

# Pengantar Algoritma dan Pemrograman

Kani, M.Kom.



## PENDAHULUAN

---

Modul ini akan menyampaikan informasi dan menjelaskan pengertian algoritma, kelebihan dan kelemahan penggunaan algoritma, mengklasifikasi algoritma, membuat algoritma, selain itu juga akan menjelaskan tentang *flowchart* serta kelebihan dan kelemahannya.

Secara khusus setelah mempelajari modul ini, mahasiswa diharapkan mampu:

1. mendefinisikan algoritma;
2. menyebutkan alasan menggunakan komputer;
3. mendefinisikan program komputer;
4. mendefinisikan dan menyebutkan kelompok besar program komputer;
5. mendefinisikan *programmer*;
6. mendefinisikan bahasa pemrograman;
7. membagi bahasa pemrograman berdasarkan fungsi;
8. mengekspresikan algoritma;
9. menyebutkan manfaat algoritma;
10. menyebutkan manfaat penggunaan algoritma;
11. menyusun algoritma;
12. Mengetahui dan mengklasifikasi algoritma-algoritma umum yang ada.

**KEGIATAN BELAJAR 1**

## Pengantar Algoritma dan Pemrograman Komputer

Berkembangnya ilmu pengetahuan membuat manusia mampu menghasilkan karya yang semakin canggih. Dengan kemajuan teknologi, mereka selalu mencari formulasi yang tepat sehingga semakin algoritma optimal, sistematis dan logis. Saat ini manusia telah mampu menciptakan komputer canggih dengan kecerdasan buatan (*Artificial Intelligence*).

Komputer tidak otomatis mampu atau bisa menyelesaikan masalah begitu saja. Ada urutan langkah yang dimulai dari analisa masalah, penyusunan langkah penyelesaian (algoritma) oleh analis, kemudian diterjemahkan/menuliskan langkah-langkah ke dalam bahasa program komputer tertentu oleh programmer, kemudian dikompilasi dan dijalankan sehingga menemukan output yang diinginkan. Dalam dunia pemrograman, desain program yang tepat, bahkan sampai pada pemberian komentar pada setiap proses membantu *programmer* lain untuk menjaga dan memperbaiki fungsi program di masa depan.

### A. PENGERTIAN ALGORITMA

Perpaduan antara perkembangan teknologi mikroelektronika dan desain *chip* memberikan sumbangsih berupa perangkat keras yang lebih cepat dan tepat guna. Penggunaan algoritma efisien (teroptimasi) dan disertai berbagai macam ilmu pengetahuan yang tercakup dalam suatu kecerdasan buatan telah memberikan efek revolusi komputer dan aplikasinya.

Efek revolusi komputer dan lonjakan kemajuan teknologi menjadi titik awal dimulainya beberapa pekerjaan manusia sudah mulai diambil alih oleh mesin. Banyak permasalahan yang bisa terselesaikan dengan teknologi terbaru saat ini. Dunia pendidikan tidak perlu memberikan khayalan kepada murid jika ingin mengambil objek beruang kutub, seorang dokter kandungan tidak perlu menebak jenis kelamin anak yang dikandung oleh pasiennya, seorang arsitek tidak perlu menggambar melalui kertas lagi, pembeli tak perlu datang ke supermarket untuk sekedar membeli kebutuhan rumah tangga dan banyak lagi hal yang lainnya. Semua bisa diselesaikan dengan teknologi komputer dengan

layanan-layanan aplikasi yang dibuat oleh pengembang. Layanan-layanan aplikasi seperti ini tentu diawali dengan kegiatan pembuatan desain hingga menjadi suatu algoritma yang kemudian direalisasikan dalam bentuk program komputer.

Definisi algoritma menurut beberapa pakar adalah sebagai berikut:

1. *Abu Ja'far Muhammad Ibnu Musa Al-Khawarizmi*: Algoritma adalah suatu metode khusus untuk menyelesaikan suatu masalah.
2. *Donald E. Knuth*: Algoritma adalah sekumpulan aturan-aturan berhingga yang memberikan sederetan proses-proses untuk menyelesaikan suatu masalah yang khusus.
3. *David Bolton*: Algoritma adalah deskripsi dari suatu prosedur yang berakhir dengan sebuah *output*.
4. *Stone dan Knuth*: Algoritma adalah suatu seperangkat aturan yang tepat mendefinisikan urutan operasi hingga sedemikian rupa sehingga setiap aturan yang efektif, jelas hingga sedemikian rupa sehingga urutan berakhir dalam waktu yang terbatas.
5. *Andrey Andreyevich Markov*: Algoritma adalah hal umum untuk dipahami sebagai suatu keputusan yang tepat untuk mendefinisikan proses komputasi yang mengarahkan dari data awal hingga hasil yang diinginkan.

Jika dikolaborasi dari definisi algoritma di atas maka definisi dari algoritma adalah suatu upaya dengan urutan operasi yang disusun secara logis dan sistematis untuk menyelesaikan suatu masalah untuk menghasilkan suatu *output* tertentu.

Dalam komputasi, algoritma sangat penting karena berfungsi sebagai prosedur sistematis yang diperlukan komputer. Algoritma yang baik adalah bagaikan menggunakan alat yang tepat di bengkel. Penggunaan algoritma yang salah adalah bagaikan mencoba memotong sepotong kayu dengan menggunakan gunting yang tentu tidak efektif. Penggunaan gunting tersebut juga akan membutuhkan waktu yang lama dalam menyelesaikan permasalahan.

Untuk memperluas pemahaman kita tentang konsep algoritma dengan cara yang lebih baik, kita cermati kasus berikut: Katakanlah Anda mau melakukan sebuah perjalanan dari Jakarta ke Bogor, Anda diperhadapkan dengan berbagai pilihan untuk mencapai tujuan, misalnya dengan naik kereta, naik taksi atau naik angkot. Ketiga pilihan yang ada mempunyai kelebihan dan kelemahan baik dari sisi waktu dan biaya.

*Naik Kereta*

- ✓ Ke stasiun terdekat
- ✓ Membeli tiket kartu Jakarta - Bogor
- ✓ Gesek kartu untuk membuka pintu masuk
- ✓ Menunggu keberangkatan kereta
- ✓ Jika kursi penuh maka berdiri
- ✓ Sampai tujuan Bogor

*Naik Taksi*

- ✓ Memesan taksi lewat Android atau telepon langsung
- ✓ Menunggu taksi untuk menjemput
- ✓ Naik taksi
- ✓ Lewat jalur biasa atau lewat jalan tol
- ✓ Sampai tujuan
- ✓ Bayar taksi sesuai argo

*Naik Angkot*

- ✓ Ke terminal atau pangkalan angkot
- ✓ Menunggu angkot penuh
- ✓ Melalui jalur biasa
- ✓ Sampai Bogor
- ✓ Bayar sesuai tarif angkot.

Ketiga algoritma di atas mencapai tujuan yang sama, namun masing-masing algoritma melakukannya dengan cara yang berbeda. Setiap algoritma juga memiliki biaya yang berbeda dan waktu perjalanan yang berbeda. Dengan naik taksi misalnya, adalah cara tercepat, tapi juga yang paling mahal. Naik angkot, jauh lebih murah akan tetapi memakan waktu lebih banyak dari naik taksi, begitu juga dengan naik kereta, menunggu jadwal kereta, singgah di setiap stasiun dan walaupun bebas hambatan. Setiap pilihan punya kelebihan dan kekurangan.

**B. ALASAN MENGGUNAKAN ALGORITMA**

Istilah algoritma digunakan dalam ilmu komputer atau informatika untuk mendeskripsikan metode pemecahan masalah yang terbatas, deterministik, dan efektif sesuai tujuan implementasi suatu program komputer. Algoritma

merupakan salah satu bidang/bagian dari ilmu komputer dan objek penelitian utama di lapangan sampai saat ini. Algoritma adalah prosedur pemecahan masalah dalam bahasa sangat alami (bahasa manusia), langkah-langkah pemecahan masalah dalam algoritma nantinya akan dituangkan menjadi program komputer untuk mempercepat/mengotomasi penyelesaian masalah. Sedangkan untuk bahasa pemrograman yang digunakan untuk menuangkan algoritma kedalam bahasa program sangat tergantung selera dan penguasaan pada individu *programmer*.

Salah satu alasan utama mempelajari algoritma dilihat dari kacamata disiplin ilmu adalah bahwa algoritma adalah sebuah keterampilan yang memberi potensi untuk memecahkan masalah serumit apapun dengan waktu penyelesaian proses/eksekusi singkat, bahkan mungkin bisa meringkas langkah kerja yg tidak efisien menjadi otomatis. Dalam sebuah aplikasi bisa saja memproses jutaan objek, fungsi atau *procedure* (langkah-langkah kerja) penyelesaian masalah. Program yang dirancang dengan menggunakan algoritma yang tepat sangat mungkin membuat program jutaan kali lebih cepat dibanding dengan program dengan sebuah algoritma dengan desain asal-asalan. Pada modul-modul selanjutnya akan diberikan beberapa contoh kecil efisiensi penggeraan suatu masalah yang bisa diselesaikan secara dengan logika algoritma yang tepat. Algoritma yang tepat akan memberikan efek yang signifikan terhadap waktu dan tenaga. Tidak sedikit perusahaan yang saat ini mau dan rela menginvestasikan uang tambahan untuk membeli dan memasang komputer baru untuk mempercepat pekerjaan mereka, tidak hanya perangkat keras, mereka mau menggunakan/membeli (membayar lisensi) sebuah algoritma untuk kepentingan perusahaan.

Dilihat dari disiplin ilmu maka berikut alasan mengadopsi atau menggunakan algoritma sebagai berikut:

1. *Efisiensi*: Untuk mengukur sebuah algoritma yang efisien harus mempertimbangkan efisiensi waktu-CPU dan memori. Terkadang *programmer* hanya berhenti kepada hasil tepat, akan tetapi tidak mempertimbangkan waktu dan memori yang terkuras oleh algoritma yang digunakan. Basis untuk membuat algoritma adalah efisiensi waktu, memori dan keluaran yang tepat. Walaupun tidak bisa dielakkan bahwa bahwa setiap orang akan memiliki cara berpikir/logika dalam menyelesaikan masalah yang berbeda-beda walau menghasilkan solusi

yang sama. Dalam algoritma kecepatan dan ruang memori harus mampu diseimbangkan untuk menghasilkan solusi cepat dan tepat.

2. *Abstraksi*: Kelebihan dari pada algoritma adalah mampu memperlihatkan sebuah permasalahan yang tingkat kerumitannya besar lalu kemudian dapat diurai menjadi kelihatan mudah dan sederhana, gambaran kerumitan terkikis dengan alur algoritma yang tersusun baik dan jelas.
3. *Reusability*: Algoritma adalah metode bukan program, artinya bahwa algoritma harus mampu digunakan tanpa melihat bahasa pemrograman yang digunakan, dapat digunakan kembali dan bahkan berkali-kali pada berbagai situasi untuk menerapkan dalam bahasa pemrograman apapun.

## C. PENGERTIAN PROGRAM DAN BAHASA PEMROGRAMAN

Program dan Bahasa Pemrograman adalah sebuah istilah yang tidak bisa dipisahkan. Program adalah set intruksi dan bahasa pemrograman adalah intruksi standar dalam membuat program.

### 1. Program

Program adalah satu set intruksi yang berkode yang dapat dimengerti oleh komputer untuk memecahkan masalah atau menghasilkan hasil yang diinginkan. Terdapat dua macam kelompok besar program komputer, yaitu:

- a. Sistem Operasi Komputer (*Computer Operating System* atau lebih dikenal dengan *OS*), yakni program komputer yang menyediakan intruksi paling mendasar yang digunakan komputer dalam operasinya. OS merupakan perangkat lunak sistem yang mengelola perangkat keras komputer, sumber daya perangkat lunak, dan menyediakan layanan umum untuk program komputer lainnya. Contoh Sistem Operasi *Windows*, *Linux*, *MacOS*.
- b. Program Aplikasi, yang berjalan pada sistem operasi dan melakukan pekerjaan sesuai tujuan kehendak kita misal pengolah kata, perhitungan (olah data), presentasi video, suara dan sebagainya. Suatu program umumnya ditulis dengan menggunakan suatu bahasa pemrograman tingkat tinggi, seperti: Java, C/C++, Python, PHP dan sebagainya. Pada awal kehadirannya program komputer dibuat dengan bahasa tingkat rendah, seperti: Bahasa *Assembly* atau bahasa mesin. Kehadiran generasi bahasa pemrograman tingkat tinggi menjadikan bahasa pemrograman

tingkat rendah menjadi kurang diminati, penyebab utamanya adalah bahasa pemrograman tingkat tinggi lebih menyerupai bahasa manusia.

## 2. Bahasa Pemrograman Pertama

Bahasa Pemrograman pertama di dunia sudah berusia lebih dari 100 tahun. Ditulis oleh seorang wanita bernama Augusta Ada Byron yang lahir pada 10 Desember 1815. Dia adalah putri penyair ternama, Lord Byron. Lima pekan setelah Ada Lahir, ibunya minta cerai dari ayahnya dan hak asuh jatuh kepada Lady Byron. Alasan kuat perceraian kedua orang tuanya adalah karena ibunya tidak ingin anaknya kelak mengikuti jejaknya ayahnya menjadi seorang penyair. Berbagai upaya dilakukan oleh ibunya untuk mengalihkan anaknya dari dunia syair, diantaranya adalah dengan mengajaknya dan mengenalkannya kepada banyak ilmuwan dan ahli matematik, walaupun pada kenyataannya bahwa darah ayahnya tidak benar-benar hilang dari dirinya. Terbukti pada usia 30 tahun, dia mencampur adukkan ilmu matematika dengan imajinasi, dan dituangkan dalam bentuk tulisan-tulisan metafora.

Pada usia 17 tahun, Ada diperkenalkan ke Mary Somerville, seorang wanita luar biasa yang menerjemahkan karya LaPlace ke dalam bahasa Inggris. Mrs. Somerville memotivasi Ada untuk memperkuat ilmu matematikanya, mencoba membentuk pola pikir Ada yang menyatukan ilmu matematika dan teknologi yang layak digunakan oleh manusia. Pada suatu pesta makan malam pada bulan November 1834, Mrs. Somerville mendengar ide Babbage untuk membuat mesin penghitung baru yaitu mesin analitikal, dengan kemampuan bukan hanya bisa memprediksi akan tetapi mesin tersebut bisa memberikan misi futuristik dan hal ini memberi titik awal menemukan bahasa programnya yang pertama.

Babbage mengerjakan rencana mesin baru ini dan menseminarkan perkembangannya di Turin, Italia pada musim gugur tahun 1871. Seorang ahli matematik Italia bernama Menabrea, menulis sebuah rangkuman dari apa yang Babbage uraikan dan menerbitkan sebuah artikel berbahasa Perancis tentang perkembangan mesin analitikal tersebut. Ada menerjemahkan dan mengurai tulisan Menabrea tersebut, kemudian memperlihatkan hasilnya kepada Babbage. Babbage merasa kagum dengan tulisan Ada yang 3 kali lebih panjang dari tulisan Menabrea. Dari hasil tulisan Ada, Babbage menyarankan untuk menjadikan artikelnya sendiri dan kemudian mempublikasikan secara luas. Catatan ilmiah yang penuh fakta dan fantasi tersebut yang bakal menjadi penemuannya yang dituangkan kedalam mesin analitikal. Pada tahun 1843

presenter Lady Lovelace meramal bahwa kelak mesin semacam itu dapat digunakan untuk menyusun dan memainkan alat musik, dapat menghasilkan gambar ilmiah dan yang praktis digunakan. Hal tersebut terbukti tersedia pada saat ini.

Seiring berjalananya waktu, beberapa ilmuwan matematik menyarankan kepada Babbage dan Ada agar menyusun rencana agar mesinnya bisa menghitung angka Bernoulli. Dan rencana inilah, yang sekarang dianggap sebagai “Program Komputer” pertama, Babbage sebagai pembuat perangkat kerasnya dan Ada yang membuat dan menambahkan perangkat lunaknya. Pada tahun 1978, Departemen Pertahanan A.S. mengembangkan software pertamanya dan diberi nama “ADA” untuk menghormati Ada Byron.

### 3. *Programmer* dan Bahasa Pemrograman

*Programmer* adalah orang yang secara profesional bertanggung jawab atas perangkat lunak. Pada umumnya masyarakat berfikir bahwa semua *programmer* itu pada dasarnya melakukan hal dan pekerjaan yang sama, tapi sesungguhnya tidak demikian. Profesi *programmer* banyak memiliki spesialisasi yang terdefinisi dengan jelas, bahkan banyak bidang yang begitu berbeda satu dengan yang lain, sehingga tidak serupa sama sekali (kecuali persamaan orientasi, yaitu pada komputer).

Kedua kelompok terbesar adalah pemrograman sistem dan pemrograman aplikasi. Pemrograman sistem adalah aktivitasnya khusus pada pemrograman perangkat lunak sistem komputer (sistem operasi) dan jenis-jenis *system control program*. Pemrograman aplikasi adalah bertujuan menghasilkan perangkat lunak yang menyediakan layanan kepada pengguna secara langsung yang berjalan di atas sistem operasi.

Bahasa Pemrograman (*Programming Language*) adalah bahasa formal yang terdiri set instruksi untuk komputer yang menghasilkan keluaran. Bahasa Pemrograman digunakan dalam pemrograman komputer untuk mengimplementasikan algoritma. Bahasa pemrograman memungkinkan seseorang *programmer* dapat menentukan secara presisi data mana yang akan diolah oleh komputer. Hanya ada satu bahasa pemrograman yang benar-benar dapat dipahami dan dijalankan oleh komputer apapun, yaitu kode biner aslinya ‘0’ dan ‘1’.

Klasifikasi bahasa pemrograman tidaklah baku. Secara umum terdapat 3 (tiga) klasifikasi bahasa pemrograman, yaitu klasifikasi bahasa pemrograman tingkat rendah, tingkat menengah, dan tingkat tinggi.

- a. Bahasa pemrograman tingkat rendah atau biasa disebut dengan bahasa mesin, satu-satunya bahasa yang langsung diolah tanpa kompilasi terlebih dahulu. Bahasa pemrograman ini ditulis dengan kode-kode mesin.
- b. Bahasa pemrograman tingkat menengah atau biasa disebut dengan bahasa rakitan (*Assembly*), yaitu memberikan perintah untuk komputer dengan memakai kode-kode singkat (kode mnemonic), contohnya kode mesin: MOV, SUB, CMP, JMP, JGE, JL, LOOP. Contoh bahasa pemrograman ini adalah *Assembler*, *Microsoft Macro Assembler* (MASM).
- c. Bahasa pemrograman tingkat tinggi di awali kemunculannya pada pemrograman generasi ke-3 dan hingga generasi ke-5. Perkembangan bahasa pemrograman dari generasi ke generasi mengalami kemajuan pesat ditandai dengan bahasa sudah lebih banyak menggunakan *keyword* bahasa manusia, pemrograman berorientasi obyek, pemrograman berbasis web bahkan dengan sistem *cloud*, pemrograman berasis data dan bahkan yang lebih maju lagi adalah pemrograman *mobile*. Contoh bahasa pemrograman tingkat tinggi adalah Visual Basic, Delphi, Pascal, PHP, dan Java.

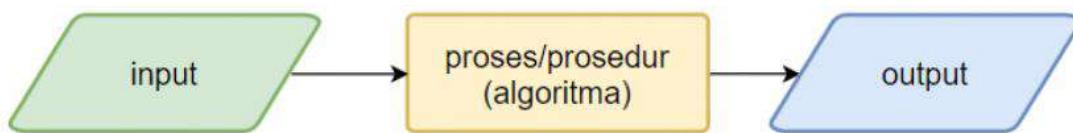
## D. PENYELESAIAN MASALAH DENGAN PROGRAM

Tiga konsep penting dalam penyelesaian masalah, yaitu: (1) Menganalisa masalah dan membuat algoritma. (2) Menuangkan algoritma ke dalam bentuk program. (3) Mengeksekusi dan menguji program.

**Penjelasan konsep di atas adalah sebagai berikut :**

1. Menganalisa masalah dan membuat algoritma

Hal yang pertama yang harus dilakukan untuk memecahkan permasalahan adalah menganalisa dan mengidentifikasi suatu permasalahan, mengidentifikasi data yang menjadi masukan/keluaran, kemudian membuat proses yang mengolah semua data yang masuk menjadi suatu keluaran yang diinginkan. Semua proses harus berisi intruksi yang jelas, urut dan runtut sampai permasalahan bisa diurai. Apabila permasalahan tersebut cukup kompleks dan terdiri dari beberapa proses; maka kegiatan tersebut dapat dipecah menjadi beberapa sub-proses. Urutan penyelesaian suatu masalah secara runtut ini dikenal sebagai algoritma.



Gambar 1.1  
Proses dalam Pemecahan Masalah (Algoritma)

## 2. Menuangkan algoritma dalam bentuk program

Konsep penyelesaian masalah, dalam bentuk urutan pemecahan masalah (algoritma) yang telah didesain, harus dapat dituangkan ke dalam bahasa program atau bahasa pemrograman oleh *programmer* dengan bahasa pemrograman yang dikuasainya.

## 3. Mengeksekusi dan menguji program

Program yang telah dibuat harus bisa dikompilasi menjadi suatu aplikasi untuk dapat di uji kebenarannya, dan apabila ditemukan bahwa telah terjadi kesalahan, maka proram tersebut harus diperbaiki sebelum diserahkan kepada pemakai.

## E. EKSPRESI ALGORITMA

Algoritma dapat diekspresikan dalam banyak notasi berbeda-beda, termasuk bahasa alami, *flowchart*, *pseudocode*, atau diagram alur, dan bahasa pemrograman. Ekspresi algoritma dengan bahasa alami cenderung bertele-tele dan ambigu, dan jarang digunakan untuk algoritma kompleks. Teknik/cara *flowchart* dan *pseudocode* merupakan cara terstruktur untuk mengekspresikan algoritma untuk menghindari ambiguitas pernyataan bahasa alami, dan tetap independen dari bahasa implementasi tertentu (tidak terikat dengan bahasa pemrograman tertentu).

Bahasa pemrograman digunakan untuk mengekspresikan algoritma dalam bentuk kode program yang dapat dijalankan komputer. Jika suatu komunitas pengembangan sistem informasi hanya menggunakan satu bahasa pemrograman, algoritma yang dibuatnya boleh dibuat mendekati bahasa pemrograman yang digunakan.

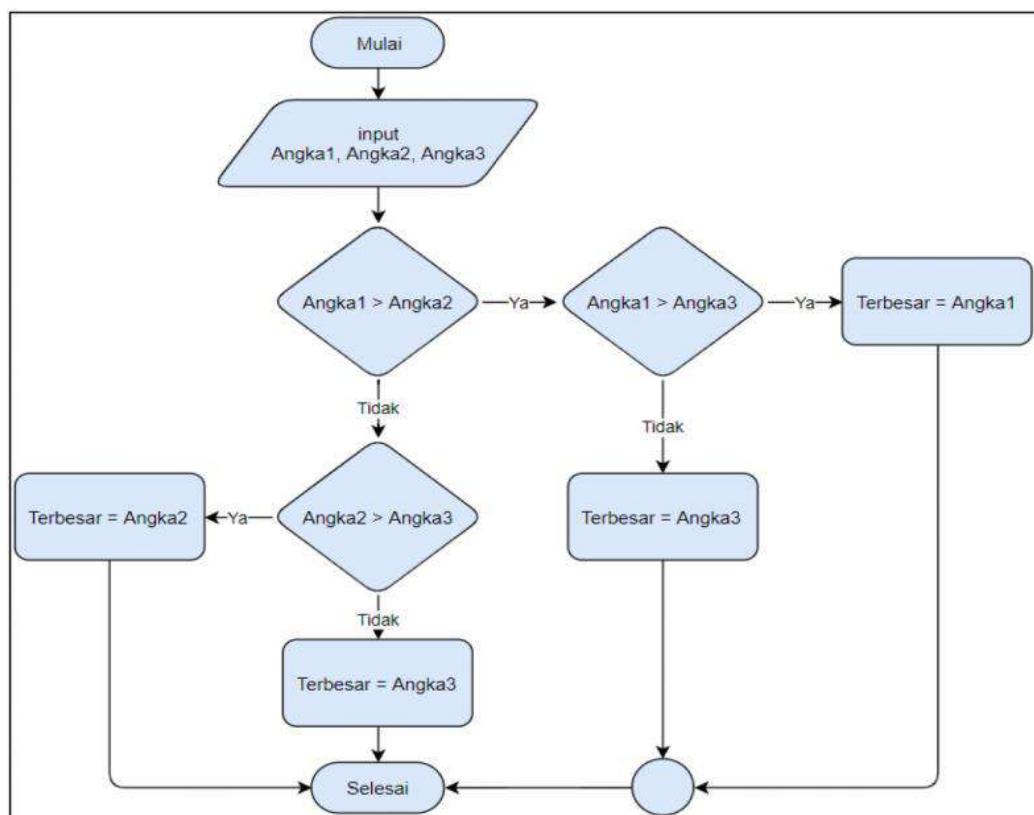
Algoritma yang digunakan dalam memecahkan masalah sangat bisa beragam karena sangat tergantung tingkat pemahaman dan logika pembuat algoritma. Perlu ditekankan bahwa algoritma yang baik adalah algoritma yang tidak banyak menggunakan sumber daya dan waktu.

Anda bisa membandingkan 2 contoh algoritma di bawah ini, yang keduanya mencari nilai terbesar pada 3 angka, yang satu dengan bahasa alami dan yang lainnya adalah menggunakan *pseudocode*.

Algoritma dengan bahasa alami:

1. Mulai
2. Masukkan *angka1*, *angka2*, dan *angka3*
3. Asumsikan terbesar adalah *angka1* untuk awal.
4. Jika *angka1* lebih besar dari terbesar maka terbesar adalah *angka1*
5. Jika tidak, apakah *angka2* lebih besar dari terbesar, Jika ya maka terbesar sama dengan *angka2*;
6. Jika tidak maka terbesar sama dengan *angka3*.
7. Selesai

Algoritma dengan *flowchart*:



Gambar 1.2  
Contoh Proses Dalam Pemecahan Masalah Algoritma (*Flowchart*)

Algoritma dengan *pseudocode*:

<b>Pseudocode Angka Terbesar (dalam bahasa Indonesia)</b>	
1	Mulai
2	Masukkan angka1
3	Masukkan angka2
4	Masukkan angka3
5	terbesar ← angka1
6	Jika angka1 > terbesar, maka
7	terbesar ← angka1
9	Jika angka2 > terbesar
10	terbesar ← angka2
11	Tapi jika tidak, maka
12	terbesar ← angka3
13	Angka terbesar ditemukan adalah terbesar
16	Selesai

Jika dibandingkan antara algoritma bahasa alami dengan *pseudocode*, maka lebih mudah memahami algoritma dengan *pseudocode*. Tapi untuk lebih praktisnya adalah dengan *pseudocode* karena dapat mengatur keterkaitan antar bloknya, serta mudah memahaminya. Contoh penggalan *pseudocode* berikut:

```

1 ...
2 Jika angka1 > terbesar, maka
3   terbesar <- angka1
4 Jika angka2 > terbesar, maka
5 ...

```

Baris yang menjorok kedalam *terbesar <- angka1* (baris 3) memberikan informasi bahwa baris ini akan dikerjakan jika memenuhi syarat pada baris di atasnya (baris 2), dan jika tidak memenuhi, maka akan melakukan perbandingan/menguji pada baris di bawahnya (baris 4), begitu seterusnya hingga perbandingan angka3. Jika pada umumnya bahasa pemrograman menggunakan Bahasa Inggris untuk *keyword*, maka dapat ditransformasikan sebagai berikut:

**PSMDL-2 : Pseudocode Angka Terbesar (dalam bahasa Inggris)**

```

1  Mulai
2  Input angka1
3  Input angka2
4  Input angka3
5  terbesar ← angka1
6  if angka1 > terbesar then
7      terbesar ← angka1
9  else if angka2 > terbesar then
10     terbesar ← angka2
11 Else
12     terbesar ← angka3
13 Print "Angka terbesar :" +Terbesar
16 Selesai

```

Harus diakui bahwa pendekatan dengan *keyword* berbahasa Inggris efektif jika acuan bahwa bahasa pemrograman pada umumnya menggunakan Bahasa Inggris. Jika algoritma di atas ditransformasi atau pendekatan kedalam bahasa pemrograman tertentu, misalnya bahasa pemrograman Java, maka hasilnya sebagai berikut:

**ProgramMDL-3: Mencari Angka/Nilai Terbesar (Java)**

```

1 //Nama Kelas (Program)
2 public class AngkaTerbesar {
3
4     public static void main(String[] arg) {
5         int angka1 = 10;
6         int angka2 = 15;
7         int angka3 = 5;
8         int terbesar = angka1;
9         if (angka1 > terbesar)
10            terbesar = angka1;

```

```
11     else if (angka2 > terbesar)
12         terbesar = angka2;
13     else
14         terbesar = angka3;
15
System.out.println("Bilangan terbesar adalah = " +
terbesar);
}
}
```

Mengapa hal ini bisa ditransformasi ke dalam bahasa pemrograman? Pada dasarnya analis dan *programmer* yang berpengalaman mempunyai naluri yang kuat membaca algoritma dengan pendekatan logika berpikir untuk menyelesaikan masalah.

## F. CIRI-CIRI DAN MANFAAT MEMPELAJARI ALGORITMA

Ciri-ciri dari algoritma adalah sebagai berikut:

1. Ada *input/masukan* dan *output/keluaran*.
2. Memiliki proses tertentu.
3. Prosesnya merupakan pola pikir dan logis dalam menghasilkan *output*.
4. Prosesnya memiliki intruksi yang jelas dan tidak ambiguitas
5. Memiliki *stopping role* atau jika pada keadaan tertentu mengalami proses iterasi yang berlebihan maka ada proses pemberhentian.

Adapun manfaat belajar algoritma adalah sebagai berikut :

1. Meningkatkan kemampuan berpikir secara logis. Logika dan algoritma pemrograman menjadi suatu hal yang sangat penting dalam membuat atau mengembangkan sebuah produk. Kesalahan logika yang digunakan, tentu akan berakibat fatal terhadap produk yang akan dikembangkan, selain *error*, tentu produk yang dikembangkan tidak akan sesuai dengan apa yang kita inginkan walaupun *programmer* telah berhasil membuat dalam program.
2. Mengembangkan cara berpikir dengan sistematis. Dalam membuat sebuah algoritma harus secara urut dan sistematis begitu juga dengan program hasil penerapan dari algoritma, seseorang akan dihadapkan pada

urutan-urutan yang disusun secara sistematis. Urutan-urutan harus terstruktur dan tidak boleh terbalik-balik baik penyusunannya maupun penulisannya, agar program yang dibangun dapat berjalan dengan baik dan benar.

3. Mempertajam analisis ketika pembuatan program. Ketika membuat algoritma maupun program terkadang muncul kesalahan-kesalahan dalam penyelesaiannya, misalnya program yang dibangun *error* saat diverifikasi atau *di-build*. Permasalahan ini akan memerlukan sedikit ketelitian untuk mengatasinya yaitu dengan melakukan pengecekan ulang kode program yang dibuat, pengecekan yang berulang-ulang akan membawa pada pelatihan menganalisis permasalahan dan meningkatkan ketelitian dalam membuat sebuah program dan membuat hasil program yang baik.
4. Meningkatkan kemampuan dalam mengatasi masalah. Tujuan utama algoritma adalah menyelesaikan masalah, jadi kita akan dilatih untuk menyelesaikan sebuah permasalahan, bahkan sampai pada memprediksi masalah yang akan muncul dan bagaimana mengelolanya. Secara tidak sadar, pola ini akan terbawa dalam kehidupan sehari-hari untuk menghadapi berbagai macam permasalahan yang terjadi. Kita secara tidak sadar akan berpikir secara logis dan sistematis.

## G. PEDOMAN PENYUSUNAN ALGORITMA

Tidak ada pedoman baku tentang teknik / cara baku untuk pembuatan dan penyusunan algoritma, namun diberikan syarat keterpenuhan. Menurut Ellis Horowitz dan Sartaj Sahni dalam bukunya berjudul “*Fundamentals of Data Structures*”, syarat ketercapaian suatu algoritma adalah apabila memenuhi syarat berikut:

1. *Input*: boleh nol atau lebih masukan dalam satu algoritma;
2. *Output*: dalam satu algoritma, dipersyaratkan memiliki satu keluaran, boleh lebih;
3. *Definiteness*: setiap intruksi harus jelas, tidak boleh ambigu (bermakna ganda atau lebih sehingga membingungkan);
4. *Finiteness*: menyatakan bahwa setelah melakukan proses maka apapun kondisinya suatu algoritma harus memiliki akhir;
5. *Effectiveness*: algoritma bekerja secara efektif, yaitu semua operasi yang dilakukan bersifat sederhana dan dapat diselesaikan dengan waktu yang singkat.



## LATIHAN

---

Untuk memperdalam pemahaman Anda mengenai materi di atas, kerjakanlah latihan berikut!

- 1) Apa yang dimaksud dengan algoritma?
- 2) Apa yang dimaksud dengan program?
- 3) Sebutkan dan jelaskan dua macam kelompok besar program komputer.
- 4) Apa yang dimaksud dengan bahasa pemrograman dan *programmer*?
- 5) Sebutkan dan jelaskan bahasa pemrograman berdasarkan fungsi kerja pada mesin komputer.
- 6) Sebutkan dan jelaskan tiga konsep penyelesaian masalah dengan program komputer.
- 7) Sebutkan dan jelaskan tiga alasan menggunakan algoritma?
- 8) Sebutkan manfaat dari menggunakan algoritma.

### *Petunjuk Jawaban Latihan*

Jawaban untuk latihan di atas dapat ditemukan pada teori yang diberikan pada Kegiatan Belajar 2 tentang Pengantar Algoritma dan Pemrograman Komputer. Jawaban dari soal-soal di atas adalah sebagai berikut:

- 1) Algoritma adalah prosedur pemecahan masalah dalam bahasa alami manusia yang tidak tergantung kepada bahasa pemrograman tertentu.
- 2) Program adalah satu set intruksi yang berkode yang dapat dimengerti oleh komputer untuk memecahkan masalah atau menghasilkan hasil yang diinginkan.
- 3) Kelompok besar program komputer adalah:
  - a. Sistem Operasi Komputer atau *Operating System* (OS): Program komputer yang menyediakan intruksi paling mendasar yang digunakan komputer dalam operasinya, OS merupakan perangkat lunak yang bisa mengelola perangkat keras, sumber daya perangkat lunak dan penyedia layanan untuk komputer lainnya. Contoh: Sistem Operasi Windows, Linux, MacOS.
  - b. Program Aplikasi: Aplikasi yang berjalan di atas sistem operasi dan melakukan pekerjaan sesuai tujuan kehendak kita, misalnya pengolah

kata, perhitungan, aplikasi pemutar video, aplikasi penampil gambar, aplikasi grafis.

- 4) Bahasa Pemrograman (*Programming Language*) adalah bahasa formal yang terdiri set intruksi untuk komputer yang menghasilkan keluaran. Bahasa Pemrograman digunakan dalam pemrograman komputer untuk mengimplementasikan algoritma. *Programmer* adalah orang yang profesional bertanggung jawab atas perangkat lunak. Profesi ini banyak memiliki spesialisasi yang terdefinisi dengan jelas, bahkan banyak bidang yang begitu berbeda satu dengan yang lain, sehingga tidak serupa sama sekali (kecuali persamaan orientasi, yaitu pada komputer).
- 5) Berdasarkan fungsi kerja pada mesin komputer bahasa pemrograman terdiri dari:
  - a. Bahasa Mesin: Instruksi komputer dengan memakai kode bahasa biner 0 dan 1.
  - b. Bahasa Tingkat Rendah: Bahasa Rakitan (*Assembly*), memberikan intruksi-intruksi singkat diluar bahasa manusia (bahasa mesin).
  - c. Bahasa Tingkat Menengah: Bahasa komputer yang memadukan bahasa manusia dengan bahasa simbolik.
  - d. Bahasa Tingkat Tinggi: Bahasa komputer dengan perintah bahasa manusia (bahasa Inggris).
- 6) Tiga konsep penyelesaian dengan program komputer yaitu:
  - a. Menganalisa (analisa) dan membuat algoritma: Tahapan analisa adalah mengenali, mengidentifikasi suatu masalah, mengidentifikasi adalah seberapa besar masalah yang ingin dipecahkan, jika permasalahan cukup besar, maka bisa dipecah menjadi sub-sub proses.
  - b. Menuangkan Algoritma kedalam bentuk program: Algoritma yang dibuat harus jelas prosesnya. Urutan antara proses atau sub proses harus sesuai, sehingga *programmer* dengan mudah menuangkan kedalam bentuk program.
  - c. Mengeksekusi dan Menguji Program: Program komputer yang telah dibuat harus di eksekusi dan diuji. Eksekusi adalah mengkompilasi kode-kode program yang sudah dibuat . Uji adalah menguji program apakah sudah sesuai dengan yang dinginkan atau sudah tercapai pemecahan yang dinginkan.

- 7) Tiga alasan kenapa menggunakan alasan menggunakan algoritma, sebagai berikut:
- Efisiensi: Untuk mengukur sebuah algoritma yang efisien harus mempertimbangkan yaitu efisiensi waktu-CPU dan memori. Terkadang *programmer* hanya berhenti kepada hasil tepat, akan tetapi tidak mempertimbangkan waktu dan memori yang terkuras oleh algoritma yang digunakan. Basis untuk membuat algoritma adalah efisiensi waktu, memori dan keluaran yang tepat.
  - Abstraksi: Kelebihan dari pada algoritma adalah mampu memperlihatkan sebuah permasalahan yang tingkat kerumitannya besar dapat diurai menjadi kelihatan mudah dan sederhana, gambaran kerumitan terkikis dengan alur algoritma yang tersusun baik dan jelas dengan pendekatan-pendekatan umum.
  - Reusability*: Algoritma adalah metode, bukan program itu sendiri, artinya bahwa algoritma harus mampu digunakan tanpa melihat bahasa pemrograman yang digunakan, dapat digunakan kembali dan bahkan berkali-kali pada pada berbagai situasi untuk menerapkan dalam bahasa program.
- 8) Adapun manfaat dari menggunakan algoritma adalah sebagai berikut:
- Meningkatkan kemampuan berfikir secara logis, logika dan algoritma pemrograman menjadi suatu hal yang sangat penting dalam membuat atau mengembangkan sebuah produk. Kesalahan logika yang digunakan, tentu akan berakibat fatal terhadap produk yang akan dikembangkan. Selain *error*, tentu produk yang dikembangkan tidak akan sesuai dengan apa yang kita inginkan.
  - Mengembangkan cara berfikir dengan sistematis. Dalam membuat sebuah algoritma harus secara urut dan sistematis begitu juga dengan program hasil penerapan dari algoritma, seseorang akan dihadapkan pada urutan-uruan yang disusun secara sistematis. Urutan-urutan harus terstruktur dan tidak boleh terbalik-balik baik penyusunannya maupun penulisannya, agar program yang dibangun dapat berjalan tanpa permasalahan.
  - Mempertajam analisis ketika pembuatan program, ketika membuat algoritma maupun program terkadang muncul kesalahan-kesalahan dalam penyelesaiannya, misalnya program yang dibangun *error* saat diverifikasi atau *di-build*. Permasalahan ini akan memerlukan sedikit ketelitian untuk mengatasinya yaitu dengan mengecek ulang kode

program yang dibuat, pengecekan yang berulang-ulang akan membawa pada pelatihan menganalisis permasalahan dan meningkatkan ketelitian dalam membuat sebuah program.

- d. Meningkatkan kemampuan dalam mengatasi masalah: Tujuan utama algoritma adalah menyelesaikan masalah, jadi kita akan dilatih untuk menyelesaikan sebuah permasalahan, bahkan sampai pada memprediksi masalah yang akan muncul dan bagaimana mengelolanya. secara tidak sadar, pola ini akan terbawa dalam kehidupan sehari-hari untuk menghadapi berbagai macam permasalahan yang terjadi. Kita secara tidak sadar akan berpikir secara logis dan sistematis.



## RANGKUMAN

---

Algoritma adalah suatu upaya dengan urutan operasi yang disusun secara logis dan sistematis untuk menyelesaikan suatu masalah untuk menghasilkan suatu *output* tertentu.

Dilihat dari disiplin ilmu maka berikut alasan mengadopsi atau menggunakan algoritma sebagai berikut:

1. Efisiensi: Untuk mengukur sebuah algoritma yang efisien harus mempertimbangkan yaitu efisiensi waktu-CPU dan memori. Terkadang *programmer* hanya berhenti kepada hasil tepat, akan tetapi tidak mempertimbangkan waktu dan memori yang terkuras oleh algoritma yang digunakan. Basis untuk membuat algoritma adalah efisiensi waktu, memori dan keluaran yang tepat. Walaupun tidak bisa dielakkan bahwa bahwa setiap orang akan memiliki cara menyelesaikan masalah yang berbeda-beda walau menghasilkan solusi yang sama. Dalam algoritma kecepatan dan ruang memori harus mampu diseimbangkan untuk menghasilkan solusi cepat dan tepat.
2. Abstraksi: Kelebihan dari pada algoritma adalah mampu memperlihatkan sebuah permasalahan yang tingkat kerumitannya besar dapat diurai menjadi kelihatan mudah dan sederhana, gambaran kerumitan terkikis dengan alur algoritma yang tersusun baik dan jelas dengan pendekatan-pendekatan umum.
3. Reusability: Algoritma adalah metode, bukan program itu sendiri, artinya bahwa algoritma harus mampu digunakan tanpa melihat bahasa pemrograman yang digunakan, dapat digunakan kembali dan bahkan berkali-kali pada berbagai situasi untuk menerapkan dalam bahasa program.

Program adalah satu set intruksi yang berkode yang dapat dimengerti oleh komputer untuk memecahkan masalah atau menghasilkan hasil yang diinginkan. Dua jenis program komputer adalah (1) Sistem Operasi, yang menyediakan intruksi paling mendasar yang digunakan komputer dalam operasinya, misalkan: Sistem Operasi Windows, Linux, MacOS dan (2) Program Aplikasi, yang berjalan pada sistem operasi dan melakukan pekerjaan seperti pengolah kata, perhitungan (olah data).

*Programmer* adalah orang yang profesional bertanggung jawab atas perangkat lunak. Profesi ini banyak memiliki spesialisasi yang terdefinisi dengan jelas, bahkan banyak bidang yang begitu berbeda satu dengan yang lain, sehingga tidak serupa sama sekali (kecuali persamaan orientasi, yaitu pada komputer).

Tiga konsep penting dalam penyelesaian masalah, yaitu: (1) Menganalisa masalah dan membuat algoritma, (2) Menuangkan algoritma kedalam bentuk program, (3) Mengeksekusi dan menguji program. Dan tiga alasan kenapa algoritma penting digunakan dalam menyelesaikan masalah yaitu: efisiensi, abstraksi dan *reusability*.

Manfaat mempelajari algoritma adalah sebagai berikut:

- Meningkatkan pengambilan keputusan dan kemampuan berfikir secara logis.
- Mengembangkan cara berfikir dengan sistematis.
- Mempertajam analisis ketika pembuatan program.
- Meningkatkan kemampuan dalam mengatasi masalah.

Syarat ketercapaian suatu algoritma adalah apabila memenuhi syarat berikut:

1. *Input*: boleh nol atau lebih masukan dalam satu algoritma;
2. *Output*: dalam satu algoritma, dipersyaratkan memiliki satu keluaran, boleh lebih;
3. *Definiteness*: setiap intruksi harus jelas, tidak boleh ambigu (bermakna ganda atau lebih sehingga membingungkan);
4. *Finiteness*: menyatakan bahwa setelah melakukan proses maka apapun kondisinya suatu algoritma harus memiliki akhir;
5. *Effectiveness*: algoritma bekerja secara efektif, yaitu semua operasi yang dilakukan bersifat sederhana dan dapat diselesaikan dengan waktu yang singkat.

**TES FORMATIF 1**

---

Pilihlah satu jawaban yang paling tepat!

- 1) Suatu upaya dengan urutan operasi yang disusun secara logis dan sistematis untuk menyelesaikan suatu masalah untuk menghasilkan suatu output tertentu definisi dari....
  - A. Logika pemrograman
  - B. Algoritma
  - C. Program komputer
  - D. Logika informatika
  
- 2) Algoritma adalah suatu metode khusus untuk menyelesaikan suatu masalah, definisi ini menurut....
  - A. Abu Ja'far Muhammad Ibnu Musa Al-Khawarizmi
  - B. Donald E. Knuth
  - C. David Bolton
  - D. Andrey Andreyevich Markov
  
- 3) Algoritma adalah hal umum untuk dipahami sebagai suatu keputusan yang tepat untuk mendefinisikan proses komputasi yang mengarahkan dari data awal hingga hasil yang diinginkan, definisi ini menurut....
  - A. Abu Ja'far Muhammad Ibnu Musa Al-Khawarizmi
  - B. Donald E. Knuth
  - C. David Bolton
  - D. Andrey Andreyevich Markov
  
- 4) Pengembang tidak mempertimbangkan waktu dan memori yang terkuras oleh algoritma yang digunakan. Narasi di atas bertentangan dengan alasan adopsi penggunaan algoritma pada poin....
  - A. Abstraksi
  - B. Efisiensi
  - C. *Reusability*
  - D. Semua benar
  
- 5) Bahasa Pemrograman yang pertama di dunia adalah....
  - A. Visual Basic
  - B. ADA
  - C. Java
  - D. Delphi

- 6) Program yang berjalan pada sistem operasi dan melakukan pekerjaan sesuai tujuan kehendak kita misal pengolah kata, perhitungan (olah data), presentasi video, suara dan sebagainya. Suatu program umumnya ditulis dengan menggunakan suatu bahasa pemrograman tingkat tinggi. Kalimat di atas mewakili dari definisi....
- Program Sistem Operasi
  - Bahasa pemrograman
  - Program aplikasi
  - Algoritma
- 7) Tiga alasan kenapa harus menggunakan algoritma, yaitu....
- Efisiensi, abstraksi, dan *reusability*
  - Efektif, hemat waktu, minim biaya
  - Fleksibel, efektif, dan normatif
  - Fleksibel, abstraksi, dan normatif
- 8) Mempertajam analisis ketika pembuatan program, adalah bagian dari....
- Manfaat mempelajari algoritma
  - Ciri sebuah algoritma
  - Ekspresi algoritma
  - Salah satu konsep penyelesaian masalah
- 9) Mengidentifikasi data yang menjadi masukan/keluaran, kemudian membuat proses yang mengolah semua data yang masuk menjadi suatu keluaran yang diinginkan. Semua proses harus berisi intruksi yang jelas, urut dan runtut sampai permasalahan bisa diurai, narasi di atas lebih tepat pada penyelesaian masalah pada bagian....
- Analisa dan membuat algoritma
  - Menuangkan algoritma dalam bentuk program
  - Mengeksekusi dan menguji program
  - Semua benar
- 10) *Effectiveness*, artinya....
- Boleh nol atau lebih masukan dalam satu algoritma;  
Output: dalam satu algoritma, dipersyaratkan memiliki satu keluaran, boleh lebih
  - Setiap intruksi harus jelas, tidak boleh ambigu (bermakna ganda atau lebih sehingga membingungkan)
  - Setelah melakukan proses maka apapun kondisinya suatu algoritma harus memiliki akhir
  - algoritma bekerja secara efektif, yaitu semua operasi yang dilakukan bersifat sederhana dan dapat diselesaikan dengan waktu yang singkat

Cocokkanlah jawaban Anda dengan Kunci Jawaban Tes Formatif 1 yang terdapat di bagian akhir modul ini. Hitunglah jawaban yang benar. Kemudian, gunakan rumus berikut untuk mengetahui tingkat penguasaan Anda terhadap materi Kegiatan Belajar 1.

$$\text{Tingkat penguasaan} = \frac{\text{Jumlah Jawaban yang Benar}}{\text{Jumlah Soal}} \times 100\%$$

Arti tingkat penguasaan: 90 - 100% = baik sekali

80 - 89% = baik

70 - 79% = cukup

< 70% = kurang

Apabila mencapai tingkat penguasaan 80% atau lebih, Anda dapat meneruskan dengan Kegiatan Belajar 2. **Bagus!** Jika masih di bawah 80%, Anda harus mengulangi materi Kegiatan Belajar 1, terutama bagian yang belum dikuasai.

**KEGIATAN BELAJAR 2****Struktur Dasar Algoritma**

Ketika ingin membangun bangunan rumah hal yang paling pertama yang harus dipikirkan adalah kontruksi dasar, seperti pondasi, balok beton dan konstruksi dinding, hal sama juga dengan membangun sebuah algoritma harus tahu struktur dasarnya. Ada tiga struktur dasar dalam algoritma yaitu: skuesial (*sequential*), seleksi (*selection*), dan perulangan (*looping*).

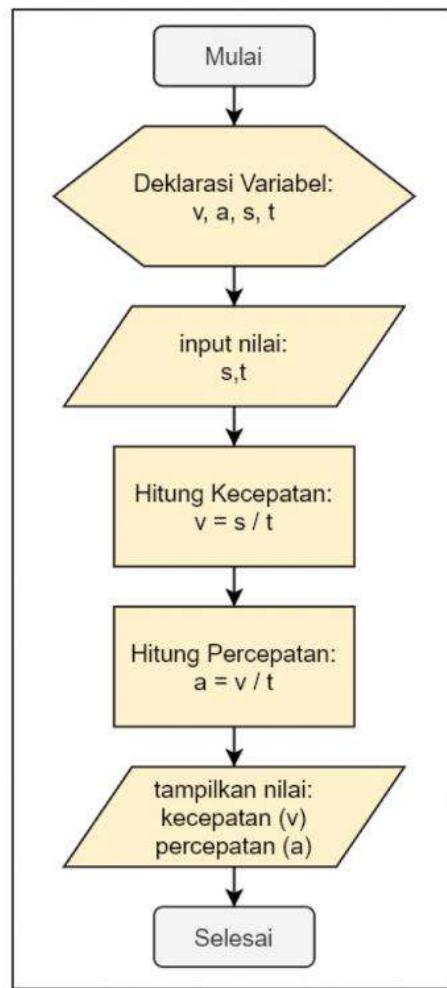
**A. SKUENSIAL (*SEQUENTIAL*)**

Struktur dasar skuensial adalah sebuah algoritma dibangun dengan langkah-langkah (instruksi/perintah) dikerjakan secara berurutan, tidak boleh melompati satu langkah perintah pun. Misal dalam sebuah algoritma terdapat 20 langkah, maka semua langkah tersebut dikerjakan berurutan mulai dari langkah 1 sampai pada langkah 20 tanpa melewatkannya satu langkah pun.

Cara kerja skuensial juga nantinya berlaku dalam bahasa pemrograman, ketika instruksi bahasa pemrograman yang kita tulis diproses oleh komputer, maka komputer akan memproses dan menterjemahkan baris demi baris intruksi-instruksi bahasa pemrograman tersebut secara beruntun dari awal hingga akhir dimulai dari instruksi pada baris awal hingga baris akhir.

Dengan struktur skuensial ini, pembuat algoritma harus mampu menganalisa dan menentukan intruksi mana yang harus ditulis lebih awal dan seterusnya dan yang mana harus paling akhir.

Pada Gambar 1.3 adalah contoh algoritma dengan skuensial untuk mencari kecepatan dan percepatan, tidak ada satupun proses yang terlewatkan atau melompati proses yang ada di bawahnya, semua dikerjakan secara berurutan. Demikian juga dengan intruksi yang ada di dalam prosesnya, misalnya rumus hitung *pecepatan* (*a*) tidak mendahului proses perhitungan *kecepatan* (*v*). hal ini karena untuk mencari percepatan harus mencari atau menghitung kecepatan terlebih dahulu.



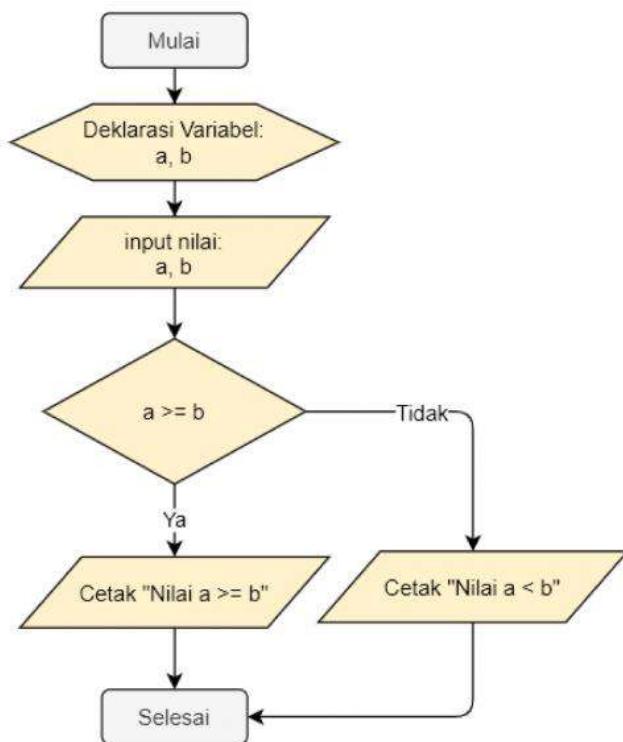
Gambar 1.3  
Algoritma dengan Cara Skuensial Mencari Kecepatan dan Percepatan

Jika struktur dasar skuensial direpresentasikan menjadi algoritma *pseudocode*, urutan pekerjaannya akan sama. Perhatikan *pseudocode* mencari kecepatan dan percepatan di bawah ini:

<b><i>Pseudocode cari kecepatan dan percepatan</i></b>	
<b>1</b>	Mulai
<b>2</b>	Deklarasi variabel: v, a, s, t
<b>2</b>	Input/masukkan nilai: s, a
<b>3</b>	$v \leftarrow s/t$
<b>4</b>	$a \leftarrow v/t$
<b>5</b>	Tampilkan nilai kecepatan: v
<b>6</b>	Tampilkan nilai percepatan: a
<b>7</b>	Selesai

## B. SELEKSI (SELECTION)

Pada kenyataannya banyak algoritma setidaknya akan mengandung proses seleksi (*selection*) pada intruksi-intruksi pada tubuh algoritma, instruksi seleksi digunakan apabila menemukan/memiliki kasus dua atau lebih alternatif penyelesaian/keputusan.



Gambar 1.4  
Flowchart dengan Struktrur Algoritma dengan Seleksi

Gambar 1.4 adalah contoh algoritma penyelesaian dengan seleksi dua keputusan. Ada dua angka yang akan diseleksi dengan membandingkan keduanya mana yang terbesar/sama dengan atau terkecil. Misal nilai  $a = 6$  dan nilai  $b = 5$ , maka seleksinya adalah akan tercetak “ $\text{Nilai } a \geq b$ ” dan jika sebaliknya maka keputusan yang lain adalah tercetak “ $\text{Nilai } a < b$ ”.

<b><i>Pseudocode menentukan nilai terbesar</i></b>	
1	Mulai
2	Deklarasi variabel: a, b
3	Input/masukkan nilai: a, b
4	if a >= b

5	Cetak "Nilai a >= b"
6	ke langkah 9
7	else //lainnya
8	Cetak "Nilai a < b"
9	Selesai

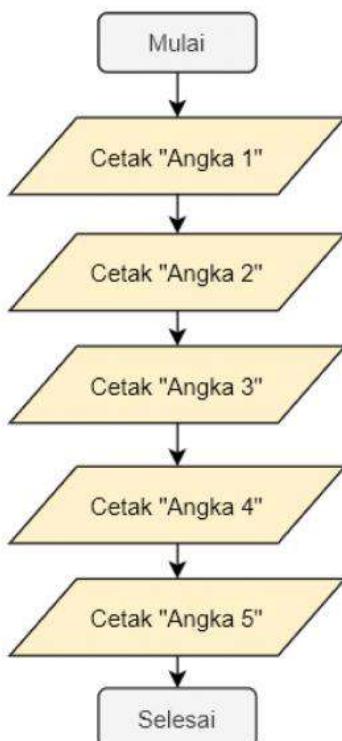
Dalam dunia algoritma dan pemrograman akan sering diperhadapkan kondisi seleksi seperti contoh di atas, baik tipe kasusnya sama maupun berbeda.

### C. PERULANGAN (*LOOPING*)

Struktur algoritma yang ketiga adalah perulangan. Banyak hal terjadi di dunia ini berulang-ulang, kita bisa menghafal sesuatu salah satu cara adalah mengulang, materi yang diberikan dikelas oleh guru atau dosen mugkin tidak serta merta langsung bisa dimengerti, akan tetapi melakukan pengulangan materi yang pernah diberikan sebelumnya bisa menjadi efektif untuk memahaminya/menghafalnya.

Perulangan dalam algoritma sangat dibutuhkan untuk menyelesaikan banyak masalah. Pengurutan data yang banyak dalam program tertentu itu karena andil sebuah algoritma perulangan. Bayangkan jika harus menuliskan angka 1 sampai 100 dengan manual dalam sebuah program, akan memakan waktu, akan tetapi dengan menggunakan algoritma perulangan dalam progam hanya terdiri dari 4 langkah intruksi dalam algoritma sudah bisa menyelesaikan/menampilkan angka 1 sampai 100 bahkan lebih. Anda pernah menggunakan aplikasi/teknologi kecerdasan buatan pengenalan wajah atau sidik jari dari smartphone anda? Yakinlah bahwa didalamnya terdapat banyak perulangan.

Di atas sudah saya sebutkan bahwa algoritma sendiri untuk mengatasi kasus pengulangan data, memiliki intruksi tersendiri, dengan intruksi tersebut pengulangan akan lebih mudah ditulis secara singkat dan praktis daripada harus di tulis satu-persatu. Akan saya berikan satu contoh *flowchart* untuk menuliskan angka 1 sampai dengan angka 5, sebagai berikut:

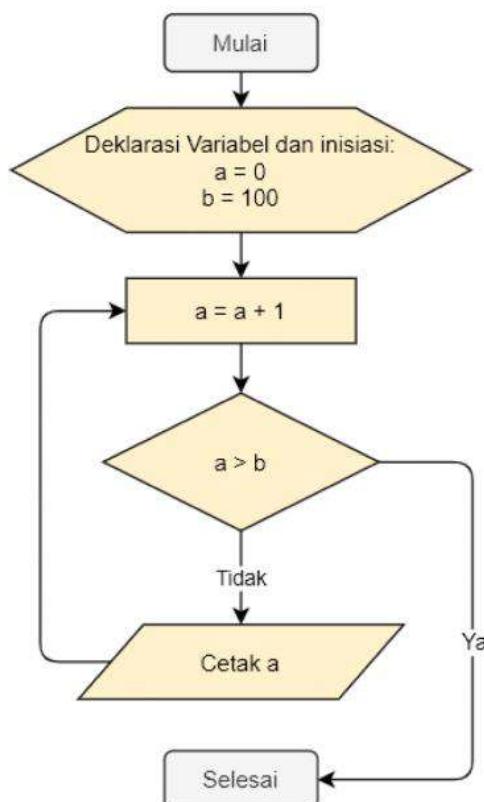


Gambar 1.5

*Flowchart dengan Struktur Algoritma Skuesial untuk Cetak Angka 1 Sampai 5*

Gambar 1.5 secara sekilas tampak tidak ada masalah, kenapa? Sebab angka yang dicetak masih sedikit, tapi coba dibayangkan jika harus mencetak angka 1 sampai angka 100 dengan mengikuti pola pada Gambar 1.5, tentu cara seperti ini tidak efektif, kenapa? Berapa anda harus membuat 102 simbol (termasuk simbol *mulai* dan *selesai*) dalam *flowchart* hanya untuk mencetak angka 1 sampai 100. Tentu *flowchart* di atas sangat tidak efektif.

Lalu apakah ada cara yang efektif? Cara yang efektif adalah dengan algoritma perulangan. Di bawah disajikan contoh *flowchart* mencetak angka 1 sampai 100 dengan kolaborasi antara algoritma perulangan dengan algoritma seleksi:



Gambar 1.6

*Flowchart dengan Algoritma untuk Perulangan dan Seleksi Cetak Angka 1 Sampai 100*

Sekarang bandingkan Gambar 1.5 dan Gambar 1.6. Gambar 1.5 hanya bisa mencetak angka 1 sampai 5, jika ingin cetak angka lebih dari 4 maka wajib menambahkan simbol *flowchart* baru. Sedangkan untuk Gambar 1.6 bisa mencetak angka 1 sampai angka 100, lalu bagaimana kalau kita ingin mencetak angka sampai 1000 untuk Gambar 1.6? jawabannya adalah tidak perlu menambah simbol *flowchart* baru, akan tetapi cukup mengubah  $b = 1000$ , cukup efektifkan?

#### Pseudocode cetak angka 1 - 100

1	Mulai
2	Deklarasi variabel dan inisiasi : $a=0$ , $b=100$
3	$a \leftarrow a + 1$
4	if $a > b$
5	Cetak a
6	ke langkah 3

7    else //lainnya

8    Selesai

Bagaimana proses perulangan berjalan pada *flowchart* Gambar 1.6 atau *pseudocode* perulangan? Berikut penjelasannya. Tahap awal variabel *a* diisi 1 dan variabel *b* diisi 100, (Perulangan Pertama) kemudian variabel *a* diisi dengan  $a = a + 1$ , sehingga  $a = 1$ , karena  $a = a + 1$  sama dengan  $a = 0 + 1$ . Langkah berikutnya *a* diuji dengan *b*, apakah  $a > b$  atau  $1 > 100$ , tentu jawabannya salah atau “Tidak”, karena hasil seleksi adalah “Tidak” maka variabel *a* dicetak. (Perulangan Kedua) Dan kemudian kembali ke  $a = a + 1$  atau  $a = 1 + 1$  sehingga  $a = 2$ , proses ini akan berulang dan mencetak angka 100 hingga sampai memenuhi kondisi  $a > b$  benar atau “Ya” dan algoritma perulangan selesai.



## LATIHAN

---

Untuk memperdalam pemahaman Anda mengenai materi di atas, kerjakanlah latihan berikut!

- 1) Sebutkan dan jelaskan 3 struktur dasar pada algoritma?

### Petunjuk Jawaban Latihan

- 1) Tiga struktur dasar algoritma adalah:
  - Skuensial: Struktur dasar algoritma yang mengerjakan semua proses perintah dalam sebuah.
  - Seleksi: Algoritma yang mengandung lebih dari satu kondisi pilihan dalam satu algoritma.
  - Perulangan: Algoritma yang mengandung perulangan-perulangan dalam proses intruksinya.



## RANGKUMAN

---

Ada tiga struktur dasar algoritma, yaitu: skuensial (*sequential*), seleksi (*selection*), dan perulangan (*looping*). Struktur dasar skuensial adalah sebuah algoritma dibangun dengan langkah-langkah (instruksi/perintah) dikerjakan secara berurutan, tidak boleh melompati satu langkah perintah pun. Struktur dasar algoritma seleksi adalah digunakan apabila dalam tubuh algoritma menemukan kasus dengan 2 atau lebih alternatif penyelesaian. Struktur dasar algoritma yang terakhir adalah perulangan, banyak hal dalam dunia algoritma bisa diselesaikan dengan perulangan.



## TES FORMATIF 2

---

Pilihlah satu jawaban yang paling tepat!

