MAKALAH KONTRUKTOR

PADA JAVA



Oleh :

DIKI CANDRA

NIM 2022903430010

D4 TEKNOLOGI REKAYASA KOMPUTER JARINGAN

POLITEKNIK NEGERI LHOKSEMAWE

2022/2023

DAFTAR ISI

[BAB I 3](#_Toc132385779)

[PENDAHULUAN 3](#_Toc132385780)

[1.1 Latar Belakang 3](#_Toc132385781)

[1.2 Tujuan 4](#_Toc132385782)

[BAB II 5](#_Toc132385783)

[KONSTRUKTOR DALAM BAHASA JAVA 5](#_Toc132385784)

[2.1 Pengertian Konstruktor 5](#_Toc132385785)

[2.2 Jenis-jenis Konstruktor 6](#_Toc132385786)

[2.3 Perbedaan antara Konstruktor dan Metode 19](#_Toc132385787)

[2.4 Penggunaan Konstruktor pada Objek 20](#_Toc132385788)

[BAB III 22](#_Toc132385789)

[PEMBAHASAN 22](#_Toc132385790)

[3.1 Kelebihan Penggunaan Konstruktor dalam Bahasa Java 22](#_Toc132385791)

[3.2 Kendala Penggunaan Konstruktor dalam Bahasa Java 23](#_Toc132385792)

[3.3 Solusi atas Kendala Penggunaan Konstruktor dalam Bahasa Java 25](#_Toc132385793)

[BAB IV 27](#_Toc132385794)

[KESIMPULAN 27](#_Toc132385795)

[4.1 Ringkasan hasil Peneliatian 27](#_Toc132385796)

[4.2 Saran 28](#_Toc132385797)

[DAFTAR PUSTAKA 29](#_Toc132385798)

# BAB I

# PENDAHULUAN

## 1.1 Latar Belakang

Pemrograman komputer adalah salah satu bidang yang sangat populer saat ini. Bahasa pemrograman yang banyak digunakan di dunia saat ini adalah Java. Bahasa ini sangat populer karena mudah dipelajari dan memiliki banyak fitur yang berguna dalam pengembangan aplikasi.

Salah satu fitur penting dalam bahasa pemrograman Java adalah konstruktor. Konstruktor adalah suatu metode khusus yang digunakan untuk menginisialisasi objek pada saat pembuatan objek. Dengan menggunakan konstruktor, kita dapat memberikan nilai awal untuk atribut objek, sehingga objek tersebut dapat digunakan secara langsung.

Konstruktor sangat berguna dalam pembuatan objek dalam bahasa Java. Karena objek adalah entitas yang sangat penting dalam bahasa pemrograman Java, penggunaan konstruktor dapat membantu mempermudah pengembangan aplikasi dalam bahasa Java.

Dalam makalah ini, akan dibahas mengenai konstruktor dalam bahasa Java. Pembahasan akan dimulai dengan menjelaskan pengertian konstruktor, jenis-jenis konstruktor, dan perbedaan antara konstruktor dan metode. Selanjutnya, akan diberikan contoh implementasi konstruktor dalam bahasa Java, seperti contoh implementasi konstruktor default, konstruktor parameter, dan konstruktor kopikan. Selain itu, juga akan dibahas kelebihan dan kendala penggunaan konstruktor dalam bahasa Java, serta solusi atas kendala penggunaan konstruktor tersebut.

Dengan membahas konstruktor dalam bahasa Java, diharapkan pembaca dapat memahami dan mengerti bagaimana penggunaan konstruktor dapat membantu mempermudah pengembangan aplikasi dalam bahasa Java. Selain itu, pembaca juga dapat mengetahui kelebihan dan kendala penggunaan konstruktor dalam bahasa Java, sehingga dapat memilih menggunakan atau tidak menggunakan konstruktor dalam pengembangan aplikasi dalam bahasa Java.

## 1.2 Tujuan

Tujuan dari makalah ini adalah untuk menjelaskan konstruktor dalam bahasa Java secara mendalam, sehingga pembaca dapat memahami konsep tersebut dan dapat mengimplementasikannya dalam pengembangan aplikasi dalam bahasa Java. Dalam makalah ini, akan dijelaskan tentang pengertian konstruktor, jenis-jenis konstruktor, dan perbedaan antara konstruktor dan metode. Selanjutnya, akan diberikan contoh implementasi konstruktor dalam bahasa Java, seperti konstruktor default, konstruktor parameter, dan konstruktor kopikan. Selain itu, akan dibahas kelebihan dan kendala penggunaan konstruktor dalam bahasa Java, serta solusi atas kendala penggunaan konstruktor tersebut.

Manfaat dari makalah ini adalah sebagai berikut:

1. Meningkatkan pemahaman pembaca tentang konstruktor dalam bahasa Java: Makalah ini akan membantu pembaca untuk memahami konsep konstruktor secara lebih mendalam. Dalam makalah ini, akan dijelaskan secara detail tentang pengertian konstruktor, jenis-jenis konstruktor, dan perbedaan antara konstruktor dan metode.
2. Membantu pembaca dalam mengimplementasikan konstruktor dalam pengembangan aplikasi: Dalam makalah ini, akan diberikan contoh implementasi konstruktor dalam bahasa Java, seperti konstruktor default, konstruktor parameter, dan konstruktor kopikan. Hal ini akan membantu pembaca untuk dapat mengimplementasikan konstruktor dalam pengembangan aplikasi yang sedang dibuat.
3. Mengetahui kelebihan dan kendala penggunaan konstruktor dalam bahasa Java: Dalam makalah ini, akan dibahas tentang kelebihan dan kendala penggunaan konstruktor dalam bahasa Java. Hal ini akan membantu pembaca untuk mengetahui kapan harus menggunakan atau tidak menggunakan konstruktor dalam pengembangan aplikasi.
4. Mengetahui solusi atas kendala penggunaan konstruktor dalam bahasa Java: Dalam makalah ini, akan diberikan solusi atas kendala penggunaan konstruktor dalam bahasa Java. Hal ini akan membantu pembaca untuk mengatasi kendala tersebut ketika mengimplementasikan konstruktor dalam pengembangan aplikasi.

Dengan membaca makalah ini, pembaca diharapkan dapat memahami konstruktor dalam bahasa Java dengan lebih baik, sehingga dapat mengimplementasikan konstruktor dalam pengembangan aplikasi secara efektif dan efisien. Selain itu, pembaca juga diharapkan dapat mengetahui kelebihan dan kendala penggunaan konstruktor dalam bahasa Java, serta solusi atas kendala penggunaan konstruktor tersebut.

1.3 Rumusan Masalah

Berikut adalah rumusan masalah yang akan dibahas dalam makalah ini berdasarkan daftar isi di atas:

1. Apa pengertian dari konstruktor dalam bahasa Java?
2. Apa saja jenis-jenis konstruktor dalam bahasa Java?
3. Apa perbedaan antara konstruktor dan metode dalam bahasa Java?
4. Bagaimana cara mengimplementasikan konstruktor dalam bahasa Java, seperti konstruktor default, konstruktor parameter, dan konstruktor kopikan?
5. Apa kelebihan dan kendala penggunaan konstruktor dalam bahasa Java?
6. Apa solusi atas kendala penggunaan konstruktor dalam bahasa Java?

