MAKALAH

OVERLOADING DALAM BAHASA JAVA



Oleh :

Diki Candra

NIM 2022903430010

D4 TEKNOLOGI REKAYASA KOMPUTER JARINGAN

POLITEKNIK NEGERI LHOKSEMAWE

2022/2023

DAFTAR ISI

[BAB I 2](#_Toc136294143)

[PENDAHULUAN 2](#_Toc136294144)

[**1.1. Latar Belakang** 2](#_Toc136294145)

[**1.2. Tujuan Penulisan Makalah** 2](#_Toc136294146)

[BAB II 4](#_Toc136294147)

[PENGENALAN OVERLOADING 4](#_Toc136294148)

[**2.1 Definisi Overloading** 4](#_Toc136294149)

[**2.2 Keuntungan Overloading** 4](#_Toc136294150)

[**2.3 Perbedaan Overloading dengan Overriding** 5](#_Toc136294151)

[BAB III 7](#_Toc136294152)

[KONSEP DASAR OVERLOADING 7](#_Toc136294153)

[**3.2 Tipe Data dalam Overloading** 8](#_Toc136294154)

[**3.3 Return Type dalam Overloading** 9](#_Toc136294155)

[BAB IV 11](#_Toc136294156)

[SYARAT OVERLOADING 11](#_Toc136294157)

[**4.1 Nama Metode** 11](#_Toc136294158)

[**4.2 Tipe Parameter** 12](#_Toc136294159)

[**4.3 Urutan Parameter** 13](#_Toc136294160)

[**4.4 Jumlah Parameter** 14](#_Toc136294161)

[BAB V 16](#_Toc136294162)

[CONTOH IMPLEMENTASI 16](#_Toc136294163)

[OVERLOAING 16](#_Toc136294164)

[**5.1 Overloading pada Metode** 16](#_Toc136294165)

[**5.2 Overloading pada Konstruktor** 18](#_Toc136294166)

[BAB VI 22](#_Toc136294167)

[PEMECAHAN MASALAH DENGAN 22](#_Toc136294168)

[OVERLOADING 22](#_Toc136294169)

[**6.1 Studi Kasus 1: Menghitung Luas Bangun Datar** 22](#_Toc136294170)

[**6.2 Studi Kasus 2: Operasi Matematika** 24](#_Toc136294171)

[KESIMPULAN 29](#_Toc136294172)

# BAB I

# PENDAHULUAN

## **1.1. Latar Belakang**

Overloading adalah salah satu konsep fundamental dalam pemrograman berorientasi objek (OOP). Dalam bahasa pemrograman Java, overloading memungkinkan penggunaan nama metode yang sama dengan parameter yang berbeda dalam satu kelas. Dengan menggunakan overloading, programmer dapat membuat metode dengan fungsionalitas yang serupa namun menerima tipe data yang berbeda, sehingga memudahkan penggunaan dan meningkatkan fleksibilitas kode.

Pemahaman yang baik tentang overloading sangat penting bagi para pengembang Java, karena dapat membantu mereka mengelola kode yang lebih efisien dan mudah dimengerti. Dengan menggunakan overloading, pengembang dapat membuat kode yang lebih bersih dan mudah dipelihara, serta menghindari duplikasi kode yang tidak perlu.

Dalam makalah ini, kami akan membahas secara mendalam tentang konsep dasar overloading dalam bahasa Java, meliputi parameter, tipe data, dan return type dalam overloading. Kami juga akan memberikan contoh implementasi overloading pada metode dan konstruktor, serta menggambarkan pemecahan masalah khusus menggunakan overloading. Selain itu, kami akan membahas batasan dan keterbatasan overloading untuk membantu para pengembang dalam menggunakan konsep ini secara efektif.

Dengan mempelajari topik ini, diharapkan pembaca dapat memahami konsep overloading dengan baik, mengimplementasikannya dengan benar dalam kode Java, dan mengoptimalkan penggunaannya dalam pengembangan perangkat lunak.

## **1.2. Tujuan Penulisan Makalah**

Tujuan makalah yang dijelaskan dalam daftar isi di atas adalah sebagai berikut:

1. Memberikan pemahaman tentang konsep overloading dalam bahasa Java: Makalah ini bertujuan untuk memberikan pemahaman yang mendalam tentang konsep overloading dalam bahasa pemrograman Java. Dengan menjelaskan definisi, keuntungan, dan perbedaan overloading dengan overriding, pembaca akan memiliki pemahaman yang kuat tentang konsep ini.
2. Menjelaskan konsep dasar overloading: Makalah ini akan menjelaskan konsep dasar overloading, termasuk parameter, tipe data, dan return type dalam overloading. Hal ini akan membantu pembaca memahami cara mengimplementasikan overloading dengan benar dalam kode Java.
3. Menyediakan contoh implementasi overloading: Makalah ini akan memberikan contoh konkret tentang bagaimana overloading dapat diimplementasikan pada metode dan konstruktor dalam bahasa Java. Melalui contoh kode program, pembaca akan dapat melihat bagaimana overloading dapat meningkatkan fleksibilitas dan efisiensi dalam pengembangan perangkat lunak.
4. Menggambarkan pemecahan masalah dengan overloading: Makalah ini akan menggambarkan studi kasus yang mengilustrasikan bagaimana overloading dapat digunakan untuk memecahkan masalah khusus. Studi kasus ini akan memberikan wawasan tentang cara menggunakan overloading untuk mengatasi tantangan pemrograman yang berbeda.
5. Menjelaskan batasan dan keterbatasan overloading: Makalah ini akan memberikan pemahaman tentang batasan dan keterbatasan overloading dalam bahasa Java. Pembaca akan mempelajari situasi di mana overloading tidak dapat digunakan atau memerlukan perhatian khusus dalam penggunaannya.

Dengan menyajikan informasi yang komprehensif tentang overloading, tujuan makalah ini adalah memberikan pembaca pemahaman yang kuat tentang konsep ini dan membantu mereka mengaplikasikannya secara efektif dalam pengembangan perangkat lunak menggunakan bahasa Java.

