

VERSION 1.6

19 Februari 2025



# PEMROGRAMAN BERORIENTASI OBJEK

MODUL 1 – JAVA BASIC, JAVA API, GIT & GITHUB

DISUSUN OLEH:

WIRA YUDHA AJI PRATAMA

KEN ARYO BIMANTORO

DIAUDIT OLEH:

Ir. Galih Wasis Wicaksono, S.Kom, M.Cs.

PRESENTED BY: TIM LAB. IT

UNIVERSITAS MUHAMMADIYAH MALANG

## PENDAHULUAN

### TUJUAN

1. Mahasiswa memahami dasar-dasar sintaks Java (input/output, variabel, tipe data, percabangan, perulangan).
2. Mahasiswa memahami konsep dasar API dan cara penggunaannya dalam program Java.
3. Mahasiswa memahami konsep dasar version control dengan Git dan cara kerja GitHub dalam kolaborasi proyek.

### TARGET MODUL

1. Mahasiswa dapat membuat program sederhana yang menerapkan input/output, percabangan, dan API dalam Java.
2. Mahasiswa dapat melakukan commit, push, pull, dan memahami dasar-dasar branching di GitHub.

### PERSIAPAN

1. Device (Laptop/PC)
2. IDE (IntelliJ)
3. Internet
4. Web Browser
5. GitBash
6. Akun GitHub

### KEYWORDS

IDE, IntelliJ, JDK, GIT, GitBash

### TABLE OF CONTENTS

<b>PENDAHULUAN</b>	<b>1</b>
TUJUAN	1
TARGET MODUL	1
PERSIAPAN	1
KEYWORDS	1
TABLE OF CONTENTS	1
<b>JAVA BASIC</b>	<b>4</b>
TEORI	4
Apa itu Java?	4

● Pengertian Umum	4
● Perbedaan Java dengan C	4
Program yang menggunakan Java	5
Singkat Tentang PBO (Pemrograman Berorientasikan Objek)	5
MATERI	6
Tipe data (data type)	6
Input Output (I/O)	8
● Input	9
● Output	11
Condition (percabangan)	14
● Switch/Case	14
● If	15
● If/Else	16
● If/Else/If	17
● Nested If	18
Loop (perulangan)	19
● For	19
● For Each	20
● While	20
● Do While	21
PRAKTEK	22
TIPS	26
<b>JAVA API</b>	<b>27</b>
TEORI	27
Apa itu Java API?	27
MATERI	27
Bagian API dalam Java	27
PRAKTEK	28
TIPS	29
<b>GIT &amp; GITHUB</b>	<b>30</b>
TEORI	30
GIT	30
GITHUB	31
Perbedaan GIT dan GitHub	32

<b>MATERI</b>	<b>32</b>
Cara upload source code ke repository Github dengan Git	32
IntelliJIDEA Built-In Version Control	37
Integrasi Github dengan IntelliJIDEA	39
Bekerja dengan Git Repository	43
<b>TIPS</b>	<b>48</b>
<b>CODELAB &amp; TUGAS</b>	<b>49</b>
CODELAB	49
TUGAS 1	50
TUGAS 2	52
<b>PENILAIAN</b>	<b>53</b>
RUBRIK PENILAIAN	53
SKALA PENILAIAN	54
<b>SUMMARY AKHIR MODUL</b>	<b>55</b>

**JAVA BASIC****TEORI****Apa itu Java?**

- Pengertian Umum

**Java** adalah bahasa **Pemrograman berorientasi objek (PBO)** atau **Object Oriented Programming (OOP)** yang berarti java adalah **kumpulan objek**, dan objek-objek ini berkomunikasi melalui **pemanggilan metode satu sama lain**. Java terkenal dengan fleksibilitas dan popularitasnya. Bahasa ini banyak digunakan untuk membangun berbagai aplikasi, mulai dari aplikasi **desktop** dan **mobile, website**, hingga **perangkat lunak korporasi** dan **teknologi big data**. Aturan dan sintaks Java didasarkan pada bahasa **C** dan **C++**.

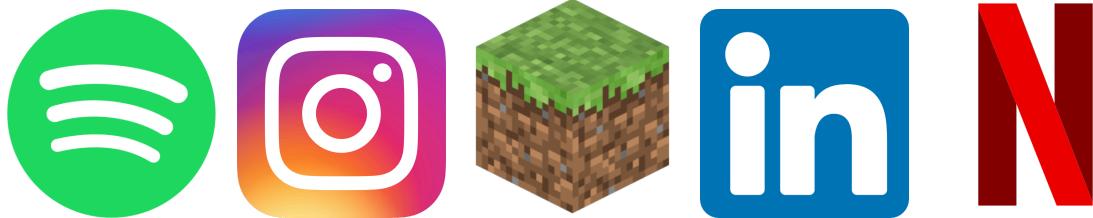
**Java** adalah bahasa pemrograman yang **case sensitive**, yang berarti bahwa huruf besar dan kecil dibedakan dalam kode. Hal ini penting untuk diingat saat menulis kode Java, karena kesalahan kecil dalam penulisan huruf kapital dapat menyebabkan **error**.

- Perbedaan Java dengan C



**Java** dan **C** adalah bahasa pemrograman dengan pendekatan yang berbeda. **C** adalah bahasa prosedural yang lebih dekat ke sistem, sering digunakan untuk **pemrograman tingkat rendah** seperti sistem operasi dan embedded systems. **Java**, di sisi lain, adalah bahasa **berbasis objek** yang dirancang agar lebih mudah dipindahkan antar platform menggunakan **Java Virtual Machine (JVM)**. Selain itu, C lebih cepat karena dikompilasi langsung ke mesin, sedangkan Java lebih fleksibel tetapi memiliki sedikit overhead karena berjalan di JVM.

## Program yang menggunakan Java

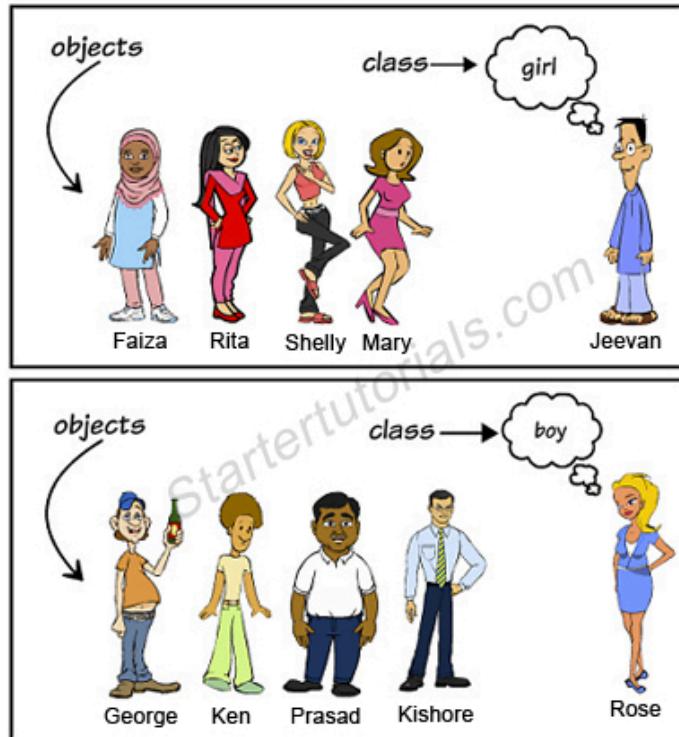


- **Android Apps** – Java adalah bahasa utama dalam pengembangan **aplikasi Android** sebelum Kotlin menjadi populer. Banyak aplikasi terkenal seperti **Instagram**, **Spotify**, dan **Twitter awalnya** dibuat menggunakan **Java**.
- **Minecraft** – Salah satu game sandbox paling populer di dunia, **Minecraft**, dikembangkan menggunakan Java. Versi Java Edition masih digunakan oleh banyak pemain di PC.
- **LinkedIn** – Sebagian besar backend LinkedIn menggunakan Java untuk menangani skalabilitas dan performa yang tinggi.
- **Netflix** – Sistem backend Netflix menggunakan Java untuk menangani streaming video dan manajemen layanan pengguna.

## Singkat Tentang PBO (Pemrograman Berorientasikan Objek)

PBO adalah konsep pemrograman yang berorientasi objek. Apa itu objek? Anda akan mengerti tentang apa itu objek selama berjalannya praktikum ini. Dalam konsep PBO, kita dapat membuat suatu objek dari class yang telah dibuat sebelumnya. **Class dapat dianggap sebagai blueprint untuk membuat suatu objek.** Atau kalian bisa menganggapnya sebagai sebuah template. Jika anda membuat objek dari kelas Mobil, maka objek tersebut memiliki pintu, dan 4 roda. Jika anda membuat objek dari kelas Motor, maka objek tersebut memiliki 2 roda, dan tidak memiliki pintu.

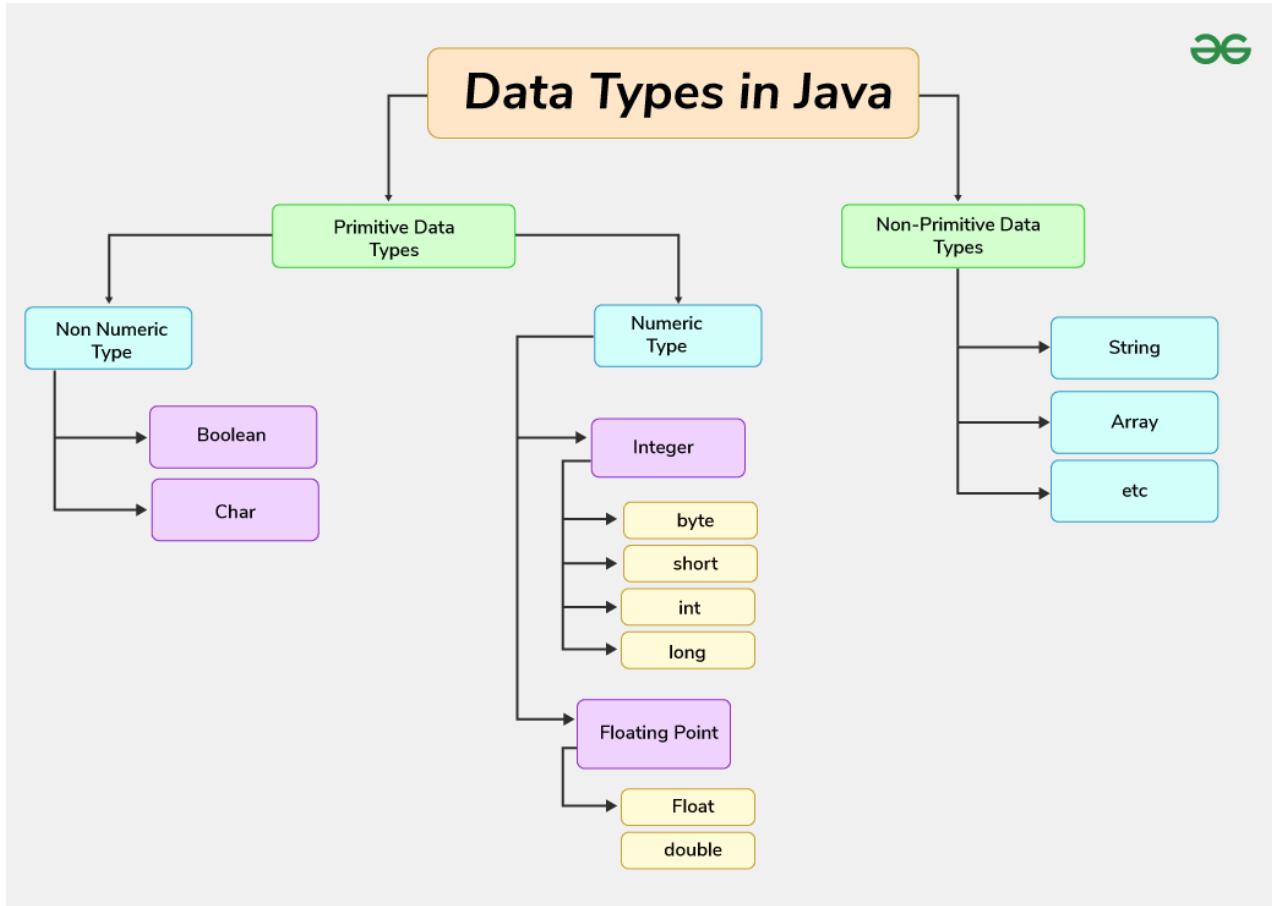
Kalian bisa membuat beberapa objek dari kelas yang sama. Contoh:



kalian membuat dua objek dari kelas mobil. Sama-sama memiliki pintu dan 4 roda. Tapi dengan warna dan kecepatan yang berbeda. Itu saja dulu. Lebih teknisnya kita akan belajar di modul berikutnya.

## MATERI

### Tipe data (data type)



**Tipe data** di java terdapat **2 jenis** yaitu tipe data **primitif** dan tipe data **non-primitif**. Tipe data **primitif** menentukan ukuran dan tipe nilai variabel tanpa memerlukan metode tambahan. Sedangkan, tipe data **non-primitif** disebut juga tipe referensi karena tipe data ini merujuk pada sebuah **objek**.

Perbedaan utama antara tipe data **primitif** dan **non-primitif** adalah:

- Jenis primitif sudah ditentukan sebelumnya di Java. Tipe non-primitif dibuat oleh programmer dan tidak bisa ditentukan oleh Java (kecuali untuk String).
- Tipe non-primitif bisa digunakan untuk memanggil method yang bisa digunakan untuk melakukan operasi tertentu, sedangkan tipe primitif tidak bisa.

- Tipe primitif dimulai dengan huruf kecil, sedangkan tipe non-primitif dimulai dengan huruf besar.
- Ukuran tipe primitif bergantung pada tipe datanya, sedangkan tipe non-primitif memiliki ukuran yang sama.

Tipe data primitif		Tipe data non primitif	
Tipe data	keterangan	Tipe data	keterangan
float	Menyimpan nilai bilangan desimal 32-bit dengan presisi 7 digit. Contoh: hargaBarang = 12.5f (float).	String	Menyimpan nilai teks. Contoh: nama = "John Doe" (String).
double	Menyimpan nilai bilangan desimal 64-bit dengan presisi 15 digit. Contoh: nilaiPi = 3.141592653589793d (double).	Array	Menyimpan kumpulan nilai dengan tipe data yang sama. Contoh: numbers = new int[]{1, 2, 3, 4, 5} (int array).
int	Menyimpan nilai bilangan bulat 32-bit dengan rentang -2,147,483,648 hingga 2,147,483,647. Contoh: jumlahPenduduk = 1000000 (int).	Objek	Menyimpan kumpulan data dan metode terkait. Contoh: person = new Person("John Doe", 18) (Person object).
char	Menyimpan nilai karakter tunggal. Contoh: hurufAwal = 'A' (char).		
boolean	Menyimpan nilai true atau false. Contoh: isLoggedIn = true (boolean).		

Contoh penggunaan:

```
● ● ●

public class modul1Practice {
    public static void main(String[] args) {
        //bilangan bulat
        int population = 1000000;

        // Desimal
        float price = 12.5f;
        double weight = 75.5d;

        // Karakter
        char letter = 'L';

        // Boolean
        boolean isLoggedIn = true;

        // String
        String name = "Wira Yudha Aji Pratama";

        // Array
        int[] numbers1 = new int[]{1, 2, 3, 4, 5};
        int[] numbers2 = {1, 2, 3, 4, 5};
        int numbers3[] = {1, 2, 3, 4, 5};

        // Objek
        Person person = new Person("Elon Musk", 53);
    }
}
```

Hal yang menjadi catatan dalam membuat array adalah tipe data yang disimpan dalam array harus sama, jika kita membuat tipe data di awal adalah tipe data int maka isi dari array juga harus int semua.