- 1) Struktur dasar algoritma yang menyelesaikan semua langkah dari setiap proses dalam algoritma adalah....
  - A. Perulangan
  - B. Skuensial
  - C. Seleksi
  - D. Semuanya benar
- 2) Sebuah algoritma yang mengandung 2 atau lebih alternatif solusi termasuk dalam struktur dasar....
  - A. Seleksi
  - B. Skuensial
  - C. Perulangan
  - D. Skuensial dan seleksi
- 3) Jika kita ingin mencetak angka 1 sampai 100, lebih cocok menggunakan struktur dasar algoritma....
  - A. Skuensial
  - B. Seleksi
  - C. Skuensial dan Seleksi
  - D. Perulangan

- 4) Algoritma yang terdapat didalamnya ada yang dibandingkan dan ada pembanding, struktur algoritma tersebut disebut dengan....
- Seleksi
  - Perulangan
  - Skuensial
  - Tidak ada yang benar
- 5) Jika sebuah algoritma mengandung seleksi maka bisa dipastikan bahwa algoritma tersebut tidak mengandung struktur dasar....
- Skuensial dan Seleksi
  - Seleksi
  - Perulangan
  - Skuensial
- 6) Struktur dasar yang bisa digabung dan saling mengisi adalah struktur dasar algoritma....
- Seleksi dan Perulangan
  - Seleksi dan skuensial
  - Skuensial dan Perulangan
  - Tidak ada yang benar
- 7) Jika ada struktur dasar algoritma skuensial, bisa dipastikan bahwa struktur dasar algoritma yang tidak ada adalah...
- Perulangan dan seleksi
  - Perulangan
  - Seleksi
  - Tidak ada yang benar
- 8) Jika sebuah variabe  $c = 5$ , kemudian dalam sebuah perulangan dibuat  $c = c + 1$ , dan dilakukan perulangan selama 5 kali, berapakah nilai  $c$  di akhir perulangan?
- 9
  - 10
  - 11
  - 12
- 9) Jika sebuah variabe  $c = 5$ , kemudian dalam sebuah perulangan dibuat kondisi jika  $c = 7$  maka operasi  $c = c + 2$ , jika tidak operasi  $c = c + 1$ , perulangan dilakukan perulangan selama 5 kali, berapakah nilai  $c$  di akhir perulangan?

- A. 9  
B. 10  
C. 11  
D. 12
- 10) Jika sebuah variabel  $k = 10$ , kemudian dalam sebuah perulangan dibuat kondisi jika  $k = 11$  maka operasi  $k = k + 1$ , jika tidak maka operasi  $k = k + 2$ , perulangan dilakukan perulangan selama 6 kali, berapakah nilai  $k$  di akhir perulangan?  
A. 17  
B. 18  
C. 19  
D. 20

Cocokkanlah jawaban Anda dengan Kunci Jawaban Tes Formatif 2 yang terdapat di bagian akhir modul ini. Hitunglah jawaban yang benar. Kemudian, gunakan rumus berikut untuk mengetahui tingkat penguasaan Anda terhadap materi Kegiatan Belajar 2.

$$\text{Tingkat penguasaan} = \frac{\text{Jumlah Jawaban yang Benar}}{\text{Jumlah Soal}} \times 100\%$$

Arti tingkat penguasaan:  $90 - 100\% =$  baik sekali

$80 - 89\% =$  baik

$70 - 79\% =$  cukup

$< 70\% =$  kurang

Apabila mencapai tingkat penguasaan 80% atau lebih, Anda dapat meneruskan dengan Modul selanjutnya. **Bagus!** Jika masih di bawah 80%, Anda harus mengulangi materi Kegiatan Belajar 2, terutama bagian yang belum dikuasai.

## Kunci Jawaban Tes Formatif

### Tes Formatif 1

- 1) B
- 2) A
- 3) D
- 4) B
- 5) B
- 6) C
- 7) A
- 8) A
- 9) A
- 10) D

### Tes Formatif 2

- 1) B
- 2) A
- 3) D
- 4) A
- 5) D
- 6) C
- 7) C
- 8) B
- 9) C
- 10) D

## Glosarium

- CPU (Central Processing Unit) : Perangkat keras komputer yang memiliki tugas untuk menerima dan melaksanakan perintah dan data dari perangkat lunak. Karena merupakan pusat pengolahan data dalam sebuah komputer, CPU sering disebut juga sebagai *processor*
- GPS : Sistem untuk menentukan letak di permukaan bumi dengan bantuan penyelarasan (*synchronization*) sinyal satelit. Sistem ini menggunakan 24 satelit yang mengirimkan sinyal gelombang mikro ke Bumi. Sinyal ini diterima oleh alat penerima di permukaan, dan digunakan untuk menentukan letak, kecepatan, arah, dan waktu.
- Kecerdasan Buatan : Kecerdasan yang ditambahkan kepada suatu sistem yang bisa diatur dalam konteks ilmiah atau bisa disebut juga intelegensi artifisial (*Artificial Intelligence*) disingkat AI, didefinisikan sebagai kecerdasan *entitas ilmiah*. Andreas Kaplan dan Michael Haenlein mendefinisikan kecerdasan buatan sebagai “kemampuan sistem untuk menafsirkan data eksternal dengan benar, untuk belajar dari data tersebut, dan menggunakan pembelajaran tersebut guna mencapai tujuan dan tugas tertentu melalui adaptasi yang fleksibel”.
- Mnemonik : Teknik yang memudahkan penyimpanan, atau penyandian dan pengingat terhadap informasi dalam memori.
- Pseudocode : Deskripsi tingkat tinggi informal dan ringkas atas algoritme pemrograman komputer yang menggunakan konvensi struktural atas suatu bahasa pemrograman, dan ditujukan untuk dibaca oleh manusia dan bukan oleh mesin.

## Daftar Pustaka

- Christodoulou, M. & Szczygiel, E. (2018). *Algoritmic and programming*. P.T.E.A. Wszechnica Sp. Z.o.o.
- Erickson, J. (2019). *Algoritms*. xxx.
- Hermin, F. & Widjati, R. (2004). *Komputer II*. Penerbit Universitas Terbuka.
- Horwits, E. & Sahni S. (1993). *Fundamental of data structure in C++, source DBLP*.
- Laaksonen, A. (2018). *Competitive programmer's handbook*.
- Nugroho, E. (2010). *Pengantar aplikasi komputer*. Penerbit Universitas Terbuka.
- Rinaldi, M. (1999). *Algoritma dan pemrograman Jilid1*. Penerbit IPB.
- Wimatra, A. dkk. (2008). *Dasar-dasar komputer*. Civil Aviation Safety and Technics Academy. Medan.
- Purbasari, I. Y., “*Desain & analisis algoritma*”. Yogjakarta: Graha Ilmu

# Representasi Algoritma

Kani, M.Kom.



## PENDAHULUAN

---

**A**nda bayangkan! Jika pada saat ini tidak ada komputer dan semua diketik manual menggunakan mesin ketik. Bayangkan bagaimana mengetik surat kabar sebelum masuknya teknologi komputer di Indonesia! Bayangkan Microsoft Excel yang tidak punya fasilitas sortir data, maka kita akan membutuhkan waktu yang lama untuk mengurutkan 100 data! Sungguh sebuah sumbangsih yang besar dengan adanya teknologi komputer dan algoritma-algoritma yang menyertainya.

*Software* komputer terdiri dari algoritma-algoritma yang mampu memudahkan pekerjaan manusia. Algoritma-algoritma yang efektif dapat menghemat waktu kerja, contoh: kita dapat mengurutkan 100 atau 1000 data dalam hitungan detik saja. Microsoft Excel, Microsoft Word, Microsoft PowerPoint, Matlab, SPSS, Bahasa R, dan yang lainnya, itu adalah *software-software* yang di dalamnya terdapat ratusan bahkan ribuan algoritma untuk menyelesaikan permasalahan tertentu. Dan beberapa *software* yang sangat berguna bagi manusia dan ilmu pengetahuan justru digratiskan oleh pembuatnya (pengembangnya), dan merupakan sumbangsih yang luar biasa.

Hampir semua *software* yang dikembangkan memiliki desain dan cetak biru (*blueprint*), dibuat mulai dari kerangka yang sangat besar, lalu kemudian menjadi sub-sub bagian, dari setiap sub-sub bagian yang mempunyai fungsi masing-masing. Algoritma-algoritma yang tertanam dalam *software* sebagian besar direpresentasikan dalam bentuk *flowchart* atau *pseudocode*. Tujuannya adalah agar memudahkan *programmer* dalam membuat programnya. Sebuah permasalahan yang rumit menjadi tampak sederhana jika sudah direpresentasikan dalam bentuk *flowchart* atau *pseudocode*. *Programmer* tidak perlu memikirkan bagaimana prosesnya sebuah pemecahan permasalahan karena itu adalah urusan analis, akan tetapi *programmer* berpikir bagaimana menjadikan sebuah algoritma menjadi bentuk program komputer dengan keluaran sesuai keinginan.

Setelah Anda mempelajari modul ini dengan baik, Anda akan dapat/mampu :

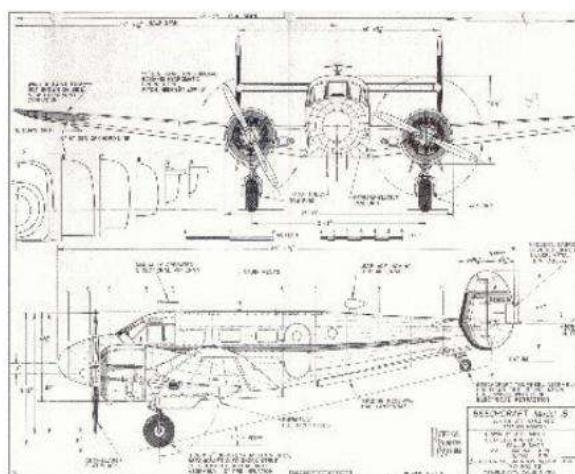
1. merepresentasikan sebuah algoritma dalam bentuk *flowchart* dan *pseudocode*;
2. membuat *flowchart* dan *pseudocode* dengan menggunakan bahasa sehari-hari yang sederhana dan mudah dimengerti orang awam sekalipun;
3. menganalisa permasalahan dan membuat *flowchart* dan *pseudocode* pada tingkat ideal;
4. mengenali simbol-simbol *flowchart* yang telah umum digunakan;
5. menyusun struktur *pseudocode* yang tepat dan memberi indentasi jika terdapat sub-proses;
6. menyusun *pseudocode* dengan logis, representatif, dan ideal;
7. membuat *flowchart* dengan aturan yang ada;
8. mempraktekkan semua contoh-contoh kasus.

## KEGIATAN BELAJAR 1

### *Flowchart*

*F*lowchart adalah jenis diagram (grafis atau simbolik) yang mewakili suatu algoritma atau proses-proses tertentu. Setiap langkah dalam algoritma diwakili oleh simbol yang sama atau berbeda dan berisi penjelasan singkat setiap langkah. Simbol *flowchart* dihubungkan dengan garis dan panah yang menunjukkan arah aliran proses. Suatu diagram alur biasanya menunjukkan aliran data dalam suatu proses, merinci operasi/langkah dalam format bergambar agar lebih mudah dipahami dibanding dengan membacanya dalam format tekstual.

*Flowchart* menggambarkan bentuk operasi dalam urutan proses yang diperlukan untuk memecahkan masalah yang diberikan. *Flowchart* dapat disamakan dengan cetak biru (*blueprint*) bangunan atau cetak biru pesawat. Seperti kita ketahui, seorang desainer pesawat menggambar cetak biru sebelum memulai pembuatan sebuah pesawat. Demikian pula seorang analis atau seorang pembuat algoritma lebih suka menggambar *flowchart* untuk memahamkan kepada orang awam atau kepada *programmer* sebelum *programmer* menuliskannya kedalam program. *Flowchart* adalah representasi gambar atau grafis dari suatu proses. Tujuan dari semua *flowchart* adalah untuk mengkomunikasikan bagaimana suatu proses bekerja tanpa peristilahan teknis atau peristilahan yang hanya dipahami suatu kelompok tertentu.



Gambar 2.1.  
*Blue Print* Pesawat

*Flowchart* merepresentasikan algoritma dalam bentuk desain, simbol dan dijadikan dokumentasi dan kemudian dituangkan menjadi kode-kode program. *Flowchart* umumnya digambar pada tahap awal pengembangan program sehingga dapat menjadi suatu gambaran aliran fungsi kerja suatu program komputer. Hadirnya *flowchart* dapat membantu *programmer* untuk memahami permasalahan yang ada dan target yang dikehendaki secara logika, walaupun suatu masalah cukup rumit dan kompleks, tapi setelah *flowchart* dibuat, *programmer* tidak kesusahan menuliskannya ke dalam program dalam bahasa pemrograman apa pun. Teks program yang dibuat oleh orang lain tanpa ada dokumentasi dalam bentuk *flowchart* atau apapun namanya, kadang sulit untuk memahami alur prosesnya. Sebaliknya, menerangkan atau memahami sebuah *flowchart* jauh lebih mudah walau sesulit apapun permasalahannya.

Di bawah ini diberikan contoh algoritma mencari kecepatan dan percepatan. Kita mengetahui bersama bahwa rumus sebuah kecepatan:

$$v = \frac{s}{t}$$

di mana:

$v$  = kecepatan

$s$  = waktu

$t$  = jarak tempuh

Sedangkan untuk percepatan (a), dengan formulasi sebagai berikut:

$$a = \frac{v}{t}$$

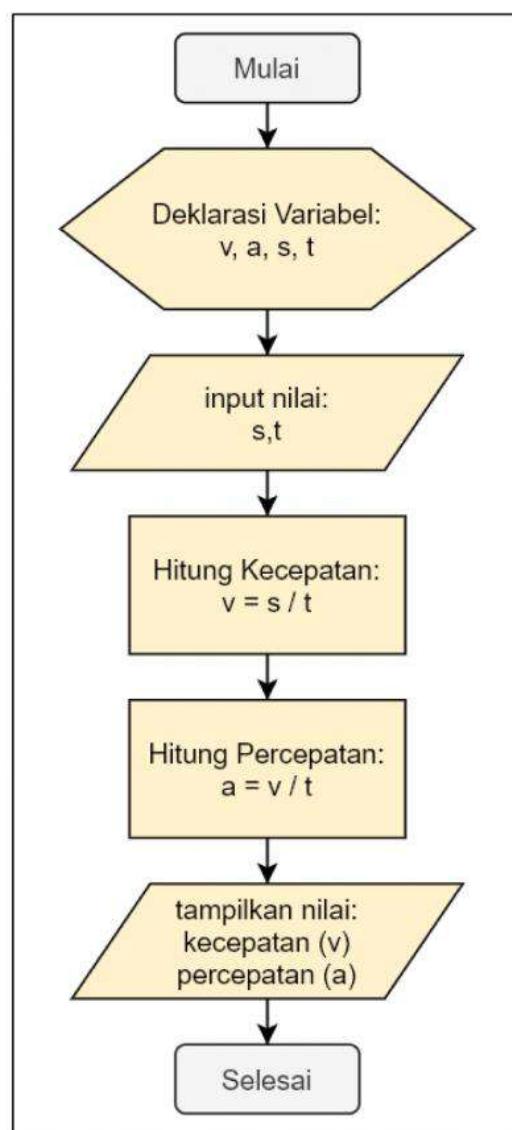
di mana:

$a$  = percepatan;

$v$  = kecepatan;

$t$  = waktu.

*Flowchart* untuk mencari kecepatan dengan langkah demi langkah dengan simbol-simbol sebagai berikut:



Gambar 2.2  
Contoh Flowchart

Tidak semua proses harus dibuat sebagai simbol dalam suatu *flowchart*. Beberapa proses dapat digabung apabila memiliki proses yang sama, misalkan pada proses menampilkan nilai pada *flowchart* di atas, beberapa langkah digabung menjadi satu yakni tampilkan nilai kecepatan dan percepatan digabung menjadi satu demikian juga dengan simbol digabung antara tampilkan *Kecepatan (v)* dan *Percepatan (a)*, tapi ketika *flowchart* tersebut dibuat dalam kode-kode program pada bahasa pemrograman tertentu, semua perintah akan dikerjakan satu demi satu.

## A. KEUNTUNGAN MENGGUNAKAN *FLOWCHART*

Ada pun keuntungan menggunakan sistem *flowchart*, sebagai berikut:

1. *Komunikasi*: *Flowchart* adalah cara yang lebih baik untuk mengkomunikasikan logika suatu sistem kepada semua pihak yang terkait.
2. *Analisis yang efektif*: Dengan bantuan *flowchart*, masalah dapat dianalisis dengan cara yang lebih efektif.
3. *Dokumentasi*: Program *flowchart* berfungsi sebagai dokumentasi program yang baik, yang diperlukan untuk berbagai tujuan.
4. *Pengkodean Efisien*: *Flowchart* bertindak sebagai panduan atau cetak biru (*blueprint*) selama analisis sistem dan fase pengembangan program.
5. *Proper Debugging*: Diagram alur membantu dalam proses *debugging* dengan cepat, karena dari awal kita sudah mengetahui secara detil permasalahan dan apa yang dikerjakan.
6. *Pemeliharaan Program yang Efisien*: Pemeliharaan program operasi menjadi mudah dengan bantuan *flowchart*. Ini membantu *programmer* untuk menempatkan upaya lebih efisien pada bagian itu.

## B. KETERBATASAN MENGGUNAKAN *FLOWCHART*

Di samping keuntungan-keuntungan menggunakan *flowchart*, ternyata penggunaan *flowchart* mempunyai keterbatasan-keterbatasan tersendiri. Beberapa keterbatasan penggunaan *flowchart* di bawah ini:

1. *Logika yang kompleks (rumit)*: Ada kalanya suatu logika program cukup rumit untuk di ekspresikan secara visual, sehingga menyajikan proses tersebut ke dalam suatu *flowchart* merupakan kendala baru (tugas tambahan lain yang berat). Beberapa pemrogram lebih memilih untuk langsung menulis programnya setelah berkomunikasi dengan pengguna, dibandingkan jika harus menunggu representasi dari analis.
2. *Perubahan/Modifikasi*: Jika diperlukan perubahan, *flowchart* mungkin memerlukan perubahan total pada desainnya (simbol-simbol *flowchart*).
3. *Reproduksi*: Karena *flowchart* berupa simbol (grafik), tidak bisa diketik, sehingga menjadi permasalahan tersendiri.

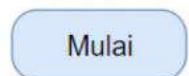
### C. SIMBOL-SIMBOL *FLOWCHART*

Berikut ini adalah beberapa unsur dalam *flowchart* tersebut :

- *Input*
- Percabangan (biasanya menggunakan perintah *if* dan *switch*)
- Perulangan (biasanya menggunakan perintah atau kode *while*, *for*, *each*, *loop*)
- *Output*

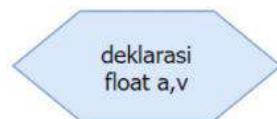
*Flowchart* biasanya digambar menggunakan beberapa simbol standar, namun tidak menutup opsi lain untuk menyertakan simbol-simbol di luar standar untuk digunakan jika memang sangat diperlukan simbol tersebut dalam desain yang kita buat. Adapun simbol-simbol standar yang sering gunakan sebagai berikut:

1. *Terminator* (Terminal): Bentuk bagan alir oval menunjukkan awal atau akhir proses, biasanya berisi kata “*Mulai*”, “*Start*”, “*Selesai*”, “*Akhir*”, atau “*End*”.



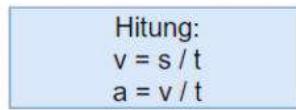
Gambar 2.3  
Simbol Terminal

2. *Preparation (predifined process)*: digunakan sebagai penyiapan *storage* (ruang) atau pendeklarasian sebuah variabel yang dibutuhkan dalam suatu proses.



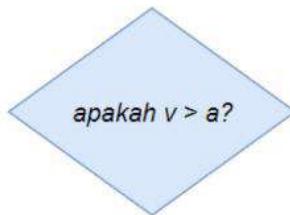
Gambar 2.4  
Simbol Preparasi

3. Proses: Bentuk *flowchart* persegi panjang menunjukkan langkah aliran proses normal/generik. Misalnya, "Hitung kecepatan  $v = s/t$ ".



Gambar 2.5  
Simbol Proses

4. *Decision (Keputusan)*: Bentuk simbol *diamond* (bentuk ketupat) menunjukkan cabang dalam aliran proses. Simbol ini digunakan ketika keputusan harus dibuat, biasanya pertanyaan Ya/Tidak atau Benar/Salah.



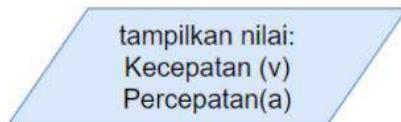
Gambar 2.6  
Simbol Keputusan

5. *Connector (Titik sambung)*: Bentuk lingkaran kecil, berlabel, yang digunakan untuk menunjukkan lompatan dalam aliran proses. *Connector* umumnya digunakan dalam diagram kompleks atau alur panjang hingga lebih dari 1 lembar.



Gambar 2.7  
Simbol Connector

6. Data (*Input/Output*): Bentuk jajaran genjang yang menunjukkan *input* atau *output* data (I/O) untuk suatu proses. Contoh: Dapatkan X dari pengguna, *Display X*.



Gambar 2.8  
Simbol Data (*Input/Output*)

7. *Delay*: Digunakan untuk menunjukkan penundaan atau menunggu proses untuk masukan dari beberapa proses lainnya.



Gambar 2.9  
Simbol *Delay*

8. *Arrow* (Tanda Panah): digunakan untuk menunjukkan aliran kontrol dalam suatu proses. Panah yang datang dari satu simbol dan berakhir pada simbol lain menunjukkan bahwa kontrol melewati simbol yang ditunjukkan oleh panah.



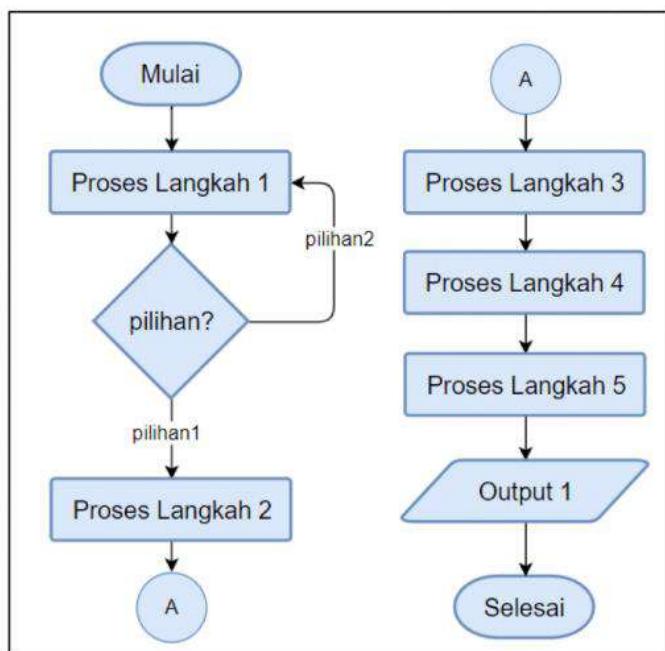
Gambar 2.10  
Simbol Tanda Panah

Simbol-simbol di atas adalah simbol dasar yang digunakan secara umum. Sekarang, pedoman dasar untuk menggambar *flowchart* dengan simbol di atas adalah:

1. Dalam menggambar *flowchart* yang tepat, semua persyaratan yang diperlukan harus dicantumkan dalam urutan logis.
2. Diagram alur harus rapi, jelas, dan mudah diikuti. Seharusnya tidak ada ruang untuk ambiguitas dalam memahami *flowchart*.
3. *Flowchart* harus dibaca dari kiri ke kanan atau dari atas ke bawah.
4. Sebuah simbol proses hanya dapat memiliki satu garis aliran yang keluar darinya.

5. Untuk simbol keputusan, hanya satu garis aliran yang dapat memasukinya, tetapi garis keluarannya memiliki 2 dengan kemungkinan (*true/false*).
6. Simbol terminal hanya dapat memiliki satu garis aliran saja.

Lihat *flowchart* di bawah, hampir memuat semua simbol dasar yang ada.



Gambar 2.11  
Contoh *Flowchart*

Di bawah ini terdapat contoh kasus dan kemudian diselesaikan dengan kalimat bebas dan sebuah *flowchart*.

Kasus 1:

Tiga angka yang di-*input* bebas dan acak, 3 angka tersebut disimpan dalam sebuah variabel, buatlah *flowchart* untuk mencari angka terbesar?

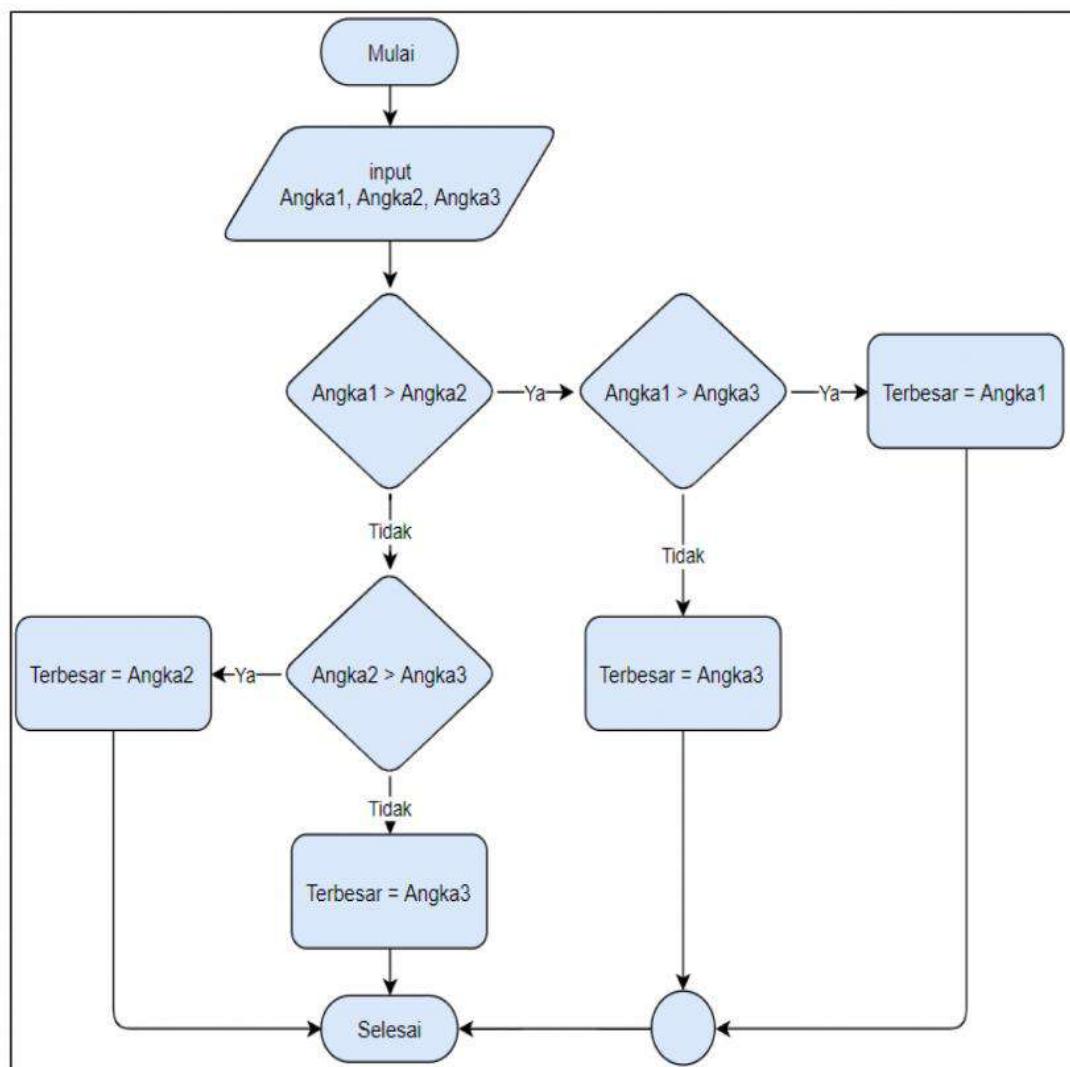
Penyelesaian kasus dengan bahasa bebas:

*Angka1, Angka2, dan Angka3* yang di-*input* kita bandingkan satu demi satu, *Angka1* dibandingkan dengan *Angka2*, jika *Angka1* lebih besar dengan *Angka2*, kemudian *Angka1* dibandingkan lagi dengan *Angka3*, jika *Angka1* lebih besar dari *Angka3*, maka Terbesar adalah *Angka1*. Tapi jika *Angka1* lebih kecil dari *Angka2*, maka *Angka2* dibandingkan dengan *Angka3*, jika

*Angka2 lebih besar dengan Angka3 maka Terbesar adalah Angka3, dan jika tidak maka Angka3 yang Terbesar.*

Membaca penyelesaian di atas, harus lebih hati-hati dalam membacanya.

Penyelesaian kasus dengan *flowchart*:



Gambar 2.12  
*Flowchart Kasus 1*

Penyelesaian dengan *flowchart* jauh lebih sederhana dibandingkan penyelesaian secara kalimat.

**LATIHAN**

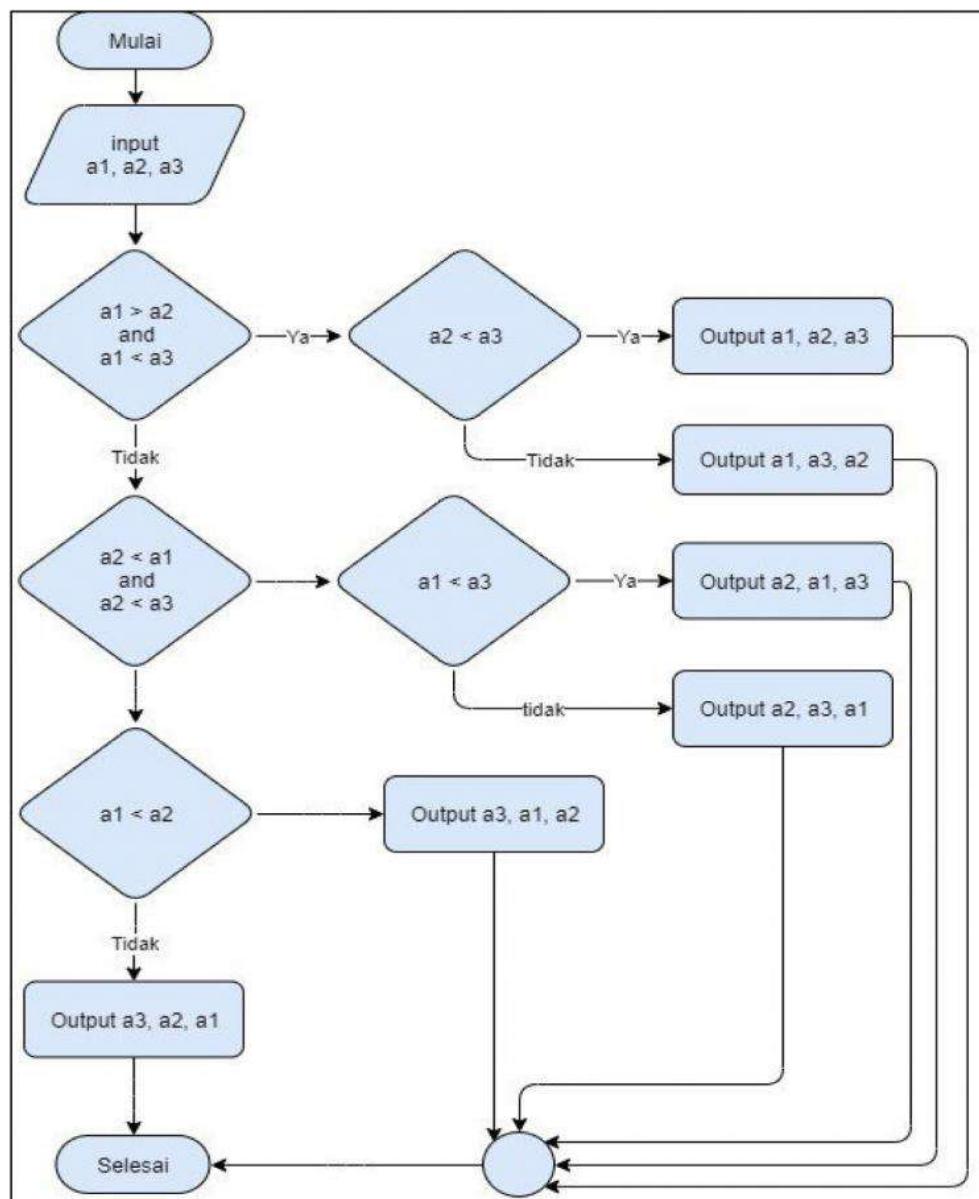
---

Untuk memperdalam pemahaman Anda mengenai materi di atas, kerjakanlah latihan berikut!

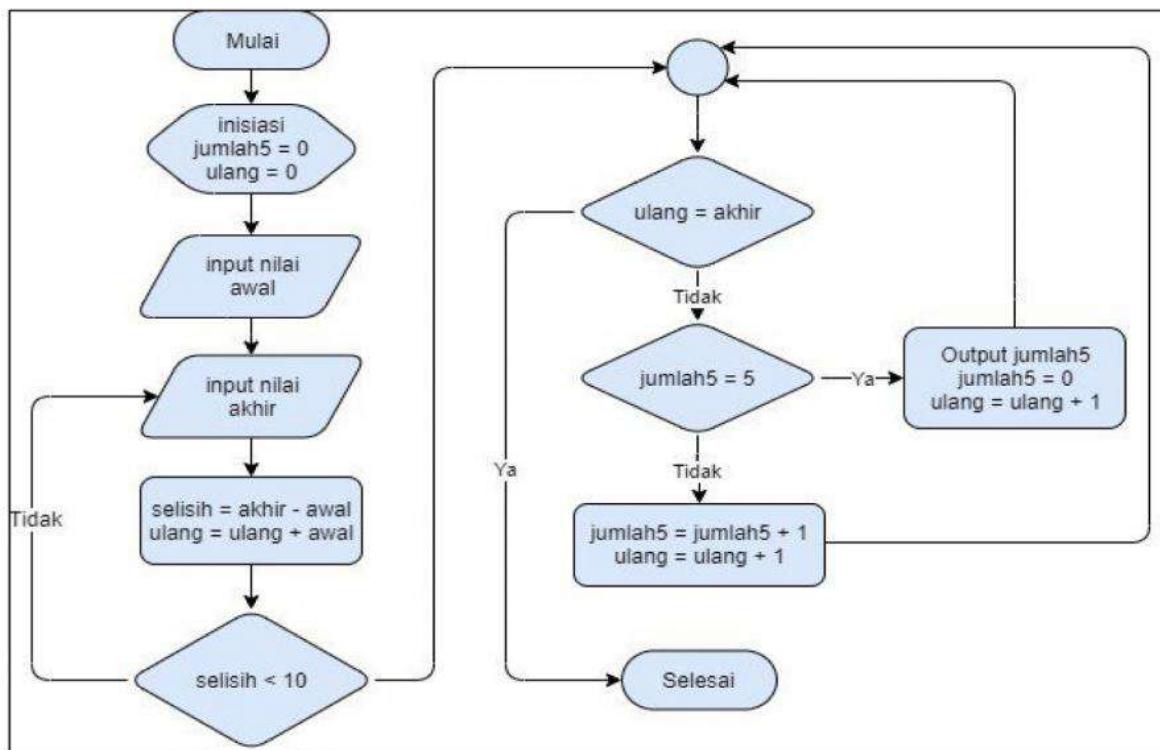
- 1) Buatlah *flowchart* untuk meng-input 3 angka ( $a_1, a_2, a_3$ ), dan kemudian angka tersebut diurutkan dari nilai terkecil ke nilai terbesar?
- 2) Buatlah *flowchart* perulangan dengan kondisi sebagai berikut:
  - a. Nilai awal dan nilai akhir *di-input*.
  - b. Nilai akhir dikurang dengan nilai awal.
  - c. Nilai selisih adalah pengurangan nilai akhir dengan nilai awal.
  - d. Jika nilai selisih kurang dari 10 maka kembali menginput nilai akhir.
  - e. Setelah ditampilkan jumlah5, nilai jumlah5 kembali menjadi 0 ( $\text{jumlah5} = 0$ ). Perulangan ini akan berulang sejumlah nilai selisih (nilai akhir – nilai awal).

*Petunjuk Jawaban Latihan*

- 1) Algoritma dalam bentuk *flowchart* untuk mengurutkan 3 angka sebagai berikut:



2) Berikut *flowchart* perulangan untuk menjawab soal nomor 2.



### RANGKUMAN

*Flowchart* adalah jenis diagram (grafis atau simbolik) yang mewakili suatu algoritma atau proses. Setiap langkah dalam proses diwakili oleh simbol yang berbeda dan berisi penjelasan singkat tentang langkah proses. Adapun keuntungan menggunakan *flowchart* adalah: komunikasi, analisis yang efektif, dokumentasi, pengkodean yang efisien, mempermudah *debugging*, dan pemeliharaan program yang efisien. *Flowchart* sangat penting digunakan pada saat berkomunikasi dengan orang tentang suatu proses dalam program. Ketika membuat proyek baru digunakan untuk mengurai masalah besar kemudian memecah menjadi bagian kecil dengan *flowchart*; penggambaran sebuah proses dilakukan secara tuntas dan detil dengan simbol-simbol; dan memberikan pemahaman umum tentang proses; membantu anggota tim dalam memahami proses secara umum dalam diskusi-diskusi kelompok; juga pada saat menjelaskan proses alur program secara umum kepada orang lain.

Simbol-simbol *flowchart* yang standar digunakan ada 7 simbol, yaitu:

1. Terminator dengan bentuk oval menunjukkan awal dan akhir dari proses;
2. Proses dengan bentuk persegi panjang menunjukkan aliran proses;
3. *Decision* (Keputusan) dengan simbol *diamond* (bentuk ketupat) menunjukkan percabangan;
4. Konektor dengan simbol lingkaran kecil terkadang berlabel walaupun kadang-kala label tersebut tidak dicantumkan, menunjukkan lompatan aliran proses;
5. Data (I/O) dengan bentuk jajaran genjang yang menunjukkan *Input/Output*;
6. *Delay* adalah menunggu proses atau menunggu masukan;
7. *Arrow* (Tanda Panah) digunakan untuk menunjukkan aliran kontrol dalam suatu proses.

Aturan main dalam menggunakan simbol-simbol *flowchart* adalah sebagai berikut: Harus dicantumkan dalam urutan logis, harus rapi, jelas, dan mudah diikuti, dan tidak mengandung ambiguitas; harus dibaca dari kiri ke kanan atau dari atas ke bawah; sebuah simbol proses hanya dapat memiliki satu garis aliran yang keluar darinya; untuk simbol keputusan, hanya satu garis aliran yang dapat memasukinya, tetapi beberapa baris dapat meninggalkannya untuk menunjukkan kemungkinan jawaban; simbol terminal hanya dapat memiliki satu garis aliran saja.



### TES FORMATIF 1

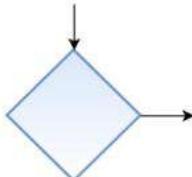
---

Pilihlah satu jawaban yang paling tepat!

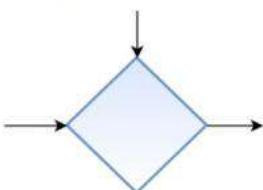
- 1) Semua pernyataan di bawah ini berkaitan dengan dengan *flowchart*, kecuali....
  - A. Grafis atau simbolik
  - B. Mewakili suatu algoritma atau proses
  - C. Algoritma lebih dulu daripada *flowchart*
  - D. *Flowchart* lebih dulu daripada algoritma
- 2) Salah satu keuntungan menggunakan *flowchart* adalah komunikasi. Artinya adalah ....
  - A. Dengan bantuan *flowchart* komunikasi pengguna dengan sistem komputer menjadi lancar
  - B. Komunikasi antar fungsi berjalan dengan sempurna

- C. Komunikasi antara pengguna dengan pengguna yang lain menjadi lancar
  - D. Cara yang lebih baik untuk mengkomunikasikan logika suatu sistem kepada semua unit terkait
- 3) Jika mengikuti aturan yang benar dalam simbol keputusan, maka penggunaan simbol keputusan yang salah di bawah ini adalah....

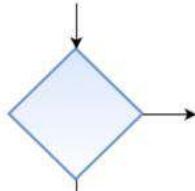
A.



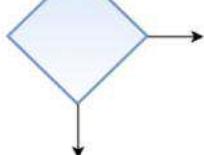
B.



C.



D.

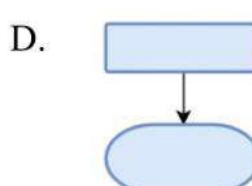
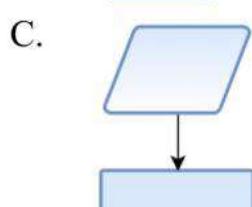
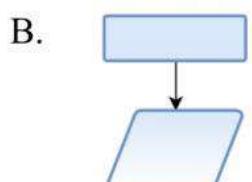
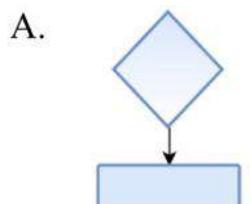


- 4) Sebuah potongan algoritma sebagai berikut:

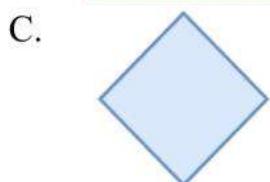
```

...
Input panjang,lebar
Luas = panjang * lebar
...
  
```

Potongan algoritma di atas cocok dengan potongan *flowchart* ....



5) Simbol yang digunakan untuk proses pada *flowchart* adalah simbol....

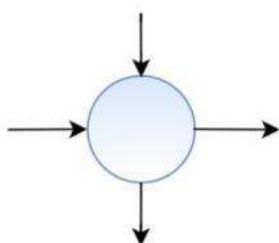


6) Pedoman dasar dalam menggunakan simbol-simbol *flowchart*, kecuali....

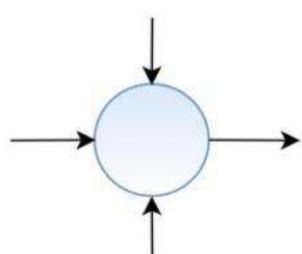
- A. Dalam menggambar *flowchart* yang tepat, semua persyaratan yang diperlukan harus dicantumkan dalam urutan logis
- B. Diagram alur harus rapi, jelas, dan mudah diikuti. Seharusnya tidak ada ruang untuk ambiguitas dalam memahami *flowchart*

- C. Direkomendasikan agar *flowchart* dibuat melalui diskusi kelompok, karena individu jarang mengetahui keseluruhan proses
- D. Untuk simbol keputusan, hanya satu garis aliran yang dapat memasukinya, tetapi beberapa baris dapat meninggalkannya untuk menunjukkan kemungkinan jawaban
- 7) Perhatikan algoritma berikut:
1. Start
  2. Input NilaiUTS
  3. Input NilaiUAS
  4. NilaiAkhir = (NilaiUTS + NilaiUAS)/2
  5. If NilaiAkhir >= 60, then
    - 5.1. print "Lulus"
  6. Else print "Tidak Lulus"
  7. End
- Pada algoritma di atas, jika dibuat menjadi *flowchart* yang utuh, maka terdiri dari simbol-simbol....
- A. 2 Terminator, 5 I/O, 2 Keputusan, 2 Proses, dan 7 Tanda panah
- B. 2 Terminator, 4 I/O, 1 Keputusan, 1 Proses, dan 7 Tanda panah
- C. 2 Terminator, 3 I/O, 1 Keputusan, 2 Proses, dan 8 Tanda panah
- D. 2 Terminator, 4 I/O, 1 Keputusan, 2 Proses, dan 9 Tanda panah
- 8) Simbol konektor dan tanda panah yang benar ditunjukkan pada gambar....

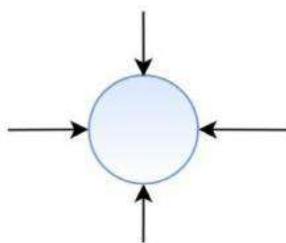
A.



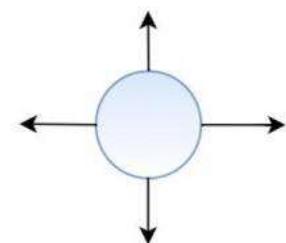
B.



C.



D.



- 9) "Simbol ini digunakan untuk menunjukkan aliran kontrol dalam suatu proses, simbol datang dari satu simbol dan berakhir pada simbol lain menunjukkan bahwa keterhubungan".

Simbol yang digambarkan oleh kalimat di atas adalah....

- A. Simbol Konektor
- B. Simbol Proses
- C. Simbol I/O
- D. Simbol Keputusan

- 10) Jumlah simbol standar dalam *flowchart* terdiri dari....

- A. 5 Simbol (Terminator, Proses, Keputusan, Konektor, I/O)
- B. 6 Simbol (Terminator, Proses, Keputusan, Konektor, I/O, Arrow)
- C. 7 Simbol (Terminator, Proses, Keputusan, Konektor, I/O, Delay, Arrow)
- D. 8 Simbol (Terminator, Proses, Keputusan, Konektor, I/O, Delay, Arrow, Print)

Cocokkanlah jawaban Anda dengan Kunci Jawaban Tes Formatif 1 yang terdapat di bagian akhir modul ini. Hitunglah jawaban yang benar. Kemudian, gunakan rumus berikut untuk mengetahui tingkat penguasaan Anda terhadap materi Kegiatan Belajar 1.

$$\text{Tingkat penguasaan} = \frac{\text{Jumlah Jawaban yang Benar}}{\text{Jumlah Soal}} \times 100\%$$

Arti tingkat penguasaan: 90 - 100% = baik sekali

80 - 89% = baik

70 - 79% = cukup

< 70% = kurang

Apabila mencapai tingkat penguasaan 80% atau lebih, Anda dapat meneruskan dengan Kegiatan Belajar 2. **Bagus!** Jika masih di bawah 80%, Anda harus mengulangi materi Kegiatan Belajar 1, terutama bagian yang belum dikuasai.

**KEGIATAN BELAJAR 2****Pseudocode**

Bahasa pemrograman yang banyak digunakan saat ini sudah menggunakan keyword bahasa alamiah (*natural language*); hal ini berbeda dengan bahasa pemrograman pada saat pertama kali komputer digunakan, yang pada saat itu menggunakan bahasa mesin. Pada saat ini, kita bebas dalam menetapkan nama variabel dengan mengikuti kaidah yang telah ditetapkan, misalnya saja variabel dengan nama “*kecepatan*”, “*tinggi*”, “*berat*”, “*luas*”. Pernyataan-pernyataan dalam bahasa pemrograman, pada umumnya menggunakan bahasa Inggris untuk keyword-nya. Namun untuk kata/kalimat dalam setiap baris pada *pseudocode* boleh menggunakan kata/kalimat yang dari bahasa yang dikuasai.

**A. KALIMAT DEKLARATIF**

Bagi analis/programmer yang sudah terbiasa dengan penggunaan bahasa Inggris maka dapat dengan mudah membuat *pseudocode* dengan keyword mendekati bahasa pemrograman pada umumnya, dan ini merupakan nilai tambah bagi analis/programmer yang berasal dari negara-negara yang berbasis bahasa Inggris. Namun kita tidak harus kecewa dengan itu semua. Analis atau programmer yang lahir di Indonesia dengan latar belakang bahasa Indonesia sangat boleh menggunakan bahasa Indonesia dalam membuat *pseudocode*. Jika nanti ada beberapa kata-kata bahasa Inggris dalam contoh-contoh kasus, semata-mata karena penulis ingin menggiring pembaca untuk lebih awal mengenali perintah-perintah dalam bahasa pemrograman Java, misalnya kata-kata “*if*”, “*else*”, “*else if*”, “*while*”, “*do while*”, “*loop*”, dan lain-lain, akan tetapi kata-kata seperti “*Add*” atau “*Divide*”, bisa diganti dengan “*Tambahkan*” atau “*Kalikan*”, atau bagi variabel yang sifatnya deklaratif, kita bisa menggunakan variabel berbahasa Indonesia.

**B. DEFINISI PSEUDOCODE**

*Pseudocode* adalah deskripsi tingkat tinggi dan singkat (ringkas) yang ditulis untuk dibaca oleh manusia dan bukan untuk dibaca oleh mesin

(kompilator). Anda dapat menganggap *pseudocode* sebagai “kode bahasa Inggris” atau “kode bahasa Indonesia” atau yang lainnya yang dapat dipahami oleh siapa pun (bukan hanya ilmuwan komputer). Walaupun *pseudocode* adalah sebuah narasi yang pendek-pendek (khusus) dan sudah mirip-mirip dengan bahasa pemrograman akan tetapi tidak terikat oleh bahasa pemrograman apapun, akan tetapi *pseudocode* dapat dikonversi ke bahasa pemrograman apapun, apalagi bahasa pemrograman tingkat tinggi, seperti Java, Pascal/Delphi, Visual Basic, C++.

### C. CARA MENULIS PSEUDOCODE

Tidak ada aturan baku yang mengikat cara menulis *pseudocode*, namun, ada standar yang bisa diterima secara umum. Seorang pembaca harus dapat mengikuti *pseudocode* dan mampu mengerti dengan baik apa yang diinginkan, sehingga dengan mudah menerjemahkan permasalahan dengan cara pemecahan pada setiap langkah-langkah yang ada. Setelah menulis *pseudocode*, seharusnya *programmer* jauh lebih mudah mengkonversi ke dalam bahasa pemrograman.

Penulisan *pseudocode* dari langkah pertama, mungkin terlihat biasa saja atau mengikuti alur yang ada, akan tetapi pada blok tertentu anda bisa membuat indentasi (tulisan menjorok kedalam) untuk menggambarkan blok kode, sehingga jelas garis mana di dalam *method*, *loop*, dan lain sebagainya. Indentasi sangat penting untuk menulis *pseudocode*. Dalam bahasa pemrograman tidak memperdulikan tidak peduli jika setelah pernyataan *if* menjorok ke dalam atau tidak pada penulisan, tetapi pembaca manusia/*programmer* akan benar-benar kehilangan jejak tanpa tanda indentasi. Penulisan *pseudocode* tidak boleh membuat *programmer* kesusahan dalam mengidentifikasi setiap langkah, sehingga penyertaan nomor urut pada setiap langkah menjadi sangat penting, bahkan jika dalam satu sub blok indentansi, diberikan dengan nomor urut yang anak beranak, misalnya suatu blok dimulai dari nomor urut 6, maka indentasi biasanya diberikan nilai 6.1 atau 6.a (sub urut). Ingat! Pemahaman manusia adalah inti dari *pseudocode*.

Walaupun dalam menyusun *pseudocode* tidak mempunyai aturan baku yang mengikat, namun beberapa panduan umum yang sering digunakan dalam membuat *pseudocode* sebagai berikut:

1. Dituliskan dalam bahasa yang sederhana dan mudah dimengerti, setiap *pseudocode* boleh dimulai dengan kata-kata “*Mulai*” dan diakhiri dengan “*Selesai*”.
2. Notasi  $\leftarrow$  digunakan untuk mengisi nilai pada sebuah variabel, contoh:

```
jarak  $\leftarrow$  10  
waktu  $\leftarrow$  20  
kecepatan  $\leftarrow$  jarak / waktu  
percepatan  $\leftarrow$  kecepatan / waktu
```

- Semua apa yang ada di belakang notasi  $\leftarrow$  adalah nilai yang akan diisi kedalam variabel yang ditujunya (variabel sebelah kiri, *jarak*, *waktu*, *kecepatan* dan *percepatan*).
3. Setiap pernyataan atau intruksi dapat *independen* atau ditulis dalam baris sendiri, contoh:

```
kecepatan  $\leftarrow$  10 / 20
```

- Pernyataan di atas menugaskan untuk mengisi hasil pembagian bilangan 10 dibagi dengan 20 ke dalam variabel *kecepatan*.
4. Disarankan (tidak wajib diikuti) untuk variabel skalar menggunakan huruf kecil, variabel skalar adalah variabel menyimpan nilai yang dapat berubah nilainya, contoh:

```
v  $\leftarrow$  s / t
```

- Variabel *v*, tentu sangat bisa berubah nilainya, tergantung nilai yang diberikan pada variabel *s* dan *t*. Atau perubahan nilai *v* bisa terjadi jika pada instruksi berikutnya.
5. Disarankan (tidak wajib) untuk variabel larik (*array*), menggunakan huruf besar semua atau menggunakan huruf besar di depannya, contoh:

```
Larik ← [6,7,9,10]
L ← [2,3,4]
LARIK ← [6,5,9,9]
```

Misalnya: Variabel  $L \leftarrow [2,3,4]$ , berisi tiga buah angka yaitu 2, 3, dan 4. Angka 2 elemen pertama, angka 3 elemen kedua, dan 4 elemen ketiga dari variabel L.

6. Notasi seperti  $L[i]$ , menyatakan elemen ke-i dari variabel larik L, Larik selalu dimulai dari 0 (Nol), sehingga  $L[0]$  merupakan elemen pertama dari variabel L. Sedangkan untuk bernalotasi  $L[i,j]$ , dengan i adalah indeks untuk baris, dan j adalah indeks untuk kolom, contoh:  $L[0,1]$  maka dibaca dengan variabel L pada baris ke-0, kolom ke-1.
7. Notasi  $jumlahElemen(**L**)$  digunakan sebagai ekspresi untuk mendapatkan jumlah elemen larik.
8. Variabel majemuk yang digunakan untuk menyimpan tipe majemuk yang dapat menyimpan beberapa jenis data sekaligus, contoh penggunaan:

```
kendaraan = GROUP
    nama
    jenis
    penemu
AKHIR-GROUP
```

Dan untuk mengisi variabel majemuk tetap dengan notasi  $\leftarrow$ , misalnya:

```
Nama ← "Sepeda"
Jenis ← "Sepeda Gunung"
Penemu ← "Baron Karl Von Drais"
```

9. Bisa menyertakan nomor urut pada setiap baris, baik pernyataan ataupun komentar, jika dalam *pseudocode* mengandung sub-blok, maka disarankan ditandai dengan indentasi, misalnya penggunaan *if*:

```
1. ...
2. if (kecepatan > 50) then
   2.1. pernyataan-1
   2.2. Pernyataan-2
   2.3. Pernyataan-3
3. end-if
4. ...
```

atau dengan bahasa Indonesia:

```
3. ...
4. jika (kecepatan > 50) maka
   4.1. pernyataan-1
   4.2. Pernyataan-2
   4.3. Pernyataan-3
5. Akhir-jika
6. ...
```

Silahkan menggunakan sesuai selera.

10. Pembacaan *pseudocode* dilakukan secara urut, jika suatu kondisi harus lompat ke urutan tertentu, maka harus ditulis lompatnya kemana. Contoh dari langkah 2.2, karena kondisi tertentu harus lompat ke langkah 3, maka harus dituliskan *lompat ke langkah 3* atau *Go to step 3*.
11. Simbol // atau /\* ... \*/ digunakan untuk komentar, contoh:

```
// komentar
```

atau

```
/* komentar multi baris
   Baris-x
   Baris-x
*/
```

12. Setiap baris instruksi harus jelas, misalnya: sebuah variabel menyertakan tipe datanya.

```
...
float kecepatan, percepatan
...
```

13. Notasi *masukan()*, *input()*, *tampilkan()*, *cetak()*, atau *print()* mewakili I/O. Contoh:

```
Masukkan(jarak)
Masukkan(waktu)
Cetak()
Print()
```

Notasi di atas mulai semakin semakin beragam, apalagi membuat *pseudocode* terafiliasi dengan bahasa pemrograman tertentu, sehingga membuat beragam untuk sebuah masukan atau keluaran.

14. Untuk Logika, jika pernah belajar logika informatika, maka dengan mudah memahami sebuah logika nilainya hanya “Benar/true” atau “Salah/false”. Jika komponen tersebut adalah sebuah operator pembanding maka operatoriannya berikut:

Operator / Notasi	Keterangan
<	Lebih kecil
$\leq$	Lebih kecil sama dengan
>	Lebih besar
$\geq$	Lebih besar sama dengan
$\neq$	Tidak sama dengan
$\neq$	Tidak sama dengan

Dan operator logika sebagai berikut:

Operator / Notasi	Keterangan
AND	dan
OR	atau

15. *Pseudocode* untuk perulangan (*while-loop*), selama kondisi bernilai benar (*true*), maka perulangan akan dijalankan terus:

```

ULANG-SELAMA (kondisi)
  pernyataan-1
  pernyataan-2
  ...
AKHIR-ULANG

```

16. *Pseudocode* perulangan (*do-loop*), blok perulangan akan berjalan minimal sekali, baru kemudian melakukan pengecekan benar (*true*) atau salah (*false*), jika kondisi benar maka akan terus melakukan perulangan.

```

ULANG
  pernyataan-1
  pernyataan-2
  ...
SELAMA (kondisi)

```

17. Untuk perulangan (*for-loop*), sebuah perulangan akan berhenti selama nilainya sampai pada akhir yang ditentukan, misal 1 sampai 10, maka akan melakukan perulangan sampai 10.

```
UNTUK (variabel = awal ke akhir)
    pernyataan-1
    pernyataan-2
    ...
AKHIR-UNTUK
```

Atau menggunakan:

```
for (variabel = 1 to 10)
    pernyataan-1
    pernyataan-2
    ...
End-for
```

18. Untuk prosedur/fungsi/*method*:

```
Procedure nama_prosedur(parameter)
    pernyataan-1
    pernyataan-2
    ...
Akhir-Procedure
```

Begitu juga dengan fungsi/*method*, atau sebuah *class* ataupun *sub-class*.

Mari kita lihat contoh *pseudocode* dan bagaimana bentuk asli dalam bentuk bahasa pemrograman Java untuk mencari kecepatan dan percepatan, sebagai berikut:

<b>Pseudocode (Ideal) Mencari Kecepatan Dan Percepatan</b>	
1	Mulai ( <i>opsional, ada yang menggunakan ada yang tidak</i> )
2	Kelas <i>MencariKecepatandanPercepatan</i>
3	//Deklarasi
4	<i>float</i> kecepatan, percepatan
5	<i>float</i> jarak,waktu
6	//Inisisasi awal: (bisa berupa input)
7	jarak ← 123
8	waktu ← 2
9	//Formulasi Kecepatan: (Komentar)
10	kecepatan ← jarak / waktu
11	//Formulasi Percepatan:
12	percepatan ← kecepatan / waktu
13	//Tampilkan:
14	Cetak kecepatan
15	Cetak percepatan
16	Selesai

*Pseudocode* di atas adalah *pseudocode* sangat ideal, begitu gamblang dan jelas dalam penyajian baris demi baris, mulai dari penamaan proses (pada Java akan membuat *class MencariKecepatandanPercepatan*), variabel-variabel yang digunakan dan lengkap dengan tipe datanya, formulasi kecepatan dan percepatan jelas, kemudian tahap akhir adalah menampilkan hasil. Proses demi proses bercerita dan memberikan instruksi yang sangat informatif, khususnya kepada *programmer* karena sudah tahu apa yang harus dikerjakan, tidak perlu mengorek informasi lebih jauh tentang variabel, formulasi dan hasil akhir akan dicetak atau tidak. Jika algoritma di atas ditransformasi ke bahasa Java, maka hasilnya sebagai berikut:

### Mencari Kecepatan dan Percepatan

```
1 //Nama Kelas (Program)
2 class MencariKecepatandanPercepatan {
3
4     public static void main(String args[])
5     {
6         //Deklarasi
7         float kecepatan,percepatan;
8         float jarak,waktu;
9         //Inisisasi
10        jarak = 123;
11        waktu = 2;
12        //formulasi kecepatan dan percepatan
13        kecepatan = jarak/waktu;
14        percepatan = kecepatan/waktu;
15        //cetak
16        System.out.println("Kecepatan :"+kecepatan);
17        System.out.println("Percepatan :"+percepatan);
18    }
19 }
```

Bandingkan dengan *pseudocode* tidak ideal di bawah ini:

### Pseudocode (Tidak Ideal) Mencari Kecepatan Dan Percepatan

- 1 Mulai
- 2 Kelas **MencariKecepatandanPercepatan**
- 3 Deklarasikan variabel
- 4 Lakukan Inisisasi awal
- 5 Hitung Kecepatan
- 6 Hitung Percepatan
- 7 Tampilkan kecepatan
- 8 Tampilkan percepatan
- 9 Selesai

Mari mengoreksi baris demi baris *pseudocode* yang tidak ideal di atas, sebagai berikut:

1. Baris 1 dan 2: tidak ada masalah karena cukup informatif
2. Baris 3: *programmer* bisa saja membuat variabel dan tipe data dan mengantisipasi tipe data dengan tipe data *float* jika ada hasil pembagian.
3. Baris 4: mulai timbul pertanyaan, berapa variabel harus dibuat untuk diinisiasi, bisa saja *programmer* buat 1, 2, 5, atau 10 variabel, kemudian diinisiasi awal dengan angka sembarang, *programmer* akan berpikir berapa variabel dan apa tipe data yang harus digunakan.
4. Baris 5 dan 6: Programmer akan sangat bingung, formulasi kecepatan dan percepatan seperti apa? Apakah betul semua *programmer* mengetahui formulasi kecepatan dan percepatan dalam ilmu fisika? Tentu tidak!, Artinya bahwa *programmer* masih butuh informasi tambahan.
5. Baris 7 dan 8: nilai kecepatan dan percepatan akan ada jika baris baris 5 dan 6 bisa diselesaikan.
6. Baris 9: hanya penutup proses.

Melihat langkah-langkah *pseudocode* yang tidak ideal di atas akan memberikan sebuah pertanyaan besar, yaitu: Variabel apa saja yang dibutuhkan untuk mencari kecepatan dan percepatan serta apa tipe datanya apa? Bagaimana mungkin bisa mencari kecepatan dan percepatan jika saya tidak mengetahui formulasi? Hal ini adalah sinyal/indikasi bahwa *pseudocode* tersebut tidak cukup detail. Berinisiatif untuk membangun deklarasi variabel tidak akan pernah memecah kebingungan dan kebuntuan yang ada.

## D. STRUKTUR PSEUDOCODE

Melihat beberapa *pseudocode* dari buku, internet atau dari analis, ditemukan beragam cara penyajian *pseudocode*, ada yang menggunakan nomor urut dan ada yang tidak, ada yang sudah menggunakan pernyataan yang mirip dengan bahasa pemrograman tertentu, bahkan ada yang hanya menyertakan prosesnya saja, ada yang menyertakan lengkap seperti tipe data dari variabel, ada yang mencampurkan bahasa Inggris dengan bahasa Indonesia, dan berbagai variasi lainnya. Kita harus menerima fakta bahwa sulit untuk menerapkan suatu standar tertentu apabila kita sudah berkiblat

kepada sebuah bahasa pemrograman tertentu dan melihat latar belakang para analis yang amat berbeda. Oleh karena itu suatu *pseudocode* diusahakan untuk dibuat mendekati standar agar dapat mudah dipahami oleh para programer.

Tentu kita tidak bisa memaksakan sebuah struktur yang baku, mengingat karakteristik setiap bahasa pemrograman yang tersedia memiliki beberapa perbedaan. Tidak ada bahasa pemrograman yang benar-benar sama persis. Beberapa bahasa pemrograman memiliki kemiripan, misalnya: bahasa pemrograman Java dan Pascal. Secara substansi kedua bahasa pemrograman ini mempunyai karakteristik yang berbeda, contoh: Dalam mendeklarasikan variabel, Pascal mendeklarasikan variabel di luar blok program (*begin...end;*), di sisi lain bahasa Java bebas dalam lokasi mendeklarasikan variabel, bahkan dalam blok (*{...}*) program sekalipun tetap diijinkan. Inti dari *pseudocode* adalah mampu dibaca oleh *programmer* atau bukan *programmer* dengan tanpa perlu mempelajari ilmu tertentu untuk bisa mengerti.

Secara umum, dapat dikatakan bahwa *pseudocode* bisa distrukturkan sebagai berikut: dimulai dengan Judul (Nama Program, *class*), Deskripsi (Deklarasi), Implementasi (Inti dari Algoritma).



