# Method Dan Rekursif

## Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

1. Mengenal konsep method.
2. Mengenal dan membedakan jangkauan variabel.
3. Mengenal dan membedakan passing argument pada method.
4. Mengenal, memahami dan menggunakan method rekursif.

## Alat & Bahan

Alat & Bahan Yang digunakan adalah hardware perangkat PC beserta Kelengkapannya berjumlah 40 PC serta Software IntelliJ IDEA yang telah terinstall pada masing-masing PC

## Dasar Teori

### Method

Method, atau dalam bahasa pemrograman lain disebut fungsi, adalah sekumpulan perintah operasi program yang dapat menerima argumen input dan dapat memberikan hasil output yang dapat berupa nilai ataupun sebuah hasil operasi. Hasil akhir fungsi akan berupa sebuah nilai balik (return). Nama method yang didefinisikan sendiri oleh pemrogram tidak boleh sama dengan nama keyword pada Java.

Penggunaan method terutama terasa pada program yang cukup besar. Pembangunan dan perawatan program yang besar akan lebih mudah dilakukan jika program dibagi ke dalam bagian-bagian yang lebih kecil, atau lebih sering disebut modul. Pembuatan modul ini dilakukan dengan menggunakan method. Selain itu, method digunakan agar pemrogram dapat menghindari penulisan bagian program (kode) berulang-ulang (*softwarereusaibility*), dan dapat menyusun kode program agar terlihat lebih rapi dan kemudahan dalam *debugging* program.

Method dapat dideklarasikan dengan sintaks sebagai berikut.

|  |
| --- |
| Tipe\_Data\_Kembalian Nama\_Method(List\_Parameter) {  Isi\_Method;  } |

Tipe data kembalian, sesuai dengan namanya, mendefinisikan tipe data yang akan dikembalikan oleh suatu method. Seluruh tipe data dapat menjadi tipe data kembalian. Sementara itu, nama dari suatu method mengikuti aturan penamaan variabel secara umum. Parameter list berisi rangkaian tipe data dan nama variabel masukan yang dipisahkan oleh koma (jika lebih dari satu parameter). Tipe data kembalian dan parameter akan dibahas lebih lanjut pada sub-bab berikutnya.

### Nilai Kembalian pada Method

Seperti telah disebutkan pada sub-bab sebelum ini, suatu method dapat mengembalikan nilai (hasil), yang tipe datanya harus sesuai dengan tipe data saat deklarasi method. Tipe data kembalian dapat berupa tipe data primitif ataupun *reference* (objek). Jika ternyata method tersebut tidak mengembalikan nilai, maka tipe data kembalian bagi method tersebut adalah **void**.

Sementara itu, bagi method yang mengembalikan nilai, maka nilai yang dikembalikan ditandai dengan sintaks *return*. Perintah *return* ini akan mengakhiri blok perintah pada method, dan mengembalikannya pada perintah pemanggilnya. Perintah sesudah *return* tidak akan dieksekusi. Untuk lebih mendalami kedua tipe ini, perhatikan Program 9-1.

|  |  |
| --- | --- |
| package praktikum9\_1;  import java.util.Scanner;  class HitungLuas {  class Segitiga{  int alas;  int tinggi;  }  void setAlas(Segitiga segitiga, int al){  segitiga.alas = al;  }  void setTinggi(Segitiga segitiga, int tgg){  segitiga.tinggi = tgg;  }  double luasSegitiga(Segitiga segitiga){  double luas = 0.5\*segitiga.alas\*segitiga.tinggi;  return luas;  }  public static void main (String args[]){  Scanner sc= new Scanner(System.in);  Segitiga segi3 = new HitungLuas().new Segitiga();  HitungLuas segi2 = new HitungLuas();  System.out.println("Masukkan alas segitiga");  segi2.setAlas(segi3, sc.nextInt());  System.out.println("Masukkan tinggi segitiga");  segi2.setTinggi(segi3, sc.nextInt());  double luas = segi2.luasSegitiga(segi3);  System.out.println("Luas segitiga: " + luas);  }  } | Apakah Outputnya?  ………………………… |

Program ‎9‑1 Return value

Pada Program 9-1, digunakan tipe data bentukan segitiga yang memiliki anggota alas dan tinggi. Nilai alas dan tinggi ini diperoleh melalui dua method void, setAlas dan setTinggi. Perhatikan, karena tipe kembalian kedua method adalah void, maka tidak ada perintah return pada keduanya. Sebaliknya, method luasSegitiga memiliki tipe data kembalian double, karena itu, method ini mengembalikan nilai double luas. Tipe data dari variabel luas harus sama dengan tipe data kembalian dari method (luas tidak bisa bertipe integer, misalnya).