# BAB II

# PENGENALAN OVERLOADING

## **2.1 Definisi Overloading**

Definisi Overloading merujuk pada bagian pendahuluan yang menjelaskan konsep overloading dalam bahasa Java. Berikut adalah penjelasan mengenai definisi overloading:

Overloading adalah konsep dalam pemrograman berorientasi objek (OOP) di mana sebuah kelas dapat memiliki beberapa metode atau konstruktor dengan nama yang sama, tetapi memiliki parameter yang berbeda. Dalam bahasa pemrograman Java, overloading memungkinkan penggunaan nama metode yang sama dalam satu kelas dengan perbedaan dalam tipe, urutan, atau jumlah parameter.

Dengan menggunakan overloading, programmer dapat membuat berbagai variasi metode dengan fungsionalitas yang serupa namun menerima jenis data yang berbeda. Ini memberikan fleksibilitas dalam penggunaan metode, karena pemanggilan metode dapat dilakukan dengan argumen yang sesuai dengan tipe data yang dibutuhkan.

Pada dasarnya, overloading memungkinkan penggunaan nama metode yang sama dalam satu kelas, namun memiliki parameter yang berbeda untuk membedakan antara metode-metode tersebut. Ketika sebuah metode dipanggil, Java akan memilih metode yang tepat berdasarkan jenis dan urutan argumen yang diberikan.

Overloading dapat membantu meningkatkan kejelasan dan kegunaan kode, karena nama metode yang sama digunakan untuk tugas yang mirip, tetapi beroperasi pada jenis data yang berbeda. Dengan demikian, overloading memungkinkan penggunaan yang lebih intuitif dan fleksibel dalam pemrograman Java.

Dalam makalah ini, akan dibahas secara mendalam tentang konsep overloading, termasuk syarat-syarat yang harus dipenuhi untuk melakukan overloading, contoh implementasi overloading pada metode dan konstruktor, serta manfaat dan batasan penggunaan overloading dalam bahasa Java.

## **2.2 Keuntungan Overloading**

Keuntungan menggunakan overloading dalam pemrograman Java antara lain:

1. Fleksibilitas: Overloading memungkinkan penggunaan metode dengan nama yang sama, tetapi dengan parameter yang berbeda. Hal ini memberikan fleksibilitas dalam pemanggilan metode, karena pengguna dapat memanggil metode yang sesuai dengan jenis dan urutan argumen yang mereka miliki.
2. Keterbacaan dan Kegunaan Kode: Dengan menggunakan overloading, programmer dapat menggunakan nama metode yang sama untuk tugas-tugas yang mirip, tetapi beroperasi pada jenis data yang berbeda. Ini membuat kode lebih mudah dipahami dan lebih intuitif bagi pengguna.
3. Pengurangan Duplikasi Kode: Overloading memungkinkan penggunaan kembali nama metode untuk berbagai tipe data atau situasi yang serupa. Dengan demikian, menghindari duplikasi kode yang tidak perlu dan memungkinkan penggunaan kembali logika yang sama untuk berbagai kasus.
4. Dukungan Terhadap Polimorfisme: Overloading merupakan salah satu bentuk polimorfisme dalam Java. Polimorfisme memungkinkan objek dengan tipe yang berbeda untuk merespons panggilan metode dengan cara yang sesuai. Overloading memperkaya polimorfisme dengan memungkinkan penggunaan nama metode yang sama untuk objek dengan tipe data yang berbeda.
5. Memudahkan Perubahan Parameter: Jika diperlukan perubahan pada parameter suatu metode, dengan overloading, programmer dapat menambahkan versi baru metode dengan parameter baru, sementara metode yang sudah ada tetap berfungsi untuk kasus yang lama. Ini mencegah perubahan yang merusak pada kode yang sudah ada.

Dengan memanfaatkan overloading, programmer dapat menciptakan kode yang lebih fleksibel, mudah dibaca, dan lebih efisien. Overloading memungkinkan penyesuaian yang lebih baik dengan berbagai jenis dan tipe data, serta memberikan dukungan polimorfisme yang penting dalam pemrograman berorientasi objek.

## **2.3 Perbedaan Overloading dengan Overriding**

berikut adalah penjelasan tentang perbedaan kedua konsep tersebut:

1. Definisi:

* Overloading: Overloading adalah konsep di mana sebuah kelas memiliki beberapa metode atau konstruktor dengan nama yang sama, tetapi memiliki parameter yang berbeda.
* Overriding: Overriding adalah konsep di mana sebuah kelas anak (subclass) mengimplementasikan kembali metode yang sudah dideklarasikan dalam kelas induk (superclass) dengan nama, parameter, dan tipe yang sama.

1. Tujuan:

* Overloading: Tujuan overloading adalah memberikan fleksibilitas dan kegunaan yang lebih baik dalam penggunaan metode dengan nama yang sama, tetapi berbeda dalam parameter.
* Overriding: Tujuan overriding adalah memungkinkan kelas anak untuk memodifikasi perilaku metode yang sudah didefinisikan dalam kelas induk sesuai dengan kebutuhan mereka sendiri.

1. Hubungan antara Kelas:

* Overloading: Overloading terjadi dalam satu kelas, di mana metode dengan nama yang sama tetapi parameter yang berbeda dapat dideklarasikan.
* Overriding: Overriding terjadi dalam hubungan kelas induk (superclass) dan kelas anak (subclass), di mana kelas anak mengganti implementasi metode yang sudah dideklarasikan dalam kelas induk.

1. Polimorfisme:

* Overloading: Overloading termasuk dalam konsep polimorfisme statis (compile-time polymorphism), di mana metode yang tepat dipilih saat kompilasi berdasarkan jenis dan urutan argumen.
* Overriding: Overriding termasuk dalam konsep polimorfisme dinamis (runtime polymorphism), di mana metode yang dipanggil ditentukan saat runtime berdasarkan jenis objek yang sebenarnya.

