



Pemrograman Java

Riky Ahmad Fathoni

Tahun 2022



Riky Ahmad Fathoni

- Telegram : @rikyahmad
- LinkedIn : linkedin.com/in/riky
- Facebook : fb.com/rikyahmadf
- Email : riky.fathoni@gmail.com



Agenda

- Pengenalan Java (Hal. 4)
- Instalasi Java (Hal. 12)
- Tipe Data (Hal. 16)
- Pengenalan OOP (Hal. 34)
- Method (Hal. 39)
- Variable (Hal. 49)
- Inheritance (Hal. 53)
- Polymorphism (Hal. 57)
- Encapsulation (Hal. 63)
- Abstract Class (Hal. 68)
- Abstract Method (Hal. 72)
- Enum Class (Hal. 75)
- Interface (Hal. 82)
- Access Modifier (Hal. 88)
- Annotation (Hal. 94)
- Reflection (Hal. 100)
- Pengenalan Software Testing (Hal. 104)
- Unit Test (Hal. 107)
- Quiz

Pengenalan Java



Sejarah Java

- Java adalah bahasa pemrograman yang dibuat oleh James Gosling saat bekerja di Sun Microsystems
- Java dirilis ke public tahun 1995
- Java adalah bahasa pemrograman berorientasi objek dan mendukung pengelolaan memori secara otomatis
- Saat ini perusahaan Sun Microsystems telah dibeli oleh Oracle
- Java terkenal dengan write once, run anywhere, karena binary program Java di-generate secara independen dan bisa dijalankan di Java Virtual Machine yang terinstall di berbagai sistem operasi



Teknologi Java

- Java Standard Edition
- Java Enterprise Edition
- Java Micro Edition
- Java FX

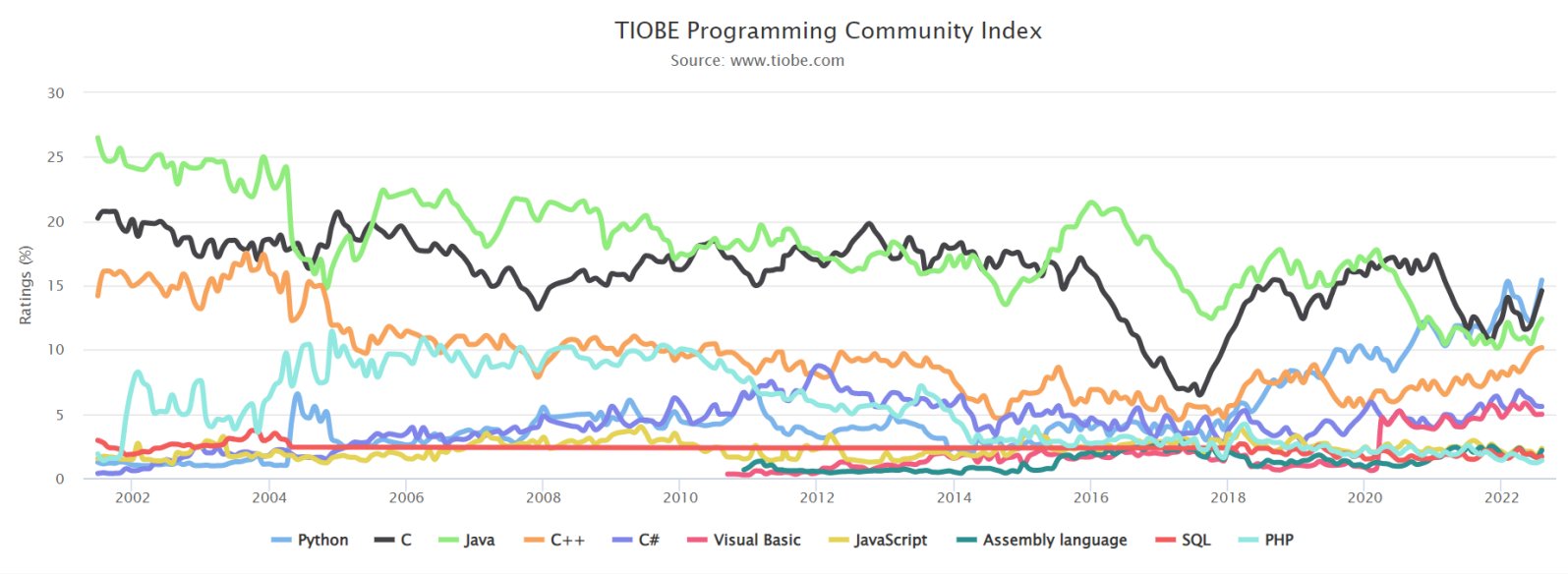
Versi Java

Version	Release date	End of Free Public Updates ^{[9][8][9][10]}	Extended Support Until
JDK Beta	1995	?	?
JDK 1.0	January 1996	?	?
JDK 1.1	February 1997	?	?
J2SE 1.2	December 1998	September 2003	?
J2SE 1.3	May 2000	?	?
J2SE 1.4	February 2002	October 2008	February 2013
Java SE 5	September 2004	November 2009	April 2015
Java SE 6	December 2006	April 2013	December 2018 December 2026 for Azul ^[11]
Java SE 7	July 2011	July 2019	July 2022
Java SE 8 (LTS)	March 2014	March 2022 for Oracle (commercial) December 2030 for Oracle (non-commercial) December 2030 for Azul May 2026 for IBM Semeru ^[12] At least May 2026 for Eclipse Adoptium At least May 2026 for Amazon Corretto	December 2030 ^[13]
Java SE 9	September 2017	March 2018 for OpenJDK	—
Java SE 10	March 2018	September 2018 for OpenJDK	—

Java SE 11 (LTS)	September 2018	September 2026 for Azul October 2024 for IBM Semeru ^[12] At least October 2024 for Eclipse Adoptium At least September 2027 for Amazon Corretto At least October 2024 for Microsoft ^{[14][15]}	September 2026 September 2026 for Azul ^[11]
Java SE 12	March 2019	September 2019 for OpenJDK	—
Java SE 13	September 2019	March 2020 for OpenJDK	—
Java SE 14	March 2020	September 2020 for OpenJDK	—
Java SE 15	September 2020	March 2021 for OpenJDK March 2023 for Azul ^[11]	—
Java SE 16	March 2021	September 2021 for OpenJDK	—
Java SE 17 (LTS)	September 2021	September 2029 for Azul October 2027 for IBM Semeru ^[12] At least September 2027 for Microsoft ^[14] At least September 2027 for Eclipse Adoptium	September 2029 or later September 2029 for Azul
Java SE 18	March 2022	September 2022 for OpenJDK and Adoptium	—
Java SE 19	September 2022	March 2023 for OpenJDK	—
Java SE 20	March 2023	September 2023 for OpenJDK	—
Java SE 21 (LTS)	September 2023	September 2028	September 2031 ^[13]

Legend: ■ Old version ■ Older version, still maintained ■ Latest version ■ Future release

Kenapa Belajar Java





Dimana Java Banyak Digunakan?