Method yang telah dibuat, dipanggil pada main. Pada Java, pemanggilan method adalah dengan meletakkan tanda titik sebelum method yang dipanggil. Karena method pada Program 9-1 dibuat di dalam kelas HitungLuas, maka yang dapat memanggil method tersebut adalah objek dari kelas HitungLuas. Dalam hal ini, objek kelas HitungLuas adalah segi2. Jadi, baris perintah segi2.setAlas(segi3, sc.nextInt()); menyatakan pemanggilan method setAlas oleh objek segi2.

Sementara itu, segi3 merupakan objek dari kelas Segitiga, yang berisi alas dan tinggi. Segi3 digunakan sebagai parameter masukan pada method setAlas, setTinggi dan hitungLuas. Nilai kembalian dari method hitungLuas disimpan pada variabel luas. Perlu diperhatikan, method penyimpan nilai kembalian harus memiliki tipe data yang sama dengan nilai kembalian. Jadi, luas tidak boleh bertipe data integer, String atau lainnya.

### Variable Scope

Scope (jangkauan) suatu variabel menentukan bagian yang dapat menggunakan variabel tersebut. Variabel lokal dideklarasikan di dalam method atau blok program tertentu (misal, suatu perulangan atau percabangan), sehingga variabel tersebut hanya dapat dikenali di dalam blok tersebut. Ketika blok atau method berakhir, nilai dari variabel lokal akan hilang.

Sementara itu, variable global merupakan variabel yang dikenali oleh keseluruhan komponen program.Pada Java, tergantung dari scope variabelnya, terdapat instance variable, static variable, method parameter/ argument dan local variable.

#### Instance Variable

Suatu kelas biasanya terdiri atas atribut yang menyimpan data mengenai objek dan method yang melakukan manipulasi pada atribut tersebut. Atribut, yang direpresentasikandengan variable, seringkalidisebut field, dandideklarasikan di dalamkelas, tapi di luar method. Field inilah yang disebutdengan instance variable. Instance variable merupakananggotadarisuatukelas, danmenyimpan data darisuatuobjek, jadiobjek yang satudenganobjek yang lain akanmemilikinilai instance variable yang berbeda. Untuklebihjelasnya, perhatikan Program 9-2.

|  |
| --- |
| package praktikum9\_2;  import java.util.Scanner;  class VariableScope {  static int data = 10;  String namaBarang;  int harga;  static void tampil(){  System.out.println("Contoh static method");  }  void setNama(String nama){  namaBarang = nama;  }  void setHarga(int harga){  this.harga = harga;  }  double hitungBelanja(int jum){  double bayar = harga \* jum;  double diskon = 0.1;  if (bayar >= 1500000)  bayar = bayar – bayar \* diskon;  return bayar;  }  public static void main (String args[]){  Scanner sc= new Scanner(System.in);  tampil();  VariableScope var1 = new VariableScope();  VariableScope var2 = new VariableScope();  var1.setNama("Sepatu Cinderella");  var2.setNama("Lampu ajaib Aladdin");  var1.setHarga(150000);  var2. setHarga(100000);  System.out.println("Masukkan jumlah pembelian: ");  int jumlah = sc.nextInt());  System.out.println("Pembayaran untuk: " + var1.namaBarang);  System.out.println(var1.hitungBelanja(jumlah));  System.out.println("Pembayaran untuk: " + var2.namaBarang);  System.out.println(var2.hitungBelanja(jumlah));  System.out.println("Nilai variabel data: " + data);  System.out.println("Data pada var1: " + var1.data);  System.out.println("Data pada var2: " + var2.data);  var2.data = 25;  System.out.println("Nilai variabel data: " + data);  System.out.println("Data pada var1: " + var1.data);  System.out.println("Data pada var2: " + var2.data);  }  } |
| Apakah Outputnya?  ………………………… |

Program ‎9‑2 Variable Scope

Instance variable dari Program 9-2 adalah variable nama Barang dan harga. Padadasarnya, instance variable merupakan variabel global, karenaitu method-method yang berada di dalam kelas Variable Scope dapat menggunakan nama Barang dan harga. Namun, karena instance variable berasosiasi dengan objek tertentu, makanilai variable ini dapat berbeda antara objek satu dan lainnya.

