2020

Konsep Dasar Algoritma & Pemrograman Dengan Bahasa Java

Evi Pratiwi

Konsep Dasar Algoritma dan Pemrograman Dengan Bahasa Java

Undang-Undang No. 28 Tahun 2014 Tentang Hak Cipta

Fungsi dan sifat hak cipta Pasal 4

Hak Cipta sebagaimana dimaksud dalam Pasal 3 huruf a merupakan hak eksklusif yang terdiri atas hak moral dan hak ekonomi.

Pembatasan Perlindungan Pasal 26

Ketentuan sebagaimana dimaksud dalam Pasal 23, Pasal 24, dan Pasal 25 tidak berlaku terhadap :

i. penggunaan kutipan singkat Ciptaan dan/atau produk Hak Terkait untuk pelaporan peristiwa aktual yang ditujukan hanya untuk keperluan penyediaan informasi aktual;
ii. Penggandaan Ciptaan dan/atau produk Hak Terkait hanya untuk kepentingan penelitian ilmu pengetahuan;
iii. Penggandaan Ciptaan dan/atau produk Hak Terkait hanya untuk keperluan pengajaran, kecuali pertunjukan dan Fonogram yang telah dilakukan Pengumuman sebagai bahan ajar; dan
iv. penggunaan untuk kepentingan pendidikan dan pengembangan ilmu pengetahuan yang memungkinkan suatu Ciptaan dan/atau produk Hak Terkait dapat digunakan tanpa izin Pelaku Pertunjukan, Produser Fonogram, atau Lembaga Penyiaran.

Sanksi Pelanggaran Pasal 113

1. Setiap Orang yang dengan tanpa hak melakukan pelanggaran hak ekonomi sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf i untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 1 (satu) tahun dan/atau pidana denda paling banyak Rp 100.000.000 (seratus juta rupiah).
2. Setiap Orang yang dengan tanpa hak dan/atau tanpa izin Pencipta atau pemegang Hak Cipta melakukan pelanggaran hak ekonomi Pencipta sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf c, huruf d, huruf f, dan/atau huruf h untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 3 (tiga) tahun dan/atau pidana denda paling banyak Rp 500.000.000,00 (lima ratus juta rupiah).
3. Setiap Orang yang dengan tanpa hak dan/atau tanpa izin Pencipta atau pemegang Hak Cipta melakukan pelanggaran hak ekonomi Pencipta sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf a, huruf b, huruf e, dan/atau huruf g untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 4 (empat) tahun dan/atau pidana denda paling banyak Rp 1.000.000.000,00 (satu miliar rupiah).
4. Setiap Orang yang memenuhi unsur sebagaimana dimaksud pada ayat (3) yang dilakukan dalam bentuk pembajakan, dipidana dengan pidana penjara paling lama 10 (sepuluh) tahun dan/atau pidana denda paling banyak Rp 4.000.000.000,00 (empat miliar rupiah).

Konsep Dasar Algoritma dan Pemrograman Dengan Bahasa Java

Evi Lestari Pratiwi

Poliban Press

KONSEP DASAR ALGORITMA DAN PEMROGRAMAN DENGAN BAHASA JAVA

Penulis :
Evi Lestari Pratiwi

ISBN Elektronis :
978-623-7694-45-8

Editor dan Penyunting :
Adi Pratomo

Desain Sampul dan Tata letak :
Rahma Indera; Eko Sabar Prihatin

Penerbit :
POLIBAN PRESS
Anggota APPTI (Asosiasi Penerbit Perguruan Tinggi Indonesia)
no.004.098.1.06.2019
Cetakan Pertama, 2020

Redaksi :
Politeknik Negeri Banjarmasin, Jl. Brigjen H. Hasan Basry,
Pangeran, Komp. Kampus ULM, Banjarmasin Utara
Telp : (0511)3305052
Email : press@poliban.ac.id

Diterbitkan pertama kali oleh :
Poliban Press, Banjarmasin, November 2020

KATA PENGANTAR

Puji syukur kehadirat Allah SWT atas limpahan rahmat dan karunianya sehingga buku Konsep Dasar Algoritma dan Pemrograman Dengan Bahasa Java tahun 2020 telah dapat diselesaikan. Buku ini merupakan pengantar bagi mahasiswa Diploma di Politeknik Negeri Banjarmasin.

Terimakasih disampaikan kepada Joni Riadi S.ST., M.T. selaku Direktur Politeknik Negeri Banjarmasin dan Nurmahaludin, S.T., M.T. selaku Ketua Pusat Penelitian dan Pengabdian Masyarakat beserta sekretaris dan staf. Terimakasih juga disampaikan kepada Faris Ade Irawan, Reza Fauzan, Eko Sabar Prihatin dan Rahma Indera yang telah berkontribusi dalam editing serta seluruh tim Poliban Press dan semua pihak yang telah ikut membantu dalam penyelesaian buku ini.

Kami menyadari masih terdapat kekurangan dalam buku ini untuk itu kritik dan saran terhadap penyempurnaan buku ini sangat diharapkan. Semoga buku ini dapat memberi maanfaat bagi semua pihak.

Banjarmasin, Oktober 2020

Poliban Press

PRAKATA

Puji syukur ke hadirat Allah SWT, Tuhan Yang Maha Esa, karena berkat rahmat, taufik serta hidayah-Nya, penulis dapat menyelesaikan bahan ajar Konsep Dasar Algoritma dan Pemrograman Dengan Bahasa Java. Bahan ajar ini disusun untuk mata kuliah yang berkaitan dengan Algoritma dan Pemrograman, yang dilengkapi dengan latihan soal agar mahasiswa dapat lebih memahami materi yang terkait.

Penulis mengucapkan terimakasih yang teramat dalam kepada pihak yang telah memberikan dukungan penuh terutama Jurusan Administrasi Bisnis, Program Studi Manajemen Infomatika, dan unit P3M Politeknik Negeri Banjarmasin, dan pihak yang tidak bisa diucapkan satu per satu, sehingga bahan ajar ini dapat tersusun dengan baik.

Dalam penulisan bahan ajar ini tentu masih banyak kekurangan. Oleh karena itu, saran dan kritik yang membangun agar dapat menyempurnakan bahan ajar menjadi lebih baik lagi. Semoga bahan ajar ini dapat bermanfaat bagi kita semua.

Banjarmasin, Oktober 2020
Penulis

Evi Lestari Pratiwi, M.Kom

DAFTAR ISI

BAB 1
MENGENAL ALGORITMA DAN PEMROGRAMAN

Capaian Pembelajaran :

1. Mampu memahami pengertian algoritma
2. Mampu memahami kriteria algoritma
3. Mampu menjelaskan struktur algoritma
4. Mampu memahami teks algoritma

1.1. Pengertian Algoritma

Asal kata Algoritma berasal dari nama Abu Ja'far Mohammed Ibn Musa al-Khowarizmi, ilmuan Persia yang menulis kitab al jabr w'al-muqabala (*rules of restoration and reduction*) sekitar tahun 825 M.

Algoritma adalah urutan langkah logis tertentu untuk memecahkan suatu masalah. Yang ditekankan adalah urutan langkah logis, yang berarti algoritma harus mengikuti suatu urutan tertentu, tidak boleh melompat-lompat. (Dari Microsoft Press Computer and Internet Dictionaary 1997, 1998)

Alur pemikiran dalam menyelesaikan suatu pekerjaan yang dituangkan secara tertulis. Yang ditekankan pertama adalah alur pikiran, sehingga algoritma seseorang dapat juga berbeda dari algoritma orang lain. Sedangkan penekanan kedua adalah tertulis, yang artinya dapat berupa kalimat, gambar, atau tabel tertentu. (Dari Algoritma dan Struktur Data dengan C, C++, dan Java oleh Moh Sjukani hal 1).

Contoh Algoritma dalam kehidupan nyata:

1. Jika seorang ingin memasak atau membuat kue, baik itu melihat resep ataupun tidak pasti akan melakukan suatu langkah-langkah tertentu sehingga masakannya atau kuenya jadi.
2. Jika seseorang ingin mengirim surat kepada kenalannya di tempat lain, langkah yang harus dilakukan adalah:
   a. Menulis surat
   b. Surat dimasukkan ke dalam amplop tertutup
   c. Amplop ditempeli perangko secukupnya.
   d. Pergi ke Kantor Pos terdekat untuk mengirimkannya.

Dalam bidang komputer, algoritma sangat diperlukan dalam menyelesaikan berbagai masalah pemrograman, terutama dalam komputasi numeris. Tanpa algoritma yang dirancang baik maka proses pemrograman akan menjadi salah, rusak, atau lambat dan tidak efisien. Pelaksana algoritma adalah Komputer. Manusia dan komputer berkomunikasi dengan cara: manusia memberikan perintah- perintah kepada komputer berupa instruksi-instruksi yang disebut program.

Alat yang digunakan untuk membuat program tersebut adalah bahasa pemrograman. Bahasa pemrograman sangat bermacam-macam: C, C++, Pascal, Java, C#, Basic, Perl, PHP, ASP, JSP, J#, J++ dan masih banyak bahasa lainnya. Dari berbagai bahasa pemrograman cara memberikan instruksinya berbeda-beda namun bertujuan menghasilkan output yang sama.

1.2. Kriteria Algoritma

Kriteria Algoritma Menurut Donald E. Knuth:

1. *Input*

   Algoritma dapat memiliki nol atau lebih inputan dari luar. Artinya, suatu algoritma itu dimungkinkan tidak memiliki masukan secara langsung dari pengguna tetapi dapat juga memiliki beberapa masukan.

2. *Output*
   Algoritma harus memiliki minimal satu buah output keluaran. Suatu algoritma yang tidak memiliki keluaran (*output*) adalah suatu algoritma yang sia-sia, yang tidak perlu dilakukan. Algoritma dibuat untuk tujuan menghasilkan sesuatu yang diinginkan, yaitu berupa hasil keluaran.

3. *Definiteness* (pasti)
   Algoritma memiliki instruksi-instruksi yang jelas dan tidak ambigu. Setiap baris aksi/pernyataan dalam suatu algoritma harus pasti, artinya tidak menimbulkan penafsiran lain bagi setiap pembaca algoritma, sehingga memberikan keluaran yang sesuai dengan yang diharapkan oleh pengguna.

4. *Finiteness* (ada batas)
   Algoritma harus memiliki titik berhenti (*stopping role*). Algoritma harus dapat dijamin akan berhenti setelah melakukan sejumlah langkah proses.

5. *Effectiveness* (tepat dan efisien)
   Algoritma sebisa mungkin harus dapat dilaksanakan dan efektif. Suatu algoritma tidak terdapat suatu aksi yang tidak perlu dilakukan. Contoh instruksi yang tidak efektif adalah: A = A + 0 atau A = A * 1.

1.3. Struktur Algoritma

Struktur dasar dari algoritma terdiri dari empat macam struktur, yaitu :

1. Algoritma Sekuensial
   Struktur sekuensial atau *Sequential* terdiri dari sebuah instruksi atau blok yang berisi langkah urut saja, satu langkah diikuti oleh langkah lainnya. Sebuah instruksi dikerjakan setelah instruksi sebelumnya dikerjakan.

2. Algoritma Percabangan

   Sruksur seleksi menyatakan pemilihan langkah yang didasarkan oleh suatu kondisi atau pengambilan suatu keputusan.

3. Algoritma Perulangan

   Struktur ini memberikan suatu perintah atau tindakkan yang dilakukan beberapa kali. Misalnya jika teman mau menuliskan kata “Belajar Algoritma dan Pemrograman” sebanyak sepuluh kali. Akan lebih efisien jika teman menggunakan sturktur ini dari pada sekedar menuliskannya berturut-turut sebanyak sepuluh kali.

4. Algoritma Paralel

   Beberapa instruksi dikerjakan secara bersamaan.

1.4. Prinsip Kerja Algoritma

Algoritma merupakan deskripsi pelaksanaan suatu proses, sehingga proses akan dikerjakan sesuai dengan algoritma yang telah ditulis. Urutan langkah dalam algoritma disusun dalam sederatan aksi, yaitu masukan, proses, dan keluaran.

Gambar 1 - Prinsip Kerja Algoritma

1.5. Teks Algoritma

Teks algoritma selalu disusun dalam tiga bagian (blok), yaitu bagian judul algoritma, bagian deklarasi, dan bagian deskripsi algoritma.

1. Judul Algoritma

   Judul algoritma terdiri dari nama dan penjelasan (spesifikasi) algoritma. Nama yang digunakan sebaiknya singkat dan informatif. Penjelasan di bawah judul merupakan spesifikasi dari

algoritma yang akan dikerjakan, da nisi algoritma harus sesuai dengan spesifikasi yang telah dituliskan.

Contoh :

```
Algoritma LUAS_PERSEGI_PANJANG
{   Menghitung luas persegi panjang dengan masukan ukuran panjang dan ukuran lebar, lalu mencetak hasil luar persegi panjang ke piranti keluaran  }
```

2. Deklarasi Algoritma

   Bagian yang digunakan untuk mendeklarasikan atau mendefinisikan semua nama yang digunakan dalam deskripsi algoritma. Bagian deklarasi ini sering disebut dengan kamus. Sesuai dengan kegunaan kamus pada umumnya adalah tempat rujukan dalam mengetahui arti dan penggunaan suatu kata.

   Deklarasi merupakan tempat untuk mendefinisikan berbagai macam nama, yaitu :

   a. Nama tipe

      ```
      { Nama tipe, hanya untuk tipe yang bukan tipe dasar }
      Type Titik : <X:real, Y:real> { koordinat pada sumbu kartesian  }
      ```

   b. Nama konstanta

      ```
      {  Deklarasi Konstanta  }
      Constant Phi  =  3.14
      Constant N : Integer = 100
      ```

   c. Nama peubah (nama variable)

      ```
      {  Deklarasi Variable  }
      Jumlah         : Integer
      Rata_rata      : Real
      Jumlah_Anak    : Real
      Nilai_huruf    : Character
      Mat_A          : Matriks
      ```

d. Nama fungsi

```
{ Spesifikasi Fungsi, menyebutkan nama fungsi, domain dan range }
function Konversi_Real_Ke_Integer (input i:real) → integer
{ Mengkonversi harga i yang bertipe real menjadi harga ekivalen yang bertipe integer }
```

e. Nama prosedur

```
{ Spesifikasi Prosedur, menyebutkan nama, parameter, kondisi awal, kondisi akhir dan proses }
procedure Tukar(input/output A:integer, input/output B:integer)
{ Kondisi awal : A dan B terdefinisi, A=a dan B=b }
{ Kondisi akhir : A=b dan B=a }
{ Proses : Mempertukarkan nilai A dan B }
```

3. Deskripsi Algoritma

Deskripsi algoritma merupakan bagian inti dari suatu algoritma, karena urutan aksi penyelesaian masalah yang akan dilaksanakan dituliskan dalam bagian deskripsi. Urutan penulisan aksi akan mempengaruhi hasil akhir dari algoritma.

```
DESKRIPSI : {Semua langkah atau aksi algoritma dituliskan di sini}
```

Contoh lengkap teks algoritma :

```
Algoritma NAMA_ALGORITMA
{Penjelasan mengenai algoritma, yang berisi uraian singkat mengenai apa yang dilakukan oleh algoritma}
DEKLARASI
{Semua nama yang dipakai, meliputi nama tipe, nama tetapan, nama peubah, nama prosedur dan nama fungsi didefinisikan di sini}
DESKRIPSI :
{Semua langkah atau aksi algoritma dituliskan di sini}
```

1.6. Latihan

Ambil suatu kegiatan Anda sehari-hari, kemudian buatlah algoritmanya yang mengandung sekuensial, percabangan, dan perulangan.

BAB 2
SKEMA DAN STRUKTUR ALGORITMA

Capaian Pembelajaran :

1. Mampu memahami skema algoritma
2. Mampu memahami teks algoritma (*pseudocode*)
3. Mampu memahami *Flowchart*

2.1. Skema Algoritma

Skema algoritma merupakan alur pemikiran dalam menyelesaikan suatu pekerjaan yang dituangkan secara tertulis. Penyajian algoritma secara garis besar dapat dibagi dalam dua bentuk penyajian yaitu tulisan dan gambar. Algoritma yang disajikan dengan tulisan yaitu dengan struktur bahasa tertentu.

Hal yang penting untuk dipahami adalah logika dalam berpikir bagaimana cara untuk memecahkan masalah pemrograman yang akan dibuat. Sebagai contoh, banyak permasalahan yang mudah jika diselesaikan secara tertulis, tetapi cukup sulit jika diterjemahkan ke dalam pemrograman. Dalam hal ini, algoritma dan logika pemrograman akan sangat penting dalam pemecahan masalah.

Kunci penyelesaian masalah dengan bantuan komputer adalah :

1. Memahami masalah domain algoritma
   Setiap persoalan harus dicari jawaban melalui proses analisis untuk mengetahui akar dari masalah sehingga jalan keluar terbaik dapat ditentukan.
2. Persiapan dalam menyusun langkah penyelesaian domain algoritma
   Menentukan jenis data atau variabel yang diperlukan, menentukan parameter masukan, proses, dan keluaran, sehingga terbentuk langkah penyelesaian algoritma, fungsi, dan prosedur yang diperlukan.

3. Input domain komputer
   Data awal yang harus tersedia, untuk dibaca dan selanjutkan akan diproses.
4. Proses domain komputer
   Mengolah data input sesuai dengan formula atau logika algoritma
5. Output domain komputer
   Jika diperlulukan untuk ditampilkan di monitor atau printer, atau dapat juga dipergunakan kembali untuk input dalam proses yang baru.

