

VICON Minggu Ke-5

Pemrograman - 1

**KEUNTUNGAN JADI PROGRAMMER
DI ZAMAN SEKARANG ADALAH
LEBIH MUDAH UNTUK MEMULAI START-UP,
BAHKAN BISA SENDIRIAN**

**KALAU GAK MAU BIKIN START-UP
BISA JADI FREELANCER**

WAKTU LUANG

Programmer

Belajar skill baru

Kerjain project freelance

Bikin product

Kalau sudah ada
product yg stabil
bisa setup team, jalan-jalan
dan tetap dapat duit

Non Teknikal

Netflix

Ngumpul sama teman

Jalan-jalan

Sebagian orang akan
mulai bangun
relasi & bisnis

1. Project Manajemen

2. Coding

3. Copywriting

4. Public Speaking

5. App Development

6. SEO/Google Analytics

7. Facebook Ads

8. Excel Knowledge

9. Web Development

10. UX Design

11. Photoshop

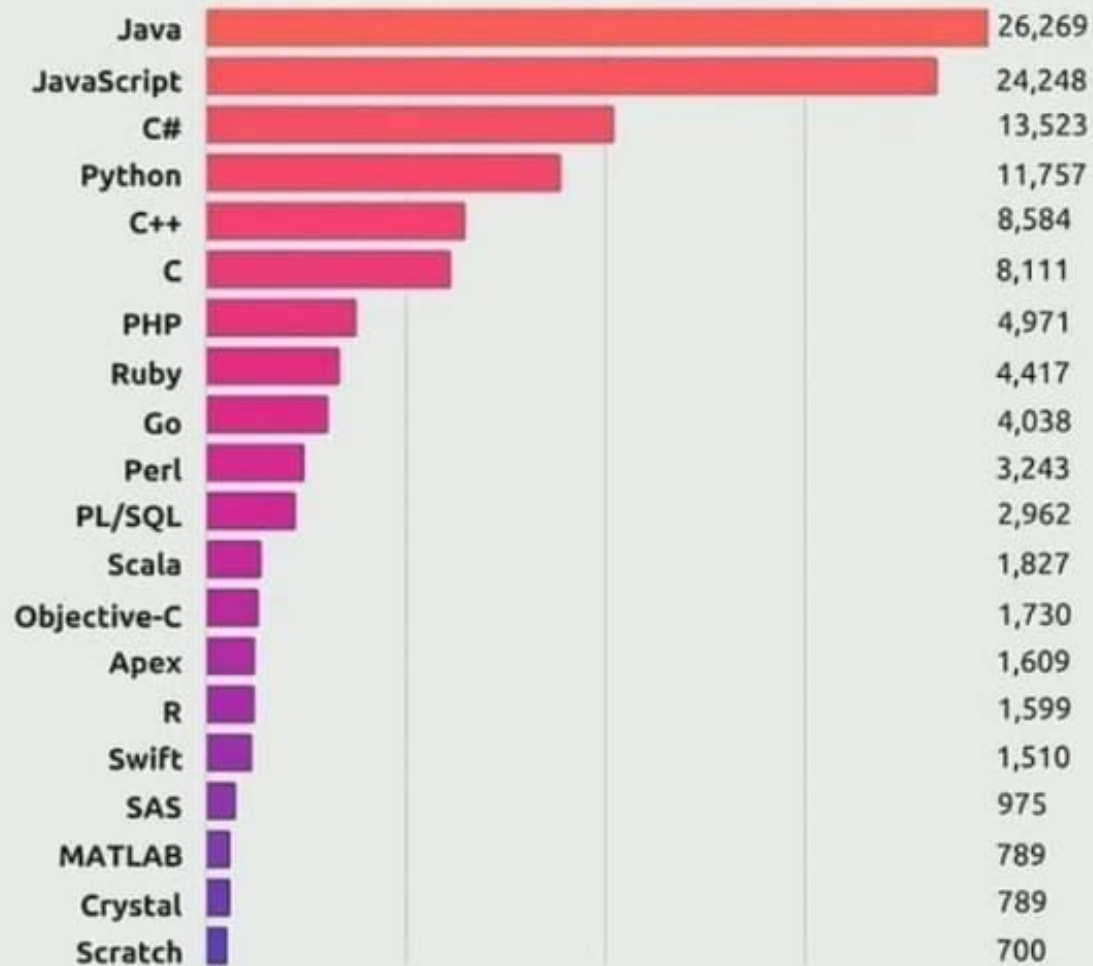
12. Sosial Media Marketing

#HOMEWITHBOOKS

🌐 bukuanakit.com

📷 [@buku.anakit](https://www.instagram.com/buku.anakit)





6 Tool Paling Berguna untuk Programmer



Text Editor



GitHub

Dev Platform



Version Control
System



POSTMAN

API Platform



Communication
Platform



List Making App

Materi

CARA INSTALL

IDENTIFIER

TIPE DATA

IF ELSE SWITCH

ARRAY

CLASS OBJECT

CONSTRUCT

INHERITANCE

ABSTRACT - INTERFACE

STRING BUFFER

EXCEPTION - ERROR - BUG

STREAM

THREADS

NETWORK

APPLET

BASIC GUI

DATABASE

AUTH

7. Construct

- Default Constructor
- Overloading Constructor
- Menggunakan Constructor
- Pemanggilan Constructor dengan this()
- Package
- Mengimport Package
- Membuat Package
- Pengaturan CLASSPATH
- Access Modifiers
- Akses Default
- Akses Public
- Akses Protected
- Akses Private



Constructor

Constructor adalah method khusus yang akan dieksekusi pada saat pembuatan objek **(instance)**.

Biasanya method ini digunakan untuk inisialisasi atau **mempersiapkan data untuk objek.**

Constructor

Aturan dalam penulisan Constructor

<modifier> <className> (<parameter>*)

Contoh jika nama classnya adalah Contoh_Constructor

```
package Bag_7_Construct;  
public class Contoh_Constructor {  
    public Contoh_Constructor() {  
        System.out.print("Ini adalah constructor");  
    }  
    public static void main(String[] ini_argumen) {  
        new Contoh_Constructor();  
    }  
}
```

Overloading Overriding

Overloading adalah teknik menggunakan Construct lebih dari 1, pembedanya adalah jumlah dan tipe data pada parameters-nya.

Overriding adalah sebuah method yang terdapat pada subclass yang nama method nya sama seperti method pada superclass.

Overloading = ada parameter

Overriding = tidak ada parameter

Overloading Overriding

Pada Constructor Overloading dan Overriding, parameters dapat disesuaikan

<modifier> <className> (<parameter>*)

Dapat digunakan atau tidak

```
package Bag_7_Construct;
public class Contoh_Constructor {
    public Contoh_Constructor () {
        System.out.print("Ini adalah constructor" );
    }
    public static void main(String[] ini_argumen) {
        new Contoh_Constructor ();
    }
}
```

```
package Bag_7_Construct;

public class Contoh_Constructor {
    public Contoh_Constructor () {
        System.out.println("Ini adalah constructor");
    }
    public Contoh_Constructor (String parameter_satu) {
        System.out.println("dengan parameters = " + parameter_satu);
    }
    public static void main(String[] ini_argumen) {
        new Contoh_Constructor ();
        new Contoh_Constructor ("ini data parameter" );
    }
}
```

Constructor - This

This digunakan untuk pemanggilan atau perubahan data variabel **antar constructor.**

Constructor - This

```
package Bag_7_Construct;

public class Contoh_Constructor_This {
    public String nama;
    public Contoh_Constructor_This() {
        this.nama = "Construct Default";
    }

    public Contoh_Constructor_This(String ini_parameters) {
        this.nama = ini_parameters;
    }

    public static void main(String[] ini_argumen) {
        Contoh_Constructor_This variabel_satu = new Contoh_Constructor_This();
        System.out.println(variabel_satu.nama);
        Contoh_Constructor_This variabel_dua = new Contoh_Constructor_This("ini data parameter");
        System.out.println(variabel_dua.nama);
    }
}
```

Package

Package digunakan pada Java sebenarnya **untuk memudahkan mengorganisir file dari class**.

Package ini merupakan mekanisme dari encapsulation suatu kelompok atau grup **yang terdiri dari class – class**, sub packages dan juga interfaces. Ini akan memberi kemudahan, karena banyak penerapan dari program java menggunakan konsep hirarki.

Ketika aplikasi semakin bertambah kompleks, maka package ini dapat membantu anda mengatur komponen-komponen di dalamnya tanpa harus membuat ulang suatu method.

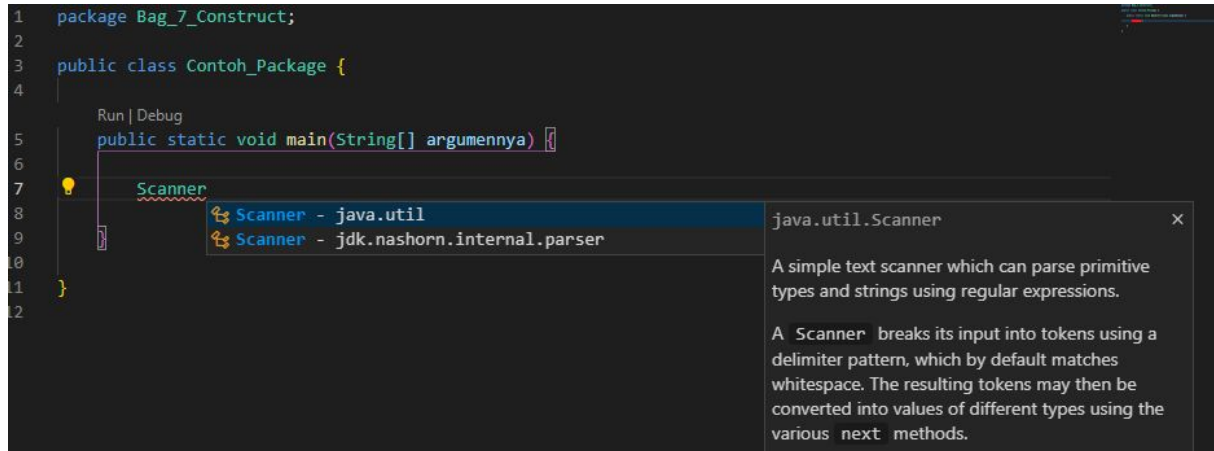
Package juga memfasilitasi **penggunaan kembali software dengan pernyataan import dari package lainnya**, ini lebih baik jika dibandingkan anda harus menyalin class-class tersebut ke dalam setiap program yang menggunakannya.

