

MODUL 2. PENGENALAN JAVA, KELAS DAN OBJEK

2.1 Tujuan

1. Mahasiswa mampu memahami dan mengimplementasikan konsep Kelas dan Obyek pada Java
2. Mahasiswa mampu memahami konsep version control

2.2 Alat dan Bahan

1. Visual Studio Code
2. Git,
3. Java versi 21

2.3 Dasar Teori

Nah, akhirnya kita masuk juga ke bagian pemrograman yang sebenarnya! Di sini, kalian akan mulai mengenal, mengetahui, dan menulis kode! Wow! Menyenangkan sekali, bukan? Sebelumnya pastikan kalian sudah menginstal Visual Studio Code, Git, Java versi 21 dan juga akun Github. Kalian akan memulai titik awal perjalanan pemrograman yang tiada akhir. Baiklah, tanpa berlama-lama, mari kita mengenal Java terlebih dahulu, yuk! Mulai dari apa itu Java, kemudian sejarahnya, digunakan untuk apa, lalu yang terakhir, struktur kode pada Java!

2.3.1 Apa itu Java?

Java adalah bahasa pemrograman yang dikembangkan oleh Sun Microsystems pada tahun 1995. Sekarang, Java dimiliki oleh Oracle. Java dirancang dengan tujuan agar mudah dipelajari dan digunakan, serta memiliki kemampuan luar biasa untuk dijalankan di berbagai perangkat/device atau sistem operasi tanpa perlu perubahan besar. Hebat, kan? Jadi Java ini mengikuti prinsip "Write Once, Run Anywhere" (WORA), yang artinya kode yang kalian tulis dalam Java bisa dijalankan di berbagai *platform* tanpa perlu penyesuaian ulang. Jadi, nggak perlu khawatir soal kompatibilitas! Nulis kodenya sekali, jalaninnya berkali-kali!

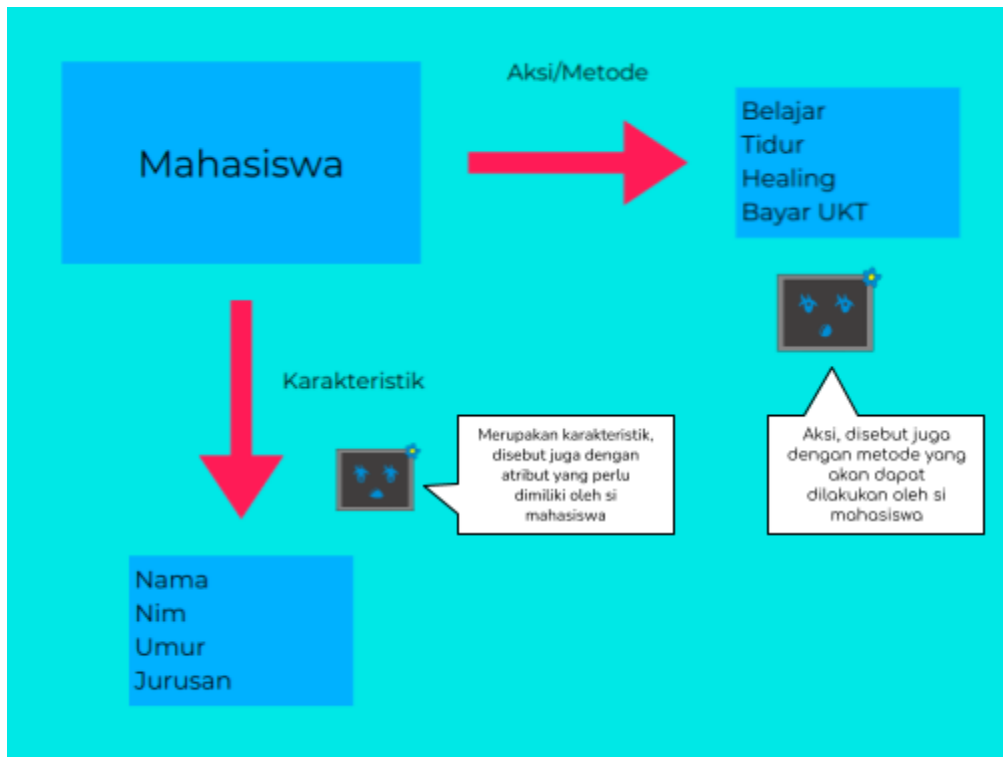
Awalnya, Java dikembangkan oleh James Gosling dan timnya di Sun Microsystems sebagai bagian dari proyek rahasia yang disebut "Green Project". Seru, ya? Pada awalnya, Java didesain untuk perangkat elektronik seperti TV pintar, tapi siapa sangka, Java justru berkembang menjadi bahasa pemrograman serbaguna yang sekarang digunakan untuk berbagai macam aplikasi, mulai dari web, desktop, hingga mobile, loh.