Contoh soal algoritma dan penyelesaiannya :

1. Susunlah langkah – langkah untuk menentukan suatu bilangan ganjil diantara 10 sampai 30, kemudian tampilkan bilangan ganjilnya.

   Persiapan : Bilangan ganjil yang terletak diantara 10 dan 30 adalah bilangan 11,13,15, dan seterusnya. Namun yang akan ditampilkan kecuali bilangan 21 dan 27. Sehingga output/keluaran yang diharapkan dari algoritma tersebut yaitu bilangan ganjil antara 10 sampai 30 kecuali bilangan 21 dan 27

   Input / data : bilangan bulat

   Proses : tes fungsi modulus (sisa pembagian dua), jika sisa = 1 berarti ganjil, jika bilangan 21 atau 27 berarti bukan bilangan ganjil

   Output : berupa tampilan bilangan ganjil

2. Buatlah langkah untuk mengimkan pesan elektronik.

   Persiapan : masuk ke situs layanann pesan elektronik, jika sudah memiliki, jika belum maka harus mendaftar terlebih dahulu.

   Input / data : masukan alamat pesan elektronik penerima, masukan subjek dari pesan, dan tulis pesan baru.

   Proses : menekan tombol kirim

Output : berupa pesan terkirim yang terdapat di surat elektronik yang dimiliki

3. Susunlah langkah – langkah untuk menghitung konversi suhu dari Celsius menjadi Reamur dan Fahrenheit, kemudian tampilkan hasil yang diperoleh

   Persiapan : mengetahui besarnya nilai suhu Celsius, kemudian menentukan rumus konversi dari Celsius ke Reamur dan Fahrenheit

   Input / data : suhu dalam Celsius

   Proses : R = 4/5 * C dan F = 9/5 * C + 32

   Output : suhu dalam Reamur dan Fahrenheit

2.2. Pseudocode

2.2.1. Pengertian Pseudocode

Pseudocode adalah kode atau tanda yang menyerupai (pseudo) atau merupakan pejelasan cara menyelesaikan suatu masalah. Pseudcode sering digunakan oleh manusia untuk menuliskan algoritma.

Pseudocode menggunakan simbol – symbol yang mirip atau menyerupai kode program dengan bahasa pemrograman tertentu. Pseudocode menggunakan kata – kata yang mempunyai tujuan untuk menjelaskan alur logika suatu masalah, sehingga memudahkan untuk pembuatan program.

Pseudocode dikatakan mirip dengan kode program dalam bahasa pemrograman, tetapi tidak spesifik pada salah satu bahasa pemrograman tertentu, sehingga algoritma yang disajikan dapat dikonversikan ke dalam semua bahasa pemrograman.

2.2.2. Fungsi Pseudocode

Pseudocode mempunyai fungsi yang berguna untuk mempermudah penulisan dan pemetaan dari sebuah algoritma. Karena pada dasarnya, seorang programmer sebelum membuat dan

menjalankan sebuah program atau aplikasi, ia terlebih dahulu harus mempunya susunan atau mempunyai pemecahan terhadap masalah yang akan ia buat biasanya disebut dengan algoritma.

Berikut ini beberapa fungsi dari pseudocode:

1. Pseudocode memiliki fungsi sebagai alat dokumentasi
2. Pseudocode dapat mempemudah penggunanya dalam memahami serta memperjelas cara menyelesaikan masalanya.
3. Pseudocode dapat membantu penggunanya dalam menuliskan sebuah algoritma yang akan dibuatnya.
4. Pseudocode memiliki aspek yang sangat ringkas dan mudah karena pseudocode tidak bergantung pada suatu sistem tertentu dan itulah yang menjadikannya prinsip utama dari sebuah algoritma.

Tujuan dari pseudocode itu sendiri adalah agar manusia khususnya para programmer bisa dengan mudah memahami dibandingkan dengan menggunakan bahasa pemrograman yang umumnya digunakan, terlebih lagi aspek pseudocode relatif ringkas dan tidak memiliki ketergantungan dengan suatu sistem tertentu yang merupakan prinsip utama dalam sebuah algoritma.

2.2.3. Struktur Pseudocode

Secara umum struktur penulisan pseudocode dibagi menjadi 3 bagian, yaitu :

1. Bagian judul

   Bagian judul iberisi judul algoritma. Biasanya diawali dengan kata “program”. Kemudian diikuti dengan nama algoritma. Pada umumnya, nama algoritma hanya terdiri dari satu kata, jika nama yang terkandung lebih dari satu kata penulisan disatukan. Artinya jika terdiri lebih dari dua kata, maka sapasi ditiadakan.

   Berikut ini aturan penulisan nama judul:

a. Judul tidak boleh terdapat spasi. Spasi bisa diganti dengan karakter "_" (underscore).
b. Judul tidak boleh diawali dengan angka.
c. Judul tidak boleh menggunakan istilah – istilah yang biasa digunakan sebagai keyword di bahasa pemrograman.
d. Judul boleh menggunakan huruf besar, huruf kecil, dan kombinasinya selama tidak melanggar aturan diatas.

2. Bagian deskripsi

   Bagian deskripsi ini digunakan untuk mendefinisikan atau mendeklarasikan jenis – jenis variabel yang akan digunakan dalam proses algoritma. Di dalam pemograman komputer sendiri terdapat beberapa variabel, seperti bilangan bulat, desimal, pecahan, dan lain sebagainya.

3. Bagian implementasi

   Bagian implementasi ini bisa dikatakan sebagai bagian inti atau utama, yang mana merupakan bagian jalannya sebuah algoritma. Pada bagian ini terdapat sekumpulan perintah algoritma, perintahnya pun bisa berupa runtutan, kondisional ataupun perulangan.

2.2.4. Gaya Penulisan Pseudocode

Berikut merupakan contoh penulisan algoritma dalam bentuk pseudocode untuk algoritma pencarian sisi miring suatu segitiga.

```
PROGRAM CariSisiMiring
DEKLARASI
int a, b
float c
ALGORITMA
read(a, b)
c = sqrt(a*a + b*b)
write("Sisi miring: ", c)
```

Penjelasan pseudocode:

1. Judul dari algoritma adalah CariSisiMiring yang dituliskan di awal pseudocode.
2. Pada bagian deklarasi, terdapat 3 buah variabel, yaitu a, b dan c. Variabel a dan b dideklarasikan sebagai variabel berjenis int atau bilangan bulat, sedangkan variabel c dideklarasikan sebagai variable berjenis float atau bilangan pecahan.
3. Pada bagian isi, terdapat 3 buah perintah, yaitu:
   a. Perintah read(a, b). Perintah ini digunakan untuk meminta masukan dari pengguna. Pada aplikasi yang telah jadi, yang memberikan masukan adalah pengguna aplikasi. Masukkan berupa 2 buah angka dapat di-inputkan melalui keyboard.
   b. Perintah c = sqrt(a*a + b*b) adalah perintah untuk memberikan nilai variabel c. Nilai variabel c didapatkan dari perhitungan akar kuadrat dari a kuadrat ditambah b kuadrat.
   c. Perintah write("Sisi miring: ", c) merupakan perintah untuk menampilkan teks “Sisi miring: ” diikuti dengan isi dari variabel c. Variabel c dicetak di luar tanda petik sebagai penanda bahwa yang dicetak bukan teks “c” melainkan isi dari variabel c.

2.2.5. Contoh Pseudocode

Soal:

Dibaca nama karyawan dan gaji pokok. Gaji bersih yang diterima pegawai adalah gaji bersih = gaji pokok + tunjangan – pajak. Tunjangan karyawan dihitung 20% dari gaji pokok, sedangkan pajak adalah 15% dari gaji pokok ditambah tunjangan. Nama karywan dan gaji bersihnya dicetak ke piranti keluaran. Buatlah pseudocode-nya.

Algoritma :

```
Input (nama)
Input (gaji pokok)
Tunjangan ← 20% * Gaji_pokok
```

```
Pajak ← 15% * (Gaji _pokok + Tunjangan)
Gaji_Bersih = Gaji_pokok + Tunjangan – Pajak
Output (Nama, Gaji_Bersih)
```

2.3. Flowchart

2.3.1. Pengertian Flowchart

Flowchart adalah bentuk gambar/diagram yang mempunyai aliran satu atau dua arah secara sekuensial. Flowchart digunakan untuk merepresentasikan maupun mendesain program. Oleh karena itu flowchart harus bisa merepresentasikan komponen-komponen dalam bahasa pemrograman. Baik flowchart maupun algoritma bisa dibuat sebelum maupun setelah pembuatan program. Flowchart dan Algoritma yang dibuat sebelum membuat program digunakan untuk mempermudah pembuat program untuk menentukan alur logika program, sedangkan yang dibuat setelah pembuatan program digunakan untuk menjelaskan alur program kepada orang lain.

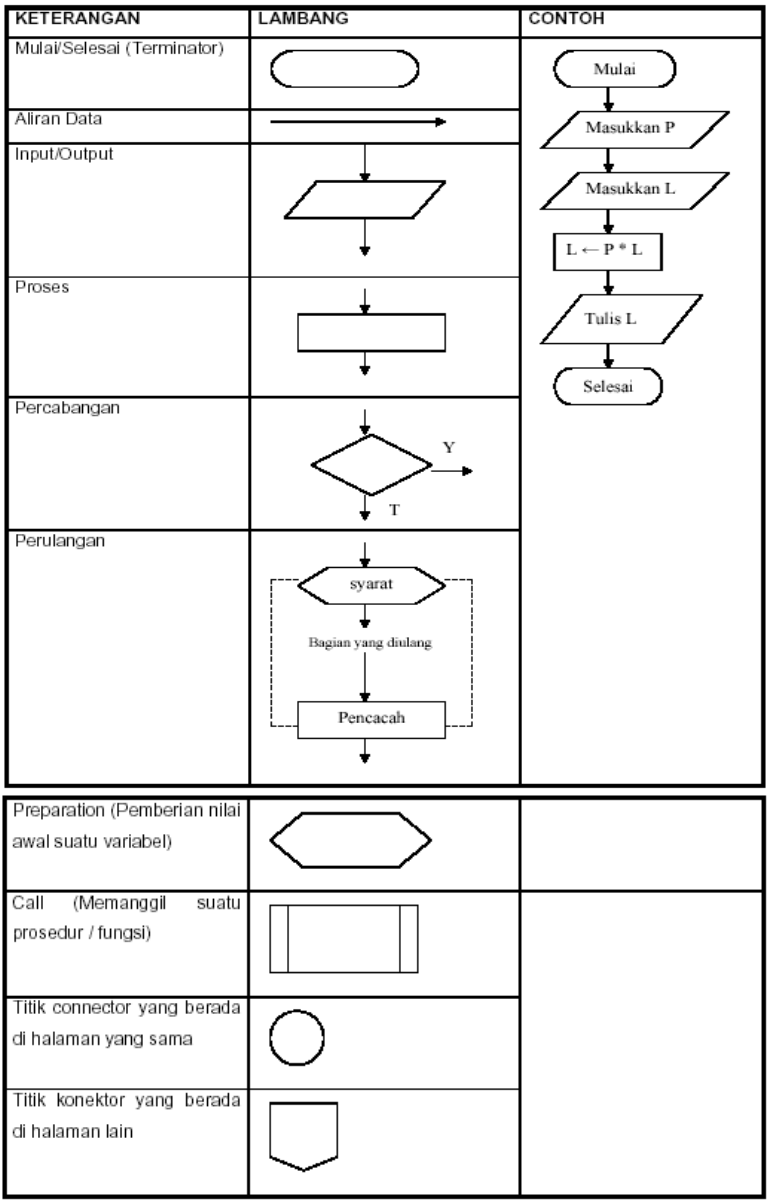
Tujuan dari flowchart adalah untuk menggambarkan suatu tahapan penyelesaian masalah secara sederhana, terurai, rapi, dan jelas dengan menggunakan simbol – simbol standar.

Beberapa petunjuk yang harus diperhatikan dalam membuat flowchart, seperti :

1. Flowchart digambarkan dari halaman atas ke bawah dan dari kiri ke kanan,
2. Aktivitas yang digambarkan harus didefinisikan secara hati – hati dan definisi harus dapat dimengerti oleh pembacanya,
3. Kapan aktivitas dimulai dan berakhir harus ditentukan secara jelas dan terinci,
4. Setiap langkah dari aktivitas harus diuraikan dengan menggunakan deskripsi kata kerja,
5. Setiap langkah dari aktivitas harus pada urutan yang benar,
6. Lingkup aktivitas yang sedang digambarkan harus ditelusuri dengan hati – hati,
7. Gunakan simbol yang standar.

2.3.2. Simbol-Simbol Flowchart

Simbol yang digunakan untuk menggambarkan algoritma dalam bentuk diagram alir yaitu:



Gambar 2 – simbol flowchart

2.3.3. Contoh Flowchart

Berikut beberapa contoh flowchart:

1. Menghitung luas persegi panjang

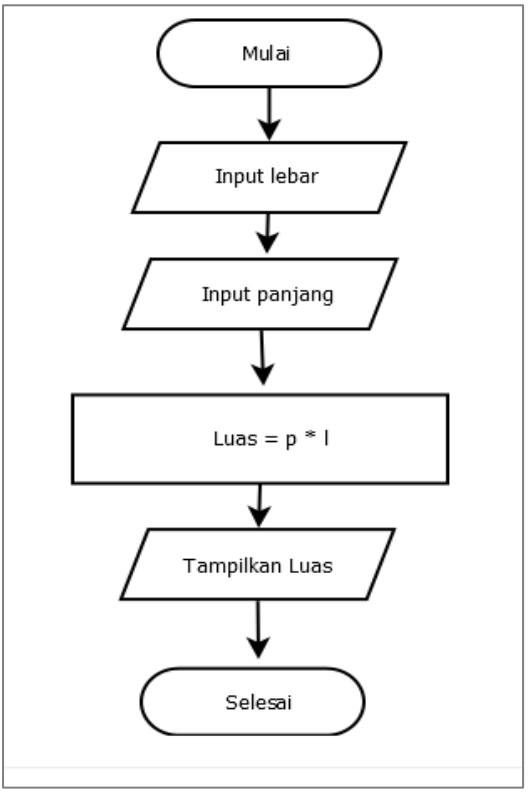
   Analisis :

   a. Input : p (panjang) dan l (lebar)
   b. Luas persegi panjang : L = p * l

   Algoritma:

   a. Inputkan panjang
   b. Inputkan lebar
   c. Rumus untuk menghitung L yaitu L = p*l
   d. Nilai L (luas) akan dcetak sebagai output ke perangkat output (keluaran)

   Flowchart Luas persegi panjang :



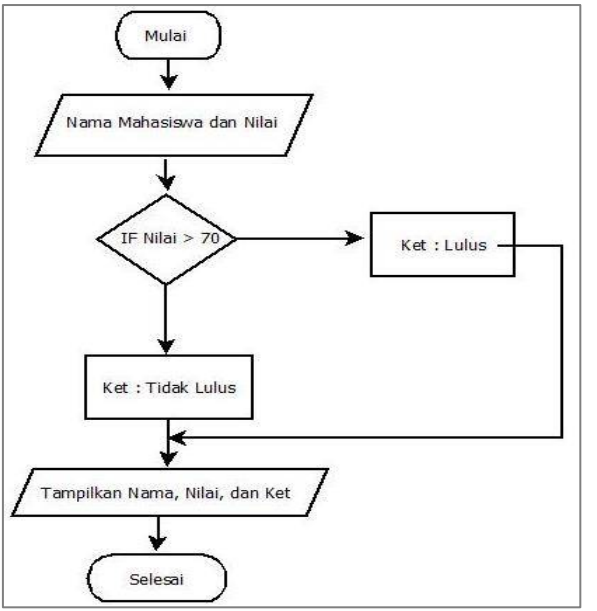
Gambar 3 – Contoh flowchart luas persegi Panjang

2. Menentukan kelulusan mahasiswa

   Analisis :

   a. Nama mahasiswa dan nilai (sudah terbaca)
   b. Kalau mahasiswa mendapat nilai >70 maka ket " lulus"
   c. Kalau mahasiswa mendapat nilai <70 maka ket "tidak lulus"
   d. Data nama, nilai dan keterangan akan ditampilkan

   Flowchart :



Gambar 4 – contoh flowchart menentukan kelulusan mahasiswa

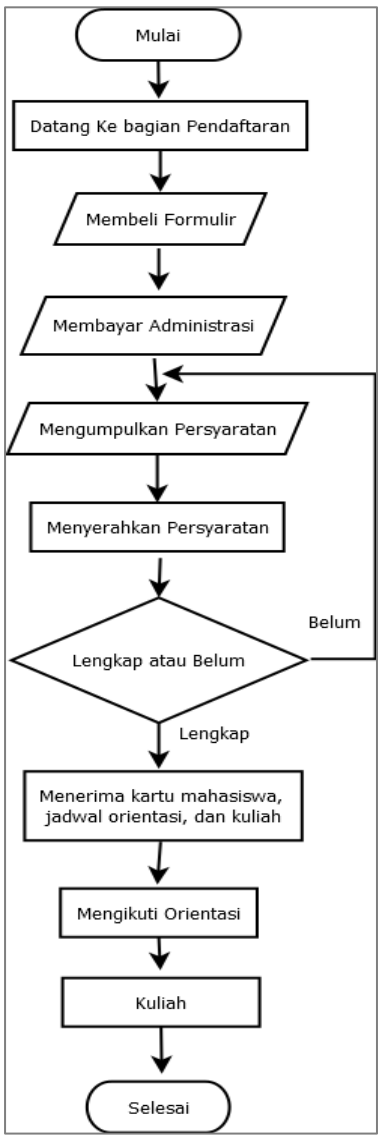
3. Penerimaan Mahasiswa Baru

   Analisis :

   a. Calon Mahasiswa datang ke Bagian Pendaftaran.
   b. Membeli dan mengisi formulir
   c. Membayar uang kuliah semester awal (sesuaikan dengan kampus)
   d. Mengumpulkan dan menyerahkan persyaratan

e. Jika persyaratan sudah lengkap? Jika tidak maka akan kembali lagi ke proses mengumpulkan persyaratan.
f. Menerima kartu mahasiswa, jadwal orientasi, dan Kuliah.
g. Mengikuti orientasi
h. Kuliah

Flowchart :



Gambar 5 – contoh flowchart Penerimaan Mahasiswa Baru

2.4. Latihan

Buatlah algoritma yang disajikan dalam flowchart dan pseudocode untuk menentukan suatu bilangan genap atau ganjil dari sebuah bilang bulat yang dibaca dari keyboard.