### **Input Output (I/O)**

**Input** dan **output** dalam bahasa **Java** mengacu pada bagaimana program berinteraksi dengan dunia luar (luar kode). Hal ini mengartikan bagaimana program yang kita buat bisa **menerima sebuah informasi dari pengguna (input)** atau **memberikan sebuah informasi untuk pengguna (output)**

- Input

Semua bahasa pemrograman sudah menyediakan fungsi-fungsi untuk melakukan **input** (contohnya scanf() di bahasa c). Java sendiri telah menyediakan **tiga class** untuk mengambil **input** dari **user**, yaitu:

1. Class Scanner
2. Class BufferedReader
3. Class Console

Pada materi kali ini kita hanya akan belajar tentang class **Scanner** saja. **Scanner** adalah sebuah **class** yang disediakan java untuk memungkinkan program kita menerima **input** dari pengguna melalui **keyboard**. Untuk menggunakan class **Scanner**, kita harus mengimport **class Scanner** ke dalam kode pada bagian atas kode, seperti ini:

```
Import java.util.Scanner;
```

Setelah itu kita harus membuat **objek** dari **Scanner** dengan nama variabel yang ingin kita buat di dalam method main, contoh di sini kita buat sebuah objek Scanner dengan nama **objInput**:

```
public class modulPractice {
    public static void main(String[] args) {
        Scanner objInput = new Scanner(System.in);
    }
}
```

Mari bedah satu-satu terlebih dahulu pada kode di atas, untuk kode **public class modulPractice{}** itu adalah nama class yang kita buat (lebih detail akan dipelajari di modul selanjutnya). Kode **public static void main(String[] args)** adalah kode untuk method main, semua program java akan selalu dijalankan dengan diawali pada method main ini. Kode **Scanner objInput = new Scanner(System.in);** yang dimaksud **Scanner** adalah Class yang sudah kita import sebelumnya, **objInput** adalah nama dari object yang kita buat dari class Scanner. **new Scanner()** itu artinya kita membuat sebuah objek baru dari class Scanner. Dan terakhir **System.in** yang dimaksudkan untuk mengambil sebuah input dari user melalui keyboard ketika program berjalan. Sepertinya akan susah untuk dipahami kalau cuma membaca saja. Jadi silahkan mencoba untuk memprakteknya. Jangan khawatir, kalian akan memahaminya seiring berjalannya waktu.

Disini kita akan coba untuk membuat sebuah program sederhana yang bisa menerima input dari user berupa **firstName** dan **age**. Berikut contohnya:

```
import java.util.Scanner;

public class modul1Practice {
    public static void main(String[] args){
        String firstName;
        int age;
        Scanner objInput = new Scanner(System.in);

        firstName = objInput.nextLine();
        age = objInput.nextInt();
    }
}
```

Mari bedah kode di atas:

1. variabel **firstName** di atas dideklarasikan dengan tipe data **String**
2. variabel **age** dideklarasikan dengan tipe data **int**
3. **firstName = objInput.nextLine();** adalah kode untuk menginputkan sebuah nilai String ke dalam variabel **firstName**, lihat pada kode di atas input harus melewati sebuah **objInput** lalu pemanggilan method **nextLine()** yang dimana method ini digunakan untuk menerima input user berupa string yang bisa disertai dengan spasi.
4. **age = objInput.nextInt();** adalah kode untuk menginputkan sebuah nilai int ke dalam variabel **age**, method **nextInt()** khusus hanya untuk tipe data int.

Lalu bagaimana dengan tipe data yang lain? Sebenarnya masih ada **method** lain untuk menginputkan sesuai dengan tipe data. Berikut adalah daftarnya:

1. **next()** : digunakan untuk input sebuah String tetapi hanya membaca sampai dengan spasi terakhir
2. **nextDouble()** : digunakan untuk input sebuah nilai double
3. **nextFloat()** : digunakan untuk input sebuah nilai float
4. **nextByte()** : digunakan untuk input sebuah nilai berupa byte
5. **nextBoolean()** : digunakan untuk input sebuah nilai boolean

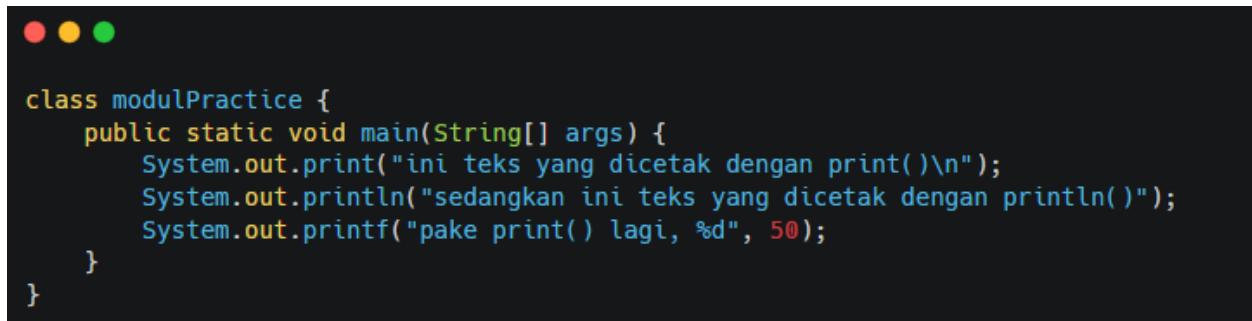
Dan masih banyak lagi **method** yang bisa digunakan dari **class Scanner**, untuk lebih lengkapnya bisa cek [disini](#).

- **Output**

Pada bahasa pemrograman java untuk melakukan output atau printf() pada bahasa c dengan syntax System.out.println() tetapi harus diperhatikan adalah case sensitive. Sebenarnya ada beberapa method yang disediakan oleh Java, yaitu:

1. System.out.print()
2. System.out.printf()
3. System.out.println()
4. System.out.format()

Method print() dan println() sama-sama digunakan untuk menampilkan sebuah teks Perbedaan utama adalah **method print()** akan menampilkan teks apa adanya. Sedangkan **println()** akan menampilkan teks dengan ditambah baris baru pada akhir teks. Mari coba untuk membuat kode seperti ini:



```
● ● ●

class modulPractice {
    public static void main(String[] args) {
        System.out.print("ini teks yang dicetak dengan print()\n");
        System.out.println("sedangkan ini teks yang dicetak dengan println()");
        System.out.printf("pake print() lagi, %d", 50);
    }
}
```

Output yang akan ditampilkan akan seperti ini:

```
ini teks yang dicetak dengan print()
sedangkan ini teks yang dicetak dengan println()
pake print() lagi, 50
Process finished with exit code 0
```

Terkadang kita tidak hanya ingin menampilkan sebuah teks, tetapi butuh sebuah teks dari variabel dan menggabungkannya dengan teks yang lain. Sebagai contoh kita punya sebuah variabel nama dan umur:

```
public static void main(String[] args) {  
    String nama = "Wira Yudha";  
    int umur = 21;  
}
```

Lalu kita ingin menampilkannya dengan method print() atau println(), maka kita hanya perlu untuk memasukkannya di dalam method. Seperti ini:

```
public class modulPractice {  
    public static void main(String[] args) {  
        String nama = "Wira Yudha";  
        int umur = 21;  
        System.out.println(nama);  
        System.out.print(umur);  
    }  
}
```

Kode di atas akan menghasilkan output:

```
Wira Yudha  
21  
Process finished with exit code 0
```

Sebenarnya kita tidak perlu untuk menggunakan dua method print() atau println(), karena kita bisa menggabungkannya dengan operator “+”. Contohnya adalah berikut:

```
System.out.println(nama + umur);
```

Atau jika ingin ada spasinya maka bisa tambahkan spasi di tengahnya menjadi seperti ini:

```
System.out.println(nama + " " + umur);
```

Method yang terakhir adalah format(), method ini digunakan untuk menggabungkan String yang lebih kompleks dan method ini hampir sama persis dengan penggunaan printf di bahasa C. Contohnya:

```

public class modulPractice {
    public static void main(String[] args) {
        String nama = "Wira Yudha";
        int umur = 21;
        System.out.format("Nama saya %s umur %d %n", nama, umur);
    }
}

```

Penjelasan kode di atas seperti ini:

- simbol %s digunakan untuk mengambil nilai dari variabel di sampingnya, %s yang artinya adalah string
- %d untuk tipe data int
- %n untuk baris baru, bisa juga menggunakan \n
- Untuk lengkapnya bisa cek [disini](#)

Ketika kode di atas dijalankan, maka output program akan seperti ini:

```

Nama saya Wira Yudha umur 21

Process finished with exit code 0

```

Berikut disertakan juga contoh untuk penggunaan gabungan dari input dan output di java:

```

● ● ●

import java.util.Scanner;

public class Main {
    public static void main(String[] args){
        String firstName;
        int age;
        Scanner objInput = new Scanner(System.in);

        System.out.print("Masukkan Nama Anda : ");
        firstName = objInput.nextLine();
        System.out.print("Masukkan Umur Anda : ");
        age = objInput.nextInt();
    }
}

```

```

        System.out.println("Nama : " + firstName);
        System.out.println("Umur : " + age);
    }
}

```

Output:

```

Masukkan Nama Anda : Wira Yudha
Masukkan Umur Anda : 21
Nama : Wira Yudha
Umur : 21

Process finished with exit code 0

```

### Condition (percabangan)

Kondisi atau percabangan di Java digunakan untuk menentukan apakah suatu blok kode akan dijalankan atau tidak. Ini didasarkan pada nilai Boolean (true atau false) dari ekspresi yang dievaluasi.

- **Switch/Case**

Percabangan ini adalah bentuk lain dari percabangan if/else/if, perbedaannya adalah pada percabangan ini menggunakan kata kunci **switch** dan **case**. Tata cara penulisan percabangan switch/case sama dengan di bahasa C. Berikut contohnya:



```

import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        String warna = "merah";

        switch (warna) {
            case "merah":
                System.out.println("Warna merah");
                break;
            case "biru":
                System.out.println("Warna biru");
        }
    }
}

```

```
        break;
    default:
        System.out.println("Warna tidak diketahui");
    }
}
}
```

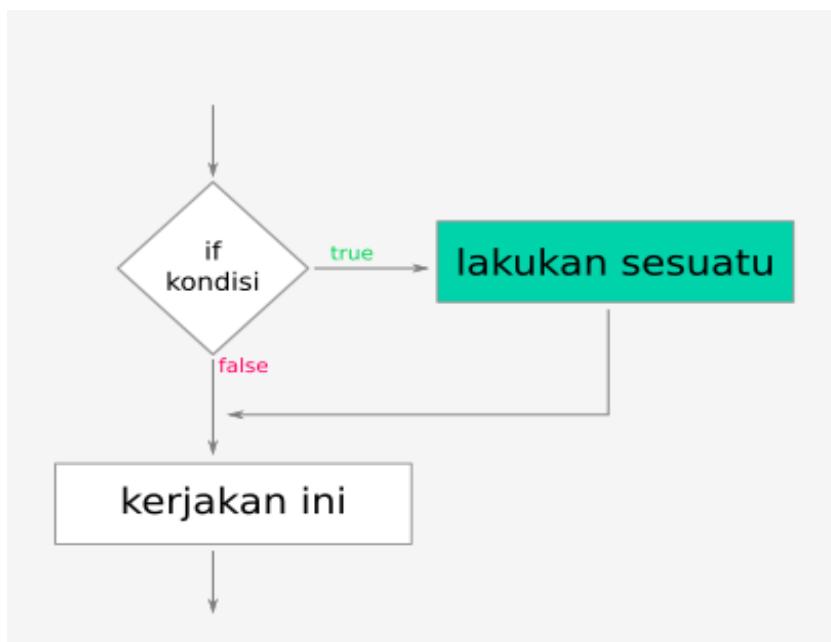
## Output:

```
Warna merah

Process finished with exit code 0
```

- If

Percabangan ini digunakan untuk mengeksekusi blok kode jika ekspresi yang dihasilkan kondisinya true. Artinya pilihan di dalam if hanya akan dikerjakan jika kondisinya benar.



Tapi kalau salah, maka tidak akan melakukan apa-apa. Contoh:

```

● ● ●

public class Main {
    public static void main(String[] args) {
        int angka = 10;

        if (angka > 5) {
            System.out.println("Angka lebih besar dari 5");
        }
    }
}

```

Output:

```

Angka lebih besar dari 5

Process finished with exit code 0

```

- **If/Else**

Percabangan ini digunakan untuk mengeksekusi blok kode yang jika hasil ekspresi pada if tidak terpenuhi maka kode else yang akan dieksekusi. Dalam artian jika IF salah, maka terdapat alternatif lain untuk menjalankan kode. Berikut contohnya:

```

public class Main {
    public static void main(String[] args) {
        int angka = 5;

        if (angka > 5) {
            System.out.println("Angka lebih besar dari 5");
        } else {
            System.out.println("Angka tidak lebih besar dari 5");
        }
    }
}

```

Output:

```

Angka tidak lebih besar dari 5

Process finished with exit code 0

```

- If/Else/If

Jika percabangan IF/ELSE hanya memiliki dua pilihan saja. Maka percabangan IF/ELSE/IF memiliki lebih dari dua pilihan. Berikut contohnya:

```
import java.util.Scanner;

public class modulPractice {
    public static void main(String[] args) {
        int nilai;
        String grade;
        Scanner scan = new Scanner(System.in);

        System.out.print("Inputkan nilai: ");
        nilai = scan.nextInt();

        if ( nilai >= 90 ) {
            grade = "A";
        } else if ( nilai >= 80 ){
            grade = "B+";
        } else if ( nilai >= 70 ){
            grade = "B";
        } else if ( nilai >= 60 ){
            grade = "C+";
        } else if ( nilai >= 50 ){
            grade = "C";
        } else if ( nilai >= 40 ){
            grade = "D";
        } else {
            grade = "E";
        }

        System.out.println("Grade: " + grade);
    }
}
```

Ketika kita coba input nilai **66**, maka outputnya akan seperti ini:

```
Inputkan nilai: 66
Grade: C+
Process finished with exit code 0
```

- Nested If

Pada semester sebelumnya kita juga sudah belajar tentang apa itu nested if di bahasa C. Pada bahasa Java juga tidak jauh beda bentuk dari nested if, langsung saja pada contoh kodennya:

```
import java.util.Scanner;
public class modulPractice {
    public static void main(String[] args) {
        int belanjaan, diskon, bayar;
        String kartu;
        Scanner scan = new Scanner(System.in);

        System.out.print("Apakah ada kartu member: ");
        kartu = scan.nextLine();
        System.out.print("Total belanjaan: ");
        belanjaan = scan.nextInt();

        if (kartu.equalsIgnoreCase("ya")) {
            if (belanjaan > 500000) {
                diskon = 50000;
            } else if (belanjaan > 100000) {
                diskon = 15000;
            } else {
                diskon = 0;
            }

        } else {
            if (belanjaan > 100000) {
                diskon = 5000;
            } else {
                diskon = 0;
            }
        }

        bayar = belanjaan - diskon;
        System.out.println("Total Bayar: Rp " + bayar);
    }
}
```

## Loop (perulangan)

**Loop (perulangan)** dalam bahasa Java memungkinkan kita untuk menjalankan sebuah blok kode **berulang kali** selama **kondisi tertentu** terpenuhi. Ini sangat berguna ketika kita ingin melakukan sebuah tugas secara **otomatis terus-menerus** dan **berurutan**. Karena di semester sebelumnya materi ini sudah dibahas, maka pembahasan kali ini hanya pengantar bagaimana cara menulis di bahasa Java.