2.3.2 Class

Class? Maksudnya seperti kelas yang ada di sekolah? Bukan dong, hahaha. Namun bisa saja dibuat seperti itu nantinya. Soalnya, kelas pada Java ini adalah sebuah "blueprint" atau cetak biru. Isi dari kelas ini adalah definisi-definisi dari sesuatu yang ingin kalian bangun, lho! Di sini kita hanya akan membahas

tentang pengertian dan definisi dari kelas, ya. Untuk pembuatan kelas itu sendiri pada kode Java akan dibahas di lain waktu.

Sekarang kita akan melihat sebuah contoh dari kelas yang bernama “Mahasiswa”.

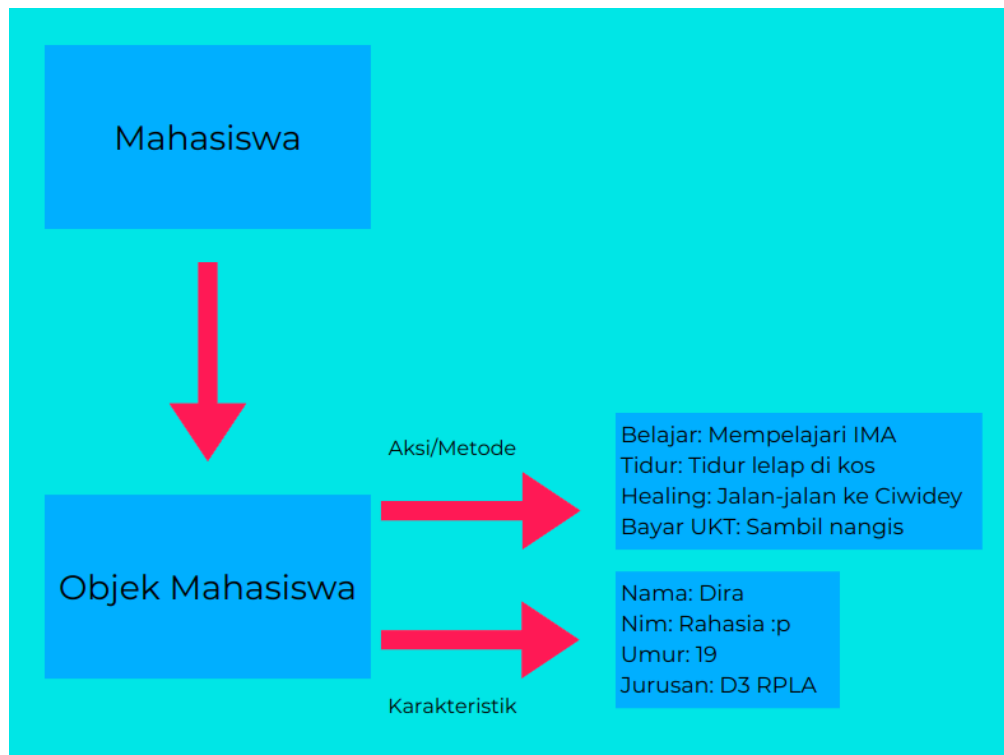


Nah, kita bisa melihat bahwa isi dari kelas “Mahasiswa” di atas hanya berisi definisi-definisi saja. Karakteristik-karakteristik apa yang dimiliki seorang mahasiswa? Kemudian aksi apa saja yang bisa dilakukan oleh seorang mahasiswa? Semua itu kita tuliskan dalam kelas. Dengan begini, kita telah mendefinisikan sebuah kelas! Dalam kelas “Mahasiswa” di atas, ia memiliki karakteristik nama, nim, umur, dan jurusan. Karakteristik ini juga dapat kita sebut sebagai “atribut”. Lalu, seorang mahasiswa dapat belajar, tidur, healing, kemudian bayar UKT (waduh)!

Seperti sebuah blueprint bukan? Hanya definisi dari apa saja yang dibutuhkan atau dimiliki, tanpa kejelasan lebih lanjut.

2.3.3 Objek

Objek? Maksudnya benda terbang aneh yang ada di langit? Haha, itu memang objek sih, *Unidentified Flying Object* (UFO) atau benda terbang tidak dikenal. Nah, sebuah objek atau benda ini merupakan sesuatu yang kita buat berdasarkan kelas yang telah kita definisikan, lho! Ketika kita membuat objek, kita baru akan mengisi atribut yang kita definisikan, dan melakukan aksi untuk objek yang sudah kita buat!