Pertanyaan-pertanyaan tersebut akan dijawab secara detail dalam makalah ini. Pertanyaan pertama akan menjelaskan tentang pengertian konstruktor, sedangkan pertanyaan kedua akan membahas jenis-jenis konstruktor yang ada dalam bahasa Java. Pertanyaan ketiga akan membahas perbedaan antara konstruktor dan metode dalam bahasa Java. Sedangkan pertanyaan keempat akan menjelaskan cara mengimplementasikan konstruktor dalam bahasa Java, seperti konstruktor default, konstruktor parameter, dan konstruktor kopikan. Selanjutnya, pertanyaan kelima akan membahas tentang kelebihan dan kendala penggunaan konstruktor dalam bahasa Java. Terakhir, pertanyaan keenam akan membahas solusi atas kendala penggunaan konstruktor dalam bahasa Java.

Dengan menjawab pertanyaan-pertanyaan tersebut, pembaca diharapkan dapat memahami konstruktor dalam bahasa Java secara lebih mendalam dan dapat mengimplementasikannya dalam pengembangan aplikasi. Selain itu, pembaca juga dapat mengetahui kelebihan dan kendala penggunaan konstruktor dalam bahasa Java, serta solusi atas kendala penggunaan konstruktor tersebut.

# BAB II

# KONSTRUKTOR DALAM BAHASA JAVA

## 2.1 Pengertian Konstruktor

Konstruktor adalah salah satu fitur yang penting dalam pemrograman berorientasi objek (OOP) yang digunakan untuk membuat objek atau instance dari suatu kelas. Konstruktor digunakan untuk menginisialisasi nilai awal dari objek yang dibuat dengan nilai-nilai default atau nilai-nilai yang diberikan saat objek tersebut dibuat. Konstruktor biasanya digunakan untuk mengalokasikan memori untuk objek yang dibuat dan memberikan nilai awal pada variabel yang terdapat dalam objek tersebut.

Dalam bahasa pemrograman Java, konstruktor adalah suatu metode yang memiliki nama yang sama dengan nama kelasnya dan tidak memiliki tipe kembalian. Konstruktor biasanya digunakan untuk menginisialisasi nilai-nilai dari variabel-variabel instance yang ada di dalam kelas. Ketika sebuah objek dibuat, konstruktor akan secara otomatis dipanggil dan nilai-nilai awal dari variabel-variabel instance tersebut akan diinisialisasi.

Ada beberapa jenis konstruktor yang dapat digunakan dalam bahasa Java, yaitu:

1. **Konstruktor Default:** Konstruktor default adalah konstruktor yang tidak memiliki parameter. Konstruktor ini akan dipanggil secara otomatis ketika objek dibuat dan tidak diberikan nilai apapun.
2. **Konstruktor Parameter:** Konstruktor parameter adalah konstruktor yang memiliki parameter. Konstruktor ini digunakan untuk menginisialisasi nilai dari variabel instance dengan nilai yang diberikan melalui parameter.
3. **Konstruktor Kopikan:** Konstruktor kopikan adalah konstruktor yang digunakan untuk menyalin nilai-nilai dari objek yang sudah ada ke objek baru.

Perbedaan antara konstruktor dan metode adalah bahwa konstruktor digunakan untuk menginisialisasi nilai awal dari objek yang dibuat, sedangkan metode digunakan untuk melakukan tindakan atau perhitungan pada objek yang sudah ada. Selain itu, konstruktor dipanggil secara otomatis ketika objek dibuat, sedangkan metode harus dipanggil secara eksplisit.

Secara umum, konstruktor adalah salah satu fitur penting dalam pemrograman berorientasi objek yang digunakan untuk menginisialisasi nilai awal dari objek yang dibuat. Konstruktor dapat digunakan untuk mengalokasikan memori untuk objek yang dibuat dan memberikan nilai awal pada variabel-variabel yang terdapat dalam objek tersebut. Dalam bahasa Java, konstruktor memiliki tiga jenis, yaitu konstruktor default, konstruktor parameter, dan konstruktor kopikan. Dengan menggunakan konstruktor, kita dapat membuat objek dengan nilai-nilai awal yang sesuai dengan kebutuhan aplikasi yang sedang dibuat.

## 2.2 Jenis-jenis Konstruktor

**A. Konstruktor default**

Konstruktor default di Java dibuat oleh kompiler itu sendiri ketika pemrogram tidak membuat [konstruktor](https://www.scaler.com/topics/java/constructor-in-java/" \o "" \t "_blank) apa pun . Tujuan dari konstruktor default adalah menginisialisasi atribut objek dengan nilai defaultnya.

**Apa itu konstruktor Default di Java?**

Konstruktor default dibuat hanya jika kita tidak mendeklarasikan konstruktor apa pun dalam kode kita. Kemudian, secara default, kompiler secara otomatis membuat konstruktor default.

Karena kita mengetahui peran konstruktor di Java, pertanyaannya adalah, apa yang akan terjadi jika konstruktor tidak tersedia selama pembuatan objek?

Tapi pertama-tama, kita perlu melihat kasus penting di mana kita mencoba mengakses variabel apapun tanpa inisialisasi. Di sini, dalam contoh ini, kami hanya akan mendeklarasikan variabel tetapi tidak akan menginisialisasinya dengan nilai defaultnya.

*//Program to show declaration of variables*

*//without initialization*

import java.io.\*;

import java.util.\*;

class Main {

public static void main (String[] args) {

*//Declaration of variables*

*//without initialization*

int id;

String first\_name;

String last\_name;

*//Trying to access the variables and print without assigning the default values*

System.out.println("Hi " + first\_name + " " + last\_name + ", your id is " + id);

}

}

**Kesalahan kompilasi:**

./Main.java:15: error: variable first\_name might not have been initialized

System.out.println"Hi " + first\_name + " " + last\_name + ", your id is " + id);

^

./Main.java:15: error: variable last\_name might not have been initialized

System.out.println"Hi " + first\_name + " " + last\_name + ", your id is " + id);

^

./Main.java:15: error: variable id might not have been initialized

System.out.println"Hi " + first\_name + " " + last\_name + ", your id is " + id);

^

3 errors

Dan memang, inilah yang mungkin Anda harapkan. Kode di atas telah menimbulkan banyak kesalahan. Mengapa? Karena kami tidak repot menginisialisasi variabel dengan nilai defaultnya. Juga, variabel-variabel ini dideklarasikan dalam metode statis , jadi ini adalah variabel statis . Penting untuk menyediakan variabel statis dengan nilai defaultnya.

Sekarang, kami akan memberikan nilai default untuk mereka.

