System.out.println("Task 1 started by " + Thread.currentThread().getName());
            try {
                Thread.sleep(2000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            System.out.println("Task 1 completed by " + Thread.currentThread().getName());
        };

        Runnable task2 = () -> {
            System.out.println("Task 2 started by " + Thread.currentThread().getName());
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            System.out.println("Task 2 completed by " + Thread.currentThread().getName());
        };

        Runnable task3 = () -> {
            System.out.println("Task 3 started by " + Thread.currentThread().getName());
            try {
                Thread.sleep(3000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            System.out.println("Task 3 completed by " + Thread.currentThread().getName());
        };

        // Execute tasks using ExecutorService
        executor.execute(task1);
        executor.execute(task2);
        executor.execute(task3);

        // Shut down ExecutorService after all tasks are finished
        executor.shutdown();
    }
}

/*
        The output shows that the three tasks are executed by different
        threads in the pool :

        Task 2 started by pool-1-thread-2
        Task 1 started by pool-1-thread-1
        Task 3 started by pool-1-thread-3
        Task 2 completed by pool-1-thread-2
        Task 1 completed by pool-1-thread-1
        Task 3 completed by pool-1-thread-3
*/
```

b. **java.util.concurrent.locks.Lock** : Digunakan untuk mengatur akses ke bagian kode agar aman saat dijalankan oleh banyak thread (thread-safe).

```java
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;

public class LockExample {
    private final Lock lock = new ReentrantLock();
    private int counter = 0;

    public void increment() {
        lock.lock(); // Lock before accessing the shared resource
        try {
            counter++;
            System.out.println(Thread.currentThread().getName() +
                    " incremented counter to: " + counter);
        } finally {
            lock.unlock(); // Always unlock in the finally block
        }
    }

    public static void main(String[] args) {
        LockExample example = new LockExample();

        // Create multiple threads that will access the increment method
        Runnable task = () -> {
            for (int i = 0; i < 5; i++) {
                example.increment();
                try {
                    Thread.sleep(100); // Simulate delay
                } catch (InterruptedException e) {
                    Thread.currentThread().interrupt();
                    e.printStackTrace();
                }
            }
        };

        Thread thread1 = new Thread(task, "Thread-1");
        Thread thread2 = new Thread(task, "Thread-2");

        thread1.start();
        thread2.start();
    }
}

/*
        The output produced by this program will show that the
        counter is accessed safely by multiple threads.

        Thread-1 incremented counter to: 1
        Thread-2 incremented counter to: 2
        Thread-1 incremented counter to: 3
        Thread-2 incremented counter to: 4
        Thread-2 incremented counter to: 5
        Thread-1 incremented counter to: 6
        Thread-1 incremented counter to: 7
        Thread-2 incremented counter to: 8
        Thread-1 incremented counter to: 9
        Thread-2 incremented counter to: 10
*/
```

## 5. Java API untuk GUI (Graphical User Interface)

```java
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JButton;
import javax.swing.JOptionPane;

public class SwingExample {
    public static void main(String[] args) {
        // Create a new frame
        JFrame frame = new JFrame("Swing Example");
        frame.setSize(400, 300);

        // Set default close operation when the frame is closed
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Create a panel
        JPanel panel = new JPanel();
        frame.add(panel);

        // Place components inside the panel
        placeComponents(panel);

        // Make the frame visible
        frame.setVisible(true);
    }

    private static void placeComponents(JPanel panel) {
        // Set layout to null
        panel.setLayout(null);

        // Create a button
        JButton button = new JButton("Click Me");
        button.setBounds(150, 100, 100, 30);

        // Add event listener to the button
        button.addActionListener(e -> JOptionPane.showMessageDialog(null, "Button clicked!"));