- **For**

Format penulisan perulangan For di java seperti ini:

```
for( int hitungan = 0; hitungan <= 10; hitungan++ ){
    // blok kode yang akan diulang
}
```

variabel hitungan adalah variabel yang digunakan untuk menyimpan nilai yang berulang, jadi selama nilai hitungan lebih kecil dari 10 maka pengulangan akan terus dilakukan dan hitungan++ memiliki fungsi untuk menambah +1 nilai hitungan pada setiap perulangan. Blok kode *For* dimulai dengan tandal ‘{‘ dan diakhiri ‘}’. Berikut adalah contohnya:

```
public class Main {
    public static void main(String[] args){
        for (int i = 0; i < 10; i++) {
            System.out.println("Iterasi ke-" + i);
        }
    }
}
```

Output:

```
Iterasi ke-0
Iterasi ke-1
Iterasi ke-2
Iterasi ke-3
Iterasi ke-4
Iterasi ke-5
Iterasi ke-6
Iterasi ke-7
Iterasi ke-8
Iterasi ke-9
```

- **For Each**

Perulangan ini sebenarnya digunakan untuk menampilkan isi dari array. Secara singkat array adalah variabel yang menyimpan lebih dari satu nilai dan memiliki indeks. Untuk lebih lengkapnya nanti akan dipelajari pada modul 5. Perulangan **for each** dilakukan dengan kata kunci **For**. Berikut adalah contohnya:

```
for ( int item : dataArray ) {
    // blok kode yang diulang
}
```

Variabel item akan menyimpan nilai dari array dan kita bisa baca seperti ini: “Untuk setiap item dalam dataArray, maka lakukan perulangan”. Contoh program dengan **For Each**:

```
public class modulPractice {
    public static void main(String[] args) {
        int angka[] = {3,1,42,24,12};

        for( int x : angka ){
            System.out.print(x + " ");
        }
    }
}
```

Output:

```
3 1 42 24 12
Process finished with exit code 0
```

- **While**

While bisa diartikan selama, cara kerja perulangan ini hampir sama dengan percabangan. Ia akan melakukan perulangan selama kondisi di dalam while bernilai true. Untuk struktur penulisan perulangan while adalah seperti berikut:

```
while (condition) {
    // kode yang akan dijalankan
}
```

kondisi bisa kita isi dengan perbandingan ataupun nilai boolean, kondisi hanya memiliki nilai true dan false. Perulangan ini akan berhenti ketika kondisi bernilai false. Contoh kode perulangan while:

```
public class Main {
    public static void main(String[] args){
        int i = 0;
        while (i < 5) {
            System.out.println("Iterasi ke-" + i);
            i++;
        }
    }
}
```

```
Iterasi ke-0
Iterasi ke-1
Iterasi ke-2
Iterasi ke-3
Iterasi ke-4

Process finished with exit code 0
```

- **Do While**

Perulangan ini cara kerjanya sebenarnya sama seperti while-loop, tetapi do-while akan melakukan perulangan satu kali perulangan terlebih dahulu baru setelah itu kondisi dicek. Struktur penulisannya seperti ini:

```
do {
    // kode yang akan diulang
} while (condition);
```

Untuk contoh kode penggunaan do-while:

```
public class Main {
    public static void main(String[] args){
        int i = 0;
        do {
            System.out.println("Iterasi ke-" + i);
            i++;
        } while (i < 3);
    }
}
```

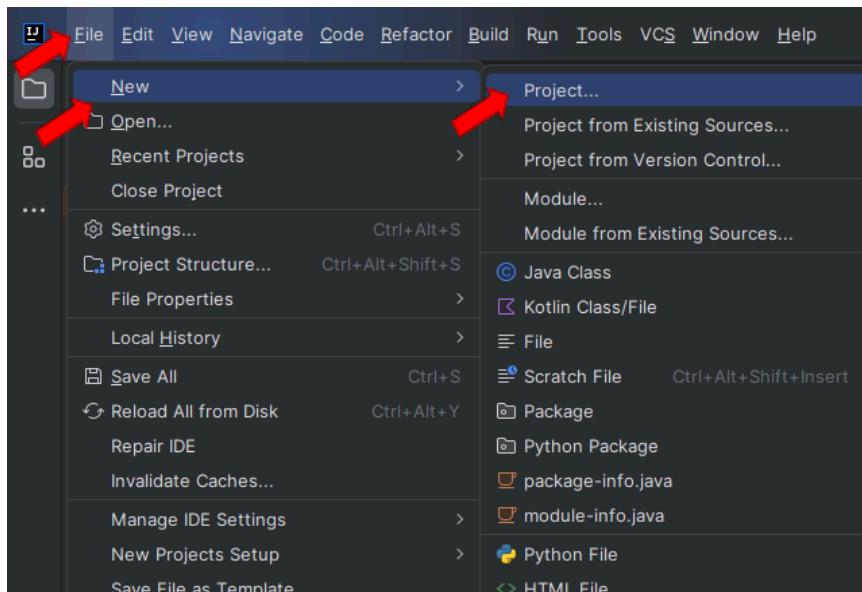
Output:

```
Iterasi ke-0
Iterasi ke-1
Iterasi ke-2

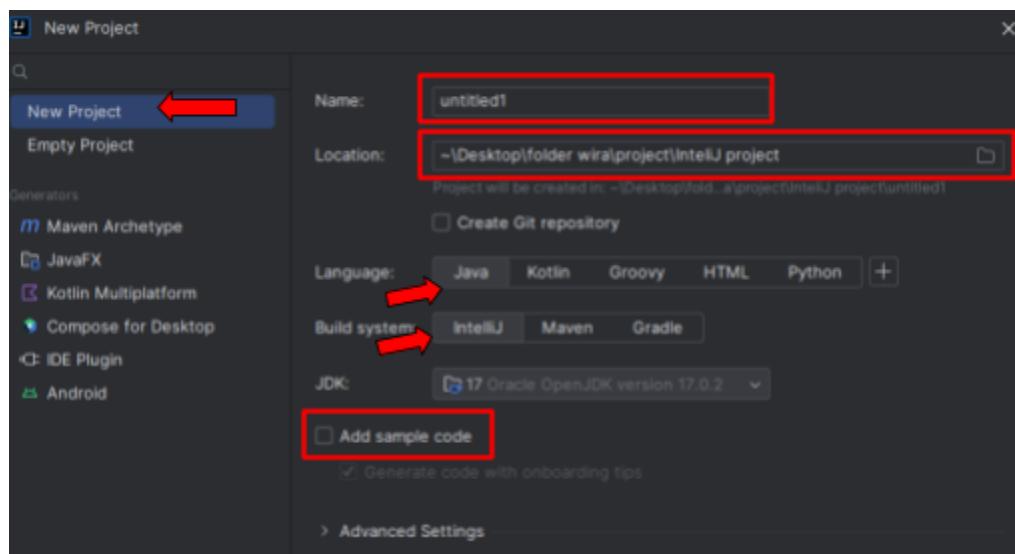
Process finished with exit code 0
```

## PRAKTEK

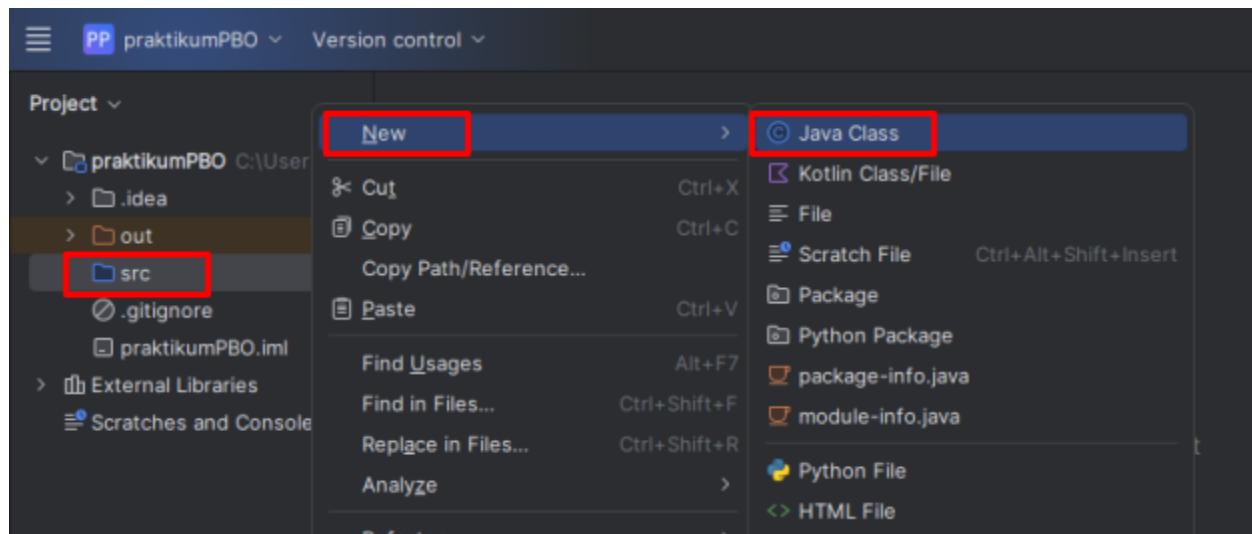
Mari kita bereksperimen dan membuat proyek baru! Buka IntelliJ dan silahkan ikuti tanda panah berwarna merah merona yang akan menemani kalian selama praktik ini.



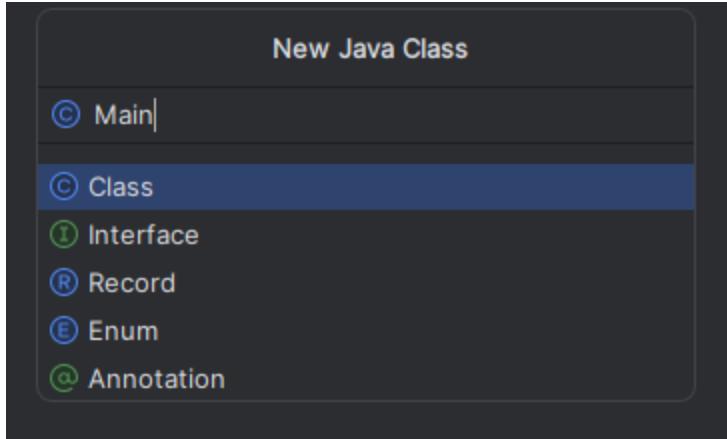
Kalian bisa mengganti nama dan directory dari proyek yang kalian buat. Kalian juga bisa mencentang bagian '**Add sample code**' untuk memulai dengan kodingan yang sudah ada (tapi kali ini kita uncheck aja dan mulai dari awal). Kalau sudah klik create yang ada dibawah.



Selanjutnya, klik kanan pada folder ‘src’ dan ikuti langkahnya untuk menambahkan kelas baru untuk memulai membuat program kalian:



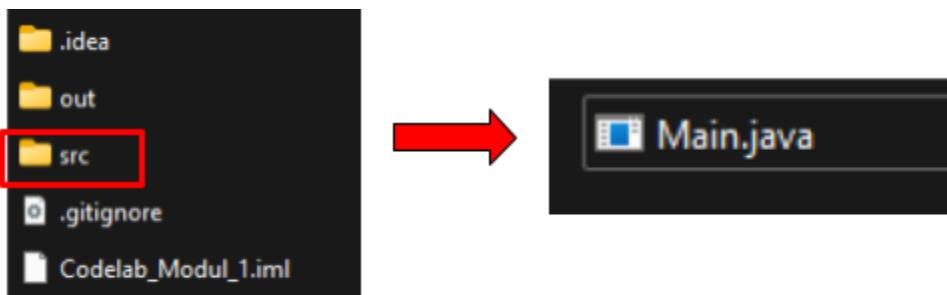
Kalian bebas menamai file kalian, namun kali ini mari kita namakan ‘Main’ saja ya ges ya.



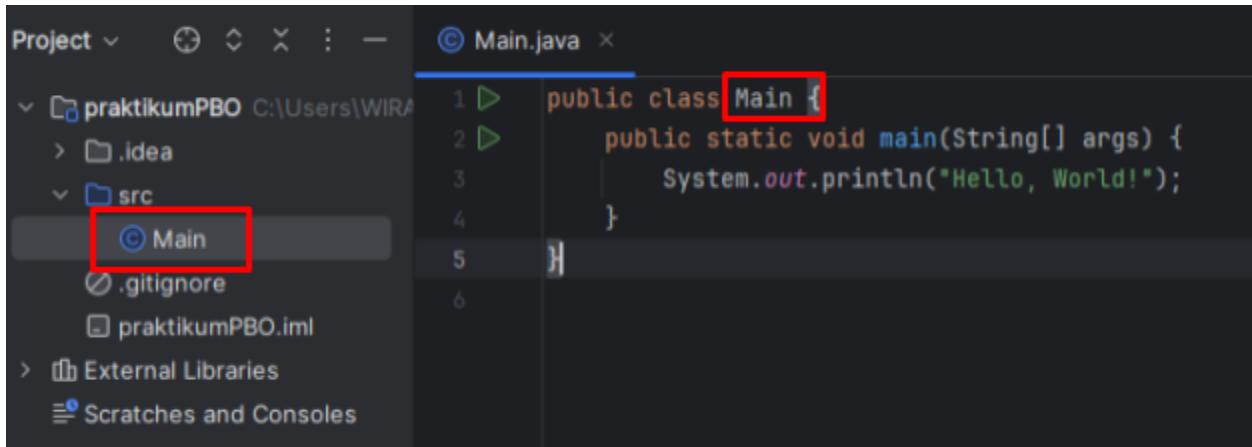
Ketika kalian membuat projek dan kelas baru di IntelliJ, project nya akan dibuat dalam bentuk folder beserta isinya. Hal ini berbeda dengan bahasa C kemarin yang hanya berupa satu file ‘.c’



Semua kelas yang kalian buat akan ditaruh di dalam folder ‘src’



