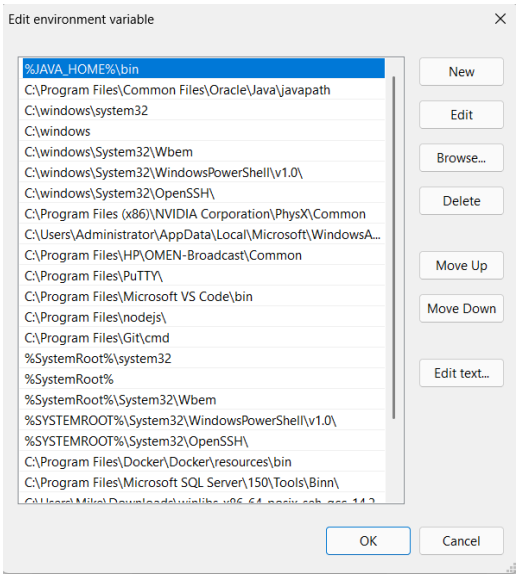
PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK

Nama	**Michael Christian**	**No. Modul**	**1**
NPM	**2406348944**	**Tipe**	**Tugas Pendahuluan**

A. PART I – INSTALL JDK 17

Screenshot JDK



Screenshot java -version

```
C:\Users\Mike>java -version
java version "17.0.16" 2025-07-15 LTS
Java(TM) SE Runtime Environment (build 17.0.16+12-LTS-247)
Java HotSpot(TM) 64-Bit Server VM (build 17.0.16+12-LTS-247, mixed mode, sharing)
```

Screenshot Main.java

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

```
PS C:\Users\Mike\IdeaProjects\TP\src> javac Main.java
PS C:\Users\Mike\IdeaProjects\TP\src> java Main.java
Hello World!
Saya Michael!
```

Screenshot javac Main.java

```
C:\Users\Mike\IdeaProjects\TP\src>javac Main.java
```

Screenshot java Main.java

```
C:\Users\Mike\IdeaProjects\TP\src>java Main.java
Hello World!
Saya Michael!
```

B. PART II – INSTALL GIT

Screenshot git version

```
C:\Users\Mike\IdeaProjects\TP\src>git --version
git version 2.47.0.windows.2
```

C. PART III – TEORI

1. Interface dalam Java adalah sebuah blueprint dalam program yang berisi kumpulan method abstract (hanya deklarasi tanpa implementasi) serta static constants. Setiap class yang mengimplementasikan interface wajib memberikan implementasi nyata untuk semua method yang dideklarasikan. Interface berperan untuk menyembunyikan detail implementasi dan hanya menekankan pada apa yang harus dilakukan oleh class, bukan *bagaimana* cara melakukannya. Perbedaan utamanya yaitu:

   1) Implementation Method

   Abstract class dapat berisi implementation method (fungsi yang sudah ada isinya) sekaligus abstract method (fungsi tanpa isi yang harus diimplementasikan di subclass). Sedangkan, interface pada dasarnya hanya berisi abstract method, tapi semenjak Java 8 bisa memiliki default dan static implementation method.

   2) Inheritance

   Abstract class hanya bisa diteruskan dengan extends dan terbatas hanya pada satu class saja.

Beda halnya, interface diimplementasikan dengan implements, dan sebuah class dapat mengimplementasikan lebih dari satu interface.

3) **Variables**

Abstract class dapat memiliki berbagai macam variabel (instance, static, final, maupun non-final) serta mendukung berbagai access modifier (private, protected, public). Namun, interface hanya dapat memiliki variabel yang otomatis public static final (constant).

Referensi:

- "Tutorial Java OOP: Memahami Interface di Java (dan Contohnya)," *Petani Kode*, Dec. 28, 2019. https://www.petanikode.com/java-oop-interface/ (accessed Aug. 29, 2025).
- GeeksforGeeks, "Java Interface," *GeeksforGeeks*, May 18, 2016. https://www.geeksforgeeks.org/java/interfaces-in-java/ (accessed Aug. 29, 2025).
- GeeksforGeeks, "Difference Between Abstract Class and Interface in Java," *GeeksforGeeks*, Oct. 28,2015.https://www.geeksforgeeks.org/java/difference-between-abstract-class-and-interface-in-java/ (accessed Aug. 29, 2025).
- "Difference between Abstract Class and Interface in Java," *BYJUS*. https://byjus.com/gate/difference-between-abstract-class-and-interface-in-java/ (accessed Aug. 29, 2025).