        // Add the button to the panel
        panel.add(button);
    }
}

/*
The output of this program is a small window containing a button labeled 'Click Me'.
When the button is clicked, a pop-up appears displaying 'Button clicked!'
*/
```

Pada contoh diatas kita menerapkan beberapa Java API sekaligus yaitu :

a. **javax.swing.JButton** : Membuat tombol pada GUI.
b. **javax.swing.JFrame** : Merepresentasikan jendela aplikasi GUI.
c. **javax.swing.JOptionPane** : Menampilkan dialog standar pada GUI
d. **javax.swing.JPanel** : Mengelompokkan komponen GUI.

## 6. Java API untuk Manajemen Basis Data

```java
import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.Statement;
import java.sql.ResultSet;
import java.sql.SQLException;

class SqliteExample {
    public static void main(String[] args) {
        // Method to connect to database
        try (Connection conn = DriverManager.getConnection(
                "jdbc:sqlite:sample.db")) {
            if (conn != null) {
                System.out.println("Connected to the database");

                // Method to create table
                Statement stmt = conn.createStatement();
                String sqlCreateTable = "CREATE TABLE IF NOT EXISTS employees " + "(name TEXT NOT NULL, " + "salary REAL)";
                stmt.execute(sqlCreateTable);
                System.out.println("Table created");

                // Method to insert data
                String sqlInsert = "INSERT INTO employees " + "(name, salary) VALUES ('John Doe', 75000.0)";
                stmt.execute(sqlInsert);
                System.out.println("Data inserted");

                // Method to query data
                String sqlSelect = "SELECT * FROM employees";
                ResultSet resultSet = stmt.executeQuery(sqlSelect);

                // While loop to print data
                while (resultSet.next()) {
                    System.out.printf("Name: %s, Salary: %.2f%n", resultSet.getString("name"), resultSet.getDouble("salary"));
                }

                // Method to update data
                String sqlUpdate = "UPDATE employees SET salary = 80000.0 " + "WHERE name = 'John Doe'";
                stmt.execute(sqlUpdate);
                System.out.println("Data updated");

                // Method to delete data
                String sqlDelete = "DELETE FROM employees " + "WHERE name = 'John Doe'";
                stmt.execute(sqlDelete);
                System.out.println("Data deleted");
            }
        } catch (SQLException e) {
            System.out.println(e.getMessage());
        }
    }
}

/*
        Output :
        Connected to the database
        Table created
        Data inserted
        Name: John Doe, Salary: 75000.00
        Data updated
        Data deleted
*/
```

Pada contoh diatas kita menerapkan beberapa Java API sekaligus yaitu :

a. **java.sql.Connection** :Digunakan untuk menjalin hubungan antara aplikasi Java dan basis data.
b. **java.sql.Statement** : Berfungsi untuk menjalankan instruksi SQL
c. **java.sql.ResultSet** : Menyimpan dan mewakili data hasil dari eksekusi perintah SQL seperti SELECT.

## 7. Java API untuk Pengolahan Gambar

```java
package Modul4.Example.Image;

import java.io.File;
import java.io.IOException;
import javax.imageio.ImageIO;
import java.awt.image.BufferedImage;

public class ImageProcessingExample {
    public static void main(String[] args) {
        // Base path and file name
        String basePath = "src/Modul4/Example/Image/";
        String inputFileName = "input.png";
        String outputFileName = "output.png";

        // Combine base path and file name
        String fullPathInput = basePath + inputFileName;
        String fullPathOutput = basePath + outputFileName;

        // Read image from file
        BufferedImage inputImage = null;
        try {
            inputImage = ImageIO.read(new File(fullPathInput));
            System.out.println("Image read successfully.");
        } catch (IOException e) {
            System.out.println("Error reading image file.");
            e.printStackTrace();
        }

        // Get image size elements
        int originalWidth = inputImage.getWidth();
        int originalHeight = inputImage.getHeight();

        // Determine new size
        int scaledWidth = originalWidth / 2;
        int scaledHeight = originalHeight / 2;