Package - Cara Meng-import

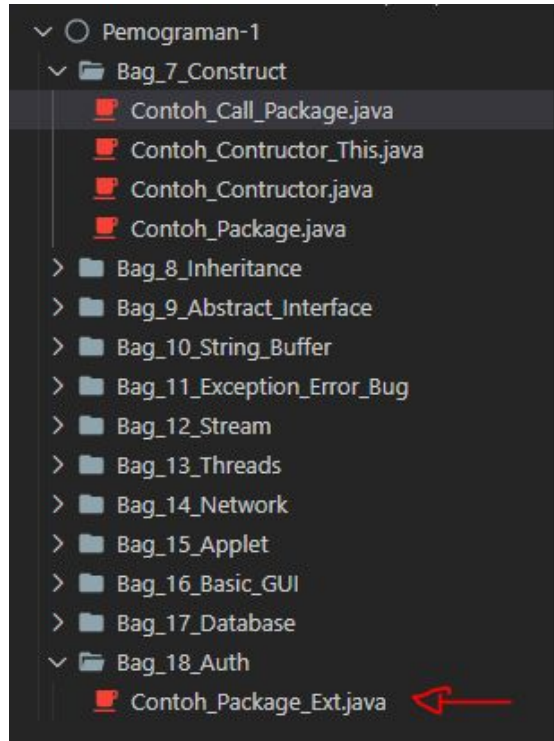
Pada pengaplikasiannya, penggunaan package dengan sintaks

`import <namaPaket>;`

Pada Visual Studio Code, import package akan lebih mudah dibantu oleh plugins yang terinstall pada IDE nya



Package - Membuat

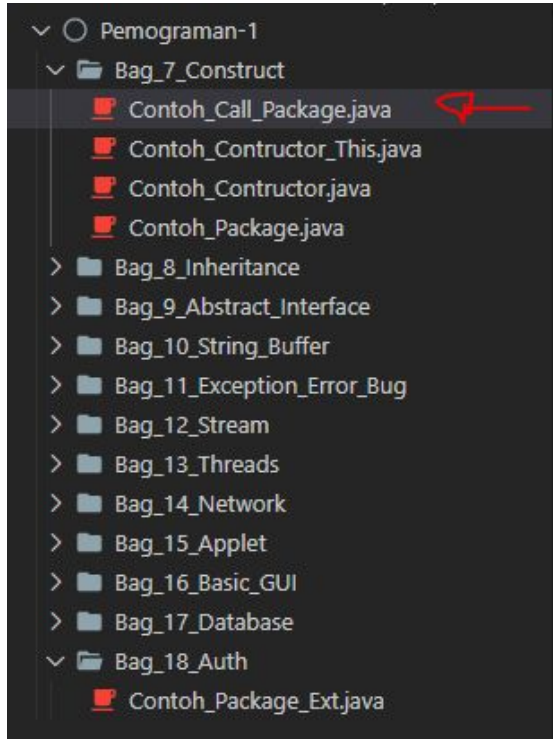


```
package Bag_18_Auth;

public class Contoh_Package_Ext {

    public String panggilKota() {
        return "Bali";
    }
}
```

Package - Memanggil



```
package Bag_7_Construct;

import Bag_18_Auth.Contoh_Package_Ext;

public class Contoh_Call_Package {

    String nama_kota;

    public Contoh_Call_Package() {
        Contoh_Package_Ext ini_variabel = new Contoh_Package_Ext();
        this.nama_kota = ini_variabel.panggilKota();
    }

    public static void main(String[] argumennya) {
        Contoh_Call_Package variabel_lagi = new Contoh_Call_Package();
        System.out.println(variabel_lagi.nama_kota);
    }
}
```

Modifier

Default

Hak akses ini, variabel/method dapat diakses dari class lain asalkan masih dalam satu package yang sama.

```
string nama;
```

Public

Hak akses yang dapat diakses dari mana saja walaupun diluar project.

```
public String nama;
```

Modifier

Protect

Biasanya digunakan untuk mewariskan variabel yang ada di super class terhadap child class.

```
protect String nama;
```

Private

Hak akses yang hanya dapat diakses dalam 1 file.

```
private String nama;
```

8. Inheritance

Mendefinisikan Superclass dan Subclass
Kata Kunci Super
Overriding Method
Method final dan class final
Polimorfisme



Inheritance

Inheritance (turunan atau pewarisan) adalah konsep yang membolehkan penggunaan kelas yang sudah ada dijadikan sebagai superclass untuk mendefinisikan kelas baru turunannya.

Extends

Kata kunci **extends** digunakan untuk mendeklarasikan kelas turunan (subclass) dari kelas atasnya (superclass). Dengan kata kunci extends, kelas turunan akan mewarisi data field dan metoda yang dimiliki oleh kelas di atasnya.

Sebenarnya tidak semua data field dan metoda akan diwariskan kepada kelas turunannya. **Data field dan method yang dideklarasikan menggunakan modifier private, tidak diwariskan kepada kelas turunannya.** Berikut ini adalah sintaks untuk mendeklarasikan kelas turunan (subclass) dari superclass:

```
public class nama-subclass extends nama-superclass {  
    // data dan metoda  
}
```

Superclass dan Subclass

Kelas atas atau kelas asal sering disebut dengan superclass, base class atau parent class, sedangkan kelas turunannya sering disebut dengan child class, derived class atau extended class.

```
package Bag_8_Inheritance;

public class SuperClass {

    public String SubClass() {
        return "ini dari sub class";
    }
}
```


Keyword Super

Sebelumnya anda sudah mengetahui mengenai **keyword this** sebagai referensi untuk objek yang dipanggil. Sedangkan keyword super akan mengacu pada superclass.

Ketika suatu subclass menerima pewarisan data fields yang bisa diakses dan method-methodnya dari superclass, apakah constructor dari superclass juga akan ikut di wariskan pada sub class tersebut.

Keyword Super

Apakah constructor dari superclass bisa dipanggil melalui sub class?

Fungsi dari keyword super itu sendiri adalah:

1. Memanggil constructor dari superclass
2. Memanggil method dari superclass
3. Mengakses variabel instance superclass

Keyword Super

Constructor digunakan untuk membentuk instance dari suatu class. Namun, tidak seperti properti dan method, constructor ini tidak diwariskan pada subclass.

Sehingga constructor dari subclass ini hanya dapat dipanggil melalui constructor dari sub class dengan menggunakan keyword super.

Syntax nya adalah:

1. `super()` : Memanggil constructor tanpa argumen dari superclass
2. `super(parameter)` : Memanggil constructor dari superclass yang memiliki argumen-argumen yang sesuai.

Keyword Super

Constructor digunakan untuk membentuk instance dari suatu class. Namun, tidak seperti properti dan method, constructor ini tidak diwariskan pada subclass.

Sehingga constructor dari subclass ini hanya dapat dipanggil melalui constructor dari sub class dengan menggunakan keyword super.

Syntax nya adalah:

1. `super()` : Memanggil constructor tanpa argumen dari superclass
2. `super(parameter)` : Memanggil constructor dari superclass yang memiliki argumen-argumen yang sesuai.

Keyword Super

```
package Bag_8_Inheritance;

public class Contoh_Super {
    private String warna;
    // Constructor
    public Contoh_Super(String warna_warna) {
        this.warna = warna_warna;
    }
    // Method getter warna
    public String getWarna() {
        return warna;
    }
    // Method setter warna
    public void setWarna(String warna_warni) {
        this.warna = warna_warni;
    }
}
```

Keyword Super

```
package Bag_8_Inheritance;

public class Contoh_Super_Extends extends Contoh_Super {

    String warna_lain;

    public Contoh_Super_Extends(String warna_warna) {
        super("pelangi");
        this.warna_lain = warna_warna;
    }

    public static void main(String array[]) {
        Contoh_Super_Extends ini_variabel = new Contoh_Super_Extends("biru muda");
        System.out.println("INI WARNA SUPER = "+ini_variabel.getWarna());
        System.out.println("INI WARNA THIS = "+ini_variabel.warna_lain);
    }
}
```

Final

Final Method adalah sebuah method dengan visibility apapun yang tidak bisa diakses oleh method lain class yang mewarisi. Sedangkan Class Final yaitu suatu class yang tidak bisa diwarisi oleh class lain.

Penggunaannya keduanya menggunakan keyword final di depan nama class maupun nama method. Jadi, Final Method dan Final Class merupakan class dan method terlarang di timpa oleh class lain, yang artinya sudah tidak bisa dilakukan proses terhadap class maupun method tersebut.

- ■ ■ ■ Keyword Final biasanya untuk class yang tidak bisa diganggu gugat baik dari pengolahan data maupun penurunan data.

Final

```
1 public final class Mobil {  
2  
3 }
```

```
1 public class Honda extends Mobil {  
2  
3     public static void main (String args[]){  
4  
5     }  
6  
7 }
```

```
1 java.lang.VerifyError: Cannot inherit from final class
```


Polimorfisme

Prinsip di mana class dapat memiliki banyak **bentuk** method yang berbeda-beda meskipun namanya sama. **Bentuk** di sini dapat kita artikan: isinya berbeda, parameternya berbeda, dan tipe datanya pun berbeda.

Polimorfisme pada Java ada dua macam:

Static Polymorphism (Polimorfisme statis) dan Dynamic Polymorphism (Polimorfisme dinamis).

Beda dari keduanya terletak pada cara membuat polimorfismenya. Polimorfisme statis menggunakan method **overloading** sedangkan polimorfisme dinamis menggunakan method **overriding**.

Polimorfisme

```
package Bag_8_Inheritance;

public class Polimorfisme {

    public String polimorfisme_static() {
        // OVERLOADING
        return "Nilai return static";
    }

    public String polimorfisme_static(String data_dinamis) {
        // OVERRIDING
        return data_dinamis;
    }
}
```

9. Abstract Interface

- Abstract Class
- Interface
- Mengapa Kita Memakai Interface ?
- Interface vs. Class Abstract
- Interface vs. Class
- Membuat Interface
- Hubungan dari Interface ke Class
- Pewarisan Antar Interface



Abstract

Class abstrak adalah class yang masih dalam bentuk abstrak. Karena bentuknya masih abstrak, dia tidak bisa dibuat langsung menjadi objek.

Sebuah class agar dapat disebut class abstrak setidaknya memiliki satu atau lebih method abstrak.

Method abstrak adalah method yang tidak memiliki implementasi atau tidak ada bentuk konkritnya.

Abstract

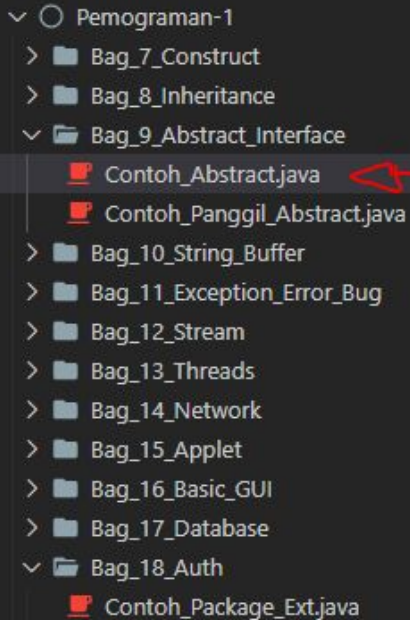
Contohnya adalah Kendaraan

Kendaraan belum diketahui cara lajunya, bahan bakar, jumlah roda dan lainnya karena Kendaraan masih bersifat Abstract.