Keuntungan Overloading: Keuntungan menggunakan overloading dalam pemrograman Java telah dijelaskan sebelumnya dalam penjelasan sebelumnya. Dalam konteks daftar isi tersebut, keuntungan overloading mencakup fleksibilitas, keterbacaan kode yang meningkat, pengurangan duplikasi kode, dukungan terhadap polimorfisme, dan kemudahan dalam melakukan perubahan parameter metode.

Dengan memanfaatkan overloading, programmer dapat menciptakan kode yang lebih fleksibel, mudah dibaca, dan lebih efisien. Overloading memungkinkan penggunaan nama metode yang sama dengan parameter yang berbeda untuk memenuhi berbagai kebutuhan dalam pemrograman Java.

# BAB III

# KONSEP DASAR OVERLOADING

**3.1 Parameter dalam Overloading**

Parameter dalam Overloading: Dalam daftar isi tersebut, parameter dalam overloading adalah salah satu konsep dasar yang akan dibahas dalam makalah. Berikut adalah penjelasan mengenai parameter dalam overloading:

1. Definisi:

* Parameter adalah nilai atau variabel yang diberikan kepada suatu metode atau konstruktor. Parameter digunakan untuk mengirimkan data ke metode atau konstruktor saat pemanggilan.
* Dalam overloading, parameter memiliki peran penting dalam membedakan metode-metode yang memiliki nama yang sama. Metode-metode tersebut memiliki jumlah, tipe, atau urutan parameter yang berbeda.

1. Peran Parameter dalam Overloading:

* Parameter digunakan untuk membedakan antara metode-metode yang memiliki nama yang sama. Java menggunakan informasi dari parameter saat memilih metode yang tepat untuk dipanggil saat pemanggilan metode.
* Jika terdapat beberapa metode dengan nama yang sama, Java akan memilih metode yang memiliki parameter yang sesuai dengan argumen yang diberikan saat pemanggilan. Jika tidak ada metode yang memiliki parameter yang cocok, maka akan terjadi kesalahan kompilasi.

1. Perbedaan Parameter dalam Overloading:

* Tipe Parameter: Overloading dapat dilakukan dengan menggunakan tipe parameter yang berbeda. Misalnya, terdapat metode "printData(int x)" dan "printData(String str)" yang dapat dipanggil dengan argumen berbeda sesuai dengan tipe parameter yang digunakan.
* Jumlah Parameter: Overloading juga memungkinkan penggunaan metode dengan jumlah parameter yang berbeda. Misalnya, terdapat metode "calculateArea(int length)" dan "calculateArea(int length, int width)" yang memungkinkan penggunaan berbagai argumen tergantung pada jumlah parameter yang digunakan.
* Urutan Parameter: Overloading dapat dilakukan dengan mengubah urutan parameter pada metode yang memiliki nama yang sama. Misalnya, terdapat metode "printData(int x, String str)" dan "printData(String str, int x)" yang memungkinkan penggunaan argumen dalam urutan yang berbeda.

Dalam makalah, parameter dalam overloading akan dibahas secara rinci, termasuk contoh implementasi dan aturan yang perlu diperhatikan saat menggunakan parameter dalam overloading. Hal ini akan memberikan pemahaman yang lebih mendalam tentang bagaimana parameter digunakan untuk membedakan metode-metode yang memiliki nama yang sama dalam bahasa Java.

## **3.2 Tipe Data dalam Overloading**

Tipe Data dalam Overloading: tipe data dalam overloading adalah salah satu aspek yang akan dibahas dalam makalah. Berikut adalah penjelasan mengenai tipe data dalam overloading:

1. Definisi:

* Tipe data mengacu pada jenis nilai yang dapat dimiliki oleh suatu variabel atau parameter dalam bahasa pemrograman. Contoh tipe data dalam Java adalah int, double, String, boolean, dan sebagainya.
* Dalam overloading, tipe data digunakan sebagai salah satu faktor yang membedakan metode-metode yang memiliki nama yang sama.

1. Peran Tipe Data dalam Overloading:

* Tipe data digunakan untuk membedakan metode-metode yang memiliki nama yang sama dengan tipe parameter yang berbeda. Java menggunakan informasi tipe data parameter saat memilih metode yang tepat untuk dipanggil saat pemanggilan metode.
* Jika terdapat beberapa metode dengan nama yang sama, Java akan memilih metode yang memiliki tipe data parameter yang sesuai dengan argumen yang diberikan saat pemanggilan. Jika tidak ada metode yang memiliki tipe data parameter yang cocok, maka akan terjadi kesalahan kompilasi.

1. Contoh Tipe Data dalam Overloading:

* Misalnya, terdapat metode "printData(int x)" dan "printData(String str)" yang dapat digunakan untuk mencetak data. Metode "printData(int x)" akan dipanggil jika argumen yang diberikan adalah tipe data int, sedangkan metode "printData(String str)" akan dipanggil jika argumen yang diberikan adalah tipe data String.

1. Keterbatasan Tipe Data dalam Overloading:

* Meskipun tipe data digunakan sebagai faktor pembeda, Java memiliki aturan ketat dalam pemilihan metode yang sesuai. Terdapat aturan konversi tipe data yang diikuti oleh Java saat mencocokkan argumen dengan parameter. Dalam beberapa kasus, Java mungkin memilih konversi tipe data yang lebih umum, yang dapat menyebabkan ambiguitas atau kesalahan pemanggilan metode.

Dalam makalah, tipe data dalam overloading akan dibahas secara rinci, termasuk contoh implementasi dan aturan yang perlu diperhatikan saat menggunakan tipe data dalam overloading. Hal ini akan memberikan pemahaman yang lebih mendalam tentang bagaimana tipe data digunakan untuk membedakan metode-metode yang memiliki nama yang sama dalam bahasa Java.