2. Tipe data primitive adalah basic data type yang sudah disediakan langsung oleh bahasa Java. Tipe ini menyimpan nilai secara langsung di memori, ukurannya tetap, dan tidak memiliki method bawaan. Contohnya adalah int, byte, long, short, float, double, char dan boolean. Sedangkan, tipe data non-primitive adalah basic data type yang berbentuk objek, sehingga yang disimpan di memori adalah referensi atau alamat objek, bukan nilainya langsung. Tipe ini dapat dibuat dari class atau interface, dan biasanya memiliki method bawaan. Contohnya adalah String, Array, Class, Object serta class buatan sendiri. Berikut ini merupakan 3 perbedaan dari primitive dan non-primitive data type, diantaranya yaitu:
    1) Primitive digunakan untuk menyimpan nilai sederhana seperti integer, boolean, dan character. Sedangkan, non-primitive merupakan tipe data kompleks yang terdiri dari satu atau lebih primitive.
    2) Primitve digunakan untuk menyimpan nilai sederhana seperti integer, boolean, dan character. Berbeda dengan non-primitive yang digunakan untuk menyimpan objek data yang lebih

kompleks seperti array, stack, queue, dan tree.

3) Ukurannya tetap dan memiliki rentang nilai tertentu, berbeda dengan non-primitve yang dapat diubah ukurannya atau dimodifikasi saat runtime.

Referensi:

- "Difference Between Primitive and Non-Primitive Data Structure," *Ccbp.in*, 2025. https://www.ccbp.in/blog/articles/difference-between-primitive-and-non-primitive-data-structure (accessed Aug. 30, 2025).
- w3schools, "Java Data Types," *www.w3schools.com*. https://www.w3schools.com/java/java_data_types.asp (accessed Aug. 30, 2025).
- E. Team, "Data Types in Java – Primitive and Non-Primitive Data Types," *Great Learning Blog: Free Resources what Matters to shape your Career!*, Jan. 06, 2025. https://www.mygreatlearning.com/blog/data-types-in-java/ (accessed Aug. 31, 2025).

3. Class adalah suatu template atau blueprint untuk membuat objects dengan tipe yang sama. Sedangkan, object adalah instance atau perwujudan nyata dari sebuah class dan juga setiap object bisa memiliki nilai atribut berbeda meskipun berasal dari class yang sama. Berikut merupakan 3 access modifiers beserta penjelasannya:

   1) Public Access Modifier

   Access modifier ini memiliki scope yang luas dibanding access modifier lainnya dan juga anggota class (classes, methods, data) yang dideklarasikan secara publik dapat diakses dari mana saja (maupun di luar package). Modifier ini digunakan ketika anggota class memang harus tersedia secara bebas untuk digunakan oleh class lain.

   2) Private Access Modifier

   Private berarti membatasi akses variabel, method, atau constructor hanya dari dalam class tempat ia dideklarasikan. Anggota yang bersifat privat tidak bisa diakses dari class lain, meskipun berada dalam package yang sama, maupun dari subclass. Modifier ini digunakan untuk keamanan data (encapsulation) dan dapat diakses melalui setter dan getter.

   3) Protected Access Modifier

   Protected access modifier memungkinkan variabel atau method diakses oleh class dalam package yang sama, serta oleh subclass meskipun berada di package berbeda. Modifier ini digunakan untuk memberi akses terbatas pada pewarisan, tapi tetap membatasi akses dari class luar yang tidak berhubungan.

Selain modifier, ada juga type class pada Java dan berikut merupakan 3 contohnya:

1) **Final Class**

   Final class adalah class yang dideklarasikan dengan keyword final sehingga tidak bisa diwariskan oleh class lain. Modifier final juga bisa digunakan pada method agar tidak bisa di-override, serta pada variabel agar nilainya tetap. Final class sering digunakan untuk membuat class immutable seperti String.

2) **POJO Class**

   POJO (Plain Old Java Object) adalah class sederhana yang biasanya hanya berisi variabel private dengan getter dan setter untuk mengaksesnya. POJO berfungsi sebagai wadah data tanpa logika kompleks, meskipun dapat meng-override method dari Object seperti equals() atau mengimplementasikan interface seperti Serializable.