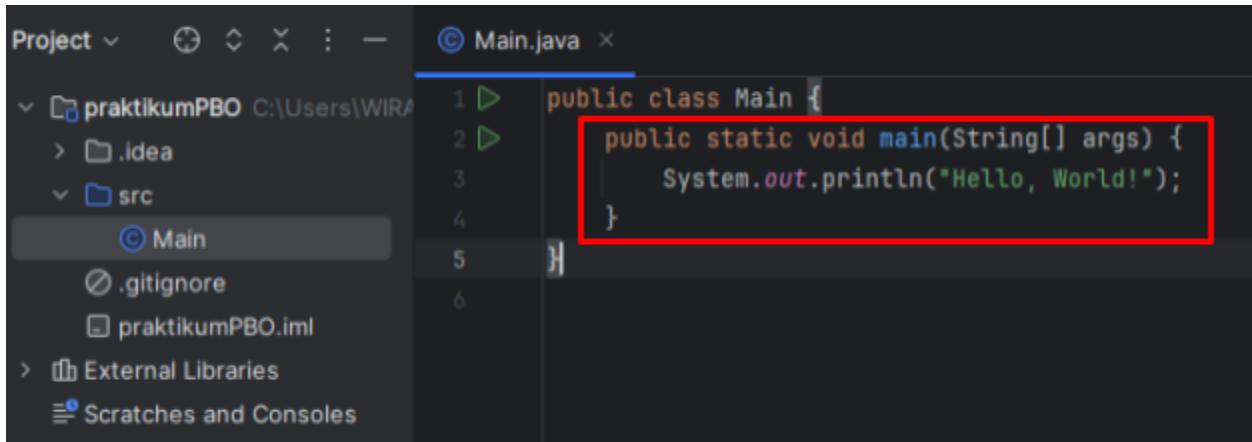
Berikut contoh isi dari kelas sederhana dalam Java. Silahkan ditulis ulang. Perhatikan yang saya tandai di gambar. Nama class pada source code dan folder harus sinkron.



```
Project ▾  © Main.java ×
  praktikumPBO C:\Users\WIRA
    .idea
    src
      © Main
      .gitignore
      praktikumPBO.iml
    External Libraries
    Scratches and Consoles

  1 public class Main {
  2   public static void main(String[] args) {
  3     System.out.println("Hello, World!");
  4   }
  5 }
  6 
```

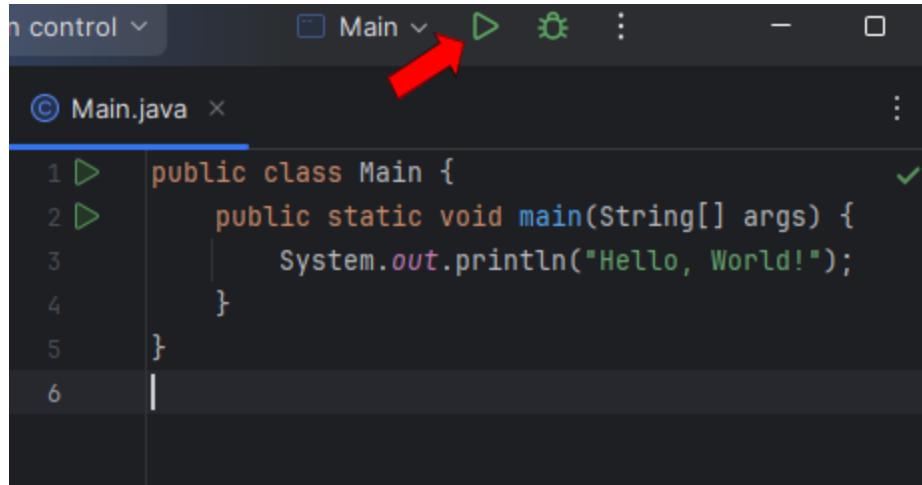
Kemudian di gambar selanjutnya, yang ditandai adalah fungsi main. Fungsi main sendiri sama seperti di bahasa C. Dimana digunakan untuk menjalankan program. Fungsi dan kelas sendiri memiliki kegunaan yang berbeda. Kita akan belajar ini di modul selanjutnya. Isi fungsi main dengan System.out.println("Hello, World!"); untuk menampilkan output seperti yang kalian sudah pelajari di sub bab sebelumnya.



```
Project ▾  © Main.java ×
  praktikumPBO C:\Users\WIRA
    .idea
    src
      © Main
      .gitignore
      praktikumPBO.iml
    External Libraries
    Scratches and Consoles

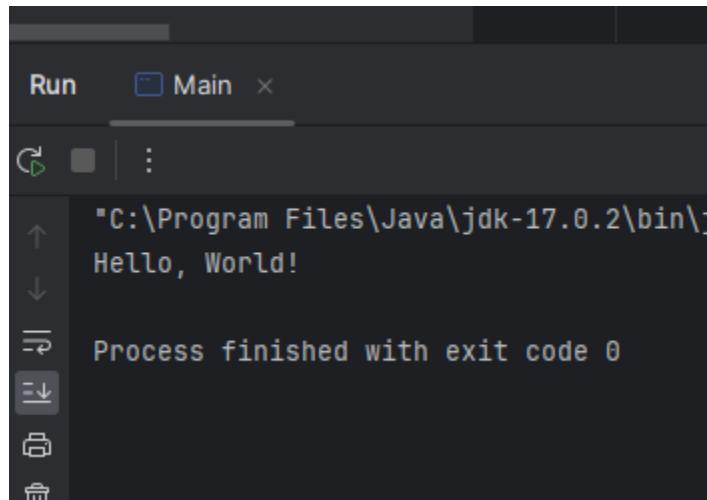
  1 public class Main {
  2   public static void main(String[] args) {
  3     System.out.println("Hello, World!");
  4   }
  5 }
  6 
```

Jika sudah, kalian bisa run program kalian dengan menekan tombol yang ditandai pada gambar selanjutnya ini:



```
1 > public class Main {  
2 >     public static void main(String[] args) {  
3 >         System.out.println("Hello, World!");  
4 >     }  
5 > }  
6 |
```

Maka output nya dapat dilihat seperti berikut:



```
"C:\Program Files\Java\jdk-17.0.2\bin\java  
Hello, World!  
Process finished with exit code 0
```

Selamat. Itu adalah program Java pertama kalian (I guess?) Silahkan kembangkan lebih lanjut ;). Jika terdapat kesulitan silahkan tanya asisten kalian masing-masing. Selamat mencoba!

### TIPS

Menampilkan Teks Hello World:

[Video tutorial](#)

Tipe Data dan Variable:

[Video tutorial](#)

Eksplorasi Variable dan Tipe Data:

[Video tutorial](#)

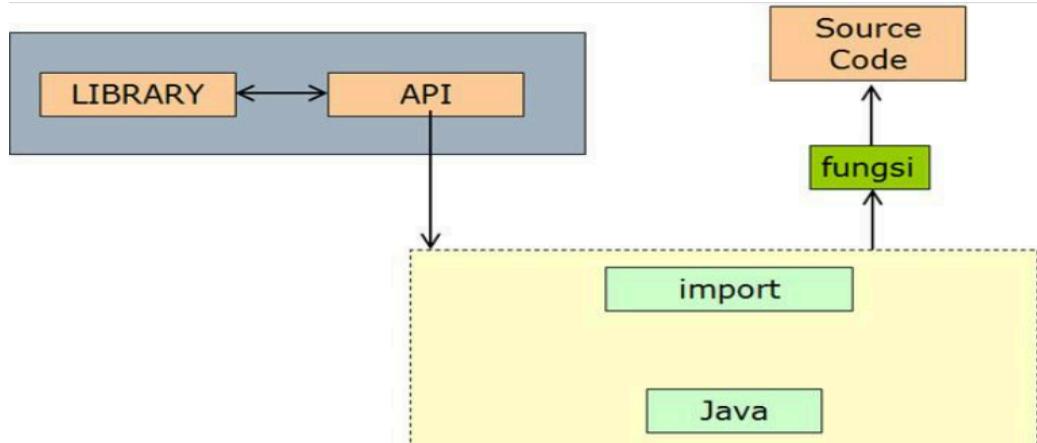
**JAVA API****TEORI****Apa itu Java API?**

**Java Application Programming Interface (Java API)** adalah **komponen-komponen** dan **kelas Java** yang sudah disediakan oleh Java dan memiliki berbagai kemampuan. Kemampuan pada java API bermacam-macam, mulai dari **menangani objek, string, angka**, dan sebagainya. Java API merupakan seperangkat method yang disediakan oleh JDK. JDK menyediakan banyak library API yang dapat melakukan tugas pemrograman dasar seperti menampilkan GUI, fungsi math, dan banyak lainnya. Kelas-kelas java API dibungkus dalam sebuah package yang ditulis dalam bahasa pemrograman java terstruktur dan berjalan pada JVM.

**MATERI****Bagian API dalam Java**

- **API standard (Java SE)** → yang digunakan untuk aplikasi dan applet dengan layanan bahasa dasar
- **API enterprise (Java EE)** → untuk aplikasi server dengan layanan database dan aplikasi server-side (servlet)
- **API untuk device micro (Java ME)** → seperti handphone

Schema Java API:



Adapun contoh untuk penggunaan Java API, seperti ini:

```

● ● ●

import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;

public class Main {
    public static void main(String[] args){
        LocalDateTime timeNow = LocalDateTime.now();
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");

        System.out.println("Waktu Sekarang Adalah : " + timeNow.format(formatter));
        System.out.println("Bulan Sekarang Adalah : " + timeNow.getMonth());
    }
}
  
```

Output:

```

Waktu Sekarang Adalah : 2025-02-05
Bulan Sekarang Adalah : FEBRUARY

Process finished with exit code 0
  
```

### PRAKTEK

Mari kita bereksperimen menggunakan API Java sederhana! Kita akan melanjutkan praktek pada bab sebelumnya. Buka project tersebut, lalu silahkan import API yang akan kita gunakan (kita akan menggunakan `java.util.Date`) dengan cara seperti berikut:

```
import java.util.Date;  
  
public class Main {  
    public static void main(String[] args) {  
        System.out.println("Hello, World!");  
    }  
}
```

java.util.Date adalah bagian dari Java API. Java API adalah kumpulan kelas dan antarmuka yang disediakan oleh Java untuk membantu pengembang dalam membangun aplikasi. Jadi, dalam program ini, kita menggunakan kelas Date dari Java API untuk mendapatkan dan menampilkan waktu saat ini. Ikuti cara berikut:

```
import java.util.Date;  
  
public class Main {  
    public static void main(String[] args) {  
        System.out.println("Hello, World!");  
        System.out.println("Current date and time: " + new Date());  
    }  
}
```

**TIPS:** kalian bisa tuliskan “sout” lalu pencet tab untuk membuat fungsi System.out.println(); secara instant

Berikut outputnya (outputnya akan menyesuaikan dengan waktu saat kalian jalankan programnya):

```
"C:\Program Files\Java\jdk-17.0.2\bin\java.exe" "-jav  
Hello, World!  
Current date and time: Fri Feb 14 20:02:28 WIB 2025  
  
Process finished with exit code 0
```

Sangat keren bukan? Silahkan bereksperimen dengan kelas yang lain. Jika kesulitan silahkan tanya asisten kalian masing-masing. Selamta mencoba!

### TIPS

Video singkat belajar API:

Video tutorial**GIT & GITHUB****TEORI****GIT**

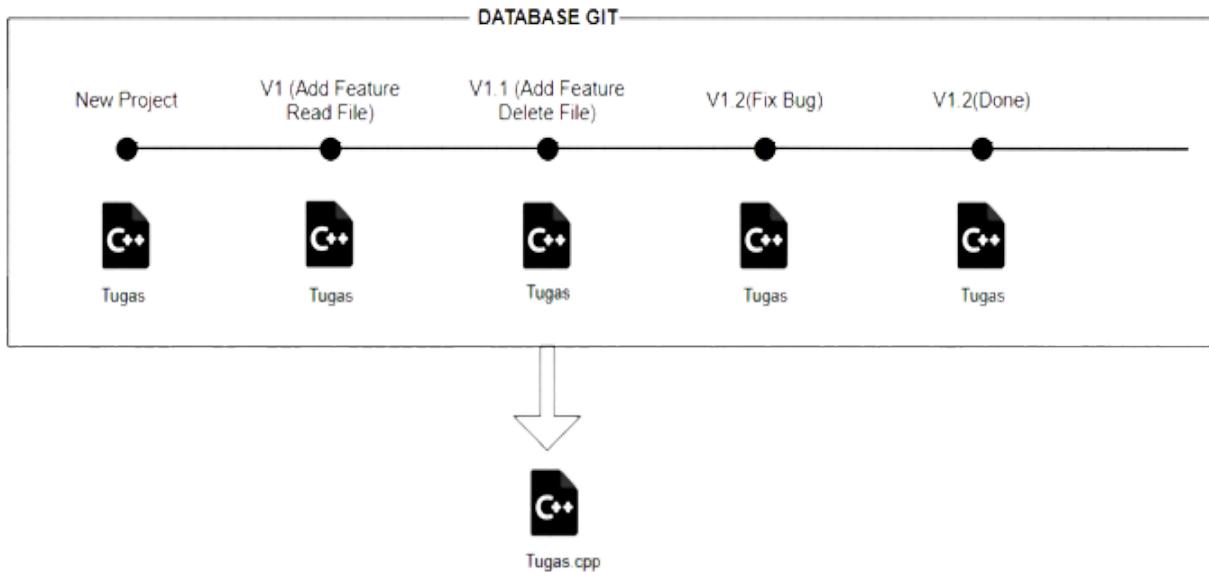
Git adalah sebuah **software** berbasis **Version Control System (VCS)** yang digunakan untuk mencatat perubahan **seluruh file** atau **repository** suatu **project** yang kita buat. Developer software biasanya menggunakan **Git** untuk **distributed revision (VCS terdistribusi)**, hal ini bertujuan untuk menyimpan **database** tidak hanya ke satu tempat, namun semua orang bisa terlibat dalam pembuatan kode yang dapat menyimpan **database** ini.

```

↳ tugas.cpp
↳ tugas V1(add feature read file).cpp
↳ tugas V1.1(add feature delete file) .cpp
↳ tugas V1.2(fix bug).cpp
↳ tugas V1.3(done nilai A).cpp
  
```

Lebih jelasnya bisa lihat pada **contoh kasus diatas**, yang biasa dilakukan oleh mahasiswa dalam mengerjakan sebuah tugas. Setiap kali melakukan revisi, file yang telah lama tidak dihapus dan disimpan dengan nama yang berbeda. Sedangkan yang terbaru disimpan dengan nama, contoh : “**Tugas V1 (add feature)**”.

Konsep penggerjaan dengan cara tersebut dianggap sangat **tidak efisien** oleh banyak developer karena kapasitas penyimpanan akan membengkak karena file lama tidak dihapus. Disinilah **VCS** berfungsi untuk membantu penyimpanan berupa history tanpa membuat sebuah file baru, yang tersimpan hanyalah perubahan data pada file tersebut. Sehingga kapasitas penyimpanan file menjadi lebih **ringan**.



Seperti pada gambar di atas, setiap perubahan data secara manual akan menghasilkan banyak file. Sedangkan VCS mengusung sebuah konsep untuk menyimpan history perubahan pada satu file saja. Prosedur yang diterapkan ini dapat membantu jika divisi pada sebuah project untuk memantau dan menghubungkan (merge) antar ekstensi yang berbeda dengan mudah. Sehingga aplikasi yang dibuat oleh sebuah tim project dapat berfungsi tanpa melakukan penamaan secara manual.

### GITHUB



**Github** merupakan **layanan cloud** yang berfungsi untuk **menyimpan** dan **mengelola** sebuah **project** yang dinamakan **repository (repo git)**. Cara kerja pada **Github** harus terkoneksi dengan **internet**, sehingga tidak perlu untuk **menginstall** sebuah **software** tambahan ke dalam komputer kita. Ini bisa memberikan keringanan penyimpanan komputer yang kita gunakan karena file project tersimpan di **cloud Github**.

Konsep kerja **Github** pada dasarnya sama dengan **Git** yaitu menulis sebuah **source code** secara individu atau time. User interface yang tersedia pada **Github** lebih menarik dan mudah dipahami oleh pengguna baru. Jika bekerja secara tim, satu pengguna bisa melihat siapa penulis kode dan tanggal berapa kode tersebut dibuat.

