

PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK

Nama	Raka Arrayan Muttaqien	No. Modul	1
NPM	2306161800	Tipe	Tugas Pendahuluan

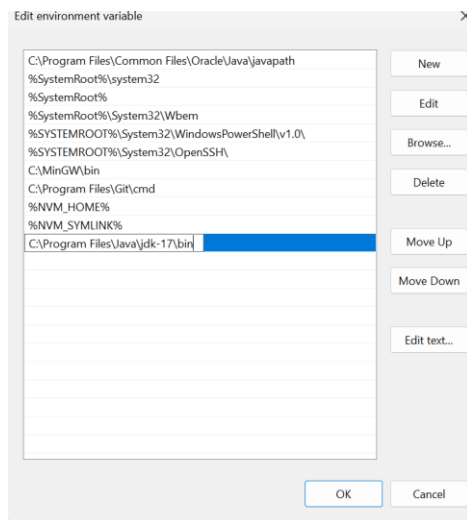
Praktikum Pemrograman Berorientasi Objek Modul 1 – Introduction to Java and Git Tugas Pendahuluan

PART I – INSTALL JDK 17

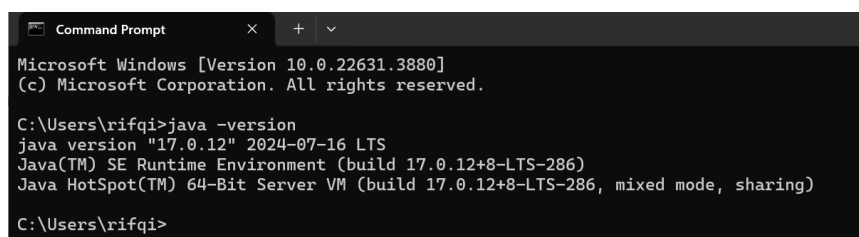
1. Update environment variable komputer Anda untuk menambahkan JDK. Anda bisa mengikuti panduan berikut ini:

https://docs.oracle.com/cd/F74770_01/English/Installing/p6_eppm_install_config/89522.htm

Sertakan screenshot sebagai bukti bahwa Anda sudah menambahkan Java ke “Path”





2. Pada command prompt atau terminal, gunakan command “java -version” untuk membuktikan bahwa Java sudah ter-install dengan benar. Sertakan screenshot-nya!



3. Buatlah program Java sederhana yang akan menampilkan “Hello, World!”, Anda bisa menggunakan aplikasi text editor seperti Notepad untuk membuatnya. Anda bisa mengikuti code di bawah ini:

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

Simpan program anda dalam bentuk .java. Sertakan screenshot-nya!

 RakaArrayan	8/30/2024 8:51 AM	Java Source File	1 KB
 TP_RakaArrayan_2306161800_OOP1	8/29/2024 9:17 PM	Microsoft Word Doc...	2,800 KB

4. Gunakan command “javac NamaDepan.java”, command ini berfungsi untuk melakukan compile pada program Java Anda. Ganti “NamaDepan” dengan nama depan Anda. Sertakan screenshot-nya!

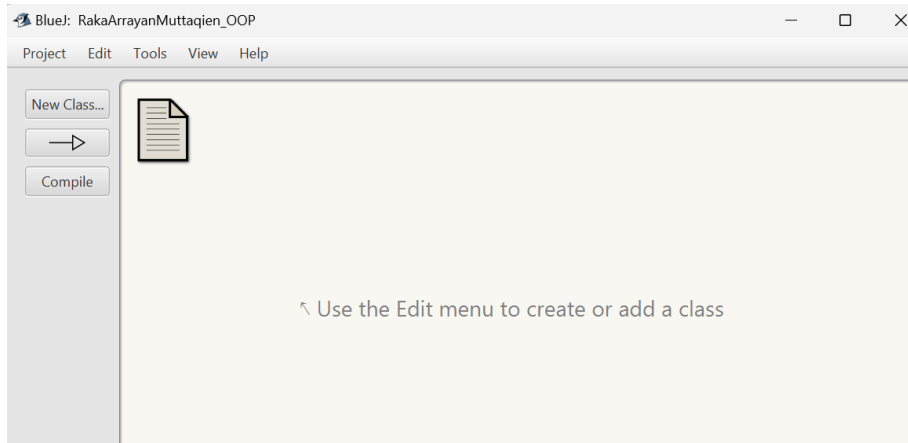
```
Microsoft Windows [Version 10.0.22631.3880]  
(c) Microsoft Corporation. All rights reserved.  
  
C:\Users\rifqi>cd  
C:\Users\rifqi  
  
C:\Users\rifqi>cd "C:\Users\rifqi\Documents\Prak OOP\Modul1"  
C:\Users\rifqi\Documents\Prak OOP\Modul1>javac RakaArrayan.java  
C:\Users\rifqi\Documents\Prak OOP\Modul1>
```