Beda jika didefinisikan menjadi Mobil

Keabstrakan tentang cara laju, bahan bakar dan lainnya sudah diketahui

Abstract - Membuat



```

v ○ Pemograman-1
  > Bag_7_Construct
  > Bag_8_Inheritance
  v Bag_9_Abstract_Interface
    Contoh_Abstract.java
    Contoh_Panggil_Abstract.java
  > Bag_10_String_Buffer
  > Bag_11_Exception_Error_Bug
  > Bag_12_Stream
  > Bag_13_Threads
  > Bag_14_Network
  > Bag_15_Applet
  > Bag_16_Basic_GUI
  > Bag_17_Database
  v Bag_18_Auth
    Contoh_Package_Ext.java

```

```

package Bag_9_Abstract_Interface;

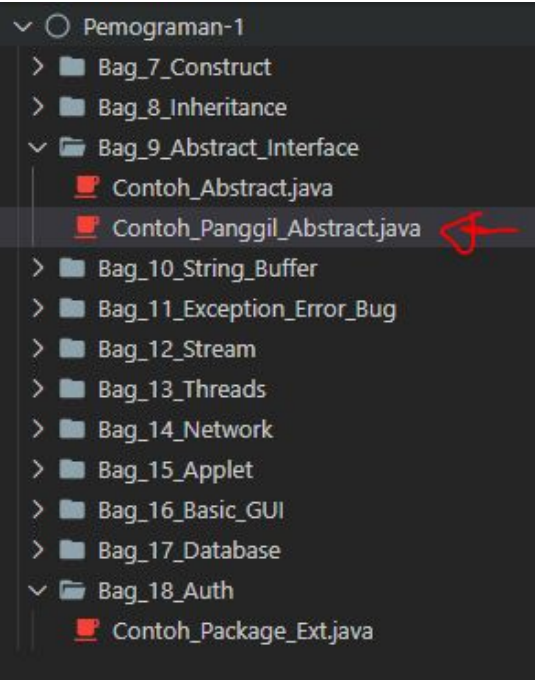
public abstract class Contoh_Abstract {

    abstract public void abstrak();

    public void bukanAbstrak() {
        System.out.println("Bukan Abstrak");
    }
}

```

Abstract - Memanggil



```
package Bag_9_Abstract_Interface;

public class Contoh_Panggil_Abstract extends
Contoh_Abstract {

    @Override
    public void abstrak() {

        System.out.println("ini abstract dari file Contoh_Abstract");

    }

    public static void main(String[] argument) {

        Contoh_Panggil_Abstract ini_variabel = new Contoh_Panggil_Abstract ();

        ini_variabel.abstrak();

    }

}
```

Interface

Apa yang dimaksud dengan interface pada java? :

Interface merupakan **kumpulan method-method abstrak**. Sebuah kelas yang mengimplementasikan interface, mewarisi method – method abstrak dari interface tersebut.

Interface adalah sebuah **tipe referensi** pada Java. Interface secara struktur serupa dengan class. Isi dari interface adalah method abstract, artinya method hanya dideklarasikan tidak ditulis secara utuh.

Interface adalah sebuah file yang hanya berisi **method kosong**. Method kosong pada interface ditujukan untuk menjadi **behaviour atau sifat wajib dari class** yang mengimplementasikannya.

Interface

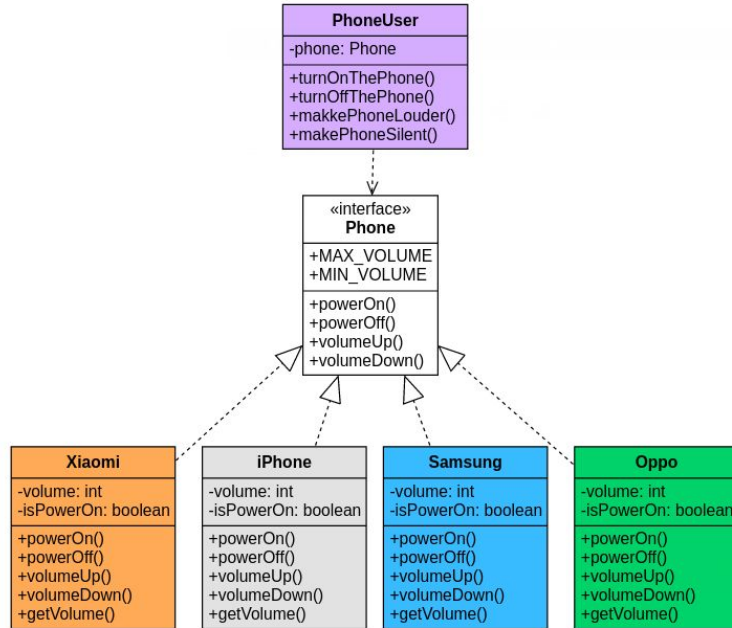
Dari penjelasan-penjelasan tadi, dapat disimpulkan bahwa Interface merupakan kumpulan method yang hanya memuat deklarasi dan struktur method, tanpa detail implementasi. Detail implementasi dari method pada interface ditulis pada kelas yang mengimplementasikan interface tersebut.

Interface

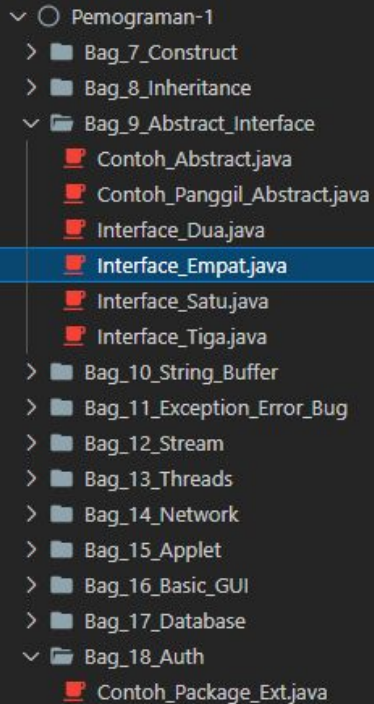
Singkatnya, Interface adalah kemungkinan-kemungkinan, bisa dikatakan method-method yang bersifat interface masih bersifat kemungkinan.

Contohnya ketika kita masuk ke mini market, kemungkinan kita membeli A, atau barang B dan lainnya, **kemungkinan membeli barang adalah adalah method abstract.**

Interface



Interface



```

v ○ Pemograman-1
  > Bag_7_Construct
  > Bag_8_Inheritance
  v Bag_9_Abstract_Interface
    Contoh_Abstract.java
    Contoh_Panggil_Abstract.java
    Interface_Dua.java
    Interface_Empat.java
    Interface_Satu.java
    Interface_Tiga.java
  > Bag_10_String_Buffer
  > Bag_11_Exception_Error_Bug
  > Bag_12_Stream
  > Bag_13_Threads
  > Bag_14_Network
  > Bag_15_Applet
  > Bag_16_Basic_GUI
  > Bag_17_Database
  v Bag_18_Auth
    Contoh_Package_Ext.java

```

Interface – Manfaat

- Interface memiliki beberapa manfaat yang dapat memudahkan kita dalam melakukan Pemrograman Berorientasi Objek, diantaranya :
- Interface memungkinkan sebuah subclass diturunkan dari beberapa superclass (multiple inheritance) Interface lebih mempermudah sistem analyst dalam membuat konsep aplikasi
- Pemrograman java menggunakan interface dapat lebih efisien karena adanya multiple inheritance

Interface – Aturan

Interface adalah sebuah tipe referensi pada java. Secara struktur hampir sama dengan class. Ada beberapa aturan dalam penulisan interface :

- Modifier method hanya boleh public, abstract atau default
- Pada class yang meng implement, modifier method – method hanya boleh public
- Jumlah parameter method interface harus sama dengan class yang mengimplementasi
- Tidak boleh ada method **concrete** di interface, apabila ada maka program akan error.
- Tidak boleh ada constructor
- Interface tidak bisa di instanisasi, tapi bisa di instanisasi melalui class yang mengimplementasi

Concrete

Concrete class adalah kelas yang tidak berisi abstract method atau berisi class yang nyata. Concrete class digunakan ketika sebuah **tidak memiliki abstract method**. Concrete class dapat menurunkan implementasi method pada superclass. Concrete class dapat membentuk atau menginstansiasi objek.

Interface vs Inheritance

Perbedaan Interface Dengan Inheritance

Inheritance (Pewarisan) digunakan ketika parent class memiliki atribut dan method lalu semuanya diwariskan ke subclass. Sedangkan Interface digunakan ketika parent tidak memiliki apa-apa, hanya method yang tidak memiliki tubuh dan harus diimplementasikan pada subclassnya. Interface juga memungkinkan terjadinya multiple inheritance, yaitu sebuah subclass yang diturunkan dari lebih dari satu superclass.

Inheritance Interface

Pewarisan Interface adalah Sebuah kelas yang dapat mewarisi interface dengan menggunakan kata kunci implements, dimana kelas tersebut dapat mewarisi beberapa interface. Interface bukan bagian dari hirarki class.

Inheritance Interface

Pewarisan Interface adalah Sebuah kelas yang dapat mewarisi interface dengan menggunakan kata kunci implements, dimana kelas tersebut dapat mewarisi beberapa interface. Interface bukan bagian dari hirarki class.

```
public interface NamaInterface extends InterfaceSatu, InterfaceDua
```

Interface vs Abstract

Perbedaan Interface Dengan Abstract

Metode interface tidak punya tubuh, sebuah interface hanya dapat mendefinisikan konstanta dan interface tidak langsung mewariskan hubungan dengan class istimewa lainnya, mereka didefinisikan secara independen.

10. String Buffer

Metode untuk Class String dan StringBuffer
Metode untuk Class String
Constructor String
Metode-metode pada String
Metode untuk Class StringBuffer
Constructor StringBuffer
Metode-Metode Pada StringBuffer
Metode untuk Class Math
Metode untuk Class Data Type Wrapper
Metode untuk Class Process dan Runtime
Class Process
Class Runtime
Membuka Registry Editor
Metode untuk Class System

String Buffer

StringBuffer merupakan sebuah class library yang digunakan untuk mengolah data dan juga nilai karakter, class tersebut terdapat di dalam package (java.lang).

StringBuffer secara konsep, memiliki kemiripan dengan tipe data String, yaitu untuk **menampilkan karakter**. Perbedaannya terletak pada beberapa fungsi serta objek dari class StringBuffer.

Pada StringBuffer dapat menambahkan ataupun menghapus karakter di dalamnya.