- Backend, banyak perusahaan besar saat ini menggunakan Java sebagai aplikasi backend nya seperti Twitter, Netflix, Spotify, Amazon, Alibaba, Bilibili, dan lain-lain
- Big Data, teknologi-teknologi big data yang saat ini populer, kebanyakan adalah teknologi Java, seperti Apache Hadoop, Elasticsearch, Apache Cassandra, Apache Spark, Apache Kafka, dan lain-lain
- Android, di Android kita bisa menggunakan Java dan Kotlin untuk membuat aplikasi nya



JRE vs JDK

- JRE singkatan dari Java Runtime Environment
- JDK singkatan dari Java Development Kit



Java Virtual Machine

- Java sendiri hanyalah bahasa pemrograman, otak dibalik teknologi Java sebenarnya sebuah teknologi yang disebut Java Virtual Machine
- Java Virtual Machine merupakan program yang digunakan untuk mengeksekusi binary file Java
- Karena JVM hanya mengerti binary file, sehingga akhirnya banyak bahasa pemrograman yang mengadopsi teknologi JVM, seperti Kotlin, Scala, Groovy dan lain-lain
- Dengan begitu, banyak bahasa pemrograman yang lebih canggih dari Java, namun mereka tetap jalan di JVM yang sudah terbukti stabil dan bagus

Instalasi Java



OpenJDK

- OpenJDK adalah salah satu implementasi Java Development Kit yang opensource dan gratis
- <https://openjdk.java.net/>



OpenJDK vs yang lain

- Oracle JDK : <https://www.oracle.com/java/technologies/javase-downloads.html>
- Amazon Corretto : <https://aws.amazon.com/id/corretto/>
- Zulu : <https://www.azul.com/downloads/zulu-community/>



Integrated Development Environment

- IDE adalah smart editor yang digunakan untuk mengedit kode program
- IDE juga digunakan untuk melakukan otomatisasi proses kompilasi kode program dan otomatisasi proses menjalankan program

Berikut beberapa IDE yang familiar untuk Java

- IntelliJ IDEA Ultimate / Community : <https://www.jetbrains.com/idea/>
- Eclipse : <https://www.eclipse.org/downloads/packages/>
- NetBeans : <https://netbeans.apache.org/>
- JDeveloper : <https://www.oracle.com/application-development/technologies/jdeveloper.html>

Tipe Data



Tipe Data Number

- Integer Number
- Floating Point Number



Integer Number

Tipe Data	Min	Max	Size	Default
byte	-128	127	1 byte	0
short	-32,768	32,767	2 bytes	0
int	-2,147,483,648	2,147,483,647	4 bytes	0
long	- 9,223,372,036, 854,775,808	9,223,372,036, 854,775,807	8 bytes	0



Kode : Integer Number

```
public class Number {  
    public static void main(String[] args) {  
        byte ini_byte = 10;           // min -128 max 127  
        short ini_short = 10000;       // min -32,768 max 32,767  
        int ini_int = 10000;  
        long ini_long = 10000;  
        long ini_long2 = 10000L;  
    }  
}
```



Floating Point Number

Tipe Data	Min	Max	Size	Default
float	3.4e-038	3.4e+038	4 bytes	0.0
double	1.7e-308	1.7e+308	8 bytes	0.0



Kode : Floating Point Number

```
public class Number {  
    public static void main(String[] args) {  
        float ini_float = 10000f;  
        double ini_double = 10000.0;  
    }  
}
```



Tipe Data Character

- Data Character (huruf) di Java direpresentasikan oleh tipe char.
- Untuk membuat data char di Java, kita bisa menggunakan tanda ' (petik satu) di awal dan di akhir karakter



Kode : Character

```
public class Char {  
    public static void main(String[] args) {  
        char a = 'A';  
        char k = 'K';  
        char u = 'U';  
  
        System.out.println(a);  
        System.out.println(k);  
        System.out.println(u);  
    }  
}
```



Tipe Data String

- Tipe data String adalah tipe data yang berisikan data kumpulan karakter atau sederhananya adalah teks
- Di Java, tipe data String direpresentasikan dengan kata kunci String
- Untuk membuat String di Java, kita menggunakan karakter " (petik dua) sebelum dan setelah teksnya
- Default value untuk String adalah null



Kode : String

```
public class MyString {  
    public static void main(String[] args) {  
        String firstName = "Riky";  
        String middleName = "Ahmad";  
        String lastName = "Fathoni";  
        String fullName = firstName + " " + middleName + " " + lastName;  
  
        System.out.println(firstName);  
        System.out.println(middleName);  
        System.out.println(lastName);  
        System.out.println(fullName);  
    }  
}
```



Tipe Data Boolean

- Tipe data boolean adalah tipe data yang memiliki 2 nilai, yaitu benar dan salah
- Tipe data boolean di Java direpresentasikan dengan kata kunci boolean
- Nilai benar direpresentasikan dengan kata kunci true
- Nilai salah direpresentasikan dengan kata kunci false
- Default value untuk boolean adalah false



Kode : Boolean

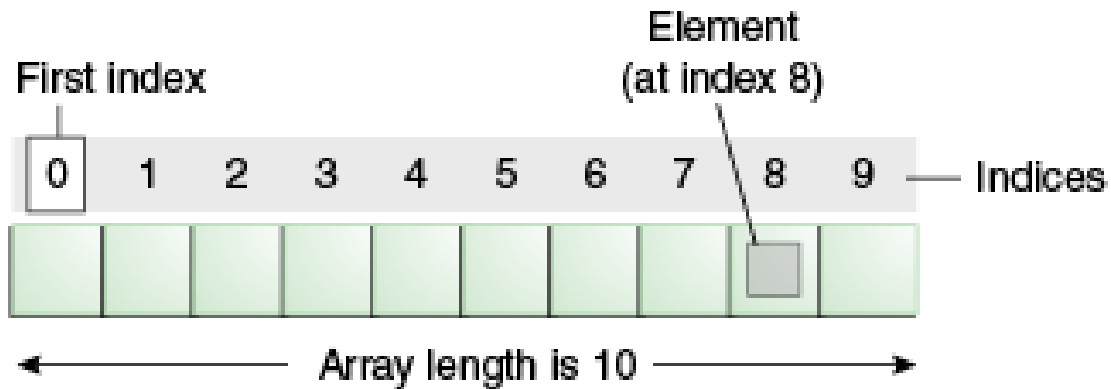
```
public class MyBoolean {  
  
    public static void main(String[] args) {  
        boolean benar = true;  
        boolean salah = false;  
  
        System.out.println(benar);  
        System.out.println(salah);  
    }  
}
```



Tipe Data Array

- Array adalah tipe data yang berisikan kumpulan data dengan tipe yang sama
- Jumlah data di Array tidak bisa berubah setelah pertama kali dibuat

Cara Kerja Array





Kode : Array_INITIALIZER

```
public class MyArray {  
  
    public static void main(String[] args) {  
        int[] arrayInt = new int[]{  
            1, 2, 3, 4, 5  
        };  
        long[] arrayLong = new long[]{  
            1000, 2000, 3000, 4000, 5000  
        };  
        String[] arrayString = new String[]{  
            "Satu", "Dua", "Tiga"  
        };  
    }  
}
```