        // Create a new image with resized dimensions
        BufferedImage outputImage = new BufferedImage(scaledWidth, scaledHeight, inputImage.getType());
        outputImage.getGraphics().drawImage(inputImage, 0, 0, scaledWidth, scaledHeight, null);

        // Save the resized image into a new BufferedImage
        try {
            ImageIO.write(outputImage, "jpg", new File(fullPathOutput));
            System.out.println("Image resized successfully.");
        } catch (IOException e) {
            System.out.println("Error: image saved was unsuccessful.");
            e.printStackTrace();
        }
    }
}

/*
        The output of this program if input.png is a valid image :
        Image read successfully.
        Image resized successfully.
*/
```

Pada contoh diatas kira menerapkan beberapa Java API sekaligus yaitu:

a. **java.awt.image.BufferedImage** : Digunakan untuk menyimpan representasi gambar secara digital di dalam memori komputer.
b. **javax.imageio.ImageIO**: Berfungsi untuk melakukan operasi pembacaan dan penulisan file gambar dalam berbagai format.

Contoh yang diberikan sebelumnya hanya mewakili sebagian kecil dari Java API yang umum digunakan. Masih terdapat banyak API lain dalam ekosistem Java yang dapat dimanfaatkan untuk berbagai keperluan pengembangan perangkat lunak. Untuk menjelajahi lebih banyak contoh API, kamu dapat mengunjungi tautan ini.

## *CONTOH IMPLEMENTASI*

1. **Implementasi Java API Waktu dan Tanggal**

   Program Java ini berfungsi untuk menampilkan waktu saat ini dalam dua zona waktu berbeda yaitu zona waktu default sistem dan zona waktu "Asia/Malang". Dengan menggunakan API dari paket **`java.util`** dan **`java.text`**, khususnya kelas **`Calendar`**, **`TimeZone`**, dan **`SimpleDateFormat`**, program ini memformat tanggal dan waktu dalam format yang mudah dibaca (`yyyy-MM-dd HH:mm:ss Z`).

```java
import java.util.TimeZone;
import java.util.Calendar;
import java.text.SimpleDateFormat;

public class TimeZoneExample {
    public static void main(String[] args) {
        // Format date and time
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss Z");

        // Set default time
        Calendar calendar = Calendar.getInstance();
        System.out.println("Current time in default time zone:");
        System.out.println(sdf.format(calendar.getTime()));

        // Set time zone to 'Asia/Malang'
        TimeZone timeZone = TimeZone.getTimeZone("Asia/Malang");
        calendar.setTimeZone(timeZone);

        System.out.println("\nCurrent time in Asia/Malang time zone:");
        System.out.println(sdf.format(calendar.getTime()));
    }
}
```

2. **Implementasi Java API dalam Program File Reader**

   Program Java ini bertujuan untuk menulis beberapa baris teks ke dalam sebuah file bernama output.txt, yang disimpan di direktori tertentu. Program ini menggunakan API dari paket java.io, khususnya kelas **FileWriter** dan **BufferedWriter**, untuk melakukan operasi penulisan file. FileWriter digunakan untuk membuka koneksi ke file tujuan, sementara BufferedWriter menyediakan buffer agar proses penulisan lebih cepat.

```java
package Modul4.Implementation;

import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;

public class FileWriterExample {
    public static void main(String[] args) {
        // Define base path and file name
        String basePath = "src/Modul4/Implementation/";
        String fileName = "output.txt";
        String filePath = basePath + fileName;

        // Write to file using FileWriter and BufferedWriter
        try {
            FileWriter fileWriter = new FileWriter(filePath);
            BufferedWriter bufferedWriter = new BufferedWriter(fileWriter);

            // Write several lines of text to the file
            bufferedWriter.write("This file was generated by a Java program.");
            bufferedWriter.newLine();
            bufferedWriter.write("It demonstrates how to write text using BufferedWriter.");
            bufferedWriter.newLine();
            bufferedWriter.write("Each line is written separately for clarity.");

