Modul 7. Method

Tugas Pendahuluan :

1. Apa itu method dalam bahasa pemrograman Java?
2. Jelaskan pentingnya method dalam pembuatan program.
3. Bagaimana cara mendefinisikan method dalam Java? Berikan contoh sederhana.
4. Apa itu "signature" method?
5. Apa perbedaan antara parameter dan argumen dalam konteks method?
6. Jelaskan tentang method overloading dan method overriding dalam Java.
7. Bagaimana cara mengembalikan nilai dari sebuah method dalam Java?
8. Apa itu "void" dalam konteks return type method?
9. Bagaimana cara menggunakan "access modifier" untuk mengatur aksesibilitas method?
10. Apa yang dimaksud dengan "recursion" dalam method? Berikan contoh sederhana.

Materi :

Dalam bahasa pemrograman Java, "Methods" (metode) adalah blok kode yang berisi serangkaian pernyataan yang dapat dieksekusi oleh program. Metode digunakan untuk mengelompokkan tindakan tertentu menjadi sebuah unit yang dapat dipanggil atau dipakai berulang kali. Setiap metode memiliki nama, tipe kembalian (return type), dan serangkaian parameter yang dapat menerima input dari pemanggil metode.

Tipe kembalian adalah tipe data yang menunjukkan nilai apa yang akan dikembalikan oleh metode setelah dieksekusi. Jika metode tidak mengembalikan nilai apa pun, tipe kembalian yang digunakan adalah "void".

Contoh deklarasi metode dalam Java:

```
// Metode tanpa parameter dan tipe kembalian void
public void sayHello() {
    System.out.println("Hello!");
}

// Metode dengan parameter dan tipe kembalian int
public int addNumbers(int a, int b) {
    int sum = a + b;
    return sum;
}

// Metode dengan parameter dan tipe kembalian String
public String getFullName(String firstName, String lastName) {
    String fullName = firstName + " " + lastName;
    return fullName;
}
```

Contoh pemanggilan metode dalam Java:

```
1. public class Example {
2.     public static void main(String[] args) {
3.         Example example = new Example();
4.
5.         // Memanggil metode sayHello()
6.         example.sayHello();
7.
8.         // Memanggil metode addNumbers() dan menampilkan hasilnya
9.         int result = example.addNumbers(10, 5);
10.        System.out.println("Hasil penjumlahan: " + result);
11.
12.        // Memanggil metode getFullName() dan menampilkan hasilnya
13.        String fullName = example.getFullName("John", "Doe");
14.        System.out.println("Nama lengkap: " + fullName);
15.    }
16.
17.    // Metode sayHello()
18.    public void sayHello() {
19.        System.out.println("Hello!");
20.    }
21.
22.    // Metode addNumbers()
23.    public int addNumbers(int a, int b) {
24.        int sum = a + b;
25.        return sum;
26.    }
27.
28.    // Metode getFullName()
29.    public String getFullName(String firstName, String lastName) {
30.        String fullName = firstName + " " + lastName;
31.        return fullName;
32.    }
33. }
34.
```

Output:

```
Hello!
Hasil penjumlahan: 15
Nama lengkap: John Doe
```

Dalam contoh di atas, kita mendeklarasikan tiga metode, yaitu sayHello(), addNumbers(), dan getFullName(). Metode sayHello() tidak memiliki parameter dan tidak mengembalikan nilai, sehingga tipe kembaliannya adalah "void". Metode addNumbers() memiliki dua parameter bertipe int dan mengembalikan hasil penjumlahan dari kedua parameter tersebut. Metode getFullName() memiliki dua parameter bertipe String dan mengembalikan hasil gabungan dari kedua parameter tersebut.

Selain metode-metode yang sudah didefinisikan oleh Java, kita juga dapat membuat metode-metode kustom sesuai dengan kebutuhan program yang sedang kita bangun. Metode memungkinkan kita untuk mengatur kode program dengan lebih terstruktur, reusable, dan mudah dipahami.