Operasi di Array

Operasi Array	Keterangan
<code>array[index]</code>	Mengambil data di array
<code>array[index] = value</code>	Mengubah data di array
<code>array.length</code>	Mengambil panjang array



Kode : Operasi di Array

```
public class MyArray {  
    public static void main(String[] args) {  
        int[] arrayInt = new int[]{  
            1, 2, 3, 4, 5  
        };  
  
        arrayInt[0] = 100;  
  
        System.out.println("Array : " + arrayInt[0]);  
    }  
}
```




Kode : Array di dalam Array

```
public class MyArray {  
    public static void main(String[] args) {  
        String[][] arrayOfArrayString = new String[][]{  
            {"Satu", "Dua", "Tiga"},  
            {"Red", "Green", "Blue"}  
        };  
        System.out.println("Array : " + arrayOfArrayString[1][0]);  
    }  
}
```

Pengenalan Object Oriented Programming



Apa itu Object Oriented Programming?

- Object Oriented Programming adalah sudut pandang bahasa pemrograman yang berkonsep “objek”
- Ada banyak sudut pandang bahasa pemrograman, namun OOP adalah yang sangat populer saat ini.
- Ada beberapa istilah yang perlu dimengerti dalam OOP, yaitu: Object dan Class



Apa itu Class?

- Class adalah blueprint, prototype atau cetakan untuk membuat Object
- Class berisikan deklarasi semua properties dan functions yang dimiliki oleh Object
- Setiap Object selalu dibuat dari Class
- Dan sebuah Class bisa membuat Object tanpa batas



Apa itu Object?

- Object adalah data yang berisi field / properties / attributes dan method / function / behavior
- Semua data bukan primitif di Java adalah object, dari mulai Integer, Boolean, Character, String dan yang lainnya
- Object adalah hasil instansiasi dari sebuah class
- Untuk membuat object kita bisa menggunakan kata kunci new, dan diikuti dengan nama Class dan kurung ()



Kode : Object Car

```
package object;  
  
public class TestVehicle {  
    public static void main(String[] args) {  
        Car car1 = new Car();  
        Car car2 = new Car();  
        var car3 = new Car();  
    }  
}
```

Method



Method

- Method adalah block kode program yang akan berjalan saat kita panggil
- Sebelumnya kita sudah menggunakan method `println()` untuk menampilkan tulisan di console
- Untuk membuat method di Java, kita bisa menggunakan kata kunci `void`, lalu diikuti dengan nama method, kurung `()` dan diakhiri dengan block
- Kita bisa memanggil method dengan menggunakan nama method lalu diikuti dengan kurung `()`
- Di bahasa pemrograman lain, Method juga disebut dengan Function



Kode : Method

```
public static void main(String[] args) {  
    sayHelloWorld();  
}
```

```
static void sayHelloWorld(){  
    System.out.println("Hello World");  
}
```

```
}
```

```
|
```



Method Parameter

- Kita bisa mengirim informasi ke method yang ingin kita panggil
- Untuk melakukan hal tersebut, kita perlu menambahkan parameter atau argument di method yang sudah kita buat
- Cara membuat parameter sama seperti cara membuat variabel
- Parameter ditempatkan di dalam kurung () di deklarasi method
- Parameter bisa lebih dari satu, jika lebih dari satu, harus dipisah menggunakan tanda koma



Kode : Method Parameter

```
public static void main(String[] args) {  
    sayHello("Eko", "Khannedy");  
}  
  
static void sayHello(String firstName, String lastName) {  
    System.out.println("Hello " + firstName + " " + lastName);  
}  
}
```



Method Return Value

- Secara default, method itu tidak menghasilkan value apapun, namun jika kita ingin, kita bisa membuat sebuah method mengembalikan nilai
- Agar method bisa menghasilkan value, kita harus mengubah kata kunci void dengan tipe data yang dihasilkan
- Dan di dalam block method, untuk menghasilkan nilai tersebut, kita harus menggunakan kata kunci return, lalu diikuti dengan data yang sesuai dengan tipe data yang sudah kita deklarasikan di method
- Di Java, kita hanya bisa menghasilkan 1 data di sebuah method, tidak bisa lebih dari satu



Kode : Method Return Value

```
public static void main(String[] args) {  
    var a = 100;  
    var b = 200;  
    var c = sum(a, b);  
  
    System.out.println(c);  
}  
  
static int sum(int value1, int value2) {  
    var total = value1 + value2;  
    return total;  
}  
}
```



Kode : Kata Kunci var

```
var name; // error
name = "Eko Kurniawan Khannedy";

var age = 30;
var address = "Indonesia";

System.out.println(name);
System.out.println(age);
System.out.println(address);
```



Kata Kunci final

- Secara default, variable di Java bisa diubah-ubah nilainya
- Jika kita ingin membuat sebuah variable yang datanya tidak boleh diubah setelah pertama kali dibuat, kita bisa menggunakan kata kunci final
- Istilah variabel seperti ini, banyak juga yang menyebutnya konstan



Kode : Kata Kunci final

```
final String name = "Eko Kurniawan Khannedy";  
var age = 30;  
var address = "Indonesia";  
  
name = "Nama Diubah"; // error  
  
System.out.println(name);  
System.out.println(age);  
System.out.println(address);
```


—

Variable



Variable

- Variable adalah tempat untuk menyimpan data
- Java adalah bahasa static type, sehingga sebuah variable hanya bisa digunakan untuk menyimpan tipe data yang sama, tidak bisa berubah-ubah tipe data seperti di bahasa pemrograman PHP atau JavaScript
- Untuk membuat variable di Java kita bisa menggunakan nama tipe data lalu diikuti dengan nama variable nya
- Nama variable tidak boleh mengandung whitespace (spasi, enter, tab), dan tidak boleh seluruhnya number



Kode : Variable

```
String name;  
name = "Eko Kurniawan Khannedy";  
  
int age = 30;  
String address = "Indonesia";  
  
System.out.println(name);  
System.out.println(age);  
System.out.println(address);
```



Kata Kunci var

- Sejak versi Java 10, Java mendukung pembuatan variabel dengan kata kunci var, sehingga kita tidak perlu menyebutkan tipe datanya
- Namun perlu diingat, saat kita menggunakan kata kunci var untuk membuat variable, kita harus menginisiasi value / nilai dari variable tersebut secara langsung