**Perbedaan GIT dan GitHub**

Git	GitHub
Meng-install software di penyimpanan lokal	Host melalui layanan cloud
Dikelola oleh The Linux Foundation	Diakuisisi oleh Microsoft pada 2018
Berfokus pada version control dan code sharing	Berfokus pada source code hosting terpusat
Akses secara offline	Akses secara online
Tidak menggunakan fitur user management	Menggunakan user management
Menyediakan desktop interface bernama "Git GUI"	Menggunakan nama desktop interface "GitHub Desktop"
Bersaing dengan Mercurial, Subversion, IBM, Rational Team, Concert, dan ClearCase	Bersaing dengan GitLab dan Atlassian BitBucket
Open sourced licensed	Pilihan bagi pengguna gratis dan pengguna berbayar

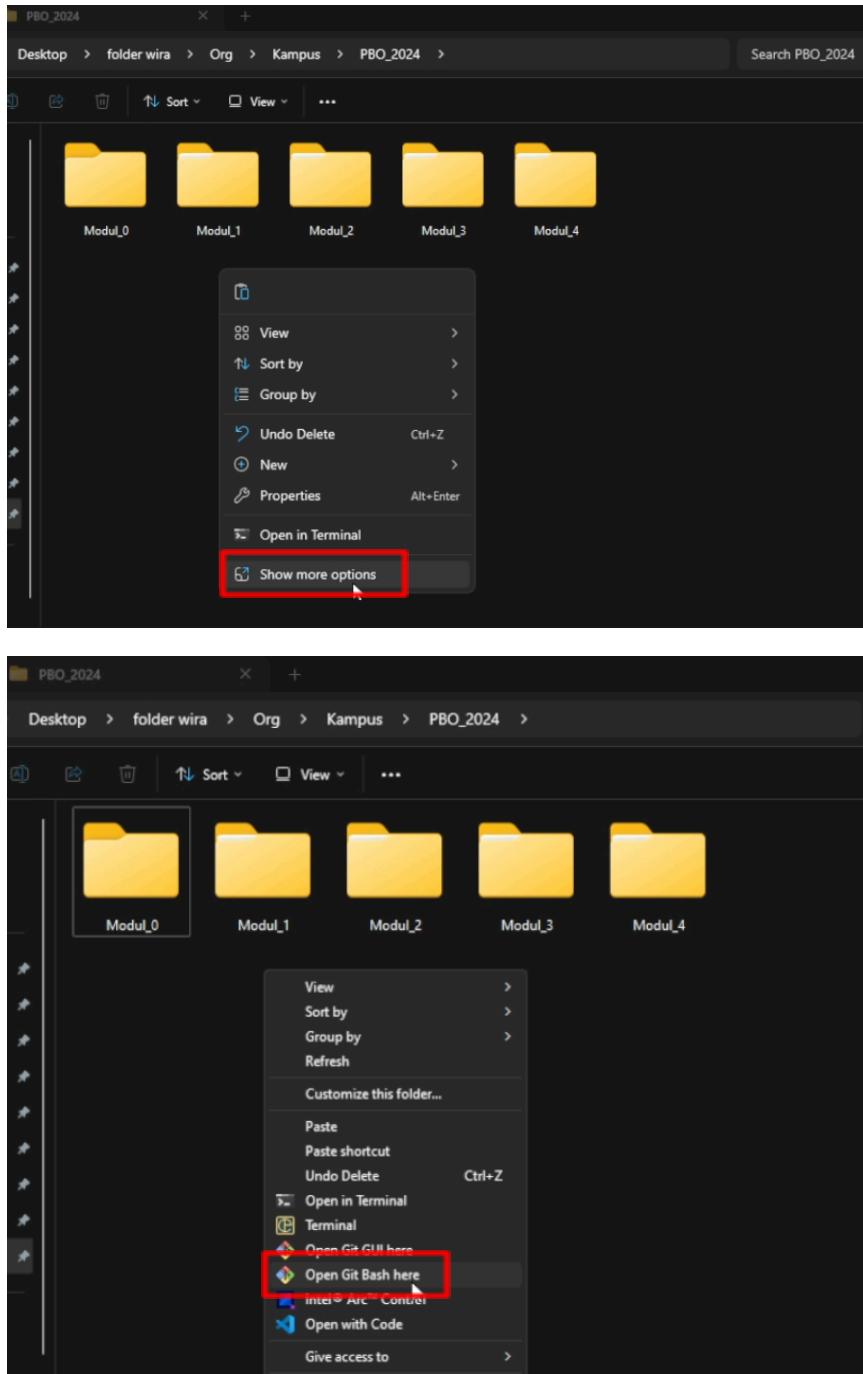
**MATERI****Cara upload source code ke repository Github dengan Git**

Requirement:

- Buat akun di <https://github.com>
- Untuk **windows**: Download dan install git <https://git-scm.com/downloads>
- Untuk **Linux** : apt-get install git
- Untuk **MacOS** : \*mohon maaf kami terkendala tidak ada device mac T-T\*

Masuk ke step-stepnya:

1. Buka **directory/folder** source code yang akan di upload di **github**. Jika sudah berada di **directory/folder** maka klik kanan dan pilih **open git bash here** seperti contoh yang ada di gambar:

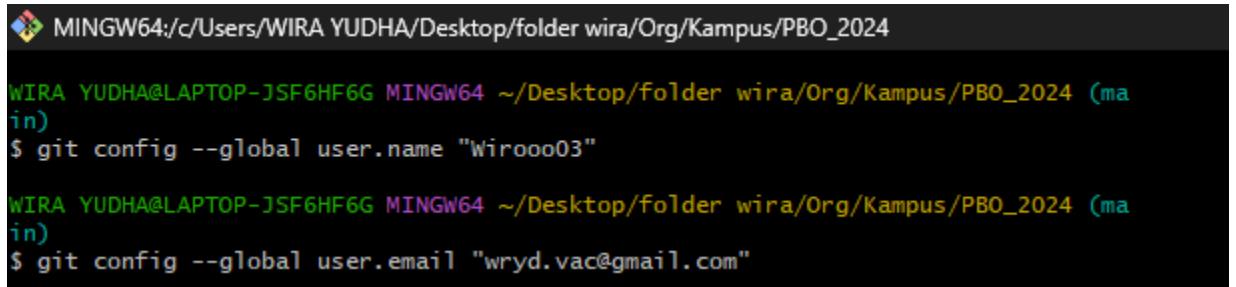


2. Masukkan **user** dan **password** menggunakan perintah berikut:

```
git config --global user.name "username"
git config --global user.email "email@gmail.com"
```

**Catatan:** Ganti kata-kata di dalam tanda petik dua dengan **username** dan **email** yang sesuai dengan **username** dan **email** yang telah kalian buat di **github**. Langkah kedua ini hanya diperlukan bagi kalian yang baru **pertama kali** memasang **Git**. Jika kamu sudah pernah menggunakan **Git**, bisa langsung masuk ke **langkah 3**

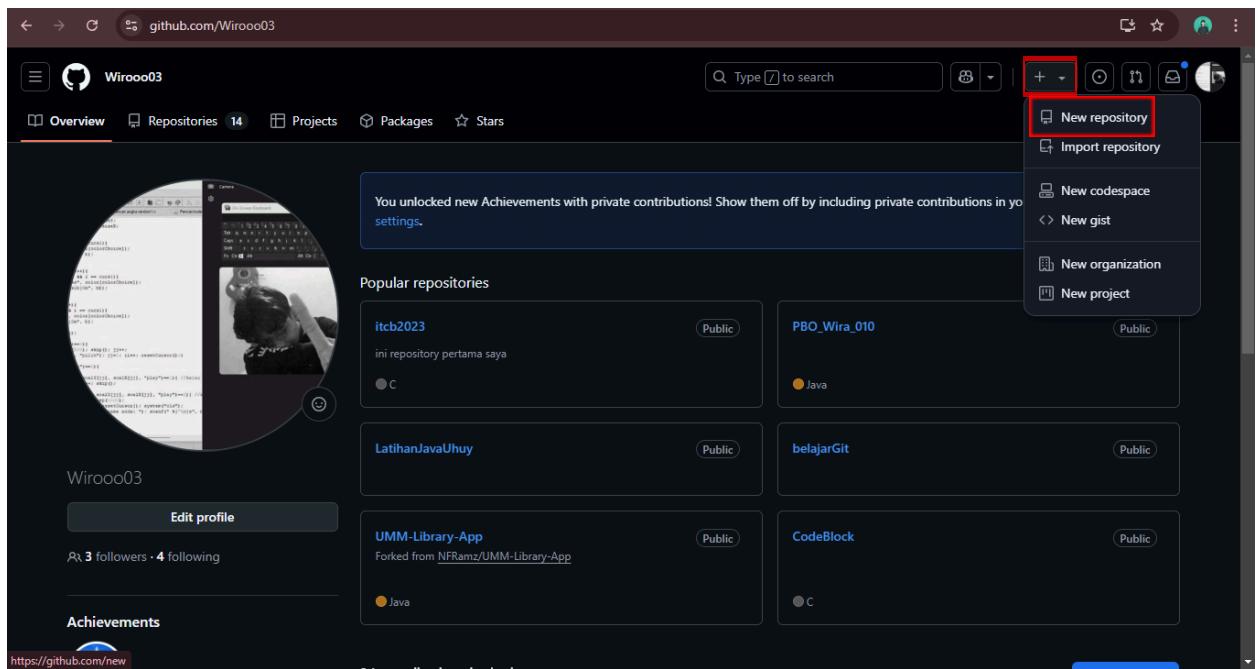
Contoh:



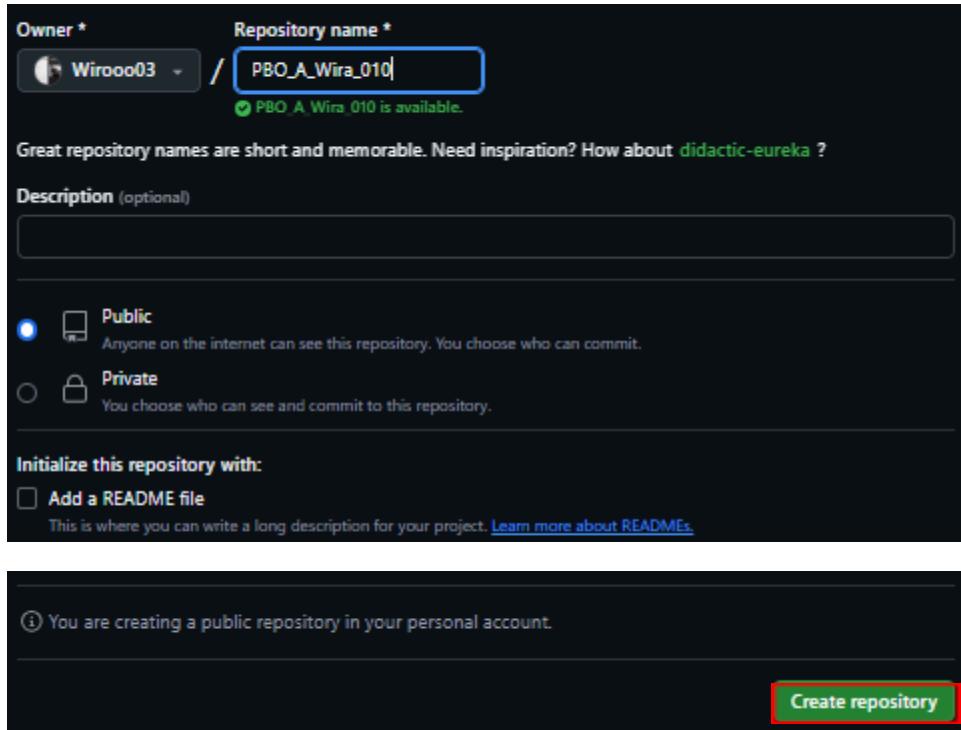
```
MINGW64:/c/Users/WIRA YUDHA/Desktop/folder wira/Org/Kampus/PBO_2024
WIRA YUDHA@LAPTOP-JSF6HF6G MINGW64 ~/Desktop/folder wira/Org/Kampus/PBO_2024 (main)
$ git config --global user.name "Wiroooo03"

WIRA YUDHA@LAPTOP-JSF6HF6G MINGW64 ~/Desktop/folder wira/Org/Kampus/PBO_2024 (main)
$ git config --global user.email "wryd.vac@gmail.com"
```

3. **Login** akun **github** yang telah dibuat sebelumnya. Setelah berhasil login, klik ikon plus pada bagian kanan atas halaman aktif GitHub. Selanjutnya, klik **New Repository**

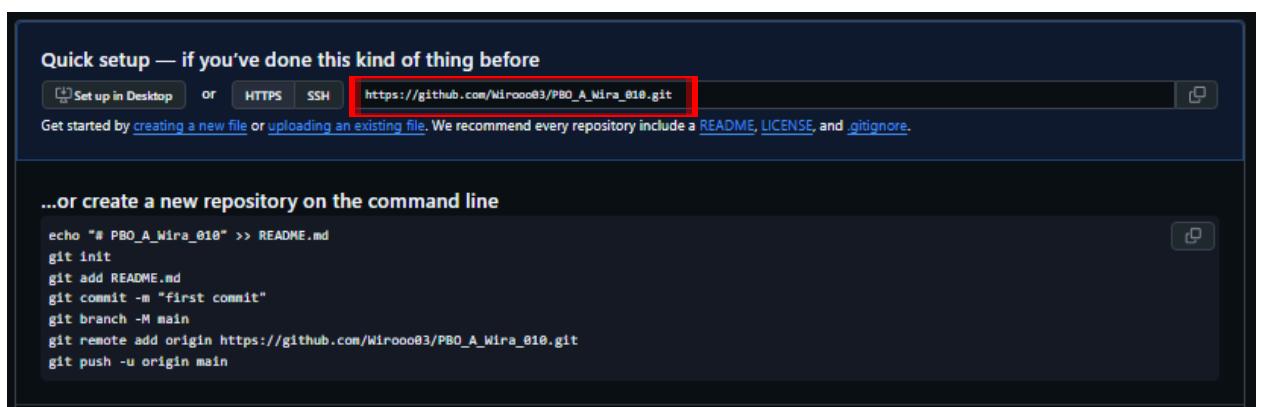


4. Isi kolom **Repository Name**. Pastikan menu lainnya terisi sesuai dengan gambar yang di contohkan. Jika sudah sama, klik **Create Repository**.



**Catatan penting:** **Repository** merupakan sebuah folder pusat yang berfungsi untuk menyimpan **source code** kita nantinya. **Owner (pembuat)** bisa membatasi hak akses repository pada siapa saja yang bisa mengaksesnya. Hanya pada orang-orang tertentu yang kalian beri izin dapat **membaca, menulis, dan menghapus** file di **repository**. Pengaturan akses perizinan ini ada di bagian **collaborator**. Untuk sekarang, uncheck dulu **Add a README file** nya.

