

VERSION 1.2

13 Februari 2024



# PEMROGRAMAN BERORIENTASI OBJEK

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

DISUSUN OLEH:

TAUFIQ RAMADHAN

SUTRISNO ADIT PRATAMA

DIAUDIT OLEH:

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

PRESENTED BY: TIM LAB. IT

UNIVERSITAS MUHAMMADIYAH MALANG

## PEMROGRAMAN BERORIENTASI OBJEK

---

### TUJUAN

1. Mahasiswa dapat memahami apa itu java
  2. Mahasiswa dapat memahami konsep Version Control System
- 

### TARGET MODUL

1. Mahasiswa dapat menulis dan memahami syntax dasar dari java
  2. Mahasiswa dapat menggunakan Version Control System menggunakan Git dan Github
- 

### PERSIAPAN

1. Java Development Kit.
  2. Text Editor / IDE (Visual Studio Code, Netbeans, IntelliJ IDEA, atau yang lainnya).
- 

### KEYWORDS

- Java basic
- Java syntax
- Git
- Github
- IntelliJIDEA
- Version Control System

---

## TEORI

- **Apa itu Java**

Java adalah bahasa pemrograman berorientasi objek 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.

- **Tipe data**

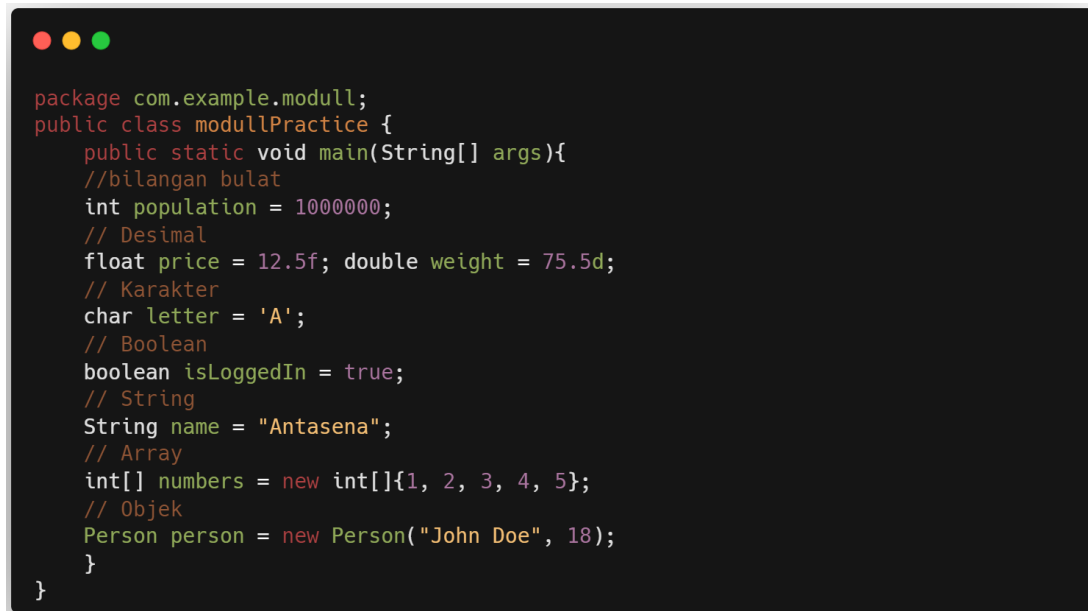
Tipe data di java terdapat 2 jenis yaitu tipe data primitif dan tipe data non-primitif. Tipe data primitif menentukan ukuran dan tipe nilai variabel tanpa memerlukan metode tambahan. Sedangkan, tipe data non-primitif disebut juga tipe referensi karena tipe data ini merujuk pada sebuah objek.

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:



```
package com.example.modul1;
public class modulPractice {
    public static void main(String[] args){
        //bilangan bulat
        int population = 1000000;
        // Desimal
        float price = 12.5f; double weight = 75.5d;
        // Karakter
        char letter = 'A';
        // Boolean
        boolean isLoggedIn = true;
        // String
        String name = "Antasena";
        // Array
        int[] numbers = new int[]{1, 2, 3, 4, 5};
        // Objek
        Person person = new Person("John Doe", 18);
    }
}
```

Perhatikan pada kode di atas, untuk membuat sebuah array juga bisa dengan menulis seperti ini:

```
int[] numbers = {1, 2, 3, 4, 5};
```

atau seperti ini:

```
int numbers[] = {1, 2, 3, 4, 5};
```

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 sumber lain.

- **Input**

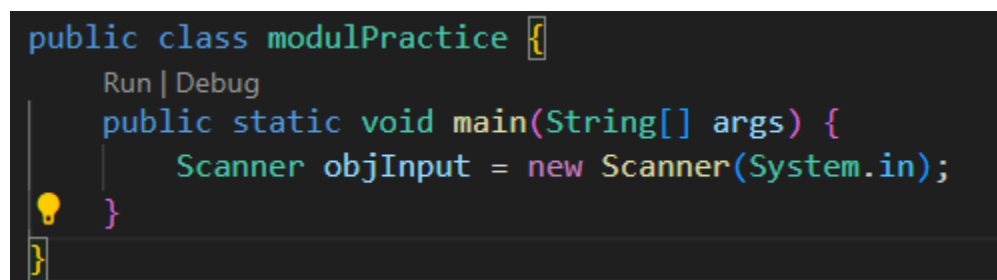
Semua bahasa pemrograman sudah menyediakan fungsi-fungsi untuk melakukan input dan output. Java sendiri telah menyediakan tiga class untuk mengambil input dari user, yaitu:

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 {  
    Run | Debug  
    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 (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 variabel yang kita gunakan (termasuk instance objek) untuk class Scanner, **new Scanner()** itu artinya kita membuat sebuah objek baru dari class Scanner dan referensinya adalah Scanner itu sendiri, dan terakhir **System.in** yang dimaksudkan untuk mengambil sebuah input dari user melalui keyboard ketika program berjalan.

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

```

package com.example.modul1;
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:

- variabel `firstName` di atas dideklarasikan dengan tipe data `String`
- variabel `age` dideklarasikan dengan tipe data `int`
- **`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.
- **`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:

- `next()` : digunakan untuk input sebuah `String` tetapi hanya membaca sampai dengan spasi terakhir
- `nextDouble()` : digunakan untuk input sebuah nilai `double`
- `nextFloat()` : digunakan untuk input sebuah nilai `float`
- `nextByte()` : digunakan untuk input sebuah nilai berupa `byte`
- `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.println()
3. 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:

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

Output yang akan ditampilkan akan seperti ini:

```
PS C:\Users\radan> cd "c:\Users\radan\Music\" ; if ($?) { javac modulPractice.java } ; if ($?) { java modulPractice }
ini teks yang dicetak dengan print()sedangkan ini teks yang dicetak dengan println()
pake print() lagi
PS C:\Users\radan\Music>
```

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 = "Adit";
    int umur = 30;
}
```

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 = "Adit";
        int umur = 30;
        System.out.println(nama);
        System.out.print(umur);
    }
}
```

Kode di atas akan menghasilkan output:

**Adit**

**30**

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 ditengahnya 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 = "Adit";
        int umur = 30;
        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 selengkapnya bisa cek [disini](#)

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

```
PS C:\Users\radan> cd "c:\U
Nama saya Adit umur 30
```

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

```
package com.example;
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);
    }
}