3) **Final Class**

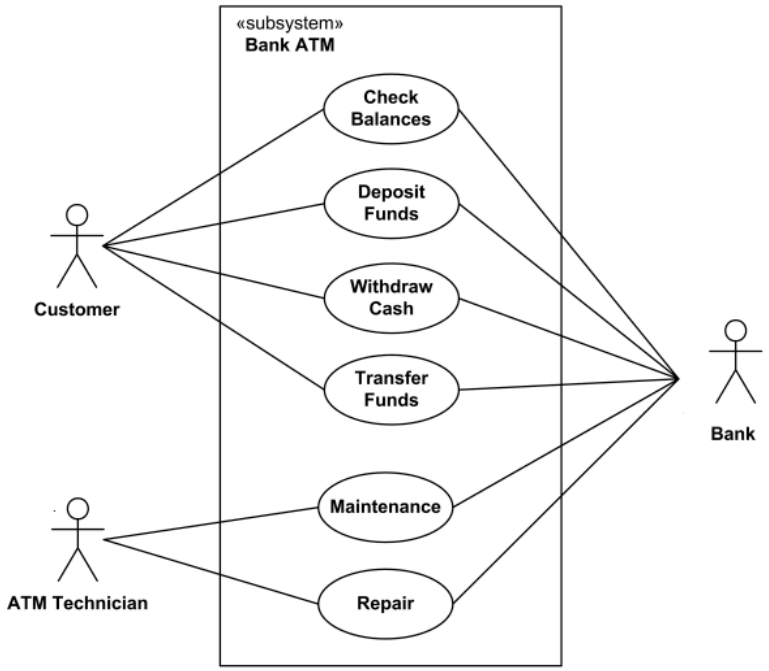
   Final class adalah class yang dideklarasikan dengan keyword final sehingga tidak bisa diwariskan oleh class lain. Modifier final juga bisa digunakan pada method agar tidak bisa di-override, serta pada variabel agar nilainya tetap. Final class sering digunakan untuk membuat class immutable seperti String.

Referensi:

- GeeksforGeeks, "Classes and Objects in Java," *GeeksforGeeks*, Feb. 07, 2017. https://www.geeksforgeeks.org/java/classes-objects-java/ (accessed Aug. 31, 2025).
- F. Darari and Tim DDP 2 Fasilkom UI, *Dasar-Dasar Pemrograman 2 (Java)*, vol. 6. Gedung A lantai dasar, Kompleks eks Perpustakaan Pusat Kampus UI Depok, Jawa Barat, 16424: Open Course Ware UI, 2020, pp. 2–22. Accessed: Aug. 31, 2025. [Online]. Available: https://ocw.ui.ac.id/pluginfile.php?file=%2F1449%2Fmod_resource%2Fcontent%2F0%2F06-fop2-oop_rev5.pdf
- GeeksforGeeks, "Access Modifiers in Java," *GeeksforGeeks*, Feb. 13, 2017. https://www.geeksforgeeks.org/java/access-modifiers-java/ (accessed Aug. 31, 2025).
- GeeksforGeeks, "Types of Classes in Java," *GeeksforGeeks*, Mar. 05, 2022. https://www.geeksforgeeks.org/java/types-of-classes-in-java/ (accessed Aug. 31, 2025).

4. UML (Unified Modeling Language) adalah bahasa pemodelan visual standar yang digunakan untuk merancang, memvisualisasikan, dan mendokumentasikan sistem berorientasi objek. UML berperan juga sebagai blueprint dalam pengembangan perangkat lunak, sehingga memudahkan programmer

maupun pemula dalam memahami struktur, alur, dan interaksi sistem. Dengan UML, proses pengembangan menjadi lebih efektif karena mampu memenuhi kebutuhan pengguna secara tepat, mempertimbangkan aspek seperti skalabilitas, keamanan, dan keandalan, sekaligus mempermudah komunikasi antar tim serta pengguna aplikasi. Berikut adalah contoh dari diagram UML beserta penjelasan dari class, atribut, dan methodnya:



1) UML Class: Customer

No	Bagian	Nama	Tipe Data	Keterangan
1.	Atribut	customerID	int	Identitas unik nasabah
		name	String	Nama nasabah
		balance	double	Saldo nasabah
2.	Method	checkBalance()	void	Menampilkan saldo nasabah
		deposit(amount)	void	Menampilkan saldo sesuai jumlah setoran
		withdraw(amount)	void	Mengurangi saldo sesuai jumlah tarik tunai
		transfer(toAcc, amount)	void	Mengirim dana ke tabungan lain

2) UML Class: ATM Technician

No	Bagian	Nama	Tipe Data	Keterangan
1.	Atribut	technicianID	int	Identitas unik teknisi
		name	String	Nama teknisi
2.	Method	doMaintenance()	void	Melakukan perawatan mesin ATM
		repairATM()	void	Memperbaiki mesin ATM rusak

3) UML Class: Bank

Pada diagram, aktor Bank bisa melakukan semua use case tersebut, namun disini saya akan menambahkan nama use case baru yang lebih singkat:

No	Bagian	Nama	Tipe Data	Keterangan
1.	Atribut	bankName	String	Nama bank penyedia layanan
		bankCode	String	Kode identitas bank
2.	Method	authTransaction()	boolean	Memvalidasi transaksi nasabah
		updateBalance()	void	Memperbaiki saldo nasabah setelah transaksi

Referensi:

- D. Intern, "Apa itu UML? Beserta Pengertian dan Contohnya," *Dicoding Blog*, May 11, 2021. https://www.dicoding.com/blog/apa-itu-uml/ (accessed Aug. 31, 2025).
- A. R. Faulina, "Apa itu UML? Ini Pengertian, Fungsi, dan Contohnya," *Sekawan Media*, Mar. 23, 2023. https://www.sekawanmedia.co.id/blog/apa-itu-uml/ (accessed Aug. 31, 2025).

5. Constructor adalah blok khusus yang digunakan untuk menginisialisasi object saat dibuat. Constructor selalu dipanggil secara otomatis ketika object dibuat dengan keyword new, dan namanya harus sama dengan nama class tanpa tipe kembalian. Sebaliknya, method adalah kumpulan pernyataan yang digunakan untuk menjalankan suatu tugas atau perilaku dari object. Method harus dipanggil secara eksplisit, bisa memiliki tipe kembalian (atau void jika tidak mengembalikan nilai),

serta digunakan untuk menjalankan logika program secara berulang tanpa menulis ulang kode. Maka dari itu, constructor berfungsi saat awal pembuatan object, sedangkan method berfungsi untuk aksi/perilaku setelah object sudah ada.

No	Aspek	Constructor	Method
1.	Nama	Harus sama dengan nama class	Bisa nama apapun (tapi tidak boleh sama dengan kata kunci Java).
2.	Pemanggilan	Dipanggil secara implisit saat objek dibuat dengan new.	Dipanggil secara eksplisit oleh programmer setelah objek dibuat.
3.	Tujuan	Untuk memberi nilai awal pada atribut saat object dibuat.	Untuk menjalankan operasi atau perilaku pada objek yang ada
4.	Return Type	Tidak memiliki return type (bahkan void tidak boleh).	Memiliki return type (misalnya int, String, atau void).
5.	Overloading / Inheritance	Bisa overload (punya banyak constructor dengan parameter berbeda), tapi tidak bisa diwariskan ke subclass.	Bisa overload maupun override, dan bisa diwariskan ke subclass.
6.	Waktu Eksekusi	Hanya dieksekusi sekali saat objek pertama kali dibuat.	Bisa dipanggil berkali-kali selama objek masih ada.

Singkatnya, constructor digunakan saat object baru dibuat untuk langsung punya data/atribut awal. Sedangkan method digunakan saat ingin object melakukan suatu aksi berkali-kali.

Referensi:

- GeeksforGeeks, “Difference between the Constructors and Methods,” *GeeksforGeeks*, Apr. 15, 2019. https://www.geeksforgeeks.org/java/difference-between-the-constructors-and-methods/ (accessed Aug. 31, 2025).
- Sushant Gaurav, “Difference between Constructor and Method in Java- Scaler Topics,” *Scaler Topics*, Sep. 25, 2023. https://www-scaler-com.translate.goog/topics/difference-between-constructor-and-method/?_x_tr_sl=en&_x_tr_tl=id&_x_tr_hl=id&_x_tr_pto=tc&_x_tr_hist=true (accessed Aug. 31, 2025).