            System.out.println("Data has been written to the file: " + filePath);

            bufferedWriter.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

3. **Implementasi Java API dalam Matematika**

Program Java ini digunakan untuk melakukan perhitungan matematika berupa logaritma dan eksponensial menggunakan kelas **Math** dari Java API. Di dalam metode main, program mendefinisikan dua nilai (10.0 dan 0.5) dan menghitung logaritma natural (basis e) dengan Math.log(), logaritma basis 10 dengan Math.log10(), serta eksponensial dengan Math.exp().

```java
public class LogExpExample {
    public static void main(String[] args) {
        // Value to be used in calculations
        double value = 10.0;

        // Calculate natural logarithm (base e)
        double logNatural = Math.log(value); // ln(10)
        System.out.println("Natural logarithm of " + value + " is: " + logNatural);

        // Calculate logarithm base 10
        double logBase10 = Math.log10(value); // log10(10)
        System.out.println("Logarithm base 10 of " + value + " is: " + logBase10);

        // Calculate exponential (e^value)
        double exponential = Math.exp(value); // e^10
        System.out.println("Exponential of " + value + " is: " + exponential);

        // Additional: logarithm and exponential of a small value
        double smallValue = 0.5;
        System.out.println("\nNatural logarithm of " + smallValue + ": " + Math.log(smallValue));
        System.out.println("Exponential of " + smallValue + ": " + Math.exp(smallValue));
    }
}
```

## REFRENSI


what-is-an-api - geeksdorgeeks

what_are_java_apis - simplilearn

what-is-api-in-java - educba

Java-compiler - theserverside

jvm - eginnovations

java-community-process-jcp - andreserr.wordpress

Java SE 17 docs api  - oracle

## CODELAB

### *CODELAB 1*

Buatlah program yang mengimplementasikan materi Java API untuk melakukan beberapa manipulasi string sesuai ketentuan berikut :

1. Mengambil kata kedua dari kalimat yang dimasukkan user.
2. Mengubah seluruh kalimat menjadi huruf kapital.
3. Mengecek apakah kalimat mengandung kata "java".
4. Menambahkan kata atau kalimat baru di akhir kalimat.
5. Membalik kalimat yang sudah diubah.

**(Clue  java.lang.String dan java.lang.StringBuilder)**

**Contoh Output :**

```
Masukkan sebuah kalimat: Aku suka java karena
Kata kedua : suka
Huruf Kapital : AKU SUKA JAVA KARENA
Apakah mengandung kata 'java'? : true
Masukkan kata/kalimat untuk ditambah di akhir: seru
Setelah menambah kata/kalimat : Aku suka java karena seru
Kalimat terbalik: ures anerak avaj akus ukA
```

## *CODELAB 2*

Buatlah sebuah program sambil mengimplementasikan Java API :

1. Meminta user untuk memasukkan nilai minimum dan maksimum, lalu menghasilkan bilangan bulat acak dalam rentang tersebut.
2. Meminta user memasukkan sebuah string, lalu memilih satu karakter acak dari string tersebut.
3. Keluar dari program dengan menampilkan pesan keluar.

**(Clue java.util.Random)**

**Contoh Output :**

```
Menu:
1. Menghasilkan Bilangan Bulat Acak
2. Mengambil Karakter Acak dari String
3. Keluar
Pilihan: 1
Masukkan nilai minimum: 1
Masukkan nilai maksimum: 10
Bilangan bulat acak antara 1 dan 10: 7

Menu:
1. Menghasilkan Bilangan Bulat Acak
2. Mengambil Karakter Acak dari String
3. Keluar
Pilihan: 2
Masukkan sebuah kata/kalimat: Saya suka belajar bahasa java
Karakter acak dari string: a

Menu:
1. Menghasilkan Bilangan Bulat Acak
2. Mengambil Karakter Acak dari String
3. Keluar
Pilihan: 3
Keluar dari program.
```

## TUGAS

### *TUGAS 1*

Seorang manajer ingin mengelola data film. Manajer tersebut membutuhkan sebuah program yang dapat menambah data film dan mengurutkan data film berdasarkan nama film (A-Z) serta tahun rilisnya (Asc). Sehingga, buatlah sebuah program dengan mengimplementasikan Java API yang sudah disediakan untuk bisa mengurutkan data film tersebut. **(Clue: java.util.ArrayList, java.util.Collections, java.util. Comparator)**.