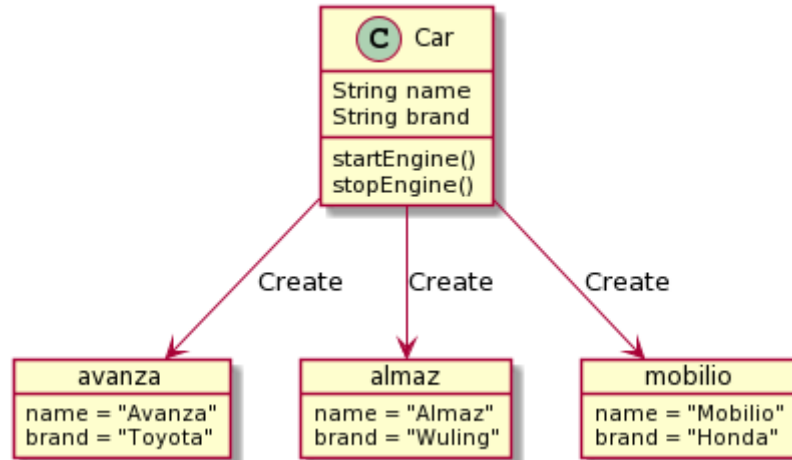
Inheritance



Inheritance

- Inheritance atau pewarisan adalah kemampuan untuk menurunkan sebuah class ke class lain
- Dalam artian, kita bisa membuat class Parent dan class Child
- Class Child, hanya bisa punya satu class Parent, namun satu class Parent bisa punya banyak class Child
- Saat sebuah class diturunkan, maka semua field dan method yang ada di class Parent, secara otomatis akan dimiliki oleh class Child
- Untuk melakukan pewarisan, di class child, kita harus menggunakan kata kunci extends lalu diikuti dengan nama class parent nya.

Class dan Object : Car



Kode : Inheritance

```
public class Vehicle {  
    protected String brand;  
    protected String name;  
  
    public Vehicle(String brand, String name) {  
        this.brand = brand;  
        this.name = name;  
    }  
  
    1 override  
    public void startEngine() {  
        System.out.println("Engine Started!");  
    }  
  
    1 override  
    public void stopEngine() {  
        System.out.println("Engine Stopped!");  
    }  
}  
  
3 public class Car extends Vehicle {  
4  
5     public Car(String brand, String name) {  
6         super(brand, name);  
7     }  
8  
9     @Override  
10    public void startEngine() {  
11        super.startEngine();  
12    }  
13  
14    @Override  
15    public void stopEngine() {  
16        super.stopEngine();  
17    }  
18 }  
19
```

Polymorphism



Polymorphism

- Polymorphism berasal dari bahasa Yunani yang berarti banyak bentuk.
- Dalam OOP, Polymorphism adalah kemampuan sebuah object berubah bentuk menjadi bentuk lain
- Polymorphism erat hubungannya dengan Inheritance



Jenis Polymorphism

- Static Polymorphism : menggunakan method overloading
- Dynamic Polymorphism : menggunakan method overriding



Aturan Method Overloading

- Nama method harus sama dengan method lainnya.
- Parameter haruslah berbeda.
- Return boleh sama, juga boleh berbeda.



Kode : Method Overloading

```
public class StaticPolymorphism {  
  
    public static void main(String[] args) {  
        var calculator = new Calculator();  
        int maxNumber = calculator.maxNumber( a: 1, b: 2);  
        double maxNumber2 = calculator.maxNumber( a: 10.0, b: 5.0);  
  
        System.out.println("Max number : " + maxNumber);  
        System.out.println("Max number : " + maxNumber2);  
    }  
}  
  
class Calculator {  
  
    public int maxNumber(int a, int b) {  
        return Math.max(a, b);  
    }  
  
    public double maxNumber(double a, double b) {  
        return Math.max(a, b);  
    }  
}
```



Kode : Method Overriding

```
class Hewan {
    protected void munculSuara() {
        System.out.println("Suara hewan");
    }
}

class Kucing extends Hewan {
    @Override
    protected void munculSuara() {
        System.out.println("Suara kucing: Meow... meow...");
    }
}

class Burung extends Hewan {
    @Override
    protected void munculSuara() {
        System.out.println("Suara burung: Cit... cit... cit...");
    }
}
```

```
public class DynamicPolymorphism {

    public static void main(String[] args) {
        var hewan = new Hewan();
        var kucing = new Kucing();
        var burung = new Burung();

        hewan.munculSuara();
        kucing.munculSuara();
        burung.munculSuara();
    }
}
```

Encapsulation



Encapsulation

- Encapsulation artinya memastikan data sensitif sebuah object tersembunyi dari akses luar
- Hal ini bertujuan agar kita bisa menjaga agar data sebuah object tetap baik dan valid
- Untuk mencapai ini, biasanya kita akan membuat semua field menggunakan access modifier private, sehingga tidak bisa diakses atau diubah dari luar
- Agar bisa diubah, kita akan menyediakan method untuk mengubah dan mendapatkan field tersebut



Getter dan Setter

- Di Java, proses encapsulation sudah dibuat standarisasinya, dimana kita bisa menggunakan Getter dan Setter method.
- Getter adalah function yang dibuat untuk mengambil data field
- Setter ada function untuk mengubah data field



Getter dan Setter Method

Tipe Data	Getter Method	Setter Method
boolean	isXxx()	setXxx(boolean value)
primitif	getXxx()	setXxx(primitif value)
Object	getXxx()	setXxx(Object value)



Kode : Getter dan Setter

```
public class Person {  
  
    public String name;  
    public int age;  
  
    public String getName() {  
        return name;  
    }  
  
    public void setName(String name) {  
        this.name = name;  
    }  
  
    public int getAge() {  
        return age;  
    }  
  
    public void setAge(int age) {  
        this.age = age;  
    }  
}
```