Input : Suatu bilangan (Bilangan)

Proses : Bilangan MOD 2 = 0 → "Genap"
Bilangan MOD 2 = 1 → "Ganjil"

Output : Genap atau Ganjil

BAB 3
TIPE DATA, NAMA, DAN NILAI

Capaian Pembelajaran :

1. Mampu menjelaskan dan membedakan tipe data yang ada
2. Mampu menentukan tipe data dari suatu nama dan algoritma

3.1. Tipe Data

Setiap data memiliki tipe data, tipe data menentukan jenis data apa yang disimpan berikut operasi apa saja yang bisa dilakukan. Dengan kata lain tipe data merupakan pengelompokkan atau klasifikasi data berdasarkan isi dan sifat dari data tersebut. Tipe data dalam bidang komputer sanga penting karena jenis data yang dapat diolah atau diproses oleh komputer untuk memenuhi kebutuhan pemrograman sangat ditentukan oleh tipe data.

Kesesuaian dalam menentukan tipe data pada variabel atau konstanta akan sangat menentukan sumber daa komputer yang dipakai. Saat kamu merancang suatu program dan algoritma adalah pastikan kamu memilih tipe data yang tepat demi menghasilkan program yang efisien dan optima kinerjanya.

Tipe data memiliki banyak jenis ang tersedia, sangat tergantung pada bahasa pemrograman ang kamu gunakan. Setiap bahasa pemrograman umumnya sudah menyediakan tipe data yang bisa kamu gunakan, tipe data yang disediakan oleh bahasa pemrograman bisa saja berbeda-beda. Secara umum tipe data terbagi menjadi tiga kelompok yaitu tipe data primitive, tipe data composite dan tipe data abstrak.

Setiap data memiliki jenis atau kategori tipe data tersendiri, tipe data dibagi menjadai dua kategori, yaitu Tipe Dasar dan Tipe Bentukan.

3.1.1. Tipe Data Dasar

Tipe data dasar adalah tipe data yang sudah siap untuk digunakan. Terdiri dari :

1. Bilangan Bulat (Integer)

Bilangan Atau Angka yang Tidak memiliki Titik Desimal atau pecahan (8, 10, +255, -34, -1024), tipe dituliskan sebagai int atau Integer didalam database, dan operasi aritmatik tambah +, Kurang -, kali *, bagi /, sisa hasil bagi %.

Kelompok	Operator	Arti	Contoh
Aritmetika	*	Perkalian	6 * 9
	/	Pembagian	9 / 6
	+	Penjumlahan	5 + 7
	-	Pengurangan	7 – 5
	^	Perpangkatan	2 ^ 3
	Div	Hasil bagi bulat	4 Div 8
	Mod	Sisa bagi	4 Mod 8
	Abs	Harga mutlak	Abs(-7)

Tipe Data	Ukuran	Tempat Rentang Nilai
Byte	1 Byte	0 s/d +255
Shortint	1 Byte	-28 s/d +127
Integer	2 Byte	-32768 s/d 32767
Word	2 Byte	0 s/d 65535
Longint	4 Byte	2147483648 s/d 2147483647

2. Bilangan Biasa (Riil)

Bilangan atau angka yang memiliki titik desimal atau pecahan (bisa diakatan kebalikan dari bilangan bulat), tipe ini dituliskan sebagai real atau float didalam database, dan dapat diimplementasikan dengan operasi aritmatik dan pembandingan. Contoh bilangan riil ini seperti 0.0003, 235.45, +1023.55, -987.3456.

Tipe Data	Ukuran	Tempat Rentang Nilai
Real	6 Byte	2.9 x 10-39 s/d 1.7 x1038
Single	4 Byte	2.9 x 10-39 s/d 1.7 x1038
Double	8 Byte	5.0 x 10-324 s/d 1.7 x10308
Extended	10 Byte	3.4 x 10-4932 s/d 1.1 x104932
Comp	8 Byte	3.4 x 10-4932 s/d 1.1 x104932

3. Bilangan Tetap (Const)

Tipe bilangan yang nilainya tidak berubah selama algoritma berjalan, bisa bernilai bulat atau tidak, tipe dituliskan sebagai const, dan jangkauan nilai atau jumlah nilai dari bilangan ini meliputi semua bilangan yangmungkin.

4. Karakter (Char)

Tipe data tunggal atau terdiri dari satu angka, huruf, simbol baca, juga simbol-simbol yang tidak dapat diimplementasikan dengan operasi aritmatik atau tidak dapat dioperasikan matematis. Tipe dituliskan sebagai char dan diawali dan diakhiri dengan tanda kutip tunggal atau ganda "A", 'b', "?", '3', "@" dan seterusnya). Karakter (char) memiliki jangkauan nilai meliputi ASCII (American Standard Code for Information Interchange).

Karakter di definisikan dengan :

Nama : Character

Rentang nilai : Karakter pada tabel ASCII

Konstanta : '8' 'K' 'k' '&' spasi

Operator :

Kelompok	Operator	Arti	Contoh	Hasil
Operator Relasi	≠	Tidak sama dengan	'a' ≠ 'A'	Boolean
	=	Sama dengan	'a' = 'A'	Boolean

5. Logik (Logical atau Boolean)

Tipe data logical atau boolean digunakan untuk menentukan nilai pembandingan sebagai true atau false (benar atau salah), jangkauan nilai tipe ini hanya dua yaitu false atau true, tipe dituluskan sebagai boolean. operasi operator logical NOT, AND, dan OR , sedangkan operasi operator relational (Penghubung) adalah <>, <=, >=, <, > dan =. Contoh tipe data logical adalah 10 > 9, hasilnya False, adik < kaka, hasilnya, True.

Tipe Data	Ukuran
Boolean	1 Byte
WordBool	2 Byte
LonggBool	3 Byte

3.1.2. Tipe data Bentukan

Tipe data Bentukan adalah sekumpulan variabel – variabel dengan tipe berbeda dan bentuk berbeda pula yang dapat dibentuk sendiri sesuai kebutuhan untuk program yang akan dibuat.

1. Larik (Array)

Tipe data bentukan yang digunakan untuk mendeskripsikan kumpulan elemen-elemen (nilai atau variabel), yang tiap tiap elemennya memiliki indeks. Array dapat didefinisikan sebagai tipe data yang dibentuk untuk menciptakan variabel yang nilai datanya sejenis dan banyak. Array digunakan ketika progrmamer dihadapkan dengan kebutuhan variabel yang nilai datanyak banyak, seperti hari, nama, tanggal ataupun tahun. Contoh array didalam bahasa pemrograman PHP yaitu $bulan = ["jan","feb","maret","april","mei","juni"];.

Format deklarasi variable array :

```
var nama_variable: array[size] of tipe_data
```

2. String

Tipe data bentukan yang menampung data tekstual atau karakter (plains teks) baik itu berbentuk kata maupun kalimat. Data string di apit oleh tanda kutip ganda atau tunggal (“” dan ‘’). Contoh string adalah “ini adalah contoh string” atau ‘ini juga contoh string’.

3. Rekaman (Record)

Sebuah tipe data yang dibentuk untuk menampung sebuah variabel yang mempunyai kelompok elemen-elemen data. Sebagai contoh seorang atau sebuah variabel mahasiswa mempunyai elemen-elemen data nama, tanggal lahir, alamat, jenis kelamain. Tipe data ini dituliskan dengan sintak record.

Contoh :

```
Type Nama_Record  : Record
                 <field_1     :  type_data,
                   Field_2    :  type_data,
                   Field_3:  type_data,
                   Field_n:  type_data>
Nama_variable  : Nama_Record
```

Dimana :

Nama_Record	: nama type record
Field_1,..,field_n	: field – field yang membentuk type record
Nama_variable	: nama variable yang bertipe nama record

3.2. Nama

Nama digunakan untuk mengidentifikasikan objek dan mengacu pada objek tersebut.

3.2.1. Variable

Variabel (Variable) dapat dikatakan sebagai wadah penampung semenatara data yang diberi nama atau identitas. Bentuk umum deklarasi variable adalah :

```
Nama_variable   :   tipe data
```

Contoh :

Nama : string {variabel nama bertipe string}
Nim : integer {variabel nim bertipe integer/bilangan bulat}
jns_kelamin : char {variabel jns_kelamin bertipe karakter}

3.2.2. Konstanta

Tempat penyimpan data/informasi/nilai yang isinya tidak dapat diubah selama pelaksanaan program. Bentuk umum deklarasi konstanta adalah :

```
const nama_konstanta : tipe = nilai
```

Contoh:

const phi : real = 3.14

3.2.3. Fungsi dan Procedure

Fungsi/function adalah cuplikan program atau pengelompokan instruksi berdasarkan kegunaanya atau kumpulan statement yang dikelompokan yang mempunyai maksud dan tujuan tertentu.

Prosedur adalah suatu program terpisah dalam blok sendiri yang berfungsi sebagai subprogram (program bagian).

Contoh :

```
Function NamaFungsi (input : deklarasi parameter, jika ada) → Tipe data
{Spesifikasi fungsi, berisi penjelasan tentang apa yang dilakukan  dan dikembalikan oleh fungsi .}

Procedure hitung_volume_kubus
```

3.3. Latihan

Buat contoh rumus matematika, kemudian tentukan tipe data masing – masing variable yang digunakan.

BAB 4
PERCABANGAN

Capaian Pembelajaran :

1. Mampu menjelaskan jenis percabangan
2. Mampu mengaplikasikan berbagai struktur percabangan
3. Mampu memahami pernyataan seleksi dalam java dengan menggunakan IF dan dapat menyelesaikan masalah yang berkaitan dengan seleksi IF

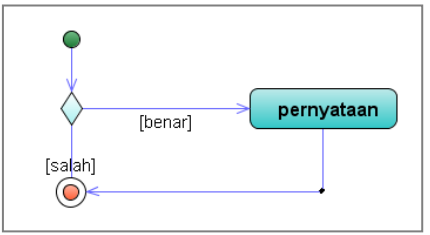
Statement percabangan digunakan untuk proses pengambilan keputusan. Dimana proses akan dikerjakan apabila kondisi yang disyaratkan sesuai (bernilai true/false).

4.1. If Kondisi Tunggal

Pernyataan seleksi dengan IF akan mempunyai beberapa bentuk. Bentuk yang pertama adalah IF dengan satu pilihan. Bentuk umumnya adalah sebagai berikut.

```
if (kondisi)
pernyataan;
```

berikut diagam alir untuk kondisi If tunggal :



Gambar 6 - Diagram alir

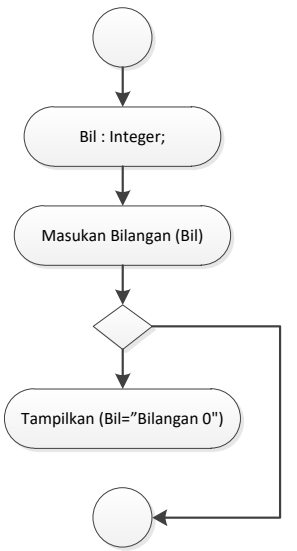
Keterangan :

1. Kondisi digunakan untuk menentukan pengambilan keputusan. Jika kondisi bernilai benar, maka pernyataan dikerjakan
2. Pernyataan, berisi perintah-perintah dan akan dijalankan jika kondisi bernilai benar. Pernyataan disini bisa berupa pernyataan tunggal maupun majemuk.

Contoh soal:

Buatlah program dan deskripsi algoritma untuk menampilkan *output* "Bilangan 0", jika N (nilai) yang dimasukan adalah angka 0.

Deskripsi Algoritma

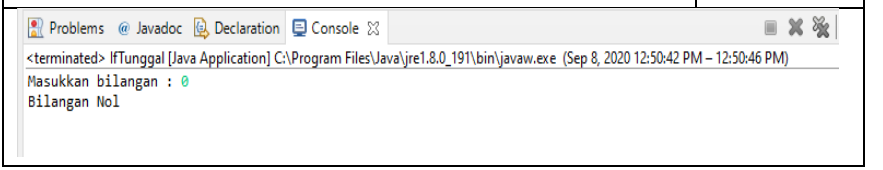
Pseudocode	Diagram Alir Cek Bilangan 0
// deklarasi variable bil : integer; *//input* Masukan Bilangan(bil) *//proses* If(Bil==0) *//output* Tampilkan(Bil=Bilangan 0)	

Implementasi Program	Output

```
1  import java.util.Scanner;
2  public class IfTunggal
3  {
4      public static void main(String args[])
5      {
6          Scanner masuk = new Scanner(System.in);
7          int bil;
8          System.out.print("Masukkan bilangan : ");
9          bil=masuk.nextInt();
10         if (bil==0)
11             System.out.println("Bilangan Nol");
12     }
13 }
```

Masukan bilangan : 0
0 = “Bilangan Nol”

Masukan bilangan : 1
{tidak ada output}



Penjelasan Program:

1. Memanggil library java.util dari class Scanner yang menyediakan method input atau output.
2. Nama class program yang dibuat adalah IfTunggal
3. Awal blok dari class IfTunggal
4. Method utama untuk mengimplementasikan class IfTunggal.
5. Awal blok dari method main().
6. Mengimplementasikan class Scanner untuk mendapatkan method input data dari keyboard : Scanner(System.in).
7. Memperkenalkan atau menginisialisasi varial : Bil, dengan tipe :Integer.
8. Menampilkan keterangan dialog input : “Masukan Bilangan : ”.
9. Menyimpan data yang dimasukan dari keyboard ke dalam variable : Bil. Dengan catatan, data yang dimasukan harus angka bilangan bulat, karena inisialisasi bertipe Integer.
10. Melakukan seleksi, apakah bilangan yang dimasukan adalah angka 0.
11. Menampilkan output dengan format “Bilangan Nol”.

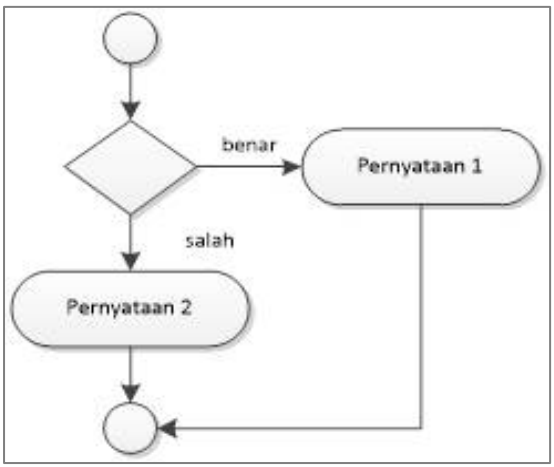
12. Akhir dari blok method main()
13. Akhir dari blok class IfTunggal.

4.2. If Kondisi Ganda

Digunakan untuk membuat percabangan yang sesuai dengan kondisi yang diinginkan. Percabangan if Kondisi ganda biasanya dikenal dengan if-else yang digunakan hanya untuk 2 percabangan. Bentuk umumnya adalah sebagai berikut.

```
if (kondisi)
{ pernyataan1; }
Else
{ pernyataan2; }
```

Berikut diagram alir untuk kondisi if Ganda :



Gambar 7 - diagram alir untuk kondisi if Ganda

Keterangan :

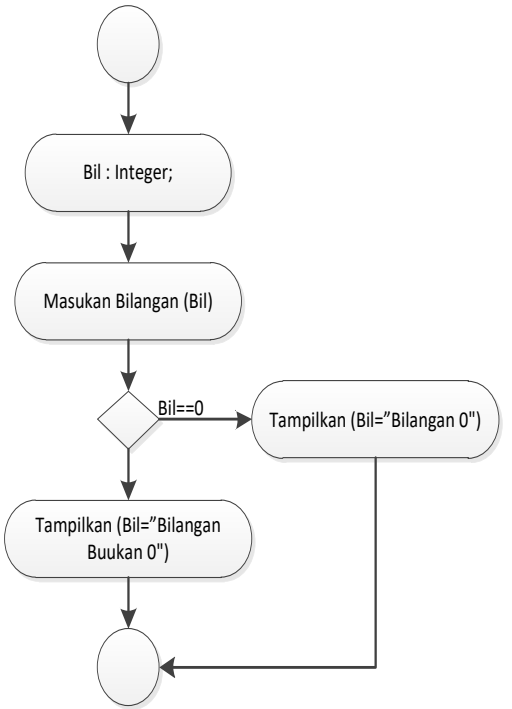
1. Jika hasil cek menghasilkan nilai *true* alur program akan melaksanakan instruksi/pernyataan pada bagian blok *true*, dalam hal ini “pernyataan 1”.

2. Jika hasil cek bernilai *false*, aliran program akan langsung menuju ke blok *false* di bawah else, dalam hal ini "pernyataan 2".
3. Pada saat proragm dijalankan hanya ada satu dari keduanya, baik itu "pernyataan 1" maupun "pernyataan 2".