**Contoh Output :**

```
=== Menu Manajemen Film ===
1. Tambah Film Baru
2. Urutkan Berdasarkan Nama Film (A-Z)
3. Urutkan Berdasarkan Tahun (Ascending)
4. Keluar Program
Masukkan pilihan (1-4): 1
Masukkan judul film: Bleach
Masukkan tahun rilis: 2004
Film berhasil ditambahkan.

=== Menu Manajemen Film ===
1. Tambah Film Baru
2. Urutkan Berdasarkan Nama Film (A-Z)
3. Urutkan Berdasarkan Tahun (Ascending)
4. Keluar Program
Masukkan pilihan (1-4): 2

=== Daftar Film (Urut Nama A-Z) ===
Judul: Bleach, Tahun: 2004
Judul: Naruto, Tahun: 2002
Judul: One Piece, Tahun: 1999

=== Menu Manajemen Film ===
1. Tambah Film Baru
2. Urutkan Berdasarkan Nama Film (A-Z)
3. Urutkan Berdasarkan Tahun (Ascending)
4. Keluar Program
Masukkan pilihan (1-4): 3

=== Daftar Film (Urut Tahun Ascending) ===
Judul: One Piece, Tahun: 1999
Judul: Naruto, Tahun: 2002
Judul: Bleach, Tahun: 2004

=== Menu Manajemen Film ===
1. Tambah Film Baru
2. Urutkan Berdasarkan Nama Film (A-Z)
3. Urutkan Berdasarkan Tahun (Ascending)
4. Keluar Program
Masukkan pilihan (1-4): 4
Keluar dari program.
```

## TUGAS 2

Seorang programmer ingin membuat program untuk memformat jumlah uang dan tanggal sesuai standar negara tertentu. Program akan meminta input jumlah uang dan tanggal dari pengguna, lalu menampilkan nama negara, kode mata uang, simbol mata uang, serta hasil format jumlah uang dan tanggal.
**(Clue: java.util.Locale, java.util.Currency, java.text.NumberFormat, java.text.DateFormat)**

**CATATAN: negara yang ditampilkan minimal ada 3 !**

**Contoh Output :**

```
Masukkan jumlah uang: 150500
Masukkan tanggal (dd-MM-yyyy): 10-12-2025

=== Locale: id_ID ===
Country: Indonesia
Currency Code: IDR
Currency Symbol: Rp
Formatted currency: Rp150.500,00
Formatted date: 10 Desember 2025

=== Locale: ja_JP ===
Country: 日本
Currency Code: JPY
Currency Symbol: ¥
Formatted currency: ¥150,500
Formatted date: 2025年12月10日
```

## TUGAS 3

Jelaskan kepada asisten perbedaan antara Java API dan Library dengan bahasa kamu sendiri !

## KRITERIA & DETAIL PENILAIAN

| KRITERIA PENILAIAN | POIN (TOTAL 100%) |
|---|---|
| **CODELAB 1** | **Total 15%** |
| Kesesuaian Output | 5% |
| Pemahaman Materi | 10% |
| **CODELAB 2** | **Total 15%** |
| Kesesuaian Output | 5% |
| Pemahaman Materi | 10% |
| **TUGAS 1** | **Total 25%** |
| Kesesuaian Output | 10% |
| Pemahaman Materi | 15% |
| **TUGAS 2** | **Total 25%** |
| Kesesuaian Output | 10% |
| Pemahaman Materi | 15% |
| **TUGAS 3** | **Total 20%** |
| Pemahaman Materi | 20% |