*//Program to show declaration of variables*

import java.io.\*;

import java.util.\*;

public class Main {

public static void main (String[] args) {

*//Declaration & initialization of variables*

int id = 2022903430010;

String first\_name = "Diki";

String last\_name = "Candra";

*//Trying to access the variables and print their default values*

System.out.println("Hi " + first\_name + " " + last\_name + ", your id is " + id);

}

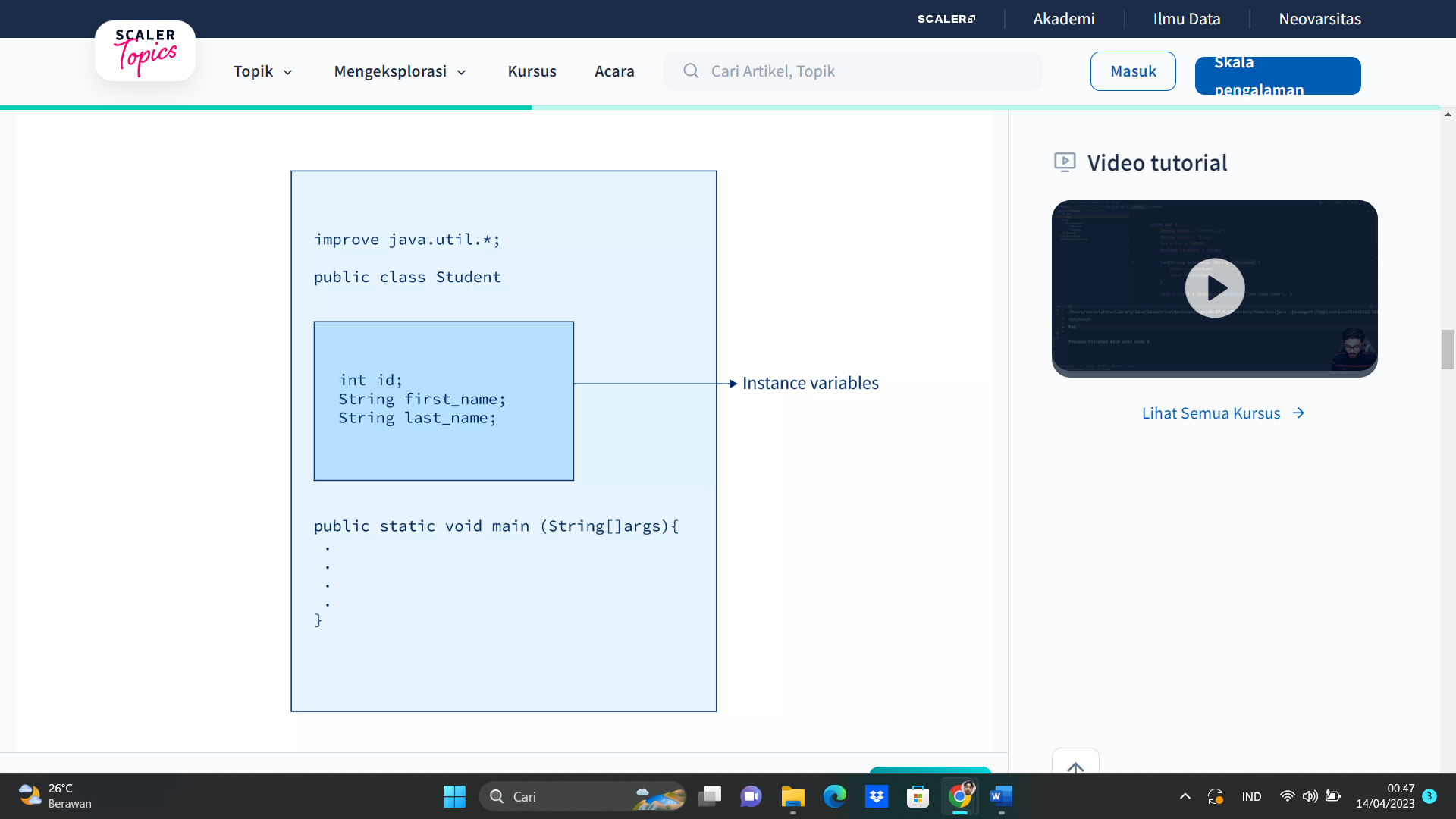
}

**Keluaran:**

Hi Diki Candra, your id is 2022903430010

Sekarang, kode berfungsi dengan baik. Jadi, contoh di atas menyoroti kesalahan umum yang mungkin kita, sebagai pemrogram, lakukan saat membuat objek juga.

**Variabel instan** adalah yang dideklarasikan sebagai properti kelas tetapi di luar konstruktor, metode, atau blok kelas.



Jadi, menginisialisasi variabel instan juga wajib. Di sinilah peran konstruktor default di Java yang menetapkan variabel instan ini dengan nilai defaultnya.

**Catatan:** Kami biasanya menyebut konstruktor default sebagai konstruktor tanpa arg tetapi sebenarnya tidak sama. Konstruktor no-arg masih merupakan konstruktor yang dibuat oleh pengguna dan bukan oleh kompiler. Meskipun keduanya memiliki tujuan yang sama, yang satu dibuat secara otomatis sedangkan yang lainnya dibuat oleh pengguna.

**Contoh Konstruktor Default**

Mari kita ambil contoh **produk** yang terdiri dari data seperti id , product\_name , dan juga price .

Perhatikan apa yang terjadi ketika tidak ada konstruktor yang dideklarasikan dalam kode oleh programmer, dan tetap saja, kami mencoba mengakses variabel instan.

*//Program to show the use of default constructor*

import java.io.\*;

import java.util.\*;

*//Creating a class 'Product'*

public class Product {

String id;

String product\_name;

double price;

*//No creation of any constructor by the user*

public static void main (String[] args) {

*//Creating an object by calling*

*//default constructor*

Product product = new Product();

*//Trying to access the variables and print their default values*

System.out.println(product.id);

System.out.println(product.product\_name);

System.out.println(product.price);

}

}

**Keluaran:**

null

null

0.0

Berhasil! Kode di atas tidak menimbulkan kesalahan apa pun karena kompiler Java menyediakan konstruktor default untuk kode untuk menginisialisasi atribut dengan nilai default. Ya, konstruktor default ini tidak terlihat.

**Tujuan dari Konstruktor Default**

Seperti dibahas di atas, tujuan konstruktor default adalah untuk memberikan nilai default ke objek. Bilangan bulat akan diinisialisasi dengan00, gandakan dengan0.00 .  0, boolean dengan False, dan String dengan Null. Itu tergantung pada jenis variable instan yang dideklarasikan di kelas tempat objek dibuat.

**Penting**

Mari pertimbangkan kasus lain di mana kita akan membuat konstruktor berparameter.

