2016

**LABORATORIUM PRIDE**

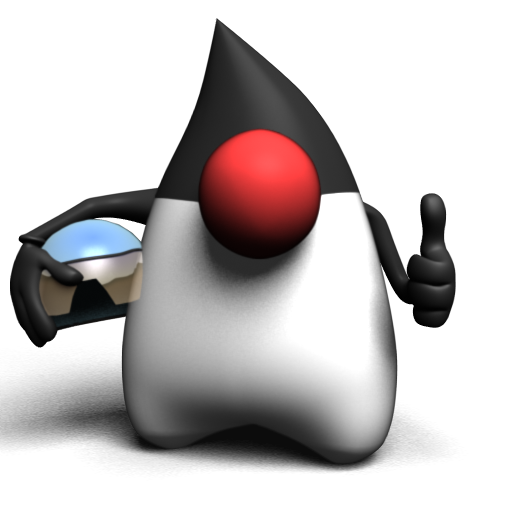
**KELOMPOK KEAHLIAN PROGRAMMING**

**FAKULTAS ILMU TERAPAN**

**UNIVERSITAS TELKOM**

Modul Praktikum

Implementasi Algoritma



**Hanya dipergunakan di lingkungan Fakultas Ilmu Terapan**

## DAFTAR PENYUSUN

Cahyana, S.T., M.Kom

Diperbaiki Oleh

Indra Azimi, S.T., M.T.

Ahmad Suryan, S.T., M.T.

## LEMBAR REVISI

|  |  |  |
| --- | --- | --- |
| No | Keterangan Revisi | Tanggal Revisi Terakhir |
| 1 | Revisi Bagian Pertama | 30 Agustus 2016 |
| 2 | Revisi Bagian Kedua | 05 September 2016 |
| 3 | Revisi Bagian Ketiga | 07 Oktober 2016 |
| 4 | Revisi Bagian Keempat | 14 Oktober 2016 |
| 5 | Revisi Bagian Kelima | 16 Oktober 2016 |

## LEMBAR PERNYATAAN

Saya yang bertanggung jawab di bawah ini:

Nama : Cahyana, S.T., M.Kom

NIP : 14781391-1

Dosen PJMP : Implementasi Algoritma

Kelompok Keahlian : Programming

Menerangkan dengan sesungguhnya bahwa modul ini telah direview dan akan digunakan untuk pelaksanaan praktikum di Semester Genap Tahun Ajaran 2016/2017 di Laboratorium PRIDE Fakultas Ilmu Terapan Universitas Telkom

Bandung, 10 Agustus 2016

Mengetahui,

Ketua Kelompok Keahlian Dosen PJMP

Hariandi Maulid, S.T., M.Sc Cahyana, S.T., M.Kom

NIP 15781201-4 NIP 14781391-1

# DAFTAR ISI

[DAFTAR PENYUSUN 1](#_Toc464206888)

[LEMBAR REVISI 2](#_Toc464206889)

[LEMBAR PERNYATAAN 3](#_Toc464206890)

[DAFTAR ISI 4](#_Toc464206891)

[DAFTAR GAMBAR 5](#_Toc464206892)

[DAFTAR PROGRAM 6](#_Toc464206893)

[DAFTAR TABEL 7](#_Toc464206894)

[Modul 8 : Tipe Data Bentukan 8](#_Toc464206895)

[8.1 Tujuan 8](#_Toc464206896)

[8.2 Alat & Bahan 8](#_Toc464206897)

[8.3 Dasar Teori 8](#_Toc464206898)

[8.3.1 String 8](#_Toc464206899)

[8.3.2 StringBuilder 11](#_Toc464206900)

[8.3.3 Tipe Data Reference 12](#_Toc464206901)

[8.4 Latihan 14](#_Toc464206902)

[Modul 9 : Method dan Rekursif 15](#_Toc464206903)

[9.1 Tujuan 15](#_Toc464206904)

[9.2 Alat & Bahan 15](#_Toc464206905)

[9.3 Dasar Teori 15](#_Toc464206906)

[9.3.1 Method 15](#_Toc464206907)

[9.3.2 Nilai Kembalian pada Method 15](#_Toc464206908)

[9.3.3 Variable Scope 17](#_Toc464206909)

[9.3.4 Argument Passing 21](#_Toc464206910)

[9.3.5 Method Overloaded 22](#_Toc464206911)

[9.3.6 Rekursif 23](#_Toc464206912)

[9.4 Latihan 24](#_Toc464206913)