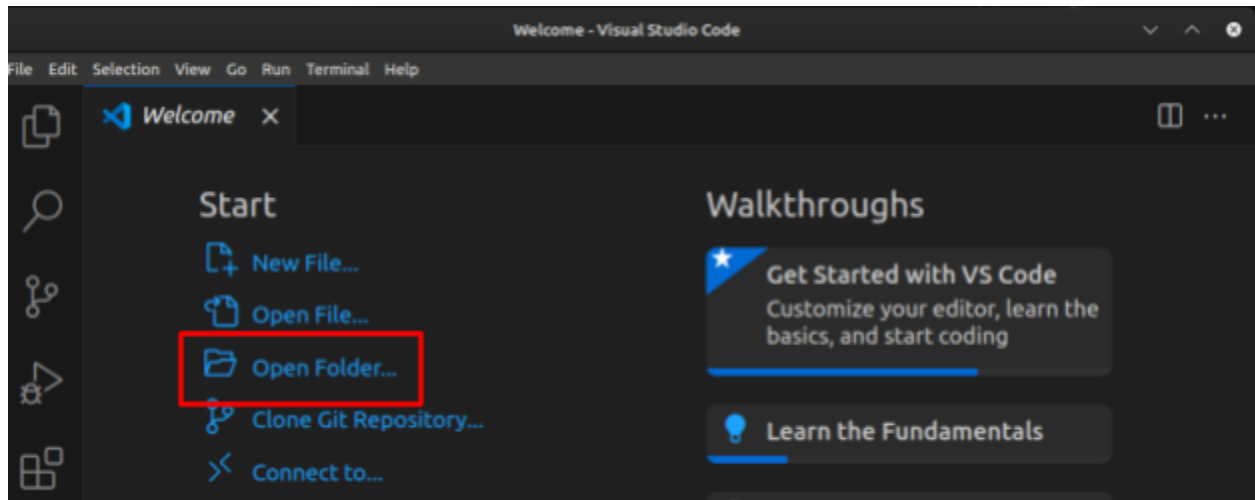


Nah, dengan begini terciptalah sebuah objek mahasiswa baru yang di sini merupakan aku, Dira! Proses pembuatan objek dari sebuah kelas disebut juga *instansiasi*, sehingga sebuah objek juga bisa disebut dengan sebuah *instance*.

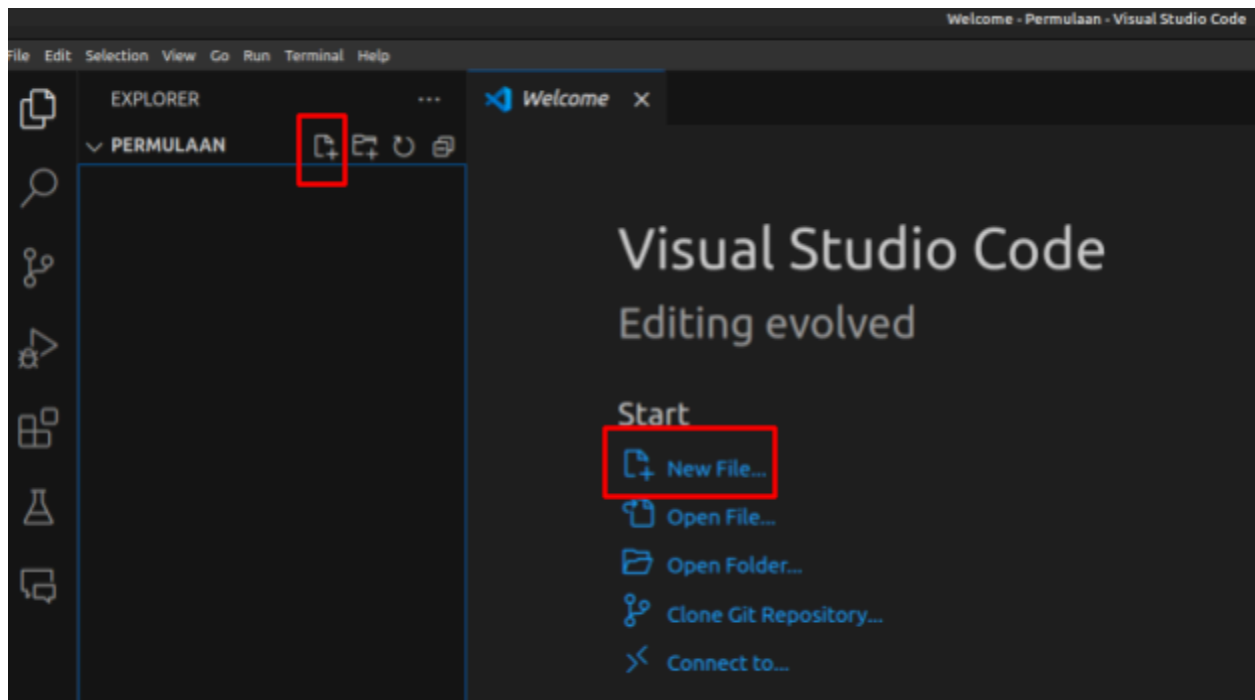
Jadi ingat, ya. Kelas hanyalah sebuah cetak biru untuk objek yang ingin kita buat. Jadi ia cukup kita definisikan sekali, dan nantinya kita bisa membuat banyak objek berbeda-beda dari kelas tersebut. Sama seperti mobil misalnya. Cetak biru sebuah mobil pastinya sama, bukan? Ia memiliki karakteristik seperti merk, jenis, nomor mesin, dan warna. Namun, objek mobil yang diproduksi bisa memiliki merk, jenis, nomor mesin, dan warna yang berbeda-beda.