Gambar 2.1  
Struktur *Pseudocode*

## E. PERBEDAAN GAYA PENULISAN (PEMBUATAN) *PSEUDOCODE*

*Pseudocode* sangat dekat dengan prosa bahasa Inggris, misalnya pernyataan “while”, “if”, “else” dan sebagainya. Namun kita masih diijinkan untuk membuat suatu toleransi selama memungkinkan dan khalayak umum mampu memahami maksud dan tujuan dari pernyataan tersebut. Gaya *pseudocode* sangat bergantung pada preferensi pribadi, masalah yang ingin dipecahkan. Makin sering Anda membuat/menulis *pseudocode* maka Anda akan menemukan gaya apa yang paling alami dan paling cocok bagi Anda.

## F. CONTOH-CONTOH ALGORITMA DENGAN *PSEUDOCODE*

Sebagai bahan pengayaan, berikut ini diberikan contoh *pseudocode*.

*Algoritma untuk mencari bilangan ganjil dari 1 sampai 10*

Pada *pseudocode* berikut adalah untuk mencari bilangan ganjil dari angka 1 sampai dengan angka 10, tentu yang dicari adalah 1, 3, 5, 7, dan 9. *Pseudocode*-nya sebagai berikut:

<b><i>Pseudocode Mencari Bilangan Ganjil Dari 1 - 10</i></b>	
1	Mulai
2	Kelas <b><i>MencariBilGanjil1sd10</i></b>
3	Deklarasi
4	int i
5	Inisisasi awal:
6	i ← 1
7	Perulangan:
8	While ( <i>selagi</i> i < 11
8.1	Kondisi:
8.2	if ( <i>jika</i> i tidak habis dibagi 2 maka
8.2.1	Cetak i
8.3	i ← i + 1
9	Selesai

Untuk baris 1 sampai dengan baris 7, sudah sangat jelas, pada baris 8 mengandung indentasi 8.1 hingga 8.3 dapat dikatakan satu blok yang utuh. Untuk 8.2 mempunyai sub indentasi 8.2.1 memberikan informasi jelas merupakan sub blok dalam satu blok.

Perulangan (lihat *While...* pada baris 8) adalah membentuk blok indentasi, minimal 1 indentasi dalam bloknya (bloknya 8.1 hingga 8.3), kemudian untuk kondisi (lihat *if...* pada baris indentasi 8.2), minimal memiliki 1 indentasi dalam bloknya.

Sekarang kita masuk kepada substansi algoritma, kenapa harus menggunakan algoritma untuk mencari nilai ganjil dari angka 1 sampai 10? Terkadang pertanyaan sepele seperti ini sering muncul dan menyepelekan hasil karya orang lain, tanpa berpikir panjang bahwa seandainya data yang dikerjakan bukan hanya 1 sampai 10, akan tetapi sampai 100, 500, 1000, atau 10.000, tentu akan sangat merepotkan jika harus mencari satu per satu. Sangat gampang menemukan dan menuliskan jika datanya hanya 1 sampai 10, dimana hasilnya adalah 1, 3, 5, 7, 9. Tetapi akan memerlukan waktu yang lama untuk menemukan angka ganjil antara 1 sampai 1000. Nah disinilah fungsinya algoritma dan bahasa program. Anda hanya menginput nilai dan selanjutnya menekan tombol *enter*, hasilnya keluar kurang dari 1 detik.

*Pseudocode* mengurutkan data di atas jika ditransformasi kedalam bahasa pemrograman Java, maka bentuknya seperti berikut:

### Mencari bilangan Ganjil

```
1 //Nama Kelas (Program)
2 class MencariBilGanjil1sd10 {
3
4     public static void main(String args[])
5     {
6         //Deklarasi
7         int i;
8         //Inisisiasi
9         i = 1;
10        //perulangan dengan while
11        while (i < 11) { //Awal Blok while
12            //kondisi
13            if(i % 2 == 1) { //Awal Blok if
14                System.out.println(i);
```

```

15 } //Akhir blok if
16 //var i ditambah 1
17 //pada setiap perulangan
18 i++;
19 } //Akhir blok while
20 }
21 }

```

Keluaran:

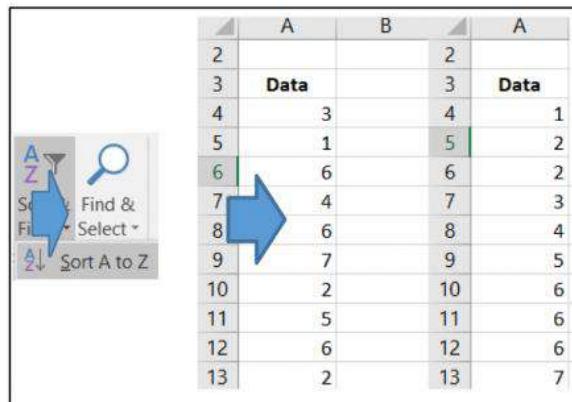
```

1
3
5
7
9

```

### *Algoritma untuk mengurutkan data*

Mungkin anda pernah menggunakan Microsoft Office Excel, dalam satu kolom terdapat data *random* (tidak berurutan), pada Excel terdapat fungsi untuk mengurutkan data dengan nama fungsi “*Sort & Filter*”.



A	B	A
2		2
3	Data	3
4	3	4
5	1	5
6	6	6
7	4	7
8	6	8
9	7	9
10	2	10
11	5	11
12	6	12
13	2	13

Gambar 2.2  
Contoh Fungsi Sorting pada Aplikasi Excel

Gambar 2.2 adalah data di aplikasi excel, data sebelah kiri adalah data yang belum disortir, dan yang sebelah kanan adalah data yang sudah disortir. Fungsi “*Sort & Filter*” sangat mudah dan sederhana penggunaannya. Di balik kemudahan penggunaan fungsi sortir, tersimpan algoritma pengurutan data.

Contoh kasus lain misalkan kita mempunyai sekumpulan data *random* yang belum terurut, adapun data-data tersebut adalah sebagai berikut: 3, 1, 6, 4, 6, 7, 2, 5, 6, 3. Buatlah *pseudocode* untuk mengurutkan data.

### Pseudocode Pengurutan Data

1	Mulai
2	//Deklarasikan Variabel:
3	int Data[] ← {3,1,6,4,6,7,2,5,6,3}
4	int i,j,hasil
5	//Cetak Data elemen sebelum diurutkan:
6	for (i ← 0; i < Data.jumlahdata; i←i+1)
6.1	Cetak i
7	//Proses pengurutan dan perbandingan data:
8	for (i ← 0; i<jumlahElemen(**Data**); i←i+1)
8.1	for (j ← 0; j<jumlahElemen(**Data**); j←j+1)
8.1.1	//Kondisi (dibandingkan):
8.1.2	if (Data[i] < Data[j])
8.1.2.1	hasil ← Data[i]
8.1.2.2	Data[i] ← Data[j]
8.1.2.3	Data[j] ← hasil
9	//Cetak hasil pengurutan:
10	for (j ← 0; j < jumlahElemen(**Data**); j←j+1)
10.1	Cetak j
11	Selesai

*Pseudocode* di atas, walau belum memiliki kerumitan penyelesaian tinggi, tetapi sudah memiliki indentasi dengan 3 kedalaman, yaitu: 8.1.2.1, 8.1.2.2, dan 8.1.2.3. Jika dirasa indentansi seperti ini terlalu panjang, maka bisa menyingkat dengan 8121, 8122, dan 8123, atau jika merasa terlalu panjang, maka bisa menggantinya dengan huruf, misalnya:

### Pseudocode Pengurutan Data

1	Mulai
2	//Deklarasikan Variabel:
3	int Data[] ← {3,1,6,4,6,7,2,5,6,3}
4	int i,j,hasil

```

5   //Cetak Data elemen sebelum diurutkan:
6   for (i ← 0; i < data.jumlahdata; i=i+1)
7     a   Cetak i
8   //Proses pengurutan dan perbandingan data:
9   for (i ← 0; i< jumlahElemen(**Data**); i←i+1)
10  a    for (j ← 0; j< jumlahElemen(**Data**); j←j+1)
11    a1   //Kondisi (dibandingkan):
12    a2    if (data[i] < data[j])
13    a21   hasil ← data[i]
14    a22   data[i] ← data[j]
15    a23   data[j] ← hasil
16  //Cetak hasil pengurutan:
17  for (j ← 0; j < jumlahElemen(**Data**); j←j+1)
18    a    Cetak j
19  Selesai

```

Dua contoh *pseudocode* di atas, akan sangat mudah dibaca oleh *programmer* secara sistematis dan jika *pseudocode* tersebut ditransformasi ke dalam bahasa pemrograman Java, maka hasilnya sebagai berikut:

### Mengurutkan Data

```

1  class Urut_Data
2    public static void main(String args[])
3    {
4      //Deklrasi dan inisiasi
5      int DATA[]={3, 1, 6, 4, 6, 7, 2, 5, 6, 3};
6      int i,j;
7
8      //mencetak data sebelum diurutkan
9      System.out.println("Data Sebelum urut ascending :");
10
11     for (i=0; i< jumlahElemen(**DATA**); i++)
12     {
13       System.out.print(data[i]+" ");
14     }
15

```

```

16    //mengurutkan data dengan membandingkan
17    //nilai dari kiri ke kanan
18    for (i=0; i< jumlahElemen(**DATA**); i=i+1)
19        for (j=0; j< jumlahElemen(**DATA**); j++)
20            //jika lebih kecil maka
21            //akan digeser ke arah kiri
22            if (data[i]<DATA[j])
23            {
24                int hasil=DATA[i];
25                DATA[i]=DATA[j];
26                DATA[j]=hasil;
27            }
28
29    //mencetak data yang sudah diurutkan
30    //secara ascending
31    System.out.println();
32    System.out.print("Data urut ascending:\n");
33
34    for (j=0; j< jumlahElemen(**DATA**); j++)
35        System.out.print(DATA[j]+ " ");
36
37    }
38 }
```



## LATIHAN

---

Untuk memperdalam pemahaman Anda mengenai materi di atas, kerjakanlah latihan berikut!

- 1) Kasus 1: Buatlah *pseudocode* untuk mengurutkan 3 angka (a1,a2,a3)
- 2) Kasus 2: Buatlah *pseudocode* perulangan dengan kondisi sebagai berikut:
  - a. Nilai awal dan nilai akhir di-*input*.
  - b. Nilai akhir dikurang dengan nilai awal.
  - c. Nilai selisih adalah pengurangan nilai akhir dengan nilai awal.
  - d. Jika nilai selisih kurang dari 10 maka kembali menginput nilai akhir.
  - e. Jumlahkan (hitung) angka setiap perulangan kelima (jumlah5).

- f. Tampilkan jumlah5.
- g. Setelah ditampilkan jumlah5, nilai jumlah5 kembali menjadi 0 (jumlah5 = 0). Perulangan ini akan berulang sejumlah nilai selisih (nilai akhir – nilai awal).

*Petunjuk Jawaban Latihan*

- 1) *Pseudocode* untuk kasus 1 sebagai berikut:

<b>Pseudocode Kasus 1</b>	
1	Mulai
2	//Melakukan penginputan
3	Input a1;
4	Input a2;
5	Input a3;
6	//Proses perbandingan data:
7	if (a1 > a2) and (a1 < a3)
7.1	if a2 < a3
7.1.1	Cetak a1, a2, a3
7.2	else
7.2.1	cetak a1, a3, a2
8	else if (a2 > a1) and (a2 < a3)
8.1	if a1 < a3
8.1.1	cetak a2, a1, a3
8.2	else
8.2.1	cetak a2, a3, a1
9	else if a1 < a2
9.1	cetak a3, a1, a2
10	else
10.1	cetak a3, a2, a1
11	Selesai

2) *Pseudocode* untuk Kasus 2 sebagai berikut:

<b><i>Pseudocode Kasus 2</i></b>	
1	Mulai
2	//Melakukan inisiasi
3	jumlah5 ← 0;
4	ulang ← 0;
5	//Proses penginputan nilai awal
6	input awal
7	Input akhir
8	//Proses aritmetika
9	selisih ← akhir - awal
10	ulang ← ulang + awal
11	//Proses seleksi
12	if selisih < 10
12.1	pergi ke
13	else
13.1	Label ulang
13.2	if ulang = akhir
13.2.1	pergi ke 14 ( <i>Selesai</i> )
13.3	else
13.3.1	if Jumlah5 = 5
13.3.1.1	cetak jumlah5
13.3.1.2	jumlah5 ← 0
13.3.1.3	ulang ← ulang + 1
13.3.1.4	pergi ke ulang 13.1
13.4.1	else
13.4.1.1	jumlah5 ← jumlah5 + 1
13.4.1.2	ulang ← ulang + 1
13.4.1.3	pergi ke ulang 13.1
14	Selesai



Bahasa Inggris memang menjadi pilihan bagi pembuat *pseudocode*. hal ini karena bahasa Inggris sudah sangat mendekati intruksi dan pernyataan-pernyataan dalam Bahasa Pemrograman, misalnya: kata “*if*”, “*else*”, “*while*”, “*do while*”, “*for*”. Akan tetapi kita bisa saja menggunakan bahasa Indonesia untuk mengganti kata-kata seperti “*Add*” menjadi “*ditambah*”, “*Divide*” menjadi “*Kalikan*”, “*Count*” menjadi “*Hitung*”, atau sifatnya variabel deklaratif dalam pembuatan *pseudocode*.

*Pseudocode* adalah deskripsi singkat (ringkas) yang ditulis untuk dibaca oleh manusia, bukan dibaca oleh mesin. *Pseudocode* adalah sebuah narasi yang pendek-pendek (khusus) dan sudah sangat mirip-mirip dengan bahasa pemrograman dan dapat ditransformasi dengan mudah oleh *programmer* kedalam bahasa pemrograman Java, C/C++, atau bahasa pemrograman lainnya.

Cara menuliskan sebuah *pseudocode* tidak memiliki aturan baku yang standar. Inti dari *pseudocode* dapat memperlihatkan alur proses yang jelas dan mudah dimengerti oleh orang awam maupun *programmer*, tidak mengandung ambiguitas. *Pseudocode* bisa diberikan nomor urut, bahkan apabila memungkinkan nomor urut yang ada mencerminkan blok setiap langkah. Penyajian bahasa yang digunakan harus sederhana dan mudah dimengerti, penamaan yang digunakan harus jelas dan detil, misalnya sebuah variabel dengan tipe data yang akan disandingkannya.

Penyajian *pseudocode* sangat beragam, dan usaha untuk menyeragamkannya juga tidak mudah, mengingat karakteristik bahasa pemrograman yang dituju juga sangat beragam, namun struktur dasar yang perlu dipahami adalah Judul (Nama Program, Kelas), Deskripsi (Deklarasi), dan Implementasi (Inti dari Algoritma). Gaya penulisan *pseudocode* ini merupakan suatu toleransi yang sangat tepat, walaupun pada beberapa peristilahan yang digunakan mirip bahasa Inggris. Namun kita tetap diijinkan untuk menggunakan gaya kita masing-masing selama produk tersebut dapat dipahami oleh para penggunanya. Hal ini tentu sangat dipengaruhi oleh preferensi pribadi, masalah yang ingin dipecahkan dan yang mau di optimalkan, misalnya kecepatan, keterbacaan, estetika, dan lain lain.



## TES FORMATIF 2

---

Pilihlah satu jawaban yang paling tepat!

- 1) Deskripsi yang cukup singkat dan ringkas yang mampu dibaca oleh manusia, bukan mesin adalah pengertian dari....
  - A. *Pseudocode*
  - B. Dengan *pseudocode*, algoritma yang rumit bisa menjadi jelas dan mudah dimengerti setiap prosesnya
  - C. Tidak memiliki standar yang jelas dalam pembuatannya
  - D. Bisa menggunakan nomor urut pada setiap proses yang ada
- 2) Salah satu kelebihan dari sebuah *pseudocode* adalah....
  - A. Langsung bisa dipakai dalam bahasa pemrograman tertentu
  - B. Komunikasi antar fungsi berjalan dengan sempurna
  - C. Komunikasi antara pengguna dengan pengguna yang lain menjadi lancar
  - D. Cara yang lebih baik untuk mengkomunikasikan logika suatu sistem kepada semua unit terkait
- 3) Jika terdapat sub proses dalam *pseudocode*, sebaiknya yang dilakukan adalah....
  - A. Tetap menulis urutan-urutan proses sejajar mulai dari awal hingga akhir
  - B. Memberikan keterangan pada sub proses yang ada
  - C. Sub proses harusnya diberikan indentasi kedalaman proses, membuat sub proses menjorok kedalam yang menandakan bahwa hal tersebut adalah sub blok, atau bisa diberikan nomor urut dan sub nomor urut.
  - D. Tidak ada yang benar
- 4) Sebuah potongan *pseudocode* berikut:  
...  
*int panjang*  
*panjang*  $\leftarrow$  20  
*if panjang*  $\leq$  20 *then*  
  *cetak* "Kurang panjang"  
*else*  
  *cetak* "Panjang"  
...

```
...  
int panjang  
panjang ← 20  
if panjang ≤ 20 then  
  cetak "Kurang panjang"  
else  
  cetak "Panjang"  
...
```

Potongan *pseudocode* di atas jika ini disempurnakan sesuai sub proses/langkah, maka yang paling benar adalah....

A. ...

```
int panjang  
panjang ← 20  
if panjang <= 20 then  
    cetak "Kurang panjang"  
else  
    cetak "Panjang"  
...
```

B. ...

```
int panjang  
panjang ← 20  
if panjang <= 20 then  
    cetak "Kurang panjang"  
else  
    cetak "Panjang"  
...
```

C. ...

```
int panjang  
panjang ← 20  
if panjang <= 20 then  
    cetak "Kurang panjang"  
else  
    cetak "Panjang"  
...
```

D. ...

```
int panjang  
panjang ← 20  
if panjang <= 20 then  
    cetak "Kurang panjang"  
else  
    cetak "Panjang"  
...
```

- 5) Walaupun tidak ditemukan struktur yang baku dalam *pseudocode*, namun panduan yang bisa diterima secara umum adalah struktur *pseudocode* dengan urutan sebagai berikut....

- A.
1. Judul, nama program, *class*
  2. Implementasi
  3. Deskripsi

- B. 1. Implementasi  
2. Deskripsi  
3. Judul, nama program, *class*
  - C. 1. Judul, nama program, *class*  
2. Deskripsi  
3. Implementasi
  - D. 1. Deskripsi  
2. Implementasi  
3. Judul, nama program, *class*
- 6) Contoh sebuah *pseudocode*

1	<i>Mulai</i>
2	<i>int panjang, Luas</i>
3	<i>panjang ← 30</i>
4	<i>Lebar ← 20</i>
5	<i>Luas ← panjang * Lebar</i>
	<i>cetak Luas</i>
	<i>Selesai</i>

Jika harus memberikan narasi pada *pseudocode* di atas adalah....

- A. Semua urutan sudah benar, tak perlu ada yang dikoreksi lagi
  - B. Perlu indentasi untuk operasi luas
  - C. Variabel harus disempurnakan, agar tidak membingungkan *programmer*
  - D. Walaupun variabel tidak disempurnakan, *programmer* akan mengetahui bahwa yang kurang adalah variabel lebar
- 7) *Pseudocode* yang tidak ideal adalah ....

- A. Gamblang dan jelas dalam penyajian baris demi baris, mulai dari penamaan proses, variabel-variabel yang digunakan dan lengkap dengan tipe datanya
- B. Formulasinya jelas jika ada, proses demi proses seolah bercerita dan memberikan intruksi yang sangat informatif, pada khususnya kepada *programmer*, *programmer* tahu apa yang harus dikerjakan, tidak perlu banyak berkhayal dan mengorek informasi lebih dalam tentang variabel, ada informasi bahwa formulasi dan hasil akhir akan dicetak atau tidak

- C. Proses demi proses seolah bercerita dan memberikan intruksi yang sangat informatif
  - D. Memberikan daya khayal yang tinggi kepada *programmer*
- 8) Perhatikan *Pseudocode* berikut:

```
1 MULAI
2 Kelas MencariKecepatandanPercepatan
3 Deklarasikan variabel
4 Lakukan Inisisasi awal
5 Hitung Kecepatan
6 Hitung Percepatan
7 Tampilkan kecepatan
8 Tampilkan percepatan
9 Selesai
```

*Pseudocode* di atas adalah masuk dalam kategori *pseudocode*....

- A. Ideal
  - B. Tidak ideal
  - C. Tidak jelas
  - D. Jawaban B dan C benar
- 9) “Untuk mendapatkan Nilai Akhir dari Mahasiswa adalah Nilai UTS ditambah dengan Nilai UAS kemudian dibagi dengan 2, baru kemudian bisa menentukan Nilai Grade Mahasiswa”.  
Formulasi yang paling tepat untuk mendapatkan Nilai Akhir Mahasiswa dengan *pseudocode* adalah....
- A. ...  

```
NilaiAkhir ← NilaiUTS + NilaiUAS /2
...
...
```
  - B. ...  

```
Nilai ← NilaiUTS + NilaiUAS
NilaiAkhir ← Nilai/2
...
...
```

C. ...

```
Pembagi ← 2
Nilai ← NilaiUTS + NilaiUAS
NilaiAkhir ← Nilai/Pembagi
...
...
```

D. ...

```
NilaiAkhir ← (NilaiUTS + NilaiUAS)/2;
...
...
```

10) Yang perlu diperhatikan dalam membuat *pseudocode* adalah....

- A. Ketersampaian informasi dan semua proses walaupun bertele-tele.
- B. Singkat padat dan tak bertele-tele gaya bahasanya dan sangat mudah di transformasi dalam bahasa pemrograman apapun.
- C. Boleh panjang yang penting informasinya bisa dicerna.
- D. Yang penting rapi dan sopan bahasanya

Cocokkanlah jawaban Anda dengan Kunci Jawaban Tes Formatif 2 yang terdapat di bagian akhir modul ini. Hitunglah jawaban yang benar. Kemudian, gunakan rumus berikut untuk mengetahui tingkat penguasaan Anda terhadap materi Kegiatan Belajar 2.

$$\text{Tingkat penguasaan} = \frac{\text{Jumlah Jawaban yang Benar}}{\text{Jumlah Soal}} \times 100\%$$

Arti tingkat penguasaan: 90 - 100% = baik sekali

80 - 89% = baik

70 - 79% = cukup

< 70% = kurang

Apabila mencapai tingkat penguasaan 80% atau lebih, Anda dapat meneruskan dengan modul selanjutnya. **Bagus!** Jika masih di bawah 80%, Anda harus mengulangi materi Kegiatan Belajar 2, terutama bagian yang belum dikuasai.

## Kunci Jawaban Tes Formatif

### Tes Formatif 1

- 1) A
- 2) B
- 3) C
- 4) D
- 5) C
- 6) D
- 7) D
- 8) D
- 9) D
- 10) B

### Tes Formatif 2

- 1) D
- 2) C
- 3) B
- 4) C
- 5) B
- 6) A
- 7) D
- 8) B
- 9) A
- 10) D

## Glosarium

- Bahasa R : pemrograman dan perangkat lunak untuk analisis statistika dan grafik. R dibuat oleh Ross Ihaka dan Robert Gentleman di Universitas Auckland, Selandia Baru, dan kini dikembangkan oleh R Development Core Team, di mana Chambers merupakan anggotanya. R dinamakan sebagian setelah nama dua pembuatnya (*Robert Gentleman* dan *Ross Ihaka*), dan sebagian dari permainan nama dari S.
- Debugging* : Sebuah metode yang dilakukan oleh para pemrogram dan pengembang perangkat lunak untuk mencari dan mengurangi bug, atau kerusakan di dalam sebuah program komputer atau perangkat keras sehingga perangkat tersebut bekerja sesuai dengan harapan  
Sumber: <https://www.dictio.id/t/seberapa-penting-kemampuan-proses-debugging-dalam-pembuatan-program/13292>
- Indentasi : Indentasi adalah penulisan paragraf yang agak menjorok masuk ke dalam. Indentasi pada umumnya digunakan jika Anda merespon pesan sebelumnya. Untuk membuat indentasi, Anda dapat menambahkan tanda titik dua (:) di awal baris/paragraf. Jika Anda menambah tanda titik dua (:) lagi, maka paragraf akan semakin menjorok.  
Sumber: <https://brainly.co.id/tugas/12100206>
- Independen* : bebas, merdeka, berdiri sendiri, swadaya, swakarsa, atau swakarya.  
Sumber: <https://id.wikipedia.org/wiki/Independen>
- Matlab : sebuah lingkungan komputasi numerikal dan bahasa pemrograman komputer generasi keempat. Dikembangkan oleh The MathWorks, MATLAB memungkinkan manipulasi matriks, pem-plot-an

fungsi dan data, implementasi algoritme, pembuatan antarmuka pengguna, dan pengantarmuka-an dengan program dalam bahasa lainnya. Meskipun hanya bermuansa numerik, sebuah kotak kakas (toolbox) yang menggunakan mesin simbolik MuPAD, memungkinkan akses terhadap kemampuan aljabar komputer. Sebuah paket tambahan, Simulink, menambahkan simulasi grafis multirancangan dan Desain Berdasarkan Model untuk sistem terlekat dan dinamik.

- Microsoft Office Excel : program aplikasi lembar kerja yang dibuat dan didistribusikan oleh Microsoft Corporation yang dapat dijalankan pada Microsoft Windows dan Mac OS. Aplikasi ini merupakan bagian dari Microsoft Office System. Aplikasi ini memiliki fitur kalkulasi dan pembuatan grafik yang, dengan menggunakan strategi marketing Microsoft yang agresif, menjadikan Microsoft Excel sebagai salah satu program komputer yang populer digunakan di dalam komputer mikro hingga saat ini. Bahkan, saat ini program ini merupakan program spreadsheet paling banyak digunakan oleh banyak pihak, baik di platform PC berbasis Windows maupun platform Macintosh berbasis Mac OS, semenjak versi 5.0 diterbitkan pada tahun 1993.  
Sumber:  
[https://id.wikipedia.org/wiki/Microsoft\\_Excel](https://id.wikipedia.org/wiki/Microsoft_Excel)
- Microsoft Powerpoint : Program komputer untuk presentasi yang dikembangkan oleh Microsoft di dalam paket aplikasi kantor mereka, Microsoft Office, selain Microsoft Word, Excel, Access dan beberapa program lainnya. PowerPoint berjalan di atas komputer PC berbasis sistem operasi Microsoft Windows dan juga Apple Macintosh yang menggunakan sistem operasi Apple Mac OS, meskipun pada awalnya aplikasi ini berjalan di atas sistem operasi Xenix. Aplikasi ini sangat

	banyak digunakan, apalagi oleh kalangan perkantoran dan pebisnis, para pendidik, siswa, dan trainer. Dimulai pada versi Microsoft Office System 2003, Microsoft mengganti nama dari sebelumnya Microsoft PowerPoint saja menjadi Microsoft Office PowerPoint. Lalu, pada Office 2013, namanya cukup disingkat PowerPoint. Versi terbaru dari PowerPoint adalah versi 15 (Microsoft Office PowerPoint 2013), yang tergabung ke dalam paket Microsoft Office 2013.
Microsoft Word	: Perangkat lunak pengolah kata (word processor) andalan Microsoft. Pertama diterbitkan pada 1983 dengan nama Multi-Tool Word untuk Xenix, versi-versi lain kemudian dikembangkan untuk berbagai sistem operasi, misalnya DOS (1983), Apple Macintosh (1984), SCO UNIX, OS/2, dan Microsoft Windows (1989). Setelah menjadi bagian dari Microsoft Office System 2003 dan 2007 diberi nama Microsoft Office Word. Di Office 2013, Namanya cukup dinamakan Word.
Random Skalar	: acak atau tidak teratur : Konsep yang digunakan dalam matematika dan fisika. Konsep yang dipakai dalam fisika adalah versi yang lebih konkret dari ide yang sama dalam matematika.skalar adalah kuantitas yang bisa dijelaskan dengan suatu angka (entah itu tanpa dimensi, atau dalam suatu kuantitas fisika). Kuantitas skalar mempunyai besar (magnitudo), tetapi tidak mempunyai arah dan oleh karena itu berbeda dengan vektor.
	Sumber: <a href="https://id.wikipedia.org/wiki/Skalar">https://id.wikipedia.org/wiki/Skalar</a>
Sort & Filter	Fungsi Sort pada excel berfungsi untuk mengurutkan data mulai dari nilai yang terendah hingga tertinggi atau sebaliknya. Sementara fungsi Filter pada excel berfungsi untuk memilih data tertentu yang ingin ditampilkan, tanpa perlu menghapus sisanya.

Spss : program komputer yang dipakai untuk analisis statistika. Sejak tanggal 28 Juli 2009, SPSS disebut sebagai PASW (*Predictive Analytics SoftWare*)

## Daftar Pustaka

Dennis, Wixom, Roth, (2009). *System analysis and design*. Wiley. USA.

Kumad, D. (2009). *Overview of system analysis & design*. xxx.

Langer, A.M. (2008). *Analysis and design of information systems*. New York: Springer.

Purbasari, I.Y. *Desain & analisis algoritma*. Yogjakarta: Graha Ilmu.

<https://brainly.co.id/tugas/12100206>

<https://id.wikipedia.org/wiki/Independen>

<https://id.wikipedia.org/wiki/Skalar>

[https://id.wikipedia.org/wiki/Microsoft\\_Excel](https://id.wikipedia.org/wiki/Microsoft_Excel)

## MODUL 3

# Praktikum 1

Kani, M.Kom.



### PENDAHULUAN

---

Praktikum Modul Algoritma dan Pemrograman terdiri dari 3 Modul, yaitu: Modul 3, 6, dan 9. Untuk Modul 3 akan berisi panduan instalasi dan praktikum materi Java.

Setelah mempelajari Praktikum 1, Mahasiswa akan dapat:

1. mengunjungi laman software *draw.io* untuk unduh versi desktop;
2. mengunduh *software draw.io*;
3. meninstalasi *draw.io*;
4. mengenali menu-menu yang terdapat pada *software draw.io*;
5. membuat *flowchart* menggunakan *draw.io*;
6. mendapatkan dan mengunjungi laman *software IDE Eclipse* untuk mengunduh *software IDE Eclipse*;
7. menginstalasi *IDE Eclipse*;
8. konfigurasi *IDE Eclipse*;
9. menggunakan *IDE Eclipse*;
10. mempraktekkan Struktur Java;
11. mempraktekkan penggunaan Tipe Data.

**PRAKTIKUM 1.1**

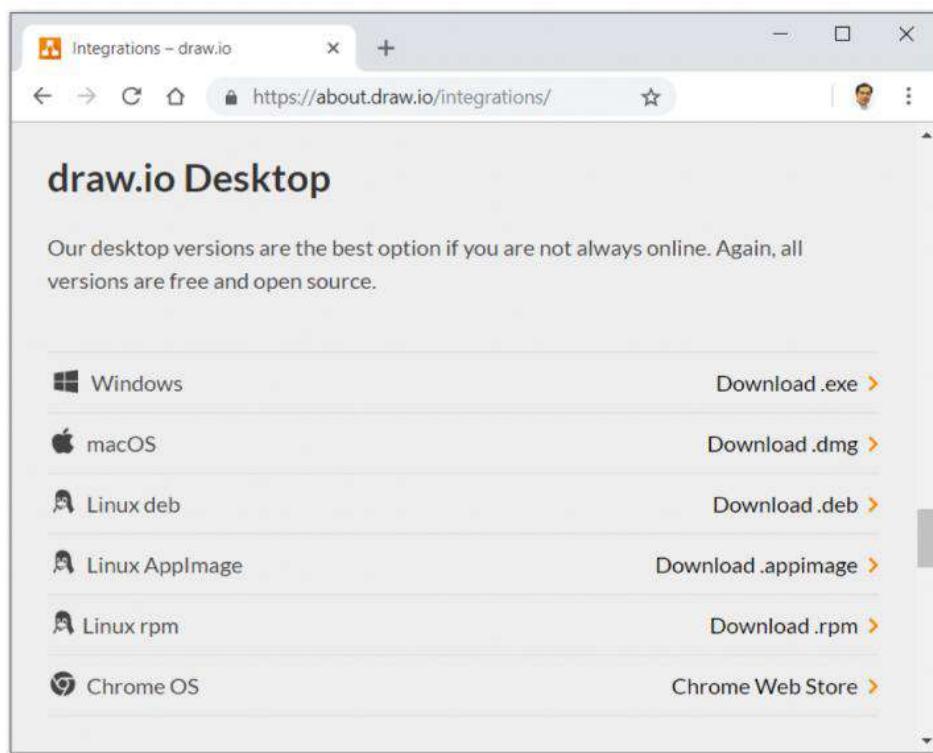
## Instalasi *Software draw.io* dan membuat *flowchart* di *draw.io*

*S*oftware *draw.io* adalah *software* atau aplikasi untuk menggambar diagram secara *online*, bahkan *software* tersebut sudah tersedia dalam bentuk versi desktop dengan beberapa keterbatasan. Untuk yang versi *online* mendukung mendukung HTML5, IE dari versi 6 sampai 8, iOS dan Android, dan tentu diperlukan tersedianya koneksi *internet* untuk dapat menggunakan fasilitas tersebut. Proses penyimpanan berkas/dokumen, dapat dilakukan secara *online* maupun *offline*. Tersedia dua opsi untuk penyimpanan secara *online* yakni menggunakan: Google Drive, OneDrive atau Dropbox.

### A. BAGAIMANA MENDAPAT *SOFTWARE DRAW.IO*

*Draw.io* disediakan secara bebas dan gratis. Software ini tersedia untuk penggunaan secara *online* maupun *offline* (desktop). Alamat lamannya adalah <https://www.draw.io>. Dengan mengakses laman tersebut, maka pada saat itu juga sudah bisa membuat diagram atau *flowchart*. Pada modul ini hanya akan dijelaskan dan mempraktekkan yang versi desktop saja.

Untuk mengunduh yang versi desktop dapat mengakses laman <https://about.draw.io/integrations/>.



Gambar 3.1.  
Versi Desktop yang Tersedia

Pada laman *draw.io* disediakan beberapa versi platform, seperti versi *Windows*, *macOS*, *Linux deb*, *Linux AppImage*, *Linux rpm*, dan *Chrome OS*. Khusus pada modul ini kita akan mengunduh versi *Windows* dan semua praktik yang akan kita lakukan dilakukan pada Sistem Operasi Windows.

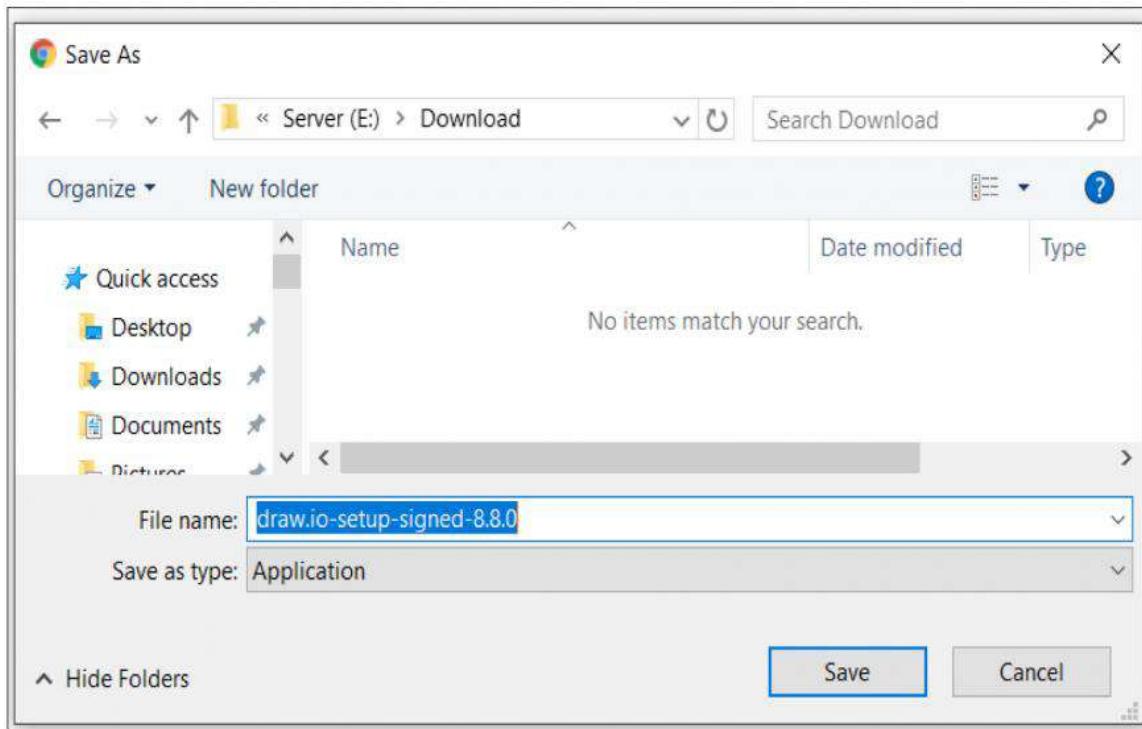
Untuk mengunduh *draw.io* versi Windows, langkah-langkah berikut ini:

1. Pada laman <https://about.draw.io/integrations/> scroll ke bawah hingga *draw.io Desktop* tampak dengan jelas.
2. Untuk mengunduh versi Windows, silahkan klik pada bagian bertuliskan *Download.exe*.



Gambar 3.2.  
Link Download Versi Windows

3. Pada windows *Save As* silahkan memberikan nama atau membiarkan dengan nama bawaannya (*default*) *draw.io-setup-signed-8.8.0.exe*.

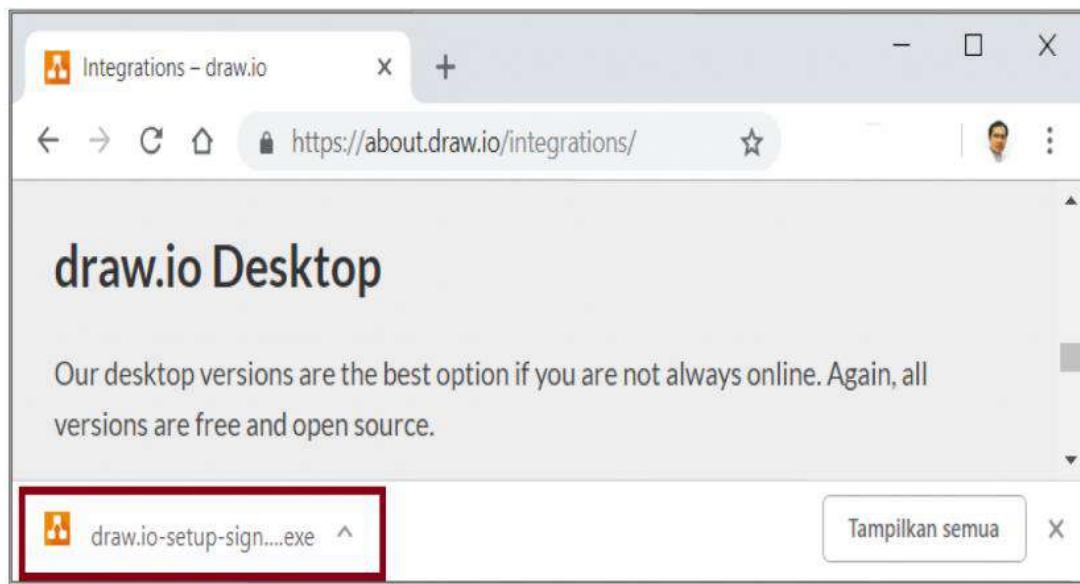


Gambar 3.3.  
Window Save As

Catatan:

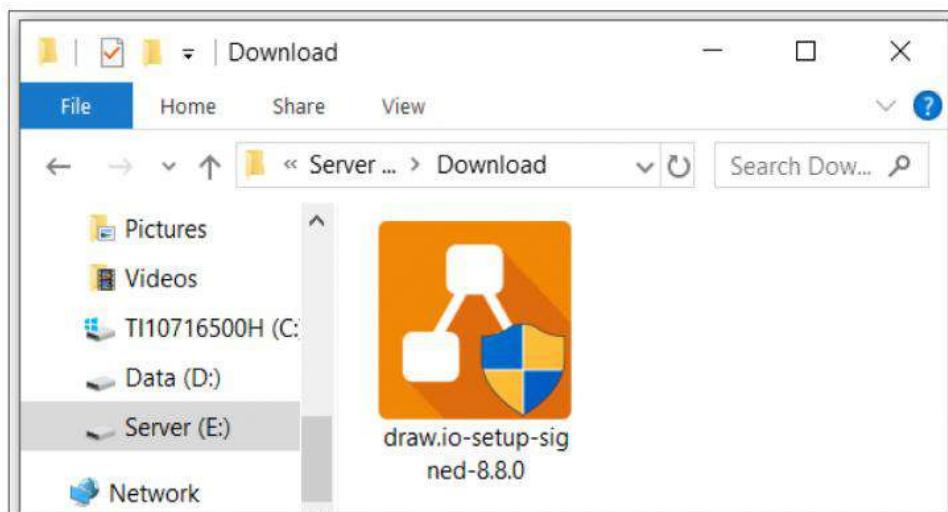
Akan sangat berbeda dalam cara mengunduh setiap browser yang digunakan, pada modul ini menggunakan aplikasi browser Chrome.

4. Klik tombol ( **Save** ) untuk memulai proses pengunduhan.
5. Perhatikan pada gambar di bawah, pada kotak sudut kiri bawah dan menandakan bahwa proses pengunduhan selesai.



Gambar 3.4.  
Proses Unduh Selesai

6. Silahkan cek hasil unduhan pada folder dimana unduhan dialamatkan.



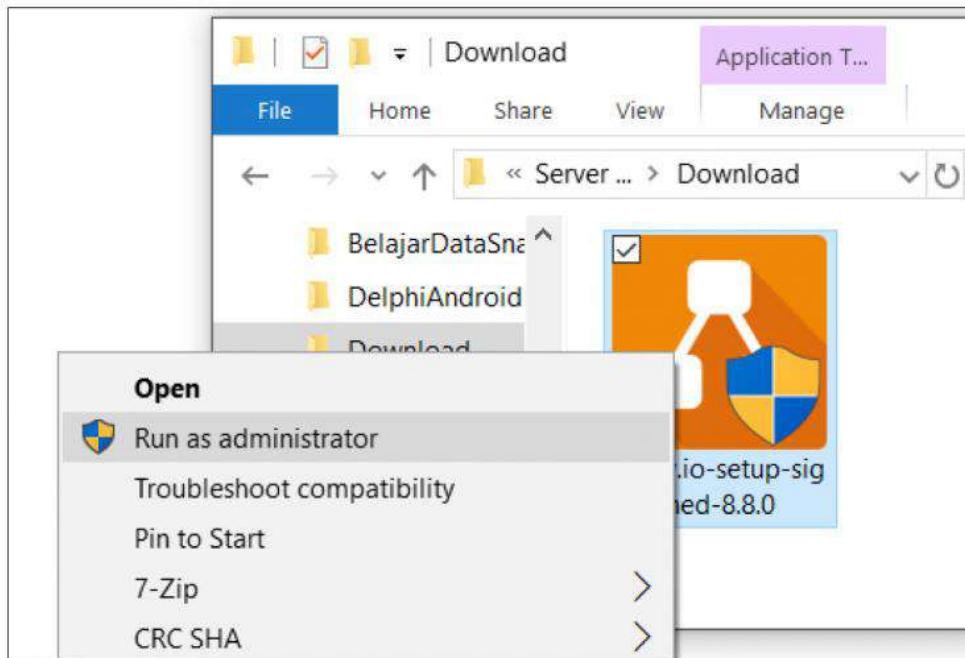
Gambar 3.5.  
Hasil Unduh *draw.io*

## B. INSTALASI *DRAW.IO*

Untuk memulai instalasi, lakukan langkah-langkah berikut:

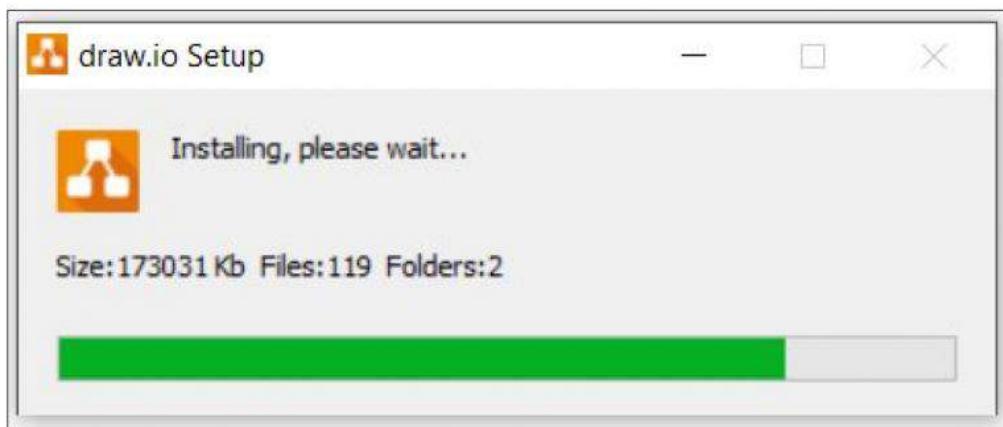
1. Pastikan file *draw.io* dengan nama file *draw.io-setup.signed-8.8.0.exe* sudah siap untuk dieksekusi.

2. Klik kanan file *draw.io-setup.signed-8.8.0.exe* dan pilih menu *Run as administrator*.



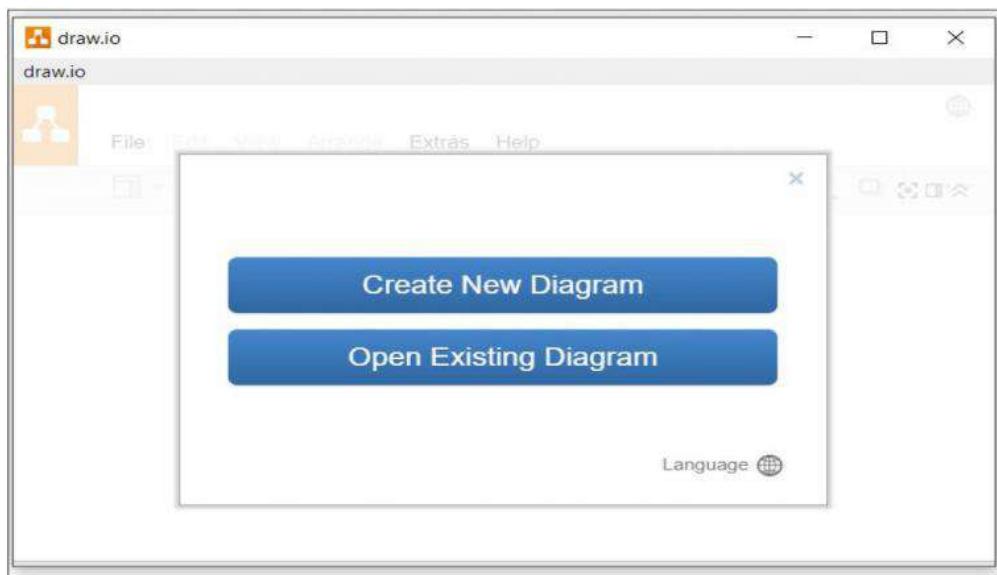
Gambar 3.6.  
Mulai Menginstalasi *draw.io*

3. Pada window *User account control* pilih tombol *Yes*, dan proses instalasi akan berjalan.



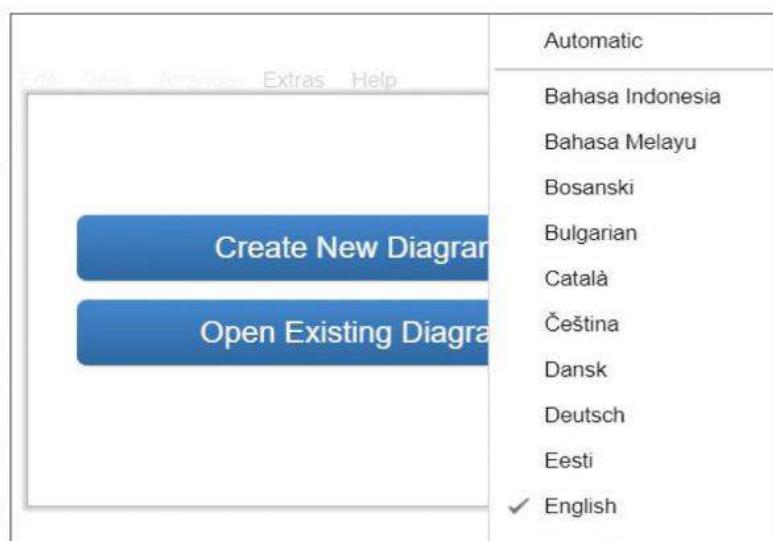
Gambar 3.7.  
Proses Instalasi *draw.io*

4. Tunggu sampai selesai hingga jendela aplikasi *draw.io* muncul seperti Gambar 3.8.:



Gambar 3.8.  
Pertama Kali Membuka Software *draw.io*

5. Untuk membuat dokumen *flowchart* baru, silahkan klik tombol *Create New Diagram*. Atau untuk membuka dokumen lama bisa menggunakan tombol *Open Existing Diagram*.
6. Dan untuk memilih bahasa yang digunakan dalam aplikasi *draw.io* maka bisa klik menu *Language* ( ). Tersedia pilihan Bahasa Indonesia.



Gambar 3.9.  
Memilih Bahasa yang Digunakan  
Pada *draw.io*

### C. MEMULAI *DRAW.IO*

Pada sub-bagian sebelumnya, panduan instalasi telah dijelaskan dan memastikan bahwa aplikasi telah berhasil diinstalasi. Sekarang kita akan memulai menggunakan *draw.io*.

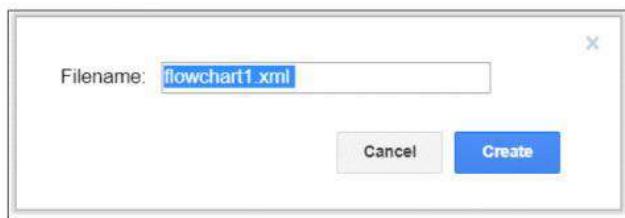
Untuk memulai *draw.io* ikuti langkah-langkah berikut:

1. Klik dua kali *shortcut draw.io* pada desktop.
2. Setelah muncul window pemilihan fitur (*Create New Diagram* dan *Open Existing Diagram*), kita pilih opsi “*Create New Diagram*” untuk membuat file baru.



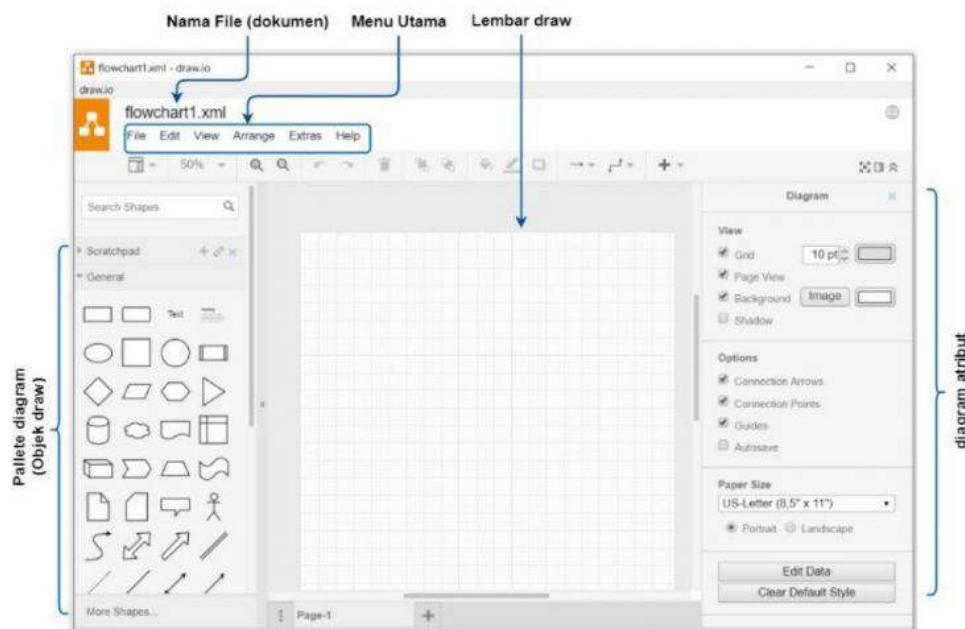
Gambar 3.10.  
Pilihan Opsi Buka *draw.io*

3. Pada window selanjutnya adalah fasilitas untuk pemberian nama, berilah nama file dengan “*flowchart1.xml*” lalu kemudian klik tombol *Create*.



Gambar 3.11.  
Memberikan Nama File *draw.io*

4. Pada window berikutnya adalah disuguhkan sebuah dokumen baru dengan format standar dari *draw.io*.



Gambar 3.12.  
Tampilan Utama Desktop *draw.io*

#### D. MENGENALI MENU-MENU DARI *DRAW.IO*

Untuk kategori menu utama, *draw.io* mempunyai 6 menu utama, yaitu: *File*, *Edit*, *View*, *Arrange*, *Extras*, dan *Help* dengan fungsi masing-masing.

##### 1. Menu File

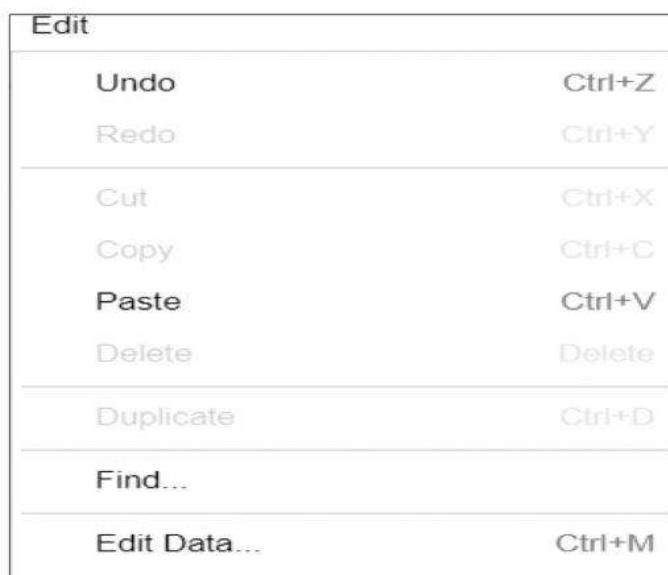
Menu File adalah menu yang berhubungan dengan pengaturan file dokumen, misalnya *New*, *Save*, *Save as* dan lain-lain.



Gambar 3.13.  
Menu *File*

## 2. Menu *Edit*

Menu *Edit* berfungsi untuk mengedit (melakukan perubahan dengan cara menambah, menghapus, menyalin, menempelkan, mengatur pemunculan objek, dan lain-lain).



Gambar 3.14.  
Menu *Edit*

### 3. Menu View

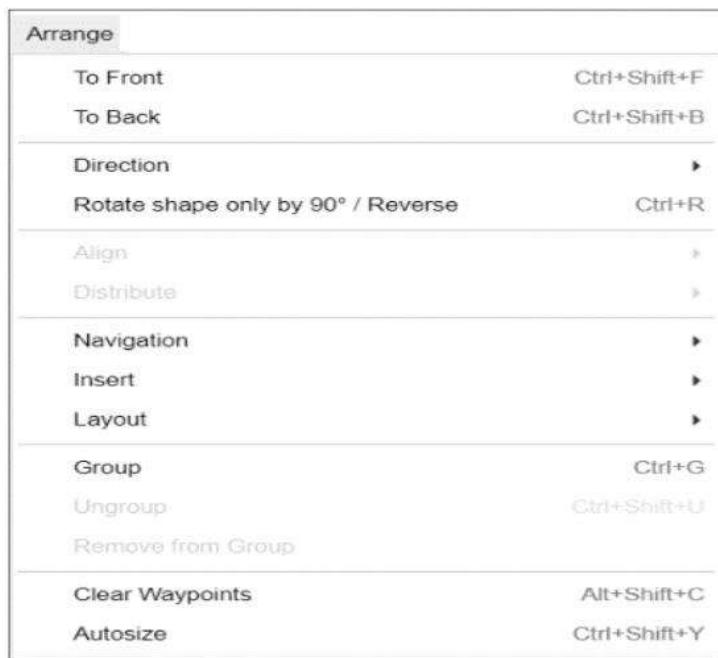
Menu *View* digunakan untuk mengatur tampilan dari *draw.io*, misalnya *Page View* untuk mengatur tampilan halaman, *Grid* untuk memberikan tampilan pada lembaran kerja, *Zoom In* dan *Zoom Out* adalah digunakan untuk memperbesar atau memperkecil tampilan halaman. Beberapa fasilitas di menu *View* memiliki *hotkey* yang bisa digunakan untuk mempercepat pekerjaan, misalnya sub menu *Grid* dengan *hotkey* perpaduan penekanan pada *keyboard* yaitu *Ctrl+Shift+G*.



Gambar 3.15.  
Menu View

### 4. Menu Arrange

Beberapa sub menu pada menu *Arrange* adalah seperti *toFront* untuk membuat suatu objek berada di atas objek yang lain, *toBack* adalah kebalikan dari fungsi *toFront*, *Layout*, *Autosize* dan lain-lain.



Gambar 3.16.  
Menu *Arrange*

## E. MEMBUAT FILE BARU, MENYIMPAN FILE, KELUAR DARI APLIKASI, MEMBUKA DAN MENYUNTING FILE

Sekarang kita akan melakukan beberapa praktik untuk memudahkan nantinya dalam beberapa operasi *file*, misalnya membuat *file* baru, menyimpan *file*, keluar dari aplikasi, membuka *file* dan menyuntingnya.

### 1. Membuat *File* Baru

Untuk membuat *file* baru pada *draw.io* silahkan mengikuti langkah-langkah berikut:

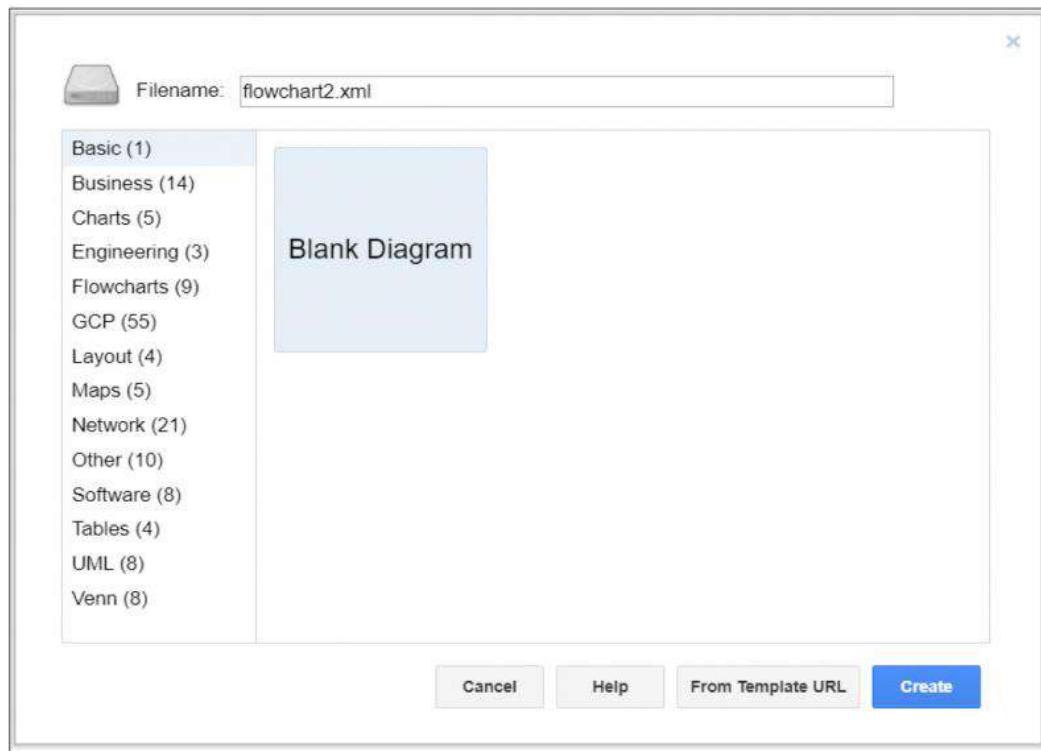
- Klik menu *File* → *New* (atau bisa dengan menekan shortcut *Ctrl+N*).
- Setelah muncul *window* opsi , klik tombol *Create New Diagram*.



Gambar 3.17  
Opsi Buka File

c. Pada Window Filename, isi dan pilih sebagai berikut:

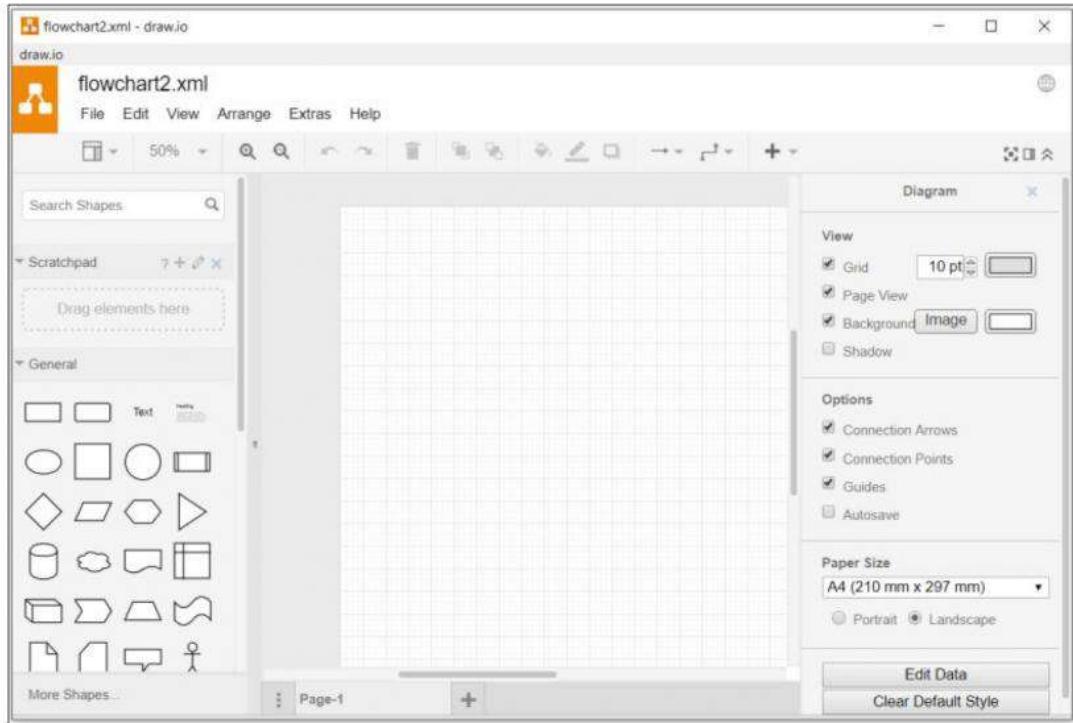
- ✓ Filename : *flowchart2.xml*
- ✓ Diagram (opsi) : *Basic(1)*
- ✓ Opsi *flowchart* : *Blank Diagram*



Gambar 3.18  
Memilih Diagram yang Akan Digunakan

Kemudian klik tombol *Create*.

- d. File *flowchart2.xml* sudah siap digunakan.