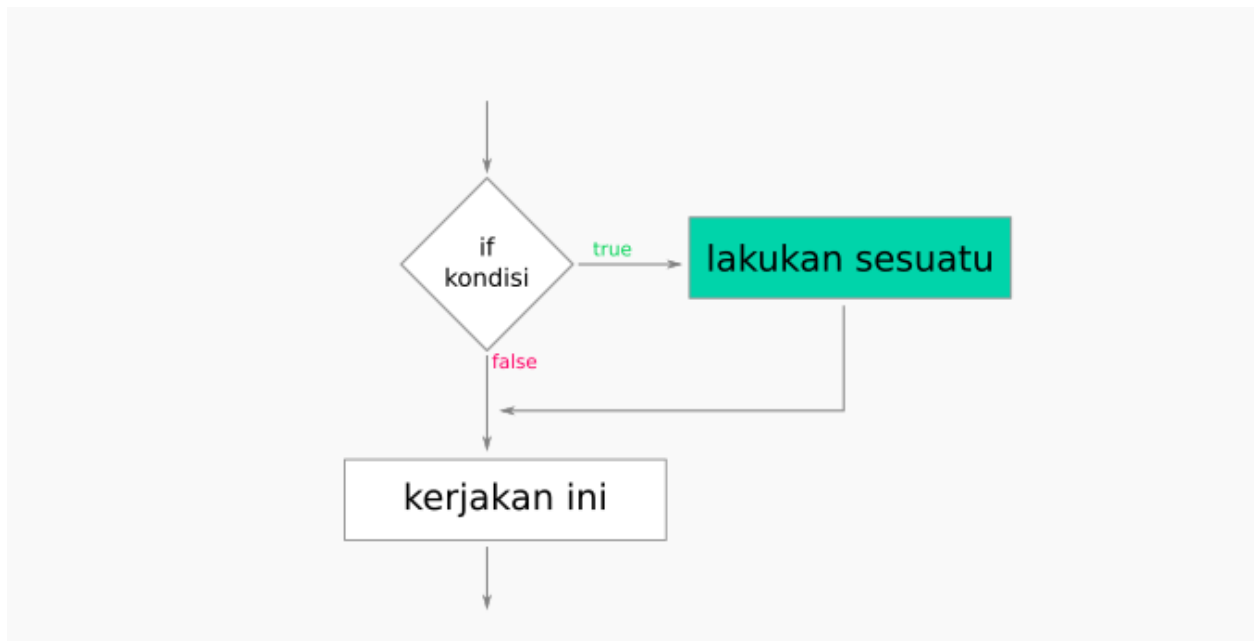
// output
(antasena@GLM4deu)
$ ./run
Masukkan Nama Anda : Khoir
Masukkan Umur Anda : 19
Nama : Khoir
Umur : 19
```

- **Conditions (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.

- Percabangan 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:

```

package com.example;

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

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

// output

(antasena@GLM4deu)
$ ./run
Angka lebih besar dari 5

```

- Percabangan 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:

```

package com.example;

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

(antasena@GLM4deu)
$ ./run
Angka tidak lebih besar dari 5

```

- Percabangan 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:

```
● PS C:\Users\radan> cd "c:\U
Inputkan nilai: 66
Grade: C+
○ PS C:\Users\radan\Music> █
```

- Percabangan 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:

```
package com.example;
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

(antasena@GLM4deu)
$ ./run
Warna Merah
```

- Percabangan bersarang (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 kodenya:

```
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.

- **Perulangan 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:



```
package com.example;

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

// output

(antasena@GLM4deu)
$ ./run
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
```

#### ▪ Perulangan 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 + " ");
        }
    }
}
```



```

    }
}
}

```

- While loop

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:



```

package com.example;

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

// output

(antasena@GLM4deu)
$ ./run
Iterasi ke-0
Iterasi ke-1
Iterasi ke-2
Iterasi ke-3
Iterasi ke-4

```

- 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:



```
package com.example;

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

// output

(antasena@GLM4deu)
$ ./run
Iterasi ke-0
Iterasi ke-1
Iterasi ke-2
```

## • Java API

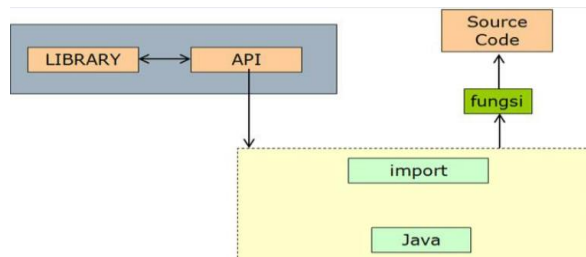
Java Application Programming Interface (Java API) adalah komponen-komponen dan kelas Java yang sudah disediakan oleh Java dan memiliki berbagai kemampuan. Kemampuan pada java API bermacam-macam, mulai dari menangani objek, string, angka, dan sebagainya. Java API merupakan seperangkat method yang disediakan 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.

Bagian API dalam JAVA:

- 1) API standard (Java SE) yang digunakan untuk aplikasi dan applet dengan layanan bahasa dasar
- 2) API enterprise (Java EE) untuk aplikasi server dengan layanan database dan aplikasi server-side (servlet)

### 3) API untuk device micro (Java ME) seperti handphone

#### Schema Java API



Adapun contoh untuk penggunaan Java API, seperti ini:

```

package com.example;
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));
    }
}