5. Gunakan command “Java NamaDepan” untuk menjalankan program dan menampilkan output. Sertakan screenshot-nya!

```
C:\Users\rifqi\Documents\Prak OOP\Modul1>java RakaArrayan  
Hello, World!  
  
C:\Users\rifqi\Documents\Prak OOP\Modul1>
```

PART II – INSTALL BLUEJ DAN GIT

1. Download dan lakukan instalasi BlueJ dari link berikut: <https://bluej.org/> Setelah selesai melakukan instalasi, buka BlueJ dan screenshot tampilan BlueJ Anda!



2. Download dan lakukan instalasi Git dari link berikut:

<https://git-scm.com/downloads>

Setelah selesai melakukan instalasi, buka Command Prompt atau terminal Anda dan gunakan command “git --version” untuk membuktikan bahwa Git sudah ter-install. Screenshot hasil dari command tersebut!

```
Microsoft Windows [Version 10.0.22631.3880]
(c) Microsoft Corporation. All rights reserved.

C:\Users\rifqi>git --version
git version 2.43.0.windows.1

C:\Users\rifqi>
```

PART III – TEORI

1. Jelaskan apa itu tipe data primitive dan non-primitive pada Java! Sebutkan minimal 3 perbedaan antara keduanya!

- **Primitive**

Struktur data yang dapat menampung satu nilai di lokasi tertentu

Contoh struktur data primitif adalah float, karakter, integer, dan pointer. Nilai pada struktur data primitif disediakan oleh programmer.

- 1) **Integer**: Tipe data integer menyimpan nilai numerik. Ini berisi bilangan bulat yang dapat berupa negatif atau positif. Ketika rentang tipe data integer tidak cukup besar, kita dapat menggunakan tipe data long.
- 2) **Float**: Float adalah tipe data yang dapat menampung nilai desimal. Ketika presisi nilai desimal meningkat, maka tipe data Double digunakan.
- 3) **Boolean**: Ini adalah tipe data yang dapat menampung nilai True (benar) atau False (salah). Tipe data ini terutama digunakan untuk memeriksa kondisi.
- 4) **Character**: Tipe data ini dapat menampung satu nilai karakter, baik huruf besar maupun huruf kecil, seperti 'A' atau 'a'.

- **Non-primitif**

Jenis struktur data yang dapat menampung beberapa nilai baik di lokasi yang berdekatan maupun acak. Tipe data non-primitif ditentukan oleh programmer.

Struktur data non-primitif selanjutnya diklasifikasikan menjadi dua kategori, yaitu struktur data linear dan non-linear. Dalam kasus struktur data linear, data disimpan dalam urutan, yaitu satu data setelah data lainnya. Ketika kita mengakses data dari struktur data linear, kita hanya perlu memulai dari satu tempat dan akan menemukan data lainnya dalam urutan.

- **Perbedaan**

Struktur Data Primitif	Struktur Data non-Primitif
Struktur data primitif adalah jenis struktur data yang menyimpan data dengan satu jenis tipe saja.	Struktur data non-primitif adalah jenis struktur data yang dapat menyimpan data dengan lebih dari satu jenis tipe.
Contoh struktur data primitif adalah integer, karakter, float	Contoh struktur data non-primitif adalah Array, Linked List, Stack.

Struktur data primitif akan selalu mengandung suatu nilai, yaitu tidak bisa bernilai NULL.	Struktur data non-primitif dapat terdiri dari nilai NULL.
Ukurannya bergantung pada tipe dari struktur data tersebut.	Dalam kasus struktur data non-primitif, ukurannya tidak tetap
Dimulai dengan huruf kecil	Dimulai dengan huruf besar
Struktur data primitif dapat digunakan untuk memanggil metode.	Struktur data non-primitif tidak dapat digunakan untuk memanggil metode.