**Gambar 3.19.**  
**Dokumen Diagram Blank**

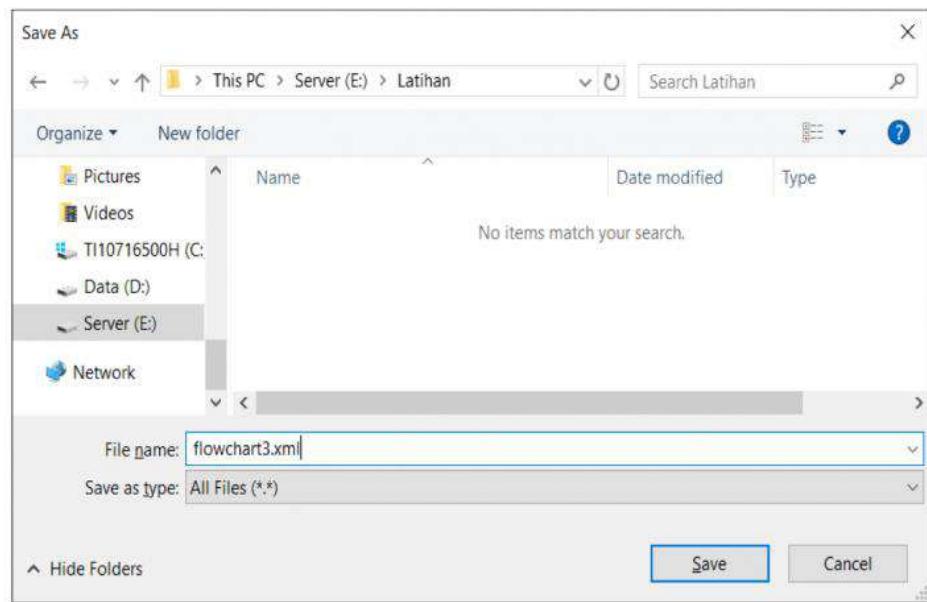
2. Menyimpan *File*

*File* yang sudah dilakukan pengeditan atau diisi dengan diagram objek, dapat disimpan dengan cara seperti berikut:

- Klik *File*
- Sorot menu *Save* atau dengan *shortcut Ctrl+S*

Jika hendak menyimpannya dengan nama lain, maka langkahnya sebagai berikut:

- Klik menu *File*
- Sorot menu *Save as...* atau dengan *shortcut Ctrl+Shift+S*
- Pada window *Save as* isi kotak *File name* dengan *flowchart3.xml* (untuk foldernya Anda bisa menyimpan sesuai keinginan Anda).



Gambar 3.20  
Proses Simpan File

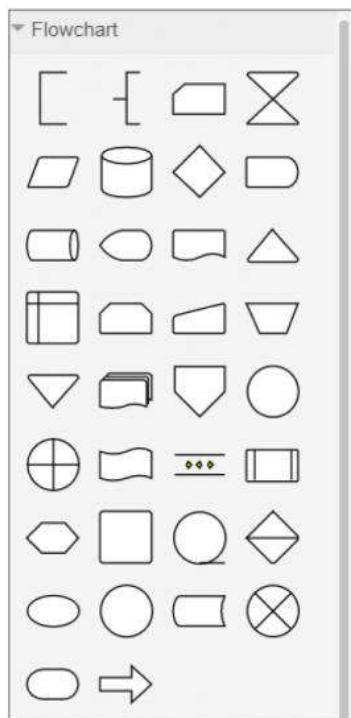
- d. Proses terakhir adalah klik tombol *Save*.
- 3. Keluar dari aplikasi draw.io  
Adapun cara untuk keluar dari aplikasi **draw.io** adalah sebagai berikut:
  - a. Klik menu *File*
  - b. Kemudian sorot dan klik menu *Close*.

## F. MENAMBAHKAN OBJEK KE DALAM LEMBARAN **DRAW.IO**

Pada praktik berikutnya adalah kita mencoba untuk membuat *file* baru dan kemudian mencoba untuk menambahkan objek *flowchart* kedalam lembaran kerja **draw.io**.

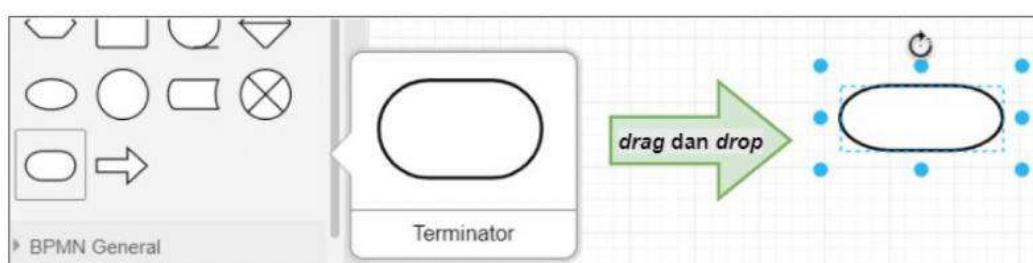
Mari praktekkan dengan langkah-langkah berikut:

1. Buka aplikasi **draw.io**;
2. Buat file baru dengan nama *Flowchart4.xml* (lihat kembali poin E pada bagian Cara membuat file);
3. Pastikan lembaran *draw.io* terbuka.
4. Pastikan *Pallete Flowchart* juga terbuka seperti Gambar 3.21. berikut:



Gambar 3.21.  
Objek (Pallete) Flowchart

5. Misalnya kita mau menambahkan objek *terminator*, sorot ke bagian objek *terminator*, lihat Gambar 3.22 berikut:



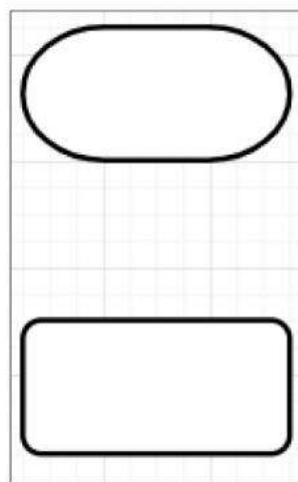
Gambar 3.22.  
Mendrop Objek Terminator ke Halaman Kerja

6. Perlakuan yang sama ketika mau menambahkan objek yang lain ke dalam lembaran **draw.io**.

## G. MENGHUBUNGKAN OBJEK DENGAN OBJEK LAINNYA DAN MENAMBAHKAN TEKS KE DALAM OBJEK

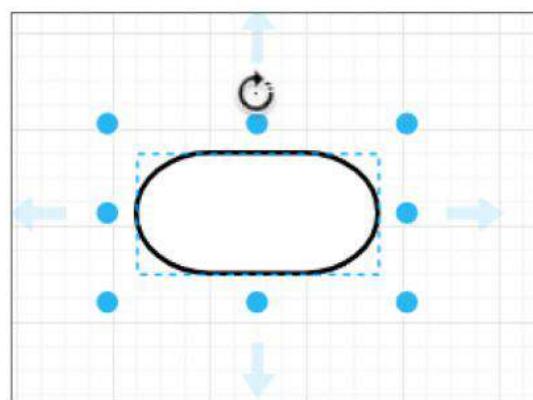
Untuk menghubungkan objek yang satu ke objek yang lainnya dengan cara sebagai berikut:

1. Buka kembali file *Flowchar4.xml*.
2. Pastikan terdapat 2 objek dalam lembaran *draw.io*. anggaplah objek *terminator* dan objek proses (*process*).



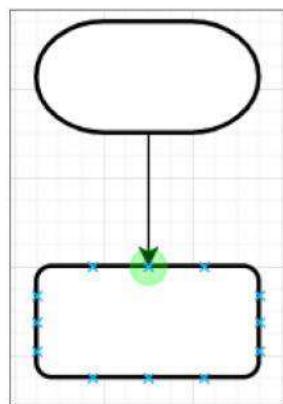
Gambar 3.23.  
Objek yang Belum Terhubung

3. Aktifkan objek *terminator* dengan cara klik tepat pada objek *terminator* yang ada pada lembaran *draw.io*.



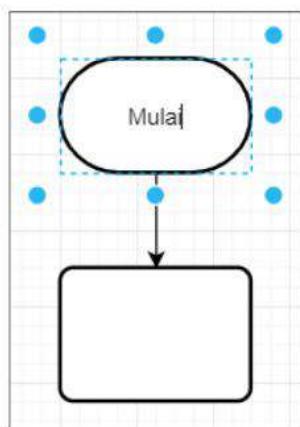
Gambar 3.24.  
Aktif pada Objek

4. Untuk menghubungkan objek *terminator* ke objek proses, klik tanda panah pada objek bagian bawah:



Gambar 3.25.  
Menghubungkan Objek

- Untuk menambahkan teks ke dalam objek, yaitu dengan melakukan klik dua kali pada objek, misalnya pada objek *terminator* mau ditambahkan tulisan *Mulai*.

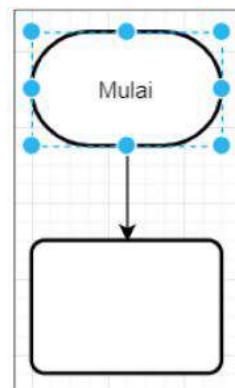


Gambar 3.26.  
Menambahkan Teks pada Objek

## H. MENGECLIKAN DAN MEMBESARKAN OBJEK

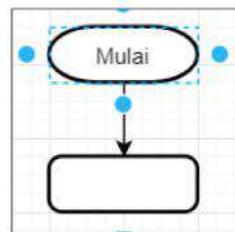
Bagaimana jika ingin mengecilkan dan membesarkan objek? Berikut langkah-langkahnya:

- Pilih objek yang ingin dikecilkan atau dibesarkan, misalnya memilih objek *terminator*.
- Setelah objek terpilih akan muncul 8 bulatan kecil berwarna biru disekeliling objek, perhatikan Gambar 3.27.



Gambar 3.27.  
Memilih Objek yang Akan Dikecilkan

3. Jika *pointer mouse* diarahkan ke salah satu bulatan kecil berwarna biru tersebut, maka *pointer mouse* akan berubah menjadi (↗) jika Anda mengarahkan ke posisi bulatan atas kanan, artinya objek siap dibesarkan dan dikecilkan secara diagonal.



Gambar 3.28.  
Objek Terminator yang Sudah Dikecilkan

4. Sisanya silahkan dipraktekkan sendiri.

Untuk memperkaya pengetahuan dan pengalaman Anda dalam penggunaan software draw.io ini, maka Anda disarankan untuk melihat tutorial draw.io yang tersedia pada channel di youtube. Beberapa sumber yang tersedia adalah:

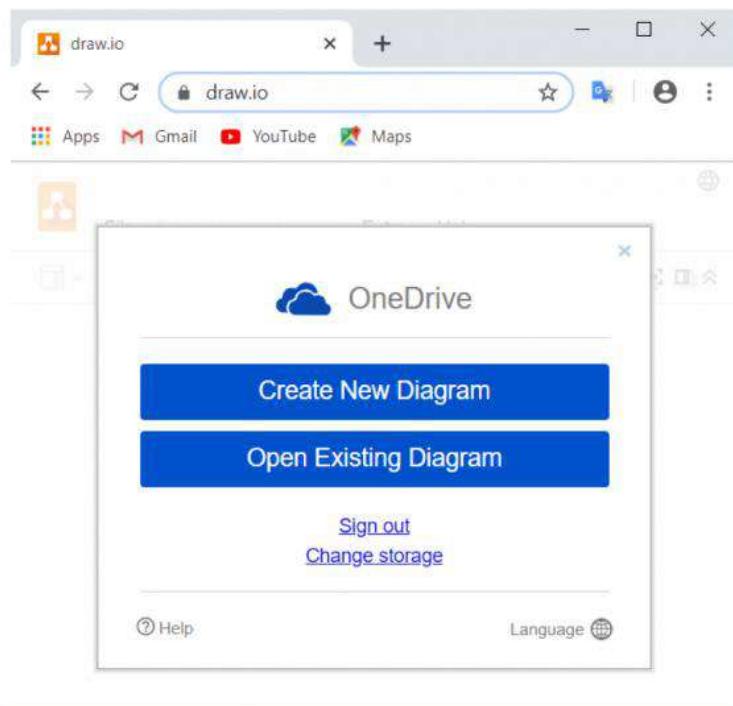
- a. <https://www.youtube.com/watch?v=DYSPEkvTWig>
- b. <https://www.youtube.com/watch?v=64MaQYyAN2w>
- c. <https://www.youtube.com/watch?v=BMDMf1D4K6Y>

## I. BEKERJA DENGAN **DRAW.IO** VERSI ONLINE (WEB)

*Software draw.io* juga menyediakan fasilitas untuk buat *flowchart* secara langsung tanpa instalasi. Syaratnya adalah harus bisa *online* ke internet.

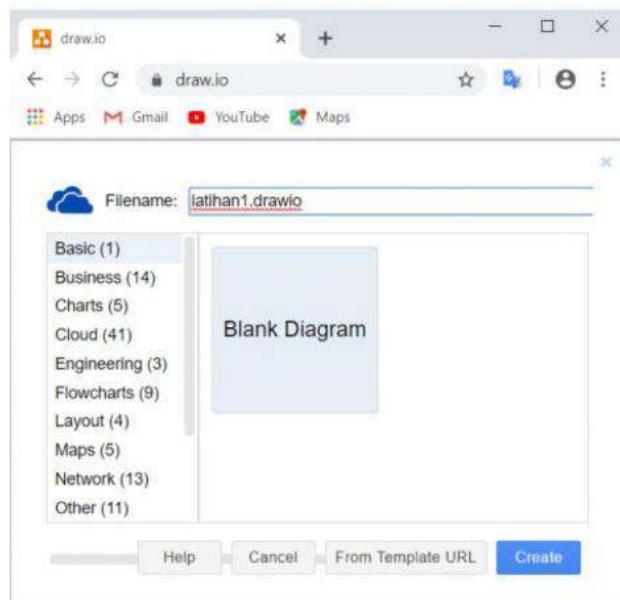
Berikut laman untuk bekerja di *draw.io* secara *online* adalah <https://www.draw.io/>. Untuk mempraktekkan langkah-langkah sebagai berikut:

1. Buka *web browser*, di Modul ini menggunakan browser Chrome.
2. Ketikkan kolom alamat isi alamat browser dengan <https://www.draw.io/>, dan kemudian *enter*.



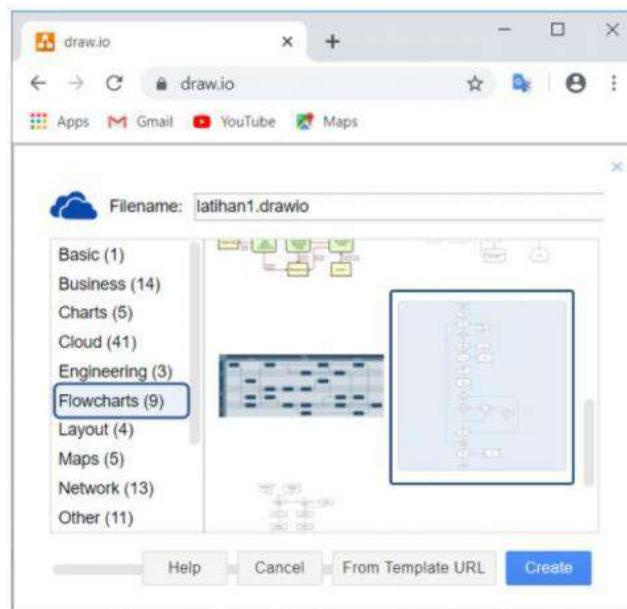
Gambar 3.29.  
Web *draw.io*

3. Klik tombol “*Create New Diagram*”.
4. Pada kolom “*File Name*” isi dengan latihan1.drawio.



Gambar 3.30.  
Menu Pengisian Identitas File dan Pilihan Objek Gambar

5. Untuk ListBox sebelah kiri, pilih "Flowcharts(9)", dan ListBox sebelah kanan, pilih yang sudah ditandai



Gambar 3.31.  
Objek *Flowcharts*

6. Klik tombol “Create” dan perintah lainnya sama dengan versi desktop.

**LATIHAN**

---

Untuk memperdalam pemahaman Anda mengenai materi di atas, kerjakanlah latihan berikut!

- 1) Carilah *software* melalui internet yang bisa digunakan untuk membuat *flowchart* baik bisa digunakan secara gratis maupun berbayar.

*Petunjuk Jawaban Latihan*

- 1) Beberapa alternatif *software* yang bisa digunakan untuk membuat *flowchart* adalah sebagai berikut:
  - Microsoft Visio: Sebuah *software* program diagram komersial untuk Microsoft Windows yang menggunakan grafik vektor untuk membuat diagram. Sangat direkomendasikan untuk membuat *flowchart* diagram.
  - Google Docs: *Software* ini disediakan secara *online* sehingga dimanapun anda bisa bekerja selama koneksi internet ada. Mengelola dokumen, *spreadsheet*, presentasi, dan survei.
  - Software Gliffy *Flowchart*: disediakan secara *online*. Hanya *drag* dan *drop* bentuk dan garis-garis dari *library* yang ada. Disediakan secara gratis, dan tidak perlu instalasi.
  - SmartDraw: Membuat *diagram*, *use case diagram*, *entity relationship diagram*, *UML diagram*, dan banyak lagi bisa dibuat beberapa menit dengan SmartDraw.
  - Creately: *Software flowchart* berbasis web untuk menggambar *Flowchart* yang bagus, *Workflow Diagram*, Proses *Flow Diagram* dan *Data Flow Diagram*.
  - Edraw Max.
  - LucidChart.
  - Cacoo.
  - Flowchart.com.
  - UML.



## RANGKUMAN

---

*Software draw.io* adalah *software* atau aplikasi untuk menggambar diagram secara *online*, bahkan *software* tersebut sudah tersedia dalam bentuk versi desktop dengan beberapa keterbatasan. Untuk yang versi *online* mendukung mendukung HTML5, IE dari versi 6 sampai 8, iOS dan Android, dan tentu yang Anda butuhkan adalah koneksi *internet*. Dan untuk penyimpanan berkas diberikan bisa disimpan secara *online* maupun *offline*.

*Draw.io* disediakan secara bebas dan disediakan secara online dan *offline* (desktop). Alamat lamannya adalah <https://www.draw.io>. Dengan mengakses laman di atas, maka pada saat itu juga kita sudah bisa membuat diagram atau *flowchart*. Namun dalam modul ini akan dijelaskan dan mempraktekkan yang versi Desktop. Dan untuk mengunduh versi desktop bisa mengunjungi laman <https://about.draw.io/integrations/>.

## PRAKTIKUM 1.2

# Instalasi *IDE Eclipse* dan membuat program *Halo Java*

Java adalah bahasa pemrograman yang sejak munculnya sudah sangat populer. Saat ini Java menjadi bahasa pemrograman yang paling diminati untuk membangun aplikasi *mobile*, *web*, maupun *enterprise*. Dalam pengembangan *software*, *programmer* Java tak lepas dari IDE (*Integrate Development Environment*).

Keuntungan menggunakan IDE adalah banyak fitur yang bisa digunakan untuk mempercepat pengembangan *software*, IDE pada Java sendiri sudah cukup banyak, misalnya *Netbeans*, *Eclipse*, *BlueJ*, *JBuilder* (Borland), *JDeveloper* (Oracle).

Pada Modul Praktikum ini akan digunakan IDE yang termasuk paling banyak digunakan yaitu Eclipse.

### A. IDE ECLIPSE

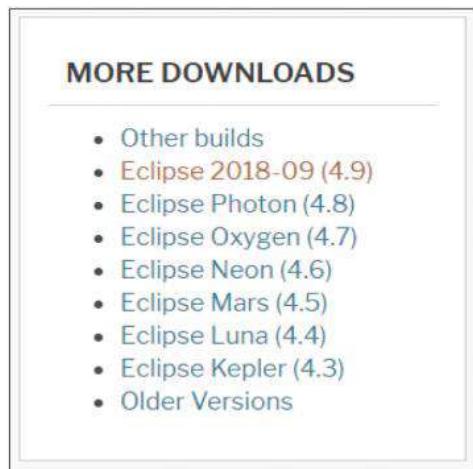
Eclipse adalah IDE yang digunakan untuk mengembangkan perangkat lunak yang bisa dijalankan diberbagai platform, Eclipse dapat berjalan pada platform *Microsoft Windows*, *Linux*, *Solaris*, *IX*, *HP-UX*, dan *Mac OS X*. Pada awal munculnya oleh *IBM Visual Age* diperuntukan untuk Java 4.0, seiring berjalannya waktu Eclipse mendukung pengembangan *software* berbasis bahasa pemrograman lainnya, seperti C/C++, Cobol, Phyton, Perl, PHP dan lain sebagainya. Eclipse dapat digunakan untuk aktivitas dalam siklus pengembangan perangkat lunak, seperti dokumentasi, ujicoba perangkat lunak, pengembangan web dan lain sebagainya.

Cukup banyak versi Eclipse yang sudah rilis yang biasanya akan selalu diperbarui dengan rilisnya versi Java terbaru. Pada modul ini akan kita gunakan Eclipse SimRel.

## B. DOWNLOAD IDE ECLIPSE

Untuk mendapatkan Eclipse versi SimRel Anda bisa mengikuti langkah-langkah berikut:

1. Kunjungi laman <https://www.eclipse.org/downloads/packages/installer>, lalu fokus pada bagian kanan laman tepat pada *MORE DOWNLOADS*



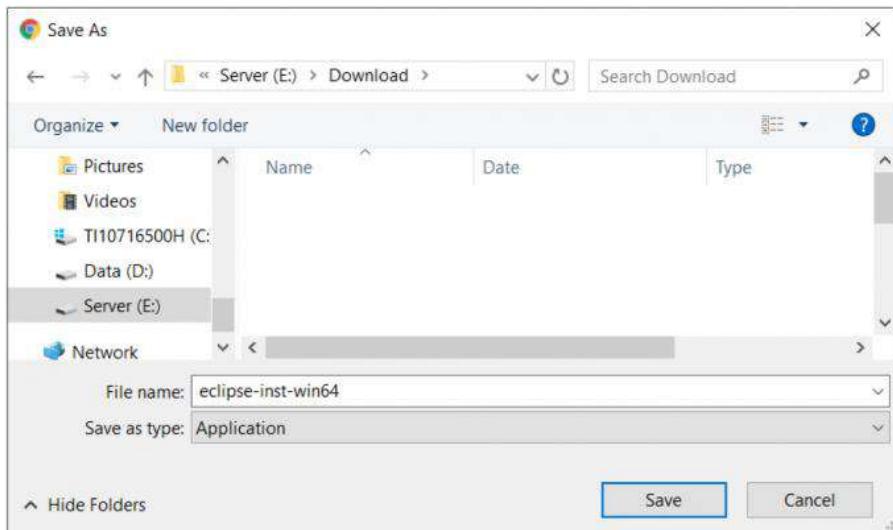
Gambar 3.29.  
Pilihan IDE Eclipse yang Ada di Web

2. Pada halaman laman berikutnya arahkan ke “*Get Eclipse SimRel 2018-09*” dan klik tombol “*Download 64 bit*”, mungkin akan sangat berbeda link pengunduhannya, apalagi jika terdapat versi Eclipse yang terbaru.



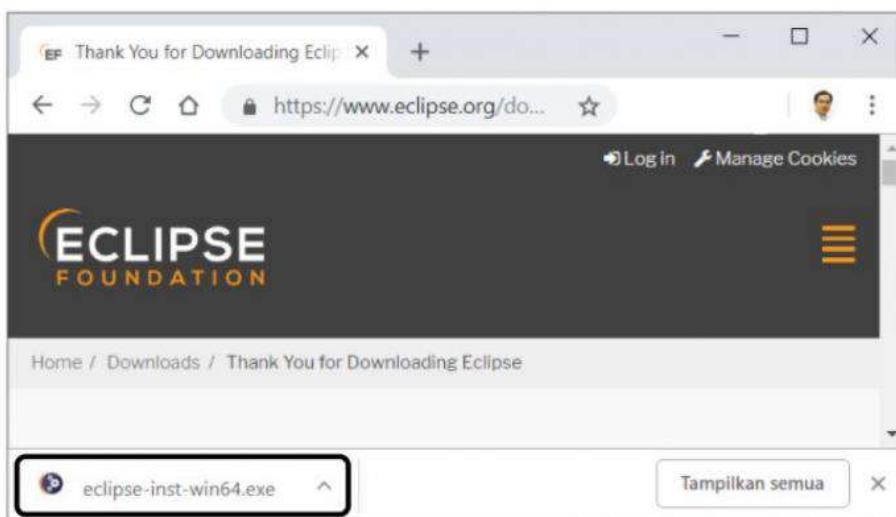
Gambar 3.30.  
Pilihan Versi IDE Eclipse yang Hendak Diunduh

3. Simpan pada folder yang Anda inginkan dan nama filenya adalah *eclipse-inst-win64.exe*, klik tombol “Save” (akan sangat berbeda jika menggunakan browser Chrome).



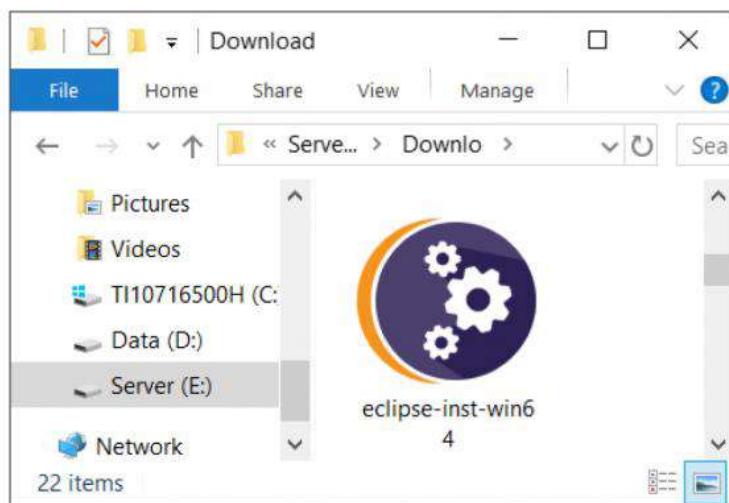
Gambar 3.31  
Menyimpan Unduhan IDE Eclipse

4. Tunggu hingga selesai terunduh dengan sempurna.



Gambar 3.32  
IDE Eclipse Berhasil Diunduh

5. Setelah terunduh dengan sempurna, maka kita akan mendapat 1 file bernama *eclipse-inst-win64.exe*.

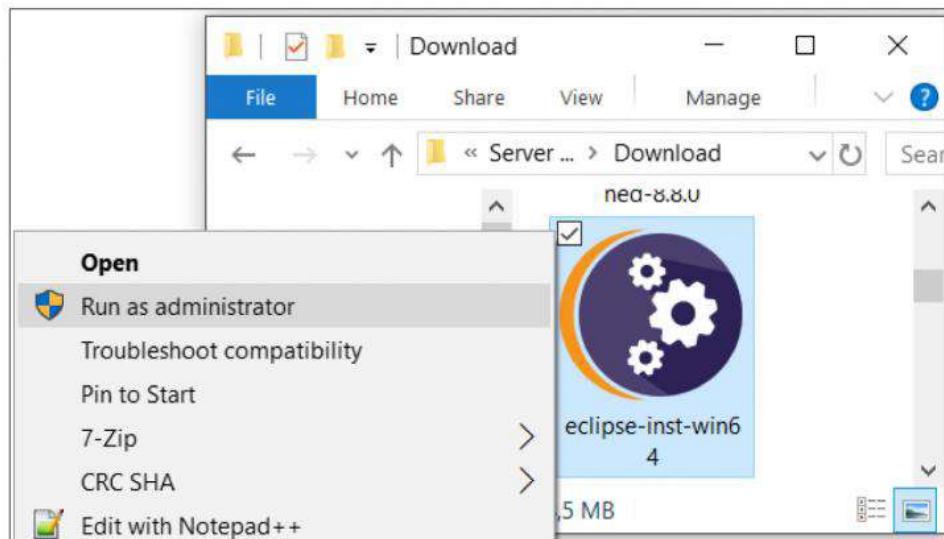


Gambar 3.33  
IDE Eclipse yang Sudah Terunduh

### C. INSTALASI IDE ECLIPSE

Untuk instalasi IDE Eclipse dengan langkah-langkah sebagai berikut:

1. Pastikan file *eclipse-inst-win64.exe* sudah siap, pastikan komputer Anda terkoneksi dengan internet.
2. Pastikan file *eclipse-inst-win64.exe* terpilih dan kemudian lakukan klik kanan sampai muncul menu *pop up*, lalu pilih menu “*Run as administrator*”.



Gambar 3.34.  
IDE Eclipse Siap di Instal

3. Hingga *splash screen* instalasi berjalan, seperti Gambar 3.35 berikut:



Gambar 3.35  
Proses Instalasi IDE Eclipse

4. Pada window pilihan paket pilihlah dengan cara klik paket “*Eclipse IDE for Java Developers*”.



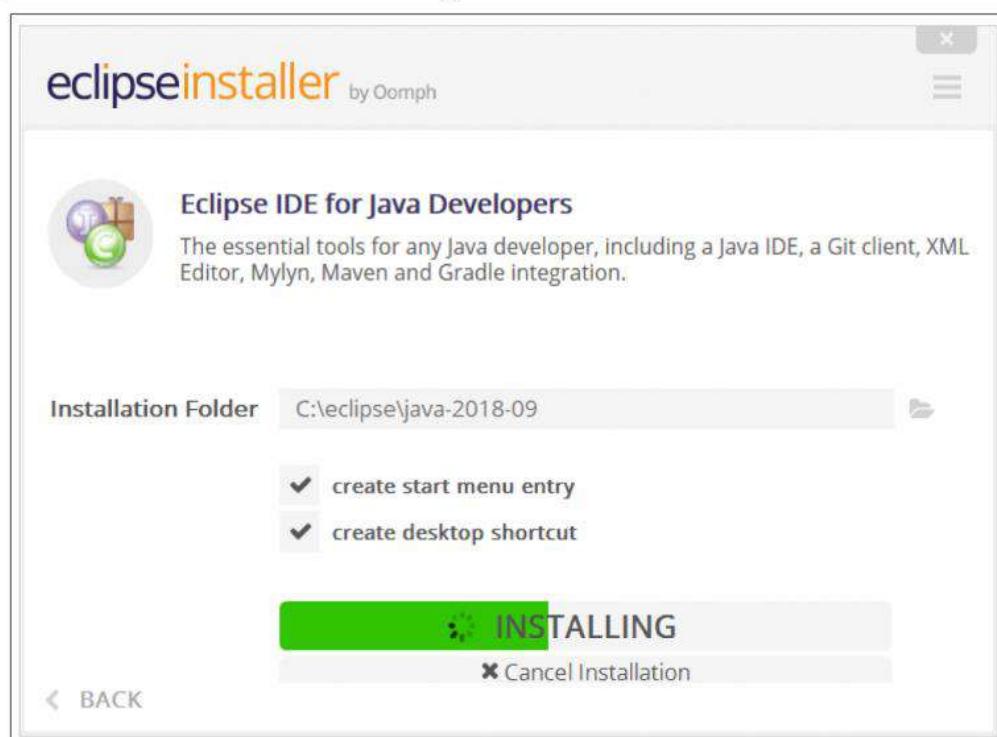
Gambar 3.36  
Memilih Versi IDE Eclipse yang Hendak di Instal

5. Pada pilihan paket “*Eclipse IDE for Java Developers*”, ubah folder instalasi menjadi “*C:\eclipse\java-2018-09*” dan untuk memulai instalasi klik tombol “*INSTALL*”.



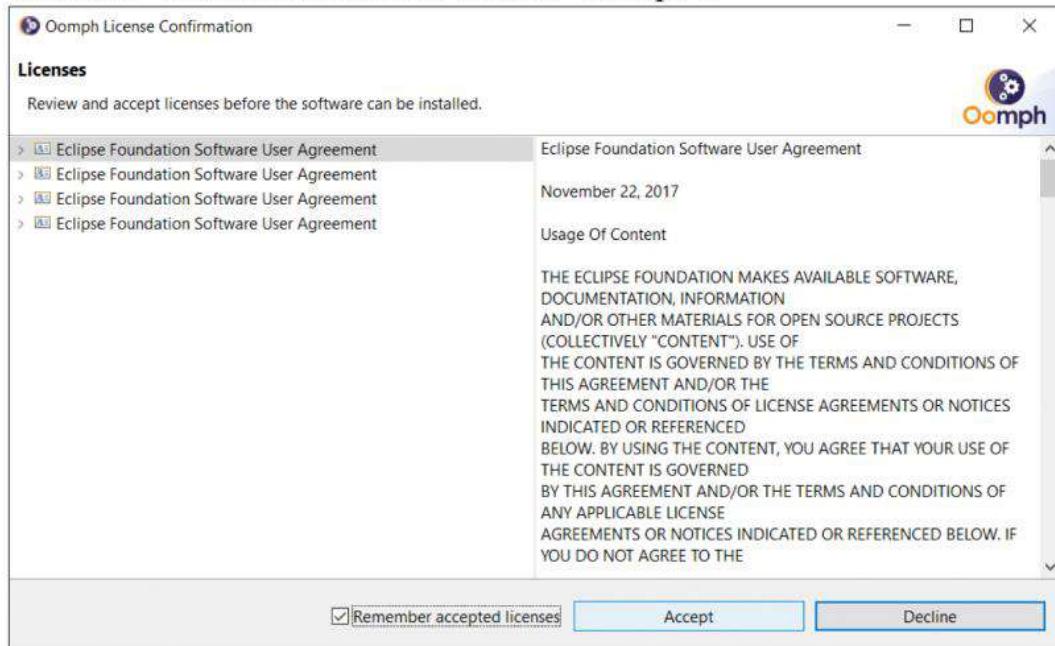
Gambar 3.37  
IDE Eclipse Terpilih Siap Instal

6. Pada proses instalasi akan tampak progress bar yang menunjukkan proses instalasi sementara berjalan.



Gambar 3.38  
Proses Instalasi IDE Eclipse

7. Setelah muncul Window Lisensi, centang opsi “Remember accepted licenses” dan kemudian klik tombol “Accept”.



Gambar 3.39.  
Window License IDE Eclipse

8. Selanjutnya tunggu hingga proses instalasi berhasil dengan sempurna, hingga tampak seperti gambar di bawah, untuk menjalankan IDE Eclipse pertama kali, klik tombol “LAUNCH”



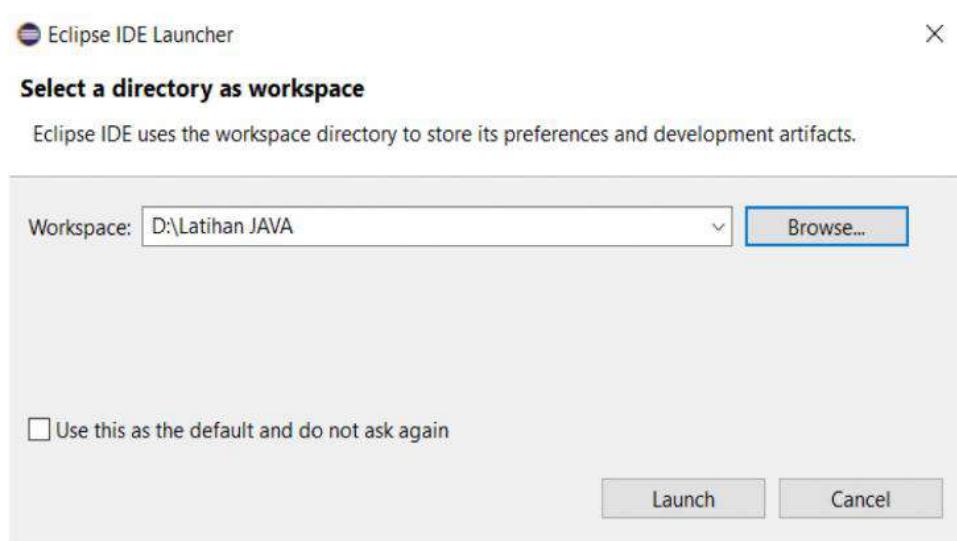
Gambar 3.40.  
Proses Instalasi Selesai

9. *Splash Screen* dari IDE Eclipse



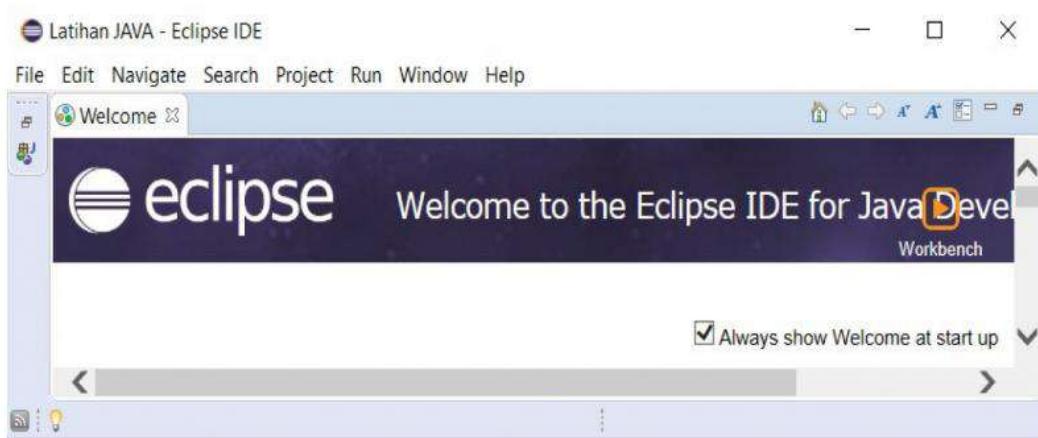
Gambar 3.41.  
*Splash Screen* IDE Eclipse

10. Pada Window “*Select a directory as workspace*”, pilihlah folder di mana Anda akan menyimpan file-file Java yang Anda buat, usahakan sama dengan folder seperti pada gambar di bawah walau *drive* nya berbeda. Di bawah saya menyimpannya pada folder “Latihan Java” dan disimpan pada drive “D:\”, sesuaikan *drive* yang ada di komputer Anda.



Gambar 3.42.  
Memilih Lokasi *Workspace* untuk Aplikasi

11. Selanjutnya klik tombol “*Launch*”, dan tunggu hingga IDE Eclipse terbuka dengan sempurna.



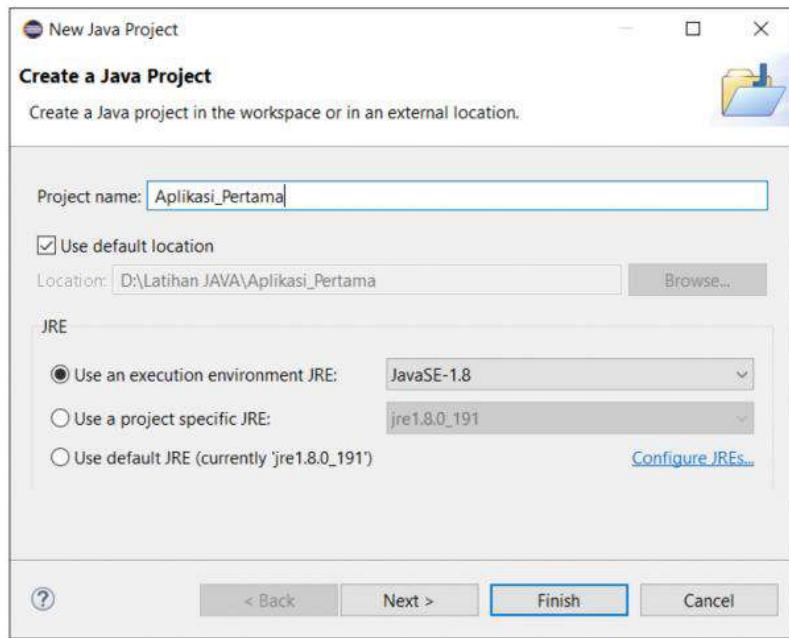
Gambar 3.43  
IDE Eclipse Pertama Kali Dibuka

12. Kemudian klik *File* → *Close*.

#### D. MENJALANKAN IDE ECLIPSE DAN MEMBUAT PROGRAM PERTAMA JAVA

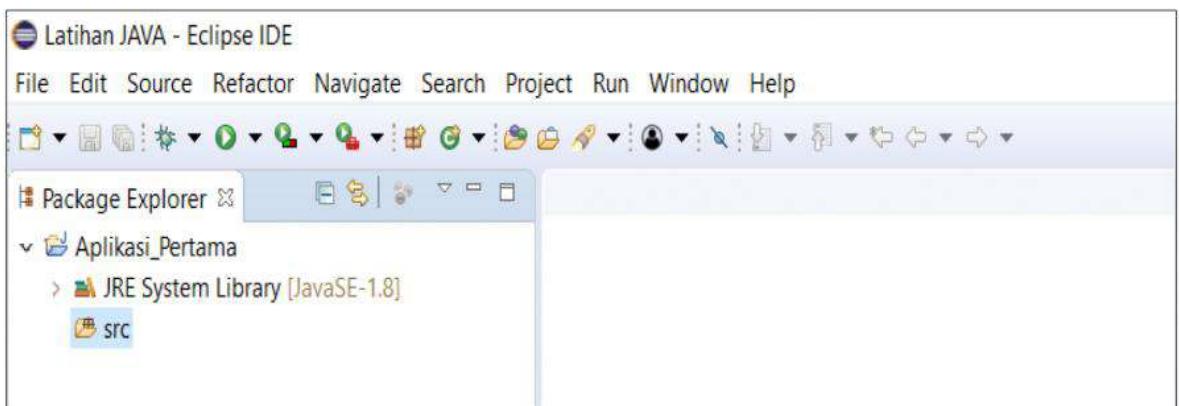
Pada sub bab C sudah dijelaskan instalasi dan menjalankan IDE Eclipse untuk pertama kali, sekarang akan dijelaskan bagaimana menjalankan kembali IDE Eclipse yang baru saja di instal sebagai berikut:

1. Klik dua kali *Icon* IDE Eclipse dari Desktop, *shortcut* dengan nama “*Eclipse Java 2018-09*”, dan tunggu sampai IDE Eclipse terbuka.
2. Klik *File* → *New* → *Java Project*.
3. Pada window *New Java Project*, isi kolom *Project Name*: dengan nama “*Aplikasi\_Pertama*”, lalu klik tombol “*Finish*”.



Gambar 3.44  
Window *Create Java Project*

4. Project *Aplikasi\_Pertama* akan tampak seperti Gambar 3.45 berikut:

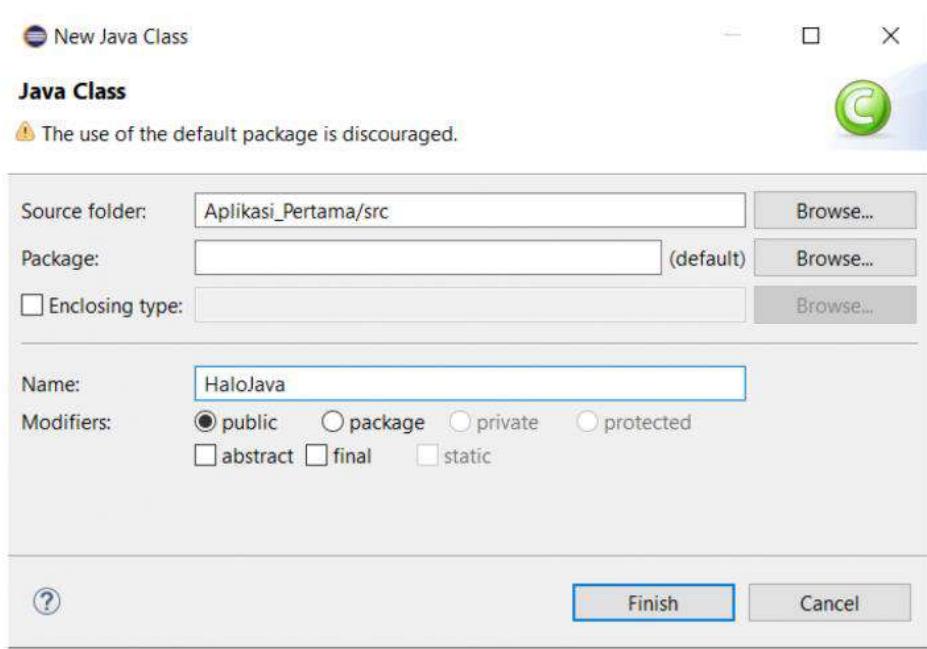


Gambar 3.45  
*Project Aplikasi\_Pertama*

Untuk:

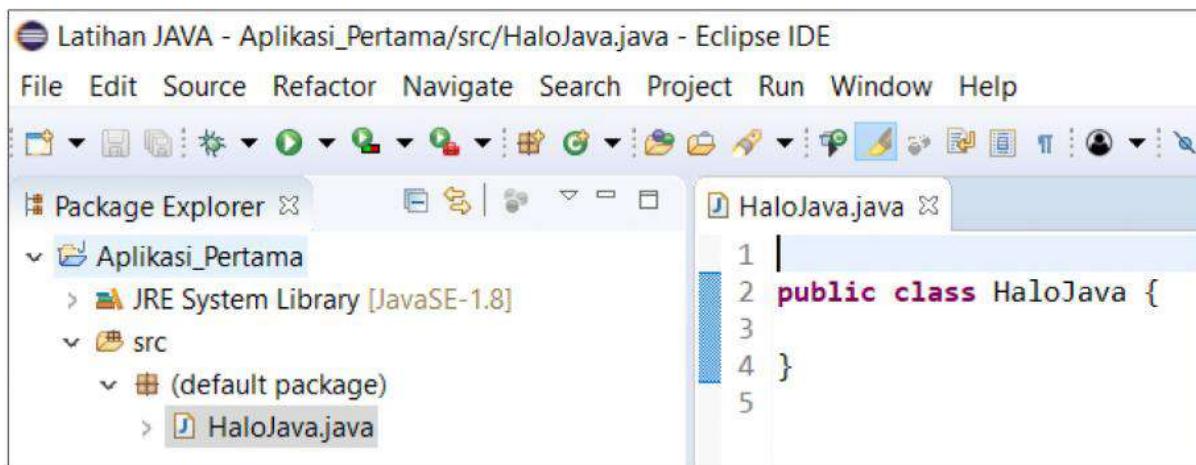
- *JRE System Library* : library-library yang tersedia bawaan JRE
- *src* : lokasi file-file Java yang akan kita buat.

5. Untuk membuat Java pertama kita, kembali klik *File* → *New* → *Class*.
6. Pada window “*New Java Class*” ini kolom *Name:* dengan *HaloJava* (tidak perlu menyertakan ekstensi *.java*, karena akan otomatis dibuat oleh IDE Eclipse)



Gambar 3.46  
Window Untuk Membuat *Class* Baru

7. Selanjutnya klik tombol *Finish* hingga terbentuk file *class* Java dengan nama *HaloJava.java*.

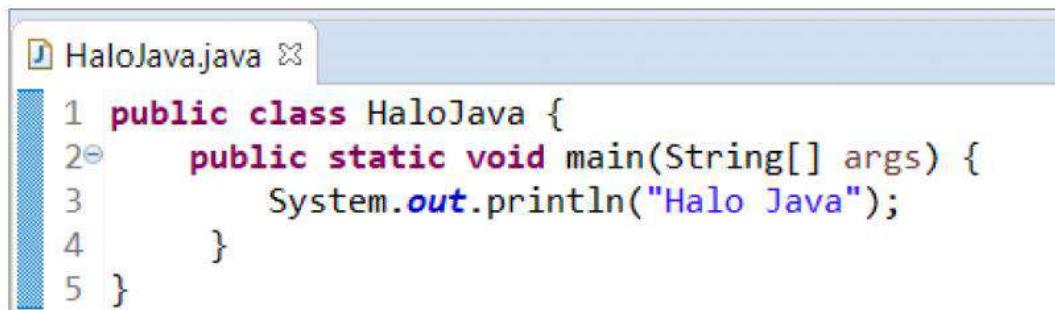


Gambar 3.47  
*Class* *HaloJava.java*

8. Sekarang kita akan menampilkan tulisan pada keluaran dengan tulisan “Halo Java”, dan untuk membuatnya, pada file *HaloJava.java* sisipkan potongan kode program berikut di antara {...}, sehingga menjadi kode program seperti di bawah ini:

```
public class HaloJava {
    public static void main(String[] args) {
        System.out.println("Halo Java");
    }
}
```

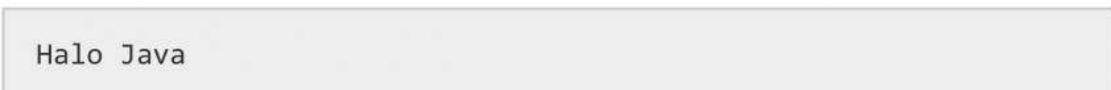
9. Kemudian simpan file *HaloJava.java* dengan cara klik *File* → *Save*.



```
1 public class HaloJava {
2     public static void main(String[] args) {
3         System.out.println("Halo Java");
4     }
5 }
```

Gambar 3.48  
*Class HaloJava.java dan Main Method*

10. Sekarang jalankan program pada file *HaloJava.java* dengan cara klik menu *Run* → *Run* atau dengan cara cepat menekan *Ctrl+F11*.  
 11. Lihat hasil keluaran program pada Tab *Console* pada bagian bawah IDE Eclipse



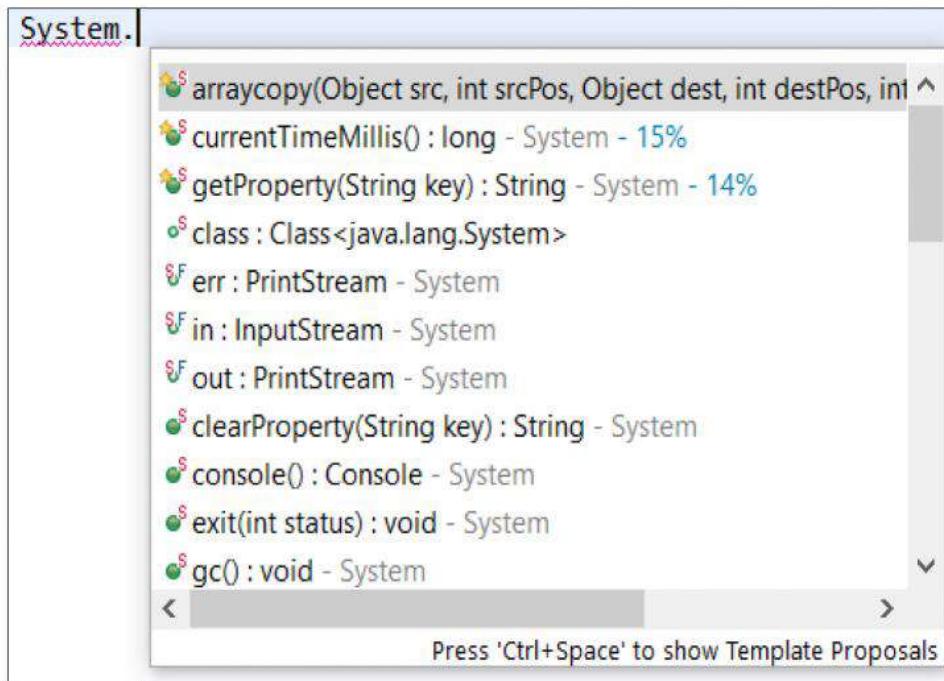
Halo Java

Catatan untuk beberapa perintah cepat pada IDE Eclipse:

1. Untuk membuat file baru/projek baru menggunakan *Alt+Shift+N*
2. Untuk menyimpan file menggunakan *Ctrl+S*
3. Untuk menjalankan program menggunakan *Ctrl+F11*.

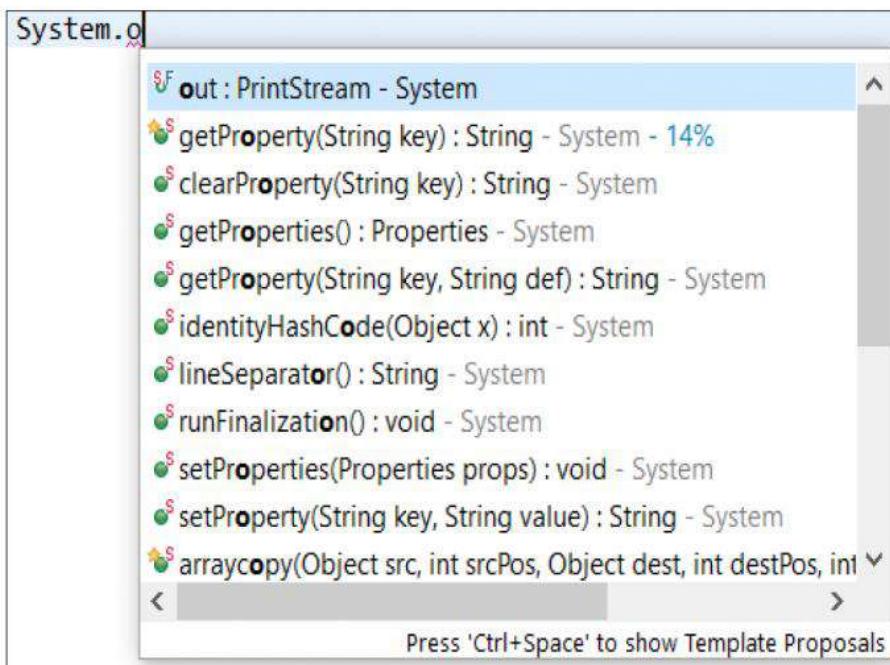
Untuk menulis kode dengan menggunakan *IntelliSense* juga berlaku pada IDE Eclipse, biasanya digunakan untuk mengurangi kesalahan-kesalahan ketik kode program. Contoh kode program `System.out.println("Halo Java");`, ini bisa dilakukan sebagai berikut:

1. Buka file *class HaloJava.java*.
2. Dalam *body main method* buat baris baru dan coba ketikkan *System.*



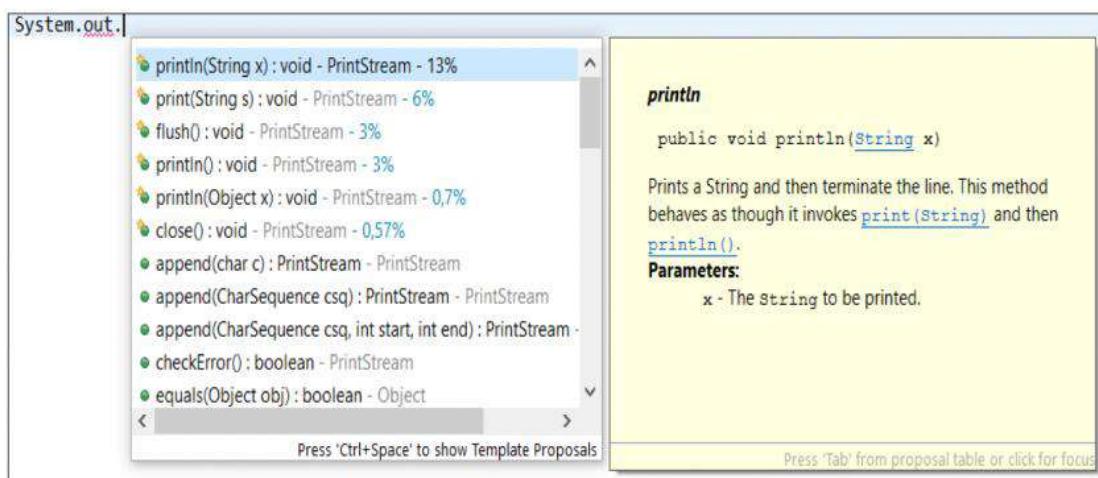
Gambar 3.49  
Fasilitas *IntelliSense* yang Mempermudah *Programmer*

### 3. Kemudian ketikkan o (hasilnya)



Gambar 3.50  
Fasilitas *IntelliSense* yang Mempermudah *Programmer*

#### 4. Ketikkan ut. (hasilnya)



**Gambar 3.51**  
**Deskripsi dari Sebuah Sintaks**

Bukan hanya turunan dari sebuah perintah yang muncul, bahkan sampai pada deskripsi dan contoh dari perintah yang ada akan muncul.



#### LATIHAN

Untuk memperdalam pemahaman Anda mengenai materi di atas, kerjakanlah latihan berikut!

- 1) Sebutkan *shortcut* Eclipse yang anda ketahui.

#### *Petunjuk Jawaban Latihan*

- 1) Beberapa perintah cepat/*shortcut* Eclipse yang tersedia dari pengembang adalah:

Ctrl + D	Hapus baris atau beberapa baris
Ctrl + /	Membuat komentar atau membatalkan komentar
Ctrl + H	Membuka dialog pencarian
Ctrl + E	Pergi ke editor yang lain
Ctrl + Shift + G	Mencari ruang kerja untuk referensi ke metode atau variabel yang lain
Ctrl + 1	Pergi ke baris tertentu

Ctrl + F	Pencarian
Ctrl + Q	Pergi ke tempat terakhir di edit



## RANGKUMAN

---

Dalam pengembangan *software*, *Programmer Java* tidak lepas dari IDE (*Integrate Development Environment*). Keuntungan menggunakan IDE adalah banyak fitur yang bisa digunakan untuk mempercepat pengembangan *software*, IDE pada Java sendiri sudah cukup banyak, misalnya Netbeans, Eclipse, BlueJ, JBuilder (Borland), JDeveloper (Oracle).

Eclipse adalah IDE yang digunakan untuk mengembangkan perangkat lunak yang bisa dijalankan diberbagai plafom, Eclipse dapat berjalan pada plafom *Microsoft Windows*, *Linux*, *Solaris*, *IX*, *HP-UX*, dan *Mac OS X*. Untuk mendownload Eclipse dapat mengunjungi laman <https://www.eclipse.org/downloads/packages/installer>.

## Glosarium

- Console* : Dalam pengertian *operating system* memiliki arti yang berarti *command line*, yakni *prompt* utama pada sebuah sistem komputer atau sistem operasi yang menggunakan perintah berbasis teks. Biasanya *command line* pada prompt utama tersebut diakhiri dengan tanda- --tanda khusus seperti \$, %, atau >. Lebih luas lagi, istilah “*console*” juga digunakan setiap kali seorang pengguna komputer mengetikkan perintah dengan menggunakan keyboard pada komputer, baik pada prompt sistem operasi, maupun dalam program. Dengan demikian, “*console*” dapat diartikan sebagai “*command line*”.
- Dropbox* : layanan penyedia data berbasis web yang dioperasikan oleh Dropbox, Inc. Dropbox menggunakan sistem penyimpanan berjaringan yang memungkinkan pengguna untuk menyimpan dan berbagi data serta berkas dengan pengguna lain di internet menggunakan sinkronisasi data. Dropbox didirikan pada tahun 2007 oleh lulusan Massachusetts Institute of Technology (MIT) Drew Houston dan Arash Ferdowsi dengan modal awal yang didapat dari Y Combinator.
- Google Drive* : layanan penyimpanan daring milik Google yang diluncurkan pada 24 April 2012. Layanan ini merupakan ekstensi dari Google Docs dan akan mengganti URL docs.google.com dengan drive.google.com setelah diaktifkan. Google Drive

memberikan layanan penyimpanan gratis sebesar 15 GB dan dapat ditambahkan dengan pembayaran tertentu.

*Hotkey*

- : jalan pintas/*shourtcut* dengan tombol keyboard, yang ketika ditekan dengan sendirinya atau dalam kombinasi dengan tombol lainnya, melakukan suatu fungsi.

**HTML5**

- : Sebuah bahasa markah untuk menstrukturkan dan menampilkan isi dari World Wide Web, sebuah teknologi inti dari Internet. HTML5 adalah revisi kelima dari HTML (yang pertama kali diciptakan pada tahun 1990 dan versi keempatnya, HTML4, pada tahun 1997) dan hingga bulan Juni 2011 masih dalam pengembangan. Tujuan utama pengembangan HTML5 adalah untuk memperbaiki teknologi HTML agar mendukung teknologi multimedia terbaru, mudah dibaca oleh manusia dan juga mudah dimengerti oleh mesin.

**IDE** (*Integrate Development Environment*)

- : program komputer yang memiliki beberapa fasilitas yang diperlukan dalam pembangunan perangkat lunak. Tujuan dari IDE adalah untuk menyediakan semua utilitas yang diperlukan dalam membangun perangkat lunak.

**IE (Internet Explorer)**

- : Sebuah peramban web dan perangkat lunak tak bebas yang gratis dari Microsoft, dan disertakan dalam setiap rilis sistem operasi Microsoft Windows sejak 1995. Pada mulanya, Internet Explorer dirilis sebagai bagian dari paket Plus! for Windows 95 (Inggris) pada saat itu.

**iOS**

- : Sistem operasi perangkat bergerak yang dikembangkan dan didistribusikan oleh Apple Inc. Sistem operasi ini pertama diluncurkan tahun 2007 untuk iPhone dan

iPod Touch, dan telah dikembangkan untuk mendukung perangkat Apple lainnya seperti iPad dan Apple TV. Tidak seperti Windows Phone (Windows CE) Microsoft dan Android Google, Apple tidak melisensikan iOS untuk diinstal di perangkat keras non-Apple.

- OneDrive : layanan komputasi awan serupa dengan Dropbox dan Google Drive yang memungkinkan penggunanya mengunggah dan mensinkronkan berkas ke suatu penyimpanan awan dan kemudian mengaksesnya melalui peramban Web atau perangkat tertentu. Layanan ini dibuat oleh Microsoft dan merupakan bagian dari layanan daring Windows Live dan memungkinkan pengguna menyimpan berkas-berkasnya secara pribadi, membagikannya dengan orang-orang dalam kontak, atau menjadikan berkas-berkas bersifat umum. Berkas-berkas yang dibagikan untuk umum tidak memerlukan akun Microsoft untuk mengaksesnya.
- Platform : kombinasi antara sebuah arsitektur perangkat keras dengan sebuah kerangka kerja perangkat lunak (termasuk kerangka kerja aplikasi). Kombinasi tersebut memungkinkan sebuah perangkat lunak, khusus perangkat lunak aplikasi, dapat berjalan. Platform yang umum sudah menyertakan arsitektur, sistem operasi, bahasa pemrograman dan antarmuka yang terkait (pustaka sistem *runtime* atau antarmuka pengguna grafis) untuk komputer.
- kombinasi antara sebuah arsitektur perangkat keras dengan sebuah kerangka kerja perangkat lunak (termasuk kerangka

kerja aplikasi). Kombinasi tersebut memungkinkan sebuah perangkat lunak, khusus perangkat lunak aplikasi, dapat berjalan. Platform yang umum sudah menyertakan arsitektur, sistem operasi, bahasa pemrograman dan antarmuka yang terkait (pustaka sistem runtime atau antarmuka pengguna grafis) untuk komputer.

## Daftar Pustaka

- Dennis, Wixom, Roth. (2009). *System analysis and design*. Wiley. USA.
- Kumad, D. (2009). *Overview of system analysis & design*. xxx.
- Langer, A.M. (2008). *Analysis and design of information systems*. New York: Springer.
- Purbasari, I.Y. Desain & analisis algoritma. Yogjakarta: Graha Ilmu.
- Hartati, G. S. (2007). Pemrograman GUI swing java dengan netbeans 5. Yogyakarta: Andi Offset.
- Kadir, A. (2005). Dasar pemrograman java 2". Yogyakarta: Andi Offset.
- <https://www.eclipse.org/downloads/packages/installer>
- <https://about.draw.io/integrations/>
- <https://www.draw.io/>

# Bahasa Pemrograman Java

Kani, M.Kom.



## PENDAHULUAN

---

Mulai dari tahun 1996 hingga sekarang, cukup banyak perusahaan dunia menggunakan Java sebagai bahasa pemrograman utama dalam mengembangkan banyak *software*. Kondisi ini mengakibatkan tenaga kerja atau pencari kerja, mahasiswa, perguruan tinggi berlomba-lomba mempelajari dan mengembangkan kurikulum dengan menggunakan bahasa Java sebagai materi utama. Kemampuan bahasa pemrograman Java dalam menyihir semua komunitasnya adalah mengikuti perkembangan teknologi terkini dengan menggunakan pemrograman *mobile*. IDE Android Studio yang menjadikan Java sebagai bahasa utama. IDE Android Studio memberikan kemudahan kepada *programmer* untuk mengembangkan aplikasi-aplikasi *mobile* yang dibutuhkan oleh masyarakat dunia. Java mampu menjawab semua kebutuhan *programmer* dengan segala kelebihan yang dimilikinya walau tidak lepas dari kekurangan. Kolaborasi antara pengembang Java dengan *vendor* pendukung memberikan khazanah pengetahuan dan kemudahan dalam mengembangkan berbagai aplikasi. Semoga harapan tersebut di atas juga terwujud kepada mahasiswa. Setelah mempelajari modul ini mahasiswa mampu :

1. menjelaskan sejarah Java;
2. menerangkan konsep pemrograman Java;
3. memahami kelebihan dan kekurangan pemrograman Java;
4. memahami pemrograman Java sebagai bahasa berbasis OOP (*Object Oriented Programming Language*);
5. mengenali kit Java;
6. mengerti cara instal Java;
7. memahami struktur pemrograman Java;
8. mengenali dan memahami tipe data dan variabel pada pemrograman Java.