*//Program to call the default constructor*

*//when parameterized, one is declared*

import java.io.\*;

import java.util.\*;

*//Creating a class for products*

public class Product {

*//Declaring instance*

String id;

String product\_name;

double price;

*//Calling parametrized constructor*

public Product(String id, String product\_name, double price)

{

this.id = id;

this.product\_name = product\_name;

this.price = price;

}

public static void main (String[] args) {

*//Creating object of class 'Product' by calling the default constructor*

Product product = new Product();

*//Trying to access the variables and print without assigning the default values*

System.out.println("Price for " + product.product\_name + " with product\_id " + product.id + " is " + product.price);

}

}

**Kesalahan kompilasi:**

Product.java:23: error: constructor Product in class Product cannot be applied to given types;

Product product = new Product();

^

required: String,String,double

found: no arguments

reason: actual and formal argument lists differ in length

1 error

Kami telah membuat konstruktor berparameter dalam kode kami, dan kami mencoba mengakses konstruktor default. Itu melempar kesalahan .

**Catatan:** Saat kita membuat konstruktor secara manual, maka kompiler tidak akan memasukkan konstruktor default. Jadi, kita juga harus membuat konstruktor default secara manual, jika diperlukan.

Jika kita telah membuat konstruktor berparameter sendiri dan kebetulan, kita mencoba memanggil konstruktor default, maka kita juga bertanggung jawab untuk membuat konstruktor default kita sendiri .

Tapi ya, ini hanya terjadi ketika kita memanggil konstruktor default, sedangkan kita hanya membuat konstruktor berparameter. Anda dapat melihat hal yang sama pada kode yang diberikan di bawah ini.

*//Program to show the use of parametrized constructor*

import java.io.\*;

import java.util.\*;

*//Creating a class for products*

public class Product {

*//Declaring instance*

String id;

String product\_name;

double price;

*//Calling parametrized constructor*

public Product(String id, String product\_name, double price)

{

this.id = id;

this.product\_name = product\_name;

this.price = price;

}

public static void main (String[] args) {

*//Creating object of class 'Product'*

*//Here, we'll assign the values for the variables*

Product product = new Product("w001", "watch", 15000);

*//Trying to access the variables and print without assigning the default values*

System.out.println("Price for " + product.product\_name + " with product\_id " + product.id + " is " + product.price);

}

}

**Keluaran:**

Price for watch with product\_id w001 is 15000.0

Masalah tidak terjadi di sini karena kami tidak membuat atau memanggil konstruktor default. Kami memiliki konstruktor berparameter dan kami hanya memanggilnya, jadi tidak perlu konstruktor default.

**Kesimpulan**

* Konstruktor default di Java dibuat secara otomatis oleh [kompiler Java online](https://www.scaler.com/topics/java/online-java-compiler/" \o "" \t "_blank) ketika pemrogram tidak membuat konstruktor apa pun di seluruh program.
* Itu dibuat untuk menetapkan nilai default ke variabel instan kelas saat objek dibuat.
* Konstruktor default terkadang disebut konstruktor tanpa arg karena keduanya bekerja sama. Tetapi konstruktor no-arg dibuat oleh pengguna sedangkan konstruktor default hanya dapat dibuat oleh kompiler.

B. Konstruktor Parameter

Konstruktor pada dasarnya adalah metode yang secara otomatis dipanggil ketika sebuah objek (instance) dibuat dari kelas tersebut. Ini digunakan untuk menginisialisasi anggota data objek.

**public** **class** Edureka{

Edureka(){ System.out.println("constructor is made");}

}

**Beberapa fitur konstruktor meliputi:**

1. Itu memiliki nama yang sama dengan nama kelas
2. Itu tidak memiliki tipe pengembalian

**Jenis Pembangun**

1. Konstruktor Default
2. Konstruktor berparameter

Konstruktor disebut Parameterized Constructor ketika menerima sejumlah parameter tertentu. Untuk menginisialisasi anggota data kelas dengan nilai yang berbeda.

Contoh yang mengilustrasikan Konstruktor Berparameter:

**public** **class** Edureka{

String studentName;

**int** studentAge;

//constructor

Edureka(String name, **int** age){

studentName = name;

studentAge = age;

}

**void** display(){

System.out.println(studentName+ " "+studentAge);

}

**public** **static** **void** main(String args[])

{

Edureka myObj = **new** Edureka("Manan" , 19);

myObj.display();

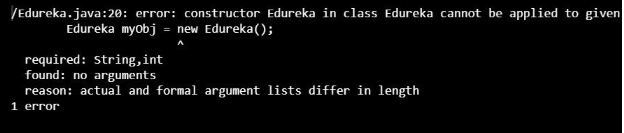
}

}

**Keluaran:** Manan-19

Dalam contoh di atas, kami mengirimkan string dan integer ke objek. Konstruktor kemudian menginisialisasi studentName dan studentAge menggunakan nilai yang diteruskan. Metode tampilan kemudian memberikan output yang diinginkan.

Dengan konstruktor berparameter untuk sebuah kelas, seseorang harus memberikan nilai awal sebagai argumen, jika tidak, kompiler akan melaporkan kesalahan.



**Melewati Objek Sebagai Argumen**

Kita juga bisa menyampaikan argumen sambil membuat instance kelas lainnya. Dengan cara ini konstruktor berparameter memenuhi kebutuhan untuk menyalin nilai satu objek ke objek lainnya.

Contoh yang mengilustrasikan Melewati Objek sebagai Argumen:

**public** **class** Edureka{

String studentName;

Edureka(String name){

studentName = name;

}

Edureka(Edureka myObj){

**this**.studentName = myObj.studentName;

}

**void** display(){

System.out.println("Student:" + studentName);

}

**public** **static** **void** main(String args[])

{

Edureka obj1 = **new** Edureka("Manan");

/\* passing the object as an argument for the constructor

\* this will invoke the copy constructor

\*/

Edureka obj2 = **new** Edureka(obj1);

System.out.println("Printing obj1 - ");

obj1.display();

System.out.println("Printing obj2 - ");

obj2.display();

}

}

**Keluaran:**

Mencetak objek 1 -

Manan

Mencetak objek 2 -

Manan

Pada contoh di atas, kita menginisialisasi obj1 menggunakan string. Kami kemudian meneruskan obj1 sebagai argumen saat membuat obj2. Terakhir, ketika kita mencetak variabel studentName kedua objek menggunakan fungsi display kita mendapatkan “Manan”.

**Memanggil Konstruktor Default Dari Konstruktor Berparameter di Java**

Terkadang ada kebutuhan untuk memanggil konstruktor default dari konstruktor lain di kelas yang sama. [kata kunci ini](https://www.edureka.co/blog/this-keyword-in-java/" \t "_blank) memenuhi tujuan ini.

Contoh ilustrasi panggilan ke konstruktor default dari konstruktor berparameter:

|  |
| --- |
| **public** **class** Edureka{  String studentName;  **int** studentAge;  String member;  Edureka(){  member = "YES";  }  Edureka(String name , **int** age){  **this**();  /\* this is used for calling the default constructor  \*from parameterized constructor  \*/  studentName = name;  studentAge = age;  }  **void** display(){  System.out.println(studentName + " -" + studentAge+ "-"+ "Member" + member);  }  **public** **static** **void** main(String args[])  {  Edureka obj = **new** Edureka("Manan" , 21);  obj.display();  }  } |

**Keluaran:** Manan – 21 – Anggota YA

Dalam contoh di atas, ketika konstruktor berparameter dipanggil, ia pertama kali memanggil konstruktor default dengan bantuan kata kunci this() . Konstruktor default menginisialisasi variabel "anggota" menjadi "YA" dan kemudian melanjutkan untuk mengeksekusi konstruktor berparameter.