2.3.4 Struktur program Java, dan “Hello World!”

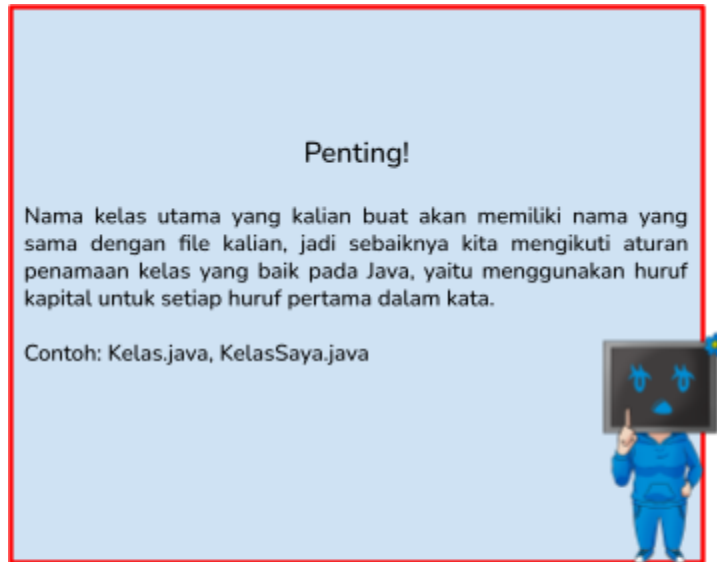
Kita sudah masuk ke bagian paling menyenangkan, nih: menulis kode! Di sini, kita akan mengenal struktur paling dasar sebuah program Java, dan cara menjalankan program sederhana yang akan memunculkan “Hello World!” di konsol kalian. Pertama-tama, kalian bisa membuat sebuah folder proyek kalian. Bebas ingin disimpan dan dinamai apapun. Aku akan membuat sebuah folder bernama “Permulaan”, karena ini adalah permulaan dari perjalanan memprogram kita. Setelah kalian membuat folder proyek kalian, bukalah Visual Studio Code kalian!



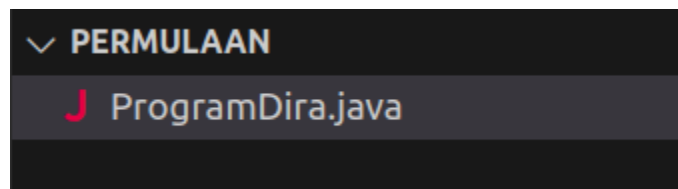
Setelah VS Code terbuka, kalian akan melihat beberapa pilihan. Klik Open Folder, kemudian carilah folder proyek kalian tadi. Bila sudah ketemu, klik folder kalian, kemudian klik “Open” dan nantinya, folder kalian akan terbuka di dalam VS Code.



Setelah terbuka, kalian bisa memilih “New File” atau ikon file baru seperti yang ada pada gambar, kemudian kalian bisa mengetikkan nama file kalian. Nama file tidak boleh ada spasi, kemudian sebaiknya menggunakan huruf kapital untuk setiap kata, ya!



Di sini, kita menamai file dengan nama “ProgramDira.java”. Kalian boleh menamainya sesuka hati kalian. Namun ingat untuk mengikuti aturan penulisan, ya.