5. Salin link yang ada di form quick setup



6. Untuk mengunggah source code yang telah dibuat ikuti command sesuai contoh berikut ini:

- Ketikkan perintah pada **gitbash** seperti pada **langkah 2**
- Sebelum mengetikkan perintah **git push -u origin main** ketikkan perintah **git remote add origin <url yang disalin pada langkah 5>**

```

WIRA YUDHA@LAPTOP-JSF6HF6G MINGW64 ~/Desktop/folder wira/Org/Kampus/PBO_2024 (main)
$ git init
Initialized empty Git repository in C:/Users/WIRA YUDHA/Desktop/folder wira/Org/Kampus/PBO_2024/.git/

WIRA YUDHA@LAPTOP-JSF6HF6G MINGW64 ~/Desktop/folder wira/Org/Kampus/PBO_2024 (master)
$ git add .
warning: in the working copy of 'Modul_1/codelab/.gitignore', LF will be replaced by CRLF. The working copy will be updated.

WIRA YUDHA@LAPTOP-JSF6HF6G MINGW64 ~/Desktop/folder wira/Org/Kampus/PBO_2024 (master)
$ git commit -m "first commit"
[master (root-commit) 6ef583b] first commit
 73 files changed, 1337 insertions(+)
 create mode 100644 Modul_1/codelab/.gitignore
 create mode 100644 Modul_1/codelab/.idea/.gitignore
 create mode 100644 Modul_1/codelab/.idea/misc.xml
 create mode 100644 Modul_1/codelab/.idea/modules.xml
 create mode 100644 Modul_1/codelab/.idea/uiDesigner.xml

WIRA YUDHA@LAPTOP-JSF6HF6G MINGW64 ~/Desktop/folder wira/Org/Kampus/PBO_2024 (master)
$ git branch -M main

WIRA YUDHA@LAPTOP-JSF6HF6G MINGW64 ~/Desktop/folder wira/Org/Kampus/PBO_2024 (main)
$ git remote add origin https://github.com/Wiroooo03/PBO_A_Wira_010.git

WIRA YUDHA@LAPTOP-JSF6HF6G MINGW64 ~/Desktop/folder wira/Org/Kampus/PBO_2024 (main)
$ git push -u origin main
Enumerating objects: 62, done.
Counting objects: 100% (62/62), done.
Delta compression using up to 4 threads

```

#### Catatan:

- Tanda “.” pada “**git add .**” berarti menambahkan apapun yang baru pada folder tersebut. Atau menambahkan apapun yang telah diubah dari berkas sebelumnya. Kamu bisa menambahkan berkas tertentu saja dengan cara “**git add tugas.java**”
- **git commit -m** merupakan perintah untuk memberikan **pesan** terhadap berkas yang diunggah. Kamu dapat mengganti pesan di dalam simbol petik dua (“....”) dengan pesan yang lain.
- Perintah **main** merupakan nama **cabang utama** dalam **repository** milikmu. Istilahnya adalah **branch**.

- Setelah kamu sukses dalam mengunggah berkas ke **GitHub**, silahkan pergi ke **repository**.

The screenshot shows a GitHub repository page for 'PBO\_A\_Wira\_010'. At the top, it says 'main' branch, '1 Branch', and '0 Tags'. Below that is a search bar with 'Go to file' and a 'Code' button. A list of commits is shown, all from 'Wirooo03' and labeled 'first commit'. The commits are for files 'Modul\_1', 'Modul\_2', 'Modul\_3', and 'Modul\_4/codelab', each made 12 minutes ago. There is also a note about 1 Commit.

Commit	File	Date
1 Commit		12 minutes ago
first commit	Modul_1	12 minutes ago
first commit	Modul_2	12 minutes ago
first commit	Modul_3	12 minutes ago
first commit	Modul_4/codelab	12 minutes ago

- Apabila berkas yang diunggah sudah sesuai, itu artinya sudah **berhasil** menambahkan **berkas** ke dalam **github**.

#### IntelliJ IDEA Built-In Version Control

Pada materi ini kita akan mencoba untuk melakukan konfigurasi Git akan tersambung dengan IntelliJ IDEA. Ikuti beberapa langkah di bawah ini untuk konfigurasi Git pada IntelliJ IDEA:

- Pastikan **git** sudah **terinstall** pada OS yang kita pakai, jika belum diinstall bisa ikuti langkah-langkah materi sebelumnya. Untuk memastikan git sudah terinstall silahkan buka cmd atau terminal untuk linux, lalu ketikan command git -v.

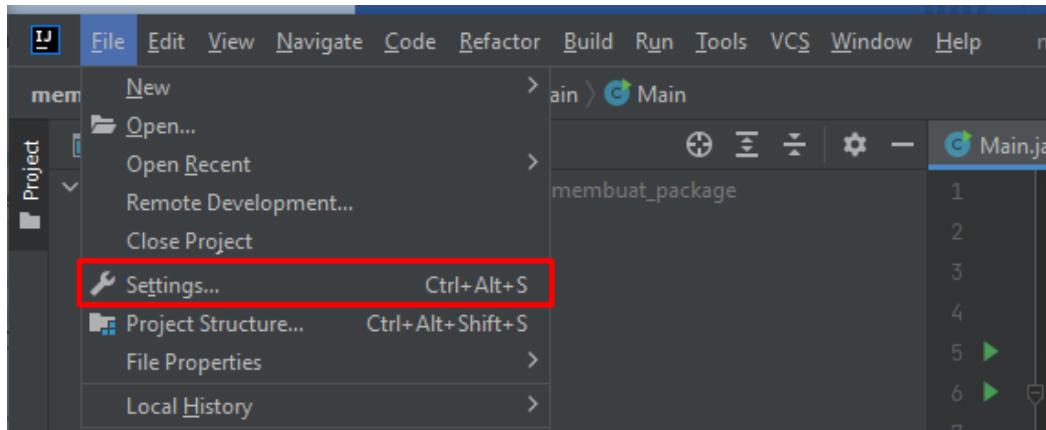
Catatan : Buat repository terlebih dahulu seperti dilangkah 3 dan 4

```
C:\Users\WIRA YUDHA>git -v
git version 2.44.0.windows.1

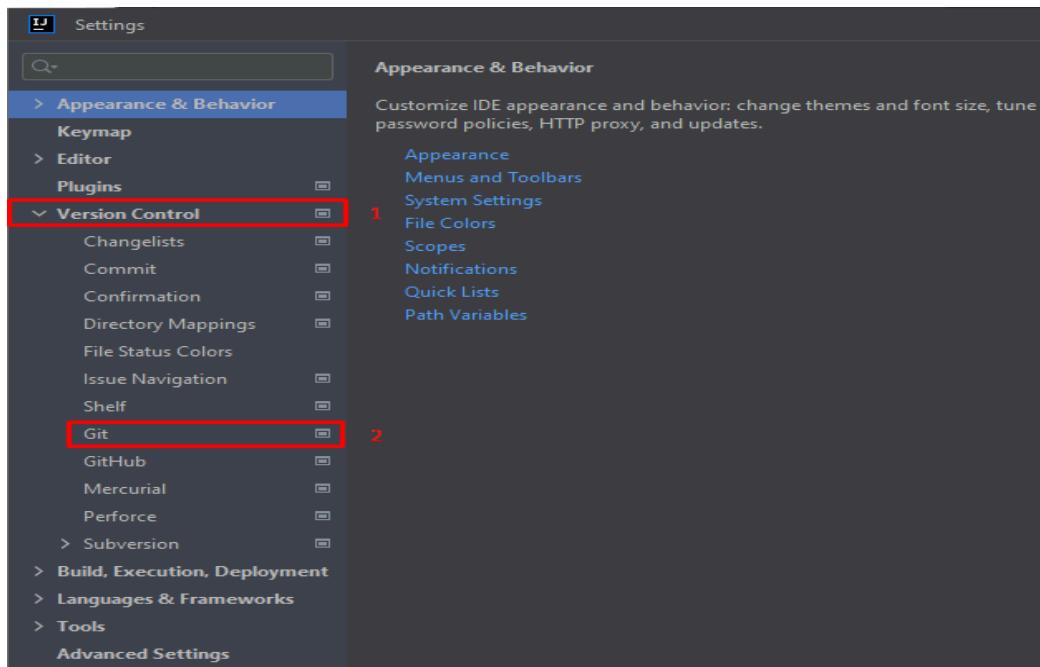
C:\Users\WIRA YUDHA>
```

Jika muncul output seperti gambar di atas itu artinya git sudah terinstall dengan versi 2.37.2.windows.2

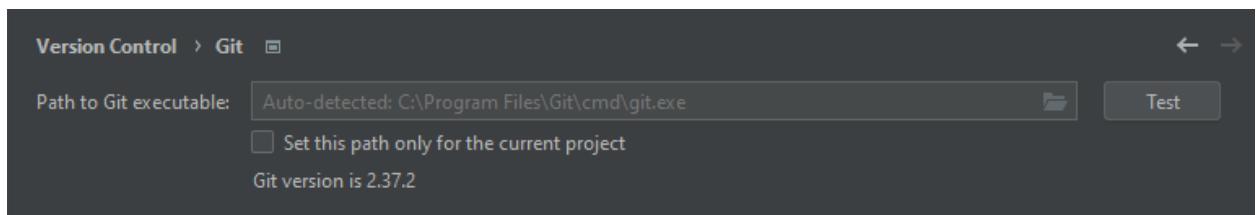
- Buka IntelliJ IDEA dan pergi ke File di pojok kiri atas > Settings.



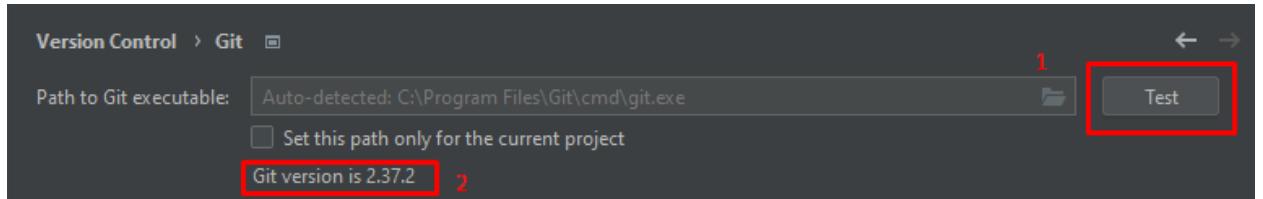
### 3. Pilih opsi Version Control > Git



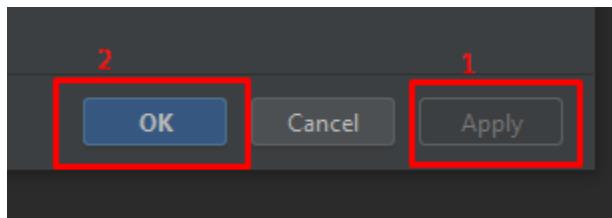
### 4. Pada bagian Path to Git executable, spesifikan path pada instalasi Git executable (biasanya git atau git.exe untuk windows). Jika tertulis autodetected seperti ini, itu artinya instalasi Git kita sudah terdeteksi otomatis.



5. Klik Test untuk memastikan path yang dimasukkan sudah benar. Akan muncul pesan versi git jika sudah benar.



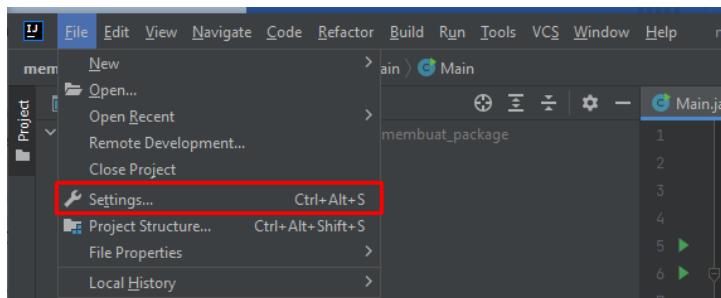
6. Klik apply dan OK untuk menyimpan konfigurasi.



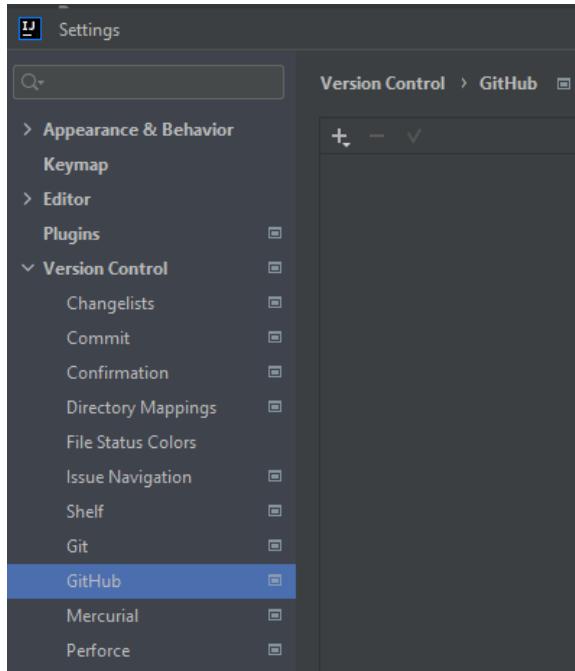
### Integrasi Github dengan IntelliJIDEA

Untuk proses integrasi Github dengan IntelliJIDEA, bisa ikuti pada beberapa langkah di bawah:

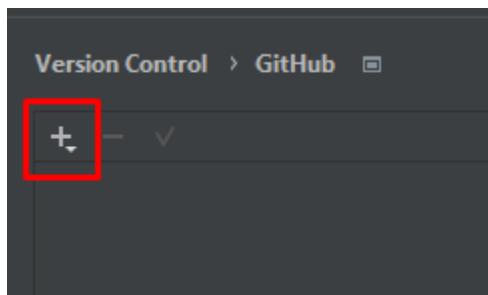
1. Pergi ke File > Settings



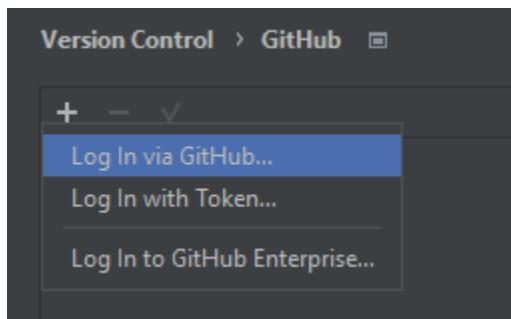
2. Navigasikan ke Version Control > Github



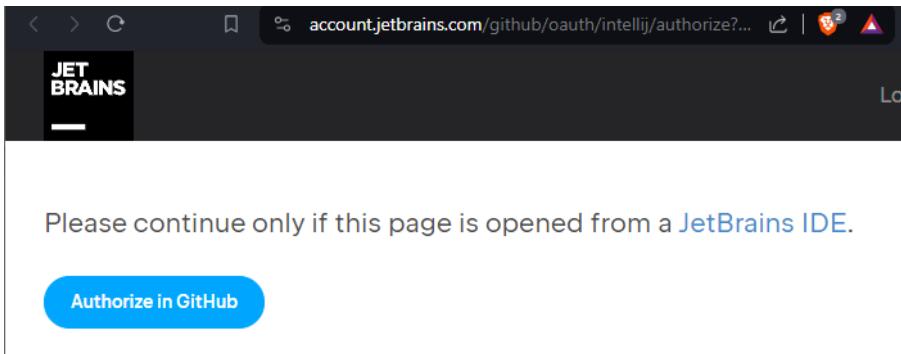
3. Tekan icon + untuk menambahkan akun Github