1. Komponen Method

Dalam bahasa pemrograman Java, komponen-komponen (elemen-elemen) sebuah metode (method) terdiri dari beberapa bagian yang harus didefinisikan secara lengkap saat kita mendeklarasikan metode. Berikut adalah komponen-komponen utama dalam sebuah metode:

1) Nama Metode (Method Name): Nama metode adalah identitas unik yang digunakan untuk memanggil atau mempergunakan metode. Nama metode harus sesuai dengan aturan penamaan Java dan harus diawali dengan huruf atau underscore (_). Nama metode biasanya menggambarkan tindakan atau fungsi yang dilakukan oleh metode tersebut.
2) Parameter (Parameters): Parameter adalah nilai yang diterima oleh metode saat dipanggil. Parameter didefinisikan dalam tanda kurung setelah nama metode dan digunakan untuk menerima input dari pemanggil metode. Jika metode tidak memerlukan input, tanda kurung tetap diberikan tanpa parameter di dalamnya. Parameter memungkinkan metode untuk menerima data yang dapat digunakan dalam proses eksekusi.
3) Tipe Kembalian (Return Type): Tipe kembalian adalah tipe data yang menentukan jenis nilai yang dikembalikan oleh metode setelah proses eksekusi selesai. Jika metode tidak mengembalikan nilai apa pun, tipe kembalian yang digunakan adalah "void". Jika metode mengembalikan nilai, tipe kembaliannya harus sesuai dengan tipe data yang diharapkan.
4) Blok Kode (Code Block): Blok kode merupakan area yang berisi serangkaian pernyataan yang akan dieksekusi oleh metode. Pernyataan-pernyataan ini menggambarkan logika atau tindakan yang akan dilakukan oleh metode. Blok kode diawali dengan tanda kurung kurawal buka { dan diakhiri dengan tanda kurung kurawal tutup }.

Contoh definisi metode dengan komponen-komponennya:

```
1. public class Example {
2.     // Nama metode: sayHello
3.     // Parameter: tidak ada
4.     // Tipe Kembalian: void (tidak mengembalikan nilai)
5.     public void sayHello() {
6.         System.out.println("Hello!");
7.     }
8.
9.     // Nama metode: addNumbers
10.    // Parameter: dua buah int (a dan b)
11.    // Tipe Kembalian: int (mengembalikan hasil penjumlahan a dan b)
12.    public int addNumbers(int a, int b) {
13.        int sum = a + b;
14.        return sum;
15.    }
16.
17.    // Nama metode: getFullName
18.    // Parameter: dua buah String (firstName dan lastName)
19.    // Tipe Kembalian: String (mengembalikan hasil gabungan firstName dan lastName)
20.    public String getFullName(String firstName, String lastName) {
```

```
21.         String fullName = firstName + " " + lastName;
22.         return fullName;
23.     }
24. }
25.
```

Dalam contoh di atas, kita memiliki tiga metode dengan komponen-komponennya masing-masing. Metode sayHello tidak memiliki parameter dan tidak mengembalikan nilai (void). Metode addNumbers menerima dua parameter bertipe int dan mengembalikan hasil penjumlahan dari kedua parameter tersebut. Metode getFullName menerima dua parameter bertipe String dan mengembalikan hasil gabungan dari kedua parameter tersebut.

Dengan adanya komponen-komponen tersebut, kita dapat mendefinisikan dan menggunakan metode dengan lebih terstruktur dan mudah dipahami dalam pengembangan aplikasi Java.

2. Void

Dalam bahasa pemrograman Java, kata kunci void digunakan sebagai tipe kembalian (return type) dari sebuah metode. Ketika sebuah metode dideklarasikan dengan tipe kembalian void, itu berarti metode tersebut tidak mengembalikan nilai apapun setelah proses eksekusinya selesai.

Jika sebuah metode memiliki tipe kembalian void, maka ketika metode tersebut dipanggil, hasil dari metode tersebut tidak dapat digunakan atau disimpan dalam variabel. Metode dengan tipe kembalian void biasanya digunakan untuk melakukan tindakan atau operasi tertentu tanpa perlu menghasilkan nilai balik.

Contoh metode dengan tipe kembalian void:

```
 1. public class Example {
 2.     // Metode dengan tipe kembalian void
 3.     public void sayHello() {
 4.         System.out.println("Hello!");
 5.     }
 6.
 7.     // Metode untuk mencetak pesan tertentu
 8.     public void printMessage(String message) {
 9.         System.out.println(message);
10.     }
11.
12.     // Metode untuk melakukan perubahan pada data
13.     public void updateData(int data) {
14.         // Melakukan perubahan pada data
15.         data = data * 2;
16.         System.out.println("Data setelah diperbarui: " + data);
17.     }
18. }
19.
```

Dalam contoh di atas, kita memiliki beberapa metode dengan tipe kembalian void. Metode sayHello() digunakan untuk mencetak pesan "Hello!" ke konsol. Metode printMessage(String message) digunakan untuk mencetak pesan yang diberikan oleh pengguna melalui argumen message. Metode

updateData(int data) digunakan untuk mengubah nilai variabel data dengan mengalikannya dengan 2 dan mencetak hasil perubahannya.