# DAFTAR GAMBAR

[Gambar ‎8‑1 Tipe data bentukan dan primitif 16](#_Toc463644272)

# DAFTAR PROGRAM

[Program ‎8‑1 Perbandingan pada String 8](#_Toc464206875)

[Program ‎8‑2 Manipulasi String 9](#_Toc464206876)

[Program ‎8‑3 Penggabungan String 10](#_Toc464206877)

[Program ‎8‑4 Perbandingan String 11](#_Toc464206878)

[Program ‎8‑5 StringBuilder 12](#_Toc464206879)

[Program ‎8‑6 Tipe data bentukan 13](#_Toc464206880)

[Program ‎9‑1 Return value 16](#_Toc464206881)

[Program ‎9‑2 Variable Scope 18](#_Toc464206882)

[Program ‎9‑3 Pass by value 21](#_Toc464206883)

[Program ‎9‑4 Pass by reference 22](#_Toc464206884)

[Program ‎9‑5 Overloading method 23](#_Toc464206885)

[Program ‎9‑6 Program faktorial 24](#_Toc464206886)

[Program ‎9‑7 Urutan bilangan rekursif 24](#_Toc464206887)

# DAFTAR TABEL

[Tabel ‎8‑1 Method pada String 11](#_Toc463644258)

[Tabel ‎8‑2 Method pada StringBuilder 13](#_Toc463644259)

# Tipe Data Bentukan

## Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

1. Mengenal tipe data bentukan (objek)
2. Membedakan antara tipe data primitif dan reference/ bentukan
3. Mengetahui String dan sifat-sifatnya
4. Mengetahui perintah-perintah pada String

## 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

Pada Java, tipe data bentukan disebut sebagai tipe data reference/ objek. Berbeda dengan bahasa C/ C++, Java tidak mengenal struct, semua objek merupakan bentukan dari suatu class. Karena itu, untuk tipe data selain tipe data primitif yang telah dipelajari pada pertemuan dua, merupakan objek yang berasal dari kelas tertentu (yang juga dapat berasal dari kelas yang dibentuk oleh programmer).

### String

String merupakan bentuk data yang sering digunakan dalam bahasa pemrograman untuk mengolah data teks atau kalimat. Jika pada bahasa C, string merupakan kumpulan karakter pada array (array of characters), maka pada bahasa Java, string merupakan suatu objek yang berasal dari class String. Deklarasi dan inisialisasi variabel dengan tipe String dapat dilakukan dengan beberapa cara, seperti berikut:

|  |
| --- |
| String nama = “Nina”;  String nama2 = new String (“Nina”);  char[] nama3 = new char[] {‘N’, ‘i’, ‘n’, ‘a’};  String nama4 = new String(nama3); |

Walaupun menyimpan data yang sama (Nina), variabel nama, nama2 dan nama4 merupakan objek yang berbeda. Jadi jika dilakukan operasi persamaan (‘==’) maka hasilnya akan ‘false’. String dapat menerima nilai dari suatu array karakter, seperti terlihat pada variabel nama4. Keyword new akan membuat sebuah objek String baru, sehingga dua objek String yang memiliki nilai sama tidak akan dianggap sama oleh Java. Perhatikan potongan program berikut.

|  |  |
| --- | --- |
| package praktikum8\_1;  class StringCompare {  public static void main (String args[]){  String nama = “Lilo”;  String nama2 = “Lilo”;  String nama3 = new String(“Lilo”);  String nama4 = new String(“Lilo”);  System.out.println(nama == nama2);  System.out.println(nama == nama3);  System.out.println(nama3 == nama4);  }  } | Apakah Outputnya?  ………………………… |

Program ‎8‑1 Perbandingan pada String

Dalam program 8-1, hasil dari nama == nama2 adalah true, karena kedua variabel merujuk pada objek yang sama (“Lilo”), namun tidak demikian halnya dengan dua perbandingan lainnya. Variabel nama3 dan nama4, walaupun memiliki nilai yang sama (“Lilo”) merupakan dua objek yang berbeda, karenanya hasil dari perbandingan adalah false. Untuk membandingkan isi dari variabel String, gunakan fungsi equals yang akan dibahas kemudian.

String termasuk tipe data yang paling banyak digunakan, dan kelas String memiliki method yang dapat digunakan untuk melakukan manipulasi String, namun tidak mengubahnya, seperti charAt, indexOf, substring, trim, replace dan length.