## KEGIATAN BELAJAR 1

### Pengantar Java

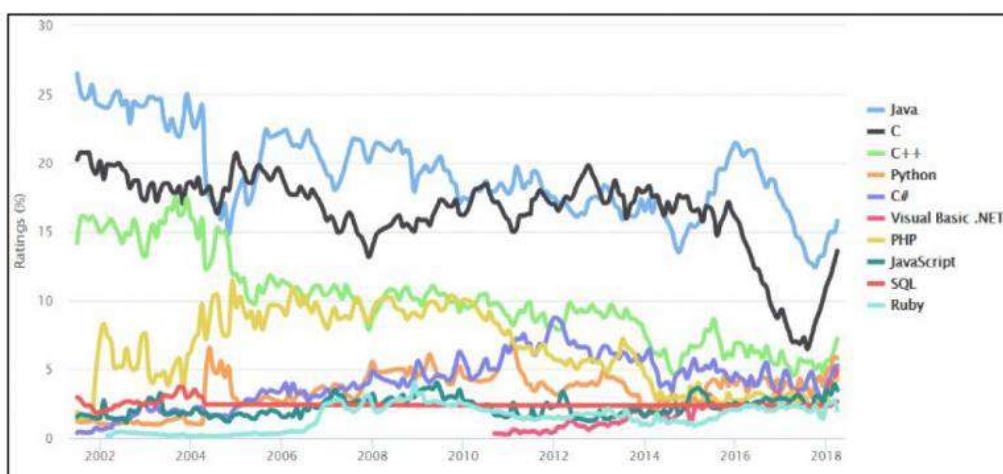
Tidak bisa dibendung! Mungkin itulah kalimat yang mewakili kepopuleran bahasa pemrograman Java. Banyak *programmer OOP* yang beralih ke pemrograman menggunakan Java. Tiobe (Perusahaan *software* terkenal) melakukan survei terhadap 20 bahasa pemrograman paling diminati, ternyata Java menempati urutan pertama dengan peminat 15.777%, kemudian disusul bahasa pemrograman C dengan peminat 13.589%, dan urutan ketiga adalah bahasa pemrograman C++ dengan peminat 7.218%. Data yang disebutkan di atas adalah data bulan April 2018.

Rank	Programming Language	Ratings	Change
1	Java	15.777%	+0.21%
2	C	13.589%	+6.62%
3	C++	7.218%	+2.66%
4	Python	5.803%	+2.35%
5	C#	5.265%	+1.69%
6	Visual Basic .NET	4.947%	+1.70%
7	PHP	4.218%	+0.84%
8	JavaScript	3.492%	+0.64%
9	SQL	2.650%	+2.65%
10	Ruby	2.018%	-0.29%

(sumber: [www.tiobe.com](http://www.tiobe.com))

**Gambar 4.1**  
Tren Peminat Bahasa Pemrograman Java Menurut Tiobe

Namun jika dibandingkan dari tahun 2002, Java memang mendominasi pertarungan dunia *programming*. Pada tahun 2004 dan 2014-2015 bahasa pemrograman C mengambil alih, setelah itu bahasa pemrograman Java nyaris tidak terkejar.



**Gambar 4.2**  
Dominasi Bahasa Pemrograman Jawa dari Tahun Ke Tahun

Pertanyaannya adalah mengapa bahasa pemrograman Java begitu diminati? Beberapa alasan berikut adalah jawabannya:

1. Tren teknologi yang berkembang pada setiap tahun, Java tetap stabil dengan OOP dengan *library* yang lengkap, dan bukan hanya Desktop akan tetapi juga pemrograman *mobile*.
2. Banyak digunakan perusahaan besar menggunakan bahasa pemrograman Java, yang kemudian mempengaruhi *programmer* pencari kerja untuk mau tidak mau belajar bahasa pemrograman Java.
3. Efisiensi waktu dan biaya (biaya eksekusi murah, biaya translasi dan kompilasi, serta pemeliharaan murah).
4. Dukungan komunitas yang luas. Komunitas Java terbesar di dunia.

## A. SEJARAH JAVA

Benarkah bahwa kata “Java” digunakan sebagai nama pemrograman Java dikarenakan pada saat dikembangkan *programmer*-nya selalu minum kopi asal Indonesia yang dikenal dengan kopi Jawa atau Java? Ataukah pada saat pengembangan bahasa pemrograman Java ada orang Jawa yang terlibat? Fakta yang benar adalah pada saat Java mulai dikembangkan oleh James Gosling dan teman-temannya di Sun Microsystem, mereka selalu minum kopi di kafe Peet yang mungkin kopinya asalnya dari Indonesia yang orang luar Indonesia menyebutnya sebagai kopi Java. Untuk pemilihan nama “Java”, James Gosling terinspirasi dari pohon yang tumbuh di depan jendela ruang kerjanya. Masalah nama memang menjadi perdebatan saat itu. Yang

terlibat dalam pemilihan nama pada saat itu adalah insinyur, manajer pemasaran, penasehat hukum, dan direksi Sun Microsystems sendiri. Ada tiga nama yang menjadi kandidat pada saat itu, yaitu: Silk, DNA, dan Java, dan secara kolektif terpilihlah kata “Java” yang digunakan untuk bahasa pemrograman Java.

Kembali ke tahun 1991, di tahun tersebut dibentuk sebuah tim yang dimotori oleh Patrick Naughton, Mike Sheridan, James Gosling dan Bill Joy, beserta sembilan *programmer* dari Sun Microsystems. Tim tersebut dikenal dengan nama *The Green Project*, tujuan dari proyek ini adalah untuk menghasilkan bahasa komputer yang sederhana dan dimungkinkan jalan pada *hardware* yang sederhana dan tidak terlalu terbebani oleh arsitektur komputer dan sistem operasi tertentu.

Pertemuan demi pertemuan proyek ini berlangsung disebuah gedung perkantoran Sand Hill Road di Menlo Park, California, dan hingga menghasilkan produk OAK pada tahun 1992, karena OAK sendiri merupakan nama dari bahasa pemrograman komputer yang sudah ada pada masa itu maka kemudian Sun mengubahnya menjadi Java.

Pada tanggal 14 Juni 1993, anak perusahaan sebuah TV Kabel tertarik dengan teknologi *The Green Project* ini, maka dikembangkan lebih jauh dengan menggunakan alumni *The Green Project* dan ditambah 70 orang *programmer* lainnya, mereka memusatkan kegiatannya di sebuah ruangan kantor di 100 Hamilton Avenue, Palo Alto.

Setelah melalui beberapa proses dan transformasi, Sun Microsystem akhirnya meluncurkan *browser* dari Java yang disebut *Hot Java* yang mampu menjalankan *applet*. Setelah itu teknologi Java diadopsi oleh Netscape yang memungkinkan program Java dijalankan di browser Netscape sejak Januari 1996, kemudian disusul oleh Internet Explorer (Microsoft). Karena keunikan dan kelebihannya, teknologi Java mulai menarik banyak vendor terkemuka seperti IBM, Symantec, Inprise, dan lain lain.

Sun Microsystem merilis versi awal Java secara resmi pada awal 1996 yang kemudian terus berkembang hingga muncul JDK 1.1, kemudian muncul JDK 1.2 yang dengan berbagai perbaikan dan penambahan fitur sehingga sejak versi 1.2, Java disebut dengan Java2. Perubahan yang utama adalah adanya *Swing* ialah teknologi GUI (*Graphical User Interface*) yang mampu menghasilkan aplikasi window yang benar-benar portabel.

Mulai tahun 1995 hingga modul ini dibuat, Java selalu memberikan perbaikan dan *update*. Dimulai dari JDK Beta (1995), JDK 1.0 (1996), JDK

1.1 (1997), J2SE 1.2 (1998), J2SE 1.3 (2000), J2SE 1.4 (2002), J2SE 5.0 (2004), Java SE 6 (2006), Java SE 7 (2011), Java SE 8 2014, Java SE 9 (2017), dan Java SE 10 (2018).

## B. KONSEP PEMROGRAMAN

Dua konsep pemrograman yang banyak dikenali oleh publik adalah konsep pemrograman terstruktur dan konsep pemrograman berorientasi objek.

### 1. Konsep Pemrograman Terstruktur

Pemrograman terstruktur adalah metode pemrograman logis yang dianggap sebagai pendahulu (metode lama), konsep pemrograman ini memfasilitasi pemahaman dan modifikasi program-program dengan pendekatan desain *top-down*, dimana sistem dibagi menjadi sub-sistem atau atau sub-prosedur.

Pemrograman terstruktur memberikan pendekatan global pada suatu permasalahan (prosedur besar), yang kemudian dipecah menjadi prosedur-prosedur yang lebih kecil dan saling terintegrasi satu dengan yang lainnya. Konsep pemrograman terstruktur diperkenalkan pada tahun 1966 oleh Corrado Böhm dan Giuseppe Jacopini yang mendemonstrasikan desain program komputer teoritis melalui urutan, keputusan, dan perulangan. Pada akhir 1960-an atau awal 1970-an, Edsger W.Dijkstra mengembangkan fungsionalitas pemrograman struktural sebagai metode yang banyak digunakan. Pada era tersebut program dibagi menjadi beberapa bagian dengan beberapa keluaran dan satu jalur akses.

Beberapa Bahasa pemrograman yang menerapkan konsep pemrograman terstruktur, seperti Pascal (bukan object Pascal), Cobol, Fortran, Basic, dan C. Pada dasarnya, bukan hanya bahasa pemrograman yang disebutkan di atas, di awal kemunculan bahasa pemrograman, ada pemrograman “ADA” yang dianggap sebagai bahasa pemrograman awal.

### 2. Konsep Pemrograman Berorientasi Objek

Pemrograman berorientasi Objek dimulai pada tahun 1960-an. Bahasa pemrograman pertama yang menggunakan objek adalah bahasa pemrograman Simula 67 (1967) yang dikembangkan oleh Kristen Nygaard dan Ole-Johan

Dahl yang berkebangsaan Norway. Simula tidak hanya memperkenalkan konsep kelas atau *class* dan kelas *instance*.

Munculnya bahasa pemrograman Simula 67 banyak menginspirasi munculnya bahasa pemrograman lain yang berbasis objek, seperti Smalltalk yang digunakan oleh perusahaan Xerox PARC yang murni menggunakan objek sebagai pondasi untuk komputasi. Tidak hanya sampai di situ, tim Smalltalk dirancang menjadi sangat/lebih dinamis. Objek dapat dibuat, diubah, atau dihapus, dan ini berbeda dari sistem statis yang umum digunakan. Salah satu kelebihan pemrograman yang berorientasi objek adalah membagi program menjadi objek-objek yang saling berinteraksi satu sama lain.

## C. KELEBIHAN DAN KEKURANGAN JAVA

Semua bahasa pemrograman yang ada di dunia ini mempunyai kelebihan dan kekurangan, tak terkecuali bahasa pemrograman Java.

### 1. Kelebihan Pemrograman Java

- a. *Multiplatform*: Bahasa pemrograman Java disebut juga sebagai bahasa pemrograman universal. Java dapat berjalan di beberapa *platform*, oleh karena itu banyak diminati oleh *programmer* dan *developer*, program hanya ditulis sekali, dan bisa dijalankan pada sistem operasi apapun. Sampai saat ini, *platform* yang mendukung bahasa pemrograman Java adalah Microsoft Windows, Linux, Mac OS, dan Sun Solaris.
- b. *Library* yang lengkap: Tidak bisa dipungkiri bahwa Java memiliki/kaya *library* (kepustakaan) yang banyak. Ketersediaan *library* ini menjadi daya tarik, *programmer* tidak perlu membangun awal sebuah *fungsi*, akan tetapi cukup menggunakan yang sudah tersedia sesuai dengan kebutuhan.
- c. Mendekati sintak C++: Banyak *programmer* bahasa lain beralih ke Java. Sebagian besar *programmer* yang beralih ke Java adalah adalah *programmer* C++. Di Indonesia, mayoritas perguruan tinggi menggunakan Java dalam matakuliah pemrograman.
- d. Berorientasi Objek: Struktur pemrograman Java berorientasi objek (agak teknis). Objek dibungkus dalam *class* yang terdiri dari *attribute* (atribut) dan *method* yang saling terintegrasi dalam satu

kelompok agar lebih terorganisir, misalnya *class Sepeda*, didalamnya terdapat macam-macam sepeda dan memiliki perlakuan, atribut dan *method* yang berbeda.

## 2. Kelemahan Pemrograman Java

- a. Fitur tidak Kompatibel: Walau dikatakan pada kelebihan tulis sekali dan bisa digunakan di *platform* mana saja, namun kenyataannya, ada beberapa fitur yang tidak bisa jalan di beberapa *platform*.
- b. Boros Penggunaan Memori: Bahasa pemrograman Java ternyata memakan memori yang tidak sedikit. Namun hal ini sudah tidak masalah besar, karena saat ini perangkat memori sudah jauh lebih murah dan mudah didapat.
- c. Mudah didekompilasi: Karena bahasa pemrograman Java merupakan *bytecode* artinya sangat mudah untuk dibalikkan ke kode sumber. Hal ini menjadi pekerjaan rumah dari pengembang / developer untuk melindungi apa yang tidak boleh diketahui dalam program.

## D. JAVA ADALAH OOP (*OBJECT ORIENTED PROGRAMMING*)

Java adalah bahasa pemrograman yang berbasis objek yang biasa dikenal dengan istilah *Object Oriented Programming* atau disingkat OOP. OOP adalah gaya pemrograman dengan pendekatan objek, kelas, *inheritance* (pewarisan), *encapsulation* (enkapsulasi), *abstraction* (abstraksi), dan *polymorphism* (polimorfisme). Pada pengembangan awal Java telah membangun konsep OOP.

### 1. *Object* (Objek)

Kalau kita mengatakan objek, kira-kira apa yang terlintas dalam pikiran Anda? Apakah kucing, anjing, mobil atau diri anda sendiri? Pandang sekeliling Anda, pandang diluar sana, apa yang Anda lihat? Apa yang anda amati? Yang Anda amati adalah objek. Objek adalah semua hal yang ada dalam dunia nyata. Pada konsep OOP, objek adalah bagian terkecil pemrograman yang memiliki sifat dan karakteristik.

Setiap objek mempunya karakteristik, sebagai berikut:

- a. Objek mempunyai atribut/properti yang dijadikan sebagai *state* (status), atribut/*property* adalah identitas atau suatu nilai yang dapat ditulis untuk

mengambarkan objek itu sendiri. Atribut adalah sebuah data yang menjadi pembeda antar objek yang satu dengan objek yang lain, kalau dalam *class* dikenal dengan istilah variabel.

- b. Objek mempunyai *method* atau *behavior* (perilaku), yaitu sebuah fungsi yang berlaku dan untuk mengakses suatu atribut dari sebuah objek, mampu menerima informasi dan memberikan informasi ke objek yang lain, contoh sepeda.



Gambar 4.3  
Sepeda adalah Objek

Sepeda mempunyai atribut: pedal, roda gigi, kecepatan, warna, jenis dan lain-lain. Sepeda mempunyai *method*: pindah roda gigi, kecepatan bertambah, kecepatan berkurang. Pada OOP, objek adalah satu bundel perangkat lunak yang tidak lepas dari status dan perilaku (*behavior*), status disimpan dalam variabel, sedangkan *behavior* disimpan dalam *method*.

## 2. *Class*

Dalam kehidupan nyata, kita sering melihat sebuah objek yang sama, misalnya sebuah sepeda, ribuan sepeda (Sepeda Gunung, Sepeda Santai, Sepeda Ontel, dan lain-lain) dibangun dari *prototype* (*blueprint*) yang sama, dibuktikan dengan sebuah sepeda mempunyai komponen yang sama, begitu juga dengan *class* dibangun dari prototipe dari sebuah objek. Jadi komponen dalam *class* tidak lain adalah atribut/property dan *method/ kelakuan* (*behavior*) yang didefinisikan.

Contoh 4.1:

```
//class cetak biru
class Sepeda {

    int kecepatan = 0;

    void tambahKecepatan(int tambah) {
        kecepatan = kecepatan + tambah;
    }

    void kurangiKecepatan(int kurang) {
        kecepatan = kecepatan - kurang;
    }

    void printTambah() {
        System.out.println("Naik ke Kecepatan\t:" + kecepatan
    );
    }

    void printKurang() {
        System.out.println("Turun ke Kecepatan\t:" + kecepatan );
    }

}

//Class Utama
public class objekSepeda {
    public static void main(String[] args) {
        // membuat objek
        // dengan nama SepedaGunung
        Sepeda sepedaGunung = new Sepeda();

        // memanggil method pada objek
        System.out.println("Sepeda Gunung");
        System.out.println("-----");
        sepedaGunung.tambahKecepatan(50);
        sepedaGunung.printTambah();
        sepedaGunung.kurangiKecepatan(10);
        sepedaGunung.printKurang();
    }
}
```

### 3. *Instance (Instans)*

Objek adalah instans dari suatu kelas, objek merupakan perwujudan dalam bentuk benda, jika dalam bahasa pemrograman mungkin disebut dengan konsep.

### 4. *Abstraction (abstraksi)*

Abstraksi adalah cara pandang terhadap suatu objek dalam bentuk sederhana, contoh objek Sepeda, tidak perlu melihat sepeda ke susunan komponen sampai se-detil-detilnya, cukup melihat itu sebagai objek sepeda yang mempunyai properti dan entitas.

Pemahaman ini diadopsi oleh pemrograman dengan sistem OOP, Anda tidak perlu tahu bagaimana suatu instruksi `System.out.println()` bekerja, akan tetapi yang perlu Anda tahu adalah bagaimana menggunakan perintah tersebut.

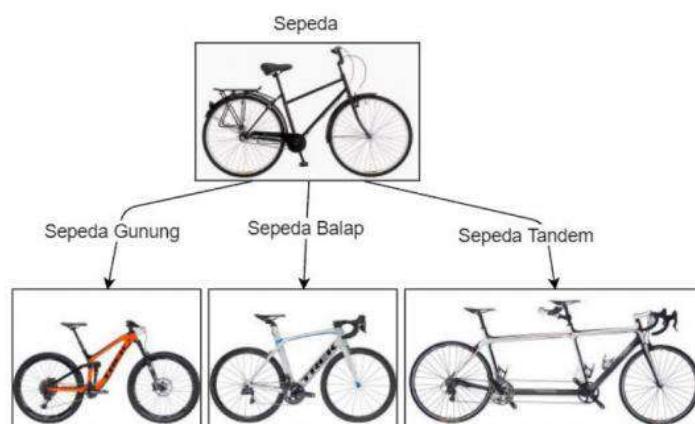
Contoh 4.2:

```
abstract class Sepeda //superClass
{
    private int kecepatan = 12;
    public void info()
    {
        System.out.println("Kecepatan = \t "+kecepatan);
        System.out.println("Panggil dari sub-class "+
                           this.getClass().getName());
    }
}
//Sub class
public class sepedaGunung extends Sepeda
{
    public static void main(String[] args)
    {
        //membuat objek
        sepedaGunung objek = new sepedaGunung();
        //memanggil objek
        objek.info();
    }
}
```

## 5. Inheritance (pewarisan)

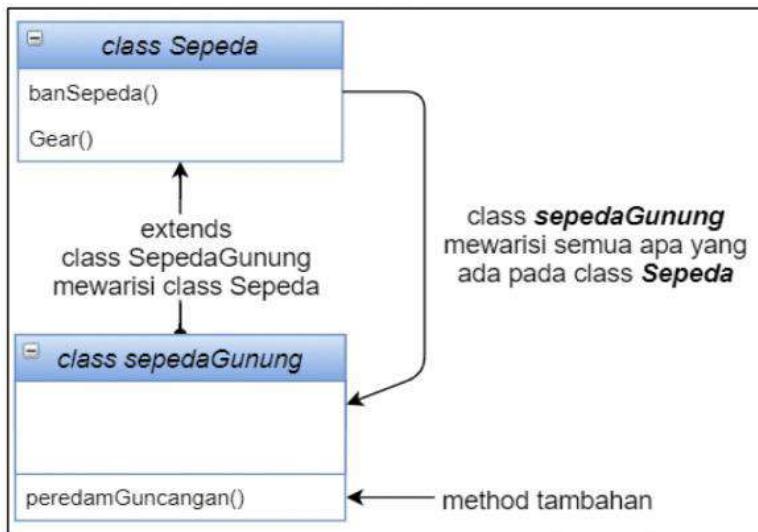
Objek sepeda yang dibangun dari prototipe yang sama. Sepeda Gunung, Sepeda Jalan, dan Sepeda Tandem atau sepeda apapun namanya, semua memiliki karakteristik yang sama, masih memiliki kecepatan, pedal, gigi. Namun masing-masing juga mendefinisikan fitur tambahan yang membuat mereka menjadi unik: sepeda tandem memiliki dua kursi dan dua set setang; sepeda jalan memiliki setang jatuh; beberapa sepeda gunung memiliki cincin rantai tambahan, memberi mereka rasio gigi yang lebih rendah.

Pemrograman berorientasi objek memungkinkan *class* untuk mewarisi keadaan dan perilaku yang umum digunakan dari *class* lain. Dalam contoh ini, Sepeda menjadi *super-class* dari Sepeda Gunung, Sepeda Balap, dan Sepeda Tandem. Dalam bahasa pemrograman Java, setiap *class* diperbolehkan memiliki satu *super-class* langsung, dan setiap *super-class* boleh memiliki sejumlah *sub-class*.



Gambar 4.4  
Macam-macam Sepeda Dibuat dari Satu Blueprint

Ilustrasi pewarisan *class* pada pemrograman Java disajikan di bawah ini:



Gambar 4.5  
Ilustrasi Pewarisan Sebuah *Class*

Cara mewariskan *class* dengan format sintaksis:

```

class superClass
{
    //blok pernyataan
}

class subClass Extends superClass
{
    //blok pernyataan
}
  
```

Keterangan:

- Untuk mewariskan *class* harus ada *class* yang akan diwariskan disebut dengan *super-class*.
- *Class* yang mewarisi disebut dengan *sub-class*.
- Untuk mewariskan *class* dari *class super-class* ke *class sub-class* harus menggunakan *keyword extends*.

Contoh 4.3:

```

class Sepeda //superClass
{
    private int kecepatan = 12;
    public void info()
    {
        System.out.println("Kecepatan = \t "+kecepatan);
        System.out.println("Panggil dari sub-class " +
                           this.getClass().getName());
    }
}
//Sub class
class sepedaGunung extends Sepeda
{
    public static void main(String[] args)
    {
        //membuat objek
        sepedaGunung objek = new sepedaGunung();
        //memanggil objek
        objek.info();
    }
}

```

Keluaran:

```

Kecepatan = 12
Panggil dari sub-class sepedaGunung

```

## 6. *Encapsulation (enkapsulasi)*

Enkapsulasi adalah pembungkus (menutupi, mengapsulkan) dengan tujuan agar suatu proses program tetap terjaga supaya tidak dapat diakses secara sembarangan oleh program lain. Pada Pemrograman Java pondasi dari enkapsulasi adalah *class*. *Method* atau variabel yang dibuat dalam *class* ditentukan oleh *programmer* menangani perijinan untuk aksesnya; apakah dapat/boleh diakses oleh program lain atau tidak. Hal ini tercantum pada *modifier* yang digunakan.

Terdapat tiga jenis *modifier* pada sistem enkapsulasi, yaitu: *Public*, *Protected*, dan *Private*.

- a. *Public*: *Keyword public* adalah *modifier* yang digunakan dalam pemrograman Java yang memungkinkan setiap variabel dan *method* yang dideklarasikan pada *class public* yang dapat diakses dan digunakan

oleh semua *class*, baik yang berada di dalam *package* yang sama maupun pada *package* berbeda.

- b. *Protected*: Keyword *protected* adalah *modifier* yang digunakan untuk memproteksi suatu *class*, misal *protected* dipasang pada *class super-class* maka hanya bisa diakses oleh *sub-class* pada *package* yang sama atau *class* pada *package* yang berbeda.
- c. *Private*: Keyword *private* adalah *modifier* yang digunakan untuk anggotanya hanya bisa diakses oleh *class* yang sama.

Tabel 4.1. Perbedaan 3 Jenis *Modifier* pada Sistem Enkapsulasi

No	Aksesibilitas	Public	Private	Protected
1	<i>Class</i> yang sama	√	√	√
2	Semua <i>class</i> dalam <i>package</i> yang sama	√	✗	√
3	Semua <i>class</i> di luar <i>package</i>	√	✗	√
4	<i>Sub-class</i> dalam <i>package</i> yang sama	√	✗	√
5	<i>Sub-class</i> di luar <i>package</i>	√	✗	✗

## 7. *Polymorphism* (polimorfisme)

Polimorfisme adalah suatu konsep yang menyatakan bahwa sesuatu yang sama dapat memiliki berbagai bentuk dan perilaku berbeda atau kemampuan sesuatu untuk mempunyai banyak bentuk. Pada OOP, polimorfisme memungkinkan kita mengenali dan mengexploitasi keserupaan di antara *class* yang berbeda.

Contoh 4.4:

```
//Super Class
class Sepeda {
    public void jenisSepeda ( )
    {
        System.out.println("Karakteristik Sepeda" + "belum
didefinisikan");
    }
}
// mendefinisikan kelas-kelas turunan dari kelas Sepeda
//Sub Class Sepeda Gunung
class sepedaGunung extends Sepeda
{
```

```

// melakukan override terhadap method jenisSepeda ( )
public void jenisSepeda ( )
{
    System.out.println("Jenis Sepeda Gunung");
    System.out.println("Jenis Sepeda dengan Ban Kasar");
}
//Sub Class Sepeda Santai
class sepedaSantai extends Sepeda {
    public void jenisSepeda ( )
    {
        System.out.println("Jenis Sepeda Santai");
        System.out.println("Jenis Sepeda dengan Ban Halus");
    }
}
//Main Class
public class ContohPolimorfisme {
    public static void main (String[] args)
    {
        Sepeda Jenis;

        sepedaGunung banKasar = new sepedaGunung ( );
        sepedaSantai banHalus = new sepedaSantai ( );

        // Jenis mengacu pada objek sepedaGunung
        Jenis = banKasar;
        // akan memanggil method pada kelas sepedaGunung
        Jenis.jenisSepeda ( );

        System.out.println("");

        // Jenis mengacu pada objek sepedaSantai
        Jenis = banHalus;
        // akan memanggil method pada kelas sepedaSantai
        Jenis.jenisSepeda ( );
    }
}

```

### Penjelasan:

- *Class Sepeda* adalah *class super-class*.
- *Class sepedaGunung* dan *class sepedaSantai* adalah *sub-class* yang mewarisi *class Sepeda* yaitu *method jenisSepeda*.
- Pada *class* utama di *method* utama, kita mendeklarasikan variabel referensi ke tipe *Sepeda* dengan nama *Jenis*. Sampai tahap ini kita belum mengetahui apakah *Jenis* merupakan Sepeda Gunung atau Sepeda Santai.
- *Jenis* kemudian mengacu ke objek dari *class sepedaGunung* dengan *banKasar*.

- *Jenis* kemudian mengacu ke objek dari *class sepedaSantai* dengan *banHalus*.
- *Jenis* mengacu ke *banKasar* dan kemudian objek *Jenis* dicetak dengan *Jenis.jenisSepeda*.
- Begitu juga dengan *banHalus*.

Keluaran:

```
Jenis Sepeda Gunung  
Jenis Sepeda dengan Ban Kasar  
  
Jenis Sepeda Santai  
Jenis Sepeda dengan Ban Halus
```

## E. JDK ATAU J2SDK

Kedua nama di atas memang agak membingungkan, sebenarnya bahasa pemrograman Java menggunakan kit yang mana? *Java Development Kit* (JDK) ataukah *Java 2 System Development Kit* (J2SDK)? Ada yang menyebutkan bahwa diantara keduanya hanyalah perbedaan dalam versi saja; namun ada juga yang berpendapat bahwa JDK adalah versi lama dan menggunakan AWT (*library* yang disediakan secara umum dan sejumlah *class* untuk membuat GUI yang berpenampilan kaku). Sedangkan J2SDK menggunakan komponen Swing (*library* Java yang digunakan untuk membuat GUI lebih dinamis dan bervariasi). Namun kita sudah fasih dengan sebutan JDK dibanding dengan J2SDK. Untuk mengetahui lebih jauh, Anda bisa mengunjungi laman <http://en.wikipedia.org/wiki/JDK> dan <http://en.wikipedia.org/wiki/J2SDK>.

JDK adalah kit pengembangan *software* yang digunakan untuk mengembangkan *software*, termasuk proses kompilasi dari kode Java ke *bytecode* yang dapat dimengerti komputer dan dapat dijalankan oleh JRE (*Java Runtime Environment*), *interpreter* (*java*), kompilator (*javac*), generator dokumentasi (*javadoc*) dan fasilitas/fitur yang lainnya yang dibutuhkan. JDK memiliki komponen utama sebagai kumpulan fitur pemrograman, dapat dilihat pada tabel berikut:

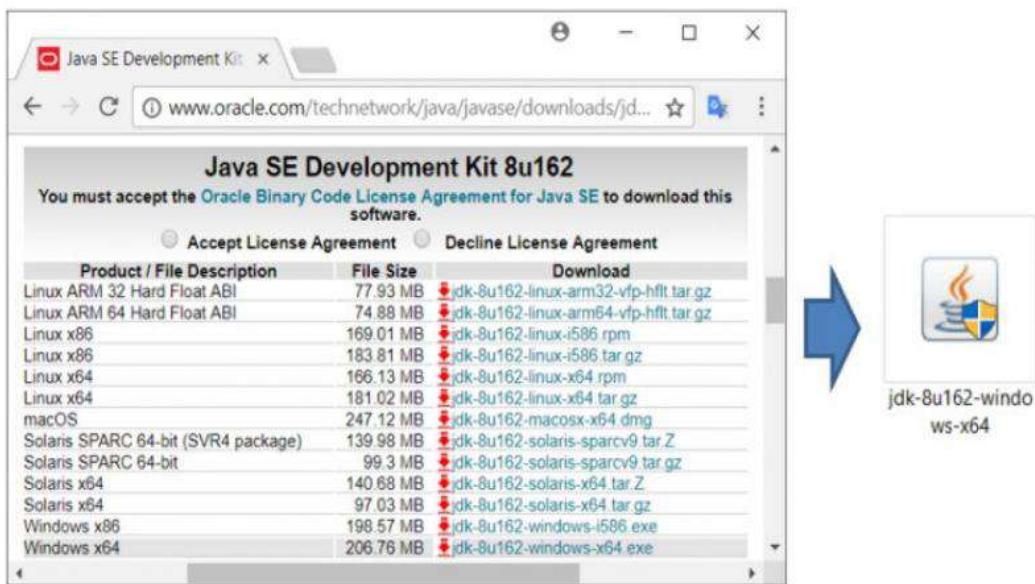
Tabel 4.2. Komponen JDK

<b>Komponen-komponen JDK</b>		
<b>1</b>	<code>javac</code>	Kompiler Java, mengubah kode sumber ke <i>bytecode</i>
<b>2</b>	<code>java</code>	<i>Loader</i> untuk aplikasi Java, digunakan sebagai penerjemah dan dapat meninterpretasikan file <i>class</i> yang dihasilkan oleh kompiler <i>javac</i> .
<b>3</b>	<code>javadoc</code>	Generator dokumentasi, menghasilkan dokumentasi dari kode sumber
<b>4</b>	<code>jar</code>	Untuk mengelola file JAR, bertugas untuk mengemas <i>library class</i> ke dalam satu file JAR
<b>5</b>	<code>jdb</code>	Merupakan suatu <i>debugger</i> digunakan untuk men <i>trace/lacak</i> , program Java, apabila terjadi kesalahan logik pada program. <i>jdb</i> menganalisa dan memunculkan pesan kesalahan ke layar.

Apa yang disebutkan pada tabel di atas, adalah sebagian kecil dari komponen SDK yang ada, dan nantinya akan digunakan langsung pada saat praktek, misalnya komponen *javac* dan *java*.

## F. INSTAL JAVA

Untuk mendapatkan paket instalasi Java , Anda bisa mengunjungi web <http://www.oracle.com/technetwork/java/javase/overview/index.html>. Pada web tersebut anda bisa memilih versi JDK yang diinginkan, termasuk menyesuaikan versi komputer yang anda gunakan apakah 32bit atau 64bit. Lebih bijak jika Anda menyesuaikan dengan detail spesifikasi komputer anda, misalnya dari segi memori, CPU dan media penyimpanan yang tersedia. Pada buku ini menggunakan versi JDK8 untuk sistem operasi Windows. Dengan berjalannya waktu tidak menutup kemungkinan akan muncul versi terbaru.



Gambar 4.6  
Alamat Laman untuk *Download* Kit Java dan Kit Java Hasil *Download*

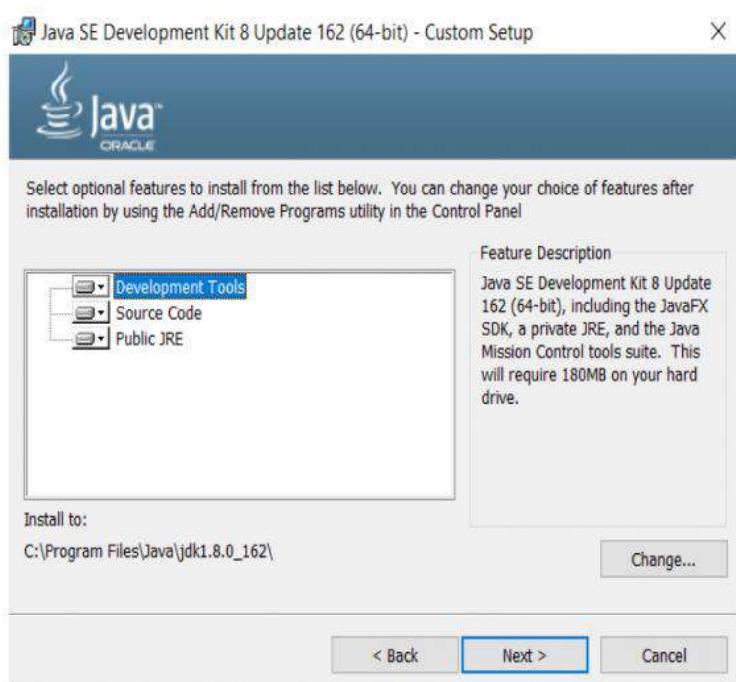
Adapun langkah-langkah instal sebagai berikut:

1. Tampilkan file yang sudah didownload (buku ini adalah jdk-10\_windows-x64.exe)
2. Klik dobel pada file yang disebutkan pada langkah 1, hingga muncul window di bawah, klik tombol ( ) untuk langkah berikutnya.
- 3.



Gambar 4.7  
Window Welcome untuk Setup Java

4. *Window Custom Setup* adalah opsi jika kita mau memindahkan tujuan instalasi. Untuk memilih fitur yang akan diinstalasi, klik tombol (  ) untuk lanjut.

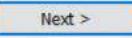


Gambar 4.8  
Pilihan fitur Java yang Akan Diinstal

5. Pada *window progress*, tunggu hingga proses berjalan hingga selesai.



Gambar 4.9  
Progres Instalasi Java

6. Pada window *Custom Setup*, tidak perlu melakukan perubahan *setup*, klik tombol ().



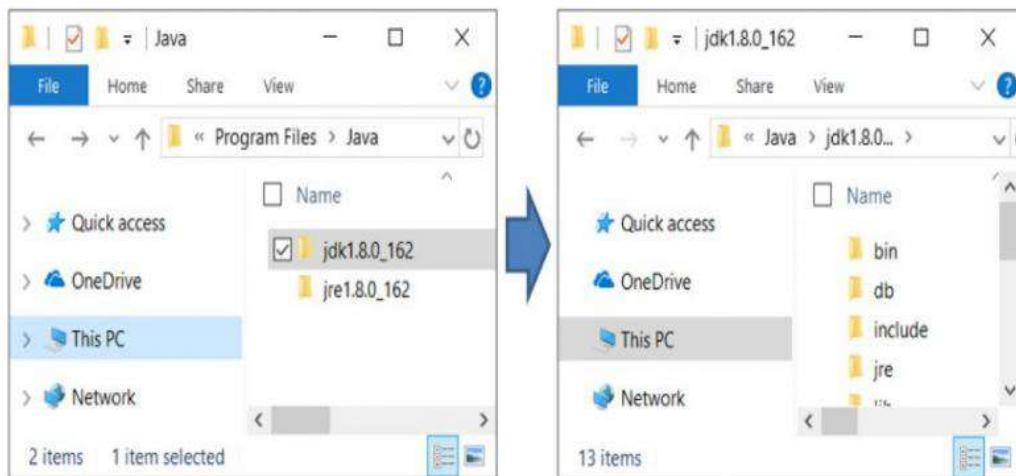
Gambar 4.10  
*Custom Setup* Folder Tujuan Instalasi

7. Window berikutnya adalah *Install Java*, tunggu hingga selesai.



Gambar 4.11  
Progress Instalasi Java dalam Pendistribusian File-file Java

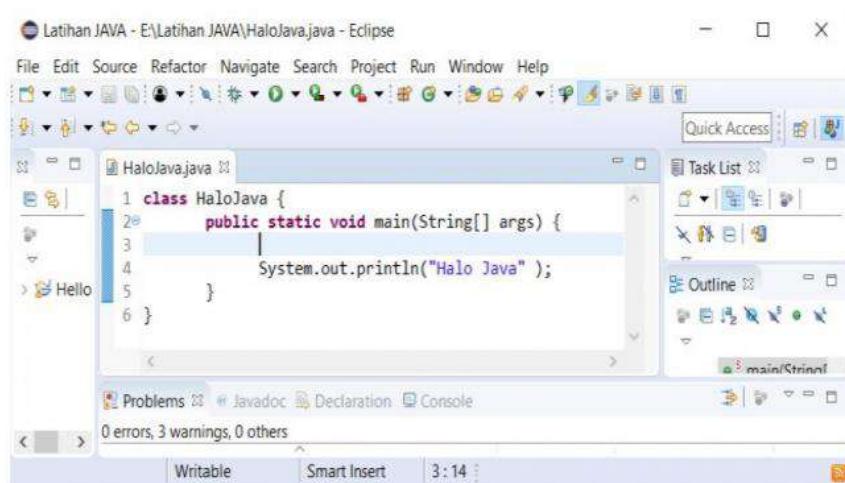
8. Jika sudah muncul *window Complete*, artinya Java sudah sukses diinstal. Klik tombol *Close* untuk menyelesaikan proses instalasi.



Gambar 4.12  
Folder-foler Fitur Bentukan Hasil Instalasi Java

## G. EDITOR PEMROGRAMAN JAVA

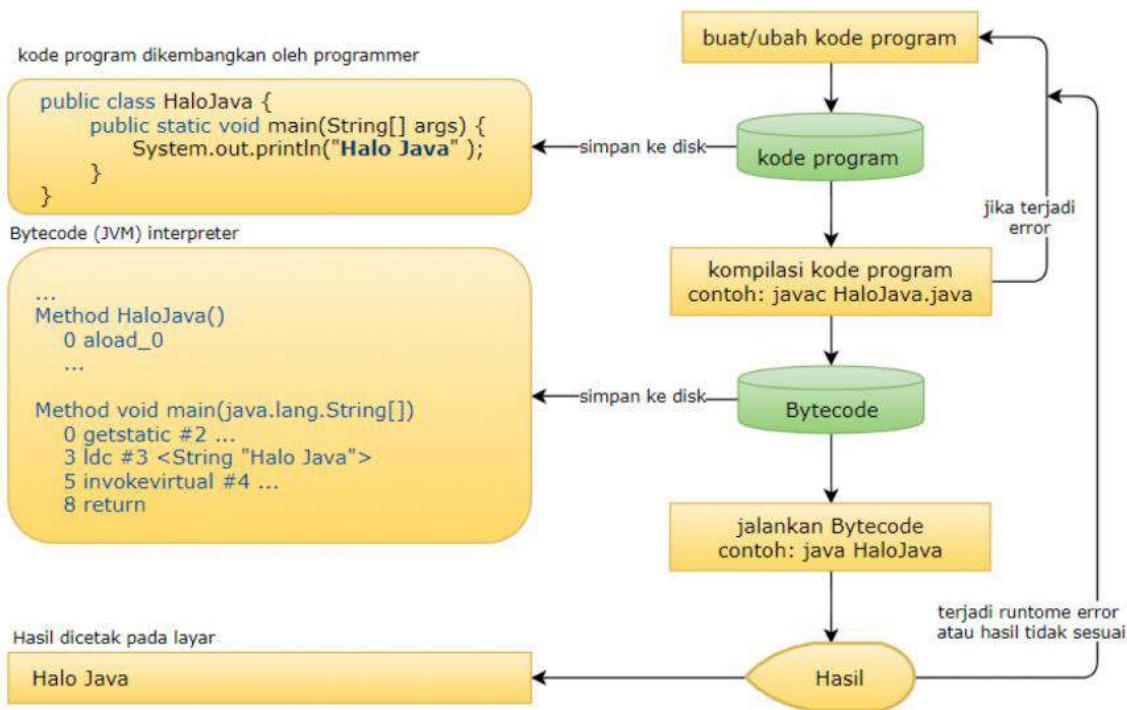
Dalam menulis program Java, diberikan kebebasan dalam menentukan editor teks atau IDE apa saja yang penting editor bisa membentuk file dengan ekstensi *.java*. Pada Windows kita bisa menggunakan editor teks Wordpad, Notepad atau bisa mendownload Notepad++ (Editor ini digunakan untuk beberapa contoh). Jika Anda ingin menggunakan IDE tools Java, mungkin IDE Netbeans dan Eclipse bisa menjadi pilihan.



Gambar 4.13  
Software Eclipse untuk Editor Java

## H. PENULISAN, KOMPILASI DAN EKSEKUSI PROGRAM JAVA

Pembuatan suatu aplikasi melalui beberapa tahap yakni tahap penulisan kode program, kompilasi dan kemudian menjalankan program Java.



Gambar 4.14  
Ilustrasi Proses Pembuatan Hingga Eksekusi Program Java

Langkah-langkah pada Gambar 4.14 dapat diterangkan sebagai berikut:

1. Program ditulis dan kemudian menjadi kode program dan diberi nama (contoh *HaloJava.java*);
2. Perintah *javac* akan menghasilkan file dengan ekstensi *.class* (contoh: *HaloJava.class*) dan tersimpan pada folder yang sama dengan file *.java*. Apabila terjadi kesalahan sintaks, maka saat proses kompilasi akan muncul pesan *error*; dan harus harus diperbaiki dengan cara kembali ke editor kemudian mengedit file *.java* tersebut, untuk file yang sudah tidak memiliki kesalahan saat kompilasi, maka program tersebut dapat dijalankan dengan perintah *java* (contoh: *java HaloJava*). Apabila tidak terjadi *runtime error* atau kesalahan hasil maka proses kompilasi dianggap berhasil. Namun apabila terjadi kesalahan *runtime error* atau kesalahan hasil, maka program harus diperbaiki dengan cara mengedit file *.java* kembali dan kompilasi ulang.

## I. APLIKASI PERTAMA JAVA

Untuk mempercepat pemahaman tentang mudahnya membuat kode program Java, di bawah ini diberikan contoh program sederhana dan lengkap dengan buka tutup blok *class* dan buka tutup blok *method*.

```
public class HaloJava {←
    public static void main(String[] args) {←
        System.out.println("Halo Java");   blok method   blok class
    } ←
}
```

Gambar 4.15  
Sebuah *Class*

Untuk membuat aplikasi pertama Anda, ikuti langkah- langkah berikut:

1. Pastikan program aplikasi Java dan Notepad++ telah terinstal di komputer yang digunakan, jika belum terinstal, maka ikuti langkah bagian F, dan untuk Notepad++, Anda bisa mengunduh di laman: <https://notepad-plus-plus.org>;
2. Buatlah file baru pada editor Notepad++;
3. Tuliskan kode program berikut:

```
public class HaloJava {
    public static void main(String[] args) {
        System.out.println("Halo Java");
    }
}
```

4. Simpan dengan nama file *HaloJava.java* (misal: Disimpan di folder *ModulJava* pada drive “E:”);
5. Buka *command line* untuk Sistem Operasi Windows, dan masukkan ke folder “*ModulJava*” pada drive “E:”, kemudian berikan perintah “*dir*” untuk memastikan file *HaloJava.java* ada.

```
E:\ModulJava>dir
Volume in drive E is Server
Volume Serial Number is AE04-9E1B

Directory of E:\ModulJava

16/04/2018  13.56    <DIR>      .
16/04/2018  13.56    <DIR>      ..
15/04/2018  18.41          141 HaloJava.java
                  1 File(s)   141 bytes
                  2 Dir(s)  83.660.185.600 bytes free
```

6. Kemudian berikan perintah *javac HaloJava.java* dan tunggu hingga kompiler menyelesaikan tugasnya;

```
E:\ModulJava>javac HaloJava.java
```

7. Untuk memastikan bahwa proses kompilasi telah selesai adalah adanya file baru dengan nama file *HaloJava.class*.

```
E:\ModulJava>dir
Volume in drive E is Server
Volume Serial Number is AE04-9E1B

Directory of E:\ModulJava

16/04/2018  13.58    <DIR>      .
16/04/2018  13.58    <DIR>      ..
16/04/2018  13.58          419 HaloJava.class
15/04/2018  18.41          141 HaloJava.java
                  2 File(s)   560 bytes
                  2 Dir(s)  83.660.185.600 bytes free
```

8. Untuk melihat hasil dari program ketikkan perintah: *java HaloJava*.

```
E:\ModulJava>java HaloJava  
Halo Java
```

Untuk melihat status *error* dalam program Java yang dibuat di atas, lakukan perubahan kode program berikut:

```
public class HaloJava {  
    public static void main(String[] args) {  
        System.out.println("Halo Java");  
    }  
}
```

menjadi:

```
public class HaloJava {  
    public static void main(String[] args) {  
        system.out.println("Halo Java");  
    }  
}
```

Perubahan yang dilakukan hanya sedikit sekali dari tulisan *System* menjadi *system* (merubah huruf depan dari huruf besar menjadi huruf kecil). Setelah itu lakukan kompilasi dan lihat hasilnya:

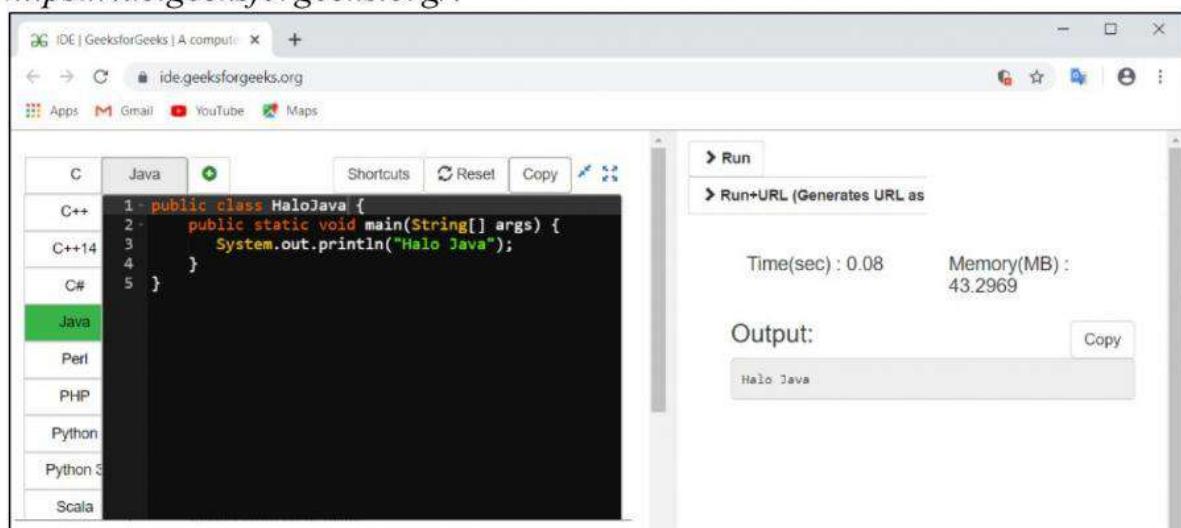
```
E:\ModulJava>javac HaloJava.java  
HaloJava.java:4: error: package system does not exist  
    system.out.println("Halo Java" );  
                           ^  
1 error
```

*Error* yang ditampilkan adalah *HaloJava.java:4: error: package system does not exist*, artinya bahwa pada baris ke 4 pada *HaloJava.java* paket *system* tidak ada, sejatinya adalah *System* dengan huruf awal besar bukan *system* dengan huruf awal kecil, dan dengan jumlah *error* ditampilkan paling

bawah, dan *error* ditunjuk dengan tanda (^). Untuk menangani hal ini, diharuskan kembali membuka file *HaloJava.java* melalui editor dan kemudian lakukan perubahan sesuai petunjuk *error* di atas.

## J. KOMPILASI PROGRAM JAVA SECARA *ONLINE*

Selain *tools* IDE untuk desktop, beberapa website telah menyediakan editor, kompilasi, dan menjalankan program Java secara *online*. Salah satu laman yang menyediakan Java secara online adalah laman <https://ide.geeksforgeeks.org/>.



Gambar 4.16  
Laman ide.geeksforgeeks.org

Bukalah *browser* yang tersedia di komputer/laptop anda, pada alamat *browser*, ketikkan alamat laman di atas, dan selanjutnya bisa mengikuti langkah-langkah berikut:

1. Pastikan laman [ide.geeksforgeeks.org](https://ide.geeksforgeeks.org/) terbuka.
2. Pada halaman laman pilih/klik menu Java.
3. Setelah Editor Java muncul, Anda sudah bisa mengetikkan kode program Java.

A screenshot of an online Java code editor. The interface has a dark theme. At the top left, it says "Java". To its right is a green button with a white plus sign. Below this, the code for a "HaloJava" class is displayed:

```
1 - public class HaloJava {  
2 -     public static void main(String[] args) {  
3 -         System.out.println("Halo Java");  
4 -     }  
5 - }
```

Gambar 4.17  
Editor Program Java Versi *Online*

4. Untuk mengkompilasi/menjalankan program klik tombol ()
5. Lihat hasilnya pada bagian *Output*.

A screenshot of the "Output" window from the online Java editor. The title "Output:" is at the top left. To its right is a "Copy" button. Below the title, the word "Halo Java" is displayed in a text area.

Gambar 4.18  
Output Program Versi *Online*

Beberapa fasilitas pada laman ide.geekforgeeks.org bisa dieksplorasi lebih lanjut.



## LATIHAN

Untuk memperdalam pemahaman Anda mengenai materi di atas, kerjakanlah latihan berikut!

- 1) Temukan 5 laman yang bisa mengkompilasi/menjalankan Java secara *online*, kemudian ujicoba kode program di bawah:

```
public class HaloJava {  
    public static void main(String[] args) {  
        System.out.println("Halo Java");  
    }  
}
```

### Petunjuk Jawaban Latihan

- 1) Untuk menemukan 5 laman yang bisa menjalankan program Java secara *online*, Anda bisa menggunakan laman Google untuk mencari, cukup memasukkan:
  - a. Dengan kalimat “menjalankan program java secara online” dan hasilnya:

menjalankan program java secara online

All Images News Videos Maps More Settings Tools

About 729,000 results (0.41 seconds)

**#10 Compiler Dan Editor Online Java Terbaik - Bahasa Java**  
bahasajava.com > 10-compiler-dan-editor-online-java-te... ▾ Translate this page  
Fitur yang ditawarkan oleh kompiler Java online dapat bervariasi, mulai dari ... kode java secara online dan juga menjalankan kode yang dikompilasi untuk ... mengeksekusi, menyimpan dan berbagi program Java secara online di browser.

**Tutorial Belajar Java: Cara Menjalankan kode Program Java ...**  
www.duniaikom.com > Tutorial Java ▾ Translate this page  
Oct 30, 2019 - Dalam lanjutan tutorial belajar Java di Duniaikom, kita akan masuk ke praktik cara menjalankan kode program Java atau cara men-compile ...

b. Dengan kalimat “*running java online*” dan hasilnya:

The screenshot shows a search results page from a web browser. The search bar at the top contains the query "running java online". Below the search bar, there are tabs for "All", "Videos", "News", "Images", "Maps", and "More", with "All" being the selected tab. To the right of the search bar are settings and tools icons. The search results section starts with a summary: "About 166,000,000 results (0.49 seconds)". The first result is a link to "Online Java Compiler - online editor - OnlineGDB" with the URL "www.onlinedb.com > online\_java\_compiler". The description for this result is: "OnlineGDB is online IDE with java compiler. Quick and easy way to run java program online.". The second result is "Online Java Compiler - Tutorialspoint" with the URL "www.tutorialspoint.com > compile\_java\_online". The description is: "Online Java Compiler, Online Java Editor, Online Java IDE, Java Coding Online, Practice Java Online, Execute Java Online, Compile Java Online, Run Java ...". The third result is "Online Java IDE (javac 1.8.0\_201)" with the URL "www.compilejava.net". The description is: "Simple, fast and secure Online Java IDE / Compiler ... arguments are passed using the text field below this editor. 12. public static void main(String[] args). 13. ....". The fourth result is "Online Java Compiler - Online Java Editor - Java Code Online" with the URL "www.jdoodle.com > online-java-compiler". The description is: "Online Java Compiler - Online Java Editor - Online Java IDE - Java Coding Online - Online Java Runner - Share Save Java online. .... XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX. Sponsored: Thanks for using our ...". The fifth result is "Online Java Editor and IDE - Fast, Powerful, Free - Repl.it" with the URL "repl.it > languages > java10". The description is: "Online Java Compiler, Online Java Editor, Online Java IDE, Online Java REPL, Online Java Coding, Online Java Interpreter, Execute Java Online, Run Java ...".



RANGKUMAN

Perusahaan *software* terkenal Tiobe melakukan survei terhadap 20 bahasa pemrograman, dan hasilnya adalah memberikan hasil akhir bahwa bahasa pemrograman Java menempati urutan teratas dengan peminat 15.777%. Data tersebut diumumkan pada April 2018. Banyak alasan kenapa pemrograman Java diminati, yaitu: 1) Tren teknologi, ternyata Java tetap stabil dengan OOP. 2) Banyak digunakan perusahaan besar. 3) Efisiensi waktu dan biaya.

Versi awal bahasa pemrograman Java dirilis pada tahun 1996, dan terus berkembang hingga disebut dengan Java2. Kelebihan Java yang paling mencolok adalah bahasa pemrograman dengan *multiplatform*, mempunyai *library* yang lengkap dan konsep OOP. Di sisi lain pemrograman Java juga mempunyai kelemahan, diantaranya: ketika

berada pada *platform* lain seperti Mac OS X, ada beberapa fitur Java tidak tersedia, boros memori, dan mudah didekompilasi.

Java dengan mudah diunduh dari Internet, demikian juga dengan editornya, semua tersedia dengan gratis di internet, misalnya Notepad++, IDE NetBeans dan Eclipse. Dengan editor yang sederhana tersebut, *programmer* dengan mudah menuliskan program Java. Kompilasi dan eksekusi program Java dapat dilakukan dengan mudah dengan melalui *command-line* dengan tools *javac* untuk kompilasi dan tools *java* untuk menjalankan hasil kompilasi. Program Java juga bisa dikompilasi/dijalankan secara *online*, misalnya pada laman <https://ide.geeksforgeeks.org/>.



#### TES FORMATIF 1

---

Pilihlah satu jawaban yang paling tepat!

- 1) Desain pemrograman dengan desain *top-down* adalah konsep pendekatan bahasa pemrograman terstruktur, pola ini pertama kali diperkenalkan oleh....
  - A. Corrado Böhm dan Giuseppe Jacopini
  - B. Edsger W. Dijkstra
  - C. Blase Pascall
  - D. James Watt
- 2) Bahasa pemrograman yang menerapkan konsep pemrograman terstruktur adalah....
  - A. Cobol
  - B. C++
  - C. Delphi
  - D. Jawa
- 3) Pemrograman Berorientasi Objek pertama kali diperkenalkan pada tahun 1960-an. Bahasa pemrograman yang pertama menggunakan konsep tersebut adalah....
  - A. Ada
  - B. C++
  - C. Simula
  - D. Java

- 4) Perusahaan yang memperkarsai lahirnya bahasa pemrograman Java adalah....
  - A. Microsoft
  - B. Apple
  - C. Xerox
  - D. Sun Microsystem
- 5) Software yang bisa digunakan untuk editor Java adalah, kecuali....
  - A. Notepad++
  - B. Worpad
  - C. Notepad
  - D. Power Point
- 6) Fungsi dari komponen Java yang bernama *jdb* adalah....
  - A. Untuk mengelola file JAR, bertugas untuk mengemas *library* kelas kedalam satu file JAR
  - B. *Loader* untuk aplikasi Java, digunakan sebagai penerjemah dan dapat menginterpretasikan file kelas yang dihasilkan oleh kompiler javac
  - C. Digunakan untuk men-*debugger* program Java, jika terjadi kesalahan pada program, maka *jdb* akan menganalisa dan memunculkan pesan kesalahan ke layar
  - D. Generator dokumentasi, menghasilkan dokumentasi dari kode sumber
- 7) Yang berkaitan dengan OOP adalah....
  - A. *Inheritance, encapsulation, abstraction*
  - B. *Class, Object, Method, Trigger*
  - C. *Trigger, encapsulation, Object*
  - D. *Class, Variabel, Behaviour, Kompilasi*
- 8) Sun Microsystem berhasil meluncurkan sebuah browser Java yang diberi nama....
  - A. Netscape
  - B. Hot Java
  - C. Opera
  - D. Maxthon
- 9) Browser Netscape berhasil mengadopsi teknologi Java pada....
  - A. Januari 1996
  - B. Februari 1996

- C. Maret 1996
  - D. April 1996
- 10) Kepanjangan dari JDK adalah....
- A. Java Deploy Kit
  - B. Java Development Kit
  - C. Java Developer Kit
  - D. Java Description Kit

Cocokkanlah jawaban Anda dengan Kunci Jawaban Tes Formatif 1 yang terdapat di bagian akhir modul ini. Hitunglah jawaban yang benar. Kemudian, gunakan rumus berikut untuk mengetahui tingkat penguasaan Anda terhadap materi Kegiatan Belajar 1.

$$\text{Tingkat penguasaan} = \frac{\text{Jumlah Jawaban yang Benar}}{\text{Jumlah Soal}} \times 100\%$$

Arti tingkat penguasaan:  $90 - 100\% = \text{baik sekali}$

$80 - 89\% = \text{baik}$

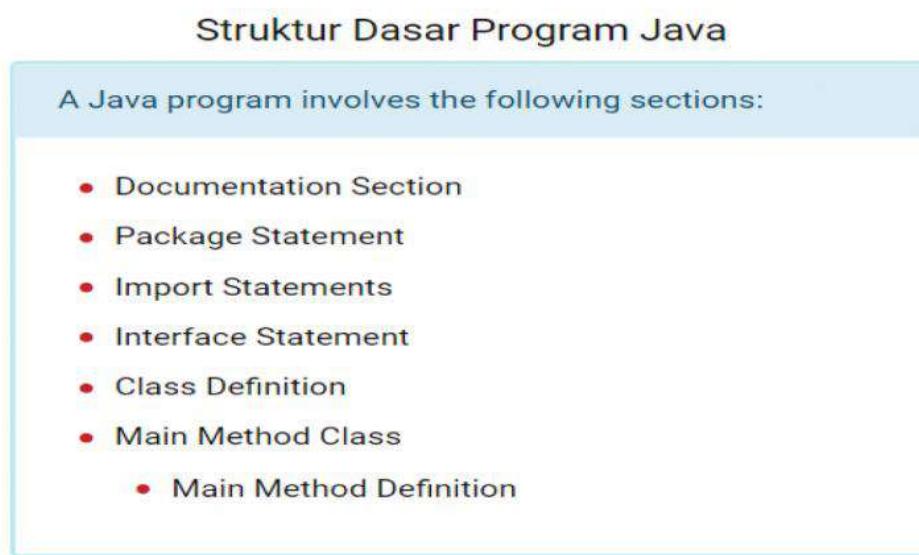
$70 - 79\% = \text{cukup}$

$< 70\% = \text{kurang}$

Apabila mencapai tingkat penguasaan 80% atau lebih, Anda dapat meneruskan dengan kegiatan belajar 2. **Bagus!** Jika masih di bawah 80%, Anda harus mengulangi materi Kegiatan Belajar 1, terutama bagian yang belum dikuasai.

**KEGIATAN BELAJAR 2****Struktur Program Java**

Struktur program Java merupakan format standar yang dirilis oleh pengembang Java (Sun Microsystem). Setiap bahasa pemrograman mempunyai struktur yang berbeda, namun ada beberapa memiliki kemiripan, khususnya bagi yang menerapkan konsep OOP. Adapun struktur program Java dapat dilihat pada gambar berikut:



Gambar 4.19  
Struktur Dasar Program Java

**A. DOCUMENTATION SECTION (BAGIAN DOKUMENTASI /KOMENTAR)**

Komentar sangat bermanfaat bagi *programmer* karena membantu pengguna untuk dapat memahami program yang dibuat. Penggunaan komentar atau penjelasan singkat untuk setiap baris atau blok tertentu membuat program tersebut makin mudah untuk dapat dipahami. Apalagi jika kode program dibuat oleh *programmer* lain dan diwariskan kepada *programmer* berikutnya. Walau komentar bersifat opsional, namun sangat disarankan untuk tetap menggunakannya. Pada bahasa pemrograman Java tersedia tiga macam bentuk penulisan komentar, yakni :

- Menggunakan bentuk /\* ... \*/, kompilator akan mengabaikan segala sesuatu dari /\* ke \*/.

Contoh 4.5:

```
...
/* Komentar ini tak akan dieksekusi */
...
```

- Menggunakan bentuk `/** ... */`, ini disebut juga dengan komentar dokumentasi. Kompilator akan mengabaikan semua apa yang ada pada/mulai dari tanda `/**` sampai ke `*/`, komentar ini bisa lebih dari 1 baris,

Contoh 4.6:

```
...
/** Komentar ini tak akan dieksekusi
 * termasuk baris ini
 * dan baris ini juga
 */
...
```

- Menggunakan bentuk `//`, Kompilator akan mengabaikan semua teks yang mulai dari tanda `//`, berlaku per baris.

Contoh 4.7:

```
...
// Komentar ini tak akan dieksekusi
...
```

Penempatan komentar bebas ditempatkan dimana saja, boleh di dalam blok program sekalipun. Namun tetap harus memenuhi kaidah pemrograman, agar program tidak memberikan pesan kesalahan.

## B. PACKAGE STATEMENT (PERNYATAAN PAKET)

Paket (*package*) adalah cara untuk mengelola sekelompok *class*, *interface*, atau sub-paket yang didefinisikan dengan sebuah nama tertentu

berdasarkan kesamaan atau kemiripan fungsinya, *programmer* bisa membuat nama paket dengan nama apapun yang mewakili fungsi sebuah paket. Mengingat sifatnya yang opsional (tidak wajib) maka penggunaannya sangat tergantung kebutuhan. Pada saat instal Java, banyak paket-paket bawaan yang sudah disediakan oleh *developer* Java, misalnya: *java.io* (paket yang digunakan untuk *input* dan *output*), *java.lang.\** (Paket ini berisi kelas-kelas dan *interfaces* yang diperlukan oleh banyak program Java. Paket ini di *import* oleh kompiler ke dalam semua program Java secara otomatis), *java.text.\** (Paket berisi *class-class* dan *interfaces* yang memungkinkan memanipulasi angka, tanggal, karakter dan juga *string*).