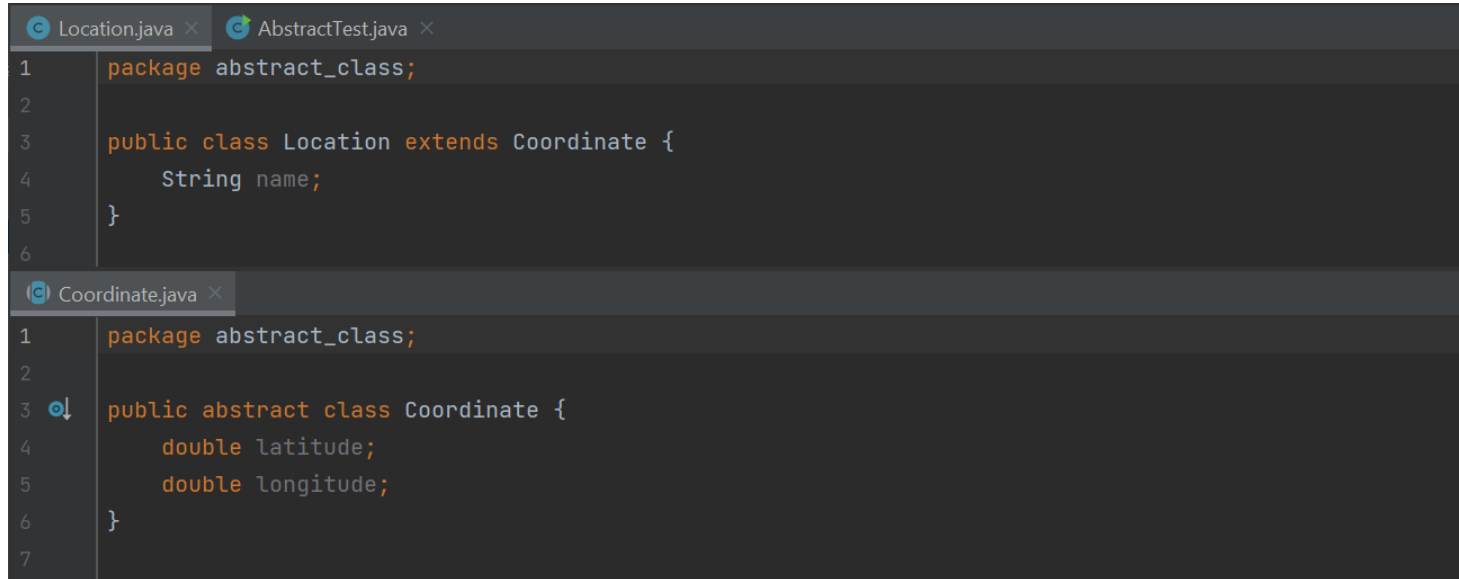
Abstract Class



Abstract Class

- Saat kita membuat class, kita bisa menjadikan sebuah class sebagai abstract class.
- Abstract class artinya, class tersebut tidak bisa dibuat sebagai object secara langsung, hanya bisa diturunkan
- Untuk membuat sebuah class menjadi abstract, kita bisa menggunakan kata kunci abstract sebelum kata kunci class
- Dengan demikian abstract class bisa kita gunakan sebagai kontrak untuk class child

Code : Abstract Class



The screenshot shows an IDE with two tabs: 'Location.java' and 'AbstractTest.java'. The 'Location.java' tab is active, displaying the following code:

```
1 package abstract_class;
2
3 public class Location extends Coordinate {
4     String name;
5 }
6
```

Below this, the 'Coordinate.java' tab is visible, displaying the following code:

```
1 package abstract_class;
2
3 public abstract class Coordinate {
4     double latitude;
5     double longitude;
6 }
7
```

Kode : Membuat Abstract Class

```
AbstractTest.java x
1 package abstract_class;
2
3 public class AbstractTest {
4     public static void main(String[] args) {
5         var location = new Location();
6         var coordinate = new Coordinate(); //Terjadi error pada abstract class
7     }
8 }
9
```

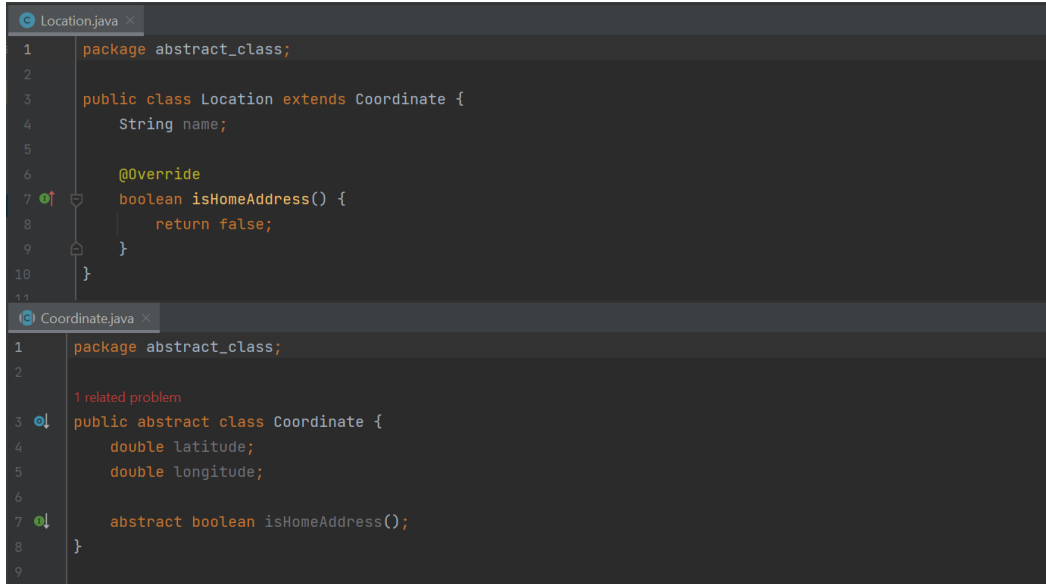
Abstract Method



Abstract Method

- Saat kita membuat class yang abstract, kita bisa membuat abstract method juga di dalam class abstract tersebut
- Saat kita membuat sebuah abstract method, kita tidak boleh membuat block method untuk method tersebut
- Artinya, abstract method wajib di override di class child
- Abstract method tidak boleh memiliki access modifier private

Code : Abstract Method



```
Location.java x
1 package abstract_class;
2
3 public class Location extends Coordinate {
4     String name;
5
6     @Override
7     boolean isHomeAddress() {
8         return false;
9     }
10 }
11

Coordinate.java x
1 package abstract_class;
2
3 1 related problem
4 public abstract class Coordinate {
5     double latitude;
6     double longitude;
7
8     abstract boolean isHomeAddress();
9 }
```