Contoh soal:
Buatlah program dan dekskripsi algoritma untuk menampilkan *output* "Bilangan 0", nilai N (nilai) yang dimasukan adalah angka 0, atau "Bilangan Bukan 0" jika N (nilai) yang dimasukan selain angka 0.

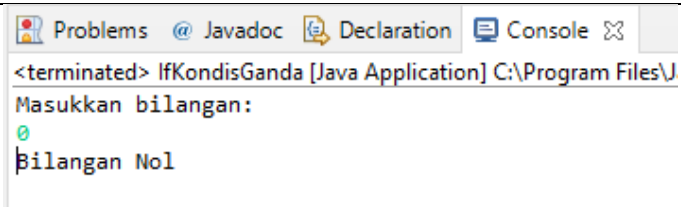
Deskripsi Algoritma

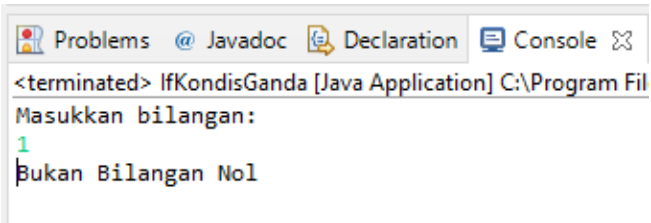
Pseudocode	Diagram Alir Cek Bilangan 0

// deklarasi variable
bil : integer;
//input
Masukan Bilangan(bil)
//proses
If(Bil==0), maka :
Tampilkan(Bil=Bilangan 0)
Else
Tampilkan(Bil=Bilangan bukan 0)
	

Implementasi Program	Output

1 import java.util.Scanner;
2 public class IfKondisGanda
3 {
4 public static void main(String[] args)
5 {
6 Scanner masuk=new Scanner(System.in);
7 int bil;
8 System.out.println("Masukkan bilangan: ");
9 bil=masuk.nextInt();
10 if(bil==0)
11 System.out.println("Bilangan Nol");
12 else
13 System.out.println("Bukan Bilangan Nol");
14 }
15 }
	Masukan bilangan : 0
	0 = "Bilangan Nol"
	
	Masukan bilangan : 1 = "Bilangan Bukan Nol"





Penjelasan program :

1. Memanggil library java.util dari class Scanner yang menyediakan method input atau output.
2. Nama class program yang dibuat adalah IfKondisiGanda
3. Awal blok dari class IfKondisiGanda
4. Method utama untuk mengimplementasikan class IfKondisiGanda.
5. Awal blok dari method main().
6. Mengimplementasikan class Scanner untuk mendapatkan method input data dari keyboard : Scanner(System.in).

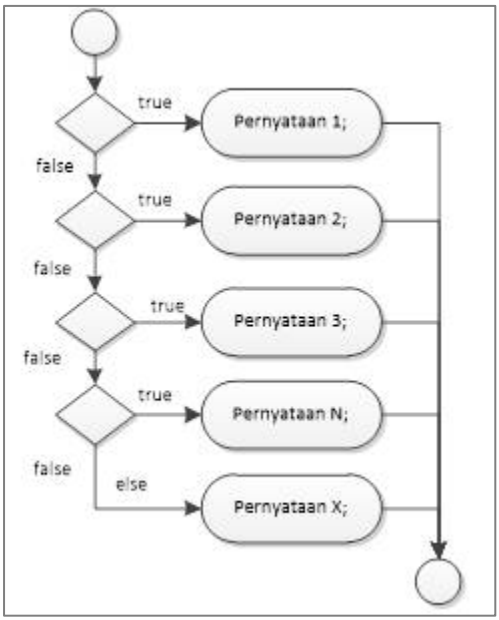
7. Memperkenalkan atau menginisialisasi varial : Bil, dengan tipe :Integer.
8. Menampilkan keterangan dialog input : “Masukan Bilangan : ”.
9. Menyimpan data yang dimasukan dari keyboard ke dalam variable : Bil. Dengan catatan, data yang dimasukan harus angka bilangan bulat, karena inisialisasi bertipe Integer.
10. Melakukan seleksi, apakah bilangan yang dimasukan adalah angka 0.
11. Menampilkan output dengan format “Bilangan Nol”.
12. Bila kondisi baris 10 tidak sesuai, program akan mengabaikan baris 11, langsung menuju dan melaksanakan instruksi baris 13. Selanjutnya ke baris 14 (akhir dari blok)
13. Menampilkan keterangan : Bil=Bilangan Bukan Nol
14. Akhir dari blok method main()
15. Akhir dari blok class IfTunggal.

4.3. If Kondisi Jamak

Kondisi percabangan If Kondisi Jamak merupakan jenis pengujian atau seleksi yang memiliki lebih dari dua kondisi, dan masing – masing kondisi yang bernilai *true*, memiliki blok pernyataan yang unik dan berbeda satu sama lain.

```
if (kondisi)
{ pernyataan1; }
Else if
{ pernyataan2; }
Else if
{ pernyataan3; }
Else if
{ pernyataanN; }
Else
{ pernyataanX; }
```

Berikut diagram alir untuk kondisi If Kondisi Jamak:

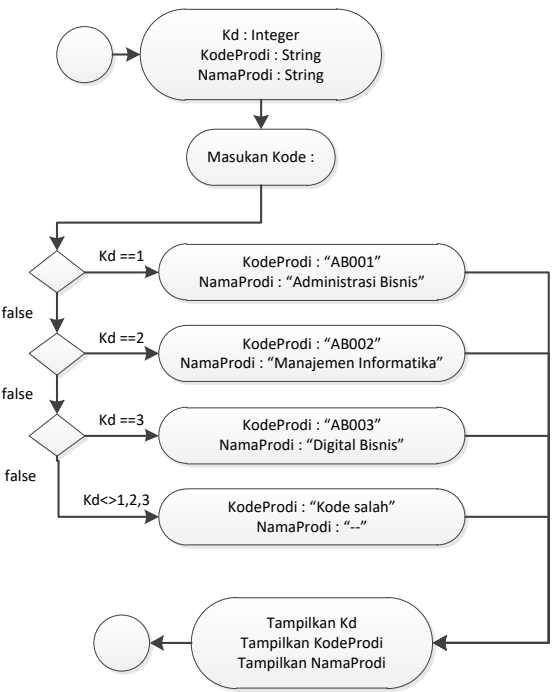


Gambar 8 - diagram alir untuk kondisi If Kondisi Jamak

Contoh soal :

Buatlah deskripsi algoritma, flowchart, dan program java, untuk melakukan proses penyeleksian input kode Program Studi, kemudian menampilkan kode Program Studi, Nama Program Studi.

Langkah algoritma dan Diagram Alir:

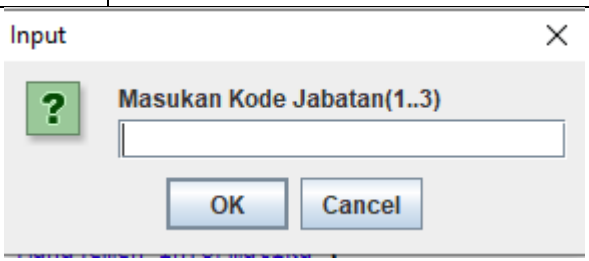
Deskripsi Algoritma	**Gambar Diagram Alir**
Inisialisasi variable: Kd : integer (input/output) KodeProdi : String (output) NamaProdi : String (output) Masukan kode[1..3] : (kd) Jika Kd=1 KodeProdi = "AB001" NamaProdi = "Administrasi Bisnis" Jika Kd=2 KodeProdi = "AB002" NamaProdi = "Manajemen Informatika" Jika Kd=3 KodeProdi = "AB003" NamaProdi = "Digital Bisnis" Selain itu (kd bukan 1 sampai 3) Jika Kd=1 KodeProdi = "salah" NamaProdi = "--"	

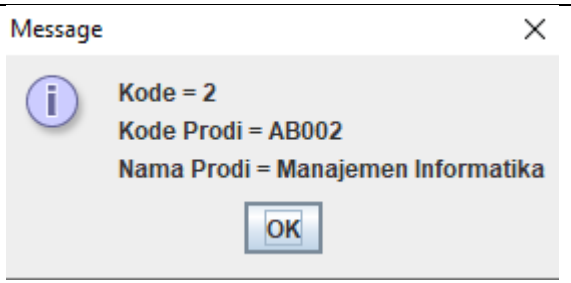
Implementasi Program

```java
import javax.swing.JOptionPane;
public class IfKondisiJamak
{
    public static void main(String args[])
    {
        //inisialisasi variable
        int Kd;
        String KodeProdi;
        String NamaProdi;

        //input : memasukan data dari keyboard versi GUI
        String Kode=JOptionPane.showInputDialog("Masukan Kode Jabatan(1..3)");
        Kd=Integer.parseInt(Kode);

        //proses----------------awal blok if else
        if(Kd==1) {
            KodeProdi= "AB001";
            NamaProdi= "Administrasi Bisnis";
        }
        else if(Kd==2) {
            KodeProdi= "AB002";
            NamaProdi= "Manajemen Informatika";
        }
        else if(Kd==3) {
            KodeProdi= "AB003";
            NamaProdi= "Bisnis Digital";
        }
        else {
            KodeProdi= "Kode yang dimasukan salah/tidak ada";
            NamaProdi= "--";
        }
        //terakhir menampilkan hasil proses versi GUI
        JOptionPane.showMessageDialog(null, "Kode = " +Kd+"\n" + "Kode Prodi = " +KodeProdi+
                "\n"+"Nama Prodi = " +NamaProdi);
    }

}
```

Keterangan		
Kd	**KodeProdi**	**NamaProdi**
1	AB001	Administrasi Bisnis
2	AB002	Manajemen Informatika
3	AB003	Digital Bisnis

Bentuk Input

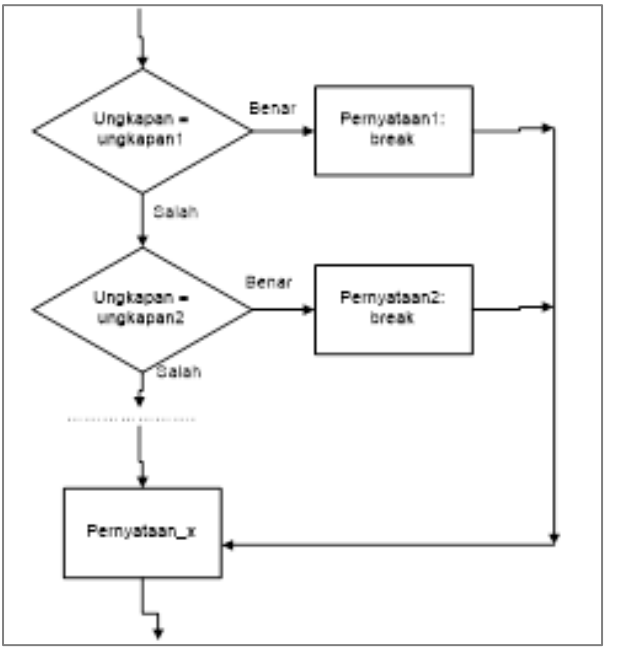


Bentuk Output	

4.4. Switch-Case

Kondisi Percabangan Switch-Case pernyataan yang digunakan untuk menjalankann salah satu pernyataan dari beberapa kemungkinan pernyataan, berdasarkan nilai dari sebuah ungkapan dan nilai penyeleksi. Setiap ungkapan diungkapkan dengan sebuah nilai integral konstan, seperti sebuah nilai dengan tipe byte, short, int atau char.. Bentuk umumnya adalah sebagai berikut.

```
switch (ungkapan)
{
    case ungkapan1:
         pernyataan1;
         break;
    case ungkapan2:
         pernyataan2;
         break;
      ..................
      default:
         pernyataan_x;
}
```



Gambar 9 – diagram alir switch case

Keterangan :

1. ungkapan1, ungkapan2 dan seterusnya dilakukan secara berurutan dimulai dari yang pertama, sekiranyanya cocok pernyataan yang mengikuti **case** dijalankan.
2. **break** ditemukan dari eksekusi pernyataan **switch** berakhir
3. **default** hanya akan dijalankan jika ungkapan pada bagian case tidak ada yang cocok.

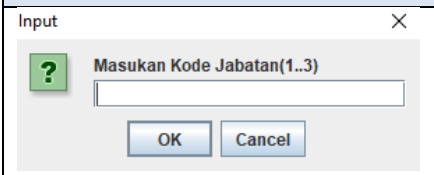
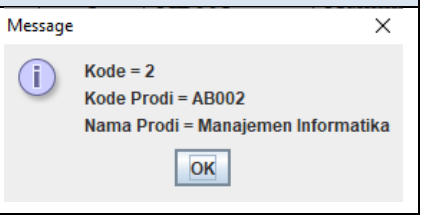
Contoh Soal:

Memodifikasi contoh soal If Kondisi Jamak dengan metode seleksi Switch-Case

Implementasi Program

```
1  import javax.swing.JOptionPane;
2  public class SwitchCase {
3      public static void main(String args[])
4      {
5          //inisialisasi variable
6          int Kd;
7          String KodeProdi;
8          String NamaProdi;
9
10         //input : memasukan data dari keyboard versi GUI
11         String Kode=JOptionPane.showInputDialog("Masukan Kode Jabatan(1..3)");
12         Kd=Integer.parseInt(Kode);
13
14         //proses----------------pemilihan switch-case
15         switch(Kd) {
16         case 1 :
17             KodeProdi= "AB001";
18             NamaProdi= "Administrasi Bisnis";
19             break;
20
21         case 2 :
22             KodeProdi= "AB002";
23             NamaProdi= "Manajemen Informatika";
24             break;
25
26         case 3 :
27             KodeProdi= "AB003";
28             NamaProdi= "Bisnis Digital";
29             break;
30         default:
31             KodeProdi= "Kode yang dimasukan salah/tidak ada";
32             NamaProdi= "--";
33             break;
34         }
35         //terakhir menampilkan hasil proses versi GUI
36         JOptionPane.showMessageDialog(null, "Kode = " +Kd+"\n" + "Kode Prodi = " +KodeProdi+
37                 "\n"+"Nama Prodi = " +NamaProdi);
38     }
39 }
40
```

Hasil:

Bentuk Input	Bentuk Output
	

4.5. Latihan

1. Modifikasi input dan output program IfKondisiJamak menggunakan class scanner dan System.out.print
2. Buatlah deskripsi algoritma, flowchart, dan program java, untuk melakukan proses penyeleksian input kode jabatan, kemudian

menampilkan kode jabatan, Nama Jabatan, dan Gaji, dengan ketentuan :

Kode	Jabatan	Gaji
1	Operator	2.000.000
2	Programmer	4.000.000
3	Analisis	5.500.000
4	Manager	7.000.000

BAB 5
PENGULANGAN

Capaian Pembelajaran :

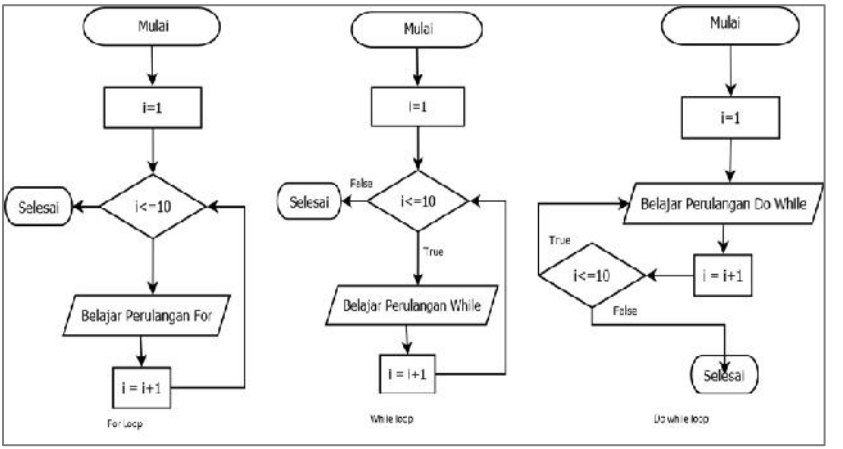
1. Mampu menjelaskan jenis pengulangan
2. Mampu mengaplikasikan berbagai struktur pengulangan
3. Mampu memahami pernyataan seleksi dalam java dengan menggunakan struktur pengulangan

Pengulangan atau disebut sebagai looping adalah instruksi khusus dalam bahasa pemrograman dan algoritma yang digunakan untuk mengulang beberapa perintah sesuai dengan jumlah yang telah ditentukan. tujuannya adalah untuk mempermudah pengerjaan program dan untuk mempersingkat instruksi program. Dalam bahasa pemrograman Java, terdapat tiga bentuk pengulangan, yaitu For, While, dan Do-While.

Bentuk pengulangan yang sederhana, yaitu :

1. Pengulangan digunakan untuk mengerjakan suatu atau beberapa perintah secara berulang – ulang sesuai dengan yang diinginkan.
2. Pengulangan sederhana yaitu pengulangan yang hanya membutuhkan 1 kali pengulangan.
3. Di dalam pengulangan sederhana tidak ada pengulangan bersarang.

Pada gambar 10 diberikan contoh flowchart dengan tiga bentuk pengulangan.