Bentuk fisik dari suatu paket antara lain berupa folder yang di dalamnya terdapat *file/class*, *interface/enum* dan dikelompokkan berdasarkan kemiripan. Dalam membuat paket terdapat 3 pokok yang harus diperhatikan, yaitu:

- Pendeklarasian dan pemberian nama paket, deklarasi nama paket diletakkan di awal sebelum kode program lainnya;
- Membuat struktur dan nama folder berdasarkan kemiripan fungsi;
- Kompilasi *class-class* sesuai dengan nama paket.
- Penamaan paket harus mengikuti kaidah yang ditetapkan pemrograman Java, yaitu:
  1. Menggambarkan *class*,
  2. Harus unik,
  3. Mempersentasikan *path* dari paket,
  4. Berada pada direktori yang sama.

Pernyataan paket sebagai berikut:

```
package nama_paket;
```

Misalnya mau membuat paket dengan nama *paketSepeda* maka caranya adalah *package paketSepeda*. Perhatikan paket di bawah ini; paketnya adalah *paketSepeda*, kemudian dibuat pada folder *paketSepeda*, dalam folder tersebut dibuat dua *class* dengan nama file *SepedaGunung.java* dan *SepedaSantai.java*.

Contoh 4.8: Membuat paket dengan nama *paketSepeda* dengan *class SepedaGunung*.

```
package paketSepeda;
public class SepedaGunung {
    public void sepedaGunung() {
        System.out.println("Ini sepeda gunung");
    }
};
```

Contoh 4.9: Membuat paket dengan nama *paketSepeda* dengan *class SepedaSantai*.

```
package paketSepeda;
public class SepedaSantai {
    public void sepedaSantai() {
        System.out.println ("Ini sepeda santai");
    }
};
```

Penggunaan paket tersebut dapat dilakukan dengan 2 cara yakni:

- 1) Paket-paket yang berada pada satu direktori dengan program Java utama tidak perlu menggunakan pernyataan *import*,
- 2) Paket-paket yang berada di luar direktori dengan program Java utama, maka wajib menggunakan pernyataan *import* dan dengan menyebutkan direktori dimana paket-paket disimpan.

### C. IMPORT STATEMENT (PERNYATAAN IMPOR)

Impor (*import*) adalah pernyataan yang digunakan untuk merujuk suatu *method/class* yang dinyatakan dalam paket/*library* lainnya untuk digunakan pada program induk. Pernyataan *import* sebagai berikut:

```
import from_package;
```

Pada bagian sebelum ini kita sudah belajar tentang cara membuat paket. Sekarang bagaimana cara mengimpor paket-paket yang sudah dibuat. Perhatikan program Java mengimpor *class-class* yang sudah dibuat di atas sebagai berikut:

```
import paketSepeda.*;  
  
class Sepeda {  
    public static void main(String[] args) {  
        System.out.println("Jenis Sepeda");  
        System.out.println("-----");  
        //memanggil paket;  
        SepedaGunung sepeda1 = new SepedaGunung();  
        sepeda1.sepedaGunung();  
        SepedaSantai sepeda2 = new SepedaSantai();  
        sepeda2.sepedaSantai();  
    }  
}
```

Penjelasan:

- Pernyataan `import paketSepeda.*;` adalah mengimpor semua file-file/*class-class* yang ada pada folder *paketSepeda* dengan adanya tanda \* dibelakang tanda titik (.). Jika bentuk impornya seperti di atas artinya bahwa program induknya berada pada folder di atasnya, dan contoh strukturnya di bawah.

```

Sepeda.java (nama file class induk nama sesuai nama class)
|
|---paketSepeda (nama folder)
|   |--Sepeda_Gunung.java (nama file)
|   |--Sepeda_Santai.java (nama file)

```

- Pada baris `SepedaGunung sepeda1 = new SepedaGunung();` adalah memanggil *method* dari paket *paketSepeda* dari *class Sepeda*.
- Baris `sepeda1.SepedaGunung` adalah mengerjakan perintah-perintah yang ada pada blok *method sepedaGunung*.

#### D. INTERFACE STATEMENT (PERNYATAAN INTERFACE)

*Interface* seperti *class* pada Java, akan tetapi hanya memiliki konstanta *static* dan *method abstract*. Java menggunakan *interface* untuk memaksimalkan multi *inheritance* (pewarisan). *Class* pada Java dapat mengimplementasikan beberapa *interface*. Semua *method* dalam *interface* secara implisit bersifat *public* dan *abstract*.

Adapun sintaks umum dari pernyataan *interface* sebagai berikut:

```

public interface nama_interface{
    void apapun_namanya();
}

```

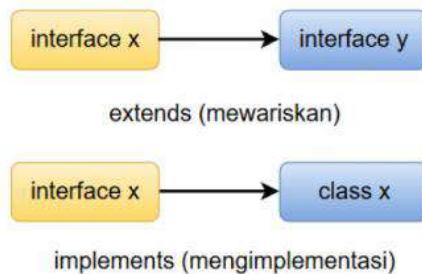
Contoh 4.10:

```

public interface Sepeda {
    public void SepedaGunung();
    public void SepedaSantai();
}

```

Sebuah *interface* dapat diwariskan ke *interface* lainnya dan *class* dan mengimplementasi *interface*.



Gambar 4.20  
*Extends dan Implements Interface*

## E. CLASS DEFINITION (PENDEFINISIAN CLASS)

Kelas (*class*) adalah template untuk membuat objek yang mendefinisikan *state* dan *behavior*. Program Java yang dibuat bisa terdiri dari beberapa *class*. Beberapa pengetahuan dasar tentang pembuatan *class* sebagai berikut:

- *Class* hanya memiliki akses *modifier public* dan *default*.
- *Class public* harus memiliki nama file Java yang sama dengan nama *class*-nya. Contoh: jika nama *class* adalah *Sepeda*, maka nama filenya wajib dengan nama *Sepeda.java*.
- Satu file Java bisa memuat lebih dari 1 *class non-public* dan wajib hanya memiliki 1 *class public*.
- *Class public* dapat dilihat oleh semua *class*.
- *Class* dengan akses *default* hanya dapat dilihat oleh *class* dalam paket yang sama.
- File Java tanpa *class public* tidak memiliki batasan penamaan.

Sintaks umum dari deklarasi *class* sebagai berikut:

```

public class {
    //field, constructor dan
    //deklarasi method
}
  
```

Contoh 4.11:

```
...
public class SepedaGunung{ public void info()
{
    System.out.println ("Ini sepeda gunung");
}
};

...
```



## LATIHAN

---

Untuk memperdalam pemahaman Anda mengenai materi di atas, kerjakanlah latihan berikut!

- 1) Carilah program Java di Internet yang dalam programnya menggunakan komentar.
- 2) Temukan contoh sederhana di Internet program Java yang menggunakan *interface*.

### Petunjuk Jawaban Latihan

- 1) Alternatif 1: Dapat menggunakan Google untuk mencari program Java yang menggunakan komentar dengan keyword : *comment in Java*.  
Alternatif 2: kunjungi lama [www.geeksforgeeks.org](http://www.geeksforgeeks.org), kemudian pada pencarian masukkan keyword “*comment in Java*”.



Gambar 4.21  
Fasilitas Pencarian pada Laman Geeksforgeeks

akan muncul hasil pencarian berikut:

**Comments in Java - GeeksforGeeks**  
<https://www.geeksforgeeks.org/comments-in-java/>

In a program, **comments** take part in making the program become more human readable by placing the detail of code involved and proper use of **comments** ...

**Executable Comments in Java - GeeksforGeeks**  
<https://www.geeksforgeeks.org/executable-comments-jav/>

A **comment** is a statement that is not executed by the compiler or the interpreter, but before the lexical transformation of the program in the compiler, the contents ...

Gambar 4.22  
Hasil Pencarian pada Laman Geeksforgeeks

Klik pilihan “*Executable Comments in Java – GeeksforGeeks*”.

## Executable Comments in Java

A comment is a statement that is not executed by the compiler or the interpreter, but before the lexical transformation of the program in the compiler, the contents of the program are encoded into ASCII to make the processing easier. Consider this program:

```

class Main{
    public static void main(String[] args) {
        // The comment below is magic..
        // \u000d System.out.println("Geek C
    }
}

```

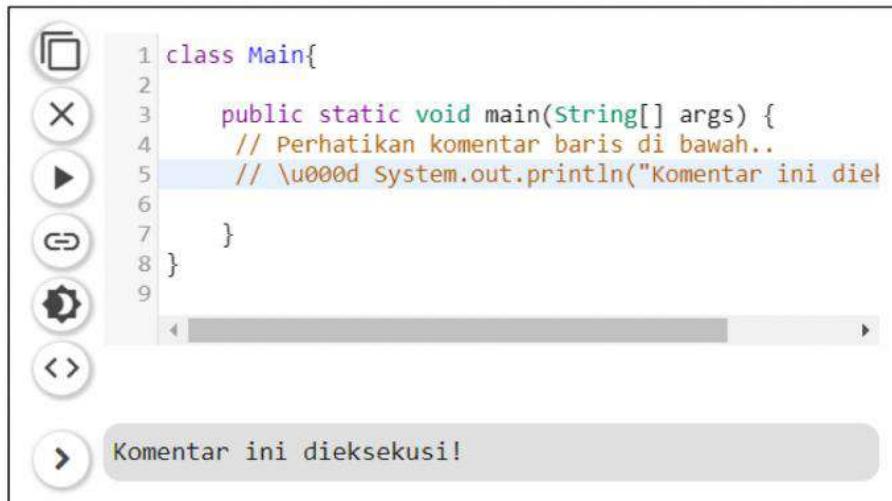
Gambar 4.23  
Contoh Program Komentar pada Geeksforgeeks

Program contoh program di atas bisa diedit dengan klik tombol ( ), misal diedit menjadi:

```
class Main{

    public static void main(String[] args) {
        // Perhatikan komentar baris di bawah..
        // \u000d System.out.println("Komentar ini dieksekusi!");
    }
}
```

Program tersebut bisa dijalankan dengan klik tombol (▶), dan hasilnya sebagai berikut:



The screenshot shows a Java code editor with the following code:

```
1 class Main{
2
3     public static void main(String[] args) {
4         // Perhatikan komentar baris di bawah..
5         // \u000d System.out.println("Komentar ini dieksekusi!");
6
7     }
8 }
```

To the left of the code editor is a vertical toolbar with several icons: a square (close), a circle with an X (cancel), a circle with a play arrow (run), a double arrow (refresh), a gear (settings), and a double-headed arrow (switch). Below the code editor, there is a greyed-out button labeled with a play arrow. At the bottom of the screen, a message box displays the output: "Komentar ini dieksekusi!".

Gambar 4.24  
Program Java Setelah Dieksekusi Laman Geeksforgeeks

Catatan:

*Baris 5 pada program seperti Gambar 4.24, tetap dieksekusi karena karakter \u000d pada Java dianggap sebagai membuat baris baru.*

- 2) Pada laman [www.ide.geeksforgeeks.org](http://www.ide.geeksforgeeks.org) juga terdapat contoh penggunaan interface, buka laman tersebut dan masukkan keyword “interfaces in Java”.

[Interfaces in Java - GeeksforGeeks](https://www.geeksforgeeks.org/interfaces-in-java/)<https://www.geeksforgeeks.org/interfaces-in-java/>

Like a class, an **interface** can have methods and variables, but the methods declared in an **interface** are by default abstract (only method signature, no body).

Gambar 4.25

Hasil Pencarian pada Laman Geeksforgeeks Tentang *Interface*

Contoh dari laman ide.geeksforgeeks.org:

```
// Java program to demonstrate working of
// interface.
import java.io.*;

// A simple interface
interface In1
{
    // public, static and final
    final int a = 10;

    // public and abstract
    void display();
}

// A class that implements the interface.
class TestClass implements In1
{
    // Implementing the capabilities of
    // interface.
    public void display()
    {
        System.out.println("Geek");
    }

    // Driver Code
    public static void main (String[] args)
    {
        TestClass t = new TestClass();
        t.display();
        System.out.println(a);
    }
}
```

Penjelasan:

- *Interface* dengan nama *In1* adalah *interface* sederhana dengan sebuah variabel *a* diisi dengan nilai 10 (`final int a = 10;`).
- Kemudian *interface In1* diimplementasi oleh *class TestClass* (`System.out.println(a);`).



### RANGKUMAN

---

Standarisasi program Java yang dirilis oleh pengembang Java (Sun Microsystem) meliputi bagian dokumentasi, pernyataan paket, pernyataan impor, pernyataan interface dan pendefinisian *class*.

Bagian dokumentasi/komentar bermanfaat bagi *programmer* dalam memberikan penjelasan singkat untuk sebuah perintah atau blok perintah pada program Java. Pernyataan paket adalah cara mengelola sekelompok *class*, *interface*, atau sub-paket yang didefinisikan dengan sebuah nama tertentu berdasarkan kesamaan atau kemiripan fungsi. Beberapa paket bawaan Java, misalnya: *java.io* merupakan paket yang digunakan untuk mengatur *input* dan *output*; *java.lang.\** merupakan paket yang berisi kelas-kelas dan *interfaces* yang diperlukan oleh banyak program java. Paket di *import* oleh kompiler ke semua program Java secara otomatis; *java.text.\** merupakan paket berisi *class-class* dan *interfaces* yang memungkinkan memanipulasi angka, tanggal, karakter dan juga *string*. Pernyataan *import* digunakan untuk merujuk suatu *method/class* yang dinyatakan dalam suatu paket (*library*) lain. Pernyataan *interface* adalah tipe referensi yang mengandung sekumpulan *method*. *Class* adalah template untuk membuat objek yang mendefinisikan *state* dan *behavior*.



### TES FORMATIF 2

---

Pilihlah satu jawaban yang paling tepat!

- 1) Komentar dokumentasi yang tepat di bawah ini adalah....  
 A. //  
 B. /\* ke \*/  
 C. /\*\* .. \*/  
 D. /\* ..
  
- 2) Paket *java.io* digunakan untuk....  
 A. Keperluan Kompilasi  
 B. *Input* dan *output*

- C. Manipulasi *String*
  - D. Fungsi matematika
- 3) *Syntaks* pernyataan paket yang benar berikut ini yang diikuti oleh nama paket sepeda adalah....
- A. paket sepeda;
  - B. Sepeda *package*;
  - C. *package* Sepeda;
  - D. name *package* Sepeda;
- 4) Sebuah pernyataan paket harus mengikuti aturan/kaidah dari pemrograman Java, empat aturan/kaidah dalam penamaan paket yang benar adalah....
- A. 1. Menggambarkan *class*  
2. Tidak harus unik  
3. Mempersentasikan *path* dari paket  
4. Berada pada *direktory* yang sama
  - B. 1. Menggambarkan *class*  
2. Tidak harus unik  
3. Mempersentasikan *path* dari paket  
4. Tak perlu berada pada *direktory* yang sama
  - C. 1. Nama boleh mengandung spasi  
2. Tidak harus unik  
3. Mempersentasikan *path* dari paket  
4. Berada pada *direktory* yang sama
  - D. 1. Menggambarkan kelas  
2. Harus unik  
3. Mempersentasikan *path* dari paket  
4. Berada pada *direktory* yang sama
- 5) Pernyataan paket import paketSepeda.\*; memberikan asumsi bahwa....
- A. Mengimpor sebagian dari semua file/*class-class* yang ada di folder paketSepeda
  - B. Mengimpor *class-class* tertentu saja dan terbatas pada folder paketSepeda
  - C. Mengimpor semua file/*class-class* yang ada pada folder paketSepeda
  - D. Tidak melakukan impor sama sekali

- 6) Pernyataan yang benar dari sebuah *interface* adalah....
- Sebuah *interface* dapat diwariskan ke *interface* yang lain
  - Sebuah *interface* tidak dapat diwariskan ke *interface* yang lain
  - Sebuah *class* tidak dapat mengimplementasi sebuah *interface*
  - Sebuah *interface* dapat mengimplementasi sebuah *class*
- 7) Kaidah penamaan *class public* yang tepat berikut ini adalah....
- Penamaan file bebas walaupun tidak sesuai dengan nama *class*.
  - Penamaan file tidak perlu berekstensi .java
  - Penamaan file harus persis sama dengan nama *method* atau *constructor* dalam *class*.
  - Penamaan file java harus sama dengan nama *class*-nya
- 8) Perhatikan potongan *script* berikut:
1. public class Sepeda\_Gunung{ public void info()
  2. {
  3. System.out.println ("Ini sepeda gunung");
  - 4.
  5. };

Agar *script* di atas berjalan pada saat dikompilasi, maka pada baris ke 4 harus dilengkapi dengan....

- {
  - ()
  - };
  - }
- 9) Maksud dari pernyataan `import paketSepeda.*;` adalah....
- Mengimpor beberapa file atau *class* yang berekstensi \*
  - Mengimpor file/*class* yang ada dalam paket `paketSepeda`.
  - Tidak memberikan ruang untuk mengimpor
  - Mengimpor file/*class* tertentu saja.
- 10) Maksud dari pernyataan `//import paketSepeda.*;` adalah....
- Mengimpor beberapa file atau *class* yang berekstensi \*
  - Mengimpor semua file dalam `paketSepeda`.
  - Tidak memberikan ruang untuk mengimpor
  - Pernyataan impor tidak berlaku karena diawali dengan simbol //.

Cocokkanlah jawaban Anda dengan Kunci Jawaban Tes Formatif 2 yang terdapat di bagian akhir modul ini. Hitunglah jawaban yang benar. Kemudian, gunakan rumus berikut untuk mengetahui tingkat penguasaan Anda terhadap materi Kegiatan Belajar 2.

$$\text{Tingkat penguasaan} = \frac{\text{Jumlah Jawaban yang Benar}}{\text{Jumlah Soal}} \times 100\%$$

Arti tingkat penguasaan: 90 - 100% = baik sekali

80 - 89% = baik

70 - 79% = cukup

< 70% = kurang

Apabila mencapai tingkat penguasaan 80% atau lebih, Anda dapat meneruskan dengan Kegiatan Belajar 3. **Bagus!** Jika masih di bawah 80%, Anda harus mengulangi materi Kegiatan Belajar 2, terutama bagian yang belum dikuasai.

**KEGIATAN BELAJAR 3****Tipe Data dan Variabel**

**V**ariabel adalah lokasi memori yang disediakan untuk menyimpan nilai pada komputer. Ketika kita membuat variabel, berarti kita melakukan pemesanan ruang di memori komputer. Sistem operasi mengalokasikan memori sesuai atau berdasarkan tipe data yang dideklarasikan pada variabel. Untuk sebuah variabel, anda bebas menentukan tipe data variabel sesuai dengan kebutuhan, apakah bilangan bulat (integer), desimal, atau karakter. Pada pemrograman Java dikenal dua tipe data, yaitu: 1) Tipe data primitif, dan 2) Tipe data referensi/objek.

**A. TIPE DATA PRIMITIF**

Ada delapan tipe data primitif yang didukung oleh Java. Tipe data primitif telah ditentukan oleh Java dan *keyword* masing-masing. Kedelapan tipe data tersebut adalah: *byte*, *short*, *int*, *long*, *float*, *double*, *char*, dan *boolean*.

Kedelapan tipe data yang sudah disebutkan dikelompokkan berdasarkan kesamaan ke dalam 4 kelompok, yaitu: tipe data bilangan bulat, tipe data bilangan real (desimal), tipe data karakter, dan tipe data logika. Yang termasuk ke dalam tipe data bilangan bulat adalah: *byte*, *short*, *int*, dan *long*. Untuk tipe data bilangan real adalah: *long* dan *float*. Tipe data karakter adalah *char* dan untuk tipe data logika yaitu *boolean* (*true/false*).

**1. Tipe Data Bilangan Bulat (integer)**

Jangkauan nilai untuk 4 tipe data bilangan bulat berbeda-beda. Adapun tipe data untuk bilangan bulat sebagai berikut:

a. Tipe data *byte*

- Tipe data byte merupakan bilangan bulat (integer) 8-bit yang *signed*.
- Nilai minimum adalah -128.
- Nilai maksimum adalah 127.
- Nilai awal adalah 0 (jika tidak diisi pada saat dideklarasikan pada variabel).

- Tipe data *byte* digunakan untuk menghemat ruang, karena ruang yang dibutuhkan lebih kecil, dan sering digunakan untuk *array*.
  - Contoh: *byte a = 50, byte b = -20.*
- b. Tipe data *short*
- Tipe data *short* adalah tipe data integer dengan 16-bit yang *signed*.
  - Nilai minimum adalah -32.768.
  - Nilai maksimum adalah 32.767.
  - Tipe data singkat juga dapat digunakan untuk menyimpan memori sebagai tipe data *byte*.
  - Nilai awal adalah 0.
  - Contoh: pendek s = 13500, r pendek = -20034.
- c. Tipe data *int*
- Tipe data *byte* merupakan bilangan bulat (*integer*) 32-bit yang *signed*.
  - Nilai minimun adalah - 2,147,483,648.
  - Nilai maksimum adalah 2,147,483,647.
  - Nilai awal adalah 0.
  - Contoh: *byte a = 100050, byte b = -210000.*
- d. Tipe data *long*
- Tipe data *long* merupakan bilangan bulat (*integer*) 64-bit yang *signed*.
  - Nilai minimun adalah -9,223,372,036,854,775,808.
  - Nilai maksimum adalah 9,223,372,036,854,775,807.
  - Nilai awal adalah 0.
  - Contoh: *byte a = 1000506666, byte b = -21000078788.*

Contoh 4.12. Penggunaan tipe data bilangan dalam pemrograman Java

```
public class BilanganBulat {  
    public static void main(String[] args) {  
        //Deklarasi  
        byte tipeByte = 50;  
        short tipeShort = 0x7e23;  
        int tipeInt = 21466666;  
        long tipeLong = 922000000;  
        //Cetak  
        System.out.println("Bilangan Bulat ");  
        System.out.println("-----");
```

```

        System.out.println("Tipe Byte tipeByte = " + tipeByte );
        System.out.println("Tipe Short tipeShort      =      "+
tipeShort );
        System.out.println("Tipe Int tipeInt       =      "+
tipeInt );
        System.out.println("Tipe Long tipeLong     =      "+
tipeLong );
    }
}

```

Keluaran:

Bilangan Bulat	-----
Tipe Byte tipeByte	= 50
Tipe Short tipeShort	= 32291
Tipe Int tipeInt	= 21466666
Tipe Long tipeLong	= 922000000

e. Tipe Data *Real* (Desimal)

Ada 2 tipe data bilangan real. Perbedaan masing-masing tipe data terletak pada jangkauan nilai yang dicakup.

- ✓ Tipe data *float*
  - Tipe data *float* dengan presisi tunggal dengan nilai desimal 32-bit.
  - Nilai awal adalah 0.0f.
  - Contoh: *float* af = 156.1f.
- ✓ Tipe data *double*
  - Tipe data *double* dengan presisi tunggal dengan nilai desimal 64-bit.
  - Biasa digunakan pada *array* dengan nilai-nilai desimal yang kecil untuk penghematan.
  - Nilai awal adalah 0.0d.
  - Contoh: *double* af = 156.1d.

Contoh 4.13. Tipe data real dalam pemrograman Java

```

public class BilanganReal {
    public static void main(String[] args) {
        //Deklarasi float
        float tipeFloat1 = 0.13f;
        float tipeFloat2 = .25f;
        //Deklarasi double
    }
}

```

```

double tipeDouble1 = 0.4e-42;
double tipeDouble2 = 432.12D;
//Cetak
System.out.println("Bilangan Real - Desimal ");
System.out.println("-----");
System.out.println("Tipe Float tipeFloat1      = "+tipeFloat1);
System.out.println("Tipe Float tipeFloat2      = "+tipeFloat2);
System.out.println("Tipe Double tipeDouble1     = "+tipeDouble1);
System.out.println("Tipe Double tipeDouble2     = "+tipeDouble2);
}
}

```

Keluaran:

```

Bilangan Real - Desimal
-----
Tipe Float tipeFloat1      = 0.13
Tipe Float tipeFloat2      = 0.25
Tipe Double tipeDouble1    = 4.0E-43
Tipe Double tipeDouble2    = 432.12

```

#### f. Tipe Data *Char*

Tipe data *char* adalah menyatakan sebuah karakter meliputi karakter apa saja yang ada pada himpunan kode (*unicode*).

- ✓ Tipe data *char*

- Tipe data *char* adalah karakter *unicode* dengan 16-bit tunggal.
- Nilai minimum adalah '\u000000'.
- Nilai maksimum adalah '\uffff'.
- Contoh: *char* karakterA = 'a'.
- Untuk karakter khusus yang umum disebut sebagai *escape sequence* disimpan dalam bentuk karakter sebagai berikut:
  1. '\b' adalah *backspace* atau '\u0008' (*unicode*).
  2. '\u0008' adalah *unicode* untuk *backspace*.
  3. '\f' adalah *formfeed*.
  4. '\t' adalah tabulator.
  5. '\n' adalah baris baru.
  6. '\r' adalah *carriage return*.

Contoh 4.14. Penggunaan tipe data *char* dalam pemrograman Java.

```
public class TipeChar {
    public static void main(String[] args) {
        //Deklarasi char
        char char1 = 'a';
        char char2 = 0x0ff32; //adalah '?'
        char char3 = '\t'; //tab

        //Cetak
        System.out.println("Tipe Data Char");
        System.out.println("-----");

        System.out.println("Tipe Char char1      = "+ char1);
        System.out.println("Tipe Char char2      = ?");
        System.out.println("Tipe Char char3      = a"+ char3+"b");

    }
}
```

Keluaran:

```
Tipe Data Char
-----
Tipe Char char1      = a
Tipe Char char2      = ?
Tipe Char char3      = a b
```

g. Tipe Data logika (*boolean*)

Tipe data *boolean* adalah sebuah tipe data logika dengan hanya 2 yaitu *true* dan *false*.

✓ Tipe data *boolean*

- Tipe data *boolean* hanya terdiri dari satu bit.
- Nilainya hanya dua yang mungkin, yaitu *true* atau *false*.
- Nilai awal adalah *false*.
- Contoh: *boolean* keputusan = *true*.

Contoh 4.15. Penggunaan tipe data logika dalam pemrograman Java.

```
public class TipeBoolean {
    public static void main(String[] args) {
        //Deklarasi
        boolean booleanKondisi, booleanStatus;
        int nilai1,nilai2;
        //Isi variabel nilai1 dan nilai2
        nilai1 = 7;
        nilai2 = 10;

        booleanKondisi = nilai1 > nilai2;
        booleanStatus = true;
        //Cetak
        System.out.println("Tipe Data Boolean");
        System.out.println("-----");

        System.out.println("Nilai BooleanKondisi 7 > 10 = " +
booleanKondisi);
        System.out.println("Nilai booleanStatus = " +
booleanStatus);

    }
}
```

Keluaran:

```
Tipe Data Boolean
-----
Nilai BooleanKondisi 7 > 10 = false
Nilai booleanStatus = true
```

## B. TIPE DATA REFERENSI/OBJEK

Tipe data referensi adalah tipe data yang digunakan untuk menentukan referensi dari sebuah object (*instance* dari *class*). Pendeklarasian tipe data ini hampir sama dengan deklarasi pada tipe data primitif. Bedanya hanya pada pendeklarasian tipe data saja, tipe data referensi harus membuat *instance* dari *class* ke objek. Variabel referensi dibuat menggunakan *constructor* pada *class*. Mereka digunakan untuk mengakses objek. Variabel-variabel referensi

dideklarasikan sebagai tipe khusus yang tidak dapat diubah, misalnya Sepeda, Kendaraan, Kecepatan, Percepatan, dan lain sebagainya.

### C. PERBEDAAN TIPE DATA PRIMITIF DAN TIPE DATA REFERENSI/OBJEK

Walaupun di atas sudah diuraikan tentang Tipe Data Primitif dan Tipe Data Referensi/Objek, namun di bawah ini disajikan tabel perbandingan (perbedaan) dari keduanya.

Tabel 4.3. Perbandingan Tipe Data

<i>Tipe Data Primitif</i>	<i>Tipe Data Referensi/Objek</i>
<b>Di awali huruf kecil</b>	<b>Di awali huruf besar</b>
<b>Built-in (tertanam) pada pemrograman Java dan tergolong ke dalam <i>reserve-keyword</i></b>	Tidak masuk dalam kategori <i>reserve-keyword</i> (kata-kata yang memiliki arti yang spesifik bagi kompiler dan tidak bisa dipakai untuk kegunaan lain pada program, misalnya: tidak bisa dijadikan variabel)
<b>Merepresentasikan nilai tunggal</b>	Dapat menampung nol atau lebih nilai primitif atau objek
<b>Contoh:</b>  <code>int nilai1 = 12; char kar1 = 'a';</code>  <b>dan lain-lain, semua di awali dengan huruf kecil</b>	Contoh:  <code>String namaSepeda = "Sepeda Santai";</code>  atau  <code>String namaSepeda = new String("Sepeda Santai");</code>  Ciri-ciri lain: <ul style="list-style-type: none"> <li>• Dibuat dengan <i>constructor</i> yang didefinisikan <i>class</i>.</li> <li>• Digunakan akses objek.</li> <li>• Nilainya secara <i>default</i> adalah null.</li> <li>• Variabel data Tipe Data</li> </ul>

	Referensi/Objek, dapat digunakan untuk merujuk pada tipe data kompatibel atau sama.
--	---

## D. JENIS VARIABEL

Sebuah variabel menyediakan nama penyimpanan pada suatu program. Setiap variabel di Java memiliki tipe data tertentu seperti yang telah dijelaskan pada sub-bab sebelumnya, nilai-nilai dalam sebuah variabel dapat dimanipulasi sesuai ketentuan dan kebutuhan.

Suatu variabel yang digunakan lebih awal harus dideklarasikan, adapun format deklarasinya sebagai berikut:

```
tipe_data nama_variabel [ = nilai][, nama_variabel [ = nilai] ...];
```

Keterangan:

tipe_data	: tipe data
nama_variabel	: nama variabel yang dideklarasikan
value	: nilai awal variabel

Variabel-variaivel dengan tipe data sama dapat dituliskan dalam satu baris deklarasi dan penulisannya dipisahkan dengan tanda koma (,).

Berikut ini adalah contoh deklarasi variabel pada pemrograman Java:

```
...
int int_va, int_vb, var_vc; //Deklarasi 3 variabel integer
int int_vd = 2, int_ve = 3; //Contoh dengan menginisiasi
variabel;
byte bit_vB = 10; //inisiasi dengan tipe Byte
double double_vpi = 3.14; //inisiasi nilai awal
char char_Vkar = 'k'; //inisiasi variabel karakter
...
```

Bahasa pemrograman Java, mengakomodasi 3 jenis variabel, yaitu: variabel lokal, variabel instan (*instance*), dan variabel *class/static*.

### Variabel Lokal

Beberapa hal yang harus diperhatikan dalam mendeklarasikan variabel lokal sebagai berikut:

- Variabel lokal dideklarasikan di dalam *method*, *constructor*, atau *block*.
- Variabel lokal dibuat pada saat masuk *method*, *constructor*, atau *block* akan dijalankan pada saat *method*, *constructor*, atau *block* dan akan dihapus setelahnya.
- *Modifier akses (access modifiers)* tidak berlaku untuk variabel lokal.
- Variabel lokal penggunaannya terbatas pada *method*, *constructor*, atau *block* variabel yang dideklarasikan.
- Tidak ada nilai awal bagi variabel lokal sehingga harus diinisiasi dari awal.

Contoh 4.16. Penggunaan variabel lokal dalam pemrograman Java

```
public class Kecepatan {  
    public void tambahKecepatan() {  
        int cepat = 0;  
        cepat = cepat + 40;  
        System.out.println("Penambahan Kecepatan : " + cepat);  
    }  
  
    public static void main(String args[]) {  
        Kecepatan kecepatan = new Kecepatan();  
        kecepatan.tambahKecepatan();  
    }  
}
```

Keterangan:

Variabel **cepat** adalah sebuah variabel lokal yang didefinisikan dan diinisiasi di dalam *method tambahKecepatan*; dan variabel **cepat** hanya akan bisa digunakan dalam *method tambahKecepatan*.

Keluaran:

```
Penambahan Kecepatan : 40
```

Jika contoh program di atas diubah pada baris ketiga dari `int cepat = 0;` menjadi `int cepat;` apa yang akan terjadi? Misal kode programmnya di bawah:

```
public class Kecepatan {
    public void tambahKecepatan() {
        int cepat;
        cepat = cepat + 40;
        System.out.println("Penambahan Kecepatan : " + cepat);
    }

    public static void main(String args[]) {
        Kecepatan kecepatan = new Kecepatan();
        kecepatan.tambahKecepatan();
    }
}
```

Perubahan yang dilakukan adalah dari varibel lokal terinisiasi (memberikan nilai awal) menjadi tidak mempunyai nilai awal, jika dikompilasi maka akan memunculkan pesan kesalahan sebagai berikut:

```
prog.java:4: error: variable cepat might not have been initialized
    cepat = cepat + 40;
               ^
1 error
```

### *Variabel Instance*

Adapun ciri-ciri dari Variabel *Instance* atau Variabel *Non-Static* dapat dilihat sebagai berikut:

- Variabel *instance* dideklarasikan di dalam kelas, tetapi di luar *method*, *constructor*, atau *block*.
- Variabel *instance* disiapkan dalam memori ketika sebuah objek dibuat.
- Variabel *instance* dibuat pada saat suatu objek dibentuk dengan keyword *new*, dan hilang saat objek tersebut dihapus.
- Variabel *instance* dapat dideklarasikan dalam *class* sebelum/sesudah penggunaan.
- *Modifier akses (access modifiers)* dapat digunakan oleh semua *method*, *constructor*, dan *block* di dalam kelas.

- Variabel *instance* memiliki nilai *default* 0 (numerik), *false* (boolean), dan referensi adalah *null*.

Contoh 4.17. Penggunaan variabel *instance* dalam pemrograman Java.

```
public class Sepeda {
    //variabel instance ini dapat digunakan oleh
    //semua subkelas.
    public String namaSepeda;
    public int hargaSepeda;

    //variabel namaSepeda dan hargaSepeda
    //diinisialisasikan dalam konstruktur.
    public Sepeda (String NamaSepeda, int HargaSepeda)
    {
        namaSepeda = NamaSepeda;
        hargaSepeda = HargaSepeda;
    }

    //Method ini menampilkan informasi sepeda.
    public void tampilSepeda()
    {
        System.out.println("Nama sepeda : "+namaSepeda);
        System.out.println("Harga sepeda : "+hargaSepeda);
    }

    public static void main(String[] args) {
        Sepeda namasepeda = new Sepeda("Sepeda Gunung",100000);
        namasepeda.tampilSepeda();
    }
}
```

### Variabel *class/static*

Adapun ciri-ciri dari Variabel *class/static* dapat dilihat sebagai berikut:

- Variabel *class/static* dideklarasikan menggunakan keyword *static* di dalam *class*, tetapi di luar *method*, *constructor*, atau *block*.
- Hanya ada satu salinan dari variabel kelas per kelas.
- Variabel *class/static* biasanya dideklarasikan sebagai konstanta. Variabel konstanta nilainya tetap dan tidak berubah.
- Variabel *class/static* dibuat ketika program dimulai dan dihapus saat program berhenti.

- Variabel *class/static* memiliki nilai *default*.
- Variabel *class/static* dapat diakses pemanggilannya dengan nama *class*, misalnya *Sepeda.namaSepeda*.
- Saat mendeklarasikan variabel kelas dengan *public static final*, maka nama variabel (konstanta) dituliskan semua dalam huruf kapital. Jika variabel *class/static* tidak dideklarasikan dengan *public final*, sintaks pertamanya sama dengan variabel *instance* dan lokal.

Sintak variabel *class/static* sebagai berikut:

```
[optional_modifier] static type nama_variabel[;][ = value];
```

Keterangan:

- *Optional\_modifier* adalah sifat dari variabel
- *type* adalah nama tipe data yang digunakan
- *nama\_variabel* adalah nama variabel yang akan digunakan
- *;* adalah memiliki nilai *default* jika langsung menggunakan ;)
- *value* adalah memberikan nilai dari variabel

Contoh 4.18:

```
public class Sepeda {  
    // Deklarasi static Variabel  
    // modifier public dan default  
    public static String jenisSepeda;  
    static int jumlahSepeda;  
  
    public static void main(String[] args){  
        /**  
         * Inisialisasi variabel dan menampilkan  
         * jenisSepeda dan jumlahSepeda  
         */  
        jenisSepeda = "Sepeda Gunung";  
        jumlahSepeda = 4;  
        System.out.println("Jenis sepeda\t: " + jenisSepeda);  
        System.out.println("Jumlah sepeda\t: " + jumlahSepeda);  
    }  
}
```

Keluaran:

```
Jenis sepeda : Sepeda Gunung  
Jumlah sepeda : 4
```

Penjelasan:

Di dalam kelas *Sepeda*, dideklarasikan variabel *static*, dengan *modifier* publik dan *default*, proses inisialisasi dan pemanggilan variabel tidak menggunakan objek sama sekali, ini karena penggunaan *keyword static* memberikan penguatan argumentasi bahwa variabel tersebut adalah milik dari kelas itu sendiri.

Contoh 4.19. Contoh lain dengan variabel *static* dengan konstan.

```
public class Sepeda {  
  
    /**  
     * variabel hargaSepeda adalah  
     * modifier private dan  
     * variabel static  
     */  
    private static int hargaSepeda;  
  
    // variabel jenisSepeda konstan  
    public static final String JENISSEPEDA = "Sepeda Gunung";  
  
    public static void main(String args[]) {  
        hargaSepeda = 12000000;  
        System.out.println(JENISSEPEDA+ " harganya : Rp. " +  
                           hargaSepeda);  
    }  
}
```

Keluaran:

```
Sepeda Gunung harganya : Rp. 12000000
```

**LATIHAN**

Untuk memperdalam pemahaman Anda mengenai materi di atas, kerjakanlah latihan berikut!

- 1) *Pseudocode* berikut adalah algoritma untuk mencari luas lingkaran:

***Program Java class cariLuasLingkarancari***

```

1  Mulai
2  //Melakukan Deklarasi
3  Double phi, jari2, luaslingkaran
4  //Inisisasi
5  phi ← 3.14;
6  jari2 ← 25.0;
7  //Proses Perhitungan Luas
8  luaslingkaran ← phi * jari2 * jari2
9  //Proses menampilkan hasil
10 cetak    "Luas      lingkaran      "      +
     luaslingkaran
11 Selesai

```

Buatlah dalam bentuk program Java sederhana dengan *pseudocode* di atas.

*Petunjuk Jawaban Latihan*

- 1) Jika *pseudocode* dikonversi kedalam bahasa pemrograman Java, lihat program Java mencari luas lingkaran berikut:

```

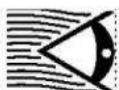
public class cariLuasLingkaran {
    public static void main(String[] args) {
        Double phi,jari2, luaslingkaran;
        jari2 = 25.0;
        phi    = 3.14;
        luaslingkaran = phi * jari2 * jari2;
        System.out.print("Luas lingkaran : ");
    }
}

```

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

Keluaran:

```
Luas lingkaran : 1962.5
```



### RANGKUMAN

Variabel adalah lokasi memori yang disediakan untuk menyimpan nilai pada komputer. Sistem operasi akan mengalokasikan memori sesuai tipe data pada variabel. Terdapat dua tipe data yang tersedia pada pemrograman Java, yaitu: tipe data primitif dan tipe data referensi/objek.

Terdapat 8 macam tipe data primitif yang semua menjadi *keyword*. Tipe data tersebut adalah: *byte*, *short*, *int*, *long*, *float*, *double*, *char*, dan *boolean*. Kedelapan tipe data primitif tersebut dapat dikelompokkan ke dalam 4 kelompok, yaitu: tipe data bilangan bulat, tipe data bilangan real (desimal), tipe data karakter, dan tipe data logika. Yang termasuk ke dalam tipe data bilangan bulat adalah: *byte*, *short*, *int*, dan *long*. Untuk tipe data bilangan real adalah: *long* dan *float*. Tipe data karakter adalah *char* sedangkan untuk tipe data logika yakni *boolean* (*true/false*).

Tipe data bilangan bulat (*integer*) terdiri dari 4 tipe bilangan bulat, yaitu *byte*, *short*, *int*, dan *short*. Tipe data real (desimal) terdiri dari 2, yaitu: *float* dan *double*, tipe data *char*, tipe data logika (*boolean*) hanya mempunyai 2 nilai, yaitu *true* dan *false*. Tipe data referensi adalah tipe data yang digunakan untuk menentukan referensi dari sebuah *object* (*instance* dari *class*). Pendeklarasian tipe data ini hampir sama dengan deklarasi pada tipe data primitif.

Jenis varibel terdiri dari varibel lokal, variabel *instance*, dan variabel *class/static*.

**TES FORMATIF 3**

Pilihlah satu jawaban yang paling tepat!

- 1) Kelompok besar tipe data pada Java terdiri dari....
  - A. 2
  - B. 4
  - C. 8
  - D. 10
- 2) Tipe data *byte*, *short*, *int*, *long*, *float*, *double*, *char*, dan *boolean* adalah jenis tipe data....
  - A. Tipe data primitif
  - B. Tipe data referensi
  - C. Tipe data objek
  - D. Tipe data boolean
- 3) Jika dikelompokkan, maka tipe data pada pemrograman Java, terdiri dari....
  - A. 3
  - B. 4
  - C. 5
  - D. 6
- 4) *True/False* termasuk dalam kelompok tipe data....
  - A. Tipe Data Logika
  - B. Tipe Data *Real*
  - C. Tipe Data *Integer*
  - D. Tipe Data *Char*
- 5) Batas minimum dan maksimum nilai tipe data *integer* adalah....
  - A. -32.768 dan 32.767.
  - B. -9,223,372,036,854,775,808 dan 9,223,372,036,854,775,807
  - C. - 2,147,483,648 dan 2,147,483,647
  - D. false dan true
- 6) Nilai awal dari tipe data *boolean* adalah....
  - A. 1
  - B. 0
  - C. *true*
  - D. *false*

- 7) Pernyataan yang benar di bawah ini tentang tipe data primitif....
- Tipe data tidak masuk sebagai *keyword*
  - Tipe data tergolong kedalam *keyword*
  - Tidak mempersentasikan pada nilai tunggal
  - Dapat digunakan untuk merujuk pada tipe data kompatibel atau sama
- 8) Pernyataan yang benar tentang variabel lokal adalah....
- Variabel lokal dideklarasikan di dalam *method*, *contructor*, atau *block*
  - Modifier akses (access modifier)* dapat digunakan oleh semua *method*, *constructor*, dan *block* di dalam kelas
  - Dapat dideklarasikan dalam *class* sebelum/sesudah penggunaan
  - Nama variabel (konstanta) dituliskan semua dalam huruf kapital. Jika variabel *class/static* tidak dideklarasikan dengan *public final*
- 9) Variabel *instance* dibuat dengan *keyword*....
- Static*
  - New*
  - Private*
  - Public*
- 10) Variabile *class* dideklarasikan menggunakan *keyword*...
- Static*
  - New*
  - private*
  - public*

Cocokkanlah jawaban Anda dengan Kunci Jawaban Tes Formatif 3 yang terdapat di bagian akhir modul ini. Hitunglah jawaban yang benar. Kemudian, gunakan rumus berikut untuk mengetahui tingkat penguasaan Anda terhadap materi Kegiatan Belajar 3.

$$\text{Tingkat penguasaan} = \frac{\text{Jumlah Jawaban yang Benar}}{\text{Jumlah Soal}} \times 100\%$$

Arti tingkat penguasaan: 90 - 100% = baik sekali

80 - 89% = baik

70 - 79% = cukup

< 70% = kurang

Apabila mencapai tingkat penguasaan 80% atau lebih, Anda dapat meneruskan dengan modul selanjutnya. **Bagus!** Jika masih di bawah 80%, Anda harus mengulangi materi Kegiatan Belajar 3, terutama bagian yang belum dikuasai.

## Kunci Jawaban Tes Formatif

### Tes Formatif 1

- 1) A
- 2) A
- 3) C
- 4) D
- 5) D
- 6) C
- 7) A
- 8) A
- 9) A
- 10) B

### Tes Formatif 2

- 1) A
- 2) B
- 3) C
- 4) D
- 5) A
- 6) A
- 7) D
- 8) D
- 9) B
- 10) D

### Tes Formatif 3

- 1) A
- 2) A
- 3) A
- 4) A
- 5) C
- 6) D
- 7) B
- 8) A
- 9) B
- 10) A

## Glosarium

- Developer* : Developer biasanya menggambarkan tujuan, dan mereka merancang perangkat lunak yang akan diambil. *Developer* memiliki lebih banyak kebebasan karena mereka memiliki pengalaman yang lebih dalam. *Programmer* juga bisa disebut coder, sedangkan developer juga bisa disebut software engineer.
- Gui (Graphical User Interface)* : jenis antarmuka pengguna yang menggunakan metode interaksi pada peranti elektronik secara grafis (bukan perintah teks) antara pengguna dan komputer.
- Ide* : program komputer yang memiliki beberapa fasilitas yang diperlukan dalam pembangunan perangkat lunak. Tujuan dari IDE adalah untuk menyediakan semua utilitas yang diperlukan dalam membangun perangkat lunak.
- Jdk (Java Development Kit)* : Sebuah perangkat lunak yang digunakan untuk melakukan proses kompilasi dari kode java ke bytecode yang dapat dimengerti dan dapat dijalankan oleh JRE (Java Runtime Environment). JDK wajib terinstall pada komputer yang akan melakukan proses pembuatan aplikasi berbasis java, namun tidak wajib terinstall di komputer yang akan menjalankan aplikasi yang dibangun dengan java.
- Library* : (Library dalam Bahasa Inggris), dalam ilmu komputer adalah koleksi dari rutin-rutin program yang digunakan untuk membangun dan mengembangkan perangkat lunak. ... Karenanya, sebagian besar kode digunakan oleh aplikasi modern disediakan dalam pustaka sistem operasi.

- Platform : Dalam ilmu komputer, platform, serambi, atau wahana merupakan kombinasi antara sebuah arsitektur perangkat keras dengan sebuah kerangka kerja perangkat lunak (termasuk kerangka kerja aplikasi). Kombinasi tersebut memungkinkan sebuah perangkat lunak, khusus perangkat lunak aplikasi, dapat berjalan. Platform yang umum sudah menyertakan arsitektur, sistem operasi, bahasa pemrograman dan antarmuka yang terkait (pustaka sistem runtime atau antarmuka pengguna grafis) untuk komputer.
- Portabel : sebuah perangkat lunak komputer yang dapat dibawa dalam peralatan portabel (contohnya: USB flash drive) dan dapat digunakan di setiap komputer tanpa perlu melalui proses instalasi terlebih dahulu. Ketika peralatan portable dihubungkan dengan komputer, aplikasi portabel tersebut dapat langsung digunakan. Keuntungan dari perangkat lunak jenis ini adalah Anda dapat membawa data beserta program yang dibutuhkan untuk membukanya ke mana saja untuk dapat dibuka di komputer manapun. Karena data disimpan di peralatan portabel, maka keamanan data tersebut juga diuntungkan karena tidak tersimpan di dalam komputer tertentu.
- Sumber:
- [https://id.wikipedia.org/wiki/Aplikasi\\_portabel](https://id.wikipedia.org/wiki/Aplikasi_portabel)
- Swt-Awt : SWT adalah sebuah alat kerja (toolkit) untuk membuat elemen (gawit) antarmuka pengguna grafis (GUI) di dalam bahasa pemrograman Java. SWT merupakan alternatif pustaka GUI di samping *Abstract Windowing Toolkit* (AWT) dan *Swing*.

AWT disebut juga “*Another Windowing*

*Toolkit*", adalah pustaka windowing bertujuan umum dan multiplatform serta menyediakan sejumlah kelas untuk membuat GUI di Java. Dengan AWT, dapat membuat *window*, menggambar, manipulasi gambar, dan komponen seperti Button, Scrollbar, Checkbox, TextField, dan menu pull-down.

## Daftar Pustaka

- Eckel, B. (2000). *Thinking in Java (2<sup>nd</sup>)*. New Jersey: Prentice Hall.
- Schildt, H. (2007). *The complete reference Java (Seventh Edition)*. New York: Mc Graw Hill.
- Flask, R. *Java for beginners (2<sup>nd</sup>)*.
- Eck, D.J. (2006). *Introduction to programming using java (Version 5.0)*. Departement of Mathematic and Computer Science. Geneva, New York 14456.
- Horstmann, C.S. & Cornell, G. (2013). *Core java volume 1 fundamental(ninth Edition)*. New York: Prentice Hall.
- Redko, A. *Adnvanve java preparing you for java mastery*.
- Purbasari, I.Y. Desain & analisis algoritma. Yogjakarta: Graha Ilmu.
- Hartati, G. S. (2007). Pemrograman GUI swing java dengan netbeans 5. Yogyakarta: Andi Offset
- Kadir, A. (2005). Dasar pemrograman Java 2. Yogyakarta: Andi Offset.
- <http://en.wikipedia.org/wiki/JDK>
- <http://en.wikipedia.org/wiki/J2SDK>
- <https://ide.geeksforgeeks.org/>
- <http://www.oracle.com/technetwork/java/javase/overview/index.html>

# Operator, Perulangan, dan Kondisi *if* pada Java

Kani, M.Kom.



## PENDAHULUAN

---

Pemrograman Java tergolong dalam bahasa pemrograman yang cukup familiar, basis OOP yang diusung oleh Java menjadikannya bahasa yang diminati oleh para *programmer*. Di Indonesia bahasa Pemrograman Java masuk menjadi 10 besar. Hal ini karena program studi Teknik Informatika dan Sistem Informasi menetapkan bahasa pemrograman Java menjadi materi ajar di kelas.

Materi Java pada modul ini disajikan dalam 4 Kegiatan Belajar. Setelah mempelajari modul ini, Mahasiswa dapat:

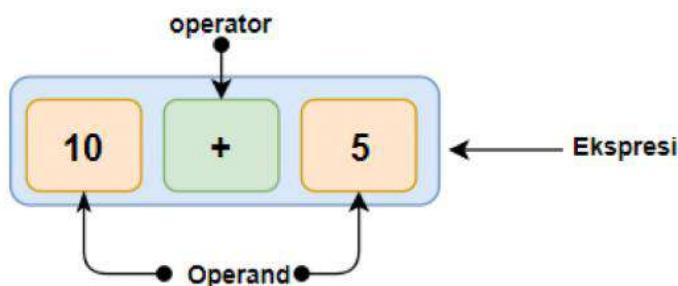
1. menjelaskan pengertian dan fungsi operator;
2. memahami pengertian dan fungsi ekspresi;
3. menyebutkan dan menjelaskan jenis-jenis operator aritmetika dan bagaimana menggunakannya;
4. menyebutkan dan menjelaskan jenis-jenis operator *Unary* dan bagaimana menggunakannya;
5. menyebutkan dan menjelaskan jenis-jenis operator penugasan dan bagaimana menggunakannya;
6. menyebutkan dan menjelaskan jenis-jenis operator relasional dan bagaimana menggunakannya;
7. menyebutkan dan menjelaskan jenis-jenis operator logika dan bagaimana menggunakannya;
8. menyebutkan dan menjelaskan jenis-jenis operator *Ternary* dan bagaimana menggunakannya;
9. menjelaskan pernyataan perulangan pada pemrograman Java dan bagaimana menggunakannya;
10. menyebutkan dan menjelaskan jenis-jenis pernyataan percabangan dan bagaimana menggunakannya.

**KEGIATAN BELAJAR 1***Pengenalan Operator*

Operator adalah simbol yang digunakan untuk memberikan perintah kepada komputer untuk melakukan aksi satu atau lebih *operand*. Dalam matematika, *operand* adalah objek dari operasi matematika. Di dunia komputer, *operand* adalah bagian dari instruksi komputer untuk menentukan data yang akan dioperasikan atau dimanipulasi dan dikembangkan dari data itu sendiri. Dalam suatu operasi boleh jadi adalah proses tambah, kurang, kali, bagi, atau yang lainnya. Pada bahasa pemrograman Java menyediakan banyak jenis operator yang dapat digunakan sesuai kebutuhan, operator pada Java diklasifikasi berdasarkan fungsionalitas. Beberapa operator yang sering digunakan pada Java, yaitu: Operator Aritmetika, *Unary*, Penugasan.

**A. EKSPRESI**

Ekspresi adalah suatu bentuk yang menghasilkan suatu nilai, dalam sebuah ekspresi terdapat operator dan *operand*, misalnya  $10 + 5$ , 10 dan 5 adalah *operand*, dan + adalah operator.



Gambar 5.1  
Ekspresi

Ekspresi di atas jika diterapkan ke dalam pemrograman Java, maka baris programnya sebagai berikut:

```

...
System.out.println(10 + 5);
...

```

## B. OPERATOR ARITMETIKA

Operator Aritmatika digunakan untuk melakukan operasi aritmetika sederhana pada tipe-tipe data primitif, misalnya: Perkalian, Penambahan, Pengurangan, dan Modulus.

Anggaplah variabel  $a = 10$  dengan tipe data *integer*, variabel  $b = 5$  dengan tipe data *integer*, kemudian dilakukan operasi sebagai berikut:

Operator	Deskripsi	Contoh
+ (Penambahan)	Menambahkan nilai di kedua sisi operator	$a + b$ hasilnya 15
- (Pengurangan)	Kurangi <i>operand</i> kanan dari <i>operand</i> kiri	$a - b$ hasilnya 5
* (Perkalian)	Mengalikan nilai di kedua sisi operator	$a * b$ hasilnya 50
/ (Pembagian)	Membagi <i>operand</i> kiri dengan <i>operand</i> kanan	$a / b$ hasilnya 2
% (Modulus)	Membagi <i>operand</i> kiri dengan <i>operand</i> kanan dan memunculkan sisanya	$a \% b$ hasilnya 0

Contoh 5.1. Operator dalam bahasa pemrograman Java:

```

// Operator Aritmetika
public class operatorAritmetika
{
    public static void main(String[] args)
    {
        int a = 10, b = 5, c = 0, d = 15, e = 12, f = 7;
        String x = "Algoritma ", y = "dan ", z = "Pemrograman ";
    }
}
```

```
System.out.println("Operator Aritmetika");
System.out.println("-----");
//operator + dan -
System.out.println("a + b = "+(a + b));
System.out.println("a - b = "+(a - b));

//operator +
//menggabung 3 string dalam 3 variabel.
System.out.println("x + y + z= "+x + y + z);

//operator * dan /
System.out.println("a * b = "+(a * b));
System.out.println("a / b = "+(a / b));

// operator modulo
System.out.println("a % b = "+(a % b));
}
}
```

Keluaran:

```
Operator Aritmetika
-----
a + b = 15
a - b = 5
x + y + z= Algoritma dan Pemrograman
a * b = 50
a / b = 2
a % b = 0
```

## C. OPERATOR UNARI (*UNARY*)

Operator *Unary* hanya membutuhkan satu *operand*, operator ini digunakan untuk menambah, mengurangi atau meniadakan nilai.

Anggaplah variabel  $a = -10$  dengan tipe data *integer*, variabel  $b = 5$  dengan tipe data *integer*,  $kondisi = false$ , kemudian dilakukan operasi sebagai berikut:

Operator	Deskripsi	Contoh
+ ( <i>Unary plus</i> )	Memberikan nilai positif.	+a hasilnya -10 (tetap, karena + (positif) diadu dengan – (negatif) maka hasilnya tetap – (negatif))
- ( <i>Unary minus</i> )	Memberikan nilai negatif jika awalnya positif, dan memberikan nilai positif jika nilai awalnya negatif	- a hasilnya 10 (negatif diadu dengan negatif, menghasilkan positif)
++ ( <i>Increment</i> )	Menambahkan nilai dengan 1. Ada dua variasi operator <i>increment</i> , yaitu: <ul style="list-style-type: none"> <li>• <i>Post-Increment</i>: Nilai pertama digunakan dulu dan kemudian ditambah</li> <li>• <i>Pre-Increment</i>: Nilai ditambah dulu baru kemudian digunakan.</li> </ul>	++a hasilnya -9
-- ( <i>Decrement</i> )	Mengurangi nilai dengan 1. Ada dua variasi operator <i>decrement</i> , yaitu: <ul style="list-style-type: none"> <li>• <i>Post-Decrement</i>: Nilai pertama digunakan dulu dan kemudian dikurangi</li> <li>• <i>Pre-Decrement</i>: Nilai dikurangi dulu baru kemudian digunakan.</li> </ul>	--b hasilnya 4
!	Membalikkan nilai Logic	!kondisi hasilnya <i>true</i>

### Contoh 5.2 . Operator Unary

```
// Operator Unary
public class operatorUnary
{
    public static void main(String[] args)
    {
        int a = 10, b = 5, c = 0, d = 15, e = 12;
        boolean kondisi = true;

        // operator pre-increment
        // a = a+1 dan kemudian c = a;
        c = ++a;
        System.out.println("Nilai c (++a) = " + c);

        // operator post-increment
        // c=b kemudian b=b+1
        c = b++;
        System.out.println("Nilai c (b++) = " + c);

        // operator pre-decrement
        // d=d-1 kemudian c=d
        c = --d;
        System.out.println("Nilai c (--d) = " + c);

        // operator post-decrement
        // c=e kemudian e=e-1
        c = --e;
        System.out.println("Nilai c (--e) = " + c);

        // Membalik kondisi
        System.out.println("Nilai !kondisi =" + !kondisi);
    }
}
```

### Keluaran:

```
Nilai c (++a) = 11
Nilai c (b++) = 5
Nilai c (--d) = 14
Nilai c (--e) = 11
Nilai !kondisi =false
```

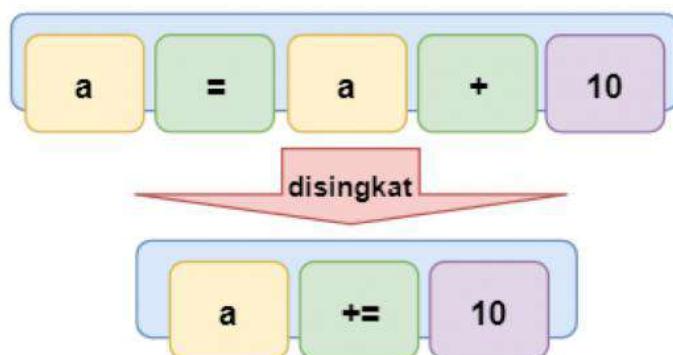
## D. OPERATOR PENUGASAN (ASSIGNMENT)

Operator penugasan digunakan untuk memasukkan nilai ke dalam variabel. Notasi operator penugasan dinotasikan dengan “=”. Pada lingkungan Java ada beberapa operator penugasan.

Format umum operator penugasan adalah :

```
variabel = nilai;
```

Dalam banyak kasus, operator penugasan dapat dikombinasikan dengan operator lain dan lebih singkat, misalnya  $a = a + 10$ , dapat disingkat menjadi  $+= 10$ .



Gambar 5.2  
Ilustrasi Penyingkatan Operasi

Anggaplah nilai  $a = 10$ ,  $b = 5$ , dan  $c = 0$ , jika dilihat pada ekspresi, maka bisa dilihat pada tabel berikut:

Operator	Deskripsi	Contoh
=	Memasukkan nilai dari <i>operand</i> kanan ke dalam <i>operand</i> kiri	$c = a + b$ hasilnya 15
+=	Menambahkan <i>operand</i> kiri dengan <i>operand</i> kanan dan kemudian menugaskannya ke variabel di sebelah kiri.	$a += 1$ hasilnya 11 $b += 1$ hasilnya 6

Operator	Deskripsi	Contoh
<code>-=</code>	Mengurangi <i>operand</i> kiri dengan <i>operand</i> kanan dan kemudian menugaskannya ke variabel di sebelah kiri.	<code>a -= 1</code> hasilnya 9 <code>b -= 1</code> hasilnya 4
<code>*=</code>	Mengalikan <i>operand</i> kiri dengan <i>operand</i> kanan dan kemudian menugaskannya ke variabel di sebelah kiri.	<code>a *= 2</code> hasilnya 20 <code>b *= 2</code> hasilnya 10
<code>/=</code>	Membagi <i>operand</i> kiri dengan <i>operand</i> kanan dan kemudian menugaskannya ke variabel di sebelah kiri	<code>a /= 2</code> hasilnya 5
<code>%=</code>	Menetapkan modulus <i>operand</i> kiri dengan <i>operand</i> kanan dan kemudian menugaskannya ke variabel di sebelah kiri.	<code>a %= 2</code> hasilnya 0
<code>^=</code>	Memangkatkan <i>operand</i> sebelah kiri	<code>a ^= 2</code> hasilnya 100

### Contoh 5.3. Operator Penugasan

```
//operator penugasan
public class operatorPenugasan
{
    public static void main(String[] args)
    {
        //Deklarasi dan INISIASI nilai awal
        int a = 20, b = 10, c, d, e = 10, f = 4;

        // contoh penugasan sederhana
        c = b;
        System.out.println("Nilai c = " + c);

        // contoh operasi
        // penambahan, pengurangan,
        // perkalian, dan pembagian
        // dengan cara biasa
        a = a + 1;
        b = b - 1;
        e = e * 2;
    }
}
```

```
f = f / 2;
System.out.println("Nilai dengan operand BIASA");
System.out.println("-----");
System.out.println("Nilai a = " + a);
System.out.println("Nilai b = " + b);
System.out.println("Nilai e = " + e);
System.out.println("Nilai f = " + f);
System.out.println("");
// mengembalikan ke nilai awal seperti pada
// saat INISIASI nilai awal
a = a - 1;
b = b + 1;
e = e / 2;
f = f * 2;

// menggunakan operator penugasan singkat
// yang hasilnya sama dengan operasi biasa
a += 1;
b -= 1;
e *= 2;
f /= 2;
System.out.println("Nilai dengan operand SINGKAT");
System.out.println("-----");
System.out.println("Nilai a = " + a);
System.out.println("Nilai b = " + b);
System.out.println("Nilai e = " + e);
System.out.println("Nilai f = " + f);
System.out.println("");

// menggunakan operator Modulus
f %= 2;
System.out.println("Modulus");
System.out.println("-----");

System.out.println("Nilai f = " + f);
}
}
```

**Keluaran:**

```
Nilai c = 10
Nilai dengan operand BIASA
-----
Nilai a = 21
Nilai b = 9
Nilai e = 20
Nilai f = 2
```

Nilai dengan operand SINGKAT

-----  
 Nilai a = 21  
 Nilai b = 9  
 Nilai e = 20  
 Nilai f = 2

Modulus

-----  
 Nilai f = 0

## E. OPERATOR RELASIONAL

Operator Relasional adalah operator yang digunakan untuk memeriksa hubungan kesetaraan, lebih besar dari, kurang dari. Operator ini mengembalikan hasil *boolean* setelah perbandingan dilakukan. Operator ini juga lebih banyak digunakan pada *looping* (perulangan) dan kondisional *if else*. Adapun format umum dari operator ini adalah:

Nilai variabel operator\_relasional

Anggaplah kita punya variabel  $a = 2$ ,  $b = 3$ , jika kita gunakan relasional untuk melihat kesetaraan, berikut beberapa contoh penggunaan dan keluarannya:

Operator	Deskripsi	Contoh
$==$	Sama dengan: Memeriksa apakah nilai dari dua <i>operand</i> sama atau tidak, jika ya maka kondisi menjadi benar ( <i>true</i> )	$a == b$ hasilnya <i>false</i> $(2 == 3)$
$!=$	Tidak sama dengan: Memeriksa apakah nilai dari dua <i>operand</i> sama atau tidak, jika nilai tidak sama maka kondisi menjadi benar ( <i>true</i> )	$a != b$ hasilnya <i>true</i> $(2 != 3)$
$<$	Lebih kecil: memeriksa apakah nilai <i>operand</i> kiri	$a < b$ hasilnya <i>true</i> $(2 < 3)$

Operator	Deskripsi	Contoh
	lebih kecil dari nilai <i>operand</i> kanan, jika ya maka kondisi menjadi benar ( <i>true</i> )	
$\leq$	Lebih kecil sama dengan: Memeriksa apakah nilai <i>operand</i> kiri kurang dari atau sama dengan nilai <i>operand</i> kanan, jika ya maka kondisi menjadi benar ( <i>true</i> )	a $\leq$ b hasilnya <i>true</i> (2 $\leq$ 3)
$>$	Lebih besar: Memeriksa apakah nilai <i>operand</i> kiri lebih besar dari nilai <i>operand</i> kanan, jika ya maka kondisi menjadi benar ( <i>true</i> ).	a $>$ b hasilnya <i>false</i> (2 $>$ 3)
$\geq$	Lebih besar sama dengan: Memeriksa apakah nilai <i>operand</i> kiri lebih besar dari atau sama dengan nilai <i>operand</i> kanan, jika ya maka kondisi menjadi benar ( <i>true</i> )	a $\geq$ b hasilnya <i>false</i> (2 $\geq$ 3)

#### Contoh 5.4. Operator Relasional

```
// operator relasional
public class operatorRelasional
{
    public static void main(String[] args)
    {
        int a = 2, b = 3;
        String x = "Relasional", y = "relasional";
        int ar[] = { 1, 2, 3 };
        int br[] = { 1, 2, 3 };
        boolean kondisi = true;

        //Penggunaan operator relasional
        System.out.println("Penggunaan operator relasional");
        System.out.println("-----");
        System.out.println("a == a : " + (a == a));
        System.out.println("a == b : " + (a == b));
        System.out.println("a < b : " + (a < b));
        System.out.println("a <= b : " + (a <= b));
    }
}
```

```

System.out.println("a > b : " + (a > b));
System.out.println("a >= b : " + (a >= b));
System.out.println("a != b : " + (a != b));

// Untuk Array tidak bisa di bandingkan
// menggunakan operator relasional
// nilai akan menghasilkan false
System.out.println("x == y : " + (ar == br));

System.out.println("kondisi==true : " + (kondisi ==
true));
}
}

```

Keluaran:

```

Penggunaan operator relasional
-----
a == b : true
a == b : false
a < b : true
a <= b : true
a > b : false
a >= b : false
a != b : true
x == y : false
kondisi==true :true

```

## F. OPERATOR LOGIKA

Operator Logika yang digunakan untuk melakukan operasi logika adalah sintak "*AND*" dan "*OR*". Operator ini digunakan secara ekstensif untuk menguji beberapa kondisi dalam membuat keputusan.