Referensi:

- [1]“Primitive vs non-primitive data structure - javatpoint,” www.javatpoint.com.
<https://www.javatpoint.com/primitive-vs-non-primitive-data-structure>

2. Jelaskan apa itu class dan object pada Java! Jelaskan secara singkat 3 access modifier dan 3 tipe class pada Java.

Jelaskan apa itu class dan object pada Java

Class

- Class bertugas untuk mengumpulkan prosedur/fungsi dan variabel dalam satu tempat. Class merupakan *blueprint* dari sebuah objek atau cetakan untuk membuat objek
- *Class* akan merepresentasikan objek yang mau dibuat. Jadi dalam membuat nama kelas harus disesuaikan dengan objek yang akan dibuat. Penulisan nama *class* memiliki aturan. Yakni dengan format **PascalCase**. Apa itu? Penulisannya diawali dengan huruf kapital. Jika nama variabel tersusun dari dua kata atau lebih maka tidak perlu diberi spasi di antaranya dan diawali dengan huruf kapital pula. Misal: *class MakananKucing*, *class Senjata*, dan *class SignIn*

Object

- *object* adalah sebuah variabel *instance* yang merupakan wujud dari *class*. *Instance* merupakan wujud dari sebuah kelas. Sebuah objek digambarkan dengan *variable* dan *method*.

3 access modifier

- **Public**

1. Modifier **public** akan membuat member dan class bisa di akses dari mana saja.

contoh

package modifier;

```
class Person {
```

```
    public String name;
```

```
    public changeName(String newName){
```

```
        this.name = newName;
```

```
    }
```

```
}
```

Pada class **Person** terdapat dua member, yaitu: atribut **name** dan method **changeName()**

Kedua member tersebut kita berikan modifier **public**. Artinya mereka akan bisa diakses dari mana saja. Namun, class **Person** tidak kita berikan modifier. Maka yang akan terjadi adalah class tersebut tidak akan bisa diimpor (diakses) dari luar package.

2. Class **Person** berada di dalam package **modifier**, bila kita coba akses dari *default package*, maka yang akan terjadi adalah error.
3. Bagaimana solusinya agar bisa diakses dari luar package? Ya kita harus menambahkan modifier **public** ke dalam class **Person**

- **Private**

1. Modifier **private** akan membuat member hanya bisa diakses oleh dari dalam class itu sendiri.
2. Modifier **private** tidak bisa diberikan kepada class, enum, dan interface.
3. Modifier **private** hanya bisa diberikan kepada member class.

```
class Person {
```

```
    private String name;
```

```
    public void setName(String name){
```

```
        this.name = name;
```

```
    }
```

```
public String getName(){  
    return this.name;  
}  
}
```

Pada contoh di atas, kita memberikan modifier **private** pada atribut **name** dan modifier **public** pada method **setName()** dan **getName()**.

4. Apabila kita coba mengakses langsung atribut **name** seperti ini:

```
Person mPerson = new Person()  
mPerson.name = "Petani Kode"; // <- maka akan terjadi error di sini
```

5. Lalu, bagaimana cara mengakses member **private** dari luar class?

Kita bisa memanfaatkan method setter dan getter Karena, method ini akan selalu diberikan modifier **public**.

- **Protected**

1. Modifier **protected** akan membuat member dan class hanya bisa diakses dari:

Class itu sendiri, Sub class atau class anak, Package (class yang berada satu package dengannya).

2. Modifier **protected** juga hanya boleh digunakan pada member saja

package modifier;

```
public class Person {  
    protected String name;
```

```
    public void setName(String name){  
        this.name = name;  
    }
```

```
    public String getName(){  
        return this.name;  
    }  
}
```

Pada contoh di atas, kita memberikan modifier **protected** pada atribut **name**.

Apabila kita coba mengakses dari class yang satu package dengannya, maka tidak akan terjadi error.

Namun, apabila kita mencoba mengakses dari luar package maka akan terjadi error

3 tipe class pada Java.