Gambar 10 - bentuk pengulangan

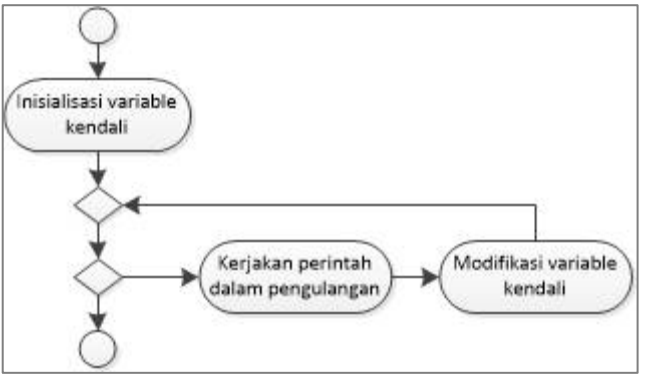
5.1. Pengulangan For

Pernyataan for digunakan untuk mengerjakan pernyataan atau sekelompok pernyataan secara berulang, dalam hitungan yang sudah pasti. Karakteristik dari pengulangan for yaitu :

1. Digunakan untuk pengulangan yang batasnya sudah diketahui dengan jelas, misalnya 5 kali.
2. Memerlukan dua buah penanda (nilai awal) dan akhir pengulangan (nilai akhir).
3. Nilai penghitung akan secara otomatis bertambah atau berkurang setiap kali sebuah pengulangan dilaksanakan, tergantung jenis pengulangannya.
4. Memerlukan suatu aksi yang dapat diulangi sebanyak pengulangannya.

Bentuk umumnya adalah sebagai berikut.

```
for (ungkapan1;ungkapan2;ungkapan3)
            Pernyataan;
```



Gambar 11 – diagram alir pengulangan for

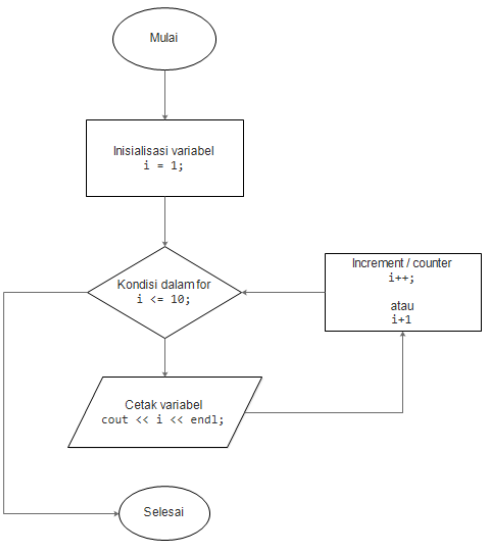
Keterangan :

1. ungkapan1 merupakan pernyataan inisialisasi
2. ungkapan2 sebagai kondisi yang menentukan pengulangan terhadap pernyataan atau tidak
3. ungkapan3 digunakan sebagai pengatur variabel yang digunakan didalam ungkapan1

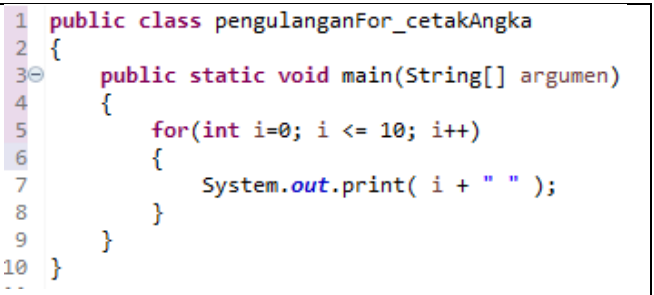
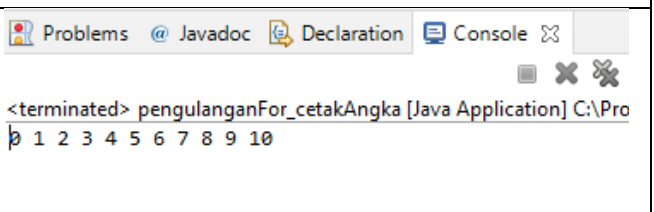
Contoh soal 1:

Buatlah program dan dekskripsi algoritma untuk menampilkan nilai dari 1 sampai 10 dengan menggunakan pengulangan for.

Deskripsi algoritma

Pseudocode	Diagram Alir
Algoritma menampilkan_nilai Deklarasi: i : integer deskripsi : for(int i=0; i <= 10; i++)	

Implementasi:

Implementasi program	
Hasil Output	

Contoh soal 2:
Buatlah program dan dekskripsi algoritma untuk menampilkan nilai berdasarkan jumlah masukan pengguna dengan menggunakan pengulangan for.

Deskripsi algoritma

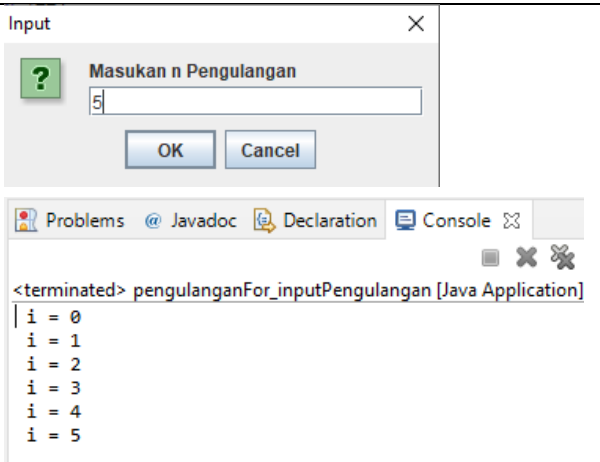
```
//deklarasi
n = integer;
msk = string

//deskripsi
i = n
read(n)
read(msk)

for(int i=0;i<=n;i++)

//output
i = 0;
i = 1;
i = 2;
i = n;
```

Implementasi:

Implementasi program

```
1  import javax.swing.*;
2  public class pengulanganFor_inputPengulangan
3  {
4      static int n;
5
6      static void masukan()
7      {
8          String msk = JOptionPane.showInputDialog("Masukan n Pengulangan");
9          n = Integer.parseInt(msk);
10     }
11
12     static void proses()
13     {
14         for(int i=0;i<=n;i++)
15         {
16             System.out.println(" i = " +i);
17         }
18     }
19
20     public static void main(String[]args)
21     {
22         masukan();
23         proses();
24     }
25
```

Hasil Output

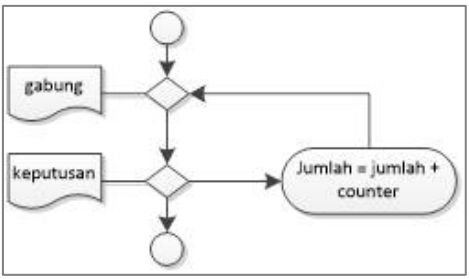


5.2. Pengulangan While

Pengulangan while berguna untuk memproses suatu pernyataan atau beberapa pernyataan beberapa kali. Selama ungkapan bernilai benar, pernyataan akan selalu dikerjakan. Pernyataan perulangan dengan while akan selalu dikerjakan jika ungkapan selalu benar. Oleh karena itu, kita harus membuat kondisi suatu saat ungkapan bernilai salah agar perulangan berakhir. Bentuk umumnya adalah :

```
while (ungkapan)
```

```
Pernyataan;
```



Gambar 12 – diagram alir pengulangan while

Keterangan :

1. Bagian pernyataan akan diekseskusi selama ungkapan dalam **while** bernilai benar.
2. Pengujian terhadap ungkapan pada **while** dilakukan sebelum bagian pernyataan.
3. Kemungkinan pernyataan pada **while** tidak dijalankan sama sekali, jika ketemu kondisi yang pertama kali bernilai salah.

Contoh soal:

Buatlah program pengulangan while dan dekskripsi algoritma untuk menghitung rata-rata bilangan positif, dimana banyaknya data ditentukan dari data yang dimasukan dengan keyboard.

Deskripsi algoritma

```
//deklarasi
i = integer;
n, jum, x, rata = float

//deskripsi
i = n
read(jum)

while (i<=n)
```

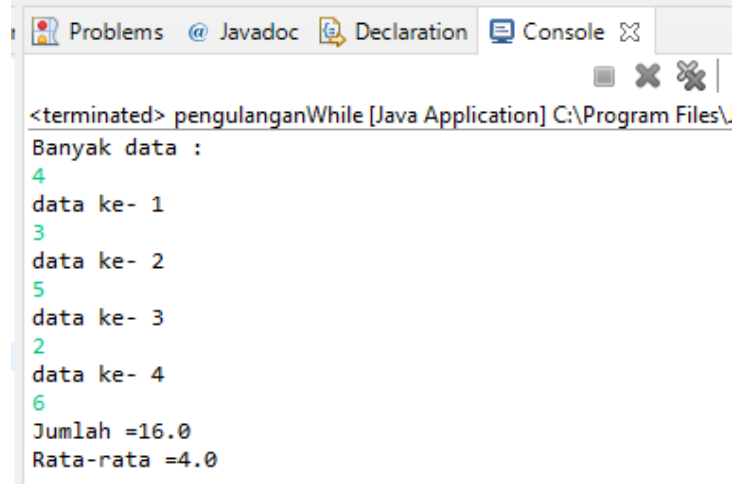
//output
Banyak data : 4
Data ke-1 : 3
Data ke-2 : 4
Data ke-3 : 3
Data ke-4 : 2
Jumlah = 12.0
Rata-rata = 3.0

Implementasi:

Implementasi program

```
1  import java.util.Scanner;
2  public class pengulanganWhile
3  {
4      public static void main(String args[])
5      {
6          Scanner masuk=new Scanner(System.in);
7          int i=1;
8          float n, jum = 0, x = 0, rata;
9          System.out.println("Banyak data : ");
10         n = masuk.nextFloat();
11
12         while (i<=n)
13         {
14             System.out.println("data ke- "+i);
15             x=masuk.nextFloat();
16             jum += x;
17             i++;
18         }
19             System.out.println("Jumlah ="+jum);
20             System.out.println("Rata-rata ="+(jum/n));
21
22     }
23 }
```

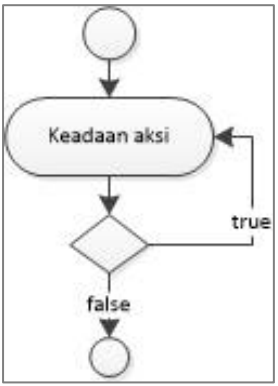
Hasil Output



5.3. Pengulangan Do..while

Pengulangan dengan do...while ini juga digunakan untuk mengerjakan sebuah atau sekelompok pernyataan berulang-ulang. Bedanya dengan while adalah pernyataan do...while akan mengecek kondisi di belakang, sementara while cek kondisi ada di depan. Bentuk umumnya adalah :

```
do
{
      pernyataan1;
      pernyataan2;
      ...........
      pernyataan_N;
}
while (ungkapan)
```



Gambar 13 – diagram alir pengulangan do..while

Keterangan :

1. Bagian pernyataan1 hingga pernyataanN dijalankan secara berulang sampai ungkapan bernilai salah.
2. Pengujian ungkapan dilakukan setelah bagian pernyataan, maka pada pernyataan do ... while minimal akan dijalankan sekali, karena begitu masuk ke blok perulangan, tidak ada cek kondisi tetapi langsung mengerjakan pernyataan.

Contoh soal:

Buatlah program pengulangan do..while dan dekskripsi algoritma untuk menghitung rata-rata bilangan positif, dimana banyaknya data ditentukan dari data yang dimasukan.

Deskripsi algoritma

```
//deklarasi
i = integer;
n, jum, x, rata = float

//deskripsi
i = n
read(jum)
do
```

while (i<=n)

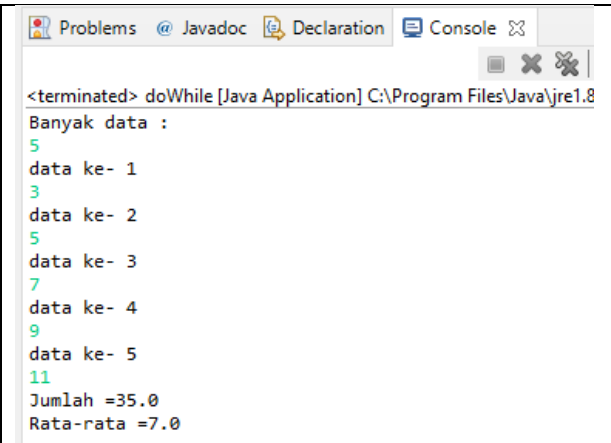
//output
Banyak data : 4
Data ke-1 : 3
Data ke-2 : 4
Data ke-3 : 3
Data ke-4 : 2
Jumlah = 12.0
Rata-rata = 3.0

Implementasi:

Implementasi program

```
1  import java.util.Scanner;
2  public class doWhile {
3      public static void main(String args[])
4      {
5          Scanner masuk=new Scanner(System.in);
6          int i=1;
7          float n, jum = 0, x = 0, rata;
8          System.out.println("Banyak data : ");
9          n = masuk.nextFloat();
10
11         do
12         {
13             System.out.println("data ke- "+i);
14             x=masuk.nextFloat();
15             jum += x;
16             i++;
17         } while (i<=n);
18             System.out.println("Jumlah ="+jum);
19             System.out.println("Rata-rata ="+(jum/n));
20     }
21 }
```

Hasil Output



5.4. Latihan

1. Buatlah program pengulangan for dan deskripsi algoritma untuk menampilkan bilangan kelipatan 5 antara 125 sampai dengan 200
2. Buatlah program pengulangan while dan deskripsi algoritma untuk mencetak bilangan genap dari 0 sampai dengan 10.

BAB 6
ARRAY

Capaian Pembelajaran :

1. Mampu memahami array satu, dua, dan tiga dimensi
2. Mampu mengaplikasikan array ke dalam program java

Array atau larik adalah koleksi data dimana setiap elemen memakai nama yang sama dan bertipe sama dan setiap elemen diakses dengan membedakan indeks arraynya. Array merupakan sebuah variabel yang menyimpan lebih dari 1 buah data yang memiliki tipe data yang sama. Jadi dapat dikatakan bahwa array merupakan kumpulan dari data-data tunggal yang dijadikan dalam 1 variabel array yang alamat memorinya berbeda yang selanjutnya disebut elemen-elemen array yang bisa kita akses berdasarkan indeks.

6.1. Array 1 Dimensi

Larik adalah sebuah struktur data yang terdiri dari data yang bertipe sama. Ukuran larik bersifat tetap, larik akan mempunyai ukuran yang sama pada saat sekali dibuat. Larik dalam Java adalah obyek, disebut juga sebagai tipe referensi. Bentuk umum :

```
Tipe [ ]  variable;
Tipe [ ]  variable = new tipe [n];
Tipe [ ]  variable = {data-1, data-2, …data-n}
```

Contoh:

1. Int [ ] A; atau int A [ ] ;
   Artinya menyiapkan deklarasi array A bertipe integer di awal program, dimana panjang array dan data array belum dinyatakan.

2. Int [ ] A = new int[8]; atau int A [ ] = new int[8];
   Artinya menyiapkan deklarasi array A bertipe integer di awal program, sekaligus menyiapkan ruang memori sebanyak 8 kotak.

Panjang array = jumlah/banyaknya data : (A.length) = 8								
A:	0	1	2	3	4	5	6	7
Data	0	0	0	0	0	0	0	0

3. Int [ ] A = {7, 12, 15, 20, 22, 24, 29, 31}; atau int A [ ] = {7, 12, 15, 20, 22, 24, 29, 31};
   Artinya menyiapkan deklarasi array A bertipe integer sekaligus memasukan data secara langsung.

Panjang array = jumlah/banyaknya data : (A.length) = 8								
A:	0	1	2	3	4	5	6	7
Data	7	12	15	20	22	24	29	31

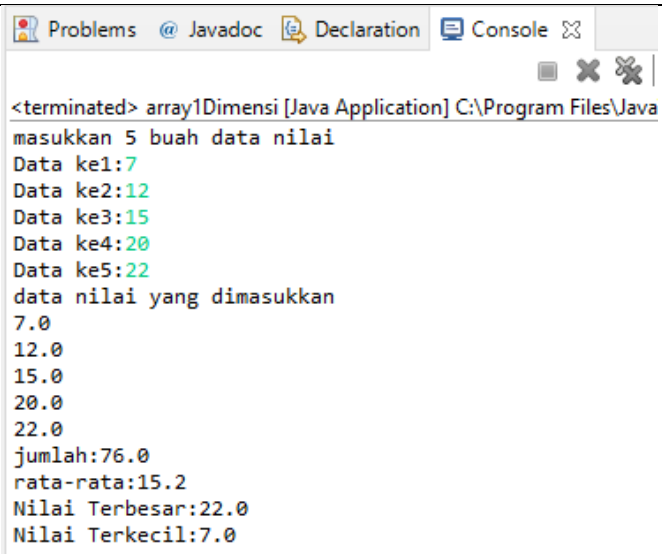
Contoh soal:
Buatlah program array 1 dimensi untuk menghitung jumlah, rata – rata serta niai terbesar dan terkecilnya!

Implementasi:

Implementasi program

```
1  import java.util.Scanner;
2  public class array1Dimensi
3  {
4      public static void main(String arg[])
5      {
6          Scanner masuk=new Scanner(System.in);
7          float jumlah,rata,maks,min;
8          float nilai[]=new float[5];
9
10         System.out.println("masukkan 5 buah data nilai");
11         for(int i=0;i<5;i++)
12         {
13             System.out.print("Data ke"+(i+1)+":");
14             nilai[i]=masuk.nextFloat();
15         }
16
17         System.out.println("data nilai yang dimasukkan");
18         for(int i=0;i<5;i++)
19             System.out.println(nilai[i]);
20             jumlah=0;
21             for(int i=0;i<5;i++)
22                 jumlah=jumlah + nilai[i];
23                 rata=jumlah/5;
24             {
25                 maks = nilai[0];
26                 min=nilai[0];
27                 for(int i=0;i<5;i++)
28                 {
29                     if(maks<nilai[i])
30                         maks= nilai[i];
31                         if(min> nilai[i])
32                             min= nilai[i];
33                 }
34             }
35              System.out.println("jumlah:"+jumlah);
36              System.out.println("rata-rata:"+rata);
37              System.out.println("Nilai Terbesar:"+maks);
38              System.out.println("Nilai Terkecil:"+min);
39     }
40 }
41
```

Hasil Output



6.2. Array 2 Dimensi

Array dua dimensi sering digambarkan sebagai sebuah matriks, merupakan perluasan dari array satu dimensi. Jika array satu dimensi hanya terdiri dari sebuah baris dan beberapa kolom elemen, maka array dua dimensi terdiri dari beberapa baris dan beberapa kolom elemen yang bertipe sama.