## **3.3 Return Type dalam Overloading**

Return Type dalam Overloading adalah salah satu konsep yang akan dibahas dalam makalah. Berikut adalah penjelasan mengenai Return Type dalam Overloading:

1. Definisi:

* Return Type merujuk pada jenis nilai yang dikembalikan oleh suatu metode dalam bahasa pemrograman. Misalnya, metode dengan Return Type int akan mengembalikan nilai bertipe int, sementara metode dengan Return Type void tidak mengembalikan nilai.
* Dalam Overloading, Return Type tidak digunakan sebagai faktor pembeda dalam pemilihan metode yang tepat. Metode-metode Overloading memiliki nama dan parameter yang sama, tetapi Return Type dapat berbeda.

1. Peran Return Type dalam Overloading:

* Return Type berfungsi untuk mengindikasikan jenis nilai yang dikembalikan oleh suatu metode setelah dieksekusi. Ini membantu pemanggil metode dalam menentukan bagaimana hasil metode dapat digunakan.
* Dalam Overloading, Return Type tidak digunakan sebagai kriteria pemilihan metode yang tepat. Java hanya menggunakan informasi nama metode dan parameter saat memilih metode Overloading yang sesuai untuk dipanggil.

1. Contoh Return Type dalam Overloading:

* Misalnya, terdapat dua metode Overloading dengan nama yang sama, "calculateArea(int length)" dan "calculateArea(int length, int width)". Metode pertama memiliki Return Type int yang mengembalikan luas persegi, sedangkan metode kedua memiliki Return Type double yang mengembalikan luas persegi panjang.
* Meskipun Return Type berbeda, Java akan memilih metode yang sesuai berdasarkan nama dan parameter yang diberikan saat pemanggilan metode.

1. Keterbatasan Return Type dalam Overloading:

* Java tidak memperhatikan Return Type saat memilih metode Overloading yang tepat untuk dipanggil. Oleh karena itu, tidak diperbolehkan memiliki dua metode Overloading dengan nama dan parameter yang sama, tetapi hanya berbeda dalam Return Type. Ini akan menghasilkan kesalahan kompilasi.

Dalam makalah, Return Type dalam Overloading akan dibahas secara lebih mendalam, termasuk aturan yang perlu diperhatikan dan contoh implementasi. Hal ini akan memberikan pemahaman yang lebih baik tentang bagaimana Return Type berperan dalam Overloading dan pembatasannya dalam pemrograman Java.

# BAB IV

# SYARAT OVERLOADING

## **4.1 Nama Metode**

Nama Metode adalah salah satu elemen yang akan dibahas dalam makalah. Berikut adalah penjelasan mengenai Nama Metode dalam Overloading:

1. Definisi:

* Nama Metode mengacu pada identifier atau nama yang digunakan untuk mengidentifikasi suatu metode dalam bahasa pemrograman. Nama metode harus unik dalam konteks kelas yang sama.
* Dalam Overloading, Nama Metode dapat digunakan kembali untuk beberapa metode dengan syarat metode-metode tersebut memiliki parameter yang berbeda.

1. Peran Nama Metode dalam Overloading:

* Nama Metode digunakan untuk membedakan antara metode-metode yang memiliki parameter yang berbeda. Dalam Overloading, metode-metode tersebut memiliki nama yang sama tetapi memiliki parameter yang berbeda.
* Saat pemanggilan metode, Java menggunakan informasi dari nama metode untuk memilih metode Overloading yang tepat berdasarkan parameter yang diberikan.

1. Contoh Nama Metode dalam Overloading:

* Misalnya, terdapat dua metode Overloading dengan nama "calculateArea". Metode pertama menerima parameter int untuk menghitung luas persegi, sedangkan metode kedua menerima parameter double untuk menghitung luas lingkaran.
* Kedua metode ini dapat menggunakan kembali nama "calculateArea" karena mereka memiliki parameter yang berbeda, dan Java dapat membedakan antara keduanya berdasarkan tipe parameter yang diberikan saat pemanggilan metode.

1. Keterbatasan Nama Metode dalam Overloading:

* Overloading hanya memungkinkan penggunaan kembali nama metode jika parameter metode tersebut berbeda dalam tipe, urutan, atau jumlah. Jika parameter metode sama persis, maka itu akan dianggap duplikat dan menghasilkan kesalahan kompilasi.

Dalam makalah, Nama Metode dalam Overloading akan dibahas secara lebih detail, termasuk aturan dan contoh implementasi yang perlu diperhatikan. Penjelasan ini akan memberikan pemahaman yang lebih baik tentang bagaimana Nama Metode berperan dalam Overloading, serta batasan-batasannya dalam pemrograman Java.

## **4.2 Tipe Parameter**

Tipe Parameter adalah salah satu konsep yang akan dibahas dalam makalah. Berikut adalah penjelasan mengenai Tipe Parameter dalam Overloading:

1. Definisi:

* Tipe Parameter merujuk pada jenis data yang diterima oleh parameter suatu metode. Dalam bahasa pemrograman Java, setiap parameter harus memiliki tipe data yang ditentukan.
* Dalam Overloading, Tipe Parameter digunakan sebagai faktor yang membedakan antara metode-metode yang memiliki nama yang sama.

1. Peran Tipe Parameter dalam Overloading:

* Tipe Parameter digunakan untuk membedakan metode-metode yang memiliki nama yang sama tetapi menerima parameter dengan tipe data yang berbeda.
* Saat pemanggilan metode, Java menggunakan informasi tipe parameter untuk memilih metode Overloading yang tepat berdasarkan argumen yang diberikan.

1. Contoh Tipe Parameter dalam Overloading:

* Misalnya, terdapat dua metode Overloading dengan nama yang sama, "printData(int x)" dan "printData(String str)". Metode pertama menerima parameter bertipe int, sedangkan metode kedua menerima parameter bertipe String.
* Dalam pemanggilan metode "printData(10)", Java akan memilih metode "printData(int x)" karena argumen yang diberikan sesuai dengan tipe parameter yang diharapkan.
* Jika kita memanggil metode "printData("Hello")", Java akan memilih metode "printData(String str)" karena argumen yang diberikan sesuai dengan tipe parameter yang diharapkan.