- **Final class**

Ketika sebuah variabel, fungsi, atau kelas dideklarasikan final, nilainya tetap ada di seluruh program. Mendeklarasikan sebuah metode dengan kata kunci final menunjukkan bahwa metode tersebut tidak dapat digantikan oleh subkelas. Artinya, kelas yang ditandai final tidak dapat menjadi subkelas. Ini sangat berguna ketika membuat kelas yang tidak dapat diubah seperti kelas String

- **Static class**

Statis adalah kata Java yang menjelaskan bagaimana objek disimpan dalam memori. Objek statis merupakan bagian dari kelas tersebut, bukan merupakan contoh dari kelas tersebut. Fungsi utama kelas adalah menyediakan cetak biru untuk kelas yang diwarisi. Kelas statis hanya memiliki anggota statis. Objek tidak dapat dibuat untuk kelas statis.

- **Abstract class**

Kelas yang memiliki nol atau lebih metode abstrak dan ditetapkan dengan kata kunci abstrak disebut kelas abstrak. Kita harus memperluas kelas abstrak secara ketat ke kelas konkret agar dapat menggunakannya karena kelas tersebut belum lengkap. Konstruktor dan metode statis juga dapat disertakan. Kelas tersebut dapat memiliki metode final, yang memaksa subkelas untuk menjaga isi metode tetap utuh.

Referensi :

- [1]D. Intern, “Apa itu OOP pada Java? Beserta Contohnya,” *Dicoding Blog*, Feb. 09, 2021. <https://www.dicoding.com/blog/apa-itu-oop-pada-java-beserta-contohnya/>
- [2]“Belajar Java OOP: Memahami Tingkatan Akses Member dan Class (Modifier),” *Petani Kode*, Dec. 28, 2017.<https://www.petanikode.com/java-oop-modifier/>
- [3]“Types of Classes in Java,” *GeeksforGeeks*, Mar. 05, 2022. <https://www.geeksforgeeks.org/types-of-classes-in-java/>

3. Jelaskan perbedaan antara constructor dan method pada Java! Untuk apa umumnya constructor digunakan?

- **Constructors**

Digunakan untuk menginisialisasi status objek. Seperti metode, konstruktor juga berisi kumpulan pernyataan (yaitu instruksi) yang dieksekusi pada saat pembuatan Objek. Setiap kali objek dibuat menggunakan kata kunci new() setidaknya satu konstruktor (bisa jadi konstruktor default) dipanggil untuk menetapkan nilai awal ke anggota data dari kelas yang sama

- **Method**

Kumpulan pernyataan yang melakukan tugas tertentu dan mengembalikan hasilnya kepada pemanggil. Metode dapat melakukan tugas tertentu tanpa mengembalikan apa pun. Metode memungkinkan kita untuk menggunakan kembali kode tanpa mengetik ulang kode tersebut.

- **Perbedaan**

Constructors	Method
blok kode yang menginisialisasi objek yang baru dibuat.	kumpulan pernyataan yang mengembalikan nilai saat dieksekusi.
Konstruktor dapat digunakan untuk menginisialisasi suatu objek	Suatu Metode terdiri dari kode Java yang akan dieksekusi.
Konstruktor dipanggil ketika suatu objek dibuat menggunakan kata kunci new .	Suatu Metode dipanggil melalui pemanggilan
tidak memiliki tipe pengembalian.	harus memiliki tipe pengembalian.
harus sama dengan nama kelas.	Nama Method dapat berupa apa saja.
Suatu kelas dapat memiliki banyak Konstruktor tetapi tidak boleh memiliki parameter yang sama.	Suatu kelas dapat memiliki banyak metode tetapi tidak boleh memiliki parameter yang sama.

Biasanya constructor digunakan untuk melakukan Overloading Constructor yaitu Memberikan berbagai cara untuk membuat objek dengan konfigurasi yang berbeda, seperti membuat objek Book dengan atau tanpa parameter judul dan penulis.