Ketika kita menggunakan metode dengan tipe kembalian void, kita tidak dapat menyimpan nilai yang dikembalikan oleh metode tersebut ke dalam variabel. Namun, kita masih dapat menggunakan metode tersebut untuk melakukan tindakan atau operasi yang diinginkan dalam program kita.

3. Access modifier

"Access modifier" atau modifier akses dalam Java digunakan untuk mengatur tingkat aksesibilitas atau visibilitas suatu method, field, atau class. Dengan menggunakan access modifier, kita dapat mengontrol bagaimana elemen-elemen tersebut dapat diakses oleh kode di luar kelas tempat elemen tersebut dideklarasikan. Ada empat jenis access modifier dalam Java:

1) public: Method dengan access modifier public dapat diakses dari mana saja, baik dari dalam kelas itu sendiri maupun dari kelas lain yang berada di package yang berbeda.
2) protected: Method dengan access modifier protected dapat diakses dari dalam kelas itu sendiri, dari kelas turunan (subclass) dari kelas tersebut, atau dari kelas yang berada dalam package yang sama.
3) default (tidak ada access modifier): Jika sebuah method tidak memiliki access modifier yang ditentukan, maka secara default method tersebut akan memiliki access modifier default. Method dengan access modifier default hanya dapat diakses dari kelas-kelas yang berada dalam package yang sama.
4) private: Method dengan access modifier private hanya dapat diakses dari dalam kelas itu sendiri. Method ini tidak dapat diakses oleh kelas lain, bahkan kelas turunan dari kelas tersebut.

Contoh penggunaan access modifier dalam method:

```
1. public class MyClass {
2.     // Method dengan access modifier public
3.     public void publicMethod() {
4.         // Isi method
5.     }
6.     // Method dengan access modifier protected
7.     protected void protectedMethod() {
8.         // Isi method
9.     }
10.    // Method dengan access modifier default
11.    void defaultMethod() {
12.        // Isi method
13.    }
14.    // Method dengan access modifier private
15.    private void privateMethod() {
16.        // Isi method
17.    }
18. }
19.
```

Dalam contoh di atas, MyClass memiliki empat method dengan berbagai access modifier. Method publicMethod() dapat diakses dari mana saja, method protectedMethod() dapat diakses dari kelas

turunan dan package yang sama, method defaultMethod() hanya dapat diakses dari package yang sama, dan method privateMethod() hanya dapat diakses dari dalam kelas MyClass itu sendiri.

4. Parameter

Dalam bahasa pemrograman Java, "parameter" adalah variabel yang digunakan dalam deklarasi metode dan berfungsi untuk menerima input dari pemanggil metode. Ketika sebuah metode dipanggil, nilai-nilai yang dikirimkan sebagai argumen akan disimpan di dalam parameter dan dapat digunakan dalam proses eksekusi metode.

Parameter memungkinkan kita untuk memberikan nilai dari luar kepada metode, sehingga metode dapat melakukan tindakan atau perhitungan berdasarkan nilai-nilai yang diberikan. Dengan menggunakan parameter, metode dapat menjadi lebih fleksibel dan dapat digunakan berulang kali dengan input yang berbeda.

Contoh deklarasi metode dengan parameter:

```
1. public class Example {
2.     // Metode dengan satu parameter bertipe int
3.     public void printNumber(int number) {
4.         System.out.println("Angka yang diterima: " + number);
5.     }
6.
7.     // Metode dengan dua parameter bertipe String
8.     public void printFullName(String firstName, String lastName) {
9.         System.out.println("Nama lengkap: " + firstName + " " + lastName);
10.    }
11.
12.    // Metode dengan parameter bertipe double dan int
13.    public double calculateTotalPrice(double price, int quantity) {
14.        double totalPrice = price * quantity;
15.        return totalPrice;
16.    }
17. }
18.
```