Tabel ‎8‑1 Method pada String

|  |  |
| --- | --- |
| Sintaks | Deskripsi |
| charAt(int index) | Mengambil satu karakter dari string sesuai indeks |
| indexOf() | Membandingkan apakah karakter atau string tertentu ada pada suatu string. Jika ada, akan mengembalikan indeks pertama yang ditemukan, jika tidak akan mengembalikan nilai -1. |
| substring() | Membuat sub-string dari suatu string. |
| trim() | Menghilangkan white space (baris baru, spasi atau tab) sebelum atau sesudah suatu string. White space di antara string tidak dihilangkan. |
| replace() | Mengembalikan string baru dengan mengganti suatu karakter (atau string) dengan karakter (atau string) lainnya. |
| length() | Mengembalikan nilai panjang suatu string (banyaknya huruf pada string tersebut) |

Seperti pada array, indeks suatu string dimulai dari 0. Yang harus diperhatikan dari operasi manipulasi string adalah, walaupun sekilas terlihat method yang ada dapat mengubah suatu string, pada kenyataannya string tersebut tidak berubah, tetap seperti sebelum dilakukan operasi pada string. Hal ini dikarenakan, dalam Java, String bersifat tidak dapat diubah (*immutable*), nilai suatu string akan tetap seperti pada saat inisialisasi. Untuk lebih jelasnya perhatikan contoh program berikut.

|  |  |
| --- | --- |
| package praktikum8\_2;  class StringManipulation {  public static void main (String args[]){  String nama = “Telkom University ”;  System.out.println(nama.charAt(2));  System.out.println(nama.indexOf(‘k’));  System.out.println(nama.indexOf(“Telkom”));  System.out.println(nama.indexOf(“koma”));  String sub1 = nama.substring(3);  String sub2 = nama.substring(7,10);  System.out.println(nama.length());  System.out.println(nama.trim());  System.out.println(nama.length());  System.out.println(nama.replace(‘r’, ‘R’));  System.out.println(nama.repplace(“ty”,”tas”));  System.out.println(nama);  }  } | Apakah Outputnya?  ………………………… |

Program ‎8‑2 Manipulasi String

Perintah indexOf dapat digunakan untuk mencari baik karakter maupun kata (string). Nilai yang dimunculkan adalah indeks dari karakter pertama yang ditemukan. Perintah substring dapat digunakan dengan menuliskan indeks dari karakter pertama yang akan diambil, atau dari karakter pertama dan karakter akhir. Pada Program 8-2, perintah substring(3) akan membuat substring baru mulai dari huruf ‘k’ sampai akhir string, sementara perintah substring(7,10) akan mengambil huruf dari indeks 7-9 (indeks ke-10 tidak diambil).

Perintah length akan mengembalikan panjang suatu string, *white* *space* dan tanda baca dihitung. Perintah trim akan menghapus *white* *space* sebelum atau sesudah string, namun tidak menghapus *white* *space* diantara string. Replace dapat mengganti karakter maupun string. Namun, semua manipulasi yang dilakukan tidak akan mengubah string awal, jadi isi dari variabel nama tetap “Telkom University”.

#### Penggabungan String

Dua string atau lebih dapat digabungkan dengan operator penggabungan (*concatenation*) + atau +=. Untuk lebih jelasnya dapat dilihat pada Program 8-3.

|  |  |
| --- | --- |
| package praktikum8\_3;  import java.util.Scanner;  class StringConcatenation {  public static void main (String args[]){  String gabung = “Telkom” + “” + “University”;  System.out.println(gabung);  gabung += “ Bandung”;  System.out.println(gabung);    Scanner in = new Scanner(System.in);  System.out.println(“Masukkan NIM Anda:”);  int NIM = in.nextInt();  System.out.println(“Masukkan Kelas Anda (1-4):”);  int kelas = in.nextInt();    System.out.println(“Masukkan nama Anda:”);  String nama = in.next();  System.out.println(“NIM, kelas dan nama Anda:”);  System.out.println(NIM + kelas + nama);  System.out.println(“Bandingkan dengan:”);  System.out.println(“”+ NIM + kelas + nama);  }  } | Apakah Outputnya?  ………………………… |

Program ‎8‑3 Penggabungan String

Operator + dan += dapat menggabungkan baik String maupun integer. Namun, harus diperhatikan, jika menggabungkan literal string dengan integer, maka nilai integer akan diperlakukan sebagai suatu integer (jadi akan dilakukan operasi terhadap nilai integer tersebut, dalam hal ini penjumlahan). Untuk mencegah hal tersebut, awali proses dengan suatu string (dalam Program 8-3 diwakili dengan “”).