String Buffer

```
package Bag_10_String_Buffer;

public class Contoh_StringBuffer {
    public static void main(String[] args) {
        // Mendefinisikan Objek Beserta Nilainya
        StringBuffer data = new StringBuffer("Belajar");
        // Menambahkan Beberapa Nilai/Karakter String
        data.append(" Pemrograman");
        data.append(" Java");
        data.append(" Dasar");
        // Menampilkan Output
        System.out.println(data);
    }
}
```

Construct String Buffer

1. `String()`
2. `String(String Value)`
3. `String(char value[])`
4. `String(char value[], int offset, int count)`
5. `String(byte ascii[], int hibyte, int offset, int count)`
6. `String(byte ascii[], int hibyte)`
7. `String(StringBuffer buffer)`

Construct String Buffer

```
package Bag_10_String_Buffer;
public class StringBuffer_Construct {
    public static void main(String args[]) {
        // String()
        String st1 = new String();
        st1 = "Constructor St1";
        // String(String Value)
        String st2 = new String("Constructor St2");
        char ch[] = { 'C', 'o', 'n', 's', 't', 'r', 'u', 'c', 't', 'o', 'r', ' ', '3' };
        // String(char value[])
        String st3 = new String(ch);
        // String(char value[], int offset, int count)
        String st4 = new String(ch, 0, ch.length);
        byte bytes[] = { 'C', 'o', 'n', 's', 't', 'r', 'u', 'c', 't', 'o', 'r', ' ', '4' };
        // String(byte ascii[], int hibyte, int offset, int count)
        String st5 = new String(bytes, 0, 0, bytes.length);
        // String(byte ascii[], int hibyte)
        String st6 = new String(bytes, 0);
        // String(StringBuffer buffer)
        String st7 = new String(new StringBuffer("Constructor St7"));
    }
}
```


Metode-metode String

`int length()`

`char charAt(int index)`

`boolean startsWith(String prefix)`

`boolean startsWith(String prefix, int toffset)`

`boolean endsWith(String suffix)`

`int indexOf(int ch)`

`int indexOf(int ch, int fromIndex)`

`int indexOf(String str)`

`int indexOf(String str, int fromIndex)`

`int lastIndexOf(int ch)`

`int lastIndexOf(int ch, int fromIndex)`

`int lastIndexOf(String str)`

`int lastIndexOf(String str, int fromIndex)`

`String substring(int beginIndex)`

`String substring(int beginIndex, int endIndex)`

`boolean equals(Object anObject)`

`boolean equalsIgnoreCase(String
anotherString)`

`int compareTo(String anotherString)`

`String concat(String str)`

`String replace(char oldChar, char newChar)`

`String trim()`

`String toLowerCase()`

`String toUpperCase()`

`static String valueOf(Object obj)`

`static String valueOf(char data[])`

`static String valueOf(char data[], int offset,
int count)`

`static String valueOf(boolean b)`

`static String valueOf(char c)`

`static String valueOf(int i)`

`static String valueOf(long l)`

`static String valueOf(float f)`

`static String valueOf(double d)`

`static Double Double.valueOf(String
st).doubleValue()`

Metode-metode String Buffer

int length()

int capacity()

synchronized void setLength(int newLength)

synchronized char charAt(int index)

synchronized void setCharAt(int index, char ch)

synchronized StringBuffer append(Object obj)

synchronized StringBuffer append(String str)

synchronized StringBuffer append(char c)

synchronized StringBuffer append(char str[])

synchronized StringBuffer append(char str[], int offset, int len)

StringBuffer append(boolean b)

StringBuffer append(int i)

StringBuffer append(long l)

StringBuffer append(float f)

StringBuffer append(double d)

synchronized StringBuffer insert(int offset, Object obj)

synchronized StringBuffer insert(int offset, String str)

synchronized StringBuffer insert(int offset, char c)

synchronized StringBuffer insert(int offset, char str[])

StringBuffer insert(int offset, boolean b)

StringBuffer insert(int offset, int i)

StringBuffer insert(int offset, long l)

StringBuffer insert(int offset, float f)

StringBuffer insert(int offset, double d)

String toString()

Math

Kelas Math berisi sekumpulan method dan konstanta matematika.

```
public static double abs(double a)
```

```
public static float abs(float a)
```

```
public static long abs(long a)
```

```
public static int abs(int a)
```

```
public static double random()
```

```
public static double max(double a, double b)
```

```
public static double min(double a, double b)
```

```
public static double ceil(double a)
```

```
public static double floor(double a)
```

```
public static double exp(double a)
```

```
public static double log(double a)
```

```
public static double pow(double a, double b)
```

```
public static long round(double a)
```

```
public static double sqrt(double a)
```

```
public static double sin(double a)
```

```
public static double cos(double a)
```

```
public static double tan(double a)
```

```
public static double asin(double a)
```

```
public static double acos(double a)
```

```
public static double atan(double a)
```

```
public static double toDegrees(double angrad)
```

```
public static double toRadians(double angdeg)
```

Wrapper

Di dalam bahasa pemrograman Java, Wrapper Class (Kelas Pembungkus) adalah suatu mekanisme yang digunakan untuk meng convert atau mengubah suatu nilai yang didefinisikan, dari tipe data primitif menjadi sebuah nilai dengan tipe data referensi (Objek). Selain itu, Wrapper Class mendukung method dari tipe data primitif, contohnya seperti Boolean, Character, Integer, Long, Float dan lainnya.

Tipe Data

Pada bahasa pemrograman Java, terdapat 2 jenis tipe data, yaitu primitif dan referensi (objek).

Tipe Data Primitif adalah tipe data yang tidak memiliki method, hanya memiliki data saja. Tipe data ini bukanlah sebuah object.

Tipe Data Referensi adalah tipe data yang digunakan untuk memegang referensi dari sebuah object (instance dari class).

Tipe Data

Dan berikut ini merupakan tabel yang menjelaskan macam-macam tipe data primitif dan referensi pada Java:

Tipe Data Primitif

char
byte
short
int
long
boolean
float
double

Tipe Data Referensi

Character
Byte
Short
Integer
Long
Boolean
Float
Double

Runtime

Kelas Runtime adalah kelas pada java.lang.* .

Jadi intinya dirinya **tak perlu lagi di import karena default akan di import/terpanggil**. Selain itu, Runtime adalah kelas bukan Interface atau Abstract class.

Tetapi tidak demikian untuk kelas Runtime. Dia menggunakan private Constructor yang berarti kita **tidak dapat menciptakan instan kelas tersebut di luar kelas**. Pemanggilan untuk penciptaan instan kelas

Runtime

Ada method pada Runtime yang mengembalikan OutputStream dan InputStream yang digunakan untuk berinteraksi. Jangan digunakan, karena itu sudah usang karena tidak aman.

Jadi cara yang benarnya untuk berinteraksi adalah menggunakan Input dan Output Stream dari Process.

Ada method **exec()** pada Runtime yang mengembalikan Objek Process.

Runtime

```
package Bag_10_String_Buffer;  
  
public class Contoh_Runtime_Exec {  
  
    public static void main(String[] zzzzzzzzzzz) {  
  
        Runtime ini_runtimenya = Runtime.getRuntime();  
  
        try {  
            ini_runtimenya.exec("notepad 'ini_akan_jadi_nama_file.txt'");  
        } catch (Exception e) {  
            System.out.println("INI ERRORNYA == " + e);  
        }  
    }  
}
```

Runtime

Fungsi dari Class Runtime ada juga untuk memanggil garbage Collection.

Kita bisa memanggil **run.gc()** apabila kita ingin memaksakan agar garbage collection bekerja.

Jadi tukang sampah dilapang sepakbola itu kita suruh kerja walaupun dia sebenarnya udah ada yang ngatur kapan kerja atau tidak.

Garbage Collection

Pada Java, Garbage Collection adalah mekanisme Java untuk menghapus suatu objek dari memori tanpa perlu dideklarasikan secara eksplisit dalam program.

Berfungsi untuk meningkatkan Manajemen Memori yaitu kita dapat menghemat penggunaan memori, java akan menghapus sebuah objek yang tidak diperlukan atau tidak di referensikan lagi, ruang memori pada objek tersebut akan dimanfaatkan untuk keperluan lain sehingga tidak ada pemborosan memori.

Garbage Collection merupakan salah satu mekanisme dari fitur JVM (Java Virtual Machine). Ini adalah salah satu kelebihan dari bahasa pemrograman Java.

11. Exception Error Bug

- Bug dan Exception
- Error dan Exception Classes
- Menangkap Exception
- Try Catch
- Keyword Finally
- Melempar Exception
- Keyword Throw
- Keyword Throws
- Kategori Exception
- Exception Classes dan Hierarki
- Checked dan Unchecked Exceptions
- User Defined Exceptions
- Assertions
- User Defined Exceptions
- Mengaktifkan dan Menonaktifkan Exceptions
- Sintaks Assertions



3 JENIS ERROR PADA PEMROGRAMAN KAMU



Syntax Error

- Salah keyword
- Kurang “;”
- Kurang petik “”
- Dan lain lain



Runtime Error

- Salah input
- Terjadi pembagian 0
- Dan lain lain



Logic Error

- Program Bug
- Perlu pemahaman betul ke syntax code

Exception

BUG = ERROR

Exception

Exception adalah **event (kejadian) yang mengacaukan jalannya suatu program**. *Worst case scenario* ketika suatu program mengalami exception adalah termination. Termination (penutupan) program adalah hal yang harus dihindari. Untuk itu kita harus menangani exception yang terjadi di program, atau yang biasa disebut sebagai handle exception.

Kode yang baik adalah yang terhindar dari segala bentuk kejadian yang menyebabkan efek buruk kepada program. Oleh karena itu mari kita kenali dulu berbagai macam exception yang ada di Java.

Exception

Sama halnya dengan Exception, Class Error juga merupakan bagian terkecil (subclass) dari `java.lang.Throwable`.

Pada umumnya Class Error dan Class Exception merupakan class yang berbeda.

Perbedaan diantara keduanya adalah **Class Error** ialah masalah yang timbul tetapi programmer tidak dapat menangkap dan menanganinya dikarenakan tidak ada alasan yang kuat. Berbeda dengan **Class Exception** yang merupakan kesalahan kecil yang timbul dan akan ditangkap oleh developer untuk segera ditangani.

Exception

Checked Exception, adalah exception yang **terjadi saat compile time**.

Di sini programmer harus menambahkan kode untuk meng-handle exception kategori ini. Jika tidak di-handle maka kode yang dituliskan akan error dan tidak akan bisa dikompilasi.

Contohnya adalah exception `java.io.FileNotFoundException`.

Exception

Unchecked Exception, adalah exception yang **terjadi saat execution time**.

Error ini terjadi dalam lingkup internal dari aplikasi kita, biasanya terjadi karena salah penulisan kode atau penggunaan salah terhadap satu API.

Contohnya adalah NullPointerException.

Exception

Error, adalah exception yang **diluar kendali user atau programmer**.

Error ini terjadi di lingkup eksternal dari aplikasi kita. Ketika exception ini terjadi maka tidak ada yang bisa kita lakukan untuk mengatasinya.

Contohnya ketika perangkat kerasnya rusak saat kita ingin membaca data.

Try { catch }

Kode yang rawan dengan exception kita masukkan ke dalam block try-catch.

Kode yang kita masukkan ke dalam block try-catch biasa disebut sebagai protected code. Kodenya seperti ini.

```
try{  
    // Protected code  
} catch (Exception e){  
    // Catch block  
}
```

Finally

Block finally adalah blok yang ditambahkan di akhir block try-catch. Finally akan selalu dijalankan setelah try-catch baik terjadi exception atau tidak.

Finally bermanfaat ketika kita ingin melakukan cleanup code.

Cleanup code di sini maksudnya adalah **de-alokasi sumber daya**, Artinya semua sumber daya yang dibuka atau digunakan pada blok try seperti koneksi jaringan, database, berkas, stream dan lainnya **akan ditutup** dengan menjalankan instruksi yang ditulis pada blok finally.