Contoh pemanggilan metode dengan parameter:

```
1. public class Main {
2.     public static void main(String[] args) {
3.         Example example = new Example();
4.
5.         // Memanggil metode printNumber() dengan argumen 10
6.         example.printNumber(10);
7.
8.         // Memanggil metode printFullName() dengan argumen "John" dan "Doe"
9.         example.printFullName("John", "Doe");
10.
11.        // Memanggil metode calculateTotalPrice() dengan argumen 25.5 dan 3
12.        double total = example.calculateTotalPrice(25.5, 3);
13.        System.out.println("Total harga: " + total);
14.    }
15. }
```

```
16.
```

Output:

```
Angka yang diterima: 10
Nama lengkap: John Doe
Total harga: 76.5
```

Dalam contoh di atas, kita memiliki beberapa metode dengan parameter. Metode printNumber() menerima satu parameter bertipe int, printFullName() menerima dua parameter bertipe String, dan calculateTotalPrice() menerima dua parameter, satu bertipe double dan satu bertipe int.

Ketika metode dipanggil, kita mengirimkan nilai-nilai sebagai argumen yang akan ditampung dalam parameter metode. Kemudian, nilai-nilai ini dapat digunakan dalam tubuh metode untuk melakukan tindakan atau perhitungan yang diinginkan.

Penting untuk memastikan bahwa jumlah, tipe, dan urutan argumen yang dikirimkan saat memanggil metode sesuai dengan definisi parameter dalam deklarasi metode.

5. Method Recursion

Rekursi (recursion) adalah konsep dalam pemrograman di mana sebuah fungsi atau method dapat memanggil dirinya sendiri secara berulang sampai mencapai kondisi berhenti (base case). Rekursi digunakan untuk memecahkan masalah yang dapat dipecahkan menjadi beberapa submasalah yang serupa dengan masalah aslinya. Untuk mengimplementasikan rekursi dalam pemrograman Java, beberapa hal harus dipertimbangkan:

1) Base Case: Setiap fungsi rekursif harus memiliki kondisi berhenti, yang disebut base case. Base case menentukan kapan rekursi harus berhenti dan fungsi tidak lagi memanggil dirinya sendiri. Jika base case tidak ada, fungsi akan terus memanggil dirinya dalam bentuk yang tak terbatas, yang menyebabkan terjadinya "stack overflow".
2) Pemanggilan Diri Sendiri: Fungsi rekursif memanggil dirinya sendiri untuk memecahkan masalah yang lebih kecil.
3) Permasalahan yang Dipecahkan: Pastikan bahwa setiap panggilan rekursif mendekati base case sehingga masalah menjadi lebih kecil setiap kali fungsi dipanggil.

Contoh implementasi rekursi dalam Java adalah mencari faktorial dari suatu bilangan:

```
1. public class RecursionExample {
 2.     // Method rekursif untuk menghitung faktorial
 3.     public static int factorial(int n) {
 4.         // Base case: jika n = 0 atau n = 1, faktorialnya adalah 1
 5.         if (n == 0 || n == 1) {
 6.             return 1;
 7.         } else {
 8.             // Panggil diri sendiri untuk permasalahan yang lebih kecil
 9.             return n * factorial(n - 1);
10.         }
11.     }
12.
```

```
    public static void main(String[] args) {
        int number = 5;
        int result = factorial(number);
        System.out.println("Faktorial dari " + number + " adalah: " + result);
    }
}
```

Dalam contoh di atas, method factorial adalah sebuah method rekursif yang menghitung faktorial dari suatu bilangan n. Ketika method dipanggil dengan n bernilai 5, maka method tersebut akan memanggil dirinya sendiri dengan nilai yang semakin kecil (misalnya, 4, 3, 2, dan seterusnya) hingga mencapai base case (n = 0 atau n = 1). Setelah mencapai base case, method akan mengembalikan hasil faktorialnya secara berurutan hingga mencapai panggilan pertama. Hasil akhir adalah faktorial dari bilangan awal yang diinginkan.

6. Overloading

Method Overloading (Pemuatan Metode) adalah konsep dalam pemrograman Java di mana kita dapat memiliki beberapa metode dengan nama yang sama dalam sebuah kelas, namun memiliki parameter yang berbeda. Dalam method overloading, metode-metode tersebut memiliki nama yang sama, tetapi jumlah atau tipe parameter yang berbeda.