Terdapat 3 jenis operator logika dalam pemrograman Java, hasil dari operasi operator logika hanya 2, yaitu *true* atau *false*. Berikut operator dan contohnya:

Operator	Deskripsi	Contoh
&&	Menghasilkan hubungan logika DAN (AND), lihat inti kebenaran dari logika AND sebagai berikut: 1. <i>true AND true = true</i>	10 > 9 && 9 > 8 hasilnya <i>true</i>

Operator	Deskripsi	Contoh
	2. <i>false AND false = true</i> 3. <i>true AND false = false</i> 4. <i>false AND true = false</i>	
	Menghasilkan hubungan logika ATAU (OR), lihat inti kebenaran dari logika OR sebagai berikut: 1. <i>true OR true = true</i> 2. <i>false OR false = false</i> 3. <i>true OR false = true</i> 4. <i>false OR true = true</i>	$10 < 9 \parallel 9 > 8$ hasilnya <i>true</i>
!	Menghasilkan hubungan logika NEGASI (NOT), lihat poin kebenaran dari logika NOT sebagai berikut: 1. <i>!(false) = true</i> 2. <i>!(true) = false</i>	$!(2 < 3)$ hasilnya <i>false</i>

### Contoh 5.5. Operator Logika

```
// operator logika
public class operatorsLogika
{
    public static void main(String[] args) {
        //Operator logika &&
        System.out.println("Operator logika AND (&&) ");
        System.out.println("-----");
        System.out.print("1. true && true : ");
        System.out.println(true && true);
        System.out.print("2. true && false : ");
        System.out.println(true && false);
        System.out.print("3. false && false : ");
        System.out.println(false && false);

        System.out.println("");

        //Operator logika ||
        System.out.println("Operator logika OR (||) ");
        System.out.println("-----");
        System.out.print("1. true || true : ");
        System.out.println(true || true);
        System.out.print("2. true || false : ");
        System.out.println(true || false);
    }
}
```

```

System.out.println(true || false);
System.out.print("3. false || false : ");
System.out.println(false || false);

System.out.println("");

//Operator logika &&
System.out.println("Operator logika NOT (!) ");
System.out.println("-----");
System.out.print("1. !(true || true) : ");
System.out.println(!(true || true));
System.out.print("2. !(true || false) : ");
System.out.println(!(true || false));
System.out.print("3. !(false || false) : ");
System.out.println(!(false || false));
}

}

```

Keluaran:

```

Operator logika AND (&&)
-----
1. true && true : true
2. true && false : false
3. false && false : false

Operator logika OR (||)
-----
1. true || true : true
2. true || false : true
3. false || false : false

Operator logika NOT (!)
-----
1. !(true || true) : false
2. !(true || false) : false
3. !(false || false) : true

```

## G. OPERATOR TERNARY

Operator *ternary* adalah operator versi singkat dari pernyataan *if-else*. Operator ini memiliki tiga *operand* sehingga dinamakan *ternary*. Format umumnya sebagai berikut:

kondisi ? if true : if else
-----------------------------

Keterangan:

- kondisi : kondisi atau bagian yang diuji
- if true : dikerjakan jika kondisi benar
- if else : dikerjakan jika kondisi salah

Contoh 5.6. Operator Ternary

```
// operator ternary
public class operatorTernary
{
    public static void main(String[] args)
    {
        int a = 20, b = 10, c = 30, hasil;

        //hasil dari ternary
        hasil = a > b ? b : c;
        System.out.println("Nilai hasil seleksi ternary = "+hasil);
    }
}
```

Keluaran:

Nilai hasil seleksi ternary = 10

Penjelasan program:

Untuk kode program baris  $a > b ? b : c;$  adalah melakukan pengecekan kebenaran, apakah  $a > b$  ( $20 > 10$ ), jika ternyata hasilnya benar (*true*), maka variabel hasil = b atau 10, namun jika kondisi awal bernilai salah (*false*), maka hasil = c atau 30.

## H. PRIORITAS OPERATOR

Perhatikan contoh kasus potongan kode program berikut dalam perhitungan matematika:

```
...
int hasil = 10 - 4 * 2;
...
```

Berapa nilai hasil? Apakah 12 atau 2? Dalam pemrograman Java dikenal operator prioritas tinggi ke rendah, prioritas tinggi akan dikerjakan terlebih dahulu, lalu kemudian mengerjakan operator lebih rendah. Dalam contoh kasus di atas bahwa operator = adalah prioritas paling rendah, operator \* lebih tinggi dibanding dengan operator -, artinya bahwa  $4 * 2$  akan dikerjakan lebih awal, lalu kemudian 10 dikurang dengan hasil perkalian antara  $4 * 2$  atau dengan operasi lebih lengkap yang informatif bagi orang awam adalah  $10 - (4 * 2)$ . Jadi hasilnya bukanlah 12 melainkan 2. Jika anda ragu dengan prioritas operator, maka disarankan menggunakan ( ) yang menandakan prioritas lebih tinggi.

Operator
Tinggi
<code>++ -- ~ !</code>
<code>* / %</code>
<code>+ -</code>
<code>&lt;&lt; &gt;&gt; &gt;&gt;&gt;</code>
<code>&lt; &gt; &lt;= &gt;=</code>
<code>== !=</code>
<code>&amp;</code>
<code>^</code>
<code> </code>
<code>&amp;&amp;</code>
<code>  </code>
<code>? :</code>
<code>= += -= *= /= %= &amp;= ^=  =</code>
<code>&lt;&lt;= &gt;&gt;= &gt;&gt;&gt;=</code>
Rendah

Misal:

```
...
a = b || c && d ^ e;
a = (b || c) && (d ^ e);
a = b || ((c && d) ^ e);
a = (b || c && d ^) e;
...
...
```

**LATIHAN**

Untuk memperdalam pemahaman Anda mengenai materi di atas, kerjakanlah latihan berikut!

- 1) Perhatikan potongan kode program berikut:

```
...
int x, y;
x = 10; y = 12;
a = x++ + (y >>2);
b = x <<2;
c = 4 + ++x - y--;
...
```

Berapakah hasil a, b, c?

- 2) Rumus Volume Prisma Segitiga adalah:

$$\text{volume } (v) = \left( \frac{1}{2} \times \text{alas } (a) \times \text{tinggi } (t) \right) \times \text{tinggi prisma } (tp)$$

Diketahui sebuah prisma sebagai berikut:

- alas (a) segitiga = 10 cm
- tinggi (t) segitiga = 12 cm
- tinggi prisma (tp) = 20 cm

Buatlah algoritma *pseudocode* dan program Java untuk mencari volume Prisma Segitiga?

*Petunjuk Jawaban Latihan*

- 1) Nilai dari:

$$a = 13$$

$$b = 44$$

$c = 4$

- 2) Algoritma *pseudocode* untuk mencari volume Prisma segitiga:

**Program Java class cariVolumePrismaSegitiga**

```

1  Mulai
2  //Melakukan Deklarasi
3  Single a, t, tp, volume
4  //Inisiasi
5  a ← 10.0
6  t ← 12.0
7  tp ← 20.0
8  //Proses Perhitungan volume
9  volume ← (1/2 * a * t) * tp
10 //Proses menampilkan volume
11 cetak "Volume : " + volume
12 Selesai

```

Kemudian untuk program Java untuk mencari volume prisma segitiga sebagai berikut:

```

public class cariVolumePrismaSegitiga {
    public static void main(String[] args) {
        //Deklarasi
        Double a,t,tp,volume;

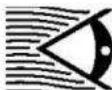
        //Inisiasi
        a = 10.0;
        t = 12.0;
        tp = 20.0;

        //Hitung
        volume = (0.5 * a * t) * tp;

        //Tampilkan
        System.out.print("Volume : ");
    }
}

```

```
    System.out.println(volume+" cm^3");  
}  
}
```



## RANGKUMAN

Operator adalah simbol yang digunakan untuk memberikan perintah kepada komputer untuk melakukan aksi satu atau lebih operand. Ekspresi adalah suatu bentuk operasi yang menghasilkan suatu nilai, dan di dalam sebuah ekspresi terdapat operator dan *operand*.

Operator aritmatika adalah operator yang melakukan operasi aritmatika sederhana pada tipe-tipe data primitif, misalnya perkalian, penambahan, pengurangan, pembagian dan lain sebagainya. Operator *unary* adalah operator yang digunakan untuk menambah, mengurangi, atau meniadakan nilai. Operator penugasan adalah operator yang digunakan untuk memasukkan nilai ke dalam variabel apapun, operator ini dilambangkan dengan “=”. Operator relasional adalah operator yang digunakan untuk memeriksa hubungan seperti kesetaraan, operator ini mengembalikan hasil *boolean* setelah dilakukan perbandingan. Operator logika digunakan untuk melakukan operasi logika “AND” dan logika “OR”. Operasi *ternary* adalah operasi versi singkat dari pernyataan *if-else*.



## TES FORMATIF 1

Pilihlah satu jawaban yang paling tepat!

- 1) Sebuah ekspresi terdiri dari....
  - A. *Operand*
  - B. Operator
  - C. Variabel dan Operator
  - D. *Operand* dan Operator
  
- 2) Penambahan (+), pengurangan (-), perkalian (\*), dan modulus (%) termasuk dalam kelopok operator....
  - A. Logika
  - B. *Unary*

- C. Aritmetika
- D. *Ternary*

3) Sebuah ekspresi sebagai berikut:

hasil = a++;

Makna dari ekspresi pada operator *unary* di atas adalah....

- A. Variabel a digunakan dulu baru kemudian ditambah 1
- B. Variabel a ditambah dulu dengan 1 baru kemudian digunakan
- C. Variabel a dikurangi dulu dengan 1 baru kemudian digunakan
- D. Variabel a digunakan dulu baru kemudian dikurangi 1

4) Perhatikan ekspresi operator penugasan berikut:

$a = a + 2;$

Ekspresi di atas dapat disingkat menjadi....

- A.  $a =+ 2;$
- B.  $a += 2;$
- C.  $a = 2;$
- D.  $a = 2 + 2;$

5) Hasil dari ekspresi modulus

$20 \% = 2;$

adalah....

- A. 0
- B. 1
- C. 5
- D. 10

6) Perhatikan *script* berikut:

```
...
System.out.println(!(3 > 2));
...
```

Hasil dari *script* di atas adalah....

- A. Dari *false* menjadi *true*
- B. Dari *true* menjadi *false*
- C. Tetap *true*
- D. Tetap *false*

- 7) Yang benar terhadap pernyataan logika adalah sebagai berikut, kecuali....
- true AND true = true*
  - true AND false = true*
  - false AND false = true*
  - false and true = false*
- 8) Panggilkan saya si A dan si B, yang datang adalah si B, maka logika bernilai....
- true*
  - false*
  - !(false)*
  - !(!(true))*
- 9)  $2 > 3 ? 5 : 4$   
Nilai dari ekspresi di atas adalah....
- 2
  - 3
  - 4
  - 5
- 10) Perhatikan *script* berikut:

```
...
kondisi := !true;
if (kondisi == true)
{
    a = 1;
    a = ++a;
}
else
{
    a = 2;
    a = a++;
    a = 5;
}
...
```

Nilai a =....

- 5
- 4
- 3
- 2

Cocokkanlah jawaban Anda dengan Kunci Jawaban Tes Formatif 1 yang terdapat di bagian akhir modul ini. Hitunglah jawaban yang benar. Kemudian, gunakan rumus berikut untuk mengetahui tingkat penguasaan Anda terhadap materi Kegiatan Belajar 1.

$$\text{Tingkat penguasaan} = \frac{\text{Jumlah Jawaban yang Benar}}{\text{Jumlah Soal}} \times 100\%$$

Arti tingkat penguasaan: 90 - 100% = baik sekali

80 - 89% = baik

70 - 79% = cukup

< 70% = kurang

Apabila mencapai tingkat penguasaan 80% atau lebih, Anda dapat meneruskan dengan Kegiatan Belajar 2. **Bagus!** Jika masih di bawah 80%, Anda harus mengulangi materi Kegiatan Belajar 1, terutama bagian yang belum dikuasai.

## KEGIATAN BELAJAR 2

**Perulangan**

Perulangan dalam bahasa pemrograman adalah fitur yang memfasilitasi eksekusi instruksi/fungsi yang berulang-ulang selama pada kondisi tertentu. Pada bahasa pemrograman Java dikenal tiga cara untuk mengeksekusi perulangan, meskipun semua cara menyediakan fungsi dasar yang mirip, namun berbeda sintaks dan peletakan kondisi.

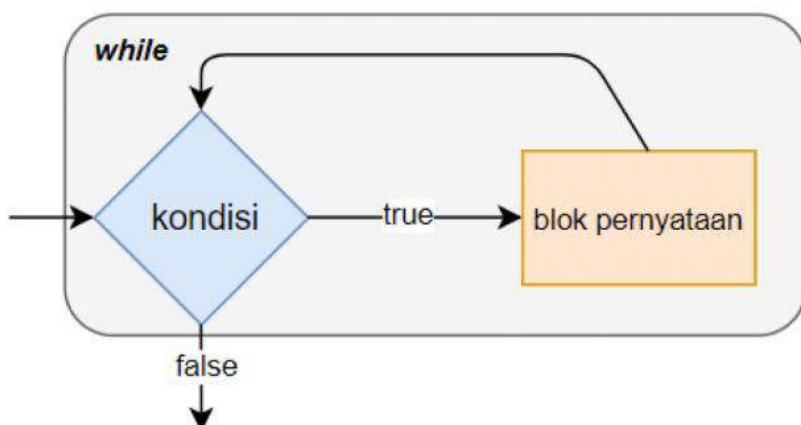
**A. PERNYATAAN WHILE**

Pernyataan *while* adalah pernyataan dengan aliran kontrol yang memungkinkan kode dieksekusi berulang-ulang selama kondisi memenuhi syarat.

Sintaks:

```
while (kondisi boolean)
{
    blok pernyataan
}
```

*Flowchart*



Gambar 5.6  
Flowchart Pernyataan While

Keterangan:

- Pemeriksaan perulangan dilakukan di awal proses dengan melakukan cek kondisi, jika kondisi bernilai *true*, maka dilanjutkan dengan mengeksekusi blok pernyataan untuk menjalankan perulangan untuk pertama kalinya.
- Kondisi akan di cek lagi, jika bernilai *true*, maka blok pernyataan akan dijalankan lagi.
- Ketika kondisi tidak memenuhi syarat atau bernilai *false*, maka perulangan menemui titik akhir.

Contoh 5.7. Perulangan *while*

```
// Perulangan while
class perulanganWhile
{
    public static void main(String args[])
    {
        int perulangan = 1;

        //jika kondisi maka blok program
        //di eksekusi
        while (perulangan <= 4)
        {
            System.out.println("nilai perulangan : " + perulangan);

            // nilai perulangan ditambah 1
            perulangan++;
        }
    }
}
```

Keluaran:

```
nilai perulangan : 1
nilai perulangan : 2
nilai perulangan : 3
nilai perulangan : 4
```

Perlu ditegaskan bahwa pada perulangan *while*, jika kondisi tidak memenuhi syarat maka tidak ada perulangan yang dilakukan.

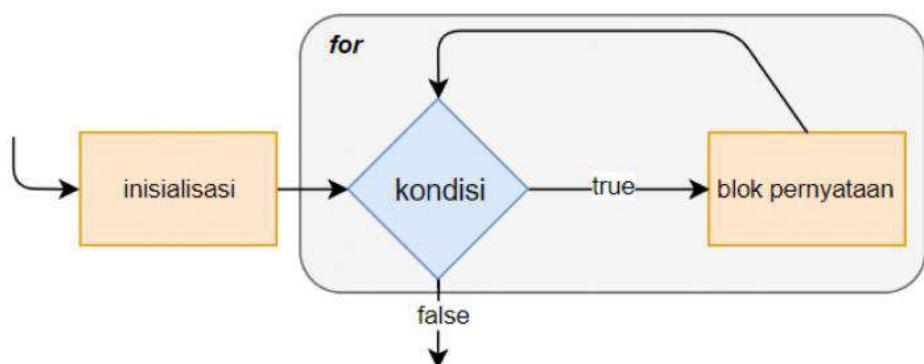
## B. PERNYATAAN *FOR*

Pernyataan *for* menyediakan cara ringkas dalam penulisan struktur perulangan. Berbeda cara dari pernyataan perulangan *while*, pernyataan *for* membutuhkan inisialisasi, kondisi dan penambahan/pengurangan dalam satu baris sehingga menyediakan struktur perulangan yang lebih singkat dan mudah untuk cek kesalahannya.

Sintaks:

```
for (kondisi inisialisasi; kondisi pengujian;
      penambahan/pengurangan)
{
    blok pernyataan
}
```

Flowchart:



Gambar 5.7  
Flowchart Pernyataan *For*

Keterangan:

- *Kondisi inisialisasi*: pada bagian ini, dilakukan inisialisasi variabel yang akan dipakai. Proses ini menandai dimulainya perulangan *for*. Variabel yang sudah dideklarasikan dapat digunakan atau untuk memulai pendeklarasian untuk perulangan *for* saja.
- *Kondisi pengujian*: digunakan untuk menguji kondisi keluar untuk putaran pertama dengan nilai balikan adalah *boolean*.

- *Blok pernyataan*: setelah mengalami pemeriksaan kondisi dan memenuhi syarat, maka perulangan akan menjalankan blok pernyataan dalam perulangan *for*.
- *Penambahan/pengurangan*: digunakan untuk selalu memperbarui nilai varabel.
- *Penghentian perulangan*: ketika kondisi salah (*false*), maka perulangan akan berakhir.

Contoh 5.8. perulangan *for*

```
// Perulangan for
class perulanganFor
{
    public static void main(String args[])
    {
        // iterasi for diawali iterasi=2
        // dan dijalankan sampai pada
        // iterasi <=9
        for (int iterasi = 2; iterasi <= 9; iterasi++)
        {
            System.out.println("Nilai iterasi :" + iterasi);
        }
    }
}
```

Keluaran:

```
Nilai iterasi :2
Nilai iterasi :3
Nilai iterasi :4
Nilai iterasi :5
Nilai iterasi :6
Nilai iterasi :7
Nilai iterasi :8
Nilai iterasi :9
```

Java memberikan versi lain untuk perulangan *for*, fitur ini diperkenalkan pada Java 5. Peningkatan pada perulangan *for* menyediakan cara yang lebih sederhana untuk melakukan iterasi melalui *array*. Penggunaan ini

memang sangat terbatas pada iterasi *array* secara berurutan dan tanpa mengetahui indeks dari *array*.

Sintaks:

```
for (array T: koleksi obj / array)
{
    blok pernyataan
}
```

Pada contoh di bawah, kita bisa lihat bagaimana peningkatan perulangan yang dapat digunakan untuk menyederhanakan pekerjaan.

Contoh 5.9. perulangan *for* menampilkan isi *array* dengan gaya lama (klasik), sebelum ada penyederhanaan:

```
//Perulangan for untuk Array klasik
public class perulanganForArrayKlasik {
    public static void main(String args[]) {
        String [] Sepeda = {"Gunung", "Santai", "Balap",
        "Ontel"};

        System.out.println("Jenis Sepeda ");
        System.out.println("-----");
        //Perulangan Array klasik
        for (int i = 0; i < Sepeda.length; i++)
        {
            System.out.println(Sepeda[i]);
        }
    }
}
```

Keluaran Contoh 5.9 :

```
Jenis Sepeda
-----
Gunung
Santai
Balap
Ontel
```

Contoh 5.10. *for* menampilkan isi *array* dengan gaya penyederhanaan sintaks

```
//Perulangan for untuk Array Sederhana
public class perulanganForArraySederhana {

    public static void main(String args[]) {
        String [] Sepeda = {"Gunung", "Santai", "Balap", "Ontel"};

        System.out.println("Jenis Sepeda ");
        System.out.println("-----");
        //Perulangan array yang disederhanakan
        for(String nama_sepeda : Sepeda ) {
            System.out.println( nama_sepeda );
        }

    }
}
```

Keluaran Contoh 5.10:

```
Jenis Sepeda
-----
Gunung
Santai
Balap
Ontel
```

Perhatikan antara Contoh 5.9 dan Contoh 5.10 pada bagian *script* yang di cetak tebal.

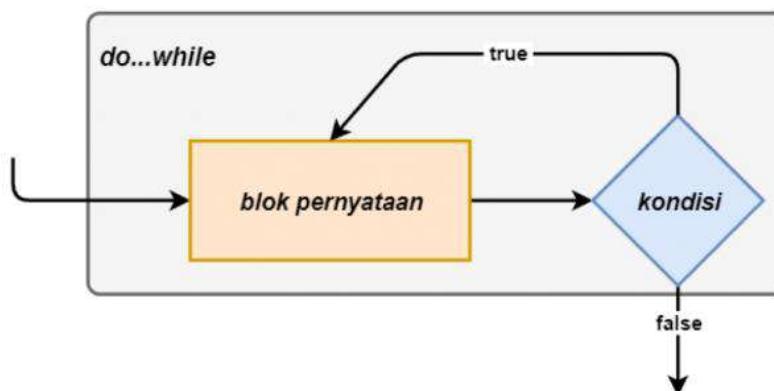
### C. PERNYATAAN DO..WHILE

Fungsi dari pernyataan *do..while* adalah mirip dengan *while*, perbedaan mendasarnya adalah kalau *while* melakukan pengecekan kondisi terlebih dahulu; sedangkan *do..while* minimal mengerjakan *blok pernyataan* sekali lalu kemudian ada pengecekan kondisi.

Sintaks:

```
do
{
    blok pernyataan
}
while (kondisi)
```

Flowchart:



Gambar 5.8  
Flowchart Pernyataan *do..while*

Keterangan:

- *do* adalah proses perulangan dimulai dan langsung mengerjakan blok pernyataan.
- Setelah mengeksekusi blok pernyataan, maka terjadi pembaharuan nilai variabel, lalu kondisi diperiksa untuk nilai benar atau salah, jika bernilai benar maka perulangan melakukan iterasi berikutnya.
- Ketika kondisi salah, maka perulangan akan berakhir.
- Perlu untuk dicatat, bahwa perulangan *do..while* akan mengeksekusi blok pernyataan minimal sekali dalam kondisi apapun.

Contoh 5.11: perulangan *do..while*

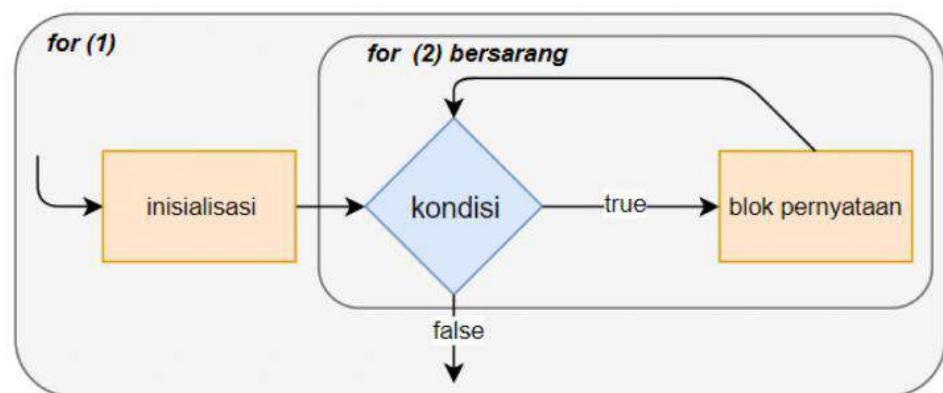
```
// perulangan do..while
class perulanganDoWhile
{
    public static void main(String args[])
    {
        int angka = 12;
        do
        {
            System.out.println("Nilai angka : " + angka);
            angka++;
        }
        while (angka < 20);
    }
}
```

Keluaran:

```
Nilai angka : 12
Nilai angka : 13
Nilai angka : 14
Nilai angka : 15
Nilai angka : 16
Nilai angka : 17
Nilai angka : 18
Nilai angka : 19
```

## D. PERULANGAN YANG BERSARANG

Perulangan bersarang adalah perulangan yang ada dalam blok pernyataan perulangan, misalnya *for* di dalam *for* atau *for* di dalam *while*.



Gambar 5.9  
Flowchart Pernyataan Perulangan Bersarang

Contoh 5.12 : perulangan *for* tanpa *for* bersarang

```
// Perulangan for
class perulanganFor {
    public static void main(String args[])
    {
        // iterasi for diawali iterasi=1
        // dan dijalankan sampai pada
        // iterasi <=15
        for (int iterasi = 1; iterasi <= 10; iterasi++)
        {
            System.out.println("*");
        }
    }
}
```

Keluaran dari Contoh 5.12 *for* tanpa *for* bersarang sebagai berikut:

```
For tanpa for bersarang
-----
*
*
*
*
```

```
*
```

```
*
```

```
*
```

```
*
```

```
*
```

```
*
```

```
*
```

```
*
```

```
*
```

```
*
```

```
*
```

Pada contoh di atas, karakter \* akan dicetak sekali pada setiap perulangan (iterasi), dan berulang selama 15 kali iterasi. Bagaimana caranya supaya setiap iterasi menampilkan jumlah karakter \*, sesuai nilai variabel iterasi, misalnya *iterasi* = 5, maka pada iterasi tersebut menampilkan 5 karakter “\*”. Solusi dari permasalah di atas, yaitu dengan membuat perulangan bersarang.

Contoh 5.13 : perulangan *for* bersarang pada *for*

```
// Perulangan for
class perulanganFor {
    public static void main(String args[])
    {
        System.out.println("For bersarang dalam for");
        System.out.println("-----");
        for (int iterasi = 1; iterasi <= 15; iterasi++)
        {
            for (int iterasi2 = 1; iterasi2 <= iterasi; iterasi2++)
            {
                System.out.print("*");
            }
            System.out.println("");
        }
    }
}
```

Keluaran:

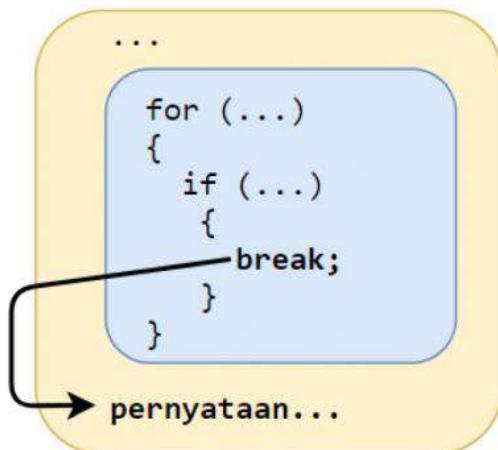
```
For bersarang dalam for
-----
*
**
***
****
```

\* \* \* \* \*  
\* \* \* \* \*  
\* \* \* \* \* \* \*  
\* \* \* \* \* \* \*  
\* \* \* \* \* \* \* \*  
\* \* \* \* \* \* \* \* \*  
\* \* \* \* \* \* \* \* \* \*  
\* \* \* \* \* \* \* \* \* \* \*  
\* \* \* \* \* \* \* \* \* \* \* \*  
\* \* \* \* \* \* \* \* \* \* \* \*

Perhatikan penambahan dan penambahan kode program pada Contoh 5.13.

#### E. KELUAR DARI PERULANGAN

Pada pemrograman Java, suatu perulangan dapat diintervensi pada iterasi yang beroperasi, dalam iterasi perulangan dapat diberhentikan dengan pernyataan *break*;, pernyataan ini dapat digunakan pada 3 pernyataan perulangan.



Gambar 5.10  
Pernyataan *break*

Contoh 5.14. *break* pada pernyataan *for*

```
// Perulangan diberhentikan oleh break
public class perulanganForBreak
{
    public static void main(String args[])
    {
        int nomor = 1;
        String     namaAnak[]      = {"Zahra",      "Zahran",
"Zahira", "Zuhaiy"};

        System.out.println("Nama ketiga anak saya:");
        System.out.println("-----");

        //pemberhentian iterasi
        for (String nama:namaAnak)
        {
            if (nama == "Zuhaily")
            {
                break;
            }
            System.out.println(nomor+". "+nama);
            nomor++;
        }

        System.out.println("baris ini lompatan break;");
    }
}
```

Isi dari *array* mempunyai 4 nilai, yaitu : “Zahra”, “Zahran”, “Zahira”, dan “Zuhaily”. Yang ingin ditampilkan adalah nama tiga anak saya saja, yaitu: “Zahra”, “Zahran”, dan “Zahira”, karena “Zuhaily” bukan anak saya. Sementara dalam perulangan *for* akan mencetak semua nama yang ada dalam *array*, maka untuk tidak mencetak nama yang terakhir; untuk itu perlu dilakukan intervensi iterasi dengan melakukan *break*; jika menemukan nama “Maulana”; Nampak pada penggalan kode program berikut:

```
...
if (nama == "Zuhaily")
{
    break;
}
...
```

Iterasi akan berhenti dan menuju (lompat) pada kode program berikut:

```
...
System.out.println("baris ini lompatan break;");
...
```

Keluaran:

```
Nama ketiga anak saya:  
-----  
1. Zahra  
2. Zahra  
3. Zahira  
baris ini lompatan break;
```

Silahkan Anda kembangkan dengan berlatih menggunakan pernyataan *break* pada perulangan *while* dan *do..while*.

## F. *CONTINUE*

Pernyataan *continue* digunakan untuk melompat ke iterasi berikutnya. Jika pernyataan *continue* dipasang pada perulangan *for* maka mengakibatkan kode program melompat pada iterasi berikutnya dengan menambah atau mengurangi nilai variabel. Apabila pernyataan *continue* dipasangkan pada perulangan *while* atau *do..while*, maka dia akan melompat ke iterasi berikutnya dan langsung cek kondisi.

Sintaks:

```
...
continue;
...
```

**Contoh 5.15. Perulangan dengan *continue***

```
// Perulangan dengan Continue
public class perulanganForContinue
{
    public static void main(String args[])
    {
        int nomor = 1;
        String namaAnak[] = {"Zahra",
"Zahra", "Zuhaily", "Zahira"};

        System.out.println("Nama ketiga anak saya:");
        System.out.println("-----");

        //iterasi lompat ke ietrasa berikutnya
        for (String nama:namaAnak)
        {
            if (nama == "Zuhaily")
            {
                continue;
            }
            System.out.println(nomor+". "+nama);
            nomor++;
        }
    }
}
```

**Penjelasan:**

Pada saat `nama == "Zuhaily"`, maka iterasi dibuat lompat ke iterasi berikutnya dengan pernyataan `continue;`, sehingga mengakibatkan program di bawahnya tidak dikerjakan, adapun hasil dari kode program tersebut tidak mencetak nama “Zuhaily”.

**Keluaran:**

```
Nama ketiga anak saya:
-----
1. Zahra
2. Zahra
3. Zahira
```

**LATIHAN**

Untuk memperdalam pemahaman Anda mengenai materi di atas, kerjakanlah latihan berikut!

- 1) Rumus Volume Prisma Segitiga adalah:

$$\text{volume } (v) = \left( \frac{1}{2} \times \text{alas } (a) \times \text{tinggi } (t) \right) \times \text{tinggi prisma } (tp)$$

Diketahui sebuah prisma sebagai berikut:

- alas (a) segitiga = 10 cm
- tinggi (t) segitiga = 0 cm
- tinggi prisma (tp) = 20 cm

Buatlah algoritma *pseudocode* dan program Java untuk mencari volume Prisma Segitiga dengan perulangan 1 sampai dengan 10, untuk tinggi(t) = nilai perulangan + 5.

*Petunjuk Jawaban Latihan*

- 1) Solusi: untuk nilai tinggi (t) adalah nilai perulangan + 5, artinya bahwa:

$t = \text{perulangan} + 5$  diulang selama 10 kali.

*Pseudocode*-nya sebagai berikut:

**Program Java class cariVolumePrismaSegitigaBerulang**

1	Mulai
2	//Melakukan Deklarasi
3	Single a, t, tp, volume
4	int perulangan
5	//Inisisasi
6	a ← 10.0
7	t ← 0.0

**Program Java class cariVolumePrismaSegitigaBerulang**

8	tp ← 20.0
9	perulangan ← 20.0
10	//Proses Perulangan
11	for (perulangan ← 0; perulangan <= 10; perulangan++)
11.1	//Proses Perhitungan volume
11.2	t ← perulangan + 5.0
11.3	volume ← (0.5 * a * t) * tp
11.4	//Proses menampilkan volume
11.5	cetak "Volume : " + volume
12	Selesai

Untuk kode program Java sebagai berikut:

```
public class cariVolumePrismaSegitigaPerulangan {
    public static void main(String[] args) {
        //Deklarasi
        Double a,t,tp,volume;
        int perulangan;
        //Inisiasi
        a = 10.0;
        t = 0.0;
        tp = 20.0;
        perulangan = 0;
        for (perulangan = 1; perulangan <= 10; perulangan++)
        {
            //Hitung
            t = perulangan + 5.0;
            volume = (0.5 * a * t) * tp;
            //Tampilkan
            System.out.print("t = "+t+" maka Volume Prisma : ");
            System.out.println(volume+" cm^3");
        }
    }
}
```

**Keluaran:**

```
t = 6.0 maka Volume Prisma : 600.0 cm^3
t = 7.0 maka Volume Prisma : 700.0 cm^3
t = 8.0 maka Volume Prisma : 800.0 cm^3
t = 9.0 maka Volume Prisma : 900.0 cm^3
t = 10.0 maka Volume Prisma : 1000.0 cm^3
t = 11.0 maka Volume Prisma : 1100.0 cm^3
t = 12.0 maka Volume Prisma : 1200.0 cm^3
t = 13.0 maka Volume Prisma : 1300.0 cm^3
t = 14.0 maka Volume Prisma : 1400.0 cm^3
t = 15.0 maka Volume Prisma : 1500.0 cm^3
```

**RANGKUMAN**

Perulangan dalam pemrograman adalah fitur yang fungsinya mengerjakan sebuah blok program berulang-ulang selama kondisi memenuhi subyek yang dipersyaratkan. Terdapat tiga macam perintah untuk melakukan perulangan pada pemrograman Java, yaitu: perulangan *while*, *for*, dan *do..while*.

Pernyataan *while* adalah perulangan dengan aliran kontrol yang mengerjakan perulangan dengan langsung mengecek kondisi yang diberikan, jika kondisi memenuhi, maka maka blok pernyataan dalam perulangan akan dikerjakan.

Pernyataan *for* adalah cara ringkas dalam penulisan perulangan, pada perulangan ini kondisi dan penambahan/pengurangan dibuat dalam 1 baris. Pernyataan *for* pada Java 5 memberikan cara singkat dalam menangani *array*, jadi dalam melihat isi dari sebuah *array* tidak perlu menggunakan cara klasik (lama) akan tetapi dengan menggunakan format baru yang telah disediakan.

Pernyataan *do..while* adalah perulangan yang mengerjakan blok pernyataan dalam perulangan minimal sekali, lalu kemudian terjadi pengecekan kondisi, jika kondisi memenuhi maka perulangan akan dilanjutkan pada iterasi selanjutnya.

Perulangan bersarang adalah perulangan yang di dalamnya terdapat perulangan, dalam perulangan bersarang boleh terkombinasi antara 3 perulangan, misalnya perulangan *for*, di dalamnya boleh terdapat perulangan *for*, *while*, dan *do..while*. Begitu juga dengan perulangan *while*, di dalamnya boleh terdapat satu atau lebih perulangan.

Pernyataan *break* adalah pernyataan yang berfungsi memberhentikan perulangan atau keluar dari perulangan. Sedangkan pernyataan *continue* adalah berfungsi untuk melanjutkan ke-iterasi berikutnya tanpa keluar dari perulangan.



## TES FORMATIF 2

---

Pilihlah satu jawaban yang paling tepat!

- 1) Pernyataan perulangan yang melakukan cek kondisi terlebih dahulu adalah....
  - A. *while*
  - B. *for*
  - C. *do..while*
  - D. *break*
- 2) Sebelum melakukan perulangan, perulangan tersebut harus didahului dengan inisialisasi variabel, pernyataan di atas mewakili pernyataan....
  - A. *while*
  - B. *for*
  - C. *di..while*
  - D. *break*
- 3) Pernyataan yang mewakili pernyataan *do..while* adalah sebagai berikut....
  - A. Pernyataan *do..while* diawali dengan inisialisasi variabel
  - B. Pernyataan *do..while* minimal melakukan 1 kali eksekusi sebelum melakukan pengecekan kondisi
  - C. Pernyataan *do..while* diawali dengan pengecekan kondisi
  - D. Pernyataan *do..while* tidak boleh ada perulangan lain dalam perulangannya
- 4) Pada Java 5, Java melakukan penyederhanaan dalam perulangan *if* untuk melakukan proses menampilkan nilai-nilai....
  - A. Variabel
  - B. Nilai pada sebuah ekspresi
  - C. *Array*
  - D. Tidak ada yang benar

5) Perhatikan potongan kode program berikut:

```
...
while (perulangan <= 4)
{
    System.out.println("nilai perulangan : " +
perulangan);

    // nilai perulangan ditambah 1
    perulangan++;
}
...
...
```

Variabel perulangan di atas harus dideklarasikan dengan tipe data....

- A. *Integer*
- B. *String*
- C. Logika
- D. *char*

6) Perhatikan kode program perulangan for berikut:

```
for (int i = 0; i < Sepeda.length; i++)
{
    System.out.println(Sepeda[i]);
}
```

Kode program di atas dapat disederhanakan dengan menggunakan tampilan *array* yang disederhakan dengan....

- A. 

```
for (int i = 0; i < Sepeda; i++)
{
    System.out.println(Sepeda[i]);
}
```
- B. 

```
for (String nama_sepeda : Sepeda)
{
    System.out.println(Sepeda[i]);
}
```

- C. 

```
for (int i = 0; i < Sepeda.length; i++)
{
    System.out.println( nama_sepeda );
}
```
- D. 

```
for (String nama_sepeda : Sepeda)
{
    System.out.println( nama_sepeda );
}
```
- 7) Perulangan bersarang adalah perulangan yang terjadi dalam perulangan.  
Contoh yang tepat di bawah ini perulangan dalam perulangan adalah....
- A. 

```
for (int iterasi = 1; iterasi <= 10; iterasi++)
{
    System.out.println("*");
}
```
- B. 

```
for (int iterasi = 1; iterasi <= 15; iterasi++)
{
    for (int iterasi2 = 1; iterasi2 <= iterasi;
iterasi2++)
    {
        System.out.print("*");
    }
    System.out.println("");
}
```
- C. 

```
for (String nama:namaAnak)
{
    if (nama == "Maulana")
    {
        break;
    }
    System.out.println(nomor+". "+nama);
    nomor++;
}
```
- D. A, B, dan C tidak ada yang benar

- 8) Pernyataan pada Java yang membuat iterasi melompat ke iterasi berikutnya tanpa mengerjakan kode program di bawahnya adalah....
- break*
  - while*
  - continue*
  - do..while*
- 9) Pernyataan yang membuat perulangan berhenti sebelum sampai pada akhir iterasi adalah pernyataan....
- break*
  - continue*
  - while*
  - for*
- 10) Perulangan yang membutuhkan ekspresi pengurangan/penambahan adalah pernyataan perulangan....
- for*
  - while*
  - do..while*
  - continue*

Cocokkanlah jawaban Anda dengan Kunci Jawaban Tes Formatif 2 yang terdapat di bagian akhir modul ini. Hitunglah jawaban yang benar. Kemudian, gunakan rumus berikut untuk mengetahui tingkat penguasaan Anda terhadap materi Kegiatan Belajar 2.

$$\text{Tingkat penguasaan} = \frac{\text{Jumlah Jawaban yang Benar}}{\text{Jumlah Soal}} \times 100\%$$

Arti tingkat penguasaan: 90 - 100% = baik sekali

80 - 89% = baik

70 - 79% = cukup

< 70% = kurang

Apabila mencapai tingkat penguasaan 80% atau lebih, Anda dapat meneruskan dengan Kegiatan Belajar 3. **Bagus!** Jika masih di bawah 80%, Anda harus mengulangi materi Kegiatan Belajar 2, terutama bagian yang belum dikuasai.

**KEGIATAN BELAJAR 3****Pernyataan *if***

**P**engambilan keputusan dalam pemrograman mirip dengan pengambilan keputusan dalam kehidupan sehari-hari. Dalam dunia pemrograman juga mengalami situasi dimana kita menginginkan suatu blok pernyataan tertentu untuk dieksekusi/dijalankan ketika kondisi terpenuhi. Bahasa pemrograman menggunakan pernyataan kontrol untuk melakukan kendali atas aliran eksekusi berdasarkan kondisi yang dipersyaratkan. Pernyataan *if* digunakan untuk mengatur aliran eksekusi untuk maju dan bercabang berdasarkan perubahan suatu kondisi.

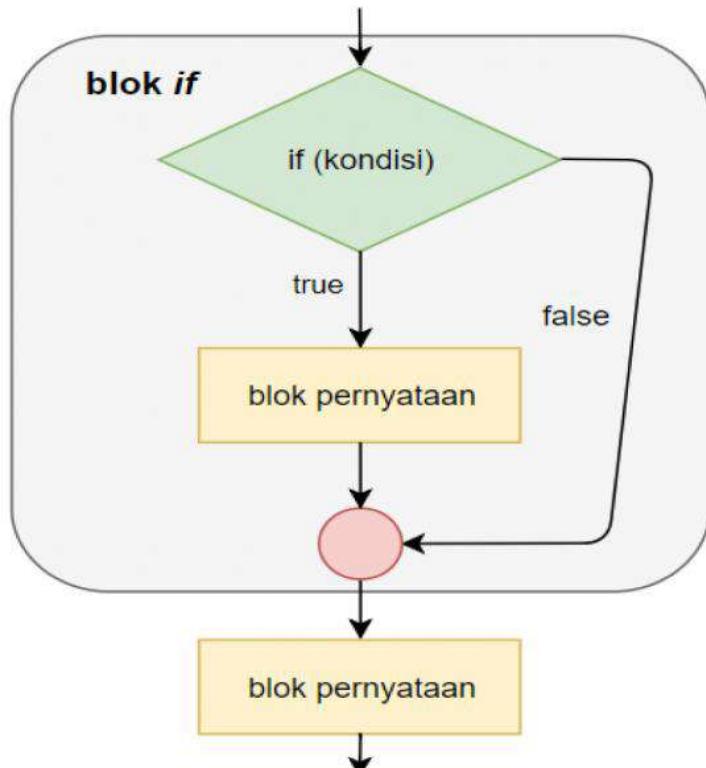
**A. PERNYATAAN *IF***

Pernyataan *if* adalah pernyataan pengambilan keputusan yang paling sederhana. Pernyataan *if* digunakan untuk memutuskan apakah suatu pernyataan atau blok pernyataan tertentu akan dieksekusi atau tidak, yaitu ditentukan oleh suatu kondisi bernilai *true* atau *false*.

Sintaks:

```
if (kondisi)
{
    // Pernyataan untuk dieksekusi jika
    // kondisi benar
}
```

*Flowchart:*



Gambar 5.11  
Flowchart Pernyataan Blok *if*

*Penjelasan:*

Setelah pernyataan *if*, ada *kondisi* yang nilainya berupa *boolean*, nilainya akan diperiksa atau dievaluasi benar (*true*) atau tidak (*false*), jika nilainya benar maka akan mengerjakan blok pernyataan di bawahnya dan jika salah maka akan lompat pada blok tertentu.

#### Contoh 5.16: pernyataan *if*

```
// Pernyataan if
class PernyataanIf
{
    public static void main(String args[])
    {
        int i = 30;

        if (i > 15)
        {
            System.out.println("i > "+15+" adalah benar");
        }
        System.out.println("ini sudah diluar if");
    }
}
```

Penjelasan:

Pada baris `if (i > 15)`, nilai variabel i akan diperiksa, karena nilai = 30 maka dibandingkan dengan operator `>`, operasi `30 > 15` memberikan nilai `true` (bernilai benar secara logika), maka blok program di bawahnya (`System.out.println("i > "+15+" adalah benar");`) akan dikerjakan.

Keluaran:

```
i > 15 adalah benar  
ini sudah diluar if
```

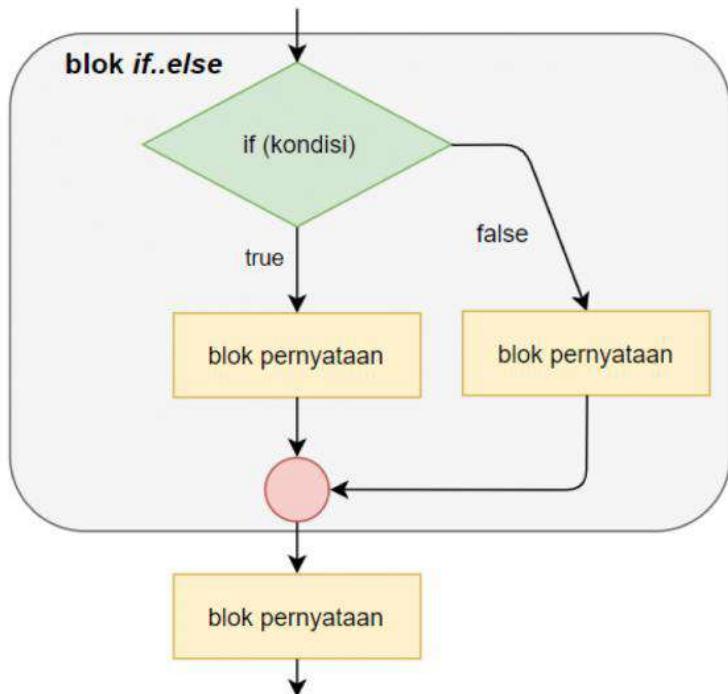
## B. PERNYATAAN IF..ELSE

Pernyataan `if` sendiri memberitahu bahwa jika suatu kondisi benar maka akan mengeksekusi blok pernyataan, dan jika salah maka tidak akan dikerjakan melainkan langsung mengerjakan blok pernyataan diluar kerangka pernyataan `if` atau mengerjakan blok di bawah `else`.

Sintaks:

```
if (kondisi)
{
    // Pernyataan untuk dieksekusi jika
    // kondisi benar
}
else
{
    // Pernyataan untuk dieksekusi jika
    // kondisi salah
}
```

*Flowchart:*



Gambar 5.12  
Flowchart Pernyataan Blok *if..else*

*Penjelasan:*

Setelah pernyataan *if*, ada *kondisi* yang nilainya berupa *boolean*, nilainya akan diperiksa atau dievaluasi nilainya benar (*true*) atau salah (*false*), maka dia akan mengerjakan blok pernyataan di bawahnya, dan jika salah, maka blok pernyataan yang ada di bawah *else* akan dikerjakan.

Contoh 5.17. Pernyataan *if..else*

```

// Pernyataan if..else
class PernyataanIfo
{
    public static void main(String args[])
    {
        int i = 30;

        if (i > 40)
        {
            System.out.println(i > 40);
        }
        else
        {
            System.out.println(i < 40);
        }
    }
}
  
```

```
    }
    System.out.println("ini sudah diluar if..else");
}
}
```

Keluaran:

```
false
ini sudah diluar if..else
```

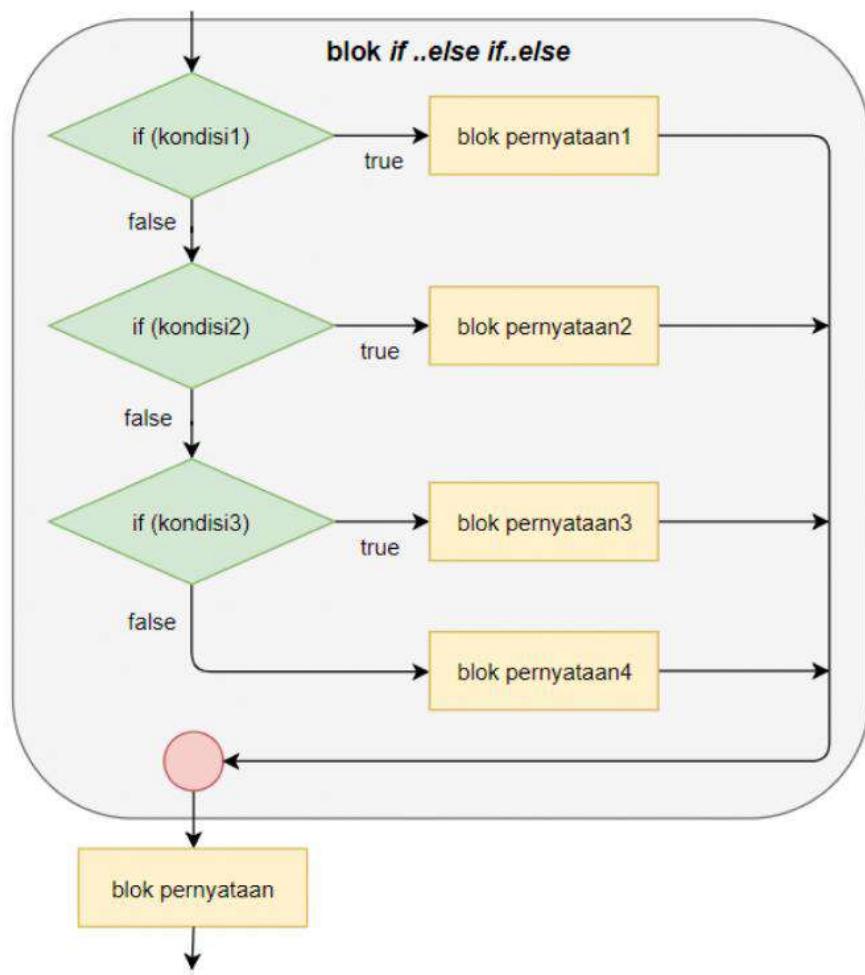
### C. PERNYATAAN *IF..ELSE IF..ELSE*

Untuk pernyataan *if..else* pada materi sebelumnya, hanya dengan 2 pilihan dengan menggunakan *else*. Lalu bagaimana jika terdapat 3 pilihan lebih? Pada pemrograman Java bisa bisa mengatasi lebih dari 2 kondisi. Perhatikan sintaksis di bawah.

Sintaksis:

```
if (kondisi1)
{
    // Pernyataan untuk dieksekusi jika
    // kondisi benar
}
else if (kondisi2)
{
    // Seleksi kedua setelah
    // kondisi1 tidak memenuhi
}
Else
{
    //kondisi salah
}
```

*Flowchart:*



Gambar 5.13  
Flowchart Pernyataan Blok *if..else if..else*

Contoh 5.18. Pernyataan *If..else if..else*

```

// Pernyataan If..else if..else
class pernyataanIfElseIfElse
{
    public static void main(String args[])
    {
        int x = 20;

        if (x == 10)
            System.out.println("x adalah 10");
        else if (x == 15)
            System.out.println("x adalah 15");
        else if (x == 20)
            System.out.println("x adalah 20");
        else
    }
  
```

```
        System.out.println("x tidak 20");  
    }  
}
```

Keluaran:

```
x adalah 20
```

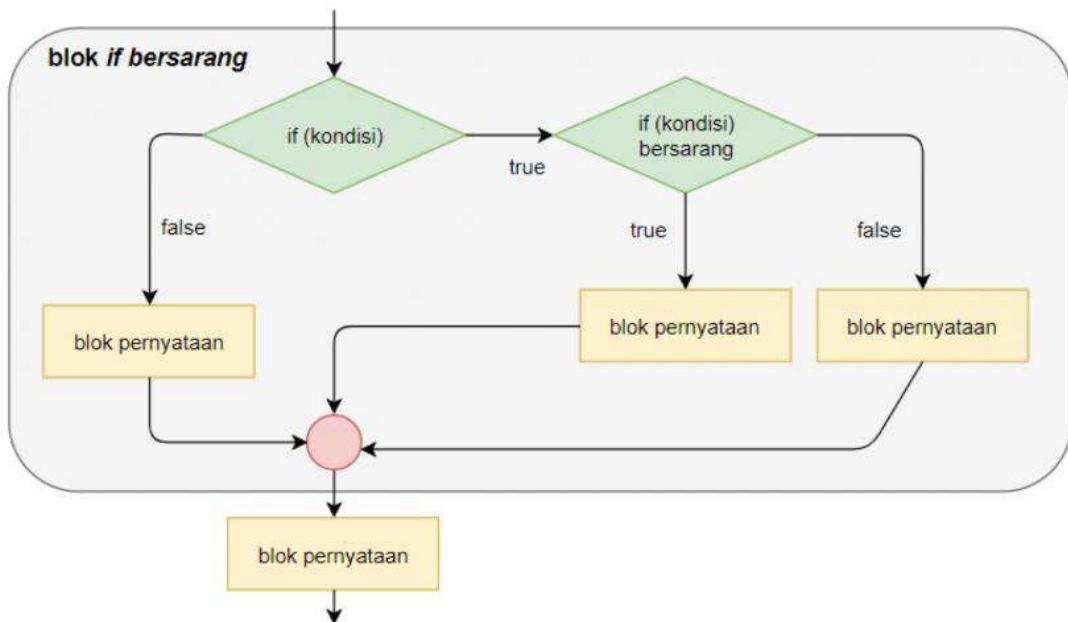
## D. PERNYATAAN *IF..ELSE* BERSARANG

Pernyataan *if* yang bersarang adalah *if* yang terdapat dalam blok pernyataan, dan hal ini dimungkinkan pada pemrograman Java.

Sintaksis:

```
if (kondisi)  
{  
    If (kondisi) //bersarang  
    {  
        ...  
    }  
    Else  
    {  
    }  
}  
else  
{  
    // Pernyataan untuk dieksekusi jika  
    // kondisi salah  
}
```

*Flowchart:*



Gambar 5.14  
Flowchart Pernyataan Blok *if* Bersarang

Contoh 5.19. Pernyataan *else* yang bersarang

```

// Pernyataan else yang bersarang
class PernyataanElseBersarang
{
    public static void main(String args[])
    {
        int x = 13;
        System.out.println("variabel x nilainya "+x);

        if (x == 13)
        {
            // if pertama yang bersarang
            if (x < 15)
                System.out.println("variabel x < 15");

            //if kedua yang bersarang
            if (x < 12)
                System.out.println("variabel x < 12");
            else
                System.out.println("variabel x >= 12");
        }
    }
}
  
```

## E. PERNYATAAN *SWITCH..CASE*

Pernyataan *switch...case* adalah pernyataan cabang dengan multi-arah. Pernyataan ini menyediakan cara mudah untuk mengirimkan eksekusi ke berbagai blok pernyataan berdasarkan nilai ekspresi.

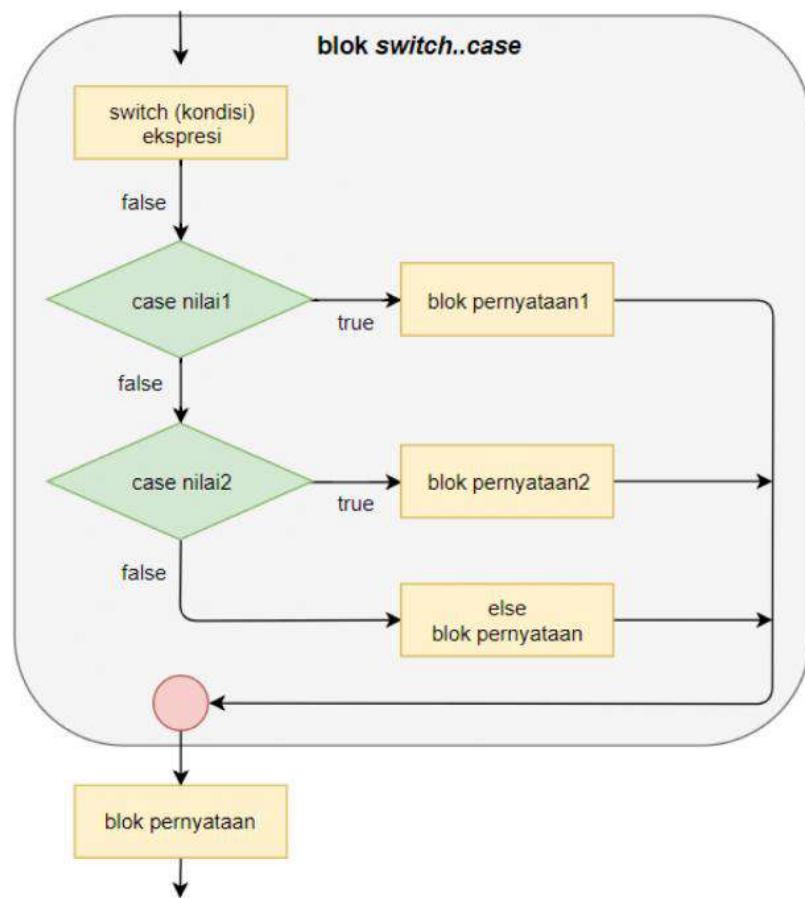
Sintaksis:

```
switch (kondisi)
{
    case nilai1:
        pernyataan1;
        break;
    case nilai2:
        pernyataan2;
        break;
    .
    .
    case nilaiN:
        pernyataanN;
        break;
    default:
        statementDefault;
}
```

### Keterangan

- Ekspresi dapat berupa tipe *byte*, *short*, *int*, *char* atau *enumeration*. Dimulai dengan JDK7, ekspresi juga bisa dari tipe *String*.
- Nilai tidak boleh sama pada dua atau lebih *case*.
- Pernyataan *default* bersifat opsional.
- Pernyataan *break* digunakan sebagai saklar untuk mengakhiri rangkaian pernyataan.
- Pernyataan *break* bersifat opsional. Jika dihilangkan, eksekusi akan berlanjut ke *case* berikutnya.

*Flowchart:*



Gambar 5.15  
Flowchart Pernyataan Blok *switch..case*

Contoh 5.20. Pernyataan *switch..case*

```

// Pernyataan switch..case
class pernyataanSwitchCase
{
    public static void main(String args[])
    {
        String nama = "Zahran";
        switch (nama)
        {
            case "Zahra":
                System.out.println("Nama : Zahra");
                break;
            case "Zahran":
                System.out.println("Nama : Zahran");
                break;
            case "Zahira":
                System.out.println("Nama : Zahira");
        }
    }
}

```

```
        break;
    default:
        System.out.println("Nama : Kani");
    }
}
```

Keluaran:

Nama : Zahran



LATIHAN

Untuk memperdalam pemahaman Anda mengenai materi di atas, kerjakanlah latihan berikut!

- 1) Rumus Volume Prisma Segitiga adalah:

$$volume(v) = \left( \frac{1}{2} x alas(a) x tinggi(t) \right) x tinggi prisma(tp)$$

Diketahui sebuah prisma sebagai berikut:

- alas (a) segitiga = 10 cm
  - tinggi (t) segitiga = 0 cm
  - tinggi prisma (tp) = 20 cm

Buatlah algoritma *pseudocode* dan program Java untuk mencari volume Prisma Segitiga dengan perulangan 1 sampai dengan 10, lakukan perhitungan tinggi( $t$ ) = nilai perulangan + 5, jika nilai perulangan sama dengan ganjil?

*Petunjuk Jawaban Latihan*

- 1) Solusi: untuk nilai tinggi ( $t$ ) adalah nilai perulangan + 5, artinya bahwa:

t = perulangan + 5 diulang selama 10 kali.

Kemudian nilai t akan dicetak pada saat nilai perulangan = 1, 3, 5, 7, 9.

*Pseudocode*-nya sebagai berikut:

**Program Java class cariVolumePrismaSegitigaBerulangGanjil**

1	Mulai
2	//Melakukan Deklarasi
3	Single a, t, tp, volume
4	int perulangan
5	//Inisisasi
6	a ← 10.0
7	t ← 0.0
8	tp ← 20.0
9	perulangan ← 20.0
10	//Proses Perulangan
11	for (perulangan ← 0; perulangan <= 10; perulangan++)
11.1	//Proses Perhitungan volume
11.2	t ← perulangan + 5.0
11.3	if (perulangan % 2 != 0) //% 2 tidak habis /2
11.3.1	volume ← (0.5 * a * t) * tp
11.3.2	//Proses menampilkan volume
11.3.3	cetak "Volume : " + volume
12	Selesai

Untuk kode program Java sebagai berikut:

```
public class cariVolumePrismaSegitigaPerulanganGanjil {
    public static void main(String[] args) {
        //Deklarasi
        Double a,t,tp,volume;
        int perulangan;

        //Inisiasi
        a = 10.0;
```

```

t = 0.0;
tp = 20.0;
perulangan = 0;
for (perulangan = 1; perulangan <= 10; perulangan++)
{
    //Hitung
    t = perulangan + 5.0;
    if (perulangan % 2 != 0)
    {
        volume = (0.5 * a * t) * tp;
        //Tampilkan
        System.out.print("Bil : "+perulangan);
        System.out.print(" t = "+t);
        System.out.print(", maka Volume Prisma : ");
        System.out.println(volume+" cm^3");
    }
}
}
}

```

#### Keluaran:

```

Bil : 1 t = 6.0, maka Volume Prisma : 600.0 cm^3
Bil : 3 t = 8.0, maka Volume Prisma : 800.0 cm^3
Bil : 5 t = 10.0, maka Volume Prisma : 1000.0 cm^3
Bil : 7 t = 12.0, maka Volume Prisma : 1200.0 cm^3
Bil : 9 t = 14.0, maka Volume Prisma : 1400.0 cm^3

```



#### RANGKUMAN

Pernyataan *if* digunakan untuk menyebabkan aliran eksekusi untuk maju dan bercabang berdasarkan perubahan suatu program. Pernyataan *if* merupakan pernyataan pengambilan keputusan yang paling sederhana. Ini digunakan untuk memutuskan apakah pernyataan atau blok pernyataan tertentu akan dieksekusi atau tidak.