Finally

```
public class Main {  
    public static void main(String[] args) {  
        int[] a = new int[5];  
        try {  
            System.out.println("Akses elemen ke 5 : " + a[5]);  
        } catch (ArrayIndexOutOfBoundsException e) {  
            System.out.println("Exception thrown : " + e);  
        } finally {  
            a[4] = 10;  
            System.out.println("Nilai elemen terakhir: " + a[4]);  
        }  
    }  
}
```

Throw(s)

Keyword throws digunakan pada sebuah method yang kemungkinan berpotensi suatu exception, sehingga perlu ditangkap exception atau errornya.

```
(modifier) nama method() throws exception_list1, exception_List2, .....
```

Throw(s)

```
package Bag_11_Exception_Error_Bug;

public class Contoh_Throws {

    static void lemparError() throws ClassNotFoundException {
        System.out.println("Ada Yang Error Ni!");
        throw new ClassNotFoundException("Saya Tangkap");
    }

    public static void main(String[] args) {
        try {
            Contoh_Throws.lemparError();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```




12. Stream

I/O Stream
Byte Stream
Character Stream
Operasi File



I / O

Stream adalah proses **membaca** data dari suatu sumber (input) atau **mengirimkan** data ke suatu tujuan (output).

Stream juga dikategorikan berdasarkan apakah mereka digunakan untuk **membaca** atau **menulis** stream.

I / O

System.in (default: keyboard)

System.out (default: layar)

System.err (default: console)

Byte Stream

Stream byte digunakan untuk **memberikan atau menyimpan informasi data dalam bentuk byte.**

Misalnya untuk menulis dan membaca file biner.

Sedangkan stream karakter digunakan untuk melakukan proses I/O yang melibatkan data-data dalam bentuk karakter. Misalnya pada saat melakukan proses baca atau tulis ke file teks.

Byte Stream

Bytes stream digunakan ketika membaca input dan output **8 bytes**.

Terdapat banyak class yang terkait dengan stream byte ini, namun yang paling banyak digunakan adalah:

FileInputStream

FileOutputStream

Byte Stream - Console

```
package Bag_12_Stream;

import java.io.IOException;

public class Byte_Stream_Console {
    public static void main(String[] args) throws IOException {
        byte[] data = new byte[10];
        System.out.println("Ketik 10 buah karakter:");
        System.in.read(data);
        System.out.println("Karakter yang anda ketik adalah:");
        for (int i = 0; i < data.length; i++) {
            System.out.println((char) data[i]);
        }
    }
}
```

Byte Stream - File

```
package Bag_12_Stream;

import java.io.FileInputStream;
import java.io.FileOutputStream;
public class Byte_Stream_File {
    public static void main(String[] args) {
        try {
            FileInputStream fileInputStream = new FileInputStream ("C:/Users/File.txt" );
            FileOutputStream fileOutputStream = new FileOutputStream ("C:/Users/Copy.txt" );
            int i;
            while ((i = fileInputStream.read()) != -1) {
                fileOutputStream.write(i);
                System.out.print ((char) i);
            }
            fileInputStream.close();
            fileOutputStream.close();
        } catch (Exception e) {
            System.out.println("Error Exception " + e);
        }
    }
}
```

Character Stream

Stream karakter (character) digunakan untuk proses input dan output **16 bytes**.

Terdapat banyak class yang berhubungan dengan stream dari karakter ini, tapi terdapat 2 class yang paling sering digunakan yaitu:

FileReader

FileWriter

Character Stream - Console

```
package Bag_12_Stream;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class Char_Stream_Console {
    public static void main(String[] args) throws IOException {
        char data;
        String str = "";
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        System.out.println("Ketik sejumlah karakter, akhiri dengan  \\Q\\");
        data = (char) br.read();
        while (data != 'Q') {
            // ELIMINASI APAPUN SETEALAH Q DAN AKAN TERUS IDLE SAMPAI DIINPUT Q
            str += data;
            data = (char) br.read();
        }
        System.out.println("Karakter yang anda ketik : " + str);
    }
}
```

Character Stream - File

```
package Bag_12_Stream;

import java.io.BufferedReader;
import java.io.FileReader;

public class Char_Stream_File {
    public static void main(String[] args) {
        try {
            BufferedReader bufferedReader = new BufferedReader(
                new FileReader("C:/Users/File.txt"));

