Method

Method

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

Kode : Method

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

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


}
```

Method Parameter

Method Parameter

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

Kode : Method Parameter

```
public static void main(String[] args) {
    sayHello("Eko", "Khannedy");
}

static void sayHello(String firstName, String lastName) {
    System.out.println("Hello " + firstName + " " + lastName);
}

}
```

Method Return Value

Method Return Value

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

Kode : Method Return Value

```
public static void main(String[] args) {
    var a = 100;
    var b = 200;
    var c = sum(a, b);

    System.out.println(c);
}

static int sum(int value1, int value2) {
    var total = value1 + value2;
    return total;
}
}
```

Method Variable Argument

Method Variable Argument

- Kadang kita butuh mengirim data ke method sejumlah data yang tidak pasti
- Biasanya, agar bisa seperti ini, kita akan menggunakan Array sebagai parameter di method tersebut
- Namun di Java, kita bisa menggunakan variable argument, untuk mengirim data yang berisi jumlah tak tentu, bisa nol atau lebih
- Parameter dengan tipe variable argument, hanya bisa ditempatkan di posisi akhir parameter

Kode : Tanpa Variable Argument

```
static void sayCongrats(String name, int[] values) {

  int total = 0;
  for (var value : values) {
    total += value;
  }
  int finalValue = total / values.length;

  if (finalValue >= 75) {
    System.out.println("Selamat " + name + ", Anda Lulus");
  } else {
    System.out.println("Maaf " + name + ", Anda Lulus");
```

Kode : Dengan Variable Argument

```
    sayCongrats( name: "Eko",  ...values: 80, 90, 79, 48);
}

static void sayCongrats(String name, int... values) {
    int total = 0;
    for (var value : values) {
        total += value;
    }
    int finalValue = total / values.length;
    if (finalValue >= 75) {
        System.out.println("Selamat " + name + ", Anda Lulus");
    } else {
```

Method Overloading

Method Overloading

- Method overloading adalah kemampuan membuat method dengan nama yang sama lebih dari sekali.
- Namun ada ketentuannya, yaitu data parameter di method tersebut harus berbeda-beda, enta jumlah atau tipe data parameternya
- Jika ada yang sama, maka program Java kita akan error

Kode : Method Overloading

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

static void sayHello(String firstName) {
    System.out.println("Hello " + firstName);
}

static void sayHello(String firstName, String lastName) {
    System.out.println("Hello " + firstName + " " + lastName);
}
}
```

Recursive Method

Recursive Method

- Recursive method adalah kemampuan method memanggil method dirinya sendiri
- Kadang memang ada banyak problem, yang lebih mudah diselesaikan menggunakan recursive method, seperti contohnya kasus factorial

Kode : Factorial Loop

```
}

static int factorial(int value) {
  var result = 1;
  for (int i = 1; i <= value; i++) {
    result *= i;
  }
  return result;
}
```

Kode : Factorial Recursive

```
static int factorialRecursive(int value) {
    if (value == 1) {
        return 1;
    } else {
        return value * factorialRecursive( value: value - 1);
    }
}
}
```

Problem Dengan Recursive

- Walaupun recursive method itu sangat menarik, namun kita perlu berhati-hati
- Jika recursive terlalu dalam, maka akan ada kemungkinan terjadi error StackOverflow, yaitu error dimana stack method terlalu banyak di Java
- Kenapa problem ini bisa terjadi? Karena ketika kita memanggil method, Java akan menyimpannya dalam stack, jika method tersebut memanggil method lain, maka stack akan menumpuk terus, dan jika terlalu dalam, maka stack akan terlalu besar, dan bisa menyebabkan error StackOverflow

Kode : Error StackOverflow

```
static void loop(int value) {
    if (value == 0) {
        System.out.println("Selesai");
    } else {
        System.out.println("Loop-" + value);
        loop( value: value - 1);
    }
}
}
```

Scope

Scope

- Di Java, variable hanya bisa diakses di dalam area dimana mereka dibuat.
- Hal ini disebut scope
- Contoh, jika sebuah variable dibuat di method, maka hanya bisa diakses di method tersebut, atau jika dibuat didalam block, maka hanya bisa diakses didalam block tersebut

Kode : Scope

```
@  static void sayHello(String name) {
     String hello = "Hello " + name;
     if (!name.isBlank()) {
       String hi = "Hi " + name;
       System.out.println(hi);
     }

     System.out.println(hello);
     System.out.println(hi); // error
   }
 }
```

Komentar

Komentar

- Kadang dalam membuat program, kita sering menempatkan komentar di kode program tersebut
- Komentar adalah kode program yang akan di hiraukan saat proses kompilasi, sehingga di binary code Java, tidak akan ada kode komentar tersebut
- Biasanya komentar digunakan untuk dokumentasi

Kode : Komentar

```
/**
 * Menghitung jumlah a + b
 *
 * @param a nilai a
 * @param b nilai b
 * @return a + b
 */
static int sum(int a, int b) {
    // jumlahkan a + b
    return a + b;
}
}
```