Referensi

- [1]“Difference between the Constructors and Methods,” *GeeksforGeeks*, Apr. 15, 2019.
<https://www.geeksforgeeks.org/difference-between-the-constructors-and-methods/>

4. Jelaskan apa itu interface! Jelaskan secara singkat minimal 3 perbedaan antara interface dengan abstract class!

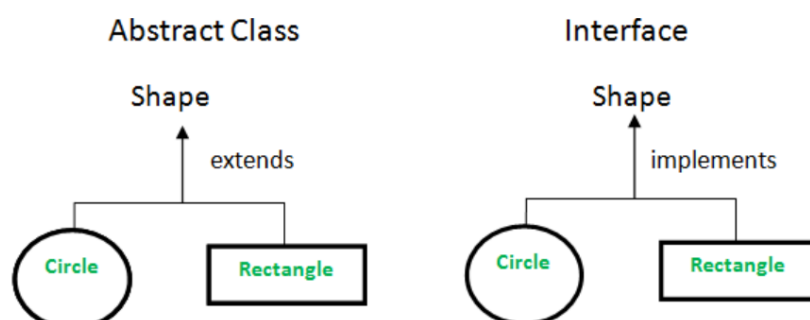
- Pengertian interface

Dalam Java, sebuah interface menentukan perilaku dari suatu class dengan menyediakan tipe abstrak. Sebagai salah satu konsep inti dari Java, abstraksi, polimorfisme, dan pewarisan ganda didukung melalui teknologi ini. Interface digunakan dalam Java untuk mencapai abstraksi. Dengan menggunakan kata kunci implements, sebuah class Java dapat mengimplementasikan sebuah interface.

Interface Java mendefinisikan tanda tangan metode tanpa implementasi, memberikan template bagi class untuk diikuti. Mereka mendorong fleksibilitas kode, yang memfasilitasi skalabilitas dan pemeliharaan yang lebih mudah. Pengembang Java harus memahami interface untuk mencapai abstraksi dan polimorfisme.

- Perbedaan

Abstract class	Interface
berisi metode abstrak (tanpa implementasi) dan konkret (dengan implementasi)	Menentukan sekumpulan metode yang harus diimplementasikan oleh suatu kelas; metode bersifat abstrak secara default.
Class hanya dapat mewarisi dari satu class abstract	Suatu class dapat mengimplementasikan beberapa interface.
Metode dan properti dapat memiliki pengubah akses apa pun (public, protected, privat).	Metode dan properti secara implisit bersifat public.
Dapat memiliki variabel anggota (final, non-final, statis, non-statis).	Variabel secara implisit bersifat publik, statis, dan final (konstanta).



Referensi:

- [1]“Difference between Abstract Class and Interface in Java GeeksforGeeks,” *GeeksforGeeks*, Oct. 28, 2015.
<https://www.geeksforgeeks.org/difference-between-abstract-class-and-interface-in-java/>

5. Jelaskan maksud dari enumeration! Dan jelaskan pada kasus seperti apa enumeration cocok untuk digunakan!

- Pengertian

Dalam Java, Enumerations atau Java Enum digunakan untuk mewakili sekelompok konstanta bernama dalam bahasa pemrograman. Java Enums digunakan ketika kita mengetahui semua nilai yang mungkin pada saat kompilasi, seperti pilihan pada menu, mode pembulatan, bendera baris perintah, dan sebagainya.

Sebuah enumerasi (enum) dalam Java adalah tipe kelas. Meskipun kita tidak perlu menginstansiasi sebuah enum menggunakan new, enum memiliki kemampuan yang sama seperti kelas lainnya. Fakta ini membuat enumerasi Java menjadi alat yang sangat kuat. Sama seperti kelas, Anda dapat memberikan konstruktor pada enum, menambahkan variabel instance dan metode, bahkan mengimplementasikan interface

- Kasus

Empat jenis dalam satu set kartu remi dapat berupa 4 enumerator yang dinamai Club, Diamond, Heart, dan Spade, yang termasuk dalam tipe enumerasi bernama Suit. Contoh lainnya termasuk tipe enumerasi alami (seperti planet, hari dalam seminggu, warna, arah, dll.).

Satu hal yang perlu diingat adalah bahwa, tidak seperti kelas, enumerasi tidak mewarisi kelas lain dan tidak dapat diperluas (yaitu menjadi superclass). Kita juga dapat menambahkan variabel, metode, dan konstruktor ke dalamnya. Tujuan utama dari sebuah enum adalah untuk mendefinisikan tipe data kita sendiri (Tipe Data Enumerasi).

contoh:

1. Declaration outside the class

```
// A simple enum example where enum is declared  
// outside any class (Note enum keyword instead of
```

```
// class keyword)
enum Color {
    RED,
    GREEN,
    BLUE;
}

public class Test {
    // Driver method
    public static void main(String[] args) {
        Color c1 = Color.RED;
        System.out.println(c1);
    }
}
```