#### Perbandingan String

Sebuah string dapat dibandingkan dengan string lainnya untuk melihat apakah kedua string tersebut sama. Perbandingan string dilakukan dengan menggunakan perintah equals. Sebaiknya tidak menggunakan operator “==” dalam perbandingan string, karena dapat menimbulkan hasil yang tidak sesuai. Untuk lebih jelasnya, perhatikan Program 8-4.

|  |  |
| --- | --- |
| package praktikum8\_4;  class StringCompare {  public static void main (String args[]){  String banding = “Telkom”;  String banding2 = “Telkom”;  System.out.println(banding == banding2);  System.out.println(banding.equals(banding2));  String banding3 = new String(“Telkom”);  String banding4 = new String(“Telkom”);  System.out.println(banding3 == banding4);  System.out.println(banding3.equals(banding4));  System.out.println(banding == banding4);  System.out.println(banding.equals(banding3));  }  } | Apakah Outputnya?  ………………………… |

Program ‎8‑4 Perbandingan String

Hasil perbandingan antara variabel banding dan banding2 adalah true baik untuk operator “==” maupun perintah equals karena kedua variabel merujuk pada objek yang sama di Java. Sebaliknya, keyword new akan selalu membuat objek baru. Hasil dari perbandingan banding3 == banding4 adalah false karena kedua variabel merujuk pada objek yang berbeda (operator “==” membandingkan apakah kedua objek sama). Namun ketika digunakan equals, hasilnya akan menjadi true karena equals membandingkan isi string.

### StringBuilder

Karena String bersifat tidak dapat diubah, bagaimana jika suatu ketika isi string tersebut harus diubah? Java menyediakan kelas StringBuilder untuk kebutuhan tersebut. Suatu StringBuilder dapat dideklarasikan dalam beberapa cara, seperti terlihat pada potongan program berikut.

|  |
| --- |
| StringBuilder nama = new StringBuilder();  StringBuilder nama2 = new StringBuilder(nama);  StringBuilder nama3 = new StringBuilder(30);  StringBuilder nama4 = new StringBuilder(“Crystal Clear”); |

Variabel nama membuat objek StringBuilder baru dengan kapasitas awal 16 karakter. Variabel nama2 menerima variabel nama sebagai nilai awalnya. Sementara itu, variabel nama3 memiliki kapasitas awal 30, sesuai dengan masukan nilai awalnya. Kemudian, suatu StringBuilder juga dapat dibuat dengan langsung memasukkan string yang diinginkan (dalam potongan program diatas adalah Crystal Clear). Pada dasarnya, semua tipe data (termasuk objek) dapat dijadikan nilai awal bagi StringBuilder.

Seperti halnya String, StringBuilder juga memiliki perintah-perintah yang dapat digunakan untuk manipulasi isinya. Namun, karena sifat StringBuilder adalah dapat berubah (mutable), maka perubahan yang dilakukan akan mengubah isi dari StringBuilder tersebut. Tabel 8-2 menampilkan beberapa perintah pada StringBuilder.

Tabel ‎8‑2 Method pada StringBuilder

|  |  |
| --- | --- |
| Sintaks | Deskripsi |
| charAt(int index) | Mengambil satu karakter dari string sesuai indeks |
| indexOf() | Membandingkan apakah karakter atau string tertentu ada pada suatu string. Jika ada, akan mengembalikan indeks pertama yang ditemukan, jika tidak akan mengembalikan nilai -1. |
| substring() | Membuat sub-string dari suatu string. |
| append() | Menambahkan nilai diujung string. Nilai yang ditambahkan dapat berasal dari seluruh tipe data. |
| insert() | Memiliki fungsi sama seperti append, tapi nilai dapat ditambahkan pada posisi manapun. |
| delete() | Menghapus karakter pada jangkauan tertentu pada suatu string. |
| deleteCharAt() | Menghapus karakter pada satu posisi tertentu pada suatu string. |
| length() | Mengembalikan nilai panjang suatu string (banyaknya huruf pada string tersebut) |

Perintah charAt, indexOf, substring dan length pada StringBuilder bersifat sama persis dengan pada String, karena itu, contoh program berikut akan membahas perintah append, insert, delete dan deleteCharAt.