1. Keterbatasan Tipe Parameter dalam Overloading:

* Overloading membutuhkan perbedaan dalam tipe parameter, tetapi tidak memperhatikan urutan atau nama parameter. Oleh karena itu, tidak diperbolehkan memiliki dua metode Overloading dengan urutan parameter yang sama tetapi tipe parameter yang berbeda, karena Java tidak dapat membedakan keduanya.

Dalam makalah, Tipe Parameter dalam Overloading akan dibahas secara lebih rinci, termasuk contoh implementasi dan aturan yang perlu diperhatikan saat menggunakan Tipe Parameter dalam Overloading. Hal ini akan memberikan pemahaman yang lebih baik tentang bagaimana Tipe Parameter berperan dalam Overloading dan bagaimana perbedaan tipe parameter memengaruhi pemilihan metode yang tepat saat pemanggilan.

## **4.3 Urutan Parameter**

Urutan Parameter adalah salah satu aspek yang akan dibahas dalam makalah. Berikut adalah penjelasan mengenai Urutan Parameter dalam Overloading:

1. Definisi:

* Urutan Parameter merujuk pada posisi atau susunan parameter dalam daftar parameter suatu metode. Ini mencerminkan urutan argumen yang diharapkan saat memanggil metode.
* Dalam Overloading, Urutan Parameter digunakan sebagai faktor yang membedakan antara metode-metode yang memiliki nama yang sama.

1. Peran Urutan Parameter dalam Overloading:

* Urutan Parameter digunakan untuk membedakan metode-metode yang memiliki nama yang sama tetapi memiliki urutan parameter yang berbeda.
* Saat pemanggilan metode, Java menggunakan informasi urutan parameter untuk memilih metode Overloading yang tepat berdasarkan argumen yang diberikan.

1. Contoh Urutan Parameter dalam Overloading:

* Misalnya, terdapat dua metode Overloading dengan nama yang sama, "calculateArea(int length, int width)" dan "calculateArea(int width, int length)". Metode pertama memiliki urutan parameter "length" diikuti oleh "width", sedangkan metode kedua memiliki urutan parameter "width" diikuti oleh "length".
* Saat memanggil metode "calculateArea(5, 10)", Java akan memilih metode "calculateArea(int length, int width)" karena urutan argumen sesuai dengan urutan parameter yang diharapkan.
* Jika kita memanggil metode "calculateArea(10, 5)", Java akan memilih metode "calculateArea(int width, int length)" karena urutan argumen sesuai dengan urutan parameter yang diharapkan.

1. Keterbatasan Urutan Parameter dalam Overloading:

* Overloading membutuhkan perbedaan dalam urutan parameter, tetapi tidak memperhatikan tipe atau nama parameter. Oleh karena itu, tidak diperbolehkan memiliki dua metode Overloading dengan tipe dan nama parameter yang sama tetapi urutan parameter yang berbeda, karena Java tidak dapat membedakan keduanya.

Dalam makalah, Urutan Parameter dalam Overloading akan dibahas secara lebih detail, termasuk contoh implementasi dan aturan yang perlu diperhatikan saat menggunakan Urutan Parameter dalam Overloading. Hal ini akan memberikan pemahaman yang lebih baik tentang bagaimana Urutan Parameter berperan dalam Overloading dan bagaimana perbedaan urutan parameter memengaruhi pemilihan metode yang tepat saat pemanggilan.

## **4.4 Jumlah Parameter**

Jumlah Parameter adalah salah satu aspek yang akan dibahas dalam makalah. Berikut adalah penjelasan mengenai Jumlah Parameter dalam Overloading:

1. Definisi:

* Jumlah Parameter mengacu pada jumlah parameter yang diperlukan oleh suatu metode. Parameter adalah nilai yang diterima oleh suatu metode dan digunakan dalam operasinya.
* Dalam Overloading, Jumlah Parameter digunakan sebagai faktor yang membedakan metode-metode yang memiliki nama yang sama.

1. Peran Jumlah Parameter dalam Overloading:

* Jumlah Parameter digunakan untuk membedakan metode-metode yang memiliki nama yang sama tetapi memiliki jumlah parameter yang berbeda.
* Saat pemanggilan metode, Java menggunakan informasi jumlah parameter untuk memilih metode Overloading yang tepat berdasarkan argumen yang diberikan.

1. Contoh Jumlah Parameter dalam Overloading:

* Misalnya, terdapat dua metode Overloading dengan nama yang sama, "calculateArea(int length)" dan "calculateArea(int length, int width)". Metode pertama memiliki satu parameter, yaitu "length", sedangkan metode kedua memiliki dua parameter, yaitu "length" dan "width".
* Saat memanggil metode "calculateArea(5)", Java akan memilih metode "calculateArea(int length)" karena jumlah argumen sesuai dengan jumlah parameter yang diharapkan.
* Jika kita memanggil metode "calculateArea(5, 10)", Java akan memilih metode "calculateArea(int length, int width)" karena jumlah argumen sesuai dengan jumlah parameter yang diharapkan.

1. Keterbatasan Jumlah Parameter dalam Overloading:

* Overloading membutuhkan perbedaan dalam jumlah parameter, tetapi tidak memperhatikan tipe atau urutan parameter. Oleh karena itu, tidak diperbolehkan memiliki dua metode Overloading dengan tipe dan urutan parameter yang sama tetapi jumlah parameter yang berbeda, karena Java tidak dapat membedakan keduanya.

Dalam makalah, Jumlah Parameter dalam Overloading akan dibahas secara lebih detail, termasuk contoh implementasi dan aturan yang perlu diperhatikan saat menggunakan Jumlah Parameter dalam Overloading. Hal ini akan memberikan pemahaman yang lebih baik tentang bagaimana Jumlah Parameter berperan dalam Overloading dan bagaimana perbedaan jumlah parameter memengaruhi pemilihan metode yang tepat saat pemanggilan.