Contoh :

1. Int [ ] [ ] A; atau A [ ] [ ];
   Artinya menyiapkan deklarasi array A dua dimensi bertipe integer di awal program, dimana panjang array dan data array belum dinyatakan.
2. Int [ ] [ ] A = new int [2][4]; atau int A [ ] [ ] = new int [2][4];
   Menyiapkan deklarasi array A dua dimensi bertipe integer di awal program sekaligus menyiapkan ruang memori sebanyak dua baris, dan masing-masing baris tersedia 4 kolom (2 x 4).

	A[x][y]	0	1	2	3
A[0][0]=0; A[0][1]=0; A[0][2]=0; A[0][3]=0;	0	0	0	0	0
A[1][0]=0; A[1][1]=0; A[1][2]=0; A[1][3]=0;	1	0	0	0	0

3. Int A[ ] [ ] = { {7,12,15,20}, {22,24,29,31} };
   Artinya menyiapkan array A dua dimensi bertipe integer, sekaligus memasukan data secara langsung di dalam program.

	A[x][y]	0	1	2	3
A[0][0]=7; A[0][1]=12; A[0][2]=15; A[0][3]=20;	0	7	12	15	20
A[1][0]=22; A[1][1]=24; A[1][2]=29; A[1][3]=31;	1	22	24	29	31

Contoh soal:

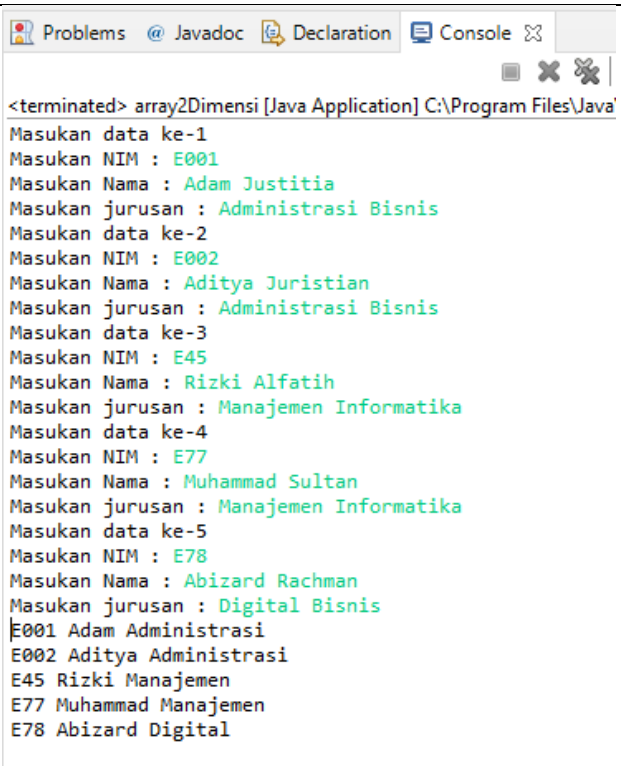
Buatlah program dengan menggunakan array 2 dimensi untuk menyimpan 5 data mahasiswa yaitu berupa NIM, Nama dan Jurusan.

Implementasi :

Implementasi program

```
import java.util.Scanner;
public class array2Dimensi
{
    public static void main(String argss[])
    {
        String[][] data=new String[11][4];
        for(int i=1; i<=10; i++)
        {
            System.out.println("Masukan data ke-" +i);
            for(int j=1; j<=3; j++)
            {
                if (j==1)
                {
                    System.out.print("Masukan NIM : ");
                    data[i][j]=new Scanner(System.in).next();
                }
                else if(j==2)
                {
                    System.out.print("Masukan Nama : ");
                    data[i][j]=new Scanner(System.in).next();
                }
                else if(j==3)
                {
                    System.out.print("Masukan jurusan : ");
                    data[i][j]=new Scanner(System.in).next();
                }
            }
        }

        for(int i=1; i<=10; i++)
        {
            for(int j=1; j<=3; j++)
            {
                System.out.print(data[i][j]+" ");
            }
            System.out.println("");
        }
    }
}
```

Hasil Output



6.3. Array 3 Dimensi

Array 3 dimensi adalah array yang tidak jauh berbeda dari array satu dimensi dan dua dimensi yang telah dijelaskan sebelumnya, kecuali pada indeks dari array. Pada tipe ruang misalnya tipe ruang = array [1..8,1..5,1..3] of integer; menunjukkan bahwa ruang adalah nama-pengenal/variabel yang berupa array yang komponennya bertipe integer dan terdiri atas 8 baris, mempunyai 5 kolom dan 3 halaman.

Contoh soal:

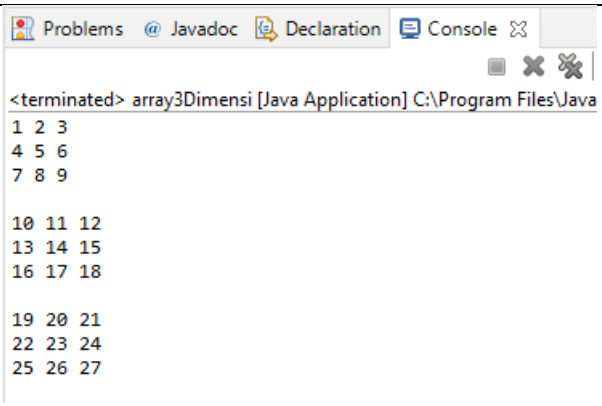
Buat array 3 dimensi sehingga menghasilkan data seperti gambar di bawah :

```
1 2 3
4 5 6
7 8 9

10 11 12
13 14 15
16 17 18

19 20 21
22 23 24
25 26 27
```

Implementasi :

Implementasi program

```
public class array3Dimensi
{
    public static void main(String[] args)
    {
        // membuat isi elemen array
        int [][][] angka = {     // {kolom1,kolom2,kolom3}
                            {{1,2,3},    // baris ke-0
                             {4,5,6},    // baris ke-1
                             {7,8,9}},   // baris ke-2

                            {{10,11,12},    // baris ke-3
                             {13,14,15},    // baris ke-4
                             {16,17,18}},   // baris ke-5

                            {{19,20,21},    // baris ke-6
                             {22,23,24},    // baris ke-7
                             {25,26,27}}    // baris ke-8
                           };  // kurung kurawa array penutup

        // mendeklarasikan baris dan kolom
        int i, j,k;  // i = baris, j = kolom
            for ( i=0;i<3; i++) // menampilkan elemen sejumlah baris
            {
                for ( j=0;j<3; j++)// menampilkan elemen sejumlah kolom
                {
                    for ( k=0;k<3; k++)
                    {
                        // menampilkan isi elemen baris dan kolom
                        System.out.print(angka[i][j][k]+" ");
                    }

                    System.out.println("");          //pindah baris
                }

                System.out.println("");          //pindah baris
            }
    }
}
```

Hasil Output	

6.4. Latihan

1. Buat program dengan menggunakan array 1 dimensi untuk menampilkan bilangan dari 1 sampai 10 dengan pangkatnya masing – masing.
2. Buat program untuk menampilkan menampilkan matrik array 2 dimensi yang elemen – elemennya dimasukkan melalui keyboard!
3. Buat program untuk menjumlahkan matrik dengan menggunakan array 2 dimensi.

BAB 7
CLASS, METHOD, OBJEK

Capaian Pembelajaran :

1. Mampu mengenal istilah kelas dalam program java dan dapat membuat sebuah kelas
2. Mampu memahami konsep class dan objek serta perbedaannya.
3. Mampu mendefinisikan kelas beserta komponen-komponen kelas.
4. Mampu menjelaskan obyek yang mengacu ke kelas tersebut.

Program yang kompleks dapat dibangun dari berbagai package (paket). Sebuat paket terdiri dari sejumlah *class*, setiap *class* terdiri dari beberapa method atau *constructor*. Setiap *class*, minimal memiliki sebuah *constructor*, walaupun secara explicit tidak dibuat, namun secara implisit sesungguhnya tetap ada, hanya tidak ditampilkan.

7.1. Class

Class adalah struktur dasar dari OOP (Object Oriented Programming). Terdiri dari dua tipe yaitu : field (*attribute/property*) dan method (*behavior*). Class digunakan untuk mendeklarasikan sebuah variabel yang berupa objek atau dinamakan "referensi objek (*object reference*)".

Kemampuan objek sangat bergantung pada sarana yang telah disediakan oleh kelas. Kelas merupakan program Java yang akan dieksekusi. Sebuah *Class* dideklarasikan dengan :

```
class KelasKu {
    //deklarasi field, konstruktor dan method
}
```

Deklarasi *class Kelasku* sudah sering dibuat sebelumnya dengan menambah kata kunci public diawalnya. Isi dari kelas (daerah antara dua tkita kurung kurawal) berisi semua kode yang disediakan untuk obyek yang diciptakan dari kelas, yaitu konstruktor untuk inisialisasi obyek baru, deklarasi field yang menetapkan keadaan kelas dan obyeknya dan method untuk mengimplementasikan lingkungan dari kelas dan obyeknya.

Secara umum, deklarasi kelas dapat termasuk komponen-komponen

1. Modifier seperti public, private dan modifier yang lain yang akan kita bicarakan kemudian.
2. Nama kelas, dengan diawali huruf besar sebagai kesepakatan.
3. Nama dari induk kelasnya (superclass), jika ada, diawali dengan kata kunci extends. Sebuah kelas hanya boleh mempunyai satu induk
4. Daftar interface (dipisahkan dengan tkita koma) yang akan diimplementasikan dalam kelas, jika ada, diawali dengan kata kunci implements. Sebuah kelas boleh mengimplementasikanlebih dari satu interface
5. Isi dari kelas yang diawali dan diakhiri dengan tanda kurung kurawal buka dan tutup { }

Bagian yang tidak kalah pentingnya adalah deklarasi variabel anggota. Ada beberapa macam variabel yang ada di bagian ini.

1. Variabel anggota dalam sebuah kelas – ini disebut fields. Fields ini terletak di luar method. Dan bisa diakses dai method dengan menggunakan referensi ke kelas yang memiliki field tersebut (dengan memperhatikan aturan akses modifier)
2. Variabel dalam sebuah method atau blok kode – ini disebut variabel lokal. Variabel ini biasanya hanya digunakan selama method itu dikerjakan. Sehingga tidak perlu diakses dari luar

method. Bahkan variabel yang ada di dalam blok bisa diakses dari blok itu saja.

3. Variabel dalam deklarasi method – ini disebut parameter. Parameter sudah pernah dibahas panjang lebar pada bagian sub program

Deklarasi field terdiri dari 3 komponen

1. Tidak ada atau ada modifier, seperti public atau private. Sebenarnya dengan tanpa menuliskan modifier, maka kita membuat deklarasi field tersebut sebagai default.
2. Tipe field. Pada perkembangan pembahasan selanjutnya, tipe ini bisa saja bukan hanya tipe sederhana tetapi tipe yang kompleks.
3. Nama field. Dalam pembuatan nama, aturan penamaan harus diikuti. Dan sangat dianjurkan untuk menggunakan huruf kecil sebagai huruf pertama.

Contoh soal:

Buat program untuk menghitung Luas dan Keliling persegi panjang.

Implementasi :

Implementasi program

```
1  class PersegiPanjang
2  {
3      // kelas PersegiPanjang mempunyai dua atribut
4      public int panjang;
5      public int lebar;
6      public void setPanjang(int nilaiBaru)
7      {
8          panjang = nilaiBaru;
9      }
10     public void setLebar(int nilaiBaru)
11     {
12         lebar = nilaiBaru;
13     }
14     public int hitungLuas()
15     {
16         return panjang*lebar;
17     }
18     public int hitungKeliling()
19     {
20         return 2*(panjang+lebar);
21     }
22 }
23
24 public class Panjang
25 {
26     public static void main(String[] args)
27     {
28         PersegiPanjang PP = new PersegiPanjang();
29         PP.setLebar(3);
30         PP.setPanjang(4);
31         System.out.println("Luas     = " + PP.hitungLuas());
32         System.out.println("Keliling= " + PP.hitungKeliling());
33     }
34 }
```

Hasil Output

```
Luas     = 12
Keliling = 14
```

7.2. Method

Method adalah bagian-bagian kode yang dapat dipanggil oleh kelas, badan program atau method lainnya untuk menjalankan fungsi yang spesifik di dalam kelas. Secara umum method dalam java adalah sebuah fungsi.

Berikut adalah karakteristik dari method :

1. Dapat mengembalikan / melaporkan nilai balikkan (return value) atau tidak (void).

2. Dapat diterima beberapa parameter yang dibutuhkan atau tidak ada parameter sama sekali. Parameter bisa juga disebut sebagai argumen dari fungsi. Parameter berguna sebagai nilai masukkan yang hendak diolah oleh fungsi.
3. Setelah method telah selesai dieksekusi, dia akan kembali pada method yang memanggilnya.

Pada method yang tidak mempunyai parameter, hasil method tersebut tidak dapat diatur dari modul yang menggunakannya, karena tidak ada parameter yang dikirimkan.

Contoh soal:
Buat program untuk penggunaan method sederhana tanpa parameter.

Implementasi :

Implementasi program

```
1  public class TestMethod
2  {
3      public static void setGaris()
4      {
5          System.out.println("=======");
6      }
7      public static void setJudul()
8      {
9          System.out.println("Simple Method");
10     }
11     public static void main(String[] args)
12     {
13         setGaris();
14         setJudul();
15         setGaris();
16     }
17 }
```

Hasil Output

```
=======
Simple Method
=======
```

Semua method dalam suatu class dapat mengakses data-datanya secara langsung tanpa melalui referensi. Pemanggilan method

dilakukan dengan menuliskan objek pemiliknya dan diikuti oleh operator titik (.) beserta nama method yang akan dieksekusi.

Untuk memanggil method adalah :

```
Objek ke-1.nama_method;
```

Contoh soal :

Buat program untuk menampilkan nilai volume kotak ke monitor.

Implementasi :

Implementasi program

```
1  class KotakMethod
2  {
3      double panjang;
4      double lebar;
5      double tinggi;
6
7      //mendefinisikan method void (tidak mengembalikan nilai)
8      void CetakVolume()
9      {
10         System.out.println("Volume Kotak = " + (panjang * lebar * tinggi));
11     }
12 }
13 public class DemoMethod
14 {
15     public static void main(String[] args)
16     {
17         KotakMethod k1, k2,k3;
18
19         //instansiasi objek
20         k1 = new KotakMethod();
21         k2 = new KotakMethod();
22         k3 = new KotakMethod();
23
24         //mengisi data untuk objek k1
25         k1.panjang = 4;
26         k1.lebar = 3;
27         k1.tinggi = 2;
28
29         //mengisi data untuk objek k2
30         k2.panjang = 6;
31         k2.lebar = 5;
32         k2.tinggi = 4;
33
34         //mengisi data untuk objek k3
35         k3.panjang = 8;
36         k3.lebar = 7;
37         k3.tinggi = 6;
38
39         //memanggil method cetakVolume() untuk masing-masing objek
40         k1.CetakVolume();
41         k2.CetakVolume();
42         k3.CetakVolume();
43     }
44 }
```

Hasil Output

```
Volume Kotak = 24.0
Volume Kotak = 120.0
Volume Kotak = 336.0
```

7.3. Objek

Setiap Object dibangun dari sekumpulan data (atribut) yang disebut "variabel" (untuk menjabarkan karakteristik khusus dari obyek) dan juga terdiri dari sekumpulan method (menjabarkan tingkah laku dari obyek) atau Obyek adalah = sebuah perangkat lunak yg berisi sekumpulan variabel dan method yang berhubungan. Obyek merupakan sebuah *instance*(keturunan) dari *class*. Variabel dan method diketahui sebagai variabel *instance* dan *method instance*.

Sebuah ciri khas program Java menciptakan banyak obyek, yang berinteraksi dengan meminta method melalui interaksi obyek-obyek tersebut, sebuah program dapat membawa bermacam-macam tugas, seperti implementasi GUI, menjalankan animasi, atau mengirimkan dan menerima informasi melalui jaringan. Sekali sebuah obyek menyelesaikan pekerjaan untuk apa obyek tersebut dibuat, sumber dayanya didaur ulang untuk untuk digunakan oleh obyek yang lain.

7.3.1. Inisialisasi sebuah obyek

Pada saat diciptakan, sebuah obyek harus diinisialisasi. Sebagai contoh akan kita lihat kode program untuk kelas Titik:

```
public class Titik {
   public int x = 0;
   public int y = 0;
   //konstruktor
   public Titik(int a, int b) {
     x = a;
     y = b;
    }
}
```

Kelas ini berisi sebuah konstruktor tunggal. Dapat dikenali dengan sebuah konstruktor karena deklarasinya menggunakan nama yang sama dengan kelas dan tidak mempunyai tipe kembalian.