2. Declaration inside a class

```
// enum declaration inside a class.
public class Test {
    enum Color {
        RED,
        GREEN,
        BLUE;
    }

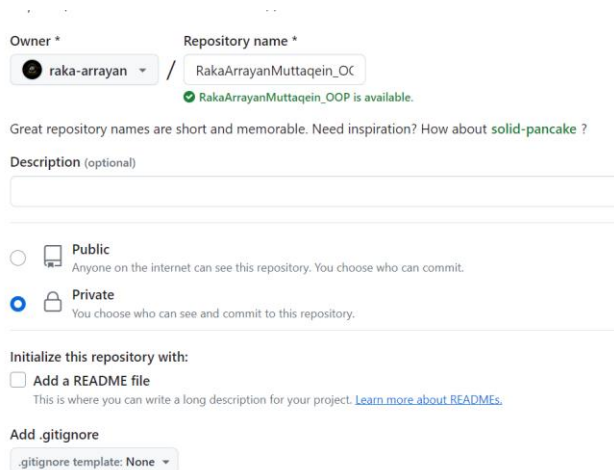
    // Driver method
    public static void main(String[] args) {
        Color c1 = Color.RED;
        System.out.println(c1);
    }
}
```

Referensi :

- [1]“enum in Java,” *GeeksforGeeks*, Sep. 26, 2016. <https://www.geeksforgeeks.org/enum-in-java/>

PART IV – MEMBUAT REPOSITORY GITHUB

1. Screenshoot Tidak perlu membuat README ,Tidak perlu membuat .gitignore ,Repository private



Owner * raka-arrayan / Repository name * RakaArrayanMuttaqein_OC
✓ RakaArrayanMuttaqein_OOP is available.

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

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

Invite collaborator untuk username berikut: kamalMakarim, den-dimas, georgewtg, emirfateen, SinurayaYasmin, ryansatj, nargafz, patuyyy

<input type="checkbox"/>	 Dimas Awaiting den-dimas's response	Pending Invite 	
<input type="checkbox"/>	 Emir Fateen Haqqi Awaiting emirfateen's response	Pending Invite 	
<input type="checkbox"/>	 George Wiliam Thomas Gonata Awaiting georgewtg's response	Pending Invite 	
<input type="checkbox"/>	 Kamal Makarim Iskandar Awaiting kamalMakarim's response	Pending Invite 	
<input type="checkbox"/>	 Fairuz Muhammad Awaiting NargaFRZ's response	Pending Invite 	
<input type="checkbox"/>	 Raihan Muhammad Ihsan Awaiting patuyyy's response	Pending Invite 	
<input type="checkbox"/>	 Ryan Tjendana Awaiting ryansatj's response	Pending Invite 	
<input type="checkbox"/>	 Yasmin Devina Sinuraya Awaiting SinurayaYasmin's response	Pending Invite 	

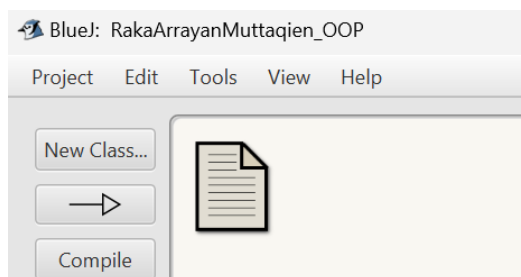
2. Setelah membuat repository, buatlah sebuah folder khusus di Laptop Anda. Folder ini akan terus digunakan sebagai tempat Anda menyimpan program Anda selama praktikum OOP. Penamaan dan penempatan direktori folder dibebaskan.

- Buka BlueJ, dan buat project baru dengan nama “[Nama_Lengkap]_OOP”. Simpan project di dalam folder yang sudah Anda buat di nomor 2. Berikut contoh nya hasil nya :

https://github.com/NetworkLaboratory/Contoh_OOP

Name	Date modified	Type	Size
Modul1	8/29/2024 8:54 PM	File folder	
Modul2	8/29/2024 8:53 PM	File folder	
Modul3	8/29/2024 8:53 PM	File folder	
Modul4	8/29/2024 8:53 PM	File folder	
Modul5	8/29/2024 8:53 PM	File folder	
Modul6	8/29/2024 8:53 PM	File folder	

Buat project baru dengan nama “[Nama_Lengkap]_OOP”.