Bila kalian sudah selesai, buka file kalian dan kita akan mulai menulis struktur program Java paling dasar: `public static void main string args`.

```
public class ProgramDira {  
    public static void main(String[] args) {  
  
    }  
}
```

Yang kalian lihat adalah sebuah struktur program Java yang paling dasar. Ingat ya, nama kelas utama kalian harus sama dengan nama file kalian! Kode di atas terdiri dari kelas utama kalian, dan juga fungsi yang disebut fungsi utama atau *main function*. Lalu, mengapa kita membutuhkan sebuah kelas utama dan juga fungsi *main* dalam Java? Oke, kalian sudah tahu kalau Java ini nantinya akan di-compile, bukan?

Saat program kalian berubah menjadi *executable* yang bisa dijalankan, program tentunya harus tahu dong dia harus mulai dari mana, dari program yang kalian tulis. Nanti kan program kalian bisa saja ada ratusan baris. Oleh karena itu, kelas utama dan fungsi *main* ini berguna untuk menjadi titik awal program Java kalian di mulai! Sekarang, mari kita membahas lebih detail tentang apa saja yang terlihat di atas.

Kalian sudah mengenal kelas, kan? Pembuatan kelas di Java adalah seperti di atas, di mana kita harus menentukan apakah kelasnya akan bisa dipakai di bagian lain program. Hal ini disebut juga dengan visibilitas kelas. Oleh karena itu, kita harus menggunakan kata kunci, atau keyword untuk menandai kelas kita sebagai kelas yang bisa diakses di seluruh bagian program, atau bagian-bagian tertentu saja. Nah, keyword tersebut di antaranya adalah “public”. Kemudian, ada keyword “class” yang digunakan untuk mendefinisikan kelas kita. Selanjutnya, kita tinggal memberi nama kelas yang kita buat sehingga kita memiliki `public class ProgramDira`.

Selanjutnya, kita bisa melihat ada kurung kurawal ({ }) yang membuka dan menutup. Guna dari kurung kurawal ini adalah untuk mendeklarasikan sebuah “blok” atau disebut juga sebagai scope. Kita bisa berpikir seperti ini: “Semua yang ada di tengah-tengah kurung kurawal buka-tutup kelas “ProgramDira” ini merupakan anggota dari kelas “ProgramDira”. Jadi, blok yang dimaksud adalah “isi” dari “sesuatu”, atau yang berada dalam “sesuatu” tersebut. Misalnya kita ingin membuat atribut untuk sebuah kelas, berarti atribut tersebut akan dimasukkan ke dalam blok kelas tersebut.

Kemudian, ada `public static void main(String[] args)`. Ia adalah sebuah fungsi, atau *method*. Program yang kita tulis akan di jalankan lewat sini! Untuk membuat fungsi, kita tidak memerlukan keyword khusus seperti class. Untuk fungsi, kita bisa melihat static dan void. Arti dari static adalah fungsi kita bersifat statis dan tidak perlu berhubungan dengan kelas utama kita. Tentu saja, kan? Karena fungsi main ini fungsi spesial yang digunakan untuk tempat program dimulai. Lalu void ini artinya fungsi tidak mengembalikan nilai apa-apa, karena memang hanya untuk sekedar dijalankan saja, bukan untuk mendapatkan hasil. Yang terakhir, ada `String[] args`. `String[] args` ini merupakan sebuah parameter, yang fungsinya untuk mendapatkan data yang dimasukkan pengguna saat program dijalankan. Detailnya akan kita bahas lain waktu!

Lebih jelasnya tentang fungsi dan method, akan kita bahas di bab-bab selanjutnya!

Selanjutnya, mari kita cetak “Hello World!” pertama kita dengan menggunakan fungsi `println` milik Java!

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

Dengan menggunakan perintah di atas, kita bisa mencetak "Hello World!" di konsol kita. Jangan lupa untuk menaruh titik koma setiap setelah selesai menuliskan sebuah perintah, ya! Selanjutnya, buka terminal di VSCode kalian dengan memilih opsi Terminal>New Terminal, dan mari kita lakukan kompilasi dengan perintah `javac NamaProgram.java`!

```
>> "Linux Mint" - rahman @ furude-PC in ~/Projects/Java/Permulaan
● -> javac ProgramDira.java
```

Seharusnya kalian sudah bisa melihat ada file baru yang namanya sama dengan nama file Java kalian, namun berakhir dengan ".class". Bila sudah, artinya kompilasi sukses, dan kalian bisa menjalankan program kalian dengan perintah `java NamaProgram`!

```
>> "Linux Mint" - rahman @ furude-PC in ~/Projects/Java/Permulaan
● -> java ProgramDira
Hello World!
```

Dan "Hello World!" pun muncul! Menyenangkan, bukan?