# BAB V

# CONTOH IMPLEMENTASI

# OVERLOAING

## **5.1 Overloading pada Metode**

Overloading pada Metode adalah salah satu topik yang akan dibahas dalam makalah. Berikut adalah penjelasan mengenai Overloading pada Metode:

1. Definisi:

* Overloading pada Metode merujuk pada kemampuan untuk mendefinisikan beberapa metode dengan nama yang sama tetapi memiliki parameter yang berbeda dalam sebuah kelas.
* Dalam Overloading, metode-metode yang memiliki nama yang sama tetapi memiliki parameter yang berbeda dianggap sebagai metode Overloading.

1. Tujuan Overloading pada Metode:

* Overloading pada Metode memungkinkan pemrogram untuk menyediakan implementasi yang berbeda dari suatu fungsi atau operasi yang sama dengan mempertimbangkan jenis atau jumlah parameter yang berbeda.
* Tujuannya adalah memberikan fleksibilitas dan keterbacaan kode yang lebih baik serta menghindari pembuatan banyak metode dengan nama yang berbeda untuk operasi yang serupa.

1. Keuntungan Overloading pada Metode:

* Memudahkan pemrogram untuk menggunakan nama yang intuitif untuk metode-metode yang berbeda, tetapi berkaitan dengan tugas yang serupa.
* Meningkatkan keterbacaan dan pemahaman kode karena metode-metode yang memiliki tujuan serupa dikelompokkan dengan nama yang sama.
* Mengurangi kesalahan karena metode yang sesuai dapat dipilih secara otomatis berdasarkan jenis atau jumlah argumen yang diberikan saat pemanggilan metode.

1. Syarat Overloading pada Metode:

* Nama metode harus sama untuk metode-metode Overloading.
* Metode-metode Overloading harus memiliki parameter yang berbeda dalam hal tipe, urutan, atau jumlah parameter.
* Tipe return metode tidak diperhitungkan dalam pemilihan metode Overloading.

Berikut adalah contoh program Java yang menunjukkan penggunaan Overloading pada Metode:

public class Calculator {

public int add(int a, int b) {

return a + b;

}

public double add(double a, double b) {

return a + b;

}

public int add(int a, int b, int c) {

return a + b + c;

}

public static void main(String[] args) {

Calculator calculator = new Calculator();

int sum1 = calculator.add(5, 10);

System.out.println("Sum 1: " + sum1);

double sum2 = calculator.add(3.5, 2.5);

System.out.println("Sum 2: " + sum2);

int sum3 = calculator.add(2, 4, 6);

System.out.println("Sum 3: " + sum3);

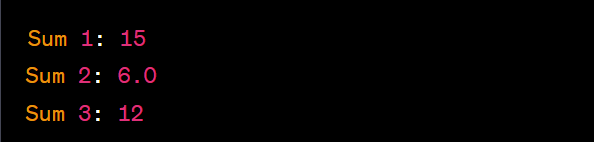
}

}

Penjelasan:

* Dalam contoh di atas, terdapat kelas **Calculator** yang memiliki beberapa metode **add** yang menggunakan Overloading.
* Metode pertama **add(int a, int b)** menerima dua parameter bertipe int dan mengembalikan hasil penjumlahan kedua parameter tersebut.
* Metode kedua **add(double a, double b)** menerima dua parameter bertipe double dan mengembalikan hasil penjumlahan kedua parameter tersebut.
* Metode ketiga **add(int a, int b, int c)** menerima tiga parameter bertipe int dan mengembalikan hasil penjumlahan ketiga parameter tersebut.
* Dalam metode **main**, objek **Calculator** dibuat dan beberapa pemanggilan metode **add** dilakukan dengan argumen yang berbeda sesuai dengan jumlah, tipe, dan urutan parameter.
* Hasil penjumlahan dari setiap pemanggilan metode ditampilkan ke layar.

Output :



Dalam contoh ini, Overloading pada metode **add** memungkinkan kita menggunakan metode yang sama dengan nama yang sama tetapi dengan parameter yang berbeda. Dalam pemanggilan metode, Java akan memilih metode yang tepat berdasarkan jenis dan jumlah argumen yang diberikan. Hal ini memberikan fleksibilitas dalam menggunakan metode yang serupa untuk tipe data yang berbeda tanpa perlu membuat banyak metode dengan nama yang berbeda.

## **5.2 Overloading pada Konstruktor**

Overloading pada Konstruktor adalah salah satu topik yang akan dibahas dalam makalah. Berikut adalah penjelasan mengenai Overloading pada Konstruktor:

1. Definisi:

* Overloading pada Konstruktor merujuk pada kemampuan untuk mendefinisikan beberapa konstruktor dengan nama yang sama tetapi memiliki parameter yang berbeda dalam sebuah kelas.
* Dalam Overloading, konstruktor-konstruktor yang memiliki nama yang sama tetapi memiliki parameter yang berbeda dianggap sebagai konstruktor Overloading.

1. Tujuan Overloading pada Konstruktor:

* Overloading pada Konstruktor memungkinkan pemrogram untuk membuat objek kelas dengan cara yang berbeda berdasarkan jenis, jumlah, atau urutan parameter yang diberikan saat pembuatan objek.
* Tujuannya adalah memberikan fleksibilitas dalam inisialisasi objek kelas dengan memberikan berbagai cara untuk mengatur nilainya.

1. Keuntungan Overloading pada Konstruktor:

* Memudahkan pemrogram untuk menggunakan konstruktor yang sesuai dengan kebutuhan mereka berdasarkan jenis, jumlah, atau urutan parameter yang diberikan saat pembuatan objek.
* Meningkatkan fleksibilitas dalam inisialisasi objek dengan memberikan berbagai cara untuk mengatur nilai awalnya.
* Menghindari pembuatan banyak konstruktor dengan nama yang berbeda untuk inisialisasi objek dengan tujuan yang serupa.