            String baris;
            while ((baris = bufferedReader.readLine()) != null) {
                System.out.println(baris);
            }
        } catch (Exception e) {
            System.out.println("Error Exception " + e);
        }
    }
}
```

13. Threads

- Definisi dan Dasar-Dasar Thread
- Definisi Thread
- State dari Thread
- Prioritas Thread
- Class Thread
- Constructor
- Constants
- Methods
- Extend vs Implement
- Sinkronisasi
- Mengunci Obyek
- Komunikasi Antar Thread (InterThread)
- Kemampuan Concurrency
- Interface Executor
- Interface Callable

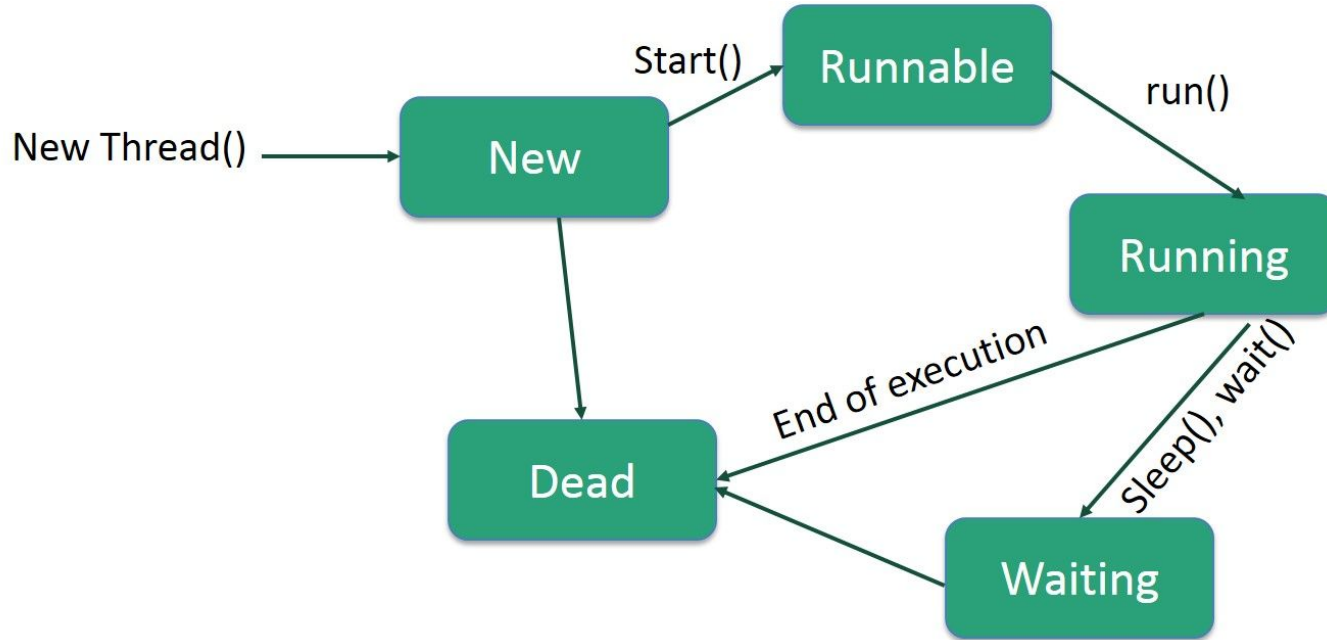
Threads

Thread adalah rangkaian eksekusi dari sebuah aplikasi java dan setiap program java minimal memiliki satu buah thread.

Untuk lebih mudahnya, bayangkanlah thread sebagai sebuah proses yang akan dieksekusi di dalam sebuah program tertentu. Penggunaan sistem operasi modern saat ini telah mendukung kemampuan untuk menjalankan beberapa program. Misalnya, pada saat Anda mengetik sebuah dokumen di komputer Anda dengan menggunakan text editor, dalam waktu yang bersamaan Anda juga dapat mendengarkan musik, dan surfing lewat internet di PC Anda.

MULTITASKING

Threads

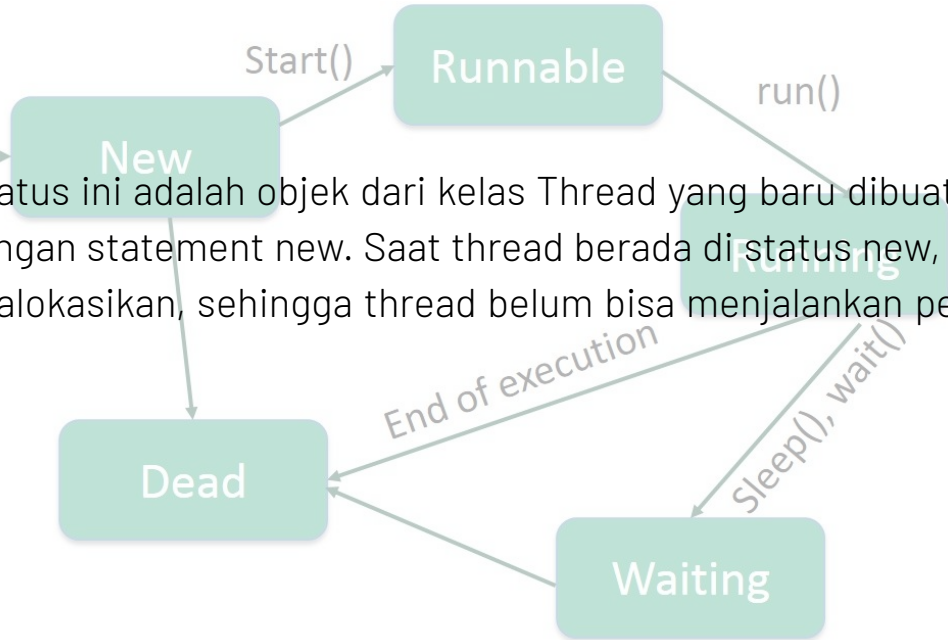


Threads - Status

New

New Thread() → New

Thread yang berada di status ini adalah objek dari kelas Thread yang baru dibuat, yaitu saat instansiasi objek dengan statement new. Saat thread berada di status new, belum ada sumber daya yang dialokasikan, sehingga thread belum bisa menjalankan perintah apapun.

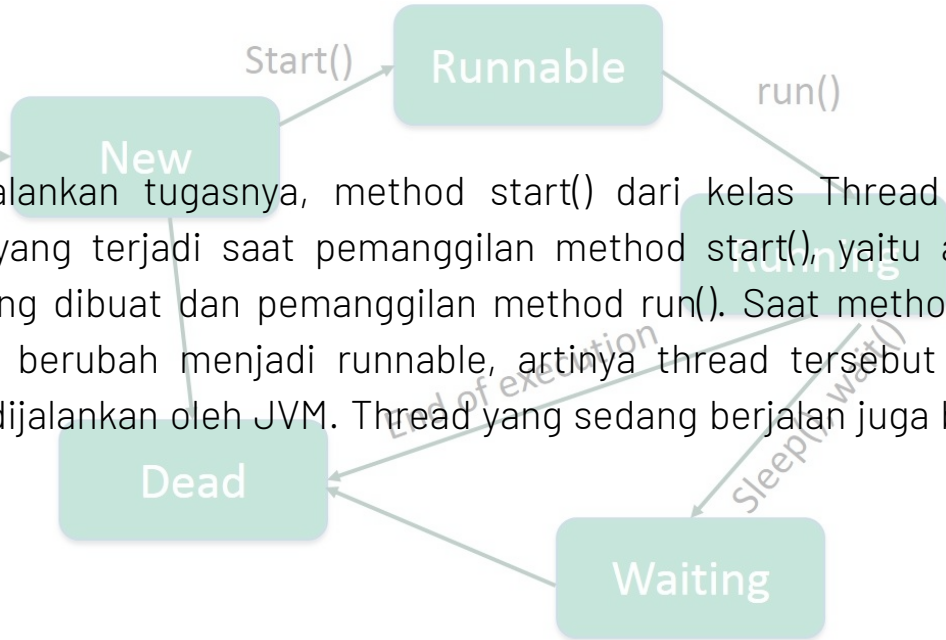


Threads - Status

Runnable

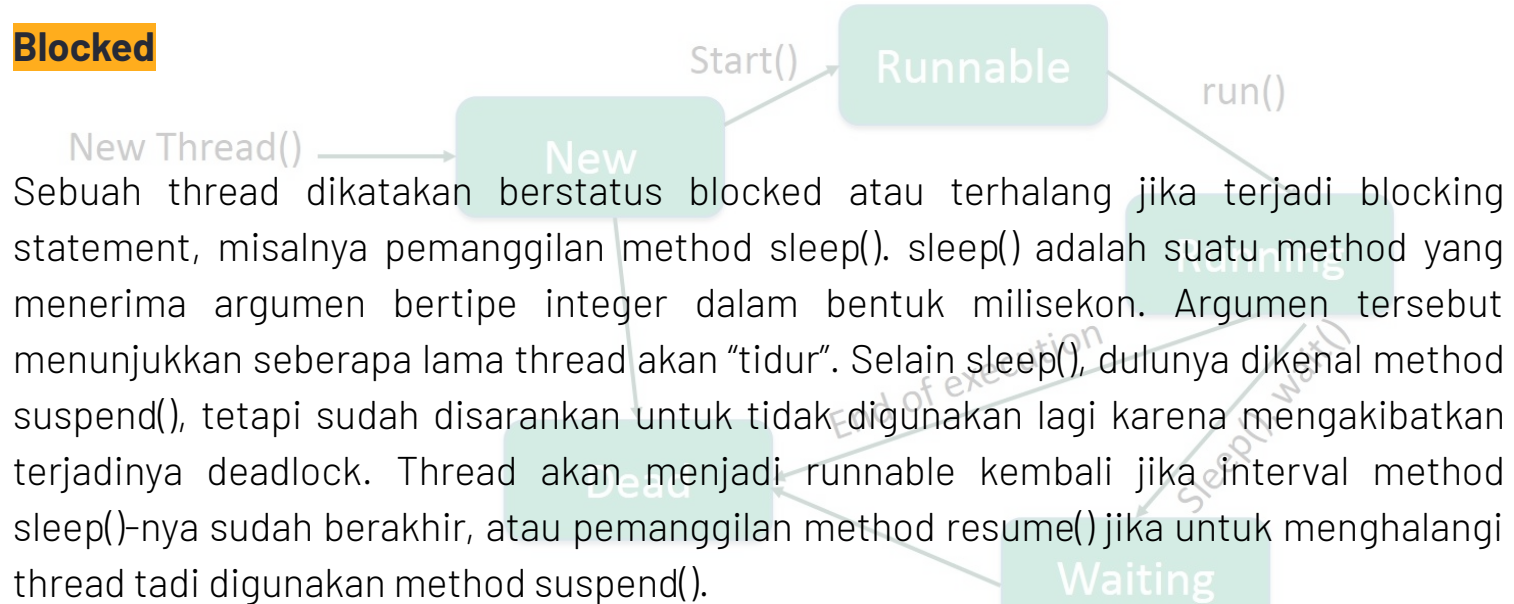
New Thread() → New

Agar thread bisa menjalankan tugasnya, method start() dari kelas Thread harus dipanggil. Ada dua hal yang terjadi saat pemanggilan method start(), yaitu alokasi memori untuk thread yang dibuat dan pemanggilan method run(). Saat method run() dipanggil, status thread berubah menjadi runnable, artinya thread tersebut sudah memenuhi syarat untuk dijalankan oleh JVM. Thread yang sedang berjalan juga berada di status runnable.



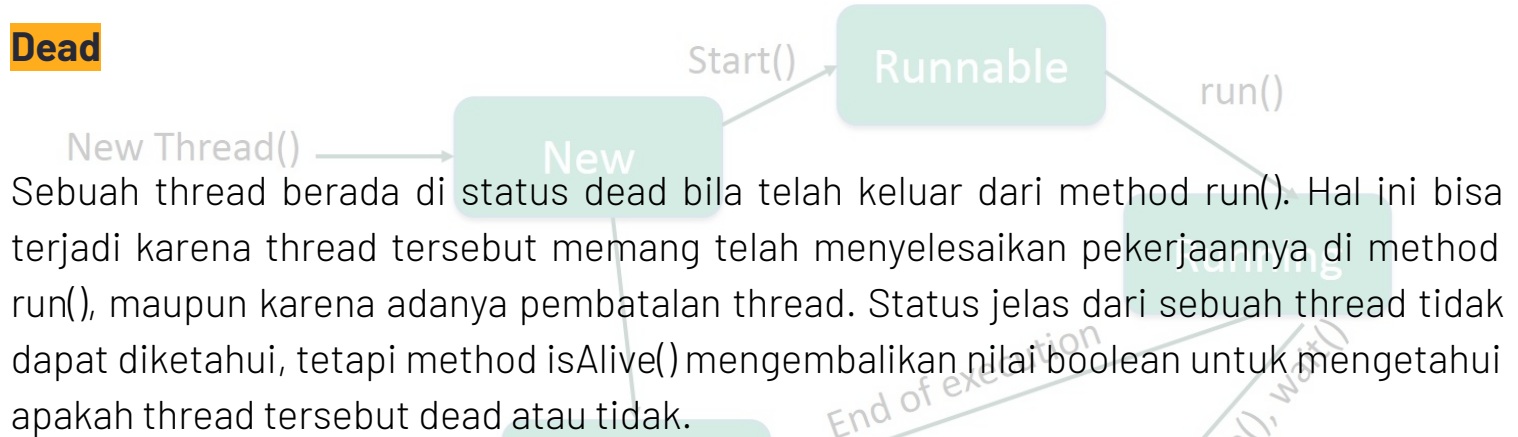
Threads - Status

Blocked

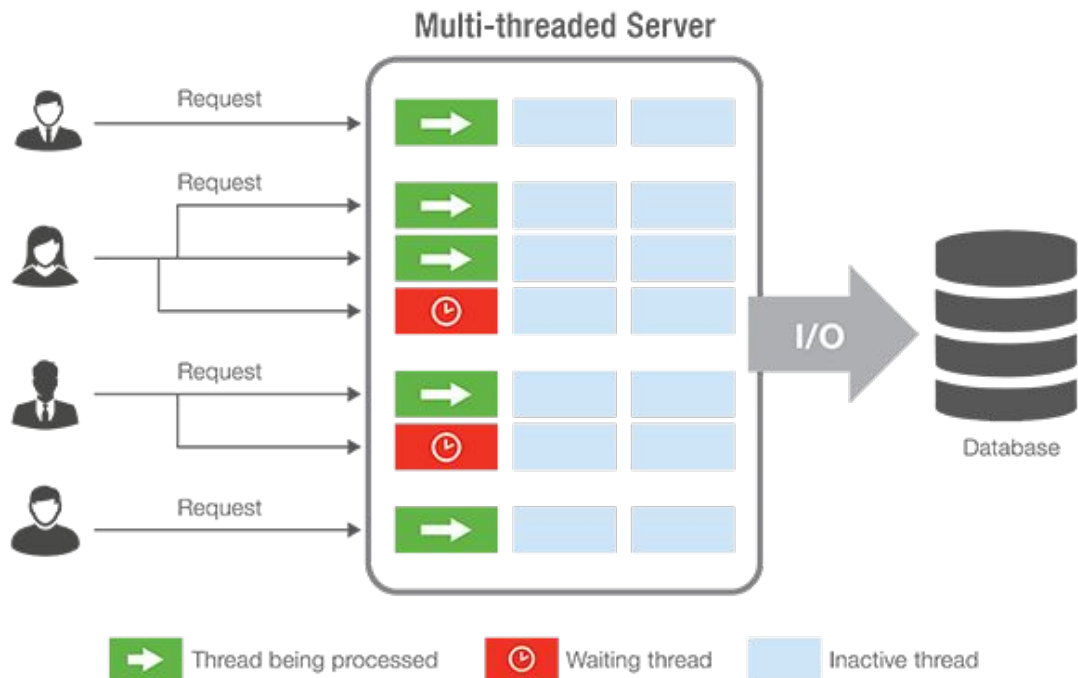


Threads - Status

Dead



Multi Threads



Runnable

Runnable

Merupakan unit abstrak, yaitu kelas yang mengimplementasikan interface ini hanya cukup mengimplementasikan fungsi `run()`.

Dalam mengimplementasi fungsi `run()`, kita akan mendefinisikan instruksi yang membangun sebuah thread.

Runnable

Implements Interface Runnable, Cara ini merupakan cara yang paling sederhana dalam membuat thread.

Runnable merupakan unit abstrak, yaitu kelas yang mengimplementasikan interface ini hanya cukup mengimplementasikan fungsi `run()`.

Dalam mengimplementasi fungsi `run()`, kita akan mendefinisikan instruksi yang membangun sebuah thread.

Runnable

```
package Bag_13_Threads;

public class Contoh_Runnable implements Runnable {
    Thread ini_thread;

    Contoh_Runnable(String ini_parameter) {
        ini_thread = new Thread(this, ini_parameter);
        ini_thread.start();
    }

    public void run() {
        String variabelnya = ini_thread.getName();
        for (int i = 0; i < 100; i++) {
            System.out.println(variabelnya);
        }
    }

    public static void main(String args[]) {
        Contoh_Runnable pnt1 = new Contoh_Runnable("A");
        Contoh_Runnable pnt2 = new Contoh_Runnable("B");
        System.out.println("Threads Berjalan");
        try {
            pnt1.ini_thread.join();
            pnt2.ini_thread.join();
        } catch (InterruptedException ie) {
            System.out.println("Kalau Ada Error " + ie);
        }
        System.out.println("Done"); // dicetak terakhir
    }
}
```

14. Network

Konsep Dasar Jaringan
IP Address
Protokol
Port
Sockets
Java Networking Package
Class ServerSocket dan Socket
Class MulticastSocket dan DatagramPacket



Jaringan

Pemrograman jaringan berarti untuk menulis dijalankan pada beberapa perangkat Program(komputer), perangkat ini terhubung melalui jaringan.

IP

Internet Protocol Address adalah label numerik yang ditetapkan untuk setiap perangkat yang terhubung ke jaringan komputer yang menggunakan Protokol Internet untuk komunikasi. Alamat IP memiliki dua fungsi utama: host atau identifikasi antarmuka jaringan dan pengalamatan lokasi.

Protokol

Protokol mengatur peraturan dan standar yang menetapkan jenis komunikasi internet yang khusus.

Port

Dalam protokol jaringan TCP/IP, sebuah port adalah mekanisme yang memungkinkan sebuah komputer untuk mendukung beberapa sesi koneksi dengan komputer lainnya dan program di dalam jaringan.

Socket

Socket menggunakan TCP menyediakan mekanisme komunikasi antara dua komputer. Program klien menciptakan socket dan mencoba untuk terhubung ke server socket.

Datagram socket (menggunakan UDP).

Stream socket (menggunakan TCP).

Socket

TCP (Transmission Control Protocol)

UDP (User Datagram Protocol)

adalah protokol jaringan yang mentransfer data Anda melalui internet dari perangkat ke server web. UDP merupakan salah satu tipe protokol yang mempunyai karakteristik tidak berbasis koneksi. Sebaliknya, TCP menggunakan koneksi.

HTTP