|  |  |
| --- | --- |
| package praktikum8\_5;  class StringBuilderDemo {  public static void main (String args[]){  StringBuilder sb = new StringBuilder(“Tkm”);  char tambah[] = {‘T’,’e’,’l’,’k’,’o’,’m’};  sb.insert(1,tambah,1,2);  System.out.println(sb);  sb.insert(4,’o’);  System.out.println(sb);  sb.delete(1,3);  System.out.println(sb);  sb.deleteCharAt(3);  System.out.println(sb);  sb.append(10.5);  System.out.println(sb);  sb.append(‘r’);  System.out.println(sb);  }  } | Apakah Outputnya?  ………………………… |

Program ‎8‑5 StringBuilder

Pada Program 8-4, perintah insert menyisipkan isi dari indeks 1 – 2 dari array tambah pada variabel StringBuilder sb. Dengan demikian, huruf ‘e’ dan ‘l’ akan menempati indeks 1 dan 2 dari sb. Perintah insert tidak menimpa isi dari StringBuilder, dengan demikian, huruf ‘k’ yang awalnya berada di indeks ke-1 akan tergeser menjadi indeks ke-3.

Sementara itu, perintah delete menghapus karakter pada sb mulai dari indeks pertama sampai kedua (walaupun pada sintaks tertulis 3, namun indeks ketiga tidak ikut terhapus). Perintah deleteCharAt menghapus karakter sesuai indeks yang tertulis (dalam Program 8-4 adalah indeks ketiga). Perintah append akan menggabungkan nilai yang tertulis pada ujung StringBuilder.

### Tipe Data Reference

Seperti telah disebutkan sebelumnya, tipe data reference merupakan tipe data bentukan. Berbeda dengan tipe data primitif yang memiliki hanya satu nilai literal untuk satu variabel, variabel dengan tipe data reference dapat memiliki lebih dari satu nilai literal dengan tipe yang berbeda.

Pembahasan lebih jauh mengenai objek, kelas dan konsep pemrograman yang berhubungan dengan keduanya akan didapatkan pada mata kuliah Pemrograman Berbasis Objek. Untuk saat ini, kita akan membuat suatu tipe data bentukan dengan objek dari suatu inner class.

Inner class merupakan suatu nested class, kelas yang berada di dalam kelas lain. Kelas yang berada di dalam kelas lain ini memilliki akses – jadi dapat menggunakan – variabel dari outer class (kelas yang tempat inner class bernaung). Namun, pada pembahasan kali ini inner class yang dibuat tidak akan menggunakan variabel dari outer class. Untuk lebih jelasnya, perhatikan Program 8-6 berikut.

|  |  |
| --- | --- |
| package praktikum8\_6;  import java.util.Scanner;  class PanggilMhs {  class Mahasiswa{  String NIM;  String nama;  int IPK;  }  public static void main (String args[]){  Scanner sc = new Scanner(System.in);  Mahasiswa mhs = new PanggilMhs().new Mahasiswa();  System.out.println(“Masukkan NIM mahasiswa:”);  mhs.NIM = sc.next();  System.out.println(“Masukkan nama mahasiswa:”);  mhs.nama = sc.next();  System.out.println(“Masukkan IPK mahasiswa:”);  mhs.IPK = sc.nextInt();  System.out.println(“Data mahasiswa adalah:”);  System.out.println(“NIM: ” + mhs.NIM  + “ Nama: ” + mhs.nama + “ IPK: ” + mhs.IPK);  }  } | Apakah Outputnya?  ………………………… |

Program ‎8‑6 Tipe data bentukan

Kelas Mahasiswa merupakan inner class dari kelas PanggilMhs. Kelas Mahasiswa memiliki tiga variabel dengan tipe data yang berbeda. Untuk saat ini, kelas Mahasiswa dapat dipandang sebagai suatu struktur pembentuk tipe data baru yang terdiri atas tiga unsur: NIM, nama dan IPK. Dengan demikian, tipe data yang terbentuk bukanlah String atau integer (yang merupakan tipe data dari masing-masing unsur di dalam Mahasiswa), namun tipe data baru yang terdiri dari ketiga tipe data masing-masing unsur.

Untuk menggunakan tipe data baru ini, pada kelas PanggilMhs dibuatlah sebuah objek dari Mahasiswa, yaitu mhs. Karena Mahasiswa memiliki tiga unsur data, maka variabel mhs pun memiliki tiga unsur; NIM, nama dan IPK. Untuk mengakses masing-masing unsur, variabel mhs harus memanggil unsur yang dibutuhkan dengan menggunakan notasi titik (.). Jadi, baris mhs.NIM merupakan baris perintah untuk memakai unsur NIM yang ada pada variabel mhs.