Simpan project di dalam folder yang sudah Anda buat di nomor 2

Name	Date modified	Type	Size
RakaArrayanMuttaqien_OOP	8/30/2024 9:04 AM	File folder	
RakaArrayan.class	8/30/2024 9:02 AM	CLASS File	1 KB
RakaArrayan	8/30/2024 9:01 AM	Java Source File	1 KB
TP_RakaArrayan_2306161800_OOP1	8/30/2024 9:36 AM	Microsoft Word Doc...	3,444 KB

- Buka Command Prompt atau Git Bash pada folder Anda. Pada tahap ini, Anda akan menghubungkan lokal repository Anda dengan remote repository Github. Ikuti langkah-langkah berikut:

git init

git remote add origin https://github.com/Username_Github>Nama_Repository.git

```
Microsoft Windows [Version 10.0.22631.3880]
(c) Microsoft Corporation. All rights reserved.

C:\Users\rifqi>cd
C:\Users\rifqi

C:\Users\rifqi>cd "C:\Users\rifqi\Documents\Prak OOP\Modul1"
C:\Users\rifqi\Documents\Prak OOP\Modul1>git init
Initialized empty Git repository in C:/Users/rifqi/Documents/Prak OOP/Modul1/.git/

C:\Users\rifqi\Documents\Prak OOP\Modul1>git remote add origin https://github.com/raka-arrayan/RakaArrayanMuttaqien_OOP.git
```

5. Selanjutnya Anda akan menambahkan seluruh file pada repository lokal Anda ke repository Github. Gunakan command “git add .” atau “git add --all” untuk menambahkan seluruh file tersebut.

```
Microsoft Windows [Version 10.0.22631.3880]
(c) Microsoft Corporation. All rights reserved.

C:\Users\rifqi>cd
C:\Users\rifqi>

C:\Users\rifqi>cd "C:\Users\rifqi\Documents\Prak OOP\Modul1"

C:\Users\rifqi\Documents\Prak OOP\Modul1>git init
Initialized empty Git repository in C:/Users/rifqi/Documents/Prak OOP/Modul1/.git/

C:\Users\rifqi\Documents\Prak OOP\Modul1>git remote add origin https://github.com/raka-arrayan/RakaArrayanMuttagein_OOP.git

C:\Users\rifqi\Documents\Prak OOP\Modul1>git add .

C:\Users\rifqi\Documents\Prak OOP\Modul1>
```

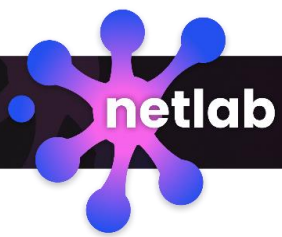
6. Gunakan command “git commit -m “[message]” untuk memberikan pesan saat Anda melakukan push nanti. Isi bagian [message] sesuai dengan hal yang Anda kerjakan, misalkan pada bagian ini Anda bisa menggunakan: git commit -m “Membuat repository”

```
C:\Users\rifqi\Documents\Prak OOP\Modul1>git commit -m"membuat repository"
[master e43735e] membuat repository
3 files changed, 0 insertions(+), 0 deletions(-)
delete mode 100644 TP_RakaArrayan_2306161800_OOP1.docx
create mode 100644 TP_RakaArrayan_2306161800_OOP1.pdf
delete mode 100644 ~WRL0933.tmp
```

7. Selanjutnya Anda akan melakukan push ke remote repository. Ikuti command berikut: git push origin [branch] Isi bagian [branch] sesuai dengan branch tempat Anda sekarang.

```
C:\Users\rifqi\Documents\Prak OOP\Modul1>git push -u origin master
Enumerating objects: 12, done.
Counting objects: 100% (12/12), done.
Delta compression using up to 16 threads
Compressing objects: 100% (12/12), done.
Writing objects: 100% (12/12), 7.68 MiB | 2.07 MiB/s, done.
Total 12 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/raka-arrayan/RakaArrayanMuttagein_OOP.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.
```

8. Jika berhasil, di repository Github Anda sekarang seharusnya ada file README.TXT dan package.bluej.



Name	Last commit message	Last commit date
..		
README.TXT	Membuat repository	9 hours ago
package.bluej	Membuat repository	9 hours ago