```
package Bag_14_Network;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;

public class Server_Socket {
    private static ServerSocket server = null;

    public static void main(String[] args) {
        Socket client;
        try {
            server = new ServerSocket(1234);
        } catch (IOException ie) {
            System.out.println("Socket Gagal");
            System.exit(1);
        }

        String InSt = "";
        while (true) {
            try {
                client = server.accept();
                OutputStream clientOut = client.getOutputStream();
                PrintWriter pw = new PrintWriter(clientOut, true);
                InputStream clientIn = client.getInputStream();
                BufferedReader br = new BufferedReader(new InputStreamReader(clientIn));
                String St = br.readLine();
                pw.println("Dari Server : " + St + InSt);
                System.out.println("Ini Log Dari Browser : " + St);
            } catch (IOException ie) {
                System.out.println("Error Pada " + ie);
            }
        }
    }
}
```

Multicast

MulticastSocket sangat berguna untuk aplikasi yang mengimplementasikan komunikasi secara berkelompok.

Alamat IP untuk kelompok multicast berkisar antara 224.0.0.0 hingga 239.255.255.255.

Meskipun begitu, alamat 224.0.0.0 telah dipesan dan seharusnya tidak digunakan. class ini memiliki tiga konstruktor tetapi kita akan membahas satu dari ketiga konstruktor ini

HTTP Multicast

```
package Bag_14_Network;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;

public class Server_Socket {
    private static ServerSocket server = null;

    public static void main(String[] args) {
        Socket client;
        try {
            server = new ServerSocket(1234);
        } catch (IOException ie) {
            System.out.println("Socket Gagal");
            System.exit(1);
        }

        String InSt = "";
        while (true) {
            try {
                client = server.accept();
                OutputStream clientOut = client.getOutputStream();
                PrintWriter pw = new PrintWriter(clientOut, true);
                InputStream clientIn = client.getInputStream();
                BufferedReader br = new BufferedReader(new InputStreamReader(clientIn));
                String St = br.readLine();
                pw.println("Dari Server : " + St + InSt);
                System.out.println("Ini Log Dari Browser : " + St);
            } catch (IOException ie) {
                System.out.println("Error Pada " + ie);
            }
        }
    }
}
```

15. Applet

Membentuk Applet
Metode-Metode Applet
Siklus Applet (The Applet Life Cycle)
Metode Paint
Menggambar di Applet
Metode ShowStatus

Applet

Applet adalah salah satu jenis program yang bisa dihasilkan oleh bahasa pemrograman Java selain program aplikasi desktop dan server. **Applet Java harus dijalankan menggunakan web browser**, misalnya di Microsoft Internet Explore, Mozilla FireFox, Google Chrome atau browser yang lain.

Applet

Applet Java dapat disertakan di dalam dokumen **HyperText Markup Language (HTML)** bila ingin dijalankan di web browser.

Dokumen HTML menggunakan tag untuk memberi instruksi ke web browser dan web browser akan menerjemahkan dan memutuskan bagaimana menampilkan atau memperlakukan konten dari dokumen HTML.

Applet

Anda mungkin mencoba menggunakan appletviewer di Java 11 atau yang lebih baru. Ini tidak akan berhasil. Dukungan Applet sudah tidak digunakan lagi di Java 9, dan telah dihapus seluruhnya di Java 11.

https://bugs.java.com/bugdatabase/view_bug.do?bug_id=JDK-8200146

ORACLE® Java Bug Database

Oracle Technology Network > Java > Java SE > Community > Bug Database

JDK-8200146 : Remove the appletviewer launcher

Applet

During the past five years, most browser vendors have withdrawn support for plugins such as Flash, Silverlight, and Java in their products. Supporting Java Applets in browsers was only possible as long as browser vendors were committed to supporting standards based plugins. By late 2015, many browser vendors had either removed or announced timelines for the removal of standards-based plugin support, while some introduced proprietary browser-specific extension APIs. Consequently:

- Existing Applet support in Java SE 8 will continue through March 2019, after which it may be removed at any time.
- Oracle announced in January 2016 that Applets would be deprecated in Java SE 9, and removed from Java SE 11 (18.9).

16. Basic G U I

Abstract Windowing Toolkit (AWT) vs. Swing
Komponen GUI
JFrame
JPanel
JLabel
JTextField
JButton
JComboBox
JMenu
JTable
GUI Event Handling
Delegation Event Model
Registrasi Listeners
Class-Class Event
Event Listeners
Method ActionListener
Method MouseListener
Method-Method MouseMotionListener
Method-Method WindowListener

JComponent

1. **JFrame** merupakan komponen dasar dalam membuat aplikasi GUI, dimana frame berfungsi sebagai container atau wadah untuk menampung komponen GUI lainnya.
2. **JPanel** digunakan untuk membuat sebuah panel yang berfungsi sebagai kontainer untuk menampung berbagai macam komponen, seperti label, button, textfield, table dan lain-lain.
3. **JLabel** digunakan untuk menampilkan teks yang berfungsi untuk memberikan keterangan atau menjelaskan komponen GUI lainnya agar mudah dimengerti oleh user.
4. **TextField** adalah komponen GUI yang biasa digunakan untuk memasukkan data dengan mengetik dari keyboard.

JComponent

- 5. **JButton** digunakan untuk membuat sebuah tombol yang berfungsi untuk menerima input dari user berupa klik menggunakan mouse atau tombol enter dari keyboard.
- 6. **JComboBox** merupakan sebuah kelas pada swing yang berguna untuk membuat sebuah ComboBox.
- 7. **JMenu** digunakan untuk membuat menu pull-down yang dapat digunakan untuk memanggil suatu form.
- 8. **JColorChooser** Turunan Jcomponent. Memungkinkan pengguna untuk memilih warna yang diinginkan.
- 9. **JTable** merupakan kelas yang digunakan untuk membuat tabel.

JComponent

10. JCheckBox Item yang dapat dipilih atau tidak oleh pengguna. Berhubungan dengan class checkbox dalam package AWT.

11. JFileChooser Mengizinkan pengguna untuk memilih sebuah file. Berhubungan dengan class jfilechooser dalam package AWT.

12. JOptionPane Turunan Jcomponent. Disediakan untuk mempermudah menampilkan popup kotak dialog.

13. JDialog Turunan dan Berhubungan dengan class dialog dalam package AWT. Biasanya digunakan untuk menginformasikan sesuatu kepada pengguna atau prompt pengguna untuk input.

17. Database

JDBC
Mengimpor Package java.sql
Memanggil Driver JDBC
Membuat Koneksi
Membuat Obyek Statemen
Melakukan Perintah SQL
Menutup Koneksi

JDBC

JDBC adalah Application Programming Interface (API) yang dirancang untuk mengakses database universal berdasarkan SQL

JDBC

Interaksi dengan database server dengan menggunakan JDBC adalah sebagai berikut :

a. Mengimpor package java.sql

b. Memanggil Driver JDBC

c. Membangun Koneksi

d. Membuat Statement

e. Melakukan Query

f. Memproses Hasil

g. Menutup Koneksi

h. Penanganan Error

JDBC

DriverManager: adalah sebuah class yang mengelola driver;

Driver: adalah interface yang menangani komunikasi dengan database.

Connection: adalah interface yang menyediakan method untuk menghubungkan database;

Statement: adalah interface untuk mengeksekusi query;

ResultSet: adalah interface untuk menampung data hasil query.

JDBC

MySQL

Driver: `com.mysql.jdbc.Driver`,

URL: `jdbc:mysql://hostname/databaseName`;

ORACLE

Driver: `oracle.jdbc.driver.OracleDriver`,

URL `jdbc:oracle:thin:@hostname:portNumber:databaseName`;

DB2

Driver: `COM.ibm.db2.jdbc.net.DB2Driver`,

URL: `jdbc:db2:hostname:portNumber/databaseName`

MYSQL

MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL atau DBMS yang multi alur, multi pengguna, dengan sekitar 6 juta instalasi di seluruh dunia.

MYSQL Pada XAMPP

Download

<https://www.apachefriends.org/xampp-files/7.4.16/xampp-windows-x64-7.4.16-0-VC15-installer.exe>



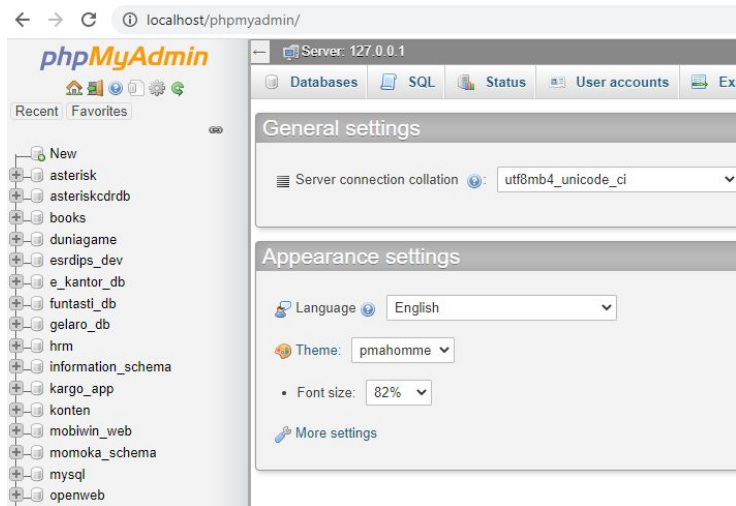
XAMPP for Windows 7.3.27, 7.4.16 & 8.0.3

Version		Checksum			Size
7.3.27 / PHP 7.3.27	What's Included?	md5	sha1	Download (64 bit)	154 Mb
7.4.16 / PHP 7.4.16	What's Included?	md5	sha1	Download (64 bit)	156 Mb
8.0.3 / PHP 8.0.3	What's Included?	md5	sha1	Download (64 bit)	156 Mb

PhpMyAdmin XAMPP

Setelah menginstall XAMPP

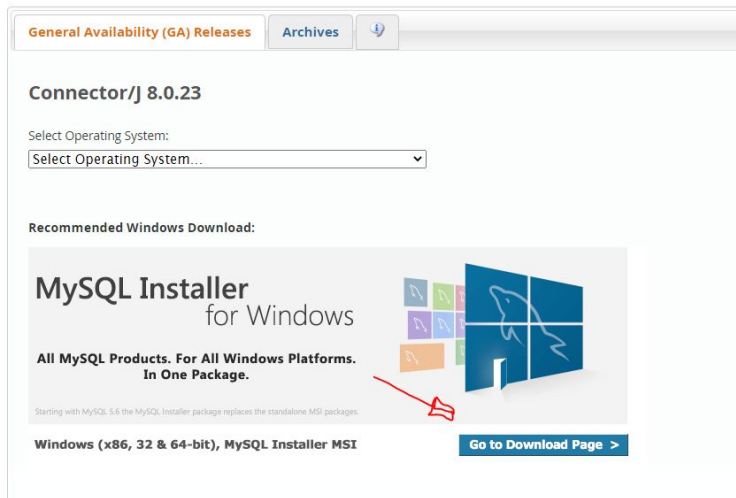
<http://localhost/phpmyadmin/>



MYSQL JDBC Driver

Download


<https://dev.mysql.com/downloads/connector/j/>



MYSQL JDBC Driver

Download

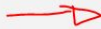
<https://dev.mysql.com/downloads/windows/installer/8.0.html>


[General Availability \(GA\) Releases](#) [Archives](#) 

MySQL Installer 8.0.23

Select Operating System:
Microsoft Windows

[Looking for previous GA versions?](#)

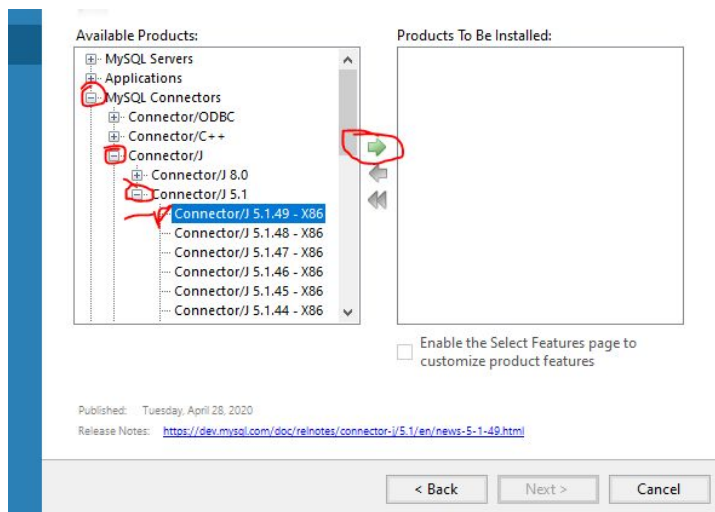
Windows (x86, 32-bit), MSI Installer (mysql-installer-web-community-8.0.23.0.msi)	8.0.23		2.4M	Download
			MD5: a3af6d91f93e046452b38a1e2589534c Signature	
Windows (x86, 32-bit), MSI Installer (mysql-installer-community-8.0.23.0.msi)	8.0.23		422.4M	Download
			MD5: 8de85ced955631901829a1a363cddf50 Signature	

 We suggest that you use the [MD5 checksums](#) and [GnuPG signatures](#) to verify the integrity of the packages you download.



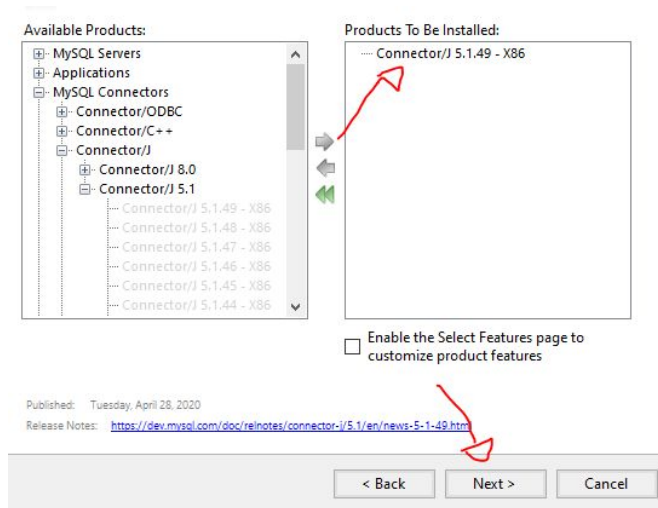
MYSQL JDBC Driver

PILIH CONNECTOR



MYSQL JDBC Driver

PILIH CONNECTOR



MYSQL JDBC Driver


EXECUTE

Akan mendownload dari

<https://cdn.mysql.com/Downloads/Connector-J/mysql-connector-java-gpl-5.1.49.msi>

Download

The following products will be downloaded.

Product	Status	Progress	Notes
 Connector/J 5.1.49	Ready to download		

Click [Execute] to install the following packages.

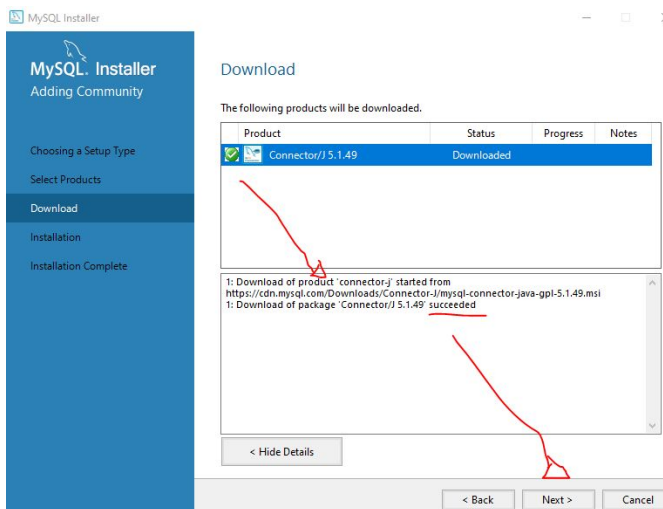
< Back

Execute

Cancel

MYSQL JDBC Driver

INSTALL




MYSQL JDBC Driver

EXECUTE

Installation

The following products will be installed.

Product	Status	Progress	Notes
 Connector/J 5.1.49	<u>Ready to Install</u>		

Click [Execute] to install the following packages.

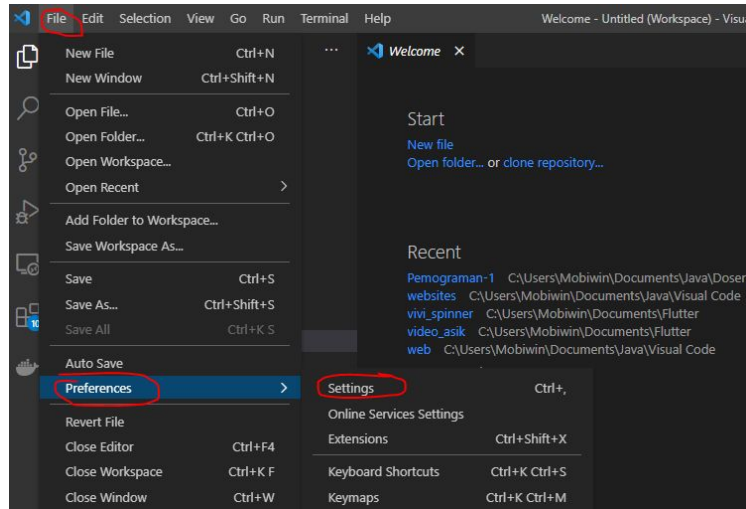
< Back

Execute

Cancel

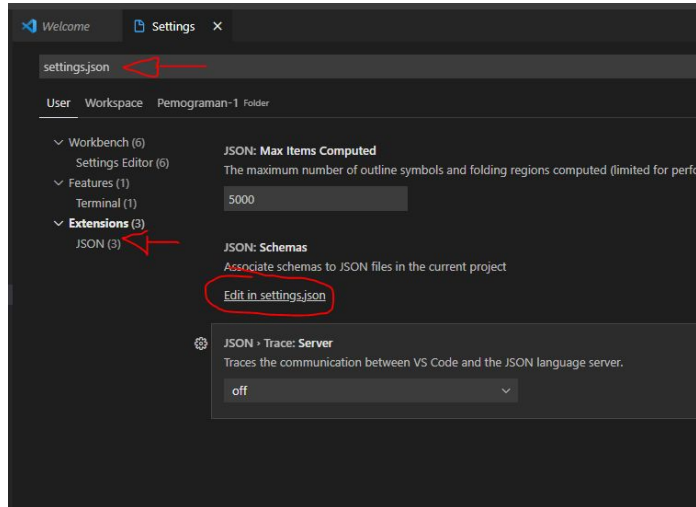
MYSQL JDBC Driver

CONFIG



MYSQL JDBC Driver

CONFIG



MYSQL JDBC Driver

CONFIG