\\ output

(arisu@GLM4deu)-[~]
$ ./run
Waktu Sekarang Adalah : 2024-01-03

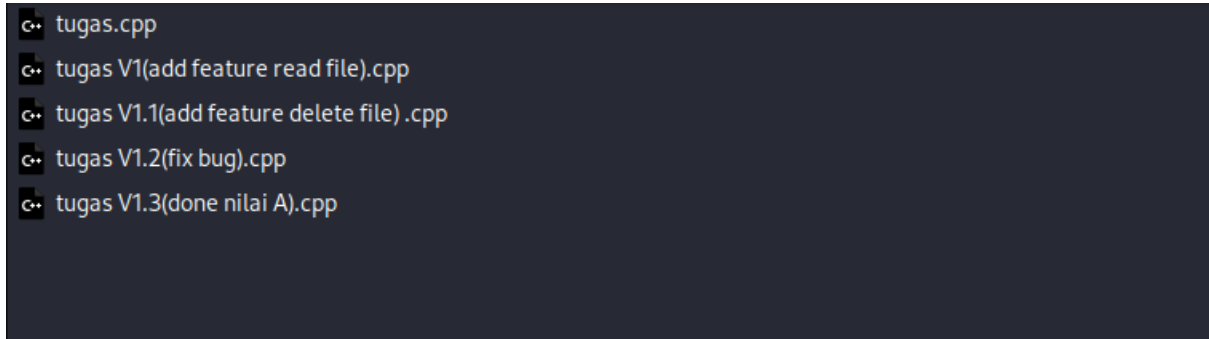
```

Pada kode di atas kita menggunakan sebuah API java.time yang dimana API ini digunakan untuk mendapatkan waktu sekarang dan memformatnya menjadi tanggal dengan pola "yyyy-MM-dd". Secara sederhana java API adalah sebuah library yang sudah disediakan oleh Java.

- GIT

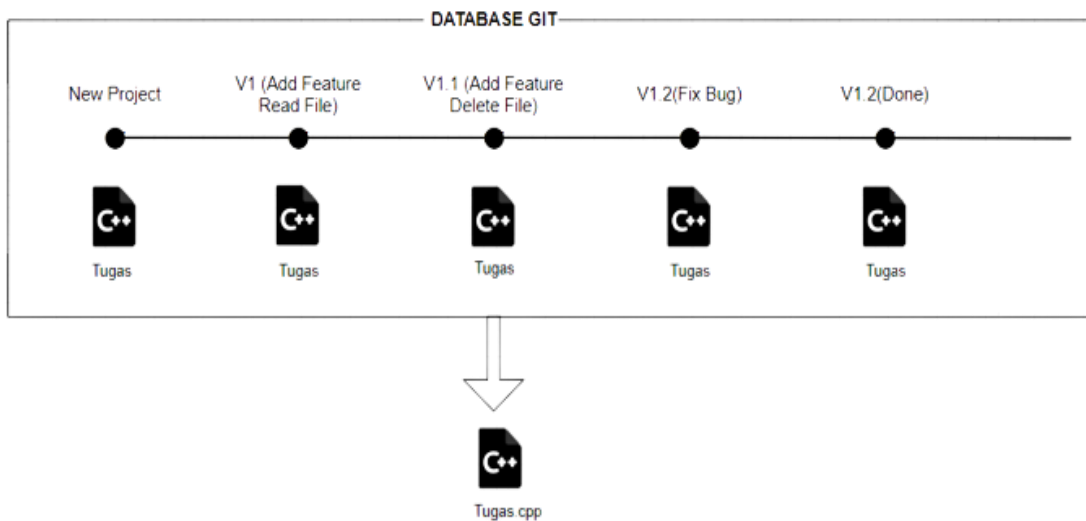


Git adalah sebuah software berbasis Version Control System (VCS) yang digunakan untuk mencatat perubahan seluruh file atau repository suatu project yang kita buat. Developer software biasanya menggunakan Git untuk distributed revision (VCS terdistribusi), hal ini bertujuan untuk menyimpan database tidak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



Lebih jelasnya bisa lihat pada contoh kasus 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 pengerjaan 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
1. Meng- <i>install software</i> di penyimpanan lokal	1. <i>Host</i> melalui layanan <i>cloud</i>
2. Dikelola oleh The Linux Foundation	2. Diakuisisi oleh Microsoft pada 2018
3. Berfokus pada <i>version control</i> dan <i>code sharing</i>	3. Berfokus pada <i>source code hosting</i> terpusat
4. Akses secara <i>offline</i>	4. Akses secara <i>online</i>
5. Tidak menggunakan fitur <i>user management</i>	5. Menggunakan <i>user management</i>
6. Menyediakan <i>desktop interface</i> bernama "Git GUI"	6. Menggunakan nama <i>desktop interface</i> "GitHub Desktop"
7. Bersaing dengan Mercurial, Subversion, IBM, Rational Team, Concert, dan ClearCase	7. Bersaing dengan GitLab dan Atlassian BitBucket
8. <i>Open sourced licensed</i>	8. Pilihan bagi pengguna gratis dan pengguna berbayar

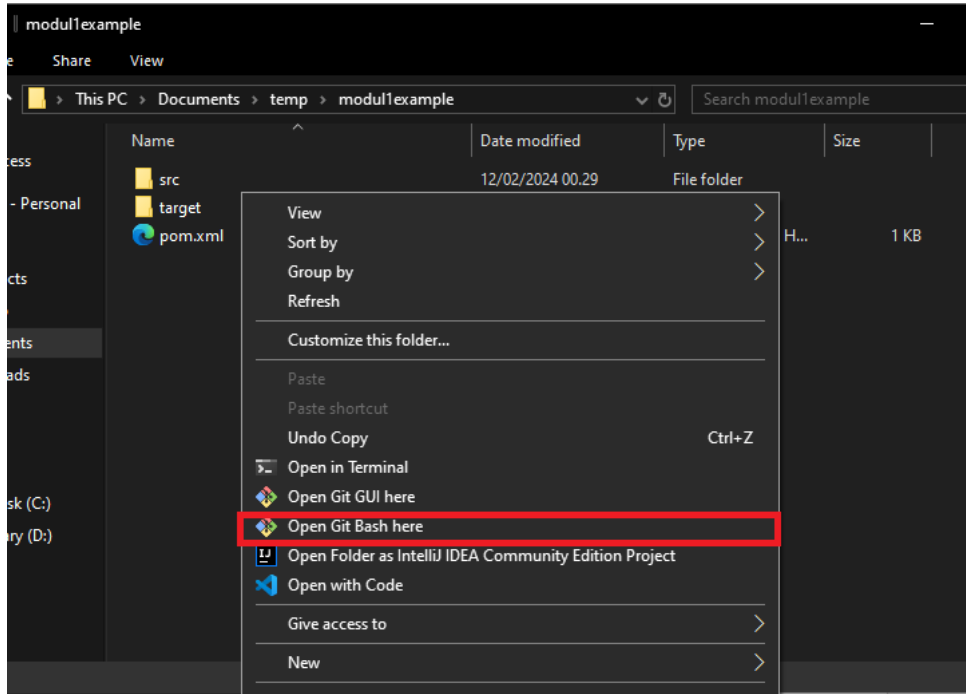
- **Cara upload source code ke repository Github dengan Git**

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



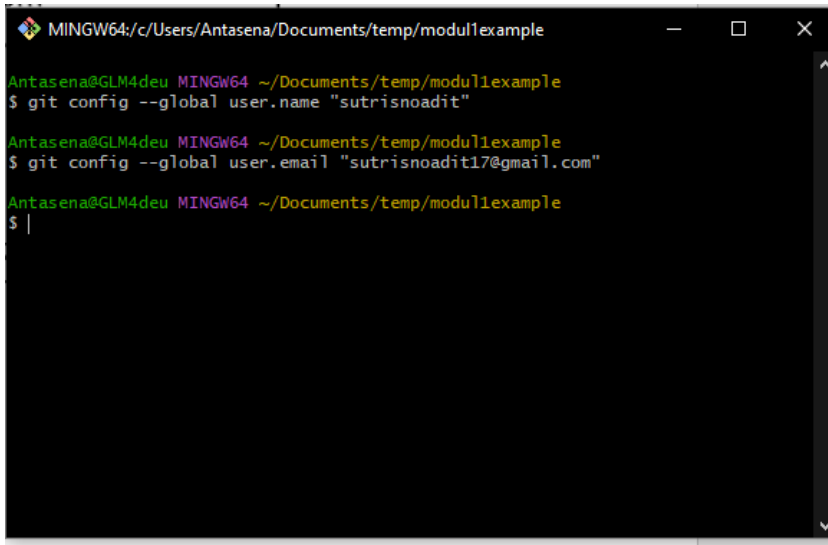
- 2) Masukkan user dan password menggunakan perintah berikut:

```
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:



```

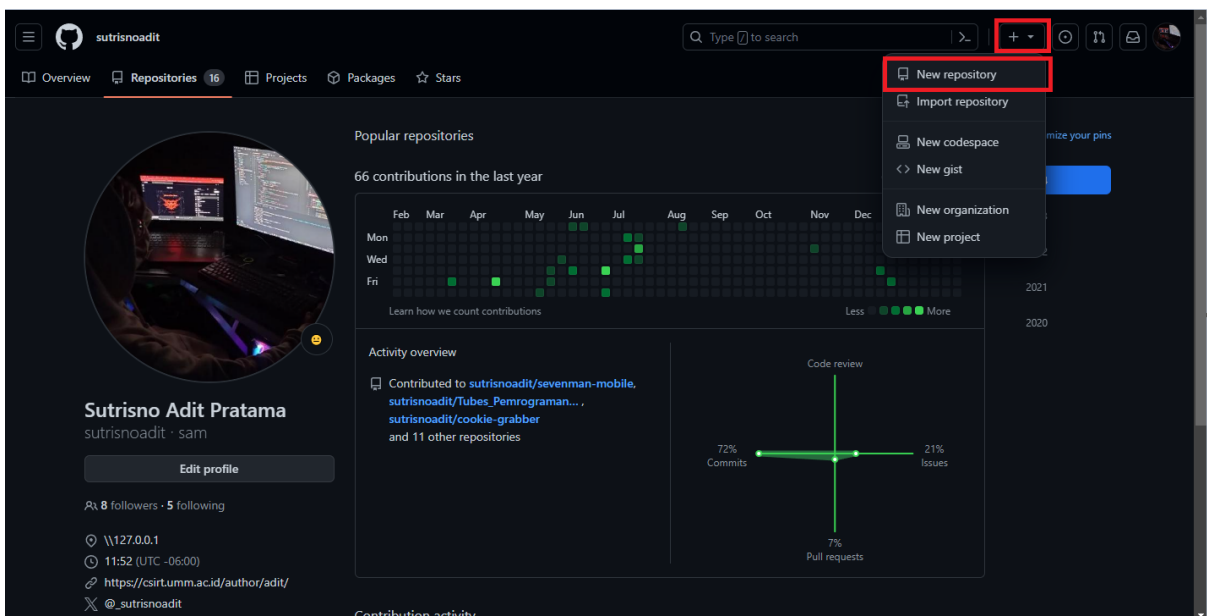
MINGW64:/c/Users/Antasena/Documents/temp/modul1example
Antasena@GLM4deu MINGW64 ~/Documents/temp/modul1example
$ git config --global user.name "sutrisnoadit"

Antasena@GLM4deu MINGW64 ~/Documents/temp/modul1example
$ git config --global user.email "sutrisnoadit17@gmail.com"

Antasena@GLM4deu MINGW64 ~/Documents/temp/modul1example
$ |

```

- 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.




## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (\*).


Owner \* Repository name \*


 sutrisnoodit / Percobaan PBO 1

✓ Your new repository will be created as Percobaan-PBO-1.  
The repository name can only contain ASCII letters, digits, and the characters -, ., and \_.

Great repository names are short and memorable. Need inspiration? How about [redesigned-eureka](#) ?

Description (optional)

☒  **Public**  
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**  
You choose who can see and commit to this repository.