Keuntungan utama dari method overloading adalah kita dapat menggunakan nama metode yang intuitif dan mudah diingat untuk berbagai kasus atau tipe data yang berbeda, tanpa perlu menggunakan nama metode yang berbeda setiap kali kita ingin melakukan tindakan yang mirip atau serupa.

Contoh method overloading:

```
public class Example {
    // Metode dengan satu parameter bertipe int
    public void printNumber(int number) {
        System.out.println("Angka yang diterima: " + number);
    }

    // Metode dengan satu parameter bertipe double
    public void printNumber(double number) {
        System.out.println("Angka desimal yang diterima: " + number);
    }

    // Metode dengan dua parameter bertipe String
    public void printFullName(String firstName, String lastName) {
        System.out.println("Nama lengkap: " + firstName + " " + lastName);
    }

    // Metode dengan tiga parameter bertipe String
    public void printFullName(String firstName, String middleName, String lastName) {
        System.out.println("Nama lengkap: " + firstName + " " + middleName + " " + lastName);
    }
}
```

Contoh pemanggilan method overloading:

```
1. public class Main {
2.     public static void main(String[] args) {
3.         Example example = new Example();
4.
5.         // Memanggil metode printNumber() dengan argumen int
6.         example.printNumber(10);
7.
8.         // Memanggil metode printNumber() dengan argumen double
9.         example.printNumber(3.14);
10.
11.        // Memanggil metode printFullName() dengan dua argumen String
12.        example.printFullName("John", "Doe");
13.
14.        // Memanggil metode printFullName() dengan tiga argumen String
15.        example.printFullName("John", "Middle", "Doe");
16.    }
17. }
18.
```

Output:

```
Angka yang diterima: 10
Angka desimal yang diterima: 3.14
Nama lengkap: John Doe
Nama lengkap: John Middle Doe
```

Dalam contoh di atas, kita memiliki beberapa metode dengan nama yang sama, yaitu printNumber() dan printFullName(), namun memiliki parameter-parameter yang berbeda. Ada dua metode printNumber(), satu menerima parameter bertipe int dan yang lain menerima parameter bertipe double. Demikian pula, ada dua metode printFullName(), satu menerima dua parameter bertipe String dan yang lain menerima tiga parameter bertipe String.

Ketika kita memanggil metode, Java akan memilih metode yang paling sesuai berdasarkan jumlah dan tipe parameter yang cocok dengan argumen yang diberikan. Method overloading memungkinkan kita untuk membuat kode yang lebih fleksibel dan mudah digunakan karena kita dapat menggunakan nama metode yang sama untuk kasus-kasus yang berbeda.

7. Overriding

Method overriding adalah konsep dalam pemrograman berorientasi objek di mana subclass menyediakan implementasi ulang (override) dari sebuah method yang telah didefinisikan di superclass. Ketika sebuah subclass mengoverride method dari superclass, artinya subclass menyediakan definisi baru untuk method tersebut dengan nama, tipe kembalian, dan parameter yang sama.

Penting untuk diingat bahwa method overriding hanya berlaku jika method tersebut merupakan method dari superclass dan subclass tersebut memiliki relasi inheritance dengan superclass. Method yang dioverride di subclass harus memiliki aksesibilitas yang sama atau lebih luas daripada method di superclass (misalnya, jika method di superclass memiliki akses modifier protected, maka method overriding di subclass harus memiliki akses modifier protected atau public).

Contoh method overriding:

```
 1. class Animal {
 2.     public void sound() {
 3.         System.out.println("Suara hewan...");
 4.     }
 5. }
 6.
 7. class Cat extends Animal {
 8.      // Method overriding, subclass Cat mengganti implementasi method sound() dari superclass
Animal
 9.     @Override
10.     public void sound() {
11.         System.out.println("Meow!");
12.     }
13. }
14.
15. class Dog extends Animal {
16.      // Method overriding, subclass Dog mengganti implementasi method sound() dari superclass
Animal
17.     @Override
18.     public void sound() {
19.         System.out.println("Woof!");
20.     }
21. }
22.
23. public class Main {
24.     public static void main(String[] args) {
25.         Animal animal1 = new Animal();
26.         Animal animal2 = new Cat(); // Polimorfisme, objek Cat diassign ke variabel tipe Animal
27.         Animal animal3 = new Dog(); // Polimorfisme, objek Dog diassign ke variabel tipe Animal
28.
29.         animal1.sound(); // Output: Suara hewan...
30.         animal2.sound(); // Output: Meow! (Method overriding di Cat dipanggil)
31.         animal3.sound(); // Output: Woof! (Method overriding di Dog dipanggil)
32.     }
33. }
34.
```

Dalam contoh di atas, terdapat superclass Animal dan dua subclass, yaitu Cat dan Dog. Subclass Cat dan Dog melakukan method overriding untuk method sound() dari superclass Animal. Ketika method sound() dipanggil pada objek dari masing-masing kelas, hasil yang dihasilkan akan berbeda berdasarkan implementasi yang dioverride di masing-masing subclass.