**Overload Konstruktor**

Konstruktor mendukung kelebihan metode seperti kelas lainnya. Berdasarkan jenis atau jumlah argumen yang berbeda, konstruktor yang berbeda akan dipanggil.

Contoh yang mengilustrasikan Constructor Overloading:

|  |
| --- |
| **public** **class** Rectangle{  **int** length;  **int** breadth;  String color;  //constructor 1  Rectangle( **int** l , **int** b){  length = l;  breadth = b;  color = "Green";  }  //constructor 2  Rectangle(**int** l, **int** b, String c){  length = l;  breadth = b;  color = c;  }  **void** display(){  System.out.println("Length-" + length + "Breadth-" + breadth+ "Color" + color);  }  **public** **static** **void** main(String args[]){  Rectangle obj1 = **new** Rectangle(2,4);  Rectangle obj2 = **new** Rectangle(2,4,"Green");  obj1.display();  obj2.display();  }  } |

**Keluaran:**

Panjang - 2 Lebar - 4 Warna - Panjang Hijau

- 2 Lebar - 4 Warna – Merah

Sekarang setelah Anda memahami apa itu konstruktor dan cara bekerja dengannya, Anda selangkah lebih dekat dalam perjalanan Anda untuk mempelajari Java. Konsep seperti konstruktor itu sederhana tetapi sangat penting karena melibatkan kelas dan objek.

C. Konstruktor Kopikan

Copy constructor di kelas Java adalah [konstruktor](https://www.baeldung.com/java-constructors) yang **membuat objek menggunakan objek lain dari kelas Java yang sama**.

Itu berguna saat kita ingin menyalin objek kompleks yang memiliki beberapa bidang, atau saat kita ingin membuat salinan [mendalam](https://www.baeldung.com/java-deep-copy) dari objek yang sudah ada.

Untuk membuat copy constructor, pertama-tama kita dapat mendeklarasikan sebuah constructor yang menggunakan objek dengan tipe yang sama sebagai parameter:

public class Employee {

private int id;

private String name;

public Employee(Employee employee) {

}

}

Kemudian, kami menyalin setiap kolom dari objek input ke instance baru:

public class Employee {

private int id;

private String name;

public Employee(Employee employee) {

this.id = employee.id;

this.name = employee.name;

}

}

Apa yang kita miliki di sini adalah  *salinan dangkal* , yang bagus karena semua bidang kita – *int* dan *String* dalam kasus ini – adalah [tipe primitif](https://www.baeldung.com/java-primitives) atau [tipe yang tidak dapat diubah](https://www.baeldung.com/java-immutable-object) .

Jika kelas Java memiliki bidang yang dapat diubah, maka kita dapat membuat *[salinan yang dalam](https://www.baeldung.com/java-deep-copy)* di dalam pembuat salinannya. Dengan salinan yang dalam, objek yang baru dibuat tidak bergantung pada objek aslinya karena kami membuat salinan berbeda dari setiap objek yang dapat diubah:

public class Employee {

private int id;

private String name;

private Date startDate;

public Employee(Employee employee) {

this.id = employee.id;

this.name = employee.name;

this.startDate = new Date(employee.startDate.getTime());

}

}

Di Java, kita juga bisa menggunakan metode *[clone](https://www.baeldung.com/java-deep-copy)* untuk membuat objek dari objek yang sudah ada. Namun, copy constructor memiliki beberapa keunggulan dibandingkan metode *clone* :

1. Pembuat salinan jauh lebih mudah diimplementasikan. Kita tidak perlu mengimplementasikan antarmuka *Cloneable* dan menangani *CloneNotSupportedException* .
2. Metode *klon*  mengembalikan referensi *Objek* umum . Oleh karena itu, kita perlu mengetikkannya ke jenis yang sesuai.
3. Kami tidak dapat menetapkan nilai ke bidang *akhir* dalam metode *klon* . Namun, kita dapat melakukannya di copy constructor.

Copy constructor di Java tidak dapat diwariskan oleh subclass. Oleh karena itu, jika kami mencoba menginisialisasi objek anak dari referensi kelas induk, kami akan menghadapi masalah casting saat mengkloningnya dengan copy constructor.

Untuk mengilustrasikan masalah ini, pertama-tama mari buat subkelas Karyawan dan konstruktor salinannya:

public class Manager extends Employee {

private List<Employee> directReports;

// ... other constructors

public Manager(Manager manager) {

super(manager.id, manager.name, manager.startDate);

this.directReports = directReports.stream()

.collect(Collectors.toList());

}

}

Kemudian, kita mendeklarasikan variabel *Employee* dan membuat instance-nya dengan konstruktor *Manager* :

Employee source = new Manager(1, "Baeldung Manager", startDate, directReports);

Karena tipe referensinya adalah *Employee* , kita harus mentransmisikannya ke tipe *Manager* sehingga kita bisa menggunakan copy constructor dari class *Manager* :

Employee clone = new Manager((Manager) source);

Kita mungkin mendapatkan *ClassCastException* saat runtime jika objek masukan bukan turunan dari kelas *Manajer* .

Salah satu cara untuk menghindari casting di copy constructor adalah dengan membuat metode baru yang dapat diwariskan untuk kedua kelas:

public class Employee {

public Employee copy() {

return new Employee(this);

}

}

public class Manager extends Employee {

@Override

public Employee copy() {

return new Manager(this);

}

}

Di setiap metode kelas, kami memanggil konstruktor salinannya dengan masukan dari objek *ini* . Dengan cara ini, kami dapat menjamin bahwa objek yang dihasilkan sama dengan objek pemanggil:

Employee clone = source.copy();

## 2.3 Perbedaan antara Konstruktor dan Metode

Konstruktor dan metode merupakan dua konsep penting dalam pemrograman berorientasi objek. Meskipun keduanya terlihat serupa karena keduanya menggunakan blok kode untuk melakukan tindakan tertentu, namun ada beberapa perbedaan utama antara konstruktor dan metode.

1. Tujuan

Tujuan utama konstruktor adalah untuk menginisialisasi objek saat objek dibuat, sedangkan tujuan utama metode adalah untuk melakukan tindakan atau operasi pada objek setelah objek dibuat. Dengan kata lain, konstruktor digunakan untuk menginisialisasi nilai-nilai awal dari variabel-variabel instance dalam suatu kelas, sementara metode digunakan untuk melakukan tindakan tertentu pada variabel-variabel instance.

1. Nama

Konstruktor memiliki nama yang sama dengan nama kelas, sedangkan metode memiliki nama yang berbeda dari nama kelas.