Initialize this repository with:

☐ **Add a README file**  
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore


.gitignore template: **None**

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: **None**

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

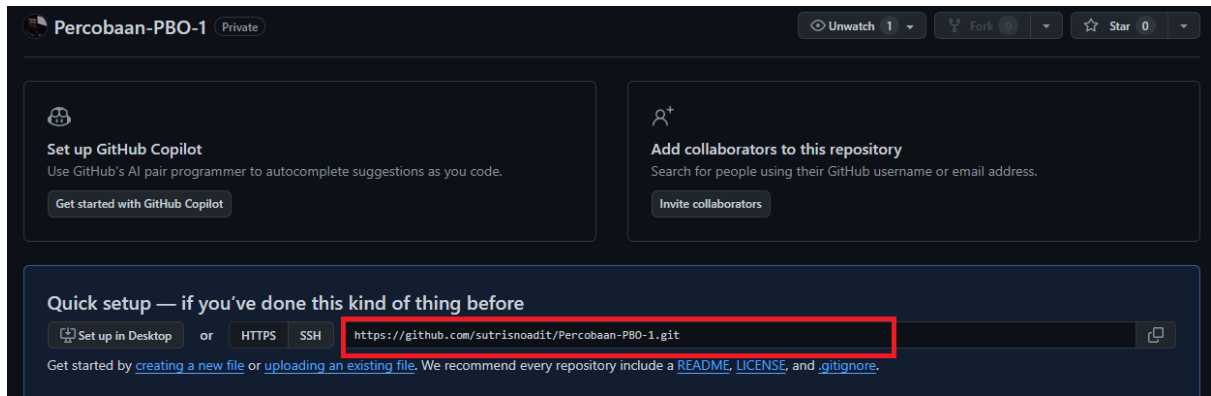
 You are creating a public repository in your personal account.

[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
- *Checkbox initialize this repository with a README* bisa dicentang atau tidak. Karena hanya berpengaruh pada pembuatan berkas README.

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>**

```

Antasena@GLM4deu MINGW64 ~/Documents/temp/modullexample
$ git config --global user.name "sutrisnoadit"

Antasena@GLM4deu MINGW64 ~/Documents/temp/modullexample
$ git config --global user.email "sutrisnoadit17@gmail.com"

Antasena@GLM4deu MINGW64 ~/Documents/temp/modullexample
$ git init

Antasena@GLM4deu MINGW64 ~/Documents/temp/modullexample (master)
$ git add.

Antasena@GLM4deu MINGW64 ~/Documents/temp/modullexample (master)
$ git commit -m "first commit"
[master (root-commit) 58355e6] first commit
4 files changed, 28 insertions(+)
create mode 100644 pom.xml
create mode 100644 src/main/java/com/example/Main.class create mode 100644 src/main/java/com/example/Main.java create mode 100644 target/classes/com/example/Main.class

Antasena@GLM4deu MINGW64 ~/Documents/temp/modullexample (master)
$ git branch -M main

Antasena@GLM4deu MINGW64 ~/Documents/temp/modullexample (main)
$ git remote add origin https://github.com/sutrisnoadit/Percobaan-PBO-1.git

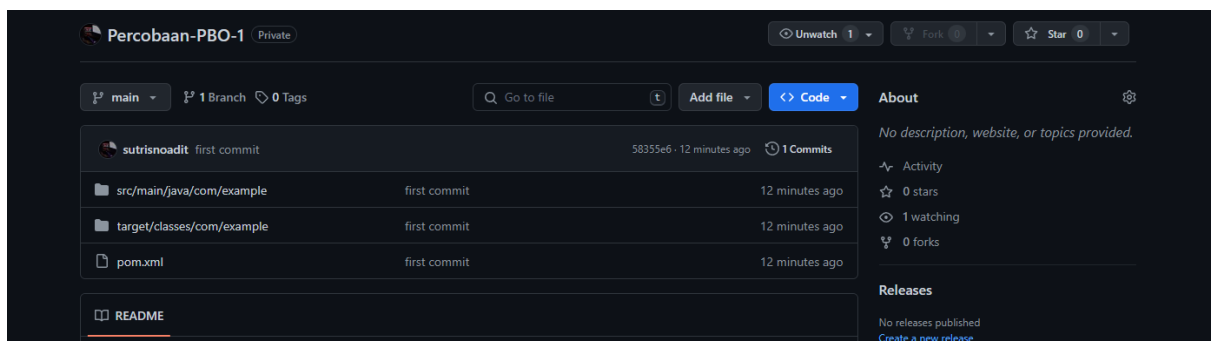
Antasena@GLM4deu MINGW64 ~/Documents/temp/modullexample (main)
$ git push -u origin main
Enumerating objects: 15, done.
Counting objects: 100% (15/15), done.
Delta compression using up to 12 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (15/15), 2.15 KiB | 2.15 MiB/s, done. Total 15 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/sutrisnoadit/Percobaan-PBO-1.git
* [new branch]
main -> main
branch 'main' set up to track 'origin/main'.

```

Catatan:

- Tanda “ . “ 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.

7) Setelah kamu sukses dalam mengunggah berkas ke GitHub, silahkan pergi ke repository.



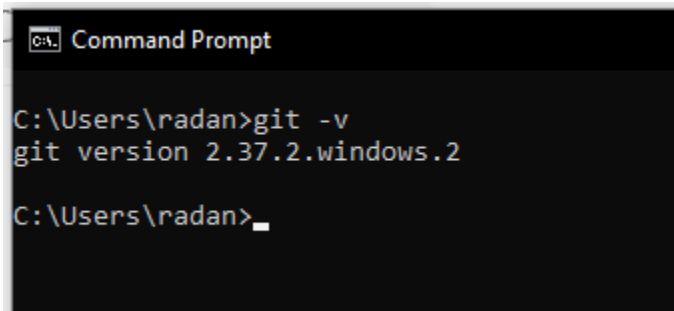
Apabila berkas yang diunggah sudah sesuai, itu artinya sudah berhasil menambahkan berkas ke dalam github.

### ● IntelliJIDEA Built-In Version Control

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

- 1) 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 ketikkan command **git -v**.

**Catatan : Buat repository terlebih dahulu seperti dilangkah 3 dan 4**



```

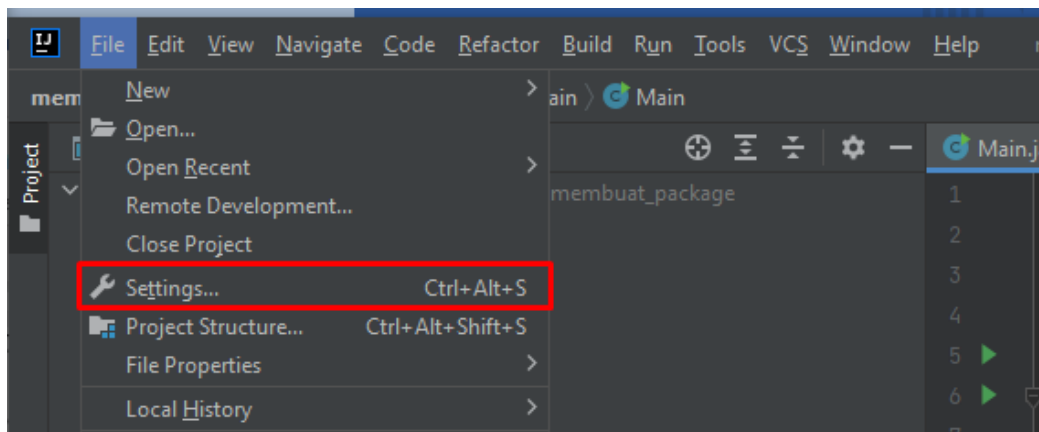
C:\Users\radan>git -v
git version 2.37.2.windows.2

C:\Users\radan>

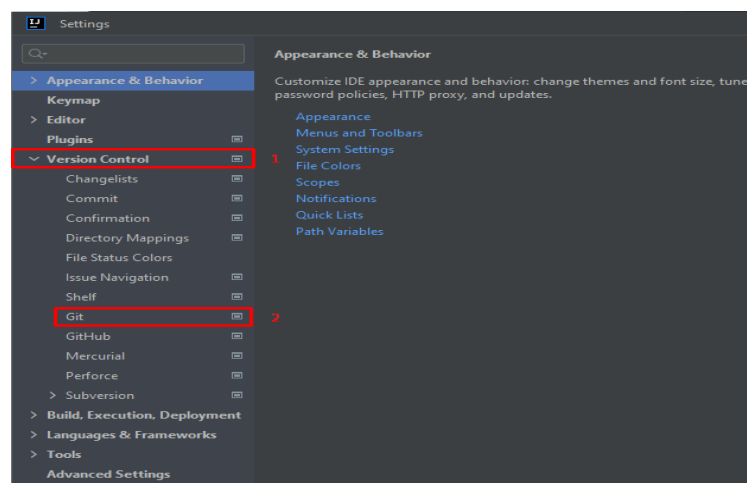
```

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

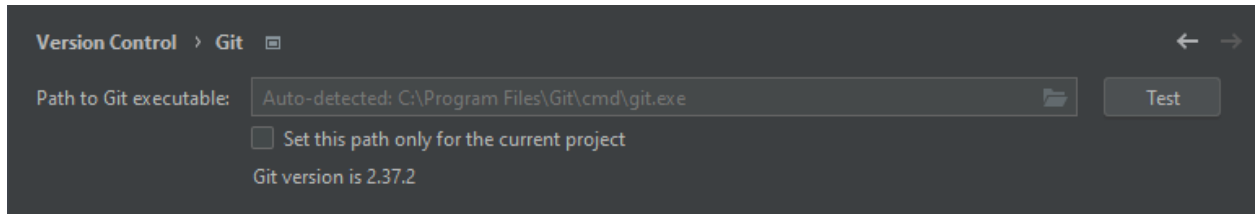
- 2) Buka IntelliJ IDEA dan pergi ke **File** di pojok kiri atas > **Settings**.