Pada Program 9-2 terlihat, nilai untuk nama pada objek var1 adalah “Sepatu Cinderella”, sementara untuk var2, nilai dari nama adalah “Lampu ajaib Aladdin”. Demikianjugauntukharga, antaraobjek var1 dan var2 nilaidarivariabelharganyaberbeda. Perubahan pada objek yang satu tidak akan mempengaruhi nilai dari objek yang lain.

#### Static Variable

Biasanya, variabel pada suatu kelas hanya dapat diakses melalui objek tertentu. Namun, terkadang diperlukan variabel yang dapat digunakan tanpa asosiasi dengan objek apapun. Tipe variable seperti itu merupakan variabel static. Ketika sebuah variable dideklarasikan sebagai static, maka variable tersebut dapat diakses sebelum ada satu pun objek dari kelas yang bersangkutan dibuat. Selain variable, suatu method juga dapat dideklarasikan sebagai static.

Static method memiliki beberapa batasan, yaitu:

* + Hanya dapat memanggil static method, tidak bisa memanggil non-static method
  + Hanya dapat mengakses static data
  + Tidak dapat merujuk ke this dan super ( pembahasan lebih lanjut pada mata kuliah Pemrograman Berbasis Objek )

Pendeklarasian variabel atau method sebagai static adalah dengan menambahkan keyword static di depan variable atau method tersebut. Pada Program 9-2, variable dan method static adalah data dantampil.

|  |
| --- |
| static int data = 10;  static void tampil(){  System.out.println("Contoh static method");} |

Variabel dan method static adalah milik kelas yang mendeklarasikannya, mereka tidak terhubung dengan objek tertentu. Pada Program 9-2, method tampil dapat dipanggil walaupun objek dari kelas (var1 dan var2) belum dibuat. Demikian juga dengan variabel data. Variabel ini dapat langsung diakses tanpa asosiasi dengan objek tertentu, namun dapat juga dihubungkan dengan objek tertentu.

|  |
| --- |
| System.out.println("Nilai variabel data: " + data);  System.out.println("Data pada var1: " + var1.data);  System.out.println("Data pada var2: " + var2.data);  var2.data = 25;  System.out.println("Nilai variabel data: " + data);  System.out.println("Data pada var1: " + var1.data);  System.out.println("Data pada var2: " + var2.data); |

Pada potongan program diatas, terlihat bahwa variabel data juga terhubung dengan objek var1 dan var2. Akses suatu class variable (static variable) dibagi kepada semua objek yang menggunakannya, objek tidak memiliki salinan variabel untuk dirinya sendiri (tidak seperti pada instance variable). Ketika dilakukan perubahan melalui salah satu objek, maka semua objek yang terhubung juga akan mengalami perubahan, seperti yang terlihat pada Program 9-2 setelah perintah var2.data = 25.

#### Parameter Formal/ Method parameter

Parameter formal adalah variabel yang ada pada daftar parameter ketika mendeklarasikanmethod. Pada Java, parameter formal ini disebut dengan method parameter. Seperti telah disebutkan pada sub-bab sebelum ini, method parameter terdiri atas rangkaian tipe\_data dan nama\_variabel. Method dapat memiliki paramater atau tidak memiliki parameter. Jika suatu method tidak memiliki parameter, maka list parameter dikosongkan.

|  |
| --- |
| static void tampil()  void setNama(String nama)  void setHarga(int harga)  int hitungBelanja(int jum) |

Pada Program 9-2, methodtampil merupakan suatu method yang tidak memiliki parameter, sementara method setNama, setHarga dan hitungBelanja memiliki parameter nama, harga dan jum sebagai parameter formalnya.