Konstruktor dalam kelas Titik mengambil dua argumen integer, seperti dideklarasikan oleh kode (int a, int b). Pernyataan berikut menyediakan 46 dan 88 sebagai nilai untuk argumen tersebut:

```
Titik titikAwal = new Titik(46, 88);
```

7.3.2. Menggunakan Obyek

Jika objek telah dibuat, maka akan bisa dipergunakan untuk sesuatu maksud tertentu.

```
int panjang = new PersegiPanjang().panjang;
```

Contoh soal:

Buat program untuk menampilkan nilai volume kotak ke monitor.

Implementasi :

Implementasi program

```
1  class Kotak
2  {
3      double panjang;
4      double lebar;
5      double tinggi;
6  }
7  class DemoKotak
8  {
9      public static void main(String[] args)
10     {
11         double volume;
12         Kotak k = new Kotak();
13
14         //mengisikan nilai ke dalam data-data kelas Kotak
15         k.panjang = 4;
16         k.lebar = 3;
17         k.tinggi = 2;
18
19         //menghitung isi/volume kotak
20         volume = k.panjang * k.tinggi * k.lebar;
21
22         //menampilkan nilai volume ke layar monitor
23         System.out.println("Volume Kotak = " + volume);
24     }
25 }
```

Hasil Output

```
Volume Kotak = 24.0
```

Program di atas harus disimpan dengan nama **DemoKotak.java** bukan Kotak.java karena *method main* ( ) terletak di class DemoKotak. Pada saat kompilasi program akan membentuk 2 buah file *class*, yaitu Kotak.class dan DemoKotak.class.

Setiap objek dari class akan memiliki salinan data sendiri-sendiri, artinya antara objek satu dengan lainnya dapat mempunyai nilai data yang berbeda.

7.4. Latihan

1. Buat program untuk menghitung volume balok/kubus dan menentukan apakah bangun yang dimasukkan balok atau kubus.
2. Modifikasi program untuk menghitung volume balok/kubus berdasarkan inputan pengguna.

BAB 8
ALGORITMA PENGURUTAN (SORTING)

Capaian Pembelajaran :

1. Mampu memahami pengertian algoritma pengurutan
2. Mampu mampu membuat dan mendeklarasikan struktur algoritma pengurutan
3. Mampu mampu menerapkan dan mengimplementasikan algoritma pengurutan

Algoritma pengurutan (*sorting*) adalah kumpulan langkah sistematis atau secara berutan untuk memperoleh hasil yang diinginkan. Algoritma pengurutan merupakan proses menyusun elemen – elemen dengan tata urut tertentu dan proses tersebut terimplementasi dalam bermacam aplikasi. Tujuannya dari algoritma pengurutan adalah untuk mendapatkan kemudahan dalam pencarian suatu himpunan.

Secara umum, ada dua jenis pengurutan data, yaitu:

1. Model urut naik (*ascending*), yang mengurutkan data dari yang mempunyai nilai terkecil sampai terbesar.
2. Model urut turun (*descending*), yang mengurutkan data dari yang mempunyai nilai terbesar sampai terkecil.

Keuntungan dari data yang sudah dalam keadaan terurutkan antara lain :

1. Data mudah dicari (misalnya dalam buku telepon atau kamus bahasa), mudah untuk dibetulkan, dihapus, disisipi atau digabungkan.
2. Melakukan kompilasi program komputer jika tabel-tabel simbol harus dibentuk

3. Mempercepat proses pencarian data yang harus dilakukan berulang kali.
4. Pemilihan algoritma sangat ditentukan oleh struktur data yang digunakan.

8.1. Bubble Sort

Bubble Sort adalah salah satu metode dalam pengurutan suatu data. Pengurutan data *Bubble Sort* dengan cara membandingkan sebuah data dengan sebelahnya, kemudian dilakukan pertukaran data jika berada dalam urutan yang salah. Jika tidak ada pertukaran data lagi, berarti data-data tersebut sudah terurut.

Dengan menggunakan algoritma *Bubble sort*, dapat mengurutkan sebuah bilangan dari terkecil sampai terbesar (*Ascending*) maupun bilangan dari terbesar sampai dengan terkecil (*Descending*).

Pripsip - prinsip pada algoritma *Bubble Sort*:

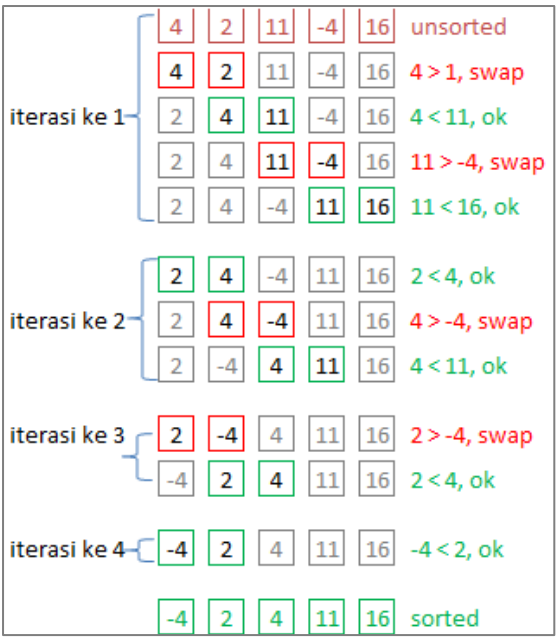
1. Jumlah proses yang dilakukan sama dengan banyaknya bilangan dikurang 1.
2. Setiap proses yang dilakukan, jumlah pertukaran bilangannya sama dengan banyaknya bilangan.
3. Dalam Bubble Sort, walaupun deretan bilangan tersebut sudah ter-sorting maka, proses sorting akan tetap dilakukan.
4. Tidak ada perbedaan cara untuk Bubble Sort Ascending dan Descending.

Berikut ini adalah sistem kerja dari algoritma *Bubble Sort* dari sebuah n elemen data yang belum terurut (*unsorted data*):

1. Bandingkan data ke-i dengan data ke-(i+1) . Jika tidak sesuai lakukan pertukaran data untuk menempati urutan yang benar dimana : data ke-i = data ke-(i+1) dan data ke-(i+1) = data ke-i. Urutannya bergantung apakah data tersebut mau diurutkan secara ascending (A-Z) maupun secara descending (Z-A).

2. Lalu bandingkan data berikutnya : data ke-(i+1) dengan data ke-(i+2). Jika tidak sesuai urutannya, tukar lagi data tersebut. Proses perbandingan data dilanjutkan sampai data terakhir.
3. Lakukan sampai beberapa kali iterasi, sampai didapat kondisi tidak terjadinya pertukaran data dalam satu iterasi. Artinya kita sudah mendapatkan data yang terurut.

Berikut gambar logika *Bubble Sort :*



Gambar 14 – logika algoritma Bubble Sort

Contoh soal:

Buatlah program java dari contoh logika algoritma *bubble sort* yang telah dijelaskan!

Implementasi :

Implementasi program

```
import java.util.Scanner;
public class BubbleSort
{
    public static void main(String[] args)
    {
        //Buat Objek Scanner
        Scanner scan = new Scanner(System.in);

        //Input jumlah Data
        System.out.print("Masukkan jumlah Data : ");
        int jlh_data = scan.nextInt();

        //Input nilai tiap Data
        int[] data = new int[jlh_data]; //Array untuk menampung nilai tiap Data
        System.out.println();
        for(int a = 0; a < jlh_data; a++)
        {
            System.out.print("Nilai Data ke-"+(a+1)+" : ");
            data[a] = scan.nextInt();
        }

        //Tampilkan Data Sebelum di Sorting
        System.out.println("\nData Sebelum di Sorting");
        for(int a = 0; a < jlh_data; a++)
            System.out.print(data[a]+" ");


        //Proses Bubble Sort
        System.out.println("\nProses Bubble Sort");
        for(int a = 0; a < jlh_data; a++)
        {
            System.out.println("Iterasi ke-"+(a+1)+" :");
            for(int b = 0; b < jlh_data; b++)
                System.out.print(data[b]+"  ");

            System.out.println("   Bandingkan "+data[0]+" dengan "+data[1]);
            for(int b = 0; b < jlh_data-1; b++)
            {
                String pesan = "   Tidak ada pertukaran";
                if(data[b] > data[b+1])
                {
                    //proses pertukaran nilai Data
                    pesan = "   Data "+data[b]+" ditukar dengan "+data[b+1];
                    int temp = data[b];        //Variable Sebagai pihak ketiga
                    data[b] = data[b+1];
                    data[b+1] = temp;
                }

                if(b < jlh_data-(a+1))
                {
                    for(int c = 0; c < jlh_data; c++)
                        System.out.print(data[c]+"  ");

                    System.out.println(pesan);;
                }
            }


            System.out.println("\n");
        }

        //Tampilkan Data Setelah di Sorting
        System.out.print("Data Setelah di Sorting : ");
        for(int a = 0; a < jlh_data; a++)
            System.out.print(data[a]+"  ");

    }
}

```

Hasil Output

```
Masukkan jumlah Data : 5

Nilai Data ke-1 : 4
Nilai Data ke-2 : 2
Nilai Data ke-3 : 11
Nilai Data ke-4 : -4
Nilai Data ke-5 : 16

Data Sebelum di Sorting
4 2 11 -4 16
Proses Bubble Sort
Iterasi ke-1 :
4  2  11  -4  16     Bandingkan 4 dengan 2
2  4  11  -4  16     Data 4 ditukar dengan 2
2  4  11  -4  16     Tidak ada pertukaran
2  4  -4  11  16     Data 11 ditukar dengan -4
2  4  -4  11  16     Tidak ada pertukaran


Iterasi ke-2 :
2  4  -4  11  16     Bandingkan 2 dengan 4
2  4  -4  11  16     Tidak ada pertukaran
2  -4  4  11  16     Data 4 ditukar dengan -4
2  -4  4  11  16     Tidak ada pertukaran


Iterasi ke-3 :
2  -4  4  11  16     Bandingkan 2 dengan -4
-4  2  4  11  16     Data 2 ditukar dengan -4
-4  2  4  11  16     Tidak ada pertukaran


Iterasi ke-4 :
-4  2  4  11  16     Bandingkan -4 dengan 2
-4  2  4  11  16     Tidak ada pertukaran


Iterasi ke-5 :
-4  2  4  11  16     Bandingkan -4 dengan 2


Data Setelah di Sorting : -4  2  4  11  16
```

8.2. Quick Sort

Quick Sort merupakan suatu algoritma pengurutan data yang menggunakan teknik pemecahan data menjadi partisi-partisi, sehingga metode ini disebut juga dengan nama partition exchange sort. Untuk memulai irterasi pengurutan, pertama-tama sebuah elemen dipilih dari data, kemudian elemen-elemen data akan diurutkan diatur sedemikian rupa.

Logika algoritma *quick sort* adalah:

8.2.1.1.1. Pilih satu elemen secara acak sebagai pivot

8.2.1.1.2. Pindahka semua elemen yang lebih kecil ke sebelah kiri pivot dan semua elemen yang lebih besar ke sebelah kanan pivot. Elemen yang nilainya sama bisa disimpan di salah satunya.

8.2.1.1.3. Lakukan sort secara rekursif terhadap sub-array sebelah kiri dan kanan pivot

Contoh soal :

Buatlah program java dari contoh algoritma *quick e sort* yang telah dijelaskan!

Implementasi :

Implementasi program

```
1  public class QuickSort {
2      public static void main(String args[])
3      {
4            int i;
5            int array[] = {14,9,4,69,125,1,3,11,13};
6
7            System.out.println(" Quick Sort\n\n");
8            System.out.println("Sebelum Sorting:\n");
9            for(i = 0; i < array.length; i++)
10           System.out.print( array[i]+"  ");
11           System.out.println();
12           quick_srt(array,0,array.length-1);
13           System.out.print("Setelah Sorting:\n");
14           for(i = 0; i <array.length; i++)
15           System.out.print(array[i]+"  ");
16           System.out.println();
17
18     }
19
20     public static void quick_srt(int array[],int low, int n)
21     {
22         int lo = low;
23         int hi = n;
24         if (lo >= n)
25         {
26           return;
27         }
28
29         int mid = array[(lo + hi) / 2];
30         while (lo < hi)
31         {
32           while (lo<hi && array[lo] < mid)
33           {
34               lo++;
35           }
36           while (lo<hi && array[hi] > mid)
37           {
38               hi--;
39           }
40           if (lo < hi)
41           {
42               int T = array[lo];
43               array[lo] = array[hi];
44               array[hi] = T;
45           }
46         }
47
48         if (hi < lo)
49         {
50            int T = hi;
51            hi = lo;
52            lo = T;
53         }
54
54
55         quick_srt(array, low, lo);
56         quick_srt(array, lo == low ? lo+1 : lo, n);
57     }
58
59 }
```

Hasil Output	Quick Sort Sebelum Sorting: 14 9 4 69 125 1 3 11 13 Setelah Sorting: 1 3 4 9 11 13 14 69 125

8.3. Exchange Sort

Exchange sort sangat mirip dengan *Bubble Sort*. Banyak yang mengatakan *Bubble Sort* sama dengan *Exchange Sort*. Pebedaannya adalah dalam hal bagaimana membandingkan antar elemen-elemennya.

Contoh soal :
Buatlah program java dari contoh algoritma *exchange sort* yang telah dijelaskan!

Implementasi :

Implementasi program

```java
import java.util.Scanner;
public class ExchangeSort
{
    public static void main(String[]args)
    {
        int[] data = new int [1000]; //tipe data int data maksimal 1000
        int n;
        Scanner msk = new Scanner(System.in);
        //masukan n
        System.out.print("Masukan Jumlah Data  = ");//input bilangan
        n = msk.nextInt();

        for(int i=0;i<=n-1;i++)
        {
            System.out.print("Masukan data = ");

            data[i]=msk.nextInt();
        }
        for(int i=0;i<n-2;i++)
        {
            for(int j=1;j<=n-1;j++)
            {
                if(data[i]>data[j])
                {
                    int temp=data[i];
                    data[i]=data[j];
                    data[j]=temp;
                }
            }
        }
        System.out.println();
        for(int i=0;i<=n-1;i++)
        {
            System.out.print("Data telah diurutkan  = "); //output
            System.out.println(data[i]);
        }
    }
}
```

Hasil Output

```
Masukan Jumlah Data  = 5
Masukan data = 4
Masukan data = 2
Masukan data = 11
Masukan data = -4
Masukan data = 15

Data telah diurutkan  = -4
Data telah diurutkan  = 11
Data telah diurutkan  = 2
Data telah diurutkan  = 4
Data telah diurutkan  = 15
```

8.4. Insertion Sort

Algoritma insertion sort pada dasarnya memilah data yang akan diurutkan menjadi dua bagian, yang belum diurutkan dan yang sudah diurutkan. Elemen pertama diambil dari bagian array yang belum diurutkan dan kemudian diletakkan sesuai posisinya pada bagian lain dari array yang telah diurutkan. Langkah ini dilakukan secara berulang hingga tidak ada lagi elemen yang tersisa pada bagian array yang belum diurutkan.

Keuntungan menggunakan insertion sort adalah :

1. Dapat diimplementasikan dengan simpel.
2. Sangat efisien untuk data berukuran kecil.
3. *Insertion sort* dapat langsung menyortir list data ketika menerima input.
4. Secara praktikal lebih efisien dibandingkan dengan *selection* dan *bubble sort*.

```
Data : 12 9 3 20 30 1

Proses Insertion Sort (Ascending)

Iterasi 1:
12  9  3  20  30  1 → (Bandingkan Data 9 dengan 12)
9  12  3  20  30  1 → (Tukar Data 9 dengan 12)

Iterasi 2:
9  12  3  20  30  1 → (Bandingkan Data 3 dengan 12)
9  3  12  20  30  1 → (Tukar 3 dengan 12. Bandingkan 3 dengan
9)
3  9  12  20  30  1 → (Tukar 3 dengan 9)

Iterasi 3:
3  9  12  20  30  1 → (Bandingkan 20 dengan 12)
3  9  12  20  30  1 → (Tidak ada pertukaran)
```

```
Iterasi 4:
3  9  12  20  30  1 → (Bandingkan 30 dengan 20)
3  9  12  20  30  1 → (Tidak ada pertukaran)

Iterasi 5:
3  9  12  20  30   1 → (Bandingkan 1 dengan 30)
3  9  12  20  1   30 → (Tukar 1 dengan 30. Bandingkan 1 dengan
20)
3  9  12  1   20  30 → (Tukar 1 dengan 20. Bandingkan 1 dengan
12)
3  9  1  12   20  30 → (Tukar 1 dengan 12. Bandingkan 1 dengan
9)
3  1  9  12   20  30 → (Tukar 1 dengan 9. Bandingkan 1 dengan
3)
1  3  9  12   20  30 → (Tukar 1 dengan 3)
```

Contoh soal :

Buatlah program java dari contoh algoritma *insertion sort* yang telah dijelaskan!

Implementasi :

Implementasi program

```
import java.util.Scanner;
public class InsertionSort
{
    public static void main(String[] args)
    {
        Scanner in = new Scanner(System.in);
        System.out.print("Banyak data : ");
        int N = in.nextInt();
        int data[] = new int[N];

        for(int i=0; i<N; i++)
        {
            System.out.print("data ke-"+(i+1)+" : ");
            data[i] = in.nextInt();
        }

        //prose isertion sort
        for(int i=1; i<data.length; i++)
        {
            int key = data[i];
            int j=i;
            while(j >0 && data[j-1]>key)
            {
                data[j]=data[j-1];
                j--;
            }

            data[j]=key;
        }
        //hasil pengurutan
        System.out.print("Data yang telah urut : ");
        for(int i=0; i<data.length; i++)
        {
            System.out.print(data[i]+" ");
        }
        System.out.println();
    }
}
```

Hasil Output

```
Banyak data : 5
data ke-1 : 4
data ke-2 : 2
data ke-3 : 11
data ke-4 : -4
data ke-5 : 15
Data yang telah urut : -4 2 4 11 15
```

8.5. Selection Sort

Selection Sort adalah pengurutan dengan cara mencari nilai elemen yang terbesar atau yang terkecil dari sekumpulan elemen nilai pada sebuah data. Selection Sort merupakan salah satu metode

pengurutan yang memiliki algoritma yang cukup gampang dalam penulisan coding-nya. Dibanding Bubble Sort, Selection Sort jelas lebih baik dari segi kecepatan proses pengurutannya. Karena, Inti dari algoritma Selection Sort ialah mencari nilai yang paling kecil(Jika Ascending) atau nilai yang paling besar(Jika Descending) di urutan Data berikutnya. Untuk lebih jelasnya saya beri contoh kasus seperti berikut ini.