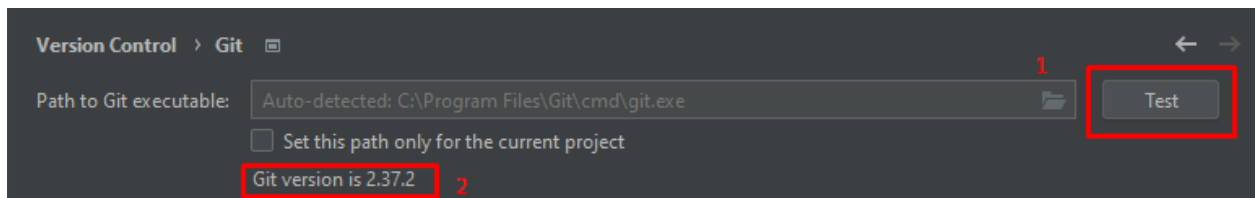
- 3) Pilih opsi **Version Control** > **Git**



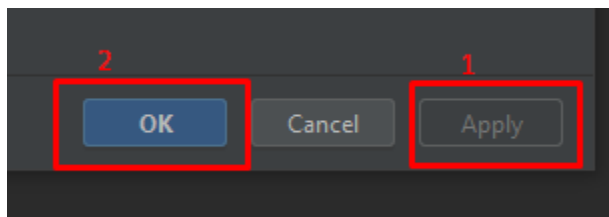
- 4) Pada bagian **Path to Git executable**, spesifikasikan path pada instalasi Git executable (biasanya **git** atau **git.exe** untuk windows). Jika tertulis autodetected seperti ini, itu artinya instalasi Git kita sudah terdeteksi otomatis.



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



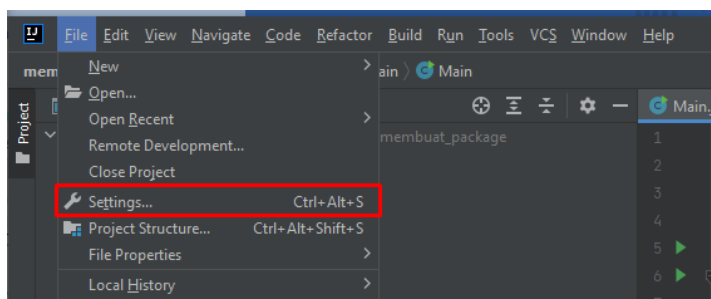
- 6) Klik **apply** dan **OK** untuk menyimpan konfigurasi.



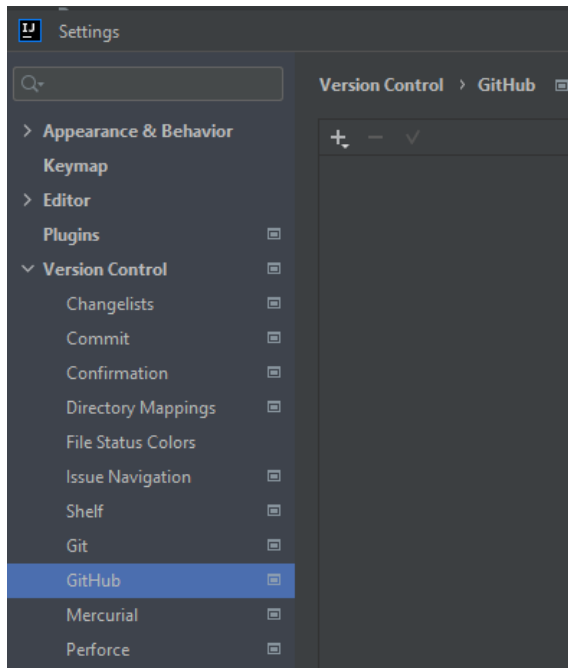
- **Integrasi Github dengan IntelliJIDEA**

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

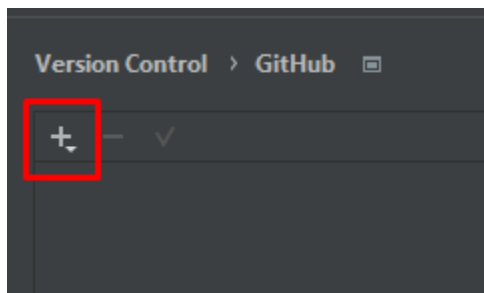
- 1) Pergi ke **File > Settings**



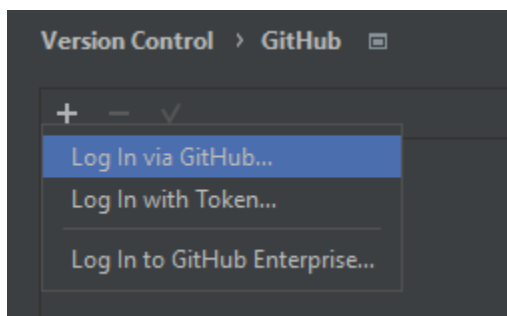
- 2) Navigasikan ke **Version Control > Github**



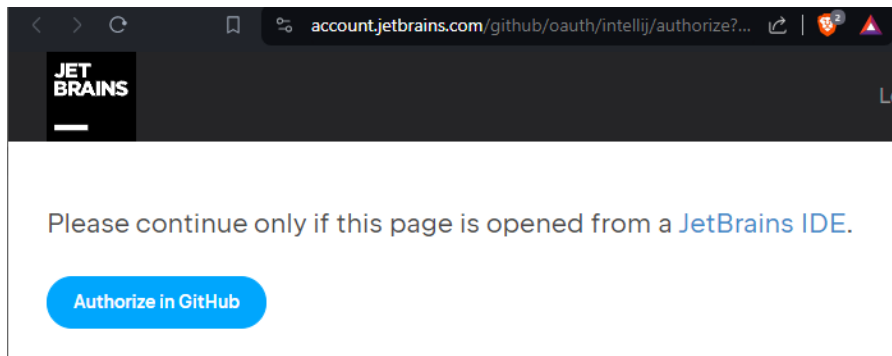
- 3) Tekan icon + untuk menambahkan akun Github



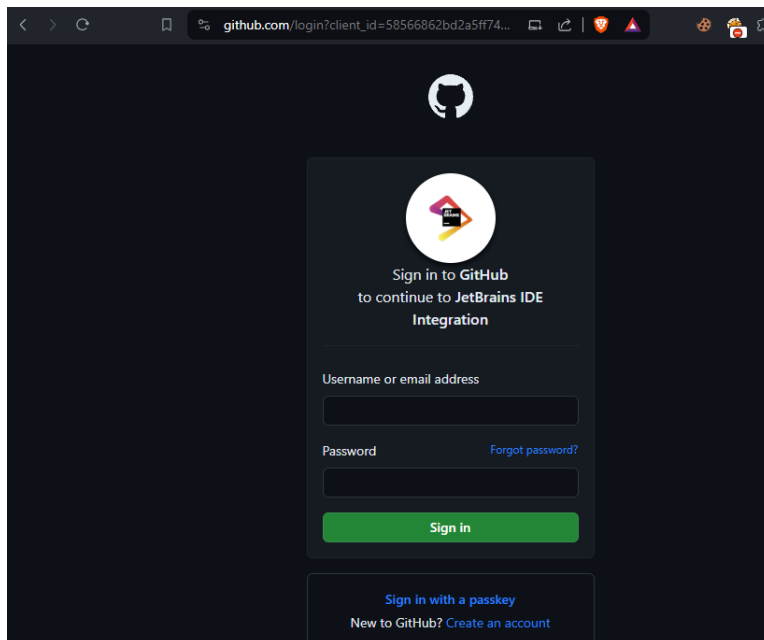
- 4) Pilih akses login melalui Github atau Token, dalam contoh disini akan menggunakan Github