Gambar 8-1 mengilustrasikan perbedaan antara tipe data bentukan (variabel mhs) dengan variabel biasa (misal, int data). Variabel mhs terdiri atas unsur-unsur lain dalam satu kesatuan, sementara data hanya dapat menyimpan satu nilai saja.

NIM

nama

IP

String

String

int

int

mhs

data

Gambar ‎8‑1 Tipe data bentukan dan primitif

## Latihan

1. Di suatu perguruan tinggi ada 2 mahasiswa yang berimprovisasi ingin membuat program untuk menulis data diri kedua mahasiswa tersebut serta menghitung Nilai Total yang dia dapat dalam 1 semester dengan spesifikasi:

* Input data Mahasiswa (Nama dan NIM) beserta Nilai UTS, UAS, dan Tugas.
* Rumus Nilai Total = 40% x Nilai UTS + 40% x Nilai UAS + 20% x Nilai Tugas.
* Setelah data nilai total dihitung, kemudian nilai total setiap mahasiswa dioutputkan ke layar.

Buatlah program dengan spesifikasi tersebut menggunakan tipe bentukan.

1. Di suatu perusahaan dibutuhkan aplikasi untuk mencatat data karyawan serta mencatat lamanya seorang karyawan bekerja, buatlah sebuah aplikasi tersebut dengan spesifikasi:

* Input data karyawan, jam masuk, dan jam keluar.
* Lama kerja = jam keluar - jam masuk.
* Kemudian lama kerja di-*output*kan ke layar.

1. Buatlah sebuah program yang menerima input nomor telepon seluler dari *user* dan mengubahnya dalam format +62 XXX-XXXX-XXXX

Contoh:

Inputan *user* : 085634784956

Keluaran : +62 856-3478-4956

Catatan:

Hati-hati untuk beberapa kasus, misal: user tidak memasukkan no telp seluler yang valid (angka kurang atau bukan no telp Indonesia). Untuk kasus seperti itu, tampilkan pesan peringatan.

Contoh:

Inputan *user* : 085634784

Keluaran : Anda tidak memasukkan nomor yang valid

Inputan *user* : 075874784956

Keluaran : Anda tidak memasukkan nomor yang valid

# 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 (*software* *reusaibility*), 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 direpresentasikan dengan variable, seringkali disebut field, dan dideklarasikan di dalam kelas, tapi di luar method. Field inilah yang disebut dengan instance variable. Instance variable merupakan anggota dari suatu kelas, dan menyimpan data dari suatu objek, jadi objek yang satu dengan objek yang lain akan memiliki nilai instance variable yang berbeda. Untuk lebih jelasnya, 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 namaBarang dan harga. Pada dasarnya, instance variable merupakan variabel global, karena itu method-method yang berada di dalam kelas VariableScope dapat menggunakan namaBarang dan harga. Namun, karena instance variable berasosiasi dengan objek tertentu, maka nilai 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”. Demikian juga untuk harga, antara objek var1 dan var2 nilai dari variabel harganya berbeda. 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 variabel seperti itu merupakan variabel static. Ketika sebuah variabel dideklarasikan sebagai static, maka variabel tersebut dapat diakses sebelum ada satupun 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 dan tampil.

|  |
| --- |
| 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 mendeklarasikan method. 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, method tampil 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);  PassByValue pbv = 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 method yang 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{  void rek(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.

**DAFTAR PUSTAKA**

|  |
| --- |
| * Anonim. 2014. Modul Implementasi Algoritma. Fakultas Ilmu Terapan-Universitas Telkom, Bandung. * Duke images. <https://duke.kenai.com/thumbsup/DukeWithHelmet.png> diakses tanggal 06 September 2016. * Deitel, Paul J dan Deitel, Harvey M. 2012. Java How to Program 9th Ed. Prentice Hall, Boston. * Gupta, Mala. 2013. OCA Java SE 7 Programmer I Certification Guide - Prepare For The 1Z0-803 Exam. Manning Publications, Co, New York. * Hortsmann, Cay. 2010. Big Java: Compatible with Java 5,6 and 7, 4th edition. John Wiley&Sons, New York. * Schildt, Herbert. 2007. Java: The Complete Reference, 7th Ex. Mc Graw Hill, New York. * <http://www.oracle.com/technetwork/java/index.html> diakses tanggal 10 Juli 2016 * Nugroho Adi. 2011. Perancangan dan Implementasi Sistem Basis Data. Penerbit Andi. Yogyakarta |