#### Parameter Aktual/ Method argument

Adapun parameter aktual adalah parameter (tidak selamanya menyatakan variabel) yang dipakai ketika suatu method dipanggil. Jika terdapat lebih dari satu parameter, maka urutan dari parameter aktual harus sama dengan parameter formal.

|  |
| --- |
| void setNama(String nama){  namaBarang = nama; }  void setHarga(int harga){  this.harga = harga;  }  int hitungBelanja(int jum){  double bayar = harga \* jum;  double diskon = 0.1;  if (bayar >= 1500000)  bayar = bayar – bayar \* diskon;  return bayar;  }  var1.setHarga(150000);  System.out.println(var1.hitungBelanja(jumlah)); |

Berdasarkan Program 9-2, terlihat bahwa parameter formal bagi method setHarga adalah variabel harga. Namun yang menjadi parameter aktualnya adalah literal 150000. Demikian juga dengan method hitungBelanja, argumen yang dilewatkan pada method tersebut adalah variabel jumlah.

Pada method setHarga, keyword this digunakan untuk menunjukkan bahwa variabel harga yang dimaksud adalah instance variable dari kelas VariableScope. Keyword this harus digunakan karena method setHarga memiliki nama method parameter sama dengan nama instance variablenya (yaitu “harga”). Jika tidak, maka keyword this tidak perlu digunakan, seperti pada method setNama.

#### Variabel lokal

Seperti telah dijelaskan sebelumnya, variabel lokal adalah variabel yang memiliki ruang lingkup hanya pada method atau blok program tertentu.

|  |
| --- |
| int hitungBelanja(int jum){  double bayar = harga \* jum;  double diskon = 0.1;  if (bayar >= 1500000)  bayar = bayar – bayar \* diskon;  return bayar;  } |

Pada contoh method hitungBelanja dari Program 9-2, variabel bayar dan diskon merupakan variabel lokal yang hanya dikenali oleh method hitungBelanja. Jika isi variabel bayar atau diskon dicoba untuk ditampilkan ke layar diluar method hitungBelanja, misal di dalam main, maka program akan tidak bisa ter-compile karena variabel tersebut tidak dikenali pada main.

### Argument Passing

Ada dua cara melewatkan parameter ke dalam fungsi, yaitu by value dan by reference.

#### Pass by Value

Pada pemanggilan dengan nilai, nilai dari parameter aktual akan disalin kedalam parameter formal, jadi parameter aktual tidak akan berubah meskipun parameter formalnya berubah. Untuk lebih jelasnya perhatikan Program 9-3.

|  |
| --- |
| package praktikum9\_3;  import java.util.Scanner;  class PassByValue {  void tukar(int a, int b){  int temp = a;  a = b;  b = temp;  }  public static void main (String args[]){  Scanner sc= new Scanner(System.in);  PassByValuepbv = new PassByValue();  System.out.println("Masukkan angka pertama: ");  int bil1 = sc.nextInt();  System.out.println("Masukkan angka kedua: ");  int bil2 = sc.nextInt();  System.out.println("Angka sebelum ditukar: " + bil1 + ", " + bil2);  pbv.tukar(bil1, bil2);  System.out.println("Angka sesudah ditukar: " + bil1 + ", " + bil2);  }  } |
| Apakah Outputnya?  ………………………… |

Program ‎9‑3 Pass by value

Pada Program 9-3, nilai bil1 dan bil2 sebelum dan sesudah pemanggilan method tukar tetap sama, operasi yang dilakukan dalam method tukar tidak mempengaruhi isi dari kedua variabel.

#### Pass by Reference

Pemanggilan dengan referensi merupakan cara untuk melewatkan alamat suatu variabel ke suatu method. Cara ini dapat merubah nilai dari variabel aktual yang dilewatkan ke dalam fungsi. Pada Java, semua objek akan dilewatkan dengan dengan cara pass by reference. Untuk lebih jelasnya perhatikan contoh pada Program 9-4.

|  |
| --- |
| package praktikum9\_4;  import java.util.Scanner;  class PassByRef {  int x;  int y;  void tukar(PassByRef pass){  int temp = pass.x;  pass.x = pass.y;  pass.y = temp;  }  public static void main (String args[]){  Scanner sc= new Scanner(System.in);  PassByRef pbr = new PassByRef();  System.out.println("Masukkan angka pertama: ");  pbr.x = sc.nextInt();  System.out.println("Masukkan angka kedua: ");  pbr.y = sc.nextInt();  System.out.println("Angka sebelum ditukar: " + pbr.x + ", " + pbr.y );  pbr.tukar(pbr);  System.out.println("Angka sesudah ditukar: " + pbr.x + ", " + pbr.y );  }  } |
| Apakah Outputnya?  ………………………… |

Program ‎9‑4 Pass by reference

Pada Program 9-4, nilai x dan y dari objek pbr sebelum dan sesudah pemanggilan method adalah berbeda, operasi penukaran di dalam method mempengaruhi isi dari objek. Hal ini terjadi karena pada dasarnya variabel dengan tipe objek memang merujuk pada objek tersebut. Jadi perubahan yang terjadi pada variabel mempengaruhi langsung objek yang dirujuk.