8. Signature

"Signature" method mengacu pada kombinasi dari nama method dan tipe serta urutan parameter yang dimilikinya. Dalam bahasa pemrograman Java, "signature" method digunakan untuk mengidentifikasi metode tertentu secara unik dalam suatu kelas. "Signature" method melibatkan nama method beserta tipe dan urutan parameter yang digunakan dalam deklarasi method.

Sebagai contoh, pertimbangkan dua method berikut:

```
// Method pertama dengan nama printMessage dan parameter bertipe String
public void printMessage(String message) {
```

```
    System.out.println(message);
}

// Method kedua dengan nama printMessage dan parameter bertipe int
public void printMessage(int number) {
    System.out.println("Number: " + number);
}
```

Kedua method memiliki nama yang sama, yaitu printMessage, tetapi "signature" method-nya berbeda karena parameter yang digunakan memiliki tipe yang berbeda. Oleh karena itu, Java memperlakukan kedua method ini sebagai dua metode yang berbeda.

Dalam "signature" method, hal-hal berikut dianggap sebagai bagian dari identifikasi unik metode:

1) Nama method.
2) Tipe dan urutan parameter yang digunakan.

Hal-hal berikut tidak dianggap sebagai bagian dari "signature" method:

1) Tipe kembalian method.
2) Modifiers (misalnya, public, private, dll.).

Dengan menggunakan "signature" method, kita dapat mengimplementasikan metode overloading di mana kita memiliki dua atau lebih method dengan nama yang sama, tetapi memiliki parameter yang berbeda sehingga metode tersebut dapat menerima tipe data yang berbeda dan diakses secara unik.

9. Command-Line Arguments

Command-Line Arguments (Argumen Baris Perintah) adalah input yang diberikan kepada sebuah program saat program dijalankan melalui baris perintah (command-line) pada sistem operasi. Ketika sebuah program dieksekusi melalui baris perintah, kita dapat menyertakan argumen tambahan setelah nama program, dan argumen-argumen ini akan diteruskan ke program sebagai input.

Dalam bahasa pemrograman Java, argumen baris perintah dapat diakses melalui parameter String[] args dalam metode main(). args adalah sebuah array dari tipe data String, dan setiap elemen dalam array ini berisi argumen yang diberikan pada baris perintah.

Contoh penggunaan argumen baris perintah dalam Java:

```
 1. public class CommandLineArgumentsExample {
 2.     public static void main(String[] args) {
 3.         if (args.length > 0) {
 4.             System.out.println("Argumen yang diberikan:");
 5.             for (String arg : args) {
 6.                 System.out.println(arg);
 7.             }
 8.         } else {
 9.             System.out.println("Tidak ada argumen yang diberikan.");
10.         }
11.     }
12. }
13.
```

Contoh pemanggilan program melalui baris perintah:

```
java CommandLineArgumentsExample arg1 arg2 arg3
```

Output:

```
Argumen yang diberikan:
arg1
arg2
arg3
```

Dalam contoh di atas, kita membuat program CommandLineArgumentsExample yang menerima argumen dari baris perintah. Saat program dijalankan dengan argumen arg1, arg2, dan arg3, program akan menampilkan argumen-argumen tersebut sebagai hasil eksekusi. Jika tidak ada argumen yang diberikan saat menjalankan program, program akan memberikan pesan "Tidak ada argumen yang diberikan."

Argumen baris perintah sering digunakan untuk memberikan konfigurasi atau input data tambahan kepada program ketika program dieksekusi. Ini memungkinkan kita untuk menjalankan program dengan berbagai konfigurasi tanpa perlu mengubah kode program secara langsung.

Latihan :

Berikut beberapa contoh kode program dengan kesalahan, dan tugas Anda adalah memperbaiki kesalahan tersebut sehingga program dapat berjalan dengan benar. memperbaiki kesalahan tersebut agar program berjalan dengan benar.