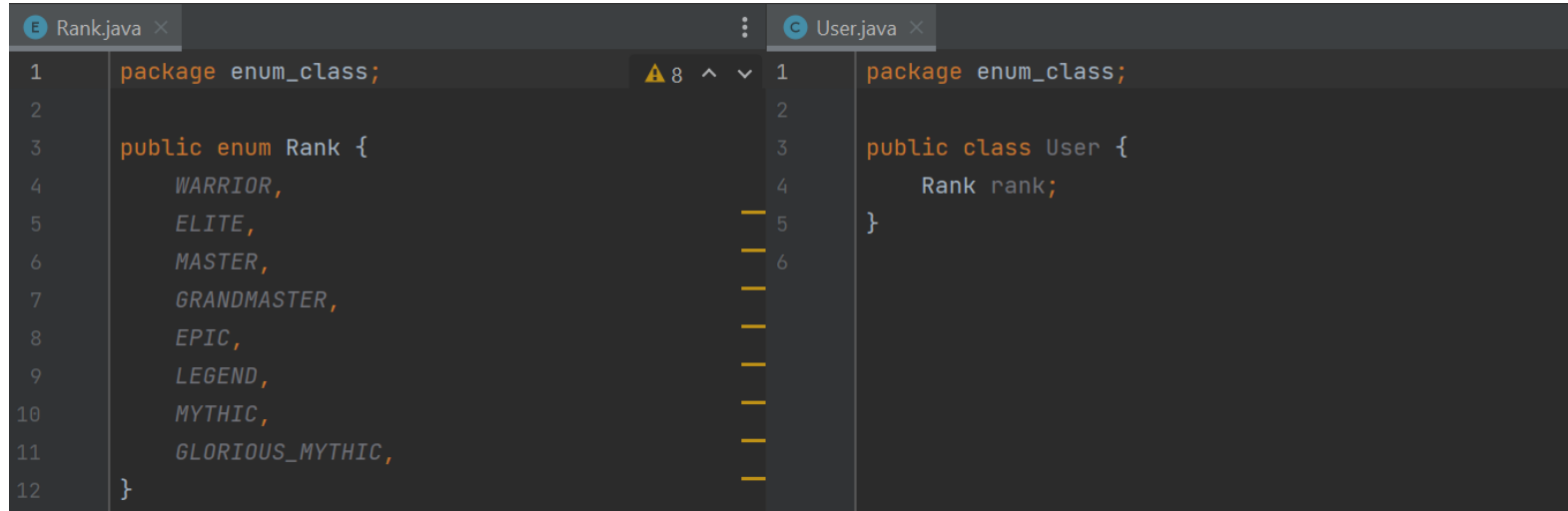
Enum Class



Enum Class

- Saat kita membuat aplikasi, kadang kita akan bertemu dengan jenis-jenis data yang nilainya terbatas
- Misal, gender, ada male dan female, atau tipe customer, ada standard, premium atau vip, dan lain-lain
- Dalam kasus seperti ini, kita bisa menggunakan enum class, yaitu class yang berisikan nilai terbatas yang sudah ditentukan
- Saat membuat enum class, secara otomatis dia akan meng-extends class `java.lang.Enum`, oleh karena itu class enum tidak bisa extends class lain, namun masih tetap bisa implements interface.

Kode : Membuat Enum Class



```
Rank.java
1 package enum_class;
2
3 public enum Rank {
4     WARRIOR,
5     ELITE,
6     MASTER,
7     GRANDMASTER,
8     EPIC,
9     LEGEND,
10    MYTHIC,
11    GLORIOUS_MYTHIC,
12 }

User.java
1 package enum_class;
2
3 public class User {
4     Rank rank;
5 }
6
```

Kode : Menggunakan Enum

```
RankTest.java x Rank.java x User.java x
1 package enum_class;
2
3 public class RankTest {
4     public static void main(String[] args) {
5         var user = new User();
6         user.rank = Rank.LEGEND;
7
8         System.out.println("User rank : " + user.rank);
9     }
10 }
```



Enum Members

- Sama seperti class biasanya, di class enum pun kita bisa menambahkan members (field, method dan constructor)
- Khusus constructor, kita tidak bisa membuat public constructor, karena memang tujuan enum bukan untuk di instansiasi secara bebas

Kode : Members di Enum

```
Hokage.java x
1 package enum_class;
2
3 public enum Hokage {
4     HOKAGE_1( name: "Hashirama"),
5     HOKAGE_2( name: "Tobirama"),
6     HOKAGE_3( name: "Sarutobi"),
7     HOKAGE_4( name: "Minato"),
8     HOKAGE_5( name: "Tsunade"),
9     HOKAGE_6( name: "Kakashi"),
10    HOKAGE_7( name: "Naruto");
11
12    String name;
13
14    Hokage(String name) {
15        this.name = name;
16    }
17 }
```


Kode : Konversi Enum

```
EnumConvertTest.java x
1  package enum_class;
2
3  ▶ public class EnumConvertTest {
4
5  ▶   public static void main(String[] args) {
6      var name :String = Hokage.HOKAGE_1.name;
7      var asString :String = Hokage.HOKAGE_2.toString();
8      var asEnum :Hokage = Hokage.valueOf( name: "HOKAGE_7");
9      var enumList :Hokage[] = Hokage.values();
10
11      System.out.println("name : " + name);
12      System.out.println("to string : " + asString);
13      System.out.println("to enum : " + asEnum);
14      System.out.println("enum length : " + enumList.length);
15  }
16  }
```

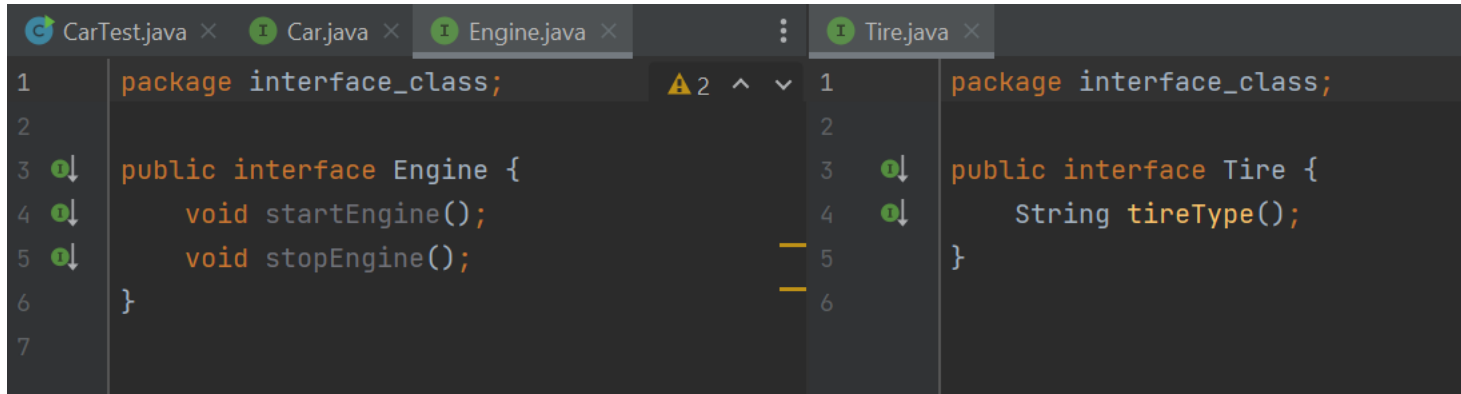
Interface



Interface

- Sebelumnya kita sudah tahu bahwa abstract class bisa kita gunakan sebagai kontrak untuk class child nya.
- Namun sebenarnya yang lebih tepat untuk kontrak adalah Interface
- Jangan salah sangka bahwa Interface disini bukanlah User Interface
- Interface mirip seperti abstract class, yang membedakan adalah di Interface, semua method otomatis abstract, tidak memiliki block
- Di interface kita tidak boleh memiliki field, kita hanya boleh memiliki constant (field yang tidak bisa diubah)
- Untuk mewariskan interface, kita tidak menggunakan kata kunci extends, melainkan implements

Kode : Membuat Interface



The screenshot displays an IDE with two open files: `Engine.java` and `Tire.java`. Both files are in the `package interface_class;` namespace. `Engine.java` defines a `public interface Engine` with two methods: `void startEngine();` and `void stopEngine();`. `Tire.java` defines a `public interface Tire` with one method: `String tireType();`. The IDE interface shows line numbers on the left of each code block. The `Engine.java` editor has a yellow warning icon and the number '2' in the top right corner.

```
1 package interface_class;
2
3 public interface Engine {
4     void startEngine();
5     void stopEngine();
6 }
7
```

```
1 package interface_class;
2
3 public interface Tire {
4     String tireType();
5 }
6
```



Interface Inheritance

- Sebelumnya kita sudah tahu kalo di Java, child class hanya bisa punya 1 class parent
- Namun berbeda dengan interface, sebuah child class bisa implement lebih dari 1 interface
- Bahkan interface pun bisa implement interface lain, bisa lebih dari 1. Namun jika interface ingin mewarisi interface lain, kita menggunakan kata kunci extends, bukan implements