1. Syarat Overloading pada Konstruktor:

* Nama konstruktor harus sama untuk konstruktor-konstruktor Overloading.
* Konstruktor-konstruktor Overloading harus memiliki parameter yang berbeda dalam hal tipe, urutan, atau jumlah parameter.

contoh program Java yang menunjukkan penggunaan Overloading pada Konstruktor:

public class Rectangle {

private int length;

private int width;

public Rectangle() {

length = 0;

width = 0;

}

public Rectangle(int side) {

length = side;

width = side;

}

public Rectangle(int length, int width) {

this.length = length;

this.width = width;

}

public int getArea() {

return length \* width;

}

public static void main(String[] args) {

Rectangle rectangle1 = new Rectangle();

System.out.println("Area of rectangle1: " + rectangle1.getArea());

Rectangle rectangle2 = new Rectangle(5);

System.out.println("Area of rectangle2: " + rectangle2.getArea());

Rectangle rectangle3 = new Rectangle(4, 6);

System.out.println("Area of rectangle3: " + rectangle3.getArea());

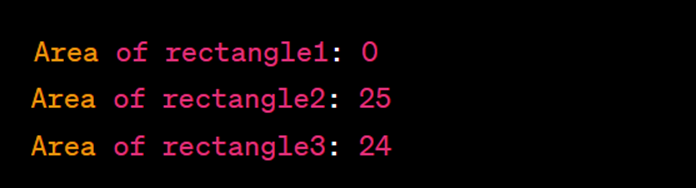
}

}

Penjelasan:

* Dalam contoh di atas, terdapat kelas **Rectangle** yang memiliki tiga konstruktor yang menggunakan Overloading.
* Konstruktor pertama **Rectangle()** tidak memiliki parameter dan menginisialisasi panjang (**length**) dan lebar (**width**) dengan nilai 0.
* Konstruktor kedua **Rectangle(int side)** menerima satu parameter **side** yang digunakan untuk menginisialisasi panjang (**length**) dan lebar (**width**) dengan nilai yang sama.
* Konstruktor ketiga **Rectangle(int length, int width)** menerima dua parameter **length** dan **width** yang digunakan untuk menginisialisasi panjang (**length**) dan lebar (**width**) dengan nilai yang berbeda.
* Metode **getArea()** menghitung dan mengembalikan luas persegi panjang berdasarkan panjang (**length**) dan lebar (**width**) yang ditentukan.
* Dalam metode **main**, tiga objek **Rectangle** dibuat menggunakan konstruktor Overloading yang berbeda, dan luas masing-masing persegi panjang ditampilkan.

Output:



Dalam contoh ini, Overloading pada konstruktor **Rectangle** memungkinkan kita untuk membuat objek **Rectangle** dengan cara yang berbeda berdasarkan jenis dan jumlah parameter yang diberikan saat pembuatan objek. Hal ini memberikan fleksibilitas dalam inisialisasi objek dengan memberikan berbagai cara untuk mengatur nilai awalnya.

# BAB VI

# PEMECAHAN MASALAH DENGAN

# OVERLOADING

## **6.1 Studi Kasus 1: Menghitung Luas Bangun Datar**

Dalam studi kasus ini, kita akan membuat program untuk menghitung luas beberapa bangun datar, seperti persegi, persegi panjang, dan lingkaran. Program ini akan mengimplementasikan konsep Overloading pada Metode.

public class ShapeAreaCalculator {

// Menghitung luas persegi

public double calculateArea(double side) {

return side \* side;

}

// Menghitung luas persegi panjang

public double calculateArea(double length, double width) {

return length \* width;

}

// Menghitung luas lingkaran

public double calculateArea(double radius) {

return Math.PI \* radius \* radius;

}

public static void main(String[] args) {

ShapeAreaCalculator calculator = new ShapeAreaCalculator();

// Menghitung luas persegi dengan sisi 5

double squareArea = calculator.calculateArea(5);

System.out.println("Luas persegi: " + squareArea);

// Menghitung luas persegi panjang dengan panjang 4 dan lebar 6

double rectangleArea = calculator.calculateArea(4, 6);

System.out.println("Luas persegi panjang: " + rectangleArea);

// Menghitung luas lingkaran dengan jari-jari 3

double circleArea = calculator.calculateArea(3);

System.out.println("Luas lingkaran: " + circleArea);

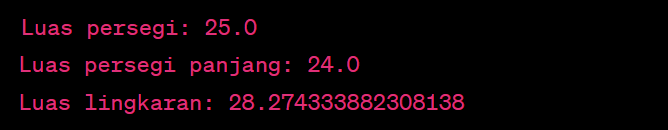
}

}

Penjelasan:

* Dalam program ini, terdapat kelas **ShapeAreaCalculator** yang memiliki tiga metode **calculateArea** yang menggunakan Overloading.
* Metode pertama **calculateArea(double side)** menghitung luas persegi berdasarkan sisi yang diberikan.
* Metode kedua **calculateArea(double length, double width)** menghitung luas persegi panjang berdasarkan panjang dan lebar yang diberikan.
* Metode ketiga **calculateArea(double radius)** menghitung luas lingkaran berdasarkan jari-jari yang diberikan.
* Dalam metode **main**, objek **ShapeAreaCalculator** dibuat dan beberapa pemanggilan metode **calculateArea** dilakukan untuk menghitung luas bangun datar.
* Hasil perhitungan luas dari setiap pemanggilan metode ditampilkan ke layar.

Output:



Dalam contoh ini, Overloading pada metode **calculateArea** memungkinkan kita menggunakan metode yang sama dengan nama yang sama tetapi dengan parameter yang berbeda. Dalam pemanggilan metode, Java akan memilih metode yang tepat berdasarkan jenis dan jumlah argumen yang diberikan. Hal ini memberikan fleksibilitas dalam menghitung luas berbagai bangun datar tanpa perlu membuat banyak metode dengan nama yang berbeda.