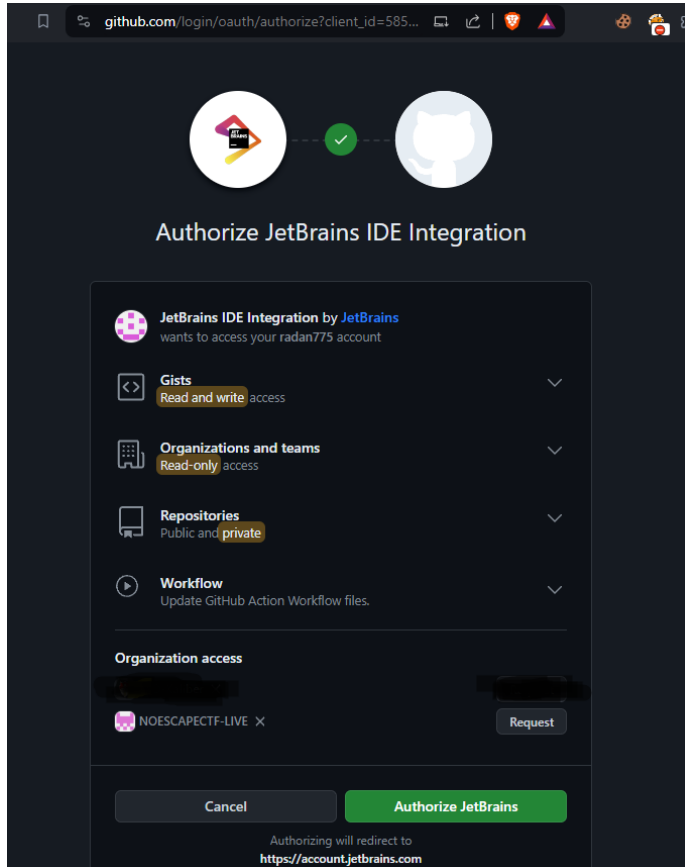
- 5) Ketika menekan **Log In via Github** akan muncul tab browser baru dan klik **Authorize in Github**



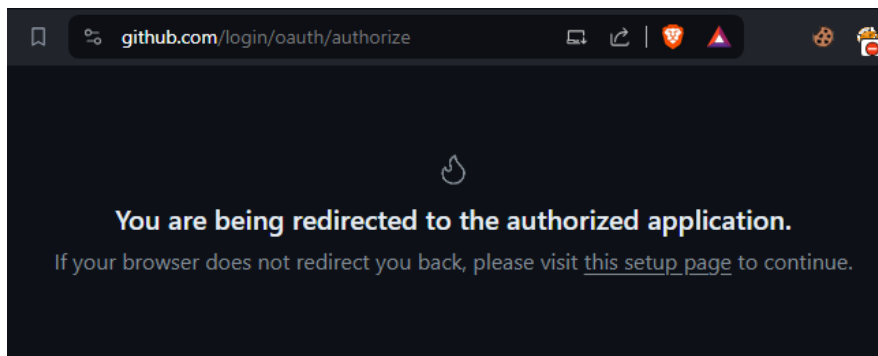
- 6) Isikan kredensial akun sesuai dengan username dan password, lalu klik Sign In.



## 7) Klik Authorize JetBrains

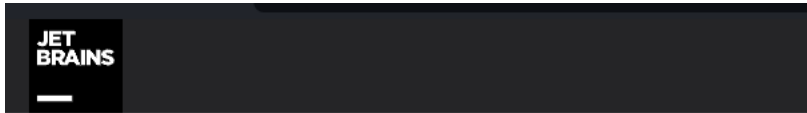


## 8) Ketika muncul tampilan seperti ini harap ditunggu



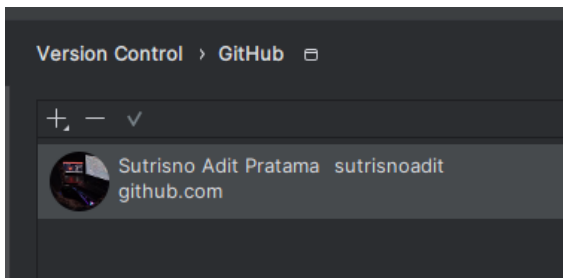


9) Dan akan muncul website dengan tampilan berikut

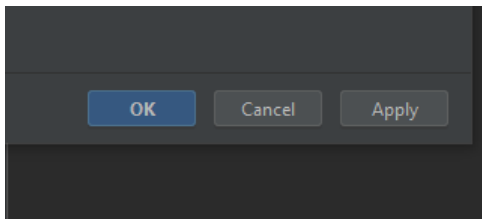


You have been successfully authorized in GitHub. You can close the page.