Kode : Interface Inheritance

```
CarTest.java × Car.java ×  
1 package interface_class;  
2  
3 1↓ public interface Car extends Engine, Tire {  
4 1↓     String name();  
5 1↓     String brand();  
6     }  
7
```

Code : Implement Interface Inheritance

```
CarTest.java x Car.java x CarImplement.java x
1 package interface_class;
2
3 public class CarImplement implements Car {
4
5     @Override
6     public String name() {
7         return "CX-9";
8     }
9
10    @Override
11    public String brand() { return "Mazda"; }
12
13
14    @Override
15    public void startEngine() {
16        System.out.println("Engine Started!");
17    }
18 }
```

```
19
20    @Override
21    public void stopEngine() {
22        System.out.println("Engine Stopped!");
23    }
24
25    @Override
26    public String tireType() { return "195/25 R16"; }
27 }
```

Access Modifiers



Access Modifier

- Access modifier adalah kemampuan membuat class, field, method dan constructor dapat diakses dari mana saja
- Sebelumnya teman-teman sudah melihat 2 access modifier, yaitu public dan default (no-modifier)
- Sekarang kita akan bahas access modifier lainnya



Access Level

Modifier	Class	Package	Subclass	World
public	Y	Y	Y	Y
protected	Y	Y	Y	N
<i>no modifier</i>	Y	Y	N	N
private	Y	N	N	N



Public Class

- Saat kita membuat public class, kita hanya bisa membuat 1 public class di 1 file java
- Selain itu, nama public class harus sama dengan nama file



Kode : Access Modifier (1)

```
3      public class Product {  
4          protected String name;  
5          protected int price;  
6  
7          public String getName() {  
8              return name;  
9          }  
10  
11         public int getPrice() {  
12             return price;  
13         }  
14     }  
15 }
```

Kode : Access Modifier (2)

```
3 ▶ public class ProductApp {
4 ▶   public static void main(String[] args) {
5       Product product = new Product();
6       product.name = "Indomie"; // bisa diakses karena di package sama
7       product.price = 2000; // // bisa diakses karena di package sama
8
9       System.out.println(product.getName());
10      System.out.println(product.getPrice());
11  }
12 }
```

Annotation



Annotation

- Annotation adalah menambahkan metadata ke kode program yang kita buat
- Tidak semua orang membutuhkan Annotation, biasanya Annotation digunakan saat kita ingin membuat library / framework
- Annotation sendiri bisa diakses menggunakan Reflection, yang akan kita bahas nanti
- Untuk membuat annotation, kita bisa menggunakan kata kunci `@interface`
- Annotation hanya bisa memiliki method dengan tipe data sederhana, dan bisa memiliki default value



Attribute Annotation

Attribute	Keterangan
@Target	Memberitahu annotation bisa digunakan di mana? Class, method, field, dan lain-lain
@Retention	Memberitahu annotation apakah disimpan di hasil kompilasi, dan apakah bisa dibaca oleh reflection?



Kode : Membuat Annotation

```
3  import java.lang.annotation.ElementType;
4  import java.lang.annotation.Retention;
5  import java.lang.annotation.RetentionPolicy;
6  import java.lang.annotation.Target;
7
8  @Target(value = {ElementType.FIELD})
9  @Retention(RetentionPolicy.RUNTIME)
10 public @interface NotEmpty {
11
12 }
```



Kode : Menggunakan Annotation

```
3  import annotation_class.NotEmpty;
4
5  public class LoginRequest {
6
7      @NotEmpty
8      private String username;
9
10     @NotEmpty
11     private String password;
12
13     public void setUsername(String username) {
14         this.username = username;
15     }
16
17     public void setPassword(String password) {
18         this.password = password;
19     }
19 }
```



Predefined Annotation

Java juga sudah memiliki annotation bawaan, seperti :

- `@Override`, untuk menandai bahwa method yang meng-override method parent class nya
- `@Deprecated`, untuk menandai bahwa method tersebut tidak direkomendasikan lagi untuk digunakan
- `@FunctionalInterface`, untuk menandai bahwa class tersebut bisa dibuat sebagai lambda expression
- dan lain-lain

Reflection



Reflection

- Reflection adalah kemampuan melihat struktur aplikasi kita pada saat berjalan
- Reflection biasanya sangat berguna saat kita ingin membuat library ataupun framework, sehingga bisa meng-otomatisasi pekerjaan
- Untuk mengakses reflection class dari sebuah object, kita bisa menggunakan method `getClass()` atau `NamaClass.class`



Kode : Annotation di Field

```
3  import annotation_class.NotEmpty;
4
5  public class LoginRequest {
6
7      @NotEmpty
8      private String username;
9
10     @NotEmpty
11     private String password;
12
13     public void setUsername(String username) {
14         this.username = username;
15     }
16
17     public void setPassword(String password) {
18         this.password = password;
19     }
19 }
```

Kode : Validasi Menggunakan Reflection

```
9 @ public static void validateEmpty(Object object) {
10     Class<?> clazz = object.getClass();
11     Field[] fields = clazz.getDeclaredFields();
12
13     for (var field : fields) {
14         if (field.getAnnotation(NotEmpty.class) != null) {
15             field.setAccessible(true);
16             try {
17                 String value = (String) field.get(object);
18                 if (value == null || value.isEmpty()) {
19                     throw new RuntimeException("Field " + field.getName() + " is empty");
20                 }
21             } catch (IllegalAccessException e) {
22                 System.out.println("Tidak dapat mengakses field : " + field.getName());
23             }
24         }
25     }
26 }
```

Kode : Implementasi Validasi Reflection

```
3  import sample.LoginRequest;
4  import sample.ValidationUtils;
5
6  public class AnnotationTest {
7
8      public static void main(String[] args) {
9          var loginRequest = new LoginRequest();
10         loginRequest.setUsername("Mahasiswa");
11         loginRequest.setPassword("Password");
12
13         ValidationUtils.validateEmpty(loginRequest);
14     }
15 }
```