1. Tipe Kembalian

Konstruktor tidak memiliki tipe kembalian, sedangkan metode memiliki tipe kembalian.

1. Parameter

Konstruktor dapat memiliki atau tidak memiliki parameter, tergantung pada kebutuhan. Namun, setidaknya satu konstruktor harus ada dalam suatu kelas. Sedangkan, metode selalu memiliki parameter yang didefinisikan di dalam kurung (). Jumlah dan jenis parameter tergantung pada kebutuhan.

1. Dipanggil

Konstruktor dipanggil secara otomatis saat objek dibuat, sementara metode harus dipanggil secara eksplisit menggunakan operator titik (.) pada objek yang sesuai.

1. Objek

Konstruktor digunakan untuk membuat objek baru dari suatu kelas, sedangkan metode digunakan untuk melakukan operasi pada objek yang sudah dibuat.

Kesimpulannya, konstruktor dan metode adalah dua konsep dasar dalam pemrograman berorientasi objek. Meskipun keduanya menggunakan blok kode untuk melakukan tindakan tertentu, tetapi terdapat perbedaan utama dalam tujuan, nama, tipe kembalian, parameter, dipanggil, dan objek. Konstruktor digunakan untuk menginisialisasi objek saat objek dibuat, sedangkan metode digunakan untuk melakukan tindakan tertentu pada objek setelah objek dibuat.

## 2.4 Penggunaan Konstruktor pada Objek

Konstruktor adalah metode khusus dalam Java yang digunakan untuk membuat objek baru dari suatu kelas. Konstruktor umumnya digunakan untuk menginisialisasi nilai-nilai awal dari variabel-variabel instance dalam suatu kelas sehingga objek dapat berfungsi sebagaimana mestinya. Setiap kali objek dibuat, konstruktor akan dipanggil secara otomatis dan menyiapkan objek untuk digunakan.

Penggunaan konstruktor pada objek sangatlah penting dalam pemrograman berorientasi objek karena konstruktor memungkinkan pengguna untuk membuat objek baru dari suatu kelas dan menginisialisasi variabel-variabel instance dengan nilai-nilai awal yang diinginkan. Konstruktor juga dapat digunakan untuk melakukan validasi pada input pengguna, sehingga memastikan objek dibuat dengan benar dan sesuai dengan kebutuhan.

Penggunaan konstruktor pada objek dapat meningkatkan efisiensi dan keamanan program karena variabel-variabel instance dalam suatu kelas sudah diinisialisasi dengan nilai-nilai awal yang benar dan sesuai dengan kebutuhan, sehingga dapat menghindari kesalahan atau bug yang mungkin terjadi karena nilai-nilai yang tidak valid.

Konstruktor juga dapat digunakan untuk membuat objek dari kelas turunan dengan cara memanggil konstruktor kelas induk. Dalam hal ini, konstruktor kelas turunan akan memanggil konstruktor kelas induk terlebih dahulu sebelum melakukan inisialisasi variabel-variabel instance kelas turunan.

Pada dasarnya, penggunaan konstruktor pada objek memungkinkan kita untuk membuat objek dengan nilai-nilai awal yang diinginkan dan memastikan objek dibuat dengan benar dan sesuai dengan kebutuhan. Konstruktor juga memungkinkan pengguna untuk membuat objek dari kelas turunan dengan memanggil konstruktor kelas induk terlebih dahulu. Oleh karena itu, penggunaan konstruktor pada objek sangatlah penting dalam pemrograman berorientasi objek dan dapat meningkatkan efisiensi dan keamanan program.

# BAB III

# PEMBAHASAN

## 3.1 Kelebihan Penggunaan Konstruktor dalam Bahasa Java

Konstruktor merupakan fitur penting dalam bahasa pemrograman Java karena memiliki berbagai kelebihan yang dapat meningkatkan efisiensi dan keamanan program. Berikut ini adalah beberapa kelebihan penggunaan konstruktor dalam bahasa Java:

1. Memastikan objek dibuat dengan benar

Dalam pemrograman berorientasi objek, objek yang dibuat harus memiliki nilai-nilai awal yang benar dan sesuai dengan kebutuhan. Konstruktor memungkinkan pengguna untuk menginisialisasi variabel-variabel instance dalam suatu kelas dengan nilai-nilai awal yang diinginkan sehingga memastikan objek dibuat dengan benar dan sesuai dengan kebutuhan. Hal ini dapat menghindari kesalahan atau bug yang mungkin terjadi karena nilai-nilai yang tidak valid.

1. Memudahkan penggunaan kelas turunan

Penggunaan konstruktor juga memudahkan penggunaan kelas turunan. Dalam bahasa Java, konstruktor kelas turunan dapat memanggil konstruktor kelas induk terlebih dahulu sebelum melakukan inisialisasi variabel-variabel instance kelas turunan. Hal ini memungkinkan pengguna untuk membuat objek dari kelas turunan dengan nilai-nilai awal yang sama dengan kelas induk.

1. Meningkatkan keamanan program

Konstruktor dapat digunakan untuk melakukan validasi pada input pengguna sehingga memastikan objek dibuat dengan benar dan sesuai dengan kebutuhan. Hal ini dapat meningkatkan keamanan program karena mencegah pengguna untuk membuat objek dengan nilai-nilai yang tidak valid atau tidak diinginkan.

1. Mempercepat eksekusi program

Penggunaan konstruktor dapat mempercepat eksekusi program karena menghindari penggunaan metode inisialisasi variabel-variabel instance dalam suatu kelas secara manual. Dalam hal ini, pengguna hanya perlu memanggil konstruktor dan variabel-variabel instance akan diinisialisasi secara otomatis.

1. Memungkinkan overloading konstruktor

Konstruktor juga dapat digunakan untuk overloading sehingga memungkinkan pengguna untuk membuat objek dengan parameter yang berbeda. Dalam hal ini, pengguna dapat membuat beberapa konstruktor dengan parameter yang berbeda untuk membuat objek dengan nilai-nilai awal yang berbeda pula.

1. Meningkatkan fleksibilitas program

Konstruktor memungkinkan pengguna untuk membuat objek baru dari suatu kelas dengan nilai-nilai awal yang diinginkan. Hal ini meningkatkan fleksibilitas program karena memungkinkan pengguna untuk mengubah nilai-nilai awal dari variabel-variabel instance dalam suatu kelas sesuai dengan kebutuhan.

Dalam kesimpulannya, penggunaan konstruktor dalam bahasa Java memiliki berbagai kelebihan yang dapat meningkatkan efisiensi dan keamanan program. Konstruktor memastikan objek dibuat dengan benar, memudahkan penggunaan kelas turunan, meningkatkan keamanan program, mempercepat eksekusi program, memungkinkan overloading konstruktor, dan meningkatkan fleksibilitas program. Oleh karena itu, penggunaan konstruktor sangat dianjurkan dalam pemrograman berorientasi objek dalam bahasa Java.