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



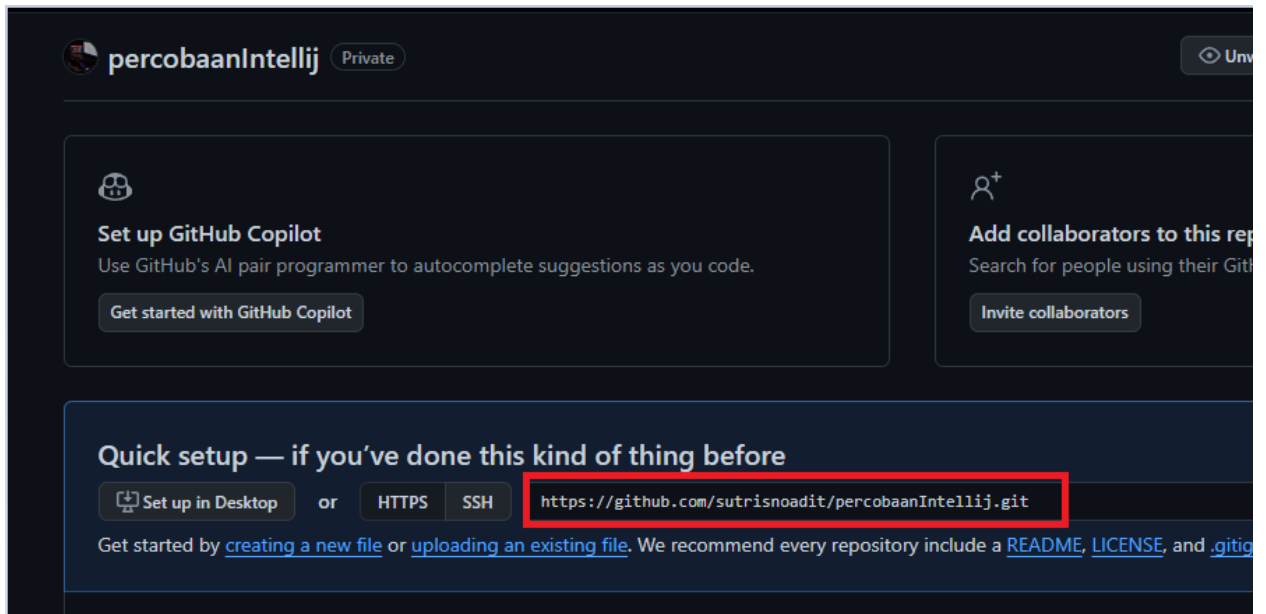
11) Terakhir klik **Apply** dan **OK**



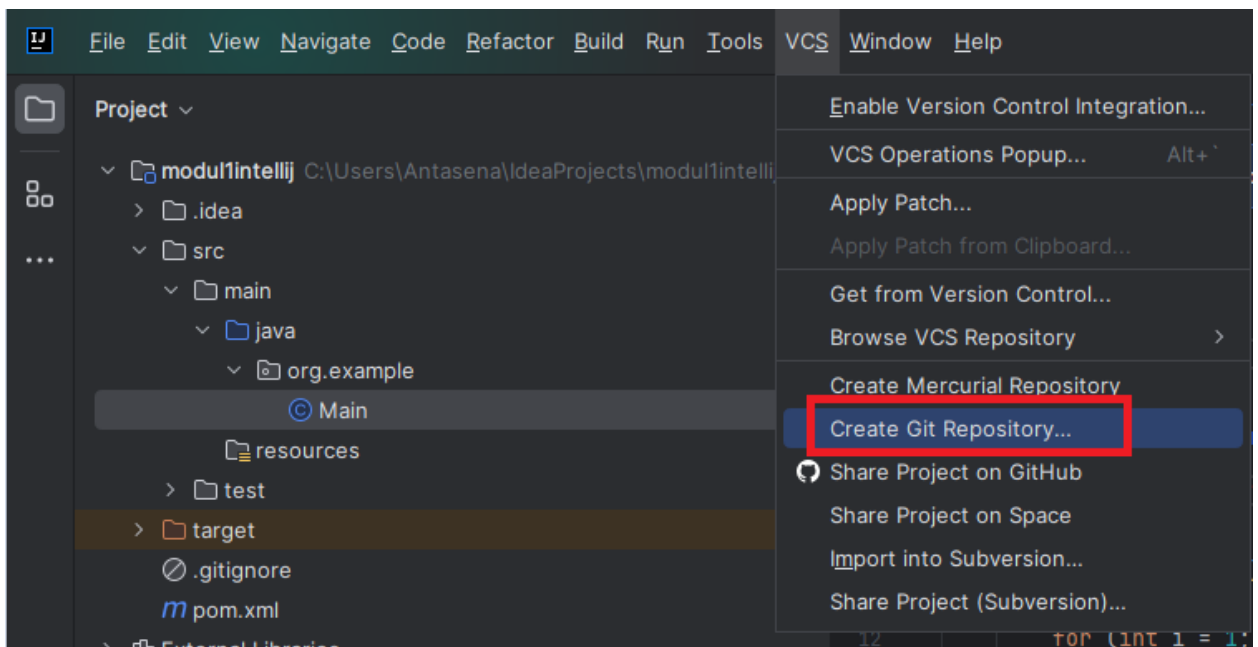
- **Bekerja dengan Git Repository**

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

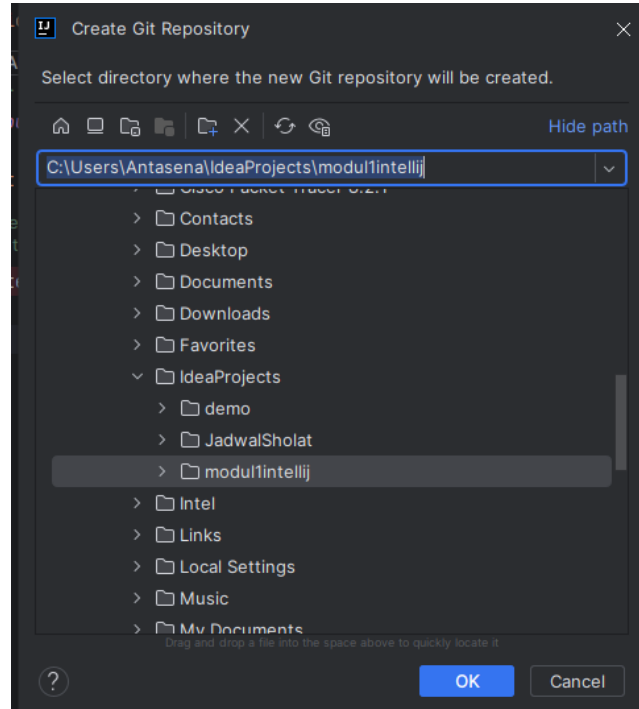
- 1) Buat repository di github seperti Langkah 3 dan 4 pada tutorial git dan github dan copykan url repositorynya



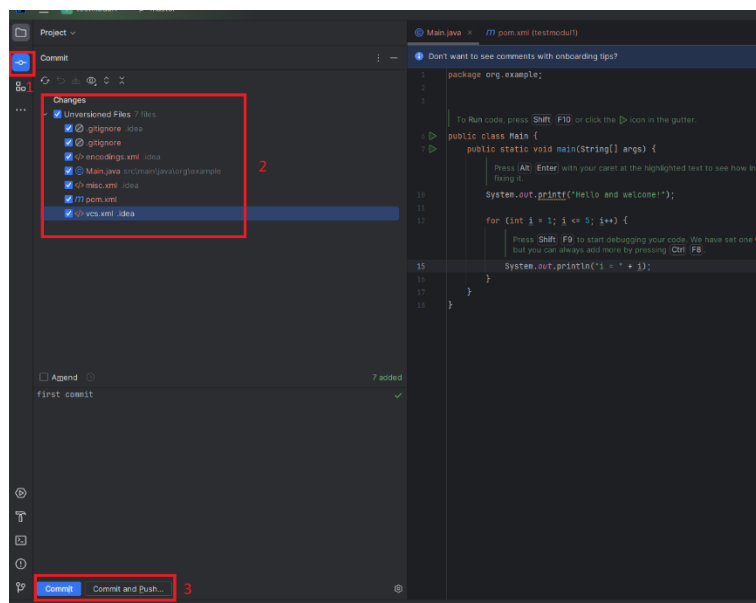
- 2) Untuk membuat repository baru, pergi ke VCS > Get From Version Control



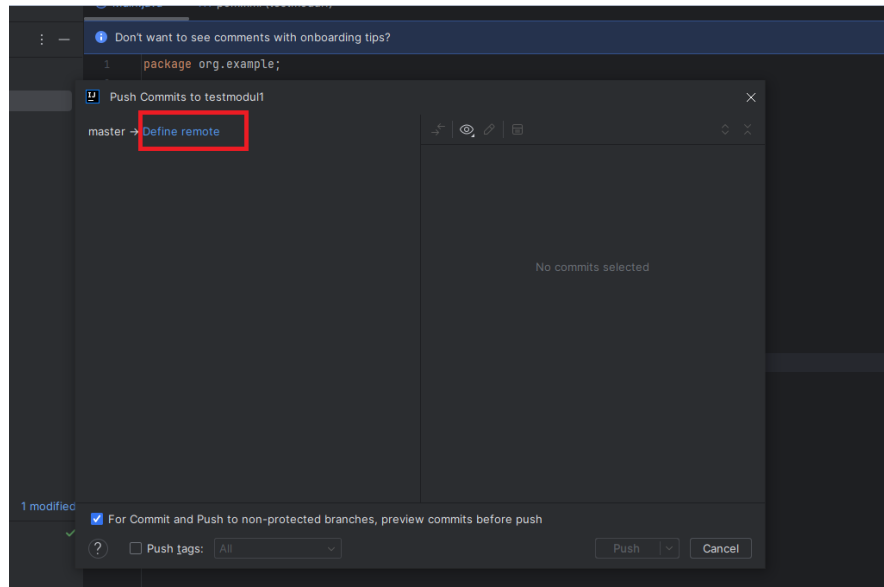
- 3) Pilih directory file yang akan di push ke github lalu klik “ok”



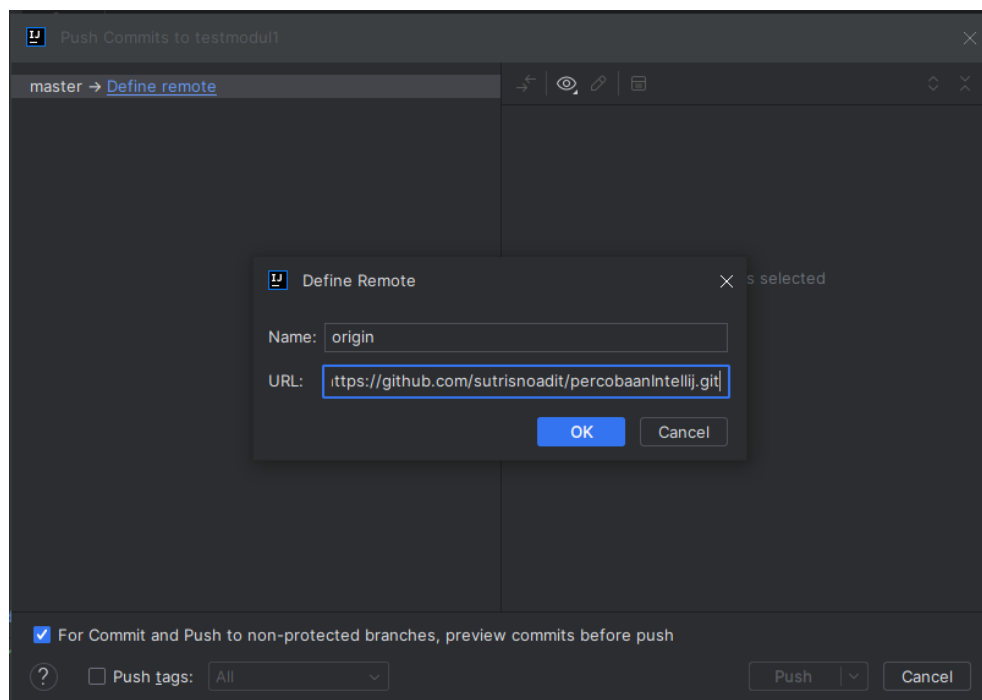
3. Pilih sidebar commit lalu centang semua file yang akan di push ke github > isi kolom yang dibawahnya sebagai keterangan commit contoh “**first commit**” pilih commit / commit and push