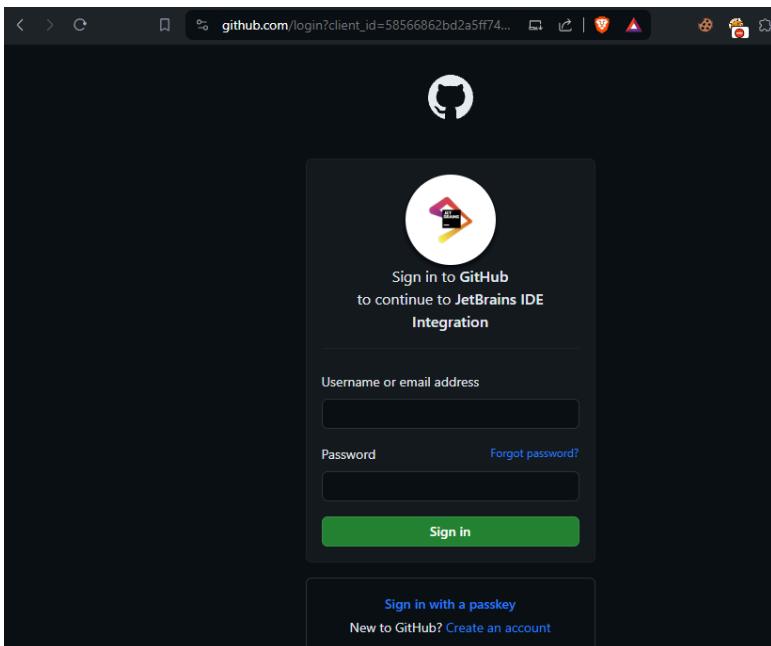
4. Pilih akses login melalui Github atau Token, dalam contoh disini akan menggunakan Github



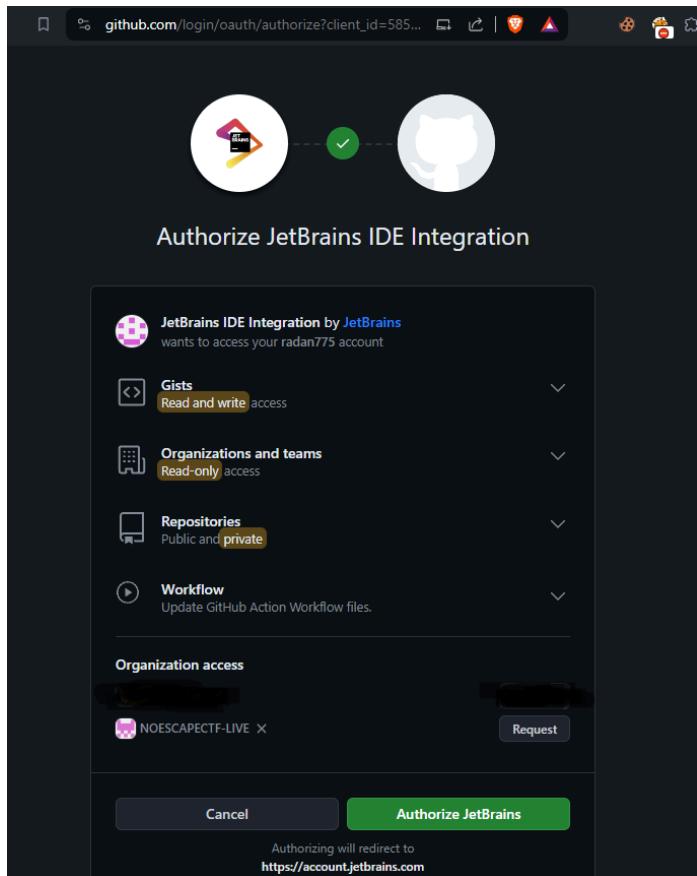
5. Ketika menekan Log In via Github akan muncul tab browser baru dan klik Authorize in Github



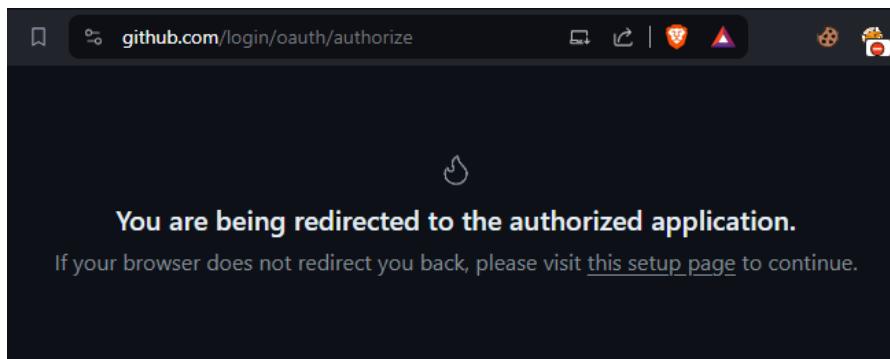
6. Isikan kredensial akun sesuai dengan username dan password, lalu klik Sign In.



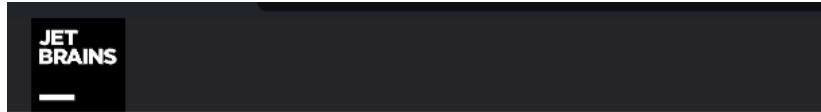
7. Klik Authorize JetBrains



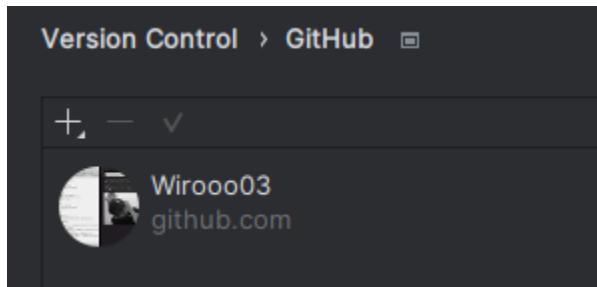
8. Ketika muncul tampilan seperti ini harap ditunggu



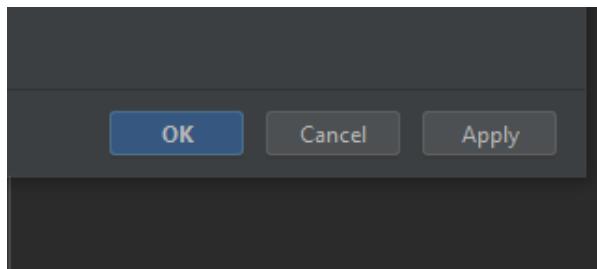
9. Dan akan muncul website dengan tampilan berikut



10. Kembali ke IntelliJIDEA, maka akan muncul akun yang ditambahkan



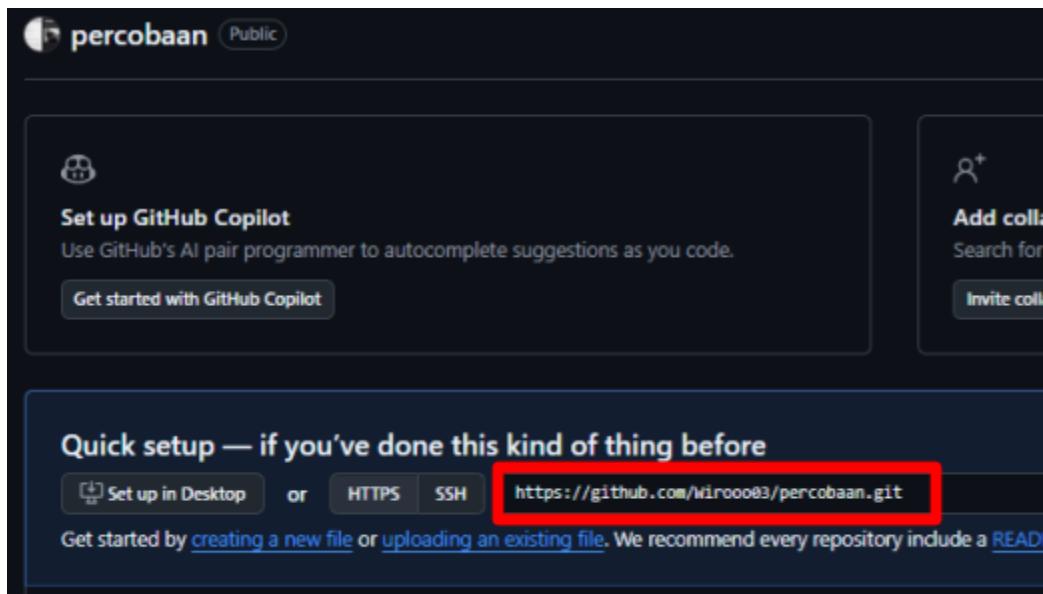
11. Terakhir klik Apply dan OK



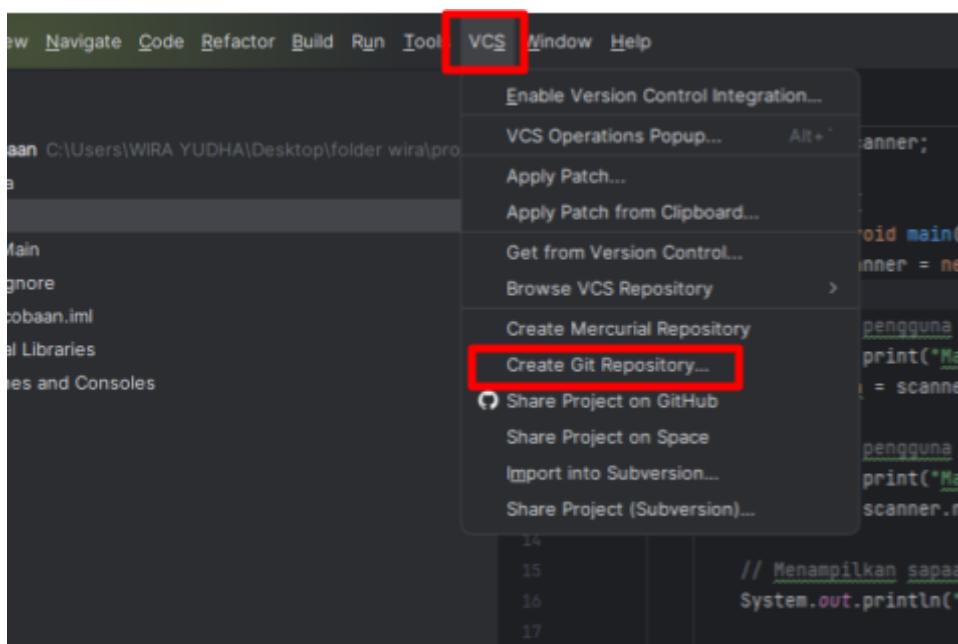
### **Bekerja dengan Git Repository**

Untuk membuat atau clone sebuah repository Git, ikuti beberapa langkah berikut:

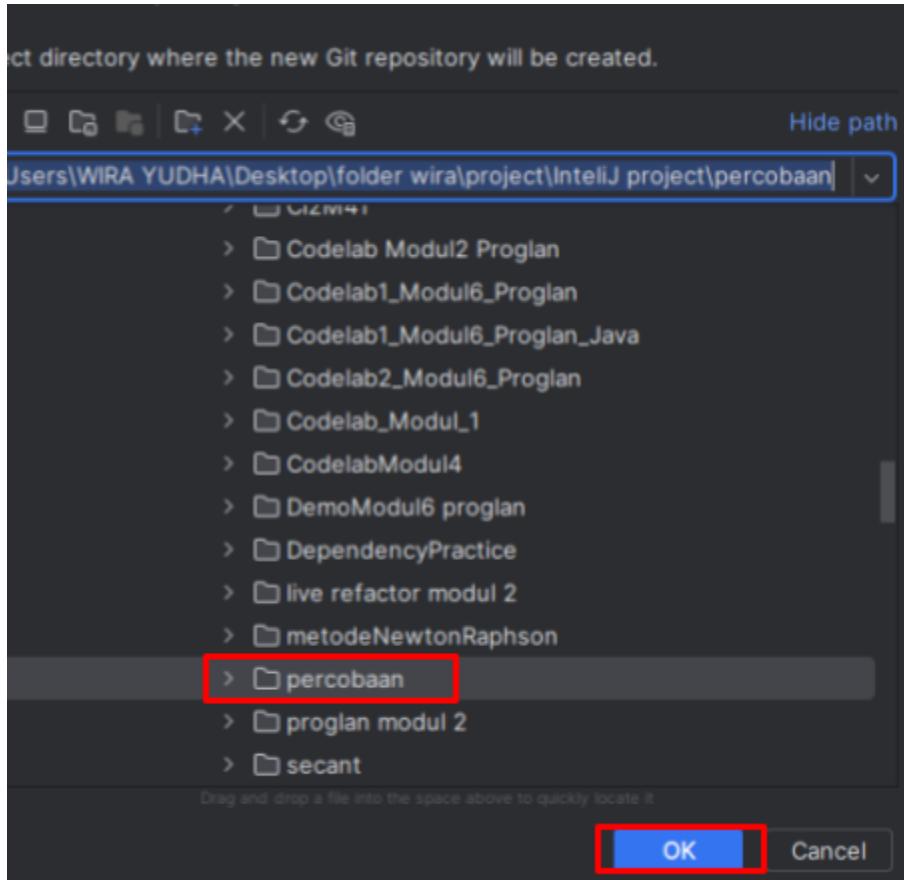
1. Buat repository di github seperti Langkah 3 dan 4 pada tutorial git dan github dan copykan url repository nya



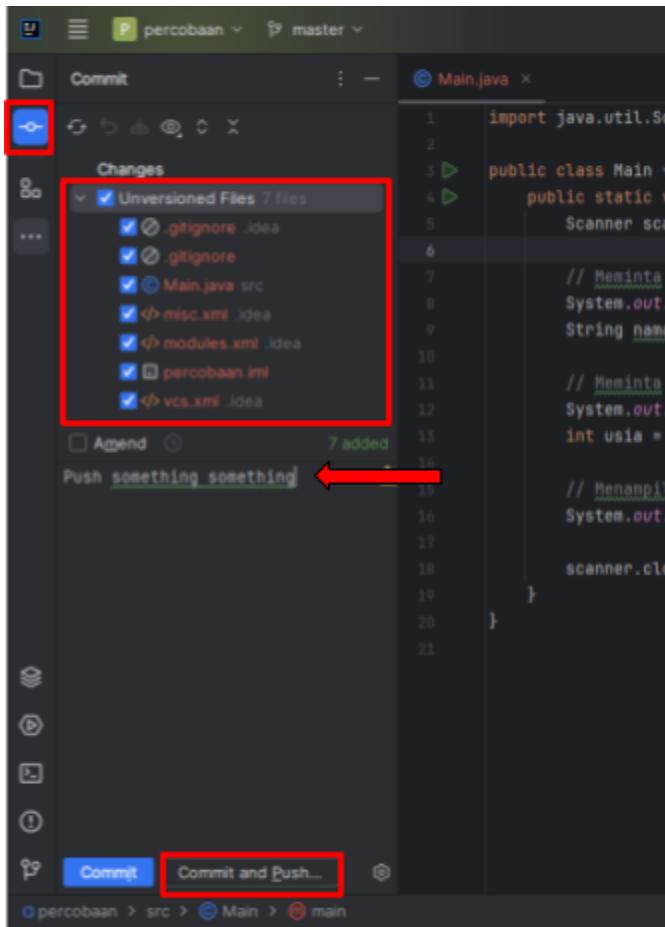
2. Untuk membuat repository baru, pergi ke VCS > Get From Version Control