## 3.2 Kendala Penggunaan Konstruktor dalam Bahasa Java

Konstruktor merupakan sebuah metode khusus dalam bahasa pemrograman Java yang digunakan untuk menginisialisasi objek saat dibuat. Konstruktor digunakan untuk menentukan nilai awal dari suatu objek, sehingga objek tersebut dapat digunakan dengan benar di dalam program. Meskipun konstruktor sangat penting dalam pembuatan objek di dalam program Java, namun penggunaan konstruktor juga memiliki beberapa kendala yang harus diperhatikan.

Berikut adalah beberapa kendala penggunaan konstruktor dalam bahasa Java:

1. Tidak Bisa Diwarisi

Konstruktor pada kelas induk tidak dapat diwarisi ke kelas anak. Ini berarti bahwa setiap kelas anak harus memiliki konstruktor sendiri untuk menginisialisasi objek yang dibuatnya. Hal ini bisa menyebabkan duplikasi kode dan membuat kode sulit untuk dipelihara dan diperbarui.

1. Penggunaan Berlebihan

Penggunaan konstruktor yang berlebihan dapat menyebabkan kompleksitas dan kebingungan dalam kode. Terlalu banyak konstruktor dapat mempersulit proses pemeliharaan dan pengembangan kode. Selain itu, konstruktor yang terlalu kompleks juga dapat menyebabkan performa program menjadi lambat.

1. Tidak Bisa Dipanggil Secara Langsung

Konstruktor tidak dapat dipanggil secara langsung oleh program. Konstruktor hanya dapat dipanggil saat objek dibuat. Hal ini dapat menyebabkan kesulitan dalam menguji dan memodifikasi kode.

1. Konstruktor Tidak Bisa Diubah

Setelah sebuah objek dibuat, konstruktor tidak dapat diubah. Ini berarti bahwa nilai-nilai yang telah diinisialisasi oleh konstruktor akan tetap sama selama objek tersebut masih ada. Hal ini dapat menyebabkan masalah jika nilai-nilai yang telah diinisialisasi perlu diubah pada saat runtime.

1. Tidak Bisa Mengembalikan Nilai

Konstruktor tidak dapat mengembalikan nilai. Ini berarti bahwa tidak mungkin untuk mengembalikan nilai dari sebuah konstruktor ke variabel lain dalam program. Hal ini dapat menyulitkan dalam beberapa situasi, seperti ketika sebuah objek perlu mengembalikan nilai-nilai yang dihitung pada saat pembuatan objek.

1. Parameter Konstruktor Tidak Terdefinisi

Parameter konstruktor tidak selalu terdefinisi dengan jelas. Hal ini dapat menyebabkan kesulitan dalam menginisialisasi objek yang benar. Terkadang, penggunaan konstruktor yang salah atau parameter yang salah dapat menyebabkan program tidak berjalan dengan benar.

1. Tidak Ada Default Constructor

Jika tidak ada konstruktor yang didefinisikan dalam sebuah kelas, maka Java akan membuat default constructor. Namun, jika sebuah kelas memiliki setidaknya satu konstruktor yang didefinisikan, maka default constructor tidak akan dibuat oleh Java. Hal ini dapat menyebabkan masalah jika sebuah objek dibuat tanpa argumen dan tidak ada konstruktor default yang tersedia.

1. Parameter Konstruktor Terlalu Banyak

Jika sebuah konstruktor memiliki terlalu banyak parameter, maka hal ini dapat menyulitkan dalam menginisialisasi objek. Terkadang, parameter yang terlalu banyak juga dapat menyebabkan kesulitan dalam membaca dan memahami kode. Hal ini juga dapat membuat program menjadi kurang fleksibel dan sulit untuk dimodifikasi.

1. Konstruktor Tidak Bisa Dibuat Private

Konstruktor tidak dapat dibuat sebagai private, karena hal ini akan mencegah pembuatan objek dari kelas tersebut oleh kelas lain di dalam program. Namun, terkadang kita ingin mencegah pembuatan objek dari kelas tersebut oleh program lain, sehingga penggunaan konstruktor tidak dapat membantu dalam kasus tersebut.

1. Konstruktor Tidak Dapat Di-Override

Konstruktor tidak dapat di-override oleh kelas anak. Hal ini berbeda dengan metode biasa yang dapat di-override oleh kelas anak. Karena konstruktor tidak dapat di-override, maka kelas anak harus membuat konstruktor sendiri untuk menginisialisasi objeknya.

Meskipun penggunaan konstruktor memiliki beberapa kendala, namun konstruktor tetap sangat penting dalam pembuatan objek di dalam program Java. Konstruktor memungkinkan kita untuk menginisialisasi objek dengan benar sehingga objek tersebut dapat digunakan dengan tepat. Oleh karena itu, kita harus memperhatikan kendala-kendala penggunaan konstruktor di atas dan berusaha untuk menghindari atau memperbaikinya sebisa mungkin.

## 3.3 Solusi atas Kendala Penggunaan Konstruktor dalam Bahasa Java

Meskipun konstruktor dalam bahasa Java memiliki beberapa kendala yang dapat mengganggu dalam proses pembuatan objek, namun kita dapat mengatasi kendala-kendala tersebut dengan beberapa solusi berikut:

1. Menggunakan Inheritance

Salah satu solusi untuk mengatasi kendala konstruktor yang tidak bisa diwariskan adalah dengan menggunakan inheritance. Dengan inheritance, kita dapat menggunakan konstruktor dari kelas induk dan meneruskan nilainya ke kelas anak. Hal ini dapat mengurangi duplikasi kode dan memudahkan pemeliharaan dan pengembangan kode.

1. Menggunakan Overloading

Menggunakan overloading adalah solusi lain untuk mengatasi kendala parameter konstruktor yang terlalu banyak atau tidak jelas. Dengan overloading, kita dapat membuat beberapa konstruktor dengan jumlah dan jenis parameter yang berbeda, sehingga pengguna program dapat memilih konstruktor yang sesuai dengan kebutuhan mereka.

1. Menggunakan Factory Method

Factory method adalah solusi yang berguna jika kita ingin membuat objek dengan nilai-nilai yang sudah diatur sebelumnya. Dalam factory method, kita membuat metode statis yang mengembalikan objek dengan nilai-nilai yang diinginkan. Dengan cara ini, kita dapat memastikan bahwa objek yang dibuat memiliki nilai-nilai yang diinginkan tanpa harus menggunakan konstruktor yang kompleks.

1. Menggunakan Singleton Pattern

Jika kita ingin memastikan bahwa hanya satu objek dari suatu kelas yang ada di dalam program, kita dapat menggunakan Singleton Pattern. Dalam Singleton Pattern, kita membuat konstruktor private dan metode statis yang mengembalikan objek dari kelas tersebut. Dengan cara ini, hanya satu objek dari kelas tersebut yang akan ada di dalam program.

1. Menggunakan Builder Pattern

Builder Pattern adalah solusi yang berguna jika kita ingin membuat objek dengan parameter yang banyak atau kompleks. Dalam Builder Pattern, kita membuat kelas builder yang memungkinkan pengguna program untuk mengatur parameter objek dengan cara yang mudah dan intuitif. Dengan cara ini, pengguna program dapat membuat objek dengan nilai-nilai yang diinginkan tanpa harus memahami kompleksitas konstruktor.