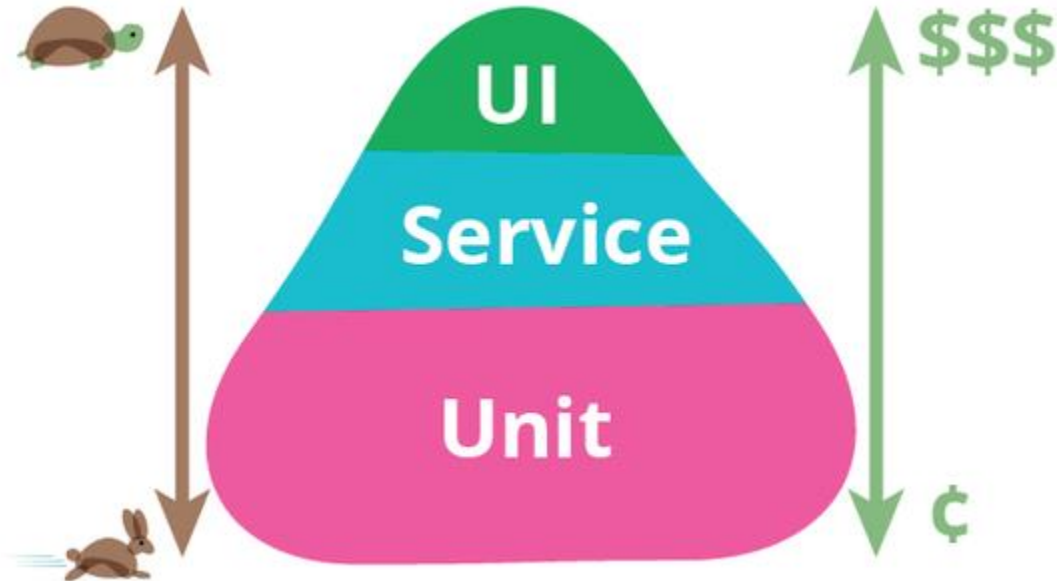
Pengenalan Software Testing



Pengenalan Software Testing

- Software testing adalah salah satu disiplin ilmu dalam software engineering
- Tujuan utama dari software testing adalah memastikan kualitas kode dan aplikasi kita baik
- Ilmu untuk software testing sendiri sangatlah luas, pada materi ini kita hanya akan fokus ke unit testing

Test Pyramid



Unit Test



Unit Test

- Unit test akan fokus menguji bagian kode program terkecil, biasanya menguji sebuah method
- Unit test biasanya dibuat kecil dan cepat, oleh karena itu biasanya kadang kode unit test lebih banyak dari kode program aslinya, karena semua skenario pengujian akan dicoba di unit test
- Unit test bisa digunakan sebagai cara untuk meningkatkan kualitas kode program kita



JUnit

- JUnit adalah test framework yang paling populer di Java
- Saat ini versi terbaru JUnit adalah versi 5
- JUnit 5 membutuhkan Java minimal versi 8
- <https://junit.org/>



Membuat Test

- Untuk membuat test di JUnit itu sederhana, kita cukup membuat class, lalu menambahkan method-method test nya
- Method akan dianggap sebuah test jika ditambahkan annotation `@Test`
- Kode test disimpan dibagian test folder di maven, bukan di main folder
- Biasanya saat membuat class untuk test, rata-rata orang biasa membuat nama class nya sama dengan nama class yang akan di test, tapi diakhiri dengan kata Test, misal jika nama class nya adalah Calculator, maka nama class test nya adalah CalculatorTest



Assertions

- Saat membuat test, kita harus memastikan bahwa test tersebut sesuai dengan ekspektasi yang kita inginkan
- Jika manual, kita bisa melakukan pengecekan if else, namun itu tidak direkomendasikan
- JUnit memiliki fitur untuk melakukan assertions, yaitu memastikan bahwa unit test sesuai dengan kondisi yang kita inginkan
- Assertions di JUnit di representasikan dalam class Assertions, dan di dalamnya terdapat banyak sekali function static
- <https://junit.org/junit5/docs/current/api/org.junit.jupiter.api/org/junit/jupiter/api/Assertions.html>



Menonaktifkan Test

- Kadang ada kalanya kita ingin menonaktifkan unit test, misal karena terjadi error di unit test tersebut, dan belum bisa kita perbaiki
- Sebenarnya cara paling mudah untuk menonaktifkan unit test adalah dengan menghapus annotation `@Test`, namun jika kita lakukan itu, kita tidak bisa mendeteksi kalo ada unit test yang di disabled
- Untuk menonaktifkan unit test secara benar, kita bisa menggunakan annotation `@Disabled`



Sebelum & Setelah Unit Test

- Kadang kita ingin menjalankan kode yang sama sebelum dan setelah eksekusi unit test
- Hal ini sebenarnya bisa dilakukan secara manual di function @Test nya, namun hal ini akan membuat kode duplikat banyak sekali
- JUnit memiliki annotation @BeforeEach dan @AfterEach
- @BeforeEach digunakan untuk menandai function yang akan dieksekusi sebelum unit test dijalankan
- @AfterEach digunakan untuk menandai function yang akan dieksekusi setelah unit test dijalankan
- Ingat, bahwa ini akan selalu dieksekusi setiap kali untuk function @Test, bukan sekali untuk class test saja



Sebelum & Setelah Semua Unit Test

- @BeforeEach & @AfterEach akan dieksekusi setiap kali function @Test jalan
- Namun kadang kita ingin melakukan sesuatu sebelum semua unit test berjalan, atau setelah semua unit test berjalan
- Ini bisa dilakukan menggunakan annotation @BeforeAll dan @AfterAll
- Namun hanya static function yang bisa menggunakan @BeforeAll dan @AfterAll



Membatalkan Test

- Kadang kita ingin membatalkan unit test ketika kondisi tertentu terjadi
- Untuk membatalkan, kita bisa menggunakan exception `TestAbortedException`
- Jika JUnit mendapatkan exception `TestAbortedException`, secara otomatis test tersebut akan dibatalkan



Menggunakan Assumptions

- Sebelumnya kita sudah tahu jika ingin membatalkan test, kita bisa menggunakan exception `TestAbortException`
- Namun sebenarnya ada cara yang lebih mudah, yaitu dengan menggunakan `Assumptions`
- Penggunaan `Assumptions` mirip seperti `Assertions`, jika nilainya tidak sama, maka function `Assumptions` akan throw `TestAbortException`, sehingga secara otomatis akan membatalkan unit test yang sedang berjalan
- <https://junit.org/junit5/docs/current/api/org.junit.jupiter.api/org/junit/jupiter/api/Assumptions.html>



Materi Selesai

Quiz



Quiz (1) – Knowledge Check

- Jelaskan apa yang kalian ketahui tentang Pemrograman Berorientasi Objek?
- Jelaskan apa perbedaan antara Inner Class dan Sub-Class?
- Jelaskan apa perbedaan class Abstrak dan Interface?
- Jelaskan apa itu primitive data di dalam java? Dan apakah String termasuk primitive data?
- Bisakah kita mendeklarasikan class Abstrak tanpa memiliki metode abstrak? Jelaskan Alasannya.
- Bisakah kita memanggil konstruktor class di dalam konstruktor turunan? Jelaskan Alasannya.
- Bisakah suatu class memiliki lebih dari satu konstruktor class? Jelaskan Alasannya.
- Bisakah kita override method static pada suatu class? Jelaskan Alasannya.



Quiz (2) – Membuat Kode

- Buatlah suatu Object apapun yang saling terikat dengan mengimplementasi berikut ini :
 - a. Abstraction (class dan method)
 - b. Inheritance
 - c. Polymorphism (static dan dynamic)
 - d. Encapsulation



Quiz - Prosedur

- Buatlah kelompok yang terdiri 4-5 orang
- Durasi pengerjaan kurang lebih 3 jam, bisa fleksibel tergantung waktu.
- Kirim hasil pengerjaan kuis ke alamat email riky.fathoni@gmail.com