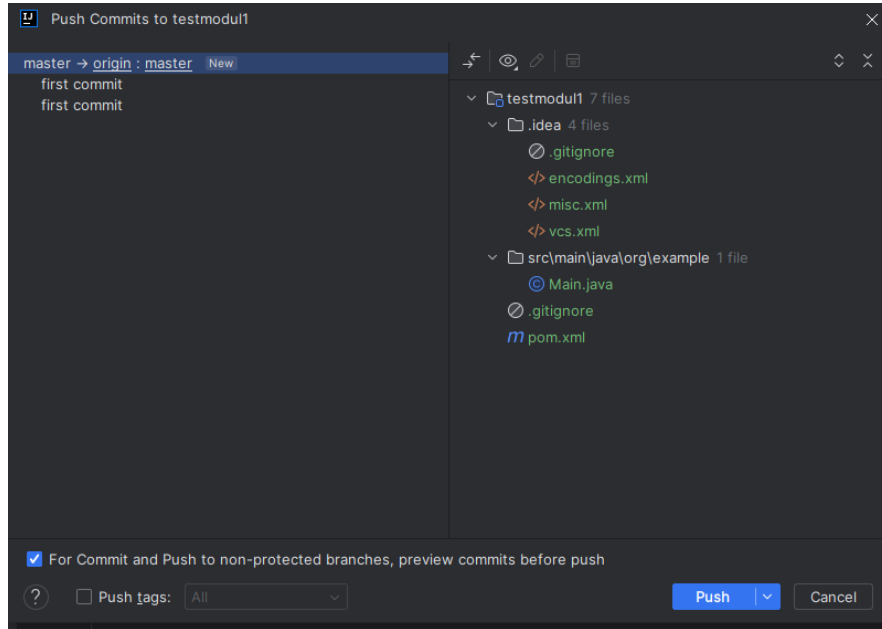
4. setelah melakukan step 3 maka akan keluar jendela baru jika kita pertama melakukan development / pertama kali push dan pilih define remote



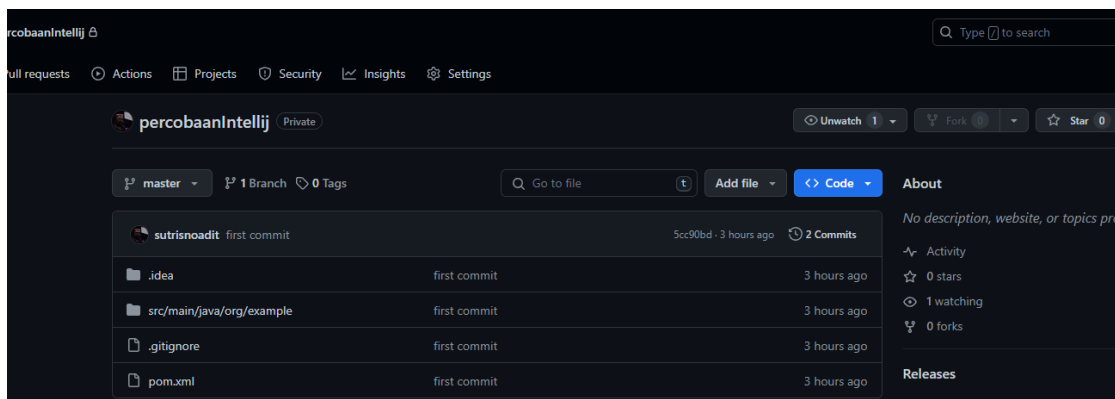
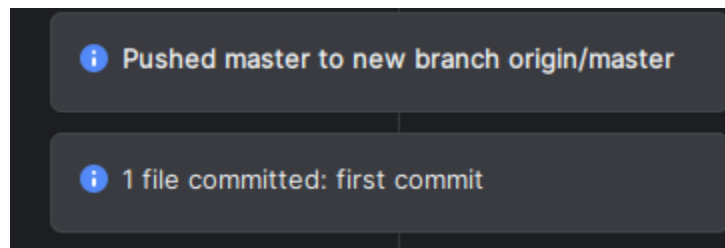
5. untuk form url isi dengan url repository yang telah dibuat di step 1 diatas dan klik ok



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



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



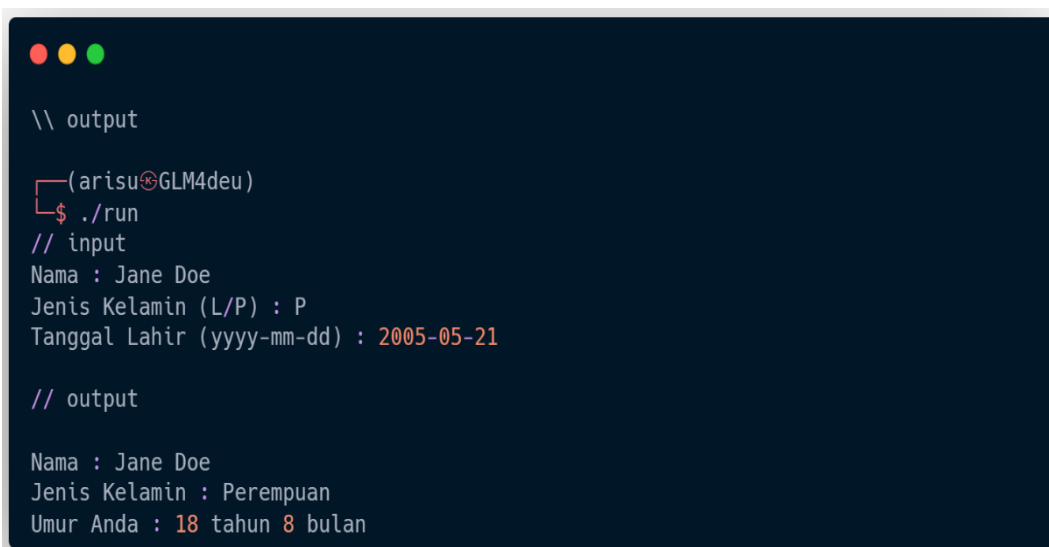
---

## CODELAB

Buatlah program input data diri yang terdiri dari nama, jenis kelamin, tanggal lahir. Dengan spesifikasi berikut:

- Inputan jenis kelamin hanya singkatan ( P atau L ) dan hasil output menampilkan lengkap contoh jika user menginput L maka output laki laki
- Menggunakan java api java.time atau bisa yang lain untuk menghitung umur yang sudah di input di tahun sekarang

Contoh output:



```
\\ output

(arisu@GLM4deu)
$ ./run
// input
Nama : Jane Doe
Jenis Kelamin (L/P) : P
Tanggal Lahir (yyyy-mm-dd) : 2005-05-21

// output

Nama : Jane Doe
Jenis Kelamin : Perempuan
Umur Anda : 18 tahun 8 bulan
```

---

## TUGAS

Buatlah sebuah sistem login library yang dimana terdapat 2 user yaitu admin dan mahasiswa. Berikut penjelasannya:

- User mahasiswa login hanya menggunakan nim dan panjang nim yang diinputkan tidak boleh lebih ataupun kurang dari panjang 15
- User admin login menggunakan username dan password
- Proses upload tugas ke github live didemokan pada saat praktikum (Saat proses upload tugas ke github diwajibkan untuk menggunakan git, tidak boleh upload manual)

Contoh Output:

```

(arisu@GLM4deu)-[~]
$ ./run

===== Library System =====
1. Login as Student
2. Login as Admin
3. Exit
Choose option (1-3): 1
Enter your NIM : 202210370311203
Successful Login as Student
===== Library System =====
1. Login as Student
2. Login as Admin
3. Exit
Choose option (1-3): 1
Enter your NIM : 20022222
User Not Found
===== Library System =====
2. Login as Admin
3. Exit
Choose option (1-3): 2
Enter your username (admin): adm00n
Enter your password (admin): adm
Admin User Not Found!!
===== Library System =====
1. Login as Student
2. Login as Admin
3. Exit
Choose option (1-3): 2
Enter your username (admin): admin
Enter your password (admin): adm1n
Successful Login as Admin
===== Library System =====
1. Login as Student
2. Login as Admin
3. Exit
Choose option (1-3): 3
adios

```

## RUBRIK PENILAIAN

Aspek Penilaian	Poin
Codelab	20
Tugas	30
Pemahaman	50
<b>Total</b>	<b>100%</b>