```
Welcome Settings settings.json
C:\Users\Mobiwin> AppData > Roaming > Code > User > {} settings.json > [ ] json.schemas
9
10  /**.classpath: true,
11  /**.project: true,
12  /**.settings: true,
13  /**.factorypath: true
14  },
15  "dart.warmWhenEditingFilesOutsideWorkspace": false,
16  "dart.openDevTools": "flutter",
17  "java.jdt.ls.vmargs": "-XX:+UseParallelGC -XX:GCTimeRatio=4 -XX:AdaptiveSizePolicyWeight=90 -Dsun.zip
18  "java.home": "C:\\Users\\Mobiwin\\AppData\\Local\\Programs\\AdoptOpenJDK\\jdk-11.0.9.101-hotspot",
19  "java.help.firstView": "gettingStarted",
20  "extensions.autoUpdate": false,
21  "java.project.importOnFirstTimeStartup": "automatic",
22  "java.refactor.renameFromFileExplorer": "autoApply",
23  "java.project.referencedLibraries": [
24    "lib/**/*.*.jar",
25    "C:\\Users\\Mobiwin\\Documents\\Java\\Dosen\\Pemograman-1\\Bag_17_Database\\mysql-connector-java-8
26  ],
27  "workbench.iconTheme": "material-icon-theme",
28  "[html]": {
29    "editor.defaultFormatter": "HookyQR.beautify"
30  },
31  "[json]": {
32    "editor.quickSuggestions": {
33      "strings": true
34    },
35    "editor.suggest.InsertMode": "replace"
36  },
37  "json.schemas": [
38  ],
39  ]
```

MYSQL JDBC Driver

CONFIG

```
"java.project.referencedLibraries": [  
    "lib/**/*.jar",  
    "C:\\\\Users\\\\Mobiwin\\\\Documents\\\\Java\\\\Dosen\\\\Pemograman-1\\\\Bag_17_Database\\\\mysql-  
connector-java-8.0.23.jar"  
],
```

SQL Package

```
import java.sql.*;
```

JDBC x MYSQL x JAVA

CODE

KONEKSI DATABASE

Read & Insert Database Mysql

CODE

QUERY



Terima Kasih!

Kalian bisa ketemu saya di:

- **Instagram** @juripebrianto
 - **Github** @aquarink
 - **Email** dosen02662@unpam.ac.id
- 

You can also split your content

White

Is the color of milk and fresh snow, the color produced by the combination of all the colors of the visible spectrum.

Black

Is the color of ebony and of outer space. It has been the symbolic color of elegance, solemnity and authority.

Let's review some concepts

Yellow

Is the color of gold, butter and ripe lemons. In the spectrum of visible light, yellow is found between green and orange.

Blue

Is the colour of the clear sky and the deep sea. It is located between violet and green on the optical spectrum.

Red

Is the color of blood, and because of this it has historically been associated with sacrifice, danger and courage.

Yellow

Is the color of gold, butter and ripe lemons. In the spectrum of visible light, yellow is found between green and orange.

Blue

Is the colour of the clear sky and the deep sea. It is located between violet and green on the optical spectrum.

Red

Is the color of blood, and because of this it has historically been associated with sacrifice, danger and courage.