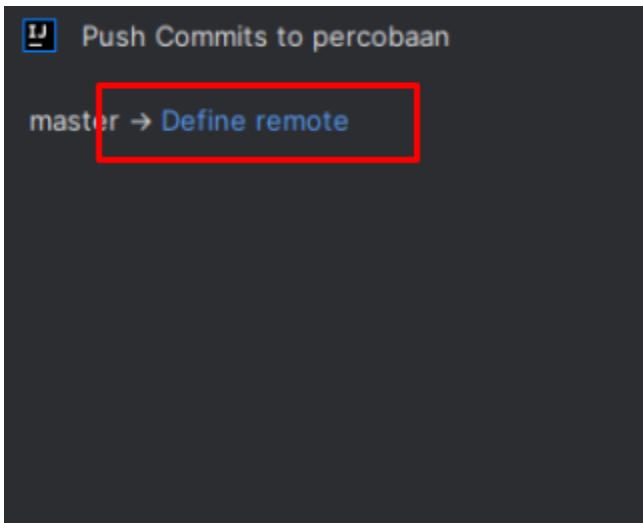
3. Pilih directory file yang akan di push ke github lalu klik "ok"



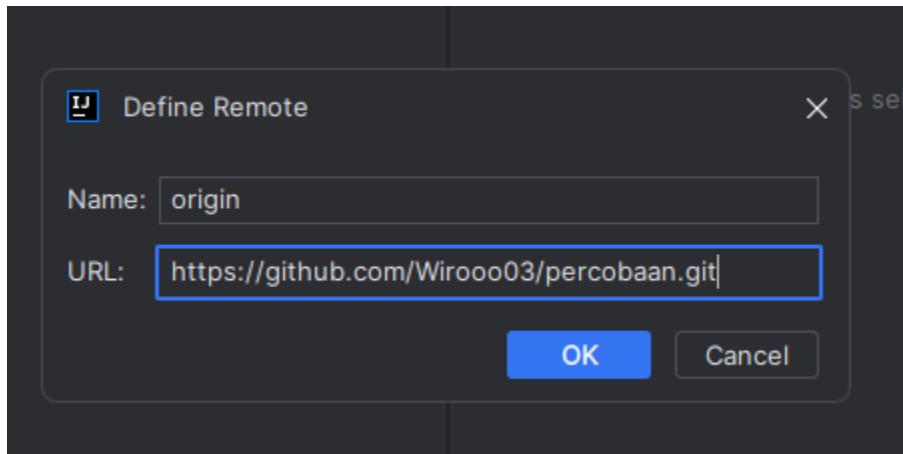
4. Pilih sidebar commit lalu centang semua file yang akan di push ke github > isi kolom yang dibawahnya sebagai keterangan commit contoh “first commit” pilih commit / commit and push



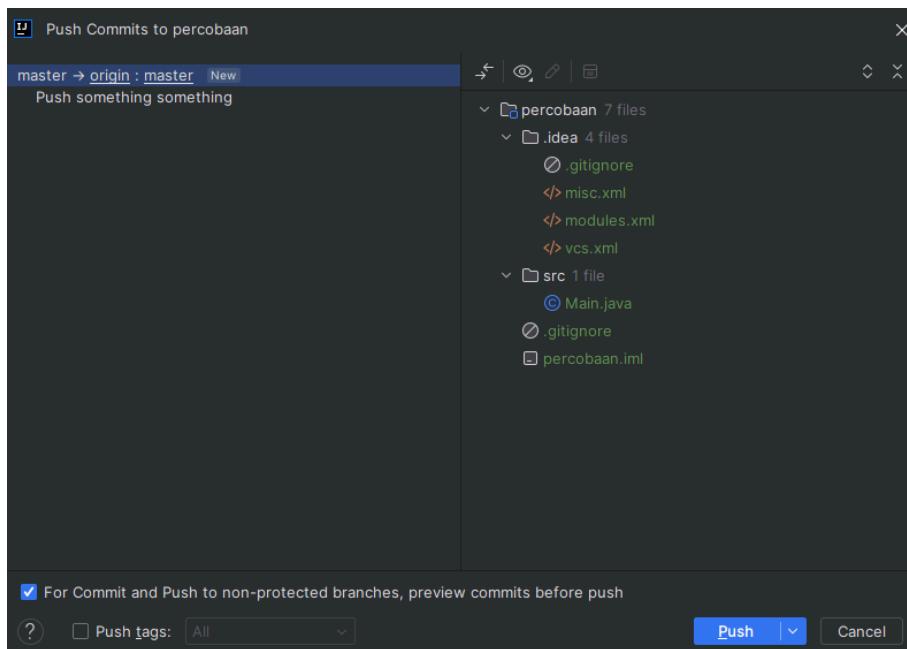
5. setelah melakukan step 3 maka akan keluar jendela baru jika kita pertama melakukan development / pertama kali push dan pilih define remote



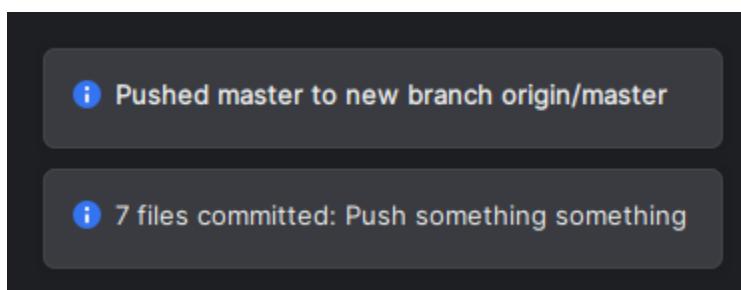
6. untuk form url isi dengan url repository yang telah dibuat di step 1 diatas dan klik ok



7. lalu muncul muncul preview folder yang telah kita pilih dan klik push



8. jika terdapat pesan seperti gambar dibawah ini pada pojok kanan bawah maka source code kita sudah berhasil terupload di github



The screenshot shows a GitHub repository page for 'percobaan'. At the top, it says 'percobaan' (Public). Below that, there are buttons for 'master' (selected), '1 Branch', '0 Tags', 'Go to file' (with a 't' filter), 'Add file', and 'Code'. A search bar is also present. The main area shows a commit by 'Wirooo03' with the message 'Push something something'. The commit was made at '99e7e6d · 3 minutes ago' and includes '1 Commit'. Below the commit, four files are listed: '.idea', 'src', '.gitignore', and 'percobaan.iml', each with the same 'Push something something' message and '3 minutes ago' timestamp.

File	Message	Time
.idea	Push something something	3 minutes ago
src	Push something something	3 minutes ago
.gitignore	Push something something	3 minutes ago
percobaan.iml	Push something something	3 minutes ago

### TIPS

Tutorial singkat git bash:

[Video tutorial](#)

**CODELAB & TUGAS****CODELAB**

Buatlah program Java sederhana yang meminta pengguna untuk memasukkan **data diri** mereka, dengan spesifikasi sebagai berikut:

1. Gunakan **Scanner** untuk membaca input dari pengguna dan pastikan program **menutup Scanner** setelah selesai.
2. **Input** yang diminta:
  - Nama (**String**)
  - Jenis Kelamin (**hanya dalam bentuk huruf 'P' atau 'L'**)
  - Tahun Lahir (**Integer**)
3. Hitung **umur pengguna** berdasarkan **tahun lahir** yang dimasukkan. Tahun saat ini harus didapatkan menggunakan Java API **java.time.LocalDate**.
4. Aturan untuk jenis kelamin:
  - Jika pengguna memasukkan '**L**' atau '**l**', maka output yang ditampilkan adalah "**Laki-laki**"
  - Jika pengguna memasukkan '**P**' atau '**p**', maka output yang ditampilkan adalah "**Perempuan**"
5. Tampilkan **Nama, Jenis Kelamin, dan Umur Pengguna**.
6. Contoh **output** yang diharapkan:

```
Masukkan nama: Wira Yudha
Masukkan jenis kelamin (P/L): L
Masukkan tahun lahir: 2004

Data Diri:
Nama : Wira Yudha
Jenis Kelamin: Laki-laki
Umur : 21 tahun

Process finished with exit code 0
```

## **TUGAS 1**

Buatlah program Java yang menerapkan konsep **percabangan (if-else)** dan **Scanner** untuk membuat sistem **login sederhana**. Program ini harus memiliki fitur sebagai berikut:

1. Memilih Jenis **Login**

- Program harus menampilkan pilihan **login**:
  1. Admin
  2. Mahasiswa

2. Pilihan **Tidak Valid**

- Jika pengguna memasukkan pilihan yang bukan **1** atau **2**, tampilkan pesan "**Pilihan tidak valid.**".

3. Login **Admin**

- Jika pengguna memilih login sebagai **Admin**, program harus meminta **username** dan **password**.
- **Username** yang valid adalah "**Admin + (3-digit NIM terakhir kalian)**" dan **password** yang valid adalah "**Password + (3-digit NIM terakhir kalian)**".
- Contoh:
  1. **Username** = Admin010
  2. **Password** = password010
- Jika **username** dan **password** benar, tampilkan pesan "**Login Admin berhasil!**".
- Jika salah, tampilkan pesan "**Login gagal! Username atau password salah.**".

4. Login **Mahasiswa**

- Jika pengguna memilih login sebagai **Mahasiswa**, program harus meminta **Nama** dan **NIM**.
- Nama yang valid adalah "**Nama Kalian**" dan NIM yang valid adalah "**Nim Kalian**".
- Contoh:
  1. **Nama** = Ken Aryo Bimantoro
  2. **NIM** = 202310370311006

- Jika **nama** dan **NIM** benar, tampilkan pesan "**Login Mahasiswa berhasil!**" serta cetak **nama** dan **NIM**.
- Jika **salah**, tampilkan pesan "**Login gagal! Nama atau NIM salah.**"

5. Contoh **output** yang diharapkan:

```
Pilih login:  
1. Admin  
2. Mahasiswa  
Masukkan pilihan: 3  
Pilihan tidak valid.  
  
Process finished with exit code 0
```

```
Pilih login:  
1. Admin  
2. Mahasiswa  
Masukkan pilihan: 1  
Masukkan username: Admin069  
Masukkan password: Password069  
Login Admin berhasil!  
  
Process finished with exit code 0
```

```
Pilih login:  
1. Admin  
2. Mahasiswa  
Masukkan pilihan: 1  
Masukkan username: Admin  
Masukkan password: Password  
Login gagal! Username atau password salah.  
  
Process finished with exit code 0
```

```

Pilih login:
1. Admin
2. Mahasiswa
Masukkan pilihan: 2
Masukkan Nama: Azril Kucai
Masukkan NIM: 202310370311069
Login Mahasiswa berhasil!
Nama: Azril Kucai
NIM: 202310370311069

○ Process finished with exit code 0

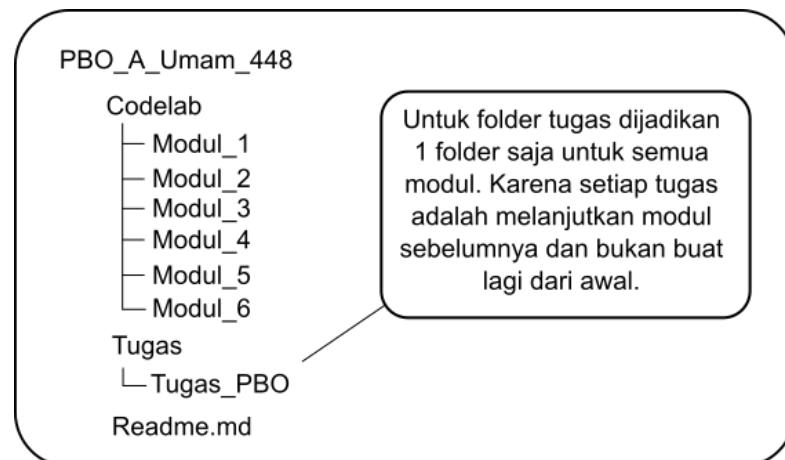
Pilih login:
1. Admin
2. Mahasiswa
Masukkan pilihan: 2
Masukkan Nama: Andre Tahapok
Masukkan NIM: 134i031420380232302303
Login gagal! Nama atau NIM salah.

○ Process finished with exit code 0

```

## TUGAS 2

**Push tugas 1** yang sudah anda buat ke akun **GIT** kalian menggunakan **Git Bash** (didemonstrasikan kepada asisten masing-masing pada saat praktikum) dengan nama Repo: **PBO\_(Kelas)\_(Nama)\_(3-digit NIM terakhir)**. Contoh: **PBO\_A\_Umam\_448**. Berikut adalah contoh struktur file untuk repo kalian:



Setelah itu list link **repository** kalian pada spreadsheet berikut:  
<https://docs.google.com/spreadsheets/d/1Y2DsBETV9JWLPKKI-UVVINhgrWvOUTgJee-AvqWYQHs/edit?usp=sharing>

Catatan:

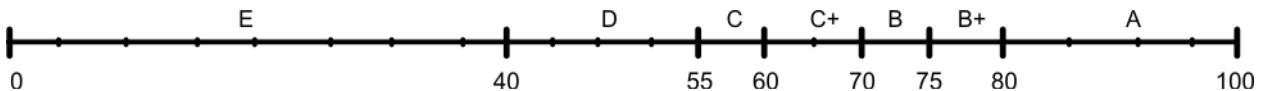
- perhatikan kelas, jangan salah list di kelas lain.
- Dilarang mengganti repository yang kalian list diawal dengan repository yang baru. Repository di awal akan dipakai sampai akhir modul.

## PENILAIAN

### RUBRIK PENILAIAN

Aspek Penilaian	Poin
<b>CODELAB</b>	<b>Total 25%</b>
Kerapian kode	5%
Ketepatan kode & output	10%
Kekreativitasan kode	5%
Kode orisinal (tidak nyontek)	5%
<b>TUGAS 1</b>	<b>Total 50%</b>
Kerapian kode	5%
Ketepatan kode & output	10%
Kode orisinal (tidak nyontek)	5%
Kemampuan menjelaskan	15%
Menjawab pertanyaan	15%
<b>TUGAS 2</b>	<b>Total 25%</b>
Berhasil push ke GITHUB	15%
Kemampuan menjelaskan	10%

TOTAL	100%
-------	------

SKALA PENILAIAN

**A** = (81 - 100) → Sepuh

**B+** = (75 - 80) → Sangat baik

**B** = (70 - 74) → Baik

**C+** = (60 - 69) → Cukup baik

**C** = (55 - 59) → Cukup

**D** = (41 - 54) → Kurang

**E** = (0 - 40) → Bro really...

## SUMMARY AKHIR MODUL

Selamat untuk kalian yang telah menyelesaikan modul ini dengan usaha dan kerja keras! Konsistensi dan kejujuran kalian dalam mengerjakan tugas tanpa menyontek menunjukkan karakter yang luar biasa. Belajar bukan sekadar mencari jawaban, tetapi memahami proses dan berkembang dari setiap tantangan.

Mudah kan? Iya, masih modul 1, tetap semangat sampai modul 6 ya! wkwkw. Terus pertahankan semangat ini, karena di dunia nyata, yang dihargai bukan hanya hasil akhir, tetapi juga usaha yang kalian lakukan untuk mencapainya. Jika ada kesulitan, jangan ragu untuk bertanya dan berdiskusi. Kita semua sedang dalam perjalanan belajar yang sama.

Tetap semangat dan sampai jumpa di modul berikutnya!