### Method Overloaded

Overloading method adalah membuat method dengan nama sama tetapi parameter berbeda. Perbedaan disini bisa jadi berupa perbedaan jumlah, urutan, ataupun tipe data dari parameter. Pengubahan nama method, maupun tipe data kembalian tidak termasuk ke dalam method overload. Untuk lebih jelasnya bisa dilihat pada Program 9-5.

|  |  |
| --- | --- |
| package praktikum9\_5;  import java.util.Scanner;  class Overloading {  double kali(int x){  double a = 10.0;  x = x \* a;  return x;  }  double kali(int x, int y){  double z = x \* y;  return z;  }  double kali(double x, int y){  double z = x \* y;  return z;  }  public static void main (String args[]){  Scanner sc= new Scanner(System.in);  Overloading over = new Overloading();  System.out.println("Masukkan angka pertama");  int bil1 = sc.nextInt();  System.out.println("Masukkan angka kedua");  int bil2 = sc.nextInt();  System.out.println("Masukkan angka ketiga");  double bil3 = sc.nextDouble();  System.out.print("Angka pertama dikali 10");  System.out.println(over.kali(bil1));  System.out.print("Perkalian angka 1 dan 2");  System.out.println(over.kali(bil1,bil2));  System.out.print("Perkalian angka 3 dan 2");  System.out.println(over.kali(bil3,bil2));  }  } | Apakah Outputnya?  ………………………… |

Program ‎9‑5 Overloading method

Pada Program 9-5 terdapat tiga method overload. Method kali pertama hanya menerima satu parameter bertipe integer, method kali kedua menerima dua parameter integer, dan method kali ketiga menerima satu parameter double dan satu parameter integer. Overloading method digunakan ketika dibutuhkan suatu method dengan fungsi hampir sama namun parameter masukannya berbeda.

### Rekursif

Rekursif berarti suatu methodyang memanggil dirinya sendiri. Berikut adalah beberapa contoh implementasi rekursif.

1. Faktorial

|  |  |
| --- | --- |
| package praktikum9\_6;  import java.util.Scanner;  class Faktorial{  int fakt(int n){  if (n<0)  return -1;  else if (n==1 || n==0)  return 1;  else  return (n \* fakt(n-1));  }  public static void main (String args[]){  Scanner sc= new Scanner(System.in);  Faktorial faktor = new Faktorial ();  System.out.println("Masukkan angka: ");  int bil = sc.nextInt();  System.out.print("Faktorial dari" + bil);  System.out.println("Adalah " + faktor.fakt(bil));  }  } | Apakah Outputnya?  …………………………  …………………………  ………………………… |

Program ‎9‑6 Program faktorial

1. Menampilkan urutan bilangan secara rekursif

|  |  |
| --- | --- |
| package praktikum9\_7;  import java.util.Scanner;  class RekursifUrut{  voidrek(int n){  if (n==1)  System.out.println(n);  else{  System.out.println(n);  rek(n-1);  }  }  public static void main (String args[]){  Scanner sc= new Scanner(System.in);  RekursifUrut rec = new RekursifUrut ();  System.out.println("Masukkan angka: ");  int bil = sc.nextInt();  System.out.println("Hasilnya ");  rec.rek(bil);  }  } | Apakah Outputnya?  …………………………  …………………………  ………………………… |

Program ‎9‑7 Urutan bilangan rekursif

## Latihan

1. Buatlah fungsi untuk menghitung pangkat dari suatu bilangan secara rekursif.

Contoh:

Input bilangan: 2

Input pangkat : 3

Output:

Pangkat 3 dari 2 adalah 8

1. Buatlah program yang dapat menyimpan data mahasiswa (max. 10) kedalam sebuah array dengan field Nama, NIM, UTS, UAS, Tugas, dan Nilai Akhir. Nilai Akhir diperoleh dari method dengan rumus 0.3 \* UTS + 0.4 \* UAS + 0.3 \* Tugas.