1. Menggunakan Dependency Injection

Dependency Injection adalah solusi yang berguna jika kita ingin membuat objek yang saling bergantung satu sama lain. Dalam Dependency Injection, kita membuat objek-objek yang dibutuhkan di dalam kelas konstruktor dan menyediakan objek-objek tersebut sebagai parameter konstruktor. Dengan cara ini, objek-objek yang saling bergantung dapat diinisialisasi dengan benar dan program dapat berjalan dengan baik.

1. Menggunakan Default Constructor

Jika kita ingin membuat objek tanpa argumen, kita dapat menggunakan default constructor. Default constructor akan dihasilkan oleh Java jika tidak ada konstruktor yang didefinisikan di dalam kelas. Namun, jika kita telah mendefinisikan konstruktor sendiri, kita harus membuat default constructor sendiri jika kita ingin membuat objek tanpa argumen.

Dalam penggunaan konstruktor di dalam program Java, kita harus memperhatikan kendala-kendala yang ada dan mencari solusi yang tepat untuk mengatasi kendala tersebut. Dengan mengatasi kendala penggunaan konstruktor, kita dapat membuat program yang lebih mudah dipelihara, mudah dimengerti, dan lebih fleksibel.

# BAB IV

# KESIMPULAN

## 4.1 Ringkasan hasil Peneliatian

penelitian tentang konstruktor dalam bahasa Java dapat memberikan beberapa hasil penting untuk memperkaya pemahaman tentang konstruktor dan penggunaannya. Dalam penelitian ini, pembahasan dimulai dengan latar belakang dan tujuan penelitian yang mengidentifikasi kebutuhan untuk memahami konstruktor dalam bahasa Java dan manfaatnya dalam pengembangan program.

Penelitian juga mengeksplorasi pengertian konstruktor, baik secara umum maupun secara khusus dalam bahasa Java. Penelitian ini menunjukkan bahwa konstruktor adalah metode khusus dalam suatu kelas yang digunakan untuk menginisialisasi variabel instance dengan nilai-nilai awal tertentu saat objek dibuat. Konstruktor default didefinisikan secara otomatis dalam setiap kelas dan digunakan saat tidak ada konstruktor lain yang didefinisikan dalam kelas tersebut.

Selanjutnya, penelitian membahas perbedaan antara konstruktor dan metode, yang penting untuk dipahami karena konstruktor memiliki fungsi khusus dalam pembuatan objek. Meskipun memiliki beberapa kesamaan dengan metode, konstruktor memiliki beberapa perbedaan utama, termasuk nama, tidak mengembalikan nilai, dan dipanggil secara otomatis saat objek dibuat.

Penelitian kemudian menjelaskan penggunaan konstruktor pada objek, yang mencakup memanggil konstruktor secara eksplisit dan mengakses variabel instance melalui konstruktor. Penelitian juga menunjukkan bahwa penggunaan konstruktor dapat memudahkan penggunaan kelas turunan, meningkatkan keamanan program, dan meningkatkan efisiensi dan fleksibilitas program.

Terakhir, penelitian mengidentifikasi kelebihan penggunaan konstruktor dalam bahasa Java, termasuk memastikan objek dibuat dengan benar, memudahkan penggunaan kelas turunan, meningkatkan keamanan program, mempercepat eksekusi program, memungkinkan overloading konstruktor, dan meningkatkan fleksibilitas program.

Secara keseluruhan, penelitian tentang konstruktor dalam bahasa Java memberikan pemahaman yang lebih dalam tentang konstruktor dan manfaatnya dalam pengembangan program. Penelitian ini menunjukkan bahwa penggunaan konstruktor sangat dianjurkan dalam pemrograman berorientasi objek dalam bahasa Java karena memiliki berbagai kelebihan yang dapat meningkatkan efisiensi dan keamanan program.

## 4.2 Saran

ada beberapa saran yang dapat diberikan terkait dengan penggunaan konstruktor dalam bahasa Java. Pertama, disarankan untuk memahami pengertian konstruktor secara menyeluruh, baik secara umum maupun secara khusus dalam bahasa Java. Ini akan membantu pengembang program memahami fungsi dan tujuan konstruktor serta cara menggunakan konstruktor dalam program mereka.

Kedua, pengembang program harus memahami perbedaan antara konstruktor dan metode untuk memastikan penggunaan yang tepat dan efektif dari konstruktor dalam program. Perbedaan penting seperti nama, pengembalian nilai, dan pemanggilan otomatis harus dipahami dengan baik.

Ketiga, disarankan untuk menggunakan konstruktor secara aktif dalam program karena keuntungan yang diberikan oleh penggunaan konstruktor sangatlah signifikan. Penggunaan konstruktor dapat meningkatkan efisiensi program, meningkatkan keamanan program, dan memudahkan penggunaan kelas turunan, serta memberikan fleksibilitas program yang lebih besar.

Keempat, disarankan untuk menggunakan konstruktor default dengan bijaksana. Meskipun konstruktor default didefinisikan secara otomatis dalam setiap kelas, penggunaannya harus dipertimbangkan secara cermat karena konstruktor default mungkin tidak memenuhi kebutuhan program dengan benar.

Kelima, disarankan untuk menggunakan konstruktor dengan baik dan sesuai dengan tujuan yang diinginkan. Konstruktor yang baik harus dapat menginisialisasi variabel instance dengan nilai-nilai awal yang tepat, serta mempertahankan struktur kelas dan objek yang benar.

Terakhir, disarankan untuk terus memperbarui pengetahuan tentang konstruktor dalam bahasa Java, karena bahasa Java terus berkembang dan ada kemungkinan fitur-fitur baru yang diperkenalkan dalam bahasa tersebut. Pengembang program harus mengikuti perkembangan terbaru dalam bahasa Java untuk memastikan penggunaan konstruktor yang efektif dan efisien dalam program mereka.

# DAFTAR PUSTAKA

* <https://frieyadie.web.id/constructor-pada-java/#:~:text=Constructor%20merupakan%20suatu%20method%20yang,perintah%20new%2C%20membuat%20suatu%20objek>
* <https://www.petanikode.com/java-oop-constructor/>
* <https://www.masgani.com/constructor-pada-pemrograman-java/>
* <https://www.duniailkom.com/tutorial-oop-java-pengertian-constructor/>
* <https://smkbinakerjapwk.sch.id/belajar-java-oop-mengenal-constructor-destructor-dalam-java/>
* <https://www.community-java.com/2017/01/contoh-program-java-overload.html>
* <https://nextgen.co.id/konstruktor-pada-java/>
* <https://www.codeternity.com/id-id/article/java/java-konstruktor>
* <https://repository.unikom.ac.id/32352/1/PERTEMUAN%206.pdf>
* <https://appkey.id/pembuatan-aplikasi/aplikasi-android/java-constructor/>