Fungsi dari `println` adalah untuk mencetak sesuatu, kemudian membuat baris baru.

```
public class ProgramDira {
    public static void main(String[] args) {
        System.out.println("Halo!");
        System.out.println("Selamat belajar Java!");
    }
}
```

```
>> "Linux Mint" - rahman
● -> java ProgramDira
Halo!
Selamat belajar Java!
```

Bila kalian tidak ingin membuat baris baru, kalian bisa menggunakan `print` saja, seperti ini.

```
public class ProgramDira {
    public static void main(String[] args) {
        System.out.print("Halo! ");
    }
}
```

Rizza Mandasari, Rahman Hakim

Ilustrasi oleh Nazwa Mursyidan Baldan

```
        System.out.print("Selamat belajar Java!");  
    }  
}
```

```
>> "Linux Mint" - rahman @  
● -> java ProgramDira  
Halo! Selamat belajar Java!
```

Dan seperti itulah!

2.3.5 Menggunakan Git

Selanjutnya, kita akan menggunakan git untuk program sederhana kita ini. Sekarang, bukalah terminal/konsol kalian di VSCode, kemudian ikuti langkah-langkah di bawah ini!

1. Git Init

Perintah yang pertama adalah git init. Perintah ini digunakan untuk melakukan inisialisasi repositori git. Gampangnya, git init itu seperti kita menginstall git di dalam folder proyek kita, agar nanti kita dapat menggunakan git pada proyek kita. Perintah ini akan menambahkan sebuah folder baru bernama “.git” yang isinya adalah file-file dari sistem git, dan juga perubahan-perubahan yang kita lakukan. Untuk menggunakan perintah ini, sangat mudah. Ketikkan saja **git init** di terminal VSCode kalian. Ingat, ya, kita harus berada di dalam folder proyek kita.

```
>> "Linux Mint" - rahman @ furude-PC in ~/Projects/Java/Permulaan  
● -> git init  
hint: Using 'master' as the name for the initial branch. This default branch name  
hint: is subject to change. To configure the initial branch name to use in all  
hint: of your new repositories, which will suppress this warning, call:  
hint:  
hint:   git config --global init.defaultBranch <name>  
hint:  
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and  
hint: 'development'. The just-created branch can be renamed via this command:  
hint:  
hint:   git branch -m <name>  
Initialized empty Git repository in /home/rahman/Projects/Java/Permulaan/.git/
```

Dan kalian akan melihat pesan “**Initialized empty Git repository in <folder proyek kalian>**”. Git pun selesai diinisialisasi!

2. Git Config

Ingat! Bila kalian belum melakukan config sebelumnya karena kalian memakai perangkat yang bukan milik kalian, kalian bisa melakukan config khusus untuk satu proyek saja, yaitu dengan memakai perintah **git config** tanpa flag **--global** seperti sebelumnya!

```
git config user.name "Rahman Hakim"  
git config user.email "rahmanhakim2435@gmail.com"
```

3. Git Add

Perintah yang kedua adalah git add. Perintah ini akan menambahkan file-file yang ingin kalian *keep track* atau kalian lacak perubahannya dengan git. Kalian bisa melakukannya dengan mengetikkan **git add <nama file kalian>**.

```
>> "Linux Mint" - rahman @ furude-PC  
● -> git add ProgramDiraja.java
```

Namun, bila kalian memiliki beberapa file, dan kalian ingin menambahkan semuanya, kalian bisa menggunakan **git add -A**.

```
>> "Linux Mint" - rahman  
● -> git add -A
```

Dan seluruh file dalam proyek kalian sudah ditambahkan dalam git!

4. Git Commit

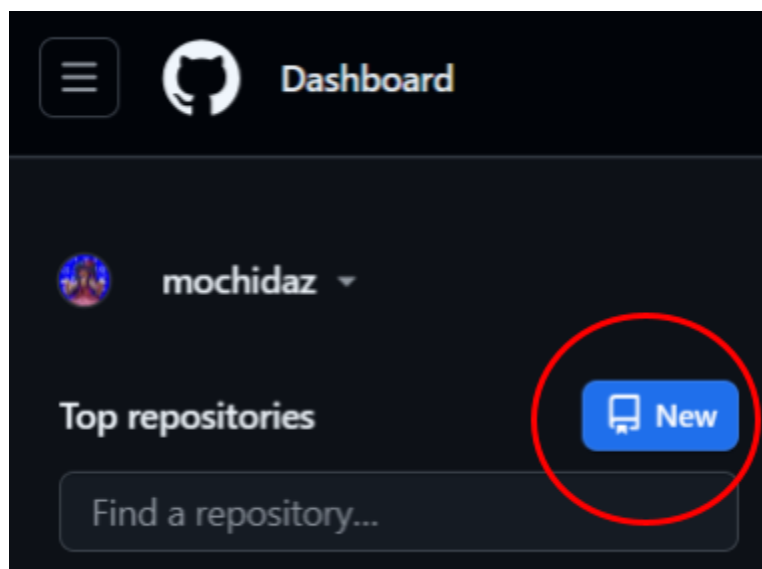
Selanjutnya, kita akan membuat sebuah “checkpoint” atau menyimpan semua perubahan yang sudah kita lakukan selama ini dengan menggunakan perintah **git commit -m “<pesan kalian>”**. Pesan kalian ini bisa kalian ketik dengan bebas. Misalnya, kalian bisa menuliskan apa saja yang kalian sudah ubah atau lakukan.

```
>> "Linux Mint" - rahman @ furude-PC in ~/Projects/Java/Permu
• -> git commit -m "Menambahkan file java pertama"
[master (root-commit) beacde2] Menambahkan file java pertama
1 file changed, 5 insertions(+)
create mode 100644 ProgramDira.java
```