## **6.2 Studi Kasus 2: Operasi Matematika**

Dalam studi kasus ini, kita akan membuat program untuk melakukan operasi matematika dasar, seperti penjumlahan, pengurangan, perkalian, dan pembagian. Program ini akan mengimplementasikan konsep Overloading pada Metode.

public class MathOperations {

// Penjumlahan dua angka integer

public int add(int a, int b) {

return a + b;

}

// Penjumlahan dua angka double

public double add(double a, double b) {

return a + b;

}

// Pengurangan dua angka integer

public int subtract(int a, int b) {

return a - b;

}

// Pengurangan dua angka double

public double subtract(double a, double b) {

return a - b;

}

// Perkalian dua angka integer

public int multiply(int a, int b) {

return a \* b;

}

// Perkalian dua angka double

public double multiply(double a, double b) {

return a \* b;

}

// Pembagian dua angka integer

public int divide(int a, int b) {

return a / b;

}

// Pembagian dua angka double

public double divide(double a, double b) {

return a / b;

}

public static void main(String[] args) {

MathOperations calculator = new MathOperations();

// Penjumlahan

int sum1 = calculator.add(5, 10);

System.out.println("Penjumlahan 1: " + sum1);

double sum2 = calculator.add(3.5, 2.5);

System.out.println("Penjumlahan 2: " + sum2);

// Pengurangan

int difference1 = calculator.subtract(8, 4);

System.out.println("Pengurangan 1: " + difference1);

double difference2 = calculator.subtract(7.5, 3.2);

System.out.println("Pengurangan 2: " + difference2);

// Perkalian

int product1 = calculator.multiply(3, 6);

System.out.println("Perkalian 1: " + product1);

double product2 = calculator.multiply(2.5, 4.8);

System.out.println("Perkalian 2: " + product2);

// Pembagian

int quotient1 = calculator.divide(20, 5);

System.out.println("Pembagian 1: " + quotient1);

double quotient2 = calculator.divide(15.6, 3.2);

System.out.println("Pembagian 2: " + quotient2);

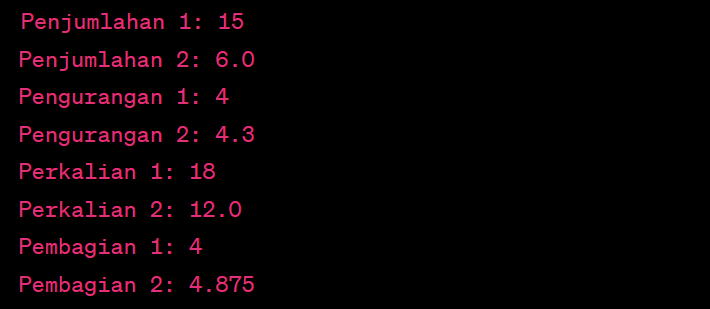
}

}

Penjelasan:

* Dalam program ini, terdapat kelas **MathOperations** yang memiliki beberapa metode untuk melakukan operasi matematika dasar.
* Metode **add** digunakan untuk penjumlahan, dengan versi Overloading untuk angka integer dan double.
* Metode **subtract** digunakan untuk pengurangan, juga dengan versi Overloading untuk angka integer dan double.
* Metode **multiply** digunakan untuk perkalian, juga dengan versi Overloading untuk angka integer dan double.
* Metode **divide** digunakan untuk pembagian, juga dengan versi Overloading untuk angka integer dan double.
* Dalam metode **main**, objek **MathOperations** dibuat dan beberapa pemanggilan metode dilakukan untuk melakukan operasi matematika.
* Hasil dari setiap operasi matematika ditampilkan ke layar.

Output:



Dalam contoh ini, Overloading pada metode **add**, **subtract**, **multiply**, dan **divide** memungkinkan kita menggunakan metode yang sama dengan nama yang sama tetapi dengan parameter yang berbeda. Dalam pemanggilan metode, Java akan memilih metode yang tepat berdasarkan jenis dan jumlah argumen yang diberikan. Hal ini memberikan fleksibilitas dalam melakukan operasi matematika dengan berbagai jenis angka tanpa perlu membuat banyak metode dengan nama yang berbeda.

# KESIMPULAN

Berdasarkan daftar isi yang telah disebutkan sebelumnya, berikut adalah kesimpulan yang dapat diambil:

1. Overloading adalah konsep dalam bahasa pemrograman Java yang memungkinkan kita untuk memiliki beberapa metode dengan nama yang sama tetapi dengan parameter yang berbeda.
2. Overloading dapat digunakan pada metode maupun konstruktor. Pada metode, kita dapat menggunakan Overloading dengan mengubah jumlah, tipe data, atau urutan parameter. Pada konstruktor, Overloading memungkinkan kita membuat beberapa konstruktor dengan parameter yang berbeda.
3. Overloading memiliki beberapa keuntungan, antara lain:

a. Memungkinkan kita untuk menggunakan nama metode yang sama untuk tugas yang berbeda, membuat kode lebih intuitif dan mudah dibaca.

b. Memungkinkan kita untuk menghindari pembuatan metode dengan nama yang berbeda hanya karena perbedaan parameter.

c. Meningkatkan fleksibilitas dalam penggunaan metode, karena kita dapat menggunakannya dengan berbagai jenis dan jumlah argumen.

1. Overloading berbeda dengan Overriding. Overloading terjadi di dalam kelas yang sama, sedangkan Overriding terjadi antara kelas induk dan kelas turunannya. Overloading berkaitan dengan metode, sedangkan Overriding berkaitan dengan substitusi metode dalam pewarisan kelas.

Dalam contoh kode program yang diberikan, telah diilustrasikan penggunaan Overloading pada metode untuk menghitung luas bangun datar dan melakukan operasi matematika dasar. Dengan menggunakan Overloading, kita dapat memiliki metode dengan nama yang sama tetapi dapat menerima parameter yang berbeda, sehingga memberikan fleksibilitas dan kesederhanaan dalam kode.

Penerapan Overloading dapat membuat kode lebih bersih, intuitif, dan mudah dimengerti. Hal ini memungkinkan programmer untuk lebih efisien dalam menulis kode, mengurangi duplikasi, dan meningkatkan fleksibilitas dalam penggunaan metode.