Latihan 1: Memperbaiki Kode Method

```
 1. public class Calculator {
 2.     // Method untuk menjumlahkan dua angka
 3.     public int addNumbers(int a, int b) {
 4.         return a + b;
 5.     }
 6.
 7.     // Method untuk mengalikan dua angka
 8.     public int multiplyNumbers(int a, int b) {
 9.         return a * b;
10.     }
11.
12.     // Method untuk menghitung kuadrat dari sebuah angka
13.     public int calculateSquare(int num) {
14.         return num * num;
15.     }
16.
17.     public static void main(String[] args) {
18.         Calculator calculator = new Calculator();
19.
20.         // Memanggil method addNumbers
21.         int sum = calculator.addNumbers(5, 10);
22.         System.out.println("Hasil penjumlahan: " + sum);
23.
24.         // Memanggil method multiplyNumbers
25.         int product = calculator.multiplyNumbers(3, 7);
```

```
26.         System.out.println("Hasil perkalian: " + product);
27.
28.         // Memanggil method calculateSquare
29.         int squaredNumber = calculator.calculateSquare(4);
30.         System.out.println("Hasil kuadrat: " + squaredNumber);
31.     }
32. }
33.
```

Latihan 2: Memperbaiki Kode Method Overloading

```
 1. public class StringUtils {
 2.     // Method untuk menggabungkan dua string
 3.     public String concatenate(String str1, String str2) {
 4.         return str1 + str2;
 5.     }
 6.
 7.     // Method untuk menggabungkan tiga string
 8.     public String concatenate(String str1, String str2, String str3) {
 9.         return str1 + str2 + str3;
10.     }
11.
12.     // Method untuk mengonversi angka menjadi string
13.     public String convertToString(int number) {
14.         return String.valueOf(number);
15.     }
16.
17.     public static void main(String[] args) {
18.         StringUtils utils = new StringUtils();
19.
20.         // Memanggil method concatenate dengan dua string
21.         String result1 = utils.concatenate("Hello, ", "world!");
22.         System.out.println(result1);
23.
24.         // Memanggil method concatenate dengan tiga string
25.         String result2 = utils.concatenate("Java", " is", " awesome!");
26.         System.out.println(result2);
27.
28.         // Memanggil method convertToString
29.         String numString = utils.convertToString(100);
30.         System.out.println("Hasil konversi: " + numString);
31.     }
32.  }
```

Latihan 3: Memperbaiki Kode Method dengan Argument Baris Perintah

```
 1. public class CommandLineArgumentsExample {
 2.     // Method untuk mencetak argumen baris perintah
 3.     public void printArguments(String[] args) {
 4.         if (args.length > 0) {
 5.             System.out.println("Argumen yang diberikan:");
 6.             for (String arg : args) {
 7.                 System.out.println(arg);
 8.             }
```

```
        } else {
            System.out.println("Tidak ada argumen yang diberikan.");
        }
    }

    public static void main(String[] args) {
        CommandLineArgumentsExample example = new CommandLineArgumentsExample();

        // Memanggil method printArguments dengan argumen baris perintah
        example.printArguments(args);
    }
}
```

Latihan 4: Perbaiki kode program berikut yang mengalami kesalahan:

```
public class NumberManipulation {
    public int multiply(int a, int b) {
        return a + b;
    }

    public void printSquare(int num) {
        int square = num * num;
        System.out.println("Kuadrat dari " + num + " adalah " + square);
    }

    public static void main(String[] args) {
        NumberManipulation manipulation = new NumberManipulation();

        int result = manipulation.multiply(5, 10);
        System.out.println("Hasil perkalian: " + result);

        manipulation.printSquare(7);
    }
}

```

Tugas :

1. Buatlah program Java yang menggunakan method overloading untuk menghitung luas dari berbagai bentuk geometri (lingkaran, segitiga, dan persegi). Setiap bentuk geometri memiliki rumus luas yang berbeda. Program harus dapat menerima input dari pengguna berupa dimensi-dimensi yang diperlukan untuk menghitung luas, lalu menampilkan hasil perhitungannya.
2. Buatlah program Java yang menerapkan konsep recursion untuk menghitung faktorial dari sebuah bilangan bulat. Program harus dapat menerima input dari pengguna berupa bilangan bulat positif, lalu menampilkan hasil faktorialnya.