Lalu setelah kalian menekan enter, maka akan muncul teks seperti di atas.

5. Membuat Repositori GitHub

Sekarang, kita akan mulai memakai GitHub! Pertama-tama, kalian harus membuat akun terlebih dahulu di <https://github.com/signup>. Dan bila sudah selesai, maka kalian akan berada di halaman home GitHub. Di sini, kalian akan bisa melihat tombol “new” di sebelah kiri.



Kalian tinggal klik itu saja, dan nanti akan ada menu pembuatan repositori. Lalu, apa sih repositori itu? Nah, repositori ini adalah folder proyek kalian yang diatur oleh git. Jadi, saat kalian menjalankan **git init**, sebenarnya kalian menjadikan folder proyek kalian menjadi sebuah repositori git. Sekarang, kalian akan bertemu dengan tampilan berikut.

Beri saja nama repositori kalian dengan `running-modul-ima`, kemudian set public, lalu langsung saja klik **Create Repository** di kanan bawah.

Dengan begini, kalian sudah membuat *remote repository* atau repositori online kalian. Di tempat ini, kita akan menyimpan repository lokal yang kita buat dengan **git init** tadi. Repositori yang berada di GitHub ini disebut juga dengan *remote repository* karena ia berada di tempat yang bukan komputer kita, ya. Jadi

namanya disebut dengan *remote*. Sedangkan folder proyek kita yang sudah diatur oleh git disebut juga dengan *local repository* atau repositori lokal karena ia berada secara lokal pada komputer kita.

Selanjutnya, kalian bisa menggunakan perintah **git remote add <nama repo remote kalian> <link repositori kalian>** untuk menambahkan repositori remote kalian ke repositori lokal. Jadi kita akan memberitahu git kemana kita akan mengupload proyek kita. Repo remote bisa kita namai apapun, namun biasanya untuk repositori utama, kita menamainya **origin**. Pada terminal, tulis saja perintah remote tadi dengan nama repo remote **origin** dan kopi link repository kalian.

TERMINAL PORTS

```
jects\Permulaan> git remote add origin https://github.com/mochidaz/running-modul-ima
```

Setelah kalian klik enter, repositori sudah berhasil ditambahkan.

6. Git Push

Perintah terakhir adalah **git push** di mana kita akan mengupload proyek kita ke GitHub. Command ini sangat mudah, yaitu dengan **git push <nama repo remote> <nama branch yang dituju>**. Karena kita memakai branch **master** dan repo remote **origin**, maka kita menulisnya seperti berikut:

```
git push origin master
```

Setelahnya, tekan enter, dan bila ini kali pertama kalian memakai git dan GitHub, maka akan ada tampilan untuk login ke GitHub.

GitHub Sign in

Browser/Device

Token

Sign in with your browser





Sign in with a code

Don't have an account? [Sign up](#)

Klik saja **Sign in with your browser** dan kalian akan diarahkan ke browser kalian untuk melakukan sign in! Setelahnya, kembali ke terminal kalian, dan bila lancar, kalian bisa melihat pesan seperti berikut:

```
PS C:\Users\msi GF66\Documents\Projects\Permulaan> git push origin master
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 675 bytes | 675.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/mochidaz/running-modul-ima
```

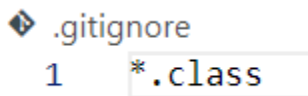
Dan tada! Bila kalian membuka repositori GitHub kalian sekarang, kalian akan melihat proyek kalian sudah dipindahkan ke GitHub!

 mochidaz	Menambahkan file java pertama	6b051e6 · 9 minutes ago	 1 Commit
 ProgramDira.class	Menambahkan file java pertama	9 minutes ago	
 ProgramDira.java	Menambahkan file java pertama	9 minutes ago	

Namun, sekarang kita bisa melihat bahwa file ProgramDira.class juga ikut masuk ke dalam repositori! Harusnya tidak boleh. Kalau kalian memakai **git add -A**, biasanya ini terjadi. Kalian bisa hanya menggunakan **git add <nama file>** saja agar file .class tidak ikut. Tapi ada cara lain, lho!