Logika Pengurutan Selection Sort sebagai berikut :

1. Mencari nilai elemen max atau min (terserah, atau pilih salah satu) pada semua nilai elemen pada array yang seharusnya (minimal pada pertama atau nilai max pada akhir). kemudian elemen array tersebut di tetapkan atau di isolasi dan tidak di ganggu lagi.
2. Temukan sebuah elemen array yang memilikidi nilai kecil atau besardari index kedua dari elemen awal jika terkecil atau dari akhir jika terbesar, setelah itu tukarkan eleman tersebut dengan elemen array pada posisi (indeks) kedua (dari awal atau dari akhir tergantung penggunaan untuk mencari nilai terkecil atau dari yang terbesar), kemudian isolasi atau tetapkan elemen array tersebut ditambah dengan elemen array yang sebelumnya.
3. Lakukan langkah seperti diatas pada elemen berikutnya sampai elemen terakhir.

Contoh soal:

Buatlah program java dari contoh algoritma logika *selection sort* yang telah dijelaskan!

Implementasi :

Implementasi program

```
import java.util.Scanner;
public class SelectionSort
{
    public static void main(String[] args)
    {
        //Buat Objek Scanner
        Scanner scan = new Scanner(System.in);

        //Input jumlah Data
        System.out.print("Masukkan jumlah Data : ");
        int jlh_data = scan.nextInt();

        //Input nilai tiap Data
        int[] data = new int[jlh_data];  //Array untuk nilai tiap Data
        System.out.println();
        for(int x = 0; x < jlh_data; x++)
        {
            System.out.print("Input nilai Data ke-"+(x+1)+" : ");
            data[x] = scan.nextInt();
        }

        //Tampilkan Data Sebelum di sorting
        System.out.println();
        System.out.print("Data Sebelum di Sorting : ");
        for(int x = 0; x < jlh_data; x++)
            System.out.print(data[x]+" ");

        //Proses Selection Sort
        System.out.println("\n\nProses Selection Sort");
        for(int x = 0; x < jlh_data-1; x++)
        {
            System.out.println("Iterasi ke-"+(x+1)+" : ");
            for(int y = 0; y < jlh_data; y++)
                System.out.print(data[y]+" ");

            System.out.println("   Apakah Data "+data[x]+" "
                    + "sudah benar pada urutannya?");

            boolean tukar = false;
            int index = 0;
            int min = data[x];
            String pesan = "   Tidak Ada Pertukaran";
            for(int y = x+1; y < jlh_data; y++)
            {
                if(min > data[y])
                {
                    tukar = true;
                    index = y;
                    min = data[y];
                }
            }

            if(tukar == true)
            {
                //Pertukaran Data
                pesan = "   Data "+data[x]+" ditukar dengan Data "
                        +data[index];
                int temp = data[x];
                data[x] = data[index];
                data[index] = temp;
            }

```

```
63              for(int y = 0; y < jlh_data; y++)
64                  System.out.print(data[y]+" ");
65
66              System.out.println(pesan+"\n");
67          }
68
69          //Tampilkan Data Setelah di Sorting
70          System.out.print("Data Setelah di sorting : ");
71          for(int x = 0; x < jlh_data; x++)
72              System.out.print(data[x]+" ");
73      }
74  }
```

Hasil Output

```
 Masukkan jumlah Data : 5

Input nilai Data ke-1 : 4
Input nilai Data ke-2 : 2
Input nilai Data ke-3 : 11
Input nilai Data ke-4 : -4
Input nilai Data ke-5 : 15

Data Sebelum di Sorting : 4 2 11 -4 15

Proses Selection Sort
Iterasi ke-1 :
4 2 11 -4 15    Apakah Data 4 sudah benar pada
urutannya?
-4 2 11 4 15    Data 4 ditukar dengan Data -4

Iterasi ke-2 :
-4 2 11 4 15    Apakah Data 2 sudah benar pada
urutannya?
-4 2 11 4 15    Tidak Ada Pertukaran

Iterasi ke-3 :
-4 2 11 4 15    Apakah Data 11 sudah benar pada
urutannya?
-4 2 4 11 15    Data 11 ditukar dengan Data 4

Iterasi ke-4 :
-4 2 4 11 15    Apakah Data 11 sudah benar pada
urutannya?
-4 2 4 11 15    Tidak Ada Pertukaran

Data Setelah di sorting : -4 2 4 11 15
```

8.6. Latihan

8.6.1. Buat program dengan inputan untuk algoritma *quick sort*!

8.6.2. Gambarkan logika algoritma *Selection Sort*!

BAB 9
ALGORITMA PENCARIAN (SEARCHING)

Capaian Pembelajaran :

1. Mampu memahami pengertian algoritma pencarian
2. Mampu mampu membuat dan mendeklarasikan struktur algoritma pencarian
3. Mampu mampu menerapkan dan mengimplementasikan algoritma pencarian

Algoritma pencarian (*searching*) merupakan proses yang fundamental dalam pengolah data. Pencarian bertujuan untuk menemukan nilai (data) tertentu di dalam sekumpulan data bertipe sama.

Sebuah algoritma pencarian di'elaskan secara luas adalah sebuah algoritma yang menerimamasukan berupa sebuah masalah dan menghasilkan sebuah solusi untuk masalah tersebut yang biasanya didapat dari evaluasi beberapa kemungkinan solusi. Algoritma pencarian (*searching algorithm*) adalah algoritma yang menerima sebuah kata kunci dan dengan langkah - langkah tertentu akan mencari rekaman dengan kata kunci tersebut. Setelah proses pencarian dilaksanakan akan diperoleh salah satu dari dua kemungkinan yaitu data yang dicari ditemukan atau tidak ditemukan.

9.1. Sequential Searching

Teknik pencarian data dari array yang paling mudah adalah dengan cara sequential search, dimana data dalam array dibaca 1 demi satu, diurutkan dari index terkecil ke index terbesar, maupun sebaliknya.

Kelebihan dari proses pencarian secara *sequential* ini adalah jika data yang dicari terletak di depan, maka data akan ditemukan dengan cepat. Tetapi dibalik kelebihannya ini, teknik ini juga memiliki kekurangan. Pertama, jika data yang dicari terletak dibelakang atau paling akhir, maka akan membutuhkan waktu yang lama dalam proses pencariannya. Kedua, beban komputer akan semakin bertambah jika jumlah data dalam array sangat banyak.

Contoh :

```
Array
Int A[5] = {12, 13, 19, 27, 28}
```

Misalkan dari data di atas, data yang akan dicari adalah 19, maka proses yang akan terjadi adalah:

1. Pencarian dimulai pada index ke-0 yaitu angka 12, kemudian dicocokan dengan angka yang akan dicari, jika tidak sama makapencarian akan dilanjutkan ke index selanjutnya.
2. Pada index ke-1, yaitu angka 13, juga bukan angka yang dicari, maka pencarian juga akan dilanjutkan pada index selanjutnya.
3. Pada index ke-2, yaitu angka 19, ternyata angka 19 merupakan angka yang dicari. Pencarian angka telah ditemukan, maka pencarian akan dihentikan dan keluar dari looping pencarian.

Contoh soal:

Buat program java, untuk mencari nilai dengan menggunakan algoritma *sequential search* berdasarkan inputan user, dengan data yang dimasukan adalah 5data.

Implementasi :

Implementasi

```
1  import javax.swing.JOptionPane;
2  public class SequentialSearch
3  {
4      public static void main(String[] args)
5      {
6          int data[]=new int[5];
7          int i, angka, cari;
8          boolean ketemu;
9
10         for (i = 0; i < data.length; i++)
11         {
12             angka=Integer.parseInt(JOptionPane.showInputDialog
13                     ("Masukkan Angka ke-"+(i+1)));
14             data[i]=angka;
15         }
16
17         System.out.print("Angka Yang Dimasukkan = { ");
18         for(i=0;i<data.length;i++)
19         {
20             System.out.print(data[i]+" ");
21         }
22
23         System.out.println("}");
24         cari = Integer.parseInt(JOptionPane.showInputDialog
25                 ("Masukkan Angka Yang Ingin Dicari :"));
26         System.out.println("Angka Yang Dicari : "+cari);
27         ketemu = false;
28         for (i = 0; i < data.length; i++)
29         {
30             if(data[i] == cari)
31             {
32                 ketemu = true;
33                 break;
34             }
35         }
36
37         if(ketemu)
38         {
39             System.out.println("Angka "+cari+" "
40                     + "ditemukan dalam urutan ke-" + (i+1));
41         }
42         else
43         {
44             System.out.println("Angka "+cari+" tidak ditemukan");
45         }
46     }
47 }
48
```

Hasil Output


```
Angka Yang Dimasukkan = { 6 9 4 2 5 }
Angka Yang Dicari : 2
Angka 2 ditemukan dalam urutan ke-4
```

9.2. Binary Search

Algoritma binary search atau pencarian biner merupakan metode pencarian pada data terurut yang paling efesien. Metode pencarian biner digunakan untuk kebutuhan pencarian dengan waktu yang cepat.

Pencarian Biner (*Binary Search*) dilakukan untuk :

1. Memperkecil jumlah operasi pembandingan yang harus dilakukan antara data yang dicari dengan data yang ada khususnya untuk jumlah data yang sangat besar ukurannya.
2. Prinsip dasarnya adalah melakukan proses pembagian ruang pencarian secara berulang-ulang sampai data ditemukan atau sampai ruang pencarian tidak dapat dibagi lagi (berarti ada kemungkinan data tidak ditemukan)
3. Syarat utama untuk *binary search* adalah data harus sudah terurut, misalkan terurut menaik.

Algoritma *binary search*:

1. Data diambil dari posisi 1 sampai posisi akhir N.
2. Kemudian cari posisi data tengah dengan rumus: (posisi awal + posisi akhir) / 2
3. Kemudian data yang dicari dibandingkan dengan data yang di tengah, apakah sama atau lebih kecil, atau lebih besar?
4. Jika lebih besar, maka proses pencarian dicari dengan posisi awal adalah posisi tengah + 1
5. Jika lebih kecil, maka proses pencarian dicari dengan posisi akhir adalah posisi tengah – 1
6. Jika data sama, berarti ketemu

Contoh soal:

Buat program dengan java, untuk mencari nilai pada indeks keberapa dengan menggunakan algoritma *binary search.*

Implementasi :

Implementasi Program

Hasil Output

```
import java.util.Scanner;
public class BinarySearch
{
    public static void main(String[]args){
        Scanner in = new Scanner (System.in);
        int n,i,j,cari;

        System.out.print("Masukkan banyaknya data :");
        n=in.nextInt();
        int [] data=new int [n];
        int [] temp=new int [n];

        boolean ada;
        for(i=0;i<n;i++)
        {
            System.out.print("Masukkan data ke-"+(i+1)+" =");
            data[i]=in.nextInt();
        }

        for(i=0;i<n;i++)
        {
            System.out.print(data[i]+" | ");
        }

        System.out.print("\nData yang akan dicari :");
        cari=in.nextInt();

        for(i=n-1;i>1;i--)
        {
            for(j=0;j<i;j++)
            {
                if(data[j]>data[j+1])
                {
                    int a;
                    a=data[j];
                    data[j]=data[j+1];
                    data[j+1]=a;
                }
            }
        }

        for(i=0;i<n;i++)
        {
            System.out.print(data[i]+" | ");
        }
        int bawah=0;
        int atas=data.length;
        ada=false;

```

```
50          while(bawah<atas)
51          {
52              i=(bawah+atas)/2;
53              if(data[i]==cari)
54              {
55                  ada=true;
56                  data[i]=data[i]*5;
57                  temp[i]=i+1;
58                  bawah=0;
59                  atas=data.length;
60              }
61              else if(data[i]<cari)
62              {
63                  bawah=i+1;
64              }
65              else {atas=i;}
66          }
67
68          for(i=0;i<n;i++)
69          {
70              if(data[i]==cari)
71              {
72                  data[i]=cari*5;
73              }
74          }
75
76          System.out.print("|");
77          if(ada)
78          {
79              System.out.print("\nData yang dicari berada pada index |");
80              for(i=0;i<n;i++)
81              {
82                  if(temp[i]!=0)
83                  {
84                      System.out.print(" "+(temp[i]-1)+" |");
85                  }
86              }
87          }
88          else { System.out.print("\nData tidak ada pada array ");}
89      }
90  }
```

```
Masukkan banyaknya data :5
Masukkan data ke-1 =8
Masukkan data ke-2 =1
Masukkan data ke-3 =4
Masukkan data ke-4 =9
Masukkan data ke-5 =12
```

9.3. Latihan

1. Buatlah sebuah program yang dapat melakukan pencarian karakter dalam kalimat string, kemudian hasil pencarian karakter tersebut diubah menjadi karakter lain dan ditampilkan.
2. Buat program input number secara random, kemudian lakukan searching banyak bilangan yang genap dan yang ganjil.

GLOSARIUM

***Activity* (aktivities) :** komponen yang dapat dilihat oleh pengguna, sehingga mereka dapat berinteraksi dengan aplikasi

Constructor : method khusus yang akan dieksekusi pada saat pembuatan objek (*instance*).

Deklarasi : memberitahu kompiler (*compiler*) Java tentang nama variabel dan tipe data yang diwakilinya.

Java Console : program khusus digunakan untuk membantu pengembang dalam menemukan dan memperbaiki bug di aplikasi

Package : sarana/cara pengelompokkan dan pengorganisasian kelas-kelas dan interface yang sekelompok menjadi suatu unit tunggal dalam library.

Protected : menyatakan bahwa kelas/method/attribute tersebut dapat diakses oleh kelas lain yang berada dalam satu package atau kelas lain tersebut merupakan turunannya.

Public : menyatakan bahwa kelas/method/attribute tersebut dapat diakses oleh kelas lain dimanapun.

Static : salah satu jenis modifier di Java yang digunakan agar suatu atribut atau pun method dapat diakses oleh kelas atau objek tanpa harus melakukan instansiasi terhadap kelas tersebut

Daftar Pustaka

Casnadi, E. (2013, Oktober 29). *slideshare.net.* Retrieved September 01, 2020, from Struktur Perulangan For: https://www.slideshare.net/casnadi/pengulangan-for

Fakultas Elektro dan Komunikasi. (2013). *Modul Praktikum Pemrograman Berorientasi Objek.* Bandung: Institut Teknologi Bandung.

Jando, E., & Nani, P. A. (2018). *Algoritma Pemrograman dengan Bahasa JAVA.* Yogyakarta: Penerbit ANDI.

Pendidikan Teknik Informatika. (2010). *Modul Praktikum Algoritma dan Struktur Data.* Malang: Universitas Negeri Malag.

Ramadhani, C. (2015). *Dasar Algoritma dan Struktur Data dengan Bahasa JAVA.* Yogyakarta: Penerbit ANDI.

Setiawan, Y. (2015, Juni 24). Retrieved September 01, 2020, from jagocoding.com: http://jagocoding.com/tutorial/776/Metode_Sequential_Searching_di_Java_Console

Setiawan, Y. (2015, Juni 23). Retrieved September 01, 2020, from Jagocoding.com: http://jagocoding.com/tutorial/764/Bubble_Sort_dan_Selection_Sort_di_Java_Console

Sitorus, L. (2015). *Algoritma dan Pemrograman.* Yogyakarta: Penerbit ANDI.

Suprapto, Yuwono, K. T., Sukardiyono, T., & Dewanto, A. (2008). *Bahasa Pemrograman Untuk SMK.* Direktorat Pembinaan Sekolah Menengah Kejuruan.

Suryadi, & Sumin, A. (1997). *Pengantar Algoritma dan Pemrograman Teknik Diagram Alur dan Bahasa Basic Dasar*. Jakarta: Gunadarma.

Yatini, I., & Sumiatun. (2009). *Modul Algoritma dan Pemrograman Java.* Yogyakarta: Akakom.

Konsep Dasar Algoritma dan Pemrograman Dengan Bahasa Java

Evi Pratiwi

Penulis mengucapkan terimakasih yang teramat dalam kepada pihak yang telah memberikan dukungan penuh terutama Jurusan Administrasi Bisnis, Program Studi Manajemen Infomatika, dan unit P3M Politeknik Negeri Banjarmasin, dan pihak yang tidak bisa diucapkan satu per satu, schingga bahan ajar ini dapat tersusun dengan baik.

Dalam penulisan bahan ajar ini tentu masih banyak kekurangan. Oleh karena itu, saran dan kritik yang membangun agar dapat menyempurnakan bahan ajar menjadi lebih baik lagi. Semoga bahan ajar ini dapat bermanfaat bagi kita semua.

Penerbit Poliban Press
Redaksi :
Politeknik Negeri Banjarmasin, Jl. Brigjen H. Hasan Basry, Pangeran, Komp. Kampus ULM, Banjarmasin Utara
Telp : (0511)3305052
Email : press@poliban.ac.id

ISBN 978-623-7694-45-8 (PDF)