7. Git Rm dan file .gitignore

Berkenalanlah dengan file **.gitignore**. Dengan file ini, apa saja yang kalian tidak ingin masukkan ke dalam commit bisa kalian tuliskan! Dan git nggak bakalan masukan file-file yang kalian tuliskan ke dalamnya! Pertama-tama, buatlah file bernama **.gitignore** di folder proyek kalian. Kemudian, isi ***.class** di dalam filenya seperti di bawah.



```
.gitignore
1 *.class
```

Tanda asterisk (*) di sini berfungsi sebagai penanda file apa saja yang berakhir dengan .class tidak boleh dimasukkan ke git. Namun, bagaimana bila file tersebut sudah terlanjur masuk duluan seperti sekarang? Nah, kita gunakan perintah **git rm --cached <nama file>**.





```
PS C:\Users\msi GF66\Documents\Projects\Permulaan> git rm --cached ProgramDira.class
rm 'ProgramDira.class'
```

Fungsinya adalah untuk menghapus (remove) file yang sudah terlanjur masuk dari git. Ingat, ya! Penggunaan flag **--cached** ini penting karena kita hanya menghapusnya dari cache saja. Kalau tidak memakai flag itu, bisa-bisa file kalian terhapus. Setelahnya, kalian bisa menggunakan **git add**, **git commit**, dan **git push** kembali.

```
PS C:\Users\msi GF66\Documents\Projects\Permulaan> git add -A
PS C:\Users\msi GF66\Documents\Projects\Permulaan> git commit -m "Menambahkan .class ke gitignore"
[master 0fc8d9d] Menambahkan .class ke gitignore
2 files changed, 1 insertion(+)
create mode 100644 .gitignore
delete mode 100644 ProgramDira.class
```

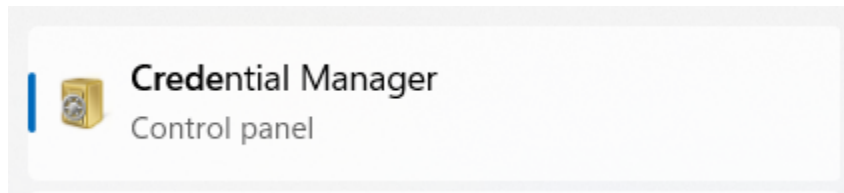
```
PS C:\Users\msi GF66\Documents\Projects\Permulaan> git push origin master
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 311 bytes | 311.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/mochidaz/running-modul-ima
```

Setelahnya, kalian bisa melihat kembali repo kalian. Dan tada! Repo sudah bersih dari file .class!

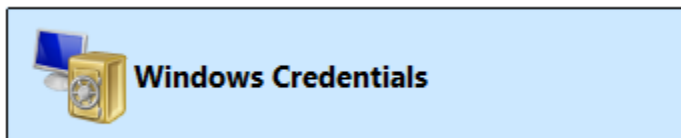
 mochidaz Menambahkan .class ke gitignore	0fc8d9d · 1 minute ago	 2 Commits
 .gitignore	Menambahkan .class ke gitignore	1 minute ago
 ProgramDirajava	Menambahkan file java pertama	20 minutes ago

8. Masalah yang sering terjadi

Seringkali saat kita memakai perangkat milik orang lain, atau misalnya meminjamkan laptop kita ke teman, kita tidak bisa login kembali ke akun GitHub kita. Cara mengatasi masalah ini di Windows adalah dengan membuka **Credential Manager** yang bisa kalian cari di search bar Windows kalian.



Kemudian, saat sudah terbuka, klik **Windows Credentials**



Lalu, kalian akan bisa melihat banyak list *credential* yang kalian punya. Scroll sampai kalian menemukan **git:https://github.com** seperti berikut.

[git:https://github.com](https://github.com)

Internet or network address: [git:https://github.com](https://github.com)

User name: mochidaz

Password: ••••••••

Persistence: Local computer

[Edit](#) [Remove](#)

Kemudian, tinggal klik remove. Dan masalah akan selesai, deh!

Bagaimana, menyenangkan bukan setelah menjalankan program kalian sendiri, dan menguploadnya ke repo GitHub kalian sendiri? Di bab selanjutnya, kita akan belajar lebih mendalam tentang Java. Sampai bertemu di bab selanjutnya!