Pernyataan *if* sendiri memberitahu bahwa jika suatu kondisi benar maka akan mengeksekusi blok pernyataan, dan jika salah maka tidak

akan dikerjakan, melainkan langsung mengerjakan blok pernyataan di luar kerangka pernyataan *if*.

Pernyataan *if..else if..else* digunakan jika menemukan banyak kondisi. Dalam pemrograman Java juga sangat dimungkinkan menggunakan *if..else* yang bersarang untuk menangani kasus-kasus tertentu. Cara/teknik ini banyak digunakan oleh para *programmer*.

Pernyataan *switch...case* adalah pernyataan cabang dengan multiarah. Pernyataan ini menyediakan cara mudah untuk mengirimkan eksekusi ke berbagai blok pernyataan berdasarkan nilai ekspresi.



### TES FORMATIF 3

---

Pilihlah satu jawaban yang paling tepat!

- 1) Pada pemrograman Java dikenal ada sejumlah percabangan, kecuali....
  - A. *if*
  - B. *If..else*
  - C. *break*
  - D. *switch case*
- 2) Percabangan yang paling sederhana yang hanya memberikan 1 pilihan adalah....
  - A. *if*
  - B. *if..else*
  - C. *continue*
  - D. *break*
- 3) Fitur perulangan yang diperbaharui atau ditingkatkan performansinya pada Java5 adalah....
  - A. *for array*
  - B. *loop array*
  - C. *while array*
  - D. *do..while array*
- 4) Jika banyak pilihan dalam sebuah percabangan, pernyataan percabangan yang cocok digunakan adalah....
  - A. *if*
  - B. *if..else*
  - C. *switch*
  - D. *if..else if..else*

- 5) Jika hanya ada dua pilihan, percabangan yang tepat digunakan adalah....
- A. *if*
  - B. *continue*
  - C. *if..else*
  - D. *break*
- 6) Perhatikan kode program berikut:

```
public static void main(String args[])
{
    int x = 13;
    System.out.println("variabel x nilainya "+x);

    if (x == 13)
    {
        // if pertama yang bersarang
        if (x < 15)
            System.out.println("variabel x < 15");

        //if kedua yang bersarang
        if (x < 12)
            System.out.println("variabel x < 12");
        else
            System.out.println("variabel x >= 12");
    }
}
```

Pada program di atas adalah pernyataan *if* yang bersarang, jumlah *if* yang bersarang adalah....

- A. Ada 2, yaitu *if* dan *if..else*
  - B. Ada 3, yaitu 2 *if* dan 1 *if..else*
  - C. Ada 2, yaitu *if* dan *if..else if..else*
  - D. Ada 2, yaitu *if..else* dan *if..else if..else*
- 7) Ekspresi *switch case* berupa *string* sudah bisa digunakan mulai dari....
- A. JDK3
  - B. JDK4
  - C. JDK5
  - D. JDK7

- 8) Pada *switch case* kita menggunakan tipe *int*, selain dari tipe *int* kita juga bisa menggunakan tipe data....
  - A. *byte, short, int*
  - B. *byte, short, int, char*
  - C. *byte, short, int, char, enumeration*
  - D. *byte, short, int, char, enumeration, string*
- 9) Fungsi *default* pada pernyataan *switch case* hanya bersifat....
  - A. Opsional
  - B. Wajib digunakan
  - C. Tidak perlu digunakan
  - D. Program tidak jalan kalau tidak digunakan
- 10) Ekspresi yang benar dalam menggunakan percabangan *if* adalah....
  - A. *if (x => 12)*
  - B. *if (x == 3)*
  - C. *if (x = 2)*
  - D. *If (x <= 2)*

Cocokkanlah jawaban Anda dengan Kunci Jawaban Tes Formatif 3 yang terdapat di bagian akhir modul ini. Hitunglah jawaban yang benar. Kemudian, gunakan rumus berikut untuk mengetahui tingkat penguasaan Anda terhadap materi Kegiatan Belajar 3.

$$\text{Tingkat penguasaan} = \frac{\text{Jumlah Jawaban yang Benar}}{\text{Jumlah Soal}} \times 100\%$$

Arti tingkat penguasaan:

90 - 100% = baik sekali

80 - 89% = baik

70 - 79% = cukup

< 70% = kurang

Apabila mencapai tingkat penguasaan 80% atau lebih, Anda dapat meneruskan dengan modul selanjutnya. **Bagus!** Jika masih di bawah 80%, Anda harus mengulangi materi Kegiatan Belajar 3, terutama bagian yang belum dikuasai.

## Kunci Jawaban Tes Formatif

### Tes Formatif 1

- 1) D
- 2) C
- 3) A
- 4) B
- 5) A
- 6) B
- 7) B
- 8) B
- 9) C
- 10) D

### Tes Formatif 2

- 1) A
- 2) B
- 3) B
- 4) C
- 5) A
- 6) B
- 7) B
- 8) C
- 9) A
- 10) A

### Tes Formatif 3

- 1) C
- 2) A
- 3) A
- 4) D
- 5) C
- 6) A
- 7) D
- 8) D
- 9) A
- 10) B

## Praktikum 2



### PENDAHULUAN

---

Kani, M.Kom.

Modul 6 merupakan Modul Praktikum seperti halnya Modul 3. Topik dalam setiap kegiatan praktikum ini adalah: Struktur Java, Tipe Data, Variabel, Operator, Perulangan dan kondisi *if*.

Pada praktikum pada Modul, diharapkan Mahasiswa dapat:

1. mengikuti petunjuk untuk mempraktekkan Struktur Java;
2. mengikuti petunjuk untuk mempraktekkan penggunaan Tipe Data dan variabel;
3. mengikuti petunjuk untuk mempraktekkan Varibel Lokal;
4. mengikuti petunjuk untuk mempraktekkan bagaimana menggunakan varibel *Instance*;
5. mengikuti petunjuk untuk mempraktekkan bagaimana menggunakan variabel *class/static*.
6. mengikuti petunjuk untuk mempraktekkan bagaimana menggunakan bagaimana penggunaan ekspresi;
7. mengikuti petunjuk untuk mempraktekkan bagaimana menggunakan bagaimana penggunaan operator Aritmetika;
8. mengikuti petunjuk untuk mempraktekkan bagaimana menggunakan bagaimana penggunaan operator Penugasan;
9. mengikuti petunjuk untuk mempraktekkan bagaimana menggunakan Operator Relasional;
10. mengikuti petunjuk untuk mempraktekkan bagaimana menggunakan operator Logika;
11. mengikuti petunjuk untuk mempraktekkan bagaimana menggunakan operator *Ternary/Unary*;
12. mengikuti petunjuk untuk mempraktekkan bagaimana menggunakan perulangan *while*;
13. mengikuti petunjuk untuk mempraktekkan bagaimana menggunakan perulangan *for*;

14. mengikuti petunjuk untuk mempraktekkan bagaimana menggunakan perulangan *do..while*;
15. mengikuti petunjuk untuk mempraktekkan bagaimana menggunakan perulangan *if*;
16. mengikuti petunjuk untuk mempraktekkan bagaimana menggunakan perulangan *if..else*;
17. mengikuti petunjuk untuk mempraktekkan bagaimana menggunakan perulangan *if..else if..else*;
18. mengikuti petunjuk untuk mempraktekkan bagaimana menggunakan perulangan *switch..case*.

**PRAKTIKUM 2.1**

## Struktur Java dan Tipe Data

• Pada Modul ini akan dipraktekkan Struktur Program Java mulai dari Dokumentasi, *Package*, *Import*, *Interface*, *Class*, hingga *Method*. Tambahan lainnya adalah akan dipraktekkan bagaimana menggunakan Tipe Data, mulai dari Tipe data primitif dan pembagiannya sampai kepada tipe data Referensi.

### A. STRUKTUR PROGRAM JAVA

Pada Kegiatan Belajar 4 Modul 2 sudah disebutkan dan dijelaskan struktur Program Java, mulai dari *Document Section* (Bagian Dokumentasi), *Package Statement* (Pernyataan Paket), *Import Statement* (Pernyataan *Import*), *Interface Statement* (Pernyataan *Interface*), *Class Definition* (Pendefinisian *Class*) sampai pada *Main Method Definition*.

#### 1. *Documentation Section*

Ada tiga bentuk komentar yang disediakan pada Java, yaitu:

- a. Bentuk `/* ... */`, artinya dibuka dengan tanda `*` dan diakhiri komentar di berikan tanda `*/`
- b. Bentuk `/** ... */`, disebut juga komentar dokumentasi, karena bisa lebih dari 1 baris
- c. Bentuk `//`, hanya akan mengabaikan komentar yang ada setelahnya pada baris tersebut.

Untuk mempraktekkan ketiga bentuk tersebut, berikut langkah-langkahnya:

- a. Buka kembali file *class HaloJava.java*;
- b. Kemudian tambahkan komentar berikut pada bodi kelas di luar *main method*:

```
/* Ini adalah komentar yang tidak di eksekusi */

/* ini adalah komentar terdiri dari
 * Baris1
 * Baris2
```

```
* Baris3  
*/
```

- c. Kemudian dalam bodi *main method* tuliskan komentar berikut:

```
//Fungsi di bawah adalah fungsi cetak ke layar
```

- d. Simpan dan coba jalankan program dengan *Ctrl+F11*;  
e. Kode program selengkapnya:

```
package haloJava;  
  
public class HaloJava {  
    /* Ini adalah komentar yang tidak di eksekusi */  
  
    /* ini adalah komentar terdiri dari  
     * Baris1  
     * Baris2  
     * Baris3  
     */  
  
    public static void main(String[] args) {  
        //Fungsi di bawah adalah fungsi cetak ke layar  
        System.out.println("Halo Java");  
    }  
}
```

- f. Keluaran:

```
Halo Java
```

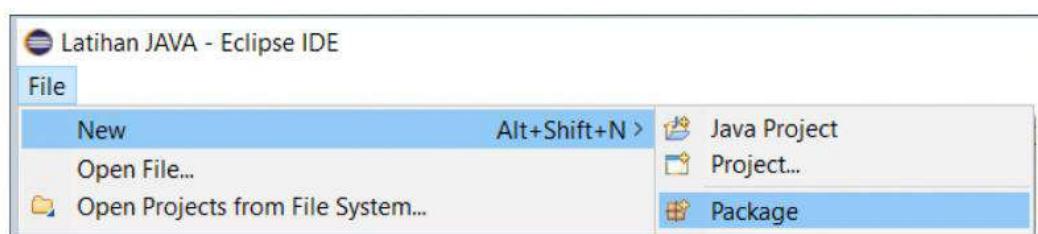
Keterangan:

*Hasil program tetap hanya mencetak “Halo Java”, tidak sedikitpun menuliskan komentar yang ada pada program. Dan komentar-komentar pada program hanya digunakan sebagai penjelas pada program atau potongan program tertentu.*

## 2. *Package Statement*

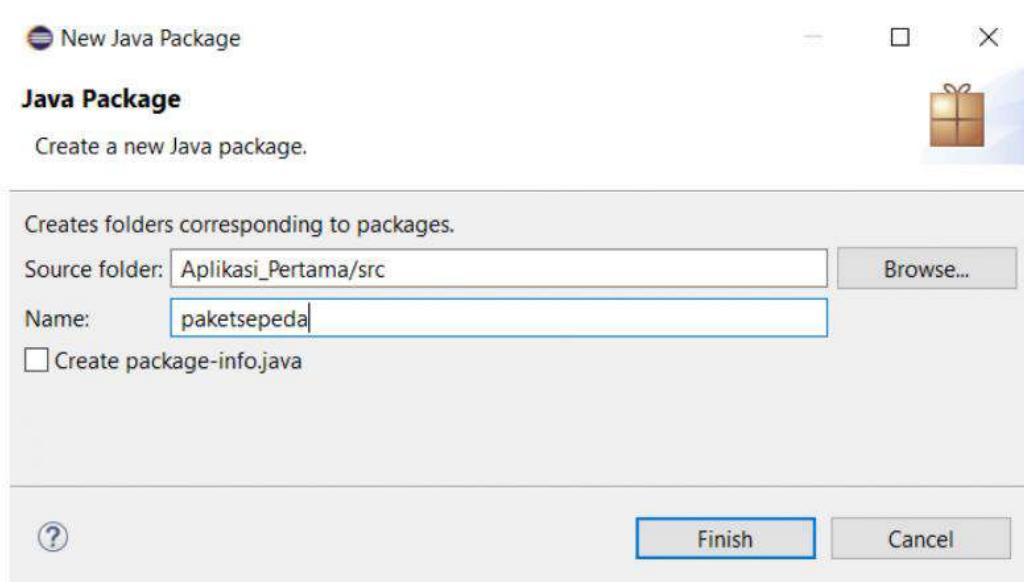
Tak mengapa mengulang kembali sekilas tentang *package* dalam Java, *Package* adalah suatu cara memanajemen *class-class* yang dibuat dalam Java, *class-class* dikelompokkan berdasarkan kemiripan fungsi dan tugas. Satu *package* adalah 1 sub folder di dalam file sistem. Sekarang praktikkan cara pembuatan *package*.

- Aktif pada *Project Aplikasi\_Pertama*;



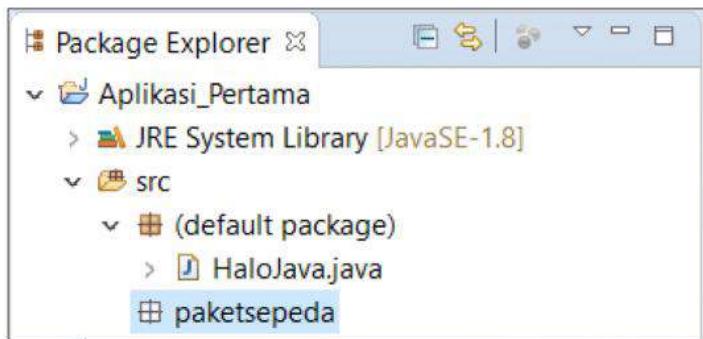
Gambar 6.1  
Menu Untuk Membuat *Package* Baru

- Pada window *New Java Package* pastikan *Source folder* adalah *Aplikasi\_Pertama/src* dan *Name* adalah *paketsepeda* (ini adalah nama *package*). Kemudian klik tombol *Finish*.



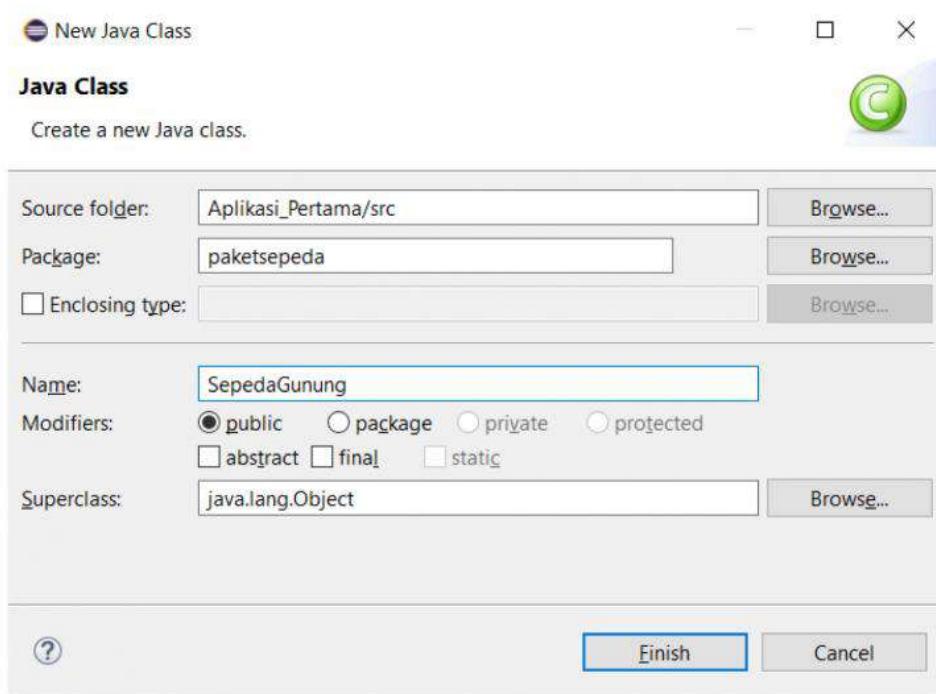
Gambar 6.2  
Window untuk Memasukkan Nama *Package*

- c. Pastikan *package paketsepeda* sudah terbentuk.



Gambar 6.3  
Package yang Sudah Terbuat

- d. Kemudian aktif pada *package paketsepeda*, buat *class* baru dengan nama *SepedaGunung* dan setelah itu klik tombol *Finish*.



Gambar 6.4  
Window Untuk Menuliskan Nama Class

- e. Buka *class SepedaGunung* yang baru saja dibuat, dan perhatikan tepat pada baris pertama tertulis *package paketsepeda*; kemudian dalam *body class* tuliskan kode program berikut:

```
public void SepedaGunung() {  
    System.out.println("Nama : Sepeda Gunung");  
    System.out.println("Harga : 2.300.000 \n");  
}
```

- f. Simpan kode program pada program *class SepedaGunung* sebagai berikut:

```
package paketsepeda;  
  
public class SepedaGunung {  
    public void sepedaGunung() {  
        System.out.println("Nama : Sepeda Gunung");  
        System.out.println("Harga : 2.300.000 \n");  
    }  
}
```

- g. Kemudian buat *class* baru juga pada *package paketsepeda* dengan nama *class SepedaSantai* dan kemudian isikan kode program persis sebagai berikut:

```
package paketsepeda;  
  
public class SepedaSantai {  
    public void sepedaSantai() {  
        System.out.println("Nama : Sepeda Santai");  
        System.out.println("Harga : 1.700.000 \n");  
    }  
}
```

- h. Simpan file *class*

Di atas kita sudah membuat satu *package* dengan nama *package paketsepeda*, dalam *package* tersebut terdapat dua *class* yaitu: *class SepedaGunung* dan *class SepedaSantai*.

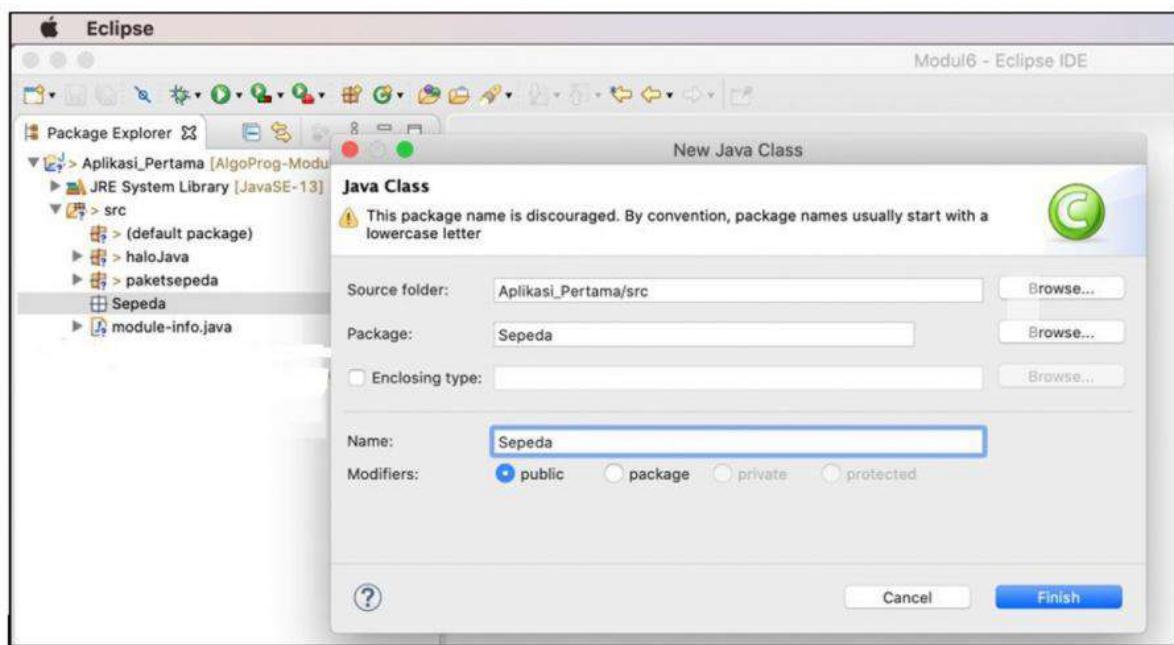
Sekarang bagaimana cara memanggil *package* yang sudah dibuat? Akan dipraktekkan pada bagian *import statement*.

### 3. Import Statement

Impor (*import*) adalah pernyataan yang digunakan untuk merujuk suatu *method/class* yang dinyatakan dalam *package/library* lainnya untuk digunakan pada program induk.

Sekarang kita akan mempraktekkan bagaimana cara memanggil *class-class* dalam *package paketsepeda* yang baru saja dibuat. Ikuti langkah-langkah berikut:

- Buat folder/*package* baru dengan nama *sepeda*.
- Buat *class* baru pada folder/*package* dengan nama *class Sepeda*.



Gambar 6.5  
Window untuk Menuliskan Nama *Class*

- Klik tombol *Finish* untuk membuat *class Sepeda*.
- Buka file *class Sepeda.java*;
- Pada baris pertama ketikkan statemen berikut:

```
import paketsepeda.*;
```

Perintah di atas adalah perintah untuk memanggil/merujuk ke suatu *package* yang namanya *package paketsepeda*, adapun makna tanda \* setelah tanda titik di belakang nama *package* adalah memuat semua *class* yang ada dalam folder *paketsepeda*.

Perhatikan kode program berikut :

```
import paketsepeda.SepedaGunung;
```

Arti dari kode program di atas adalah bahwa hanya *class SepedaGunung* pada *package sepeda* yang dirujuk (digunakan/disertakan).

- f. Pada bagian bodi dari *class Sepeda*, ketikkan kode program berikut:

```
public static void main(String[] args)
{
    System.out.println("Jenis Sepeda");
    System.out.println("-----");
    //memanggil class/method dalam package
    //dengan cara membuat objek baru untuk
    //setiap class
    SepedaGunung sepeda1 = new SepedaGunung();
    sepeda1.sepedaGunung();
    SepedaSantai sepeda2 = new SepedaSantai();
    sepeda2.sepedaSantai();
}
```

- g. Simpan *class Sepeda*. Berikut kode program *class Sepeda* secara lengkap:

```
import paketsepeda.*;

public class Sepeda {
    public static void main(String[] args)  {
        System.out.println("Jenis Sepeda");
        System.out.println("-----");
        //memanggil class/method dalam package
        //dengan cara membuat objek baru untuk
        //setiap class
        SepedaGunung sepeda1 = new SepedaGunung();
        sepeda1.sepedaGunung();
        SepedaSantai sepeda2 = new SepedaSantai();
        sepeda2.sepedaSantai();
    }
}
```

- h. Jalankan program dari file *Sepeda.java* dan keluarannya sebagai berikut:

```
Jenis Sepeda
-----
Nama : Sepeda Gunung
Harga : 2.300.000

Nama : Sepeda Santai
Harga : 1.700.000
```

Jadi antara statemen *package* dan *import* akan sangat berkaitan dalam perpaduan atau pemanggilan *class-class* dalam suatu *package*. Dan untuk melihat jelas struktur antara *package* dan *import* lihat gambar berikut:



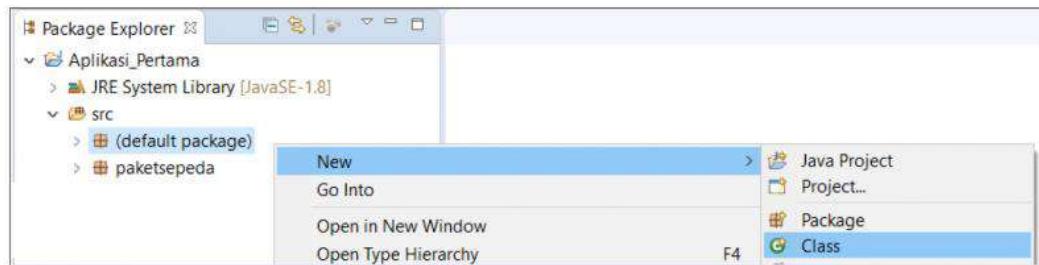
Gambar 6.6  
*Package Default dan Package Paketsepeda*

#### 4. *Class Definition*

Dengan menggunakan IDE Eclipse, *programmer* cenderung dimudahkan dengan fasilitas-fasilitas yang sudah disediakan, misalnya saja pembuatan file/*class* baru, tidak perlu membuatnya secara manual sehingga bisa menghindarkan dari kesalahan-kesalahan sintaks, bukan hanya itu, fitur pembuatan *main method* pun sudah disediakan.

Sekarang kita praktekkan membuat *class* (walaupun secara tidak langsung pembuatan *class* telah di praktekkan di atas, tak mengapa kita mengulangi supaya lancar) sebagai berikut:

- Aktif pada *project Aplikasi\_Pertama*;
- Klik kanan (*Default package*).
- Setelah muncul *pop up* menu, pilih menu *New → Class*;



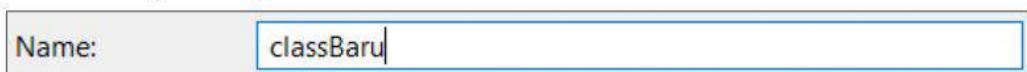
Gambar 6.7  
Menu untuk Membuat *Class* Baru

- d. Pada kolom *Source folder* adalah nama folder/package dimana *class* berada nantinya, jika tidak diisi, maka akan *default* berada pada *(default package)*, Anda bisa menggunakan tombol *Browse* untuk memilih folder tujuan dari *class* yang akan dibuat.



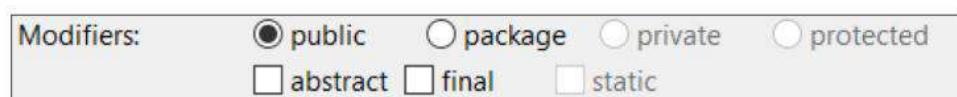
Gambar 6.8  
Folder Arah *Package*

Kolom *name* adalah tempat untuk memberikan nama *class* yang dinginkan, kita isi misalnya dengan nama *classBaru*



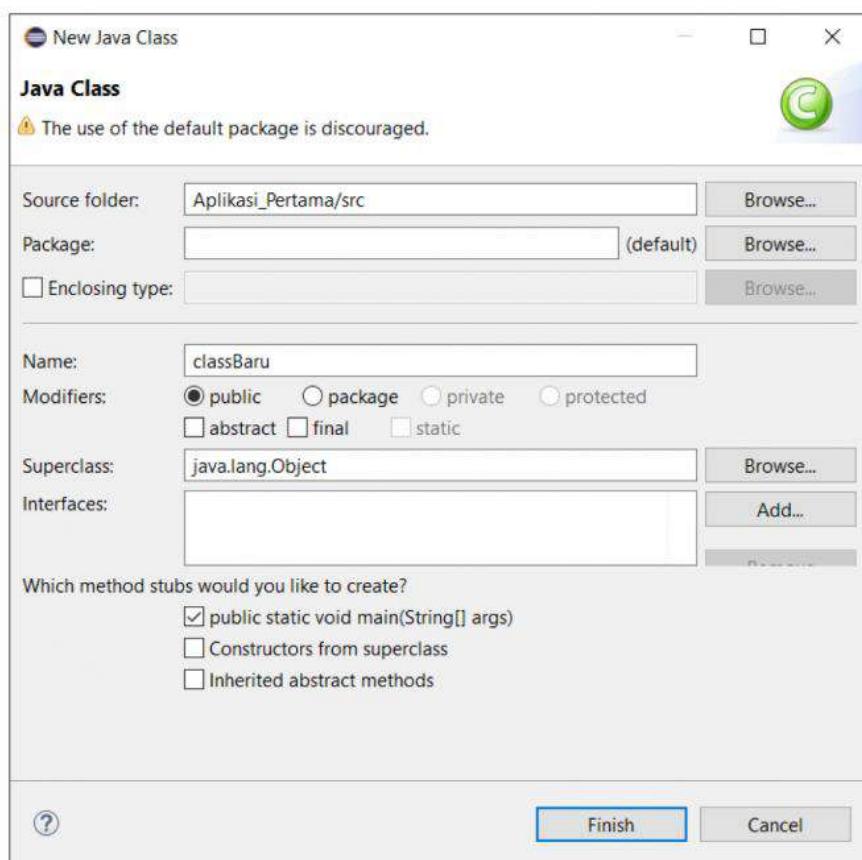
Gambar 6.9  
Nama *Class*

Untuk opsi *Modifiers* terdapat pilihan *public* dan *package*, di bawahnya terdapat pilihan untuk meng-*abstract*-kan *class* dan membuat *class* menjadi *final*, atau boleh memilih keduanya atau tidak memilih sama sekali, ini sangat tergantung kebutuhan Anda terhadap *class* yang akan dibuat. Walaupun Anda memilih/tidak memilih opsi yang ada di *modifiers*, nantinya masih bisa dilakukan modifikasi setelah *class* didefinisikan. Tapi untuk praktik kita pilih *public*.



Gambar 6.10  
Memilih *Modifiers*

Opsi yang lain yang bisa mempermudah *programmer* adalah pembuatan *method* secara otomatis, misalnya saja kita ingin secara otomatis *main method* dalam *class* yang baru dibuat, opsi yang harus Anda centang adalah *public static void main(String[] args)* pada pilihan *Which method stubs would you like to create?*. Lakukan centang bagian *public static void main(String[] args)*.



Gambar 6.11  
Menyertakan *Main Method* ke Dalam *Class*

- e. Klik tombol *Finish* untuk membuat *class* baru.

1
2 public class classBaru {
3
4 public static void main(String[] args) {
5 // TODO Auto-generated method stub
6
7 }
8
9 }

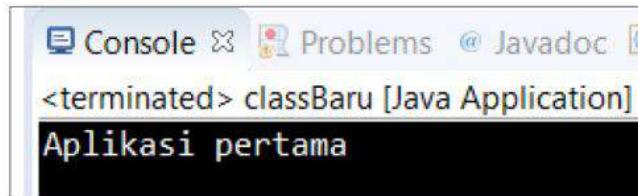
Gambar 6.12  
*Class* yang Terbentuk Berikut *Main Method*-nya

Terbentuk file baru dengan nama file *classBaru.java*, nama *class* mengikuti nama file Java.

- f. Sampai di sini kita sudah bisa melakukan running aplikasi, tapi keluaran kosong, agar supaya ada yang tampil ke layar, maka isikan kode program berikut pada *body method*:

```
System.out.println("Aplikasi pertama");
```

- g. Simpan program dengan tombol Ctrl+S, sekarang jalankan program.



Gambar 6.13  
Output dari Class

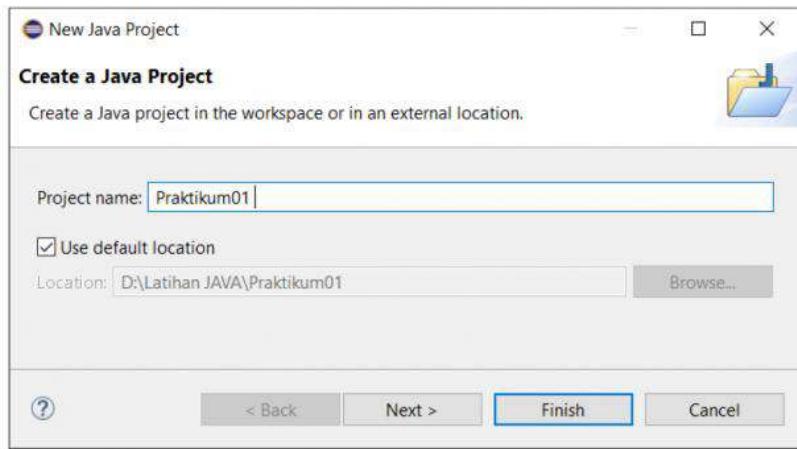
```
Aplikasi pertama
```

## B. TIPE DATA PRIMITIF

### 1. *Tipe Data Integer*

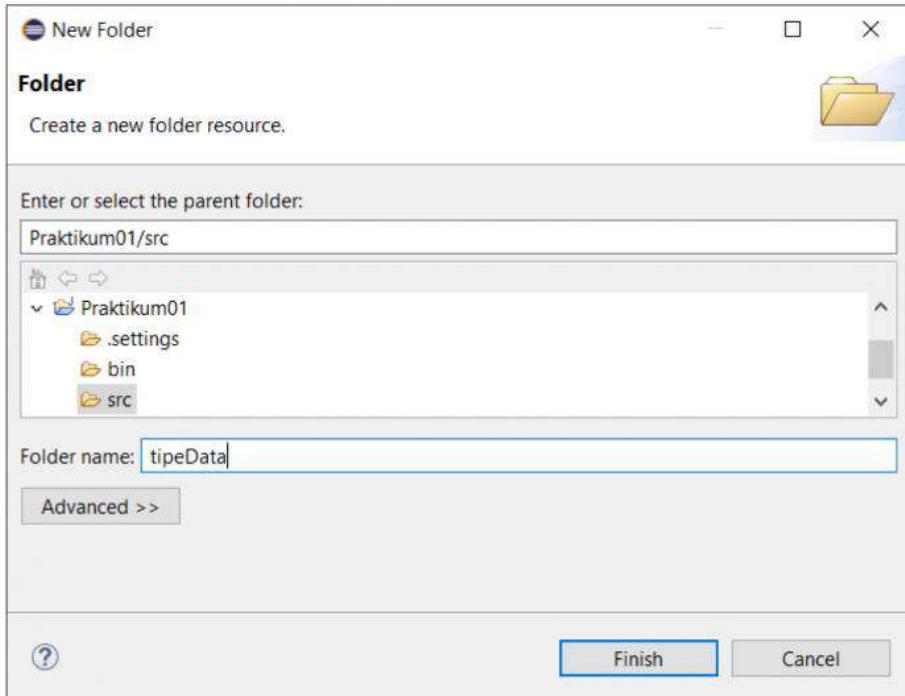
Adapun langkah-langkah untuk buat program sebagai berikut:

- a. Ciptakan *project* baru pada IDE Eclipse, dengan cara klik *File* → *New* → *Java Project*.
- b. Kemudian beri nama *project* dengan nama *Praktikum01* lalu klik tombol *Finish*.



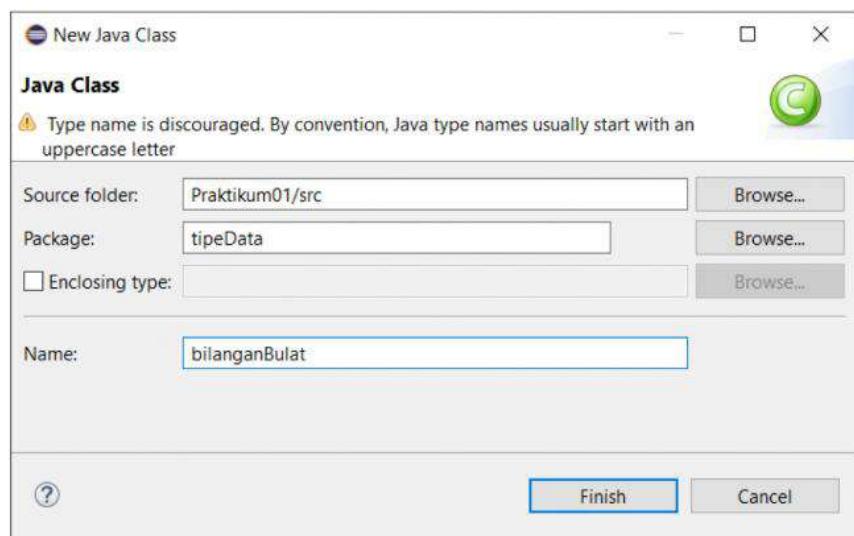
Gambar 6.14  
Window untuk Membuat *Project* Baru

- c. Buat folder baru dalam *project Praktikum01/src* dengan cara klik *File* → *New* → *Folder*, berikan nama folder/package dengan *tipeData*;



Gambar 6.15  
Membuat *Floder/Package* Baru

- d. Aktif pada *Project Praktikum01*, sekarang buat *class* baru dengan cara klik menu *File* → *New* → *Class*
- e. Beri nama *java class* dengan nama *bilanganBulat.java*, dan disimpan dalam folder/package *tipeData*.



Gambar 6.16  
Membuat *Class* Baru Dengan Nama *bilanganBulat*

- f. Ketikkan program berikut pada *body class* yaitu {...}:

```

1 package tipeData;
2
3 public class bilanganBulat {
4
5 }
```

Gambar 6.17  
*Class Bilanganbulat* pada *Package Tipedata*

Sehingga program tampak sebagai berikut:

```

package tipeData;

public class bilanganBulat {
    public static void main(String[] args) {
        //Deklarasi
        byte tipeByte = 50;
        short tipeShort = 0x7e23;
        int tipeInt = 21466666;
        long tipeLong = 922000000;
        //Cetak
        System.out.println("Bilangan Bulat ");
        System.out.println("-----");
    }
}
```

```

        System.out.println("Tipe Byte tipeByte = " + tipeByte
    );
    System.out.println("Tipe Short tipeShort = " +
tipeShort);
    System.out.println("Tipe Int tipeInt = " + tipeInt
);
    System.out.println("Tipe Long tipeLong = " + tipeLong
);
}
}

```

- g. Simpan dengan klik *File* → *Save*;
- h. Kemudian jalankan program di atas dengan menekan *Ctrl+F11* dari *keyboard*. Dan hasilnya sebagai berikut:

```

Bilangan Bulat
-----
Tipe Byte tipeByte = 50
Tipe Short tipeShort = 32291
Tipe Int tipeInt = 21466666
Tipe Long tipeLong = 922000000

```

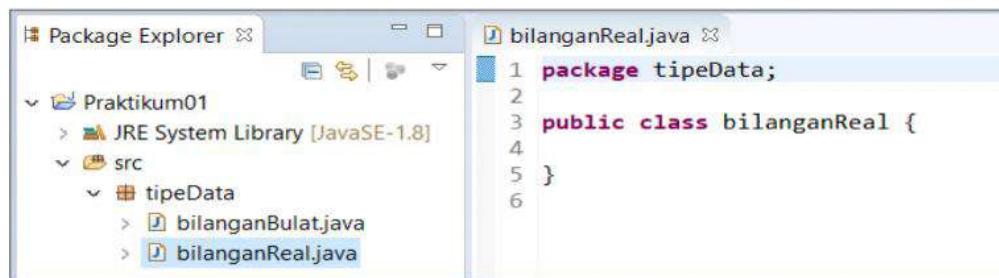
Catatan:

*Pada praktikum01 Tipe Data Integer, secara tidak langsung kita sudah belajar membuat paket (**package**) dan nama paketnya adalah folder yang dibuat dengan nama **tipeData***

## 2. **Tipe Data Real**

Ikuti langkah-langkah praktikum berikut:

- a. Tetap pada *Project Praktikum01*.
- b. Buat *class* baru dengan cara klik menu *File* → *New* → *Class*;
- c. Berikan nama file java dengan nama *bilanganReal.java*, pastikan tetap pada folder/package *tipeData*, lalu klik *Finish*;

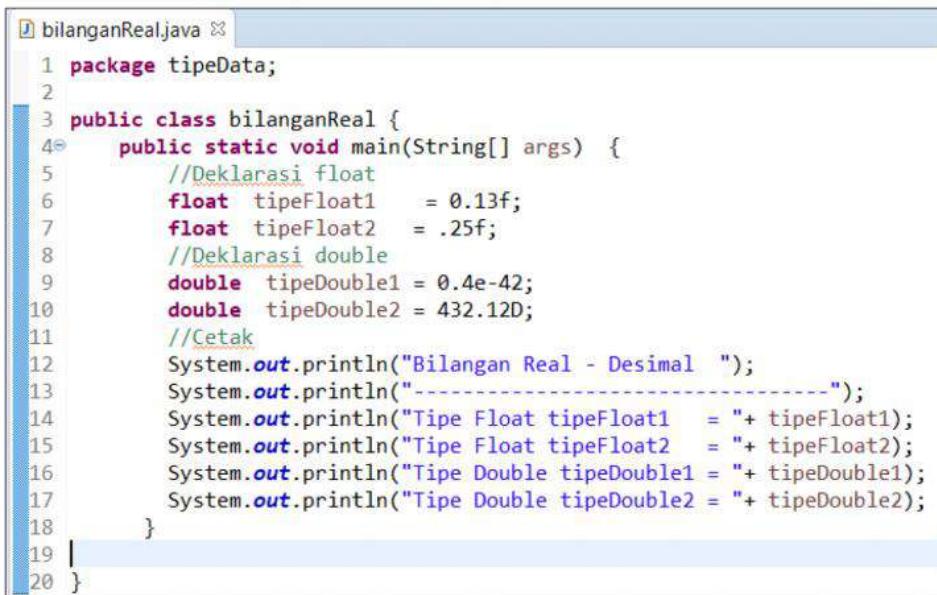


Gambar 6.18  
*Class Bilanganreal pada Package Tipedata*

- d. Selanjutnya ketikkan kode program di bawah ini:

```
...
public static void main(String[] args) {
    //Deklarasi float
    float tipeFloat1 = 0.13f;
    float tipeFloat2 = .25f;
    //Deklarasi double
    double tipeDouble1 = 0.4e-42;
    double tipeDouble2 = 432.12D;
    //Cetak
    System.out.println("Bilangan Real - Desimal ");
    System.out.println("-----");
    System.out.println("Tipe Float tipeFloat1 = "+tipeFloat1);
    System.out.println("Tipe Float tipeFloat2 = "+tipeFloat2);
    System.out.println("Tipe Double tipeDouble1 = "+tipeDouble1);
    System.out.println("Tipe Double tipeDouble2 = "+tipeDouble2);
}
...
```

- e. Lihat keseluruhan kode program sebagai berikut:

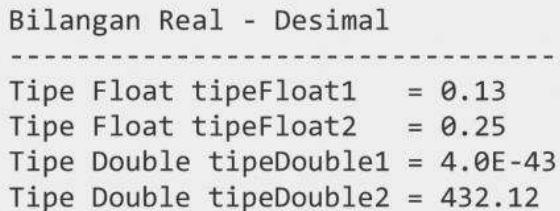


```

1 package tipeData;
2
3 public class bilanganReal {
4     public static void main(String[] args) {
5         //Deklarasi float
6         float tipeFloat1 = 0.13f;
7         float tipeFloat2 = .25f;
8         //Deklarasi double
9         double tipeDouble1 = 0.4e-42;
10        double tipeDouble2 = 432.12D;
11        //Cetak
12        System.out.println("Bilangan Real - Desimal ");
13        System.out.println("-----");
14        System.out.println("Tipe Float tipeFloat1 = " + tipeFloat1);
15        System.out.println("Tipe Float tipeFloat2 = " + tipeFloat2);
16        System.out.println("Tipe Double tipeDouble1 = " + tipeDouble1);
17        System.out.println("Tipe Double tipeDouble2 = " + tipeDouble2);
18    }
19
20 }
```

Gambar 6.19  
*Class Bilanganreal dengan Kode Program Yang Sudah Ada*

- f. Sekarang jalankan program (Ctrl+F11).



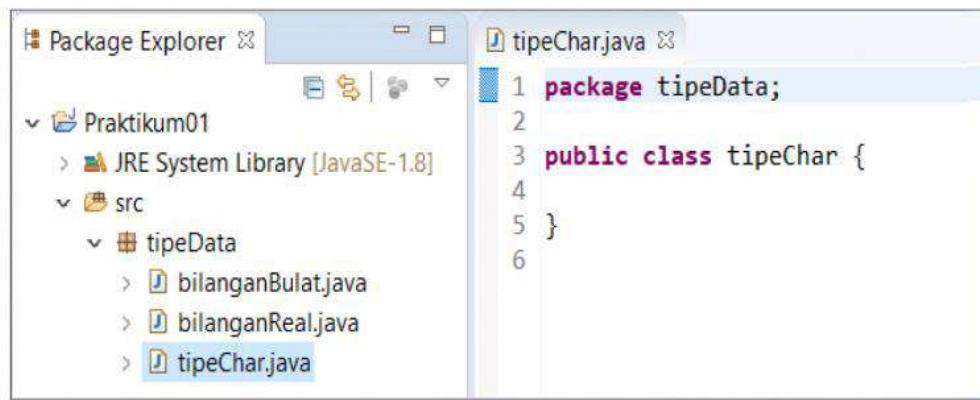
```

Bilangan Real - Desimal
-----
Tipe Float tipeFloat1 = 0.13
Tipe Float tipeFloat2 = 0.25
Tipe Double tipeDouble1 = 4.0E-43
Tipe Double tipeDouble2 = 432.12
```

### 3. *Tipe Data Char*

Mari kita praktekkan:

- a. Buat *class* baru dengan nama *tipeChar.java* tetap pada *Project Praktikum01* dan Paket *tipeData*, Klik menu *File* → *New* → *Class*;



Gambar 6.20  
Class Tipechar pada Package Tipedata

- b. Ketikkan kode program berikut pada kelas yang baru terbentuk:

```

package tipeData;

public class tipeChar {
    public static void main(String[] args) {
        //Deklarasi char
        char char1 = 'a';
        char char2 = 0x0ff32; //adalah '?'
        char char3 = '\t'; //tab

        //Cetak
        System.out.println("Tipe Data Char");
        System.out.println("-----");

        System.out.println("Tipe Char char1      = " + char1);
        System.out.println("Tipe Char char2      = " + char2);
        System.out.println("Tipe Char char3      = a" + char3);
    }
}

```

- c. Keluaran sebagai berikut:

```

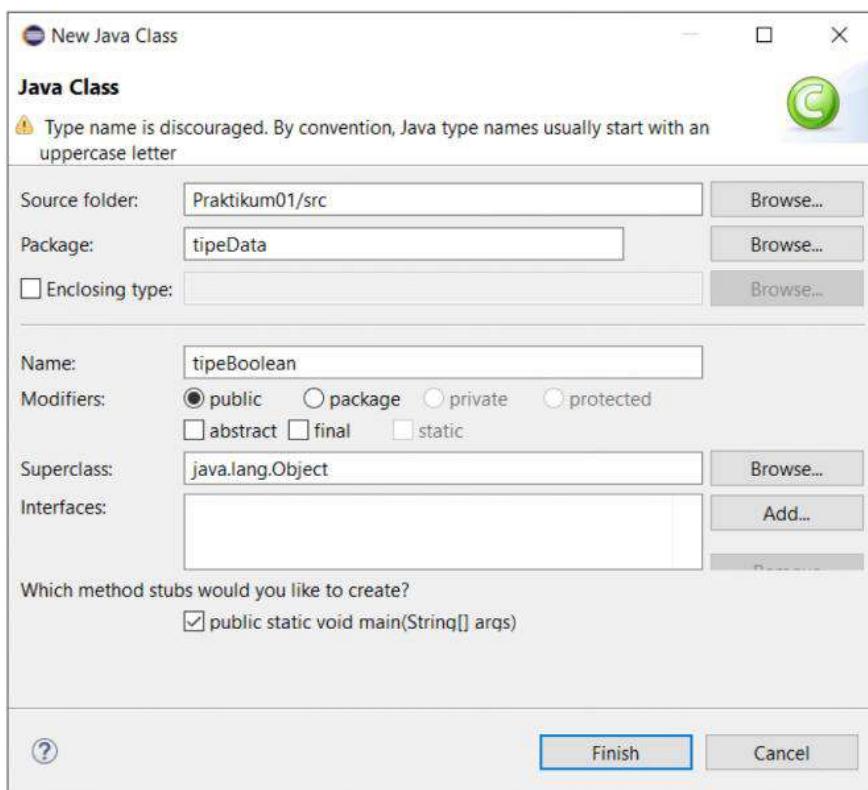
Tipe Data Char
-----
Tipe Char char1      = a
Tipe Char char1      = ?
Tipe Char char1      = a      b

```

#### 4. Tipe Data Logika (Boolean)

Supaya memperkaya pengetahuan, praktikum kali ini kita langsung membuat *main method* agar mengurangi kesalahan pengetikan script pembuatan *main method* (*method main* langsung dibuat oleh IDE Eclipse), Ikuti langkah-langkah praktikum sebagai berikut:

- Buat *class* baru pada *Project Praktikum01* dan *Package tipeData* dengan nama *class* *tipeBoolean*.
- Untuk menyertakan *main method*, pada bagian *Which methos stubs would you like to create*: centang opsi *public static void main(String[] args)*;



Gambar 6.21  
Membuat *Class Tipeboolean* pada *Package Tipedata*

- Klik tombol *Finish*, dan perhatikan pada file *class tipeBoolean.java*, dalam kelas tersebut terdapat *main method*.

```
package tipeData;

public class tipeBoolean {

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

```
// TODO Auto-generated method stub  
}  
}
```

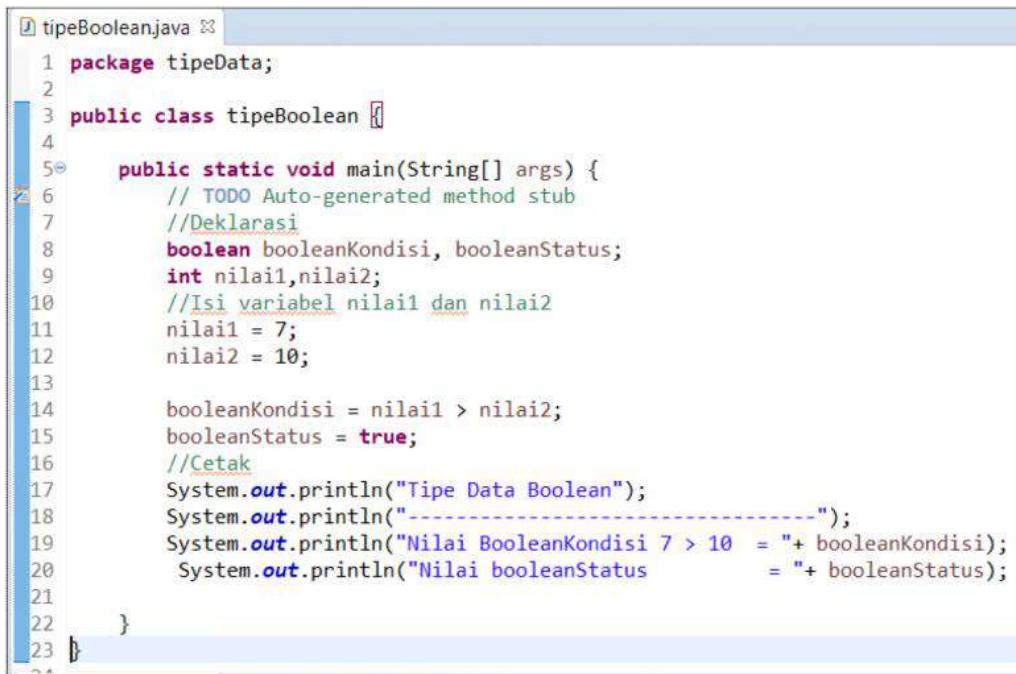
Catatan:

Pada bagian baris **// TODO auto-generate method stub**, (body main method) di bawah baris tersebutlah kita menuliskan kode program kita.

- d. Pada *body main method* isikan kode program berikut:

```
//Deklarasi  
boolean booleanKondisi, booleanStatus;  
int nilai1,nilai2;  
//Isi variabel nilai1 dan nilai2  
nilai1 = 7;  
nilai2 = 10;  
  
booleanKondisi = nilai1 > nilai2;  
booleanStatus = true;  
//Cetak  
System.out.println("Tipe Data Boolean");  
System.out.println("-----");  
  
System.out.println("Nilai BooleanKondisi 7 > 10 = "+  
booleanKondisi);  
System.out.println("Nilai booleanStatus = "+  
booleanStatus);
```

- e. Simpan *class tipeBoolean*;  
f. Kurang lebih *File class tipeBoolean* seperti berikut:



```

1 package tipeData;
2
3 public class tipeBoolean {
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7         //Deklarasi
8         boolean booleanKondisi, booleanStatus;
9         int nilai1,nilai2;
10        //Isi variabel nilai1 dan nilai2
11        nilai1 = 7;
12        nilai2 = 10;
13
14        booleanKondisi = nilai1 > nilai2;
15        booleanStatus = true;
16        //Cetak
17        System.out.println("Tipe Data Boolean");
18        System.out.println("-----");
19        System.out.println("Nilai BooleanKondisi 7 > 10 = "+ booleanKondisi);
20        System.out.println("Nilai booleanStatus = "+ booleanStatus);
21
22    }
23 }

```

Gambar 6.22

Kode Program Tipe Boolean pada Class Class Tipeboolean pada Package Tipedata

- g. Sekarang jalankan program *tipeBoolean* dengan *Ctrl+F11*
- h. Lihat hasilnya sebagai berikut:

```

Tipe Data Boolean
-----
Nilai BooleanKondisi 7 > 10 = false
Nilai booleanStatus = true

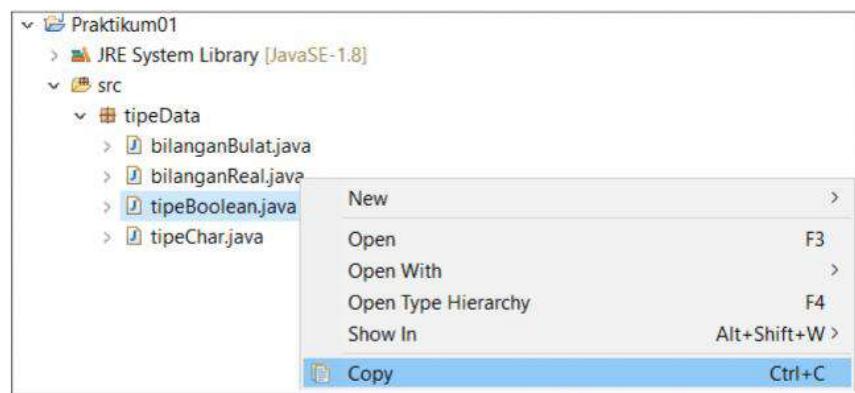
```

Penjelasan:

- Uji: Apakah  $7 > 10$ , ternyata tidak, karena tidak maka dinilai salah atau *false*.
- Variabel *booleanStatus = true*, dari awal tidak diubah.

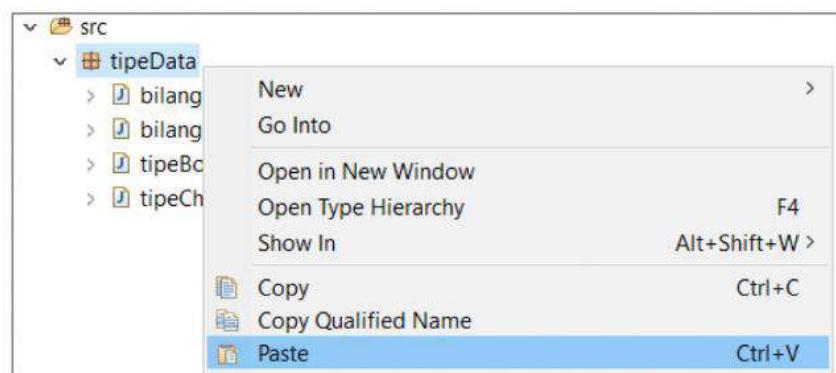
Untuk praktikum Tipe Boolean yang kedua, Variabel *Nilai1* dan *Nilai2* nilainya tidak diinisiasi langsung pada kode program, akan tetapi akan di melalui *keyboard*. Untuk membuatnya ikuti langkah-langkah berikut:

- a. Aktif pada file class *tipeBoolean.java*;
- b. Copy file class tersebut dengan cara klik kanan (*Mouse*), sorot menu *Copy*;



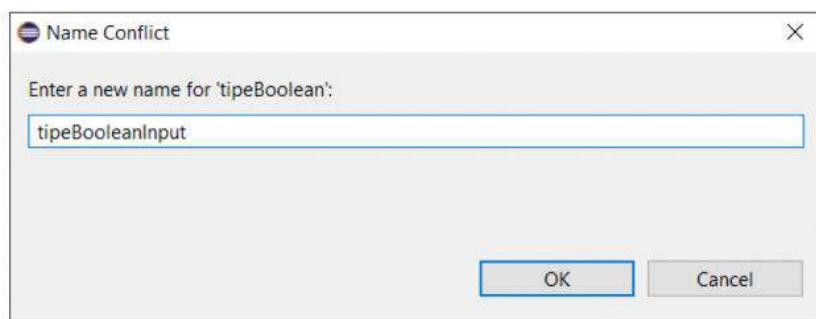
Gambar 6.23  
*Copy File tipeBoolean.java*

- c. Kemudian aktif pada *package/folder tipeData*, dan lakukan klik kanan lalu pilih menu *Paste*;



Gambar 6.24  
*Menu Paste*

- d. Kemudian berikan nama *class* dengan nama *tipeBooleanInput*;



Gambar 6.25  
*Memberikan Nama Baru dari Class TipeBoolean*

- e. Klik tombol *OK*, ketika file *class* berubah menjadi *tipeBooleanInput* maka otomatis nama *class* yang ada dalam file *class* juga berubah menjadi *class tipeBooleanInput*;
- f. Kemudian klik dobel pada File *class tipeBooleanInput.java*, lalu siap untuk dimodifikasi;
- g. Di bawah baris program *package tipeData;* tambahkan kode program berikut:

```
import java.util.Scanner;
```

- h. Di bawah baris kode program komentar `// TODO Auto-generate method stub` tambahkan kode program berikut:

```
Scanner input= new Scanner(System.in);
```

- i. Hapus baris berikut:

```
nilai1 = 7;  
nilai2 = 10;
```

- j. Kemudian di bawah baris komentar `//Isi variabel nilai1 dan nilai2` ketikkan kode program berikut:

```
System.out.println("Masukkan Nilai 1 dan Nilai 2");  
System.out.println("-----");  
System.out.print("Masukkan Nilai 1 : ");  
nilai1 = input.nextInt();  
System.out.print("Masukkan Nilai 2 : ");  
nilai2 = input.nextInt();
```

k. Baris:

```
booleanStatus = true;
```

dirubah menjadi

```
booleanStatus = booleanKondisi;
```

l. Kode Program:

```
System.out.println("Nilai BooleanKondisi 7 > 10 = "+  
booleanKondisi);
```

Dirubah menjadi:

```
System.out.println("Nilai BooleanKondisi nilai 1 > Nilai 2 = "+  
booleanKondisi);
```

m. Simpan file *tipeBooleanInput.java*, kurang lebih tampilan file *class tipeBooleanInput.java* sebagai berikut:

```
package TipeData;  
  
import java.util.Scanner;  
  
public class TipeBooleanInput {  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        Scanner input = new Scanner(System.in);  
        //Deklarasi  
        boolean booleanKondisi, booleanStatus;  
        int nilai1,nilai2;  
        //Isi variabel nilai1 dan nilai2  
        System.out.println("Masukkan Nilai 1 dan Nilai 2");  
        System.out.println("-----");  
    };
```

```

        System.out.print("Masukkan Nilai 1 : ");
        nilai1 = input.nextInt();
        System.out.print("Masukkan Nilai 2 : ");
        nilai2 = input.nextInt();

        booleanKondisi = nilai1 > nilai2;
        booleanStatus = true;
        //Cetak
        System.out.println("Tipe Data Boolean");
        System.out.println("-----");
    });
    System.out.println("Nilai BooleanKondisi Nilai 1 > Nilai 2 = " + booleanKondisi);
    System.out.println("Nilai booleanStatus = " +
    "+ booleanStatus);

}
}

```

- n. Dilihat gambarnya sebagai berikut:

```

1 package TipeData;
2
3 import java.util.Scanner;
4
5 public class TipeBooleanInput {
6     public static void main(String[] args) {
7         // TODO Auto-generated method stub
8         Scanner input = new Scanner(System.in);
9         //Deklarasi
10        boolean booleanKondisi, booleanStatus;
11        int nilai1,nilai2;
12        //Isi variabel nilai1 dan nilai2
13        System.out.println("Masukkan Nilai 1 dan Nilai 2");
14        System.out.println("-----");
15        System.out.print("Masukkan Nilai 1 : ");
16        nilai1 = input.nextInt();
17        System.out.print("Masukkan Nilai 2 : ");
18        nilai2 = input.nextInt();
19
20        booleanKondisi = nilai1 > nilai2;
21        booleanStatus = true;
22        //Cetak
23        System.out.println("Tipe Data Boolean");
24        System.out.println("-----");
25        System.out.println("Nilai BooleanKondisi Nilai 1 > Nilai 2 = " + booleanKondisi);
26        System.out.println("Nilai booleanStatus = " + booleanStatus);
27
28    }
29 }

```

Gambar 6.26  
Kode Program dari Class *TipeBooleaninput*

- o. Jalankan program (*Ctrl+F11*)
- p. Isi *Nilai 1* dengan 4, tekan *Enter*
- q. Isi *Nilai 2* dengan 5, tekan *Enter* dan hasilnya sebagai berikut:

```
Masukkan Nilai 1 dan Nilai 2
-----
Masukkan Nilai 1 : 4
Masukkan Nilai 2 : 5
Tipe Data Boolean
-----
Nilai BooleanKondisi Nilai 1 > Nilai 2 = false
Nilai booleanStatus      = true
```

Keterangan:

- Nilai 1 (4) > Nilai 2 (5) adalah tidak benar maka nilainya *false*
- r. Jalankan ulang program (*Ctrl+F11*)
- s. Isi *Nilai 1* dengan 5, tekan *Enter*
- t. Isi *Nilai 2* dengan 4, tekan *Enter* dan hasilnya sebagai berikut:

```
Masukkan Nilai 1 dan Nilai 2
-----
Masukkan Nilai 1 : 5
Masukkan Nilai 2 : 4
Tipe Data Boolean
-----
Nilai BooleanKondisi Nilai 1 > Nilai 2 = true
Nilai booleanStatus      = true
```

Keterangan:

- Nilai 1 (5) > Nilai 2 (4) adalah benar maka nilainya *true*



LATIHAN

Untuk memperdalam pemahaman Anda mengenai materi di atas, kerjakanlah latihan berikut!

- 1) Sebuah potongan program yang tidak lengkap sebagai berikut:

```
public class lengkapiScript1 {  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        ...  
        System.out.println("Nilai i : "+i);  
        System.out.println("Nilai j : "+j);  
        System.out.println("Nilai k : "+k);  
    }  
}
```

Lengkapi kode program di atas pada bagian ... untuk menghasilkan keluaran seperti di bawah ini:

```
Nilai i : 5  
Nilai j : 8  
Nilai k : 12
```

- 2) Sebuah potongan program yang tidak lengkap sebagai berikut:

```
public class LangkapiScript2 {  
  
    public static void main(String[] args) {  
  
        ...  
  
        System.out.println("Tipe data char : " + a);  
        System.out.println("Tipe data integer : " + i);  
        System.out.println("Tipe data byte : " + b);  
        System.out.println("Tipe data short : " + s);  
        System.out.println("Tipe data float : " + f);  
        System.out.println("Tipe data double : " + d);  
    }  
}
```

Lengkapi kode program di bagian atas pada bagian ... untuk menghasilkan keluaran berikut:

```
Tipe data char    : G
Tipe data integer : 89
Tipe data byte   : 4
Tipe data short  : 56
Tipe data float   : 4.7333436
Tipe data double  : 4.355453532
```

### *Petunjuk Jawaban Latihan*

- 1) Untuk menghasilkan keluaran sebagai berikut:

```
Nilai i : 5
Nilai j : 8
Nilai k : 12
```

maka program harus lengkap sebagai berikut:

```
public class lengkapiScript1 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int i,j,k;
        i = 5;
        j = 8;
        k = 12;
        System.out.println("Nilai i : "+i);
        System.out.println("Nilai j : "+j);
        System.out.println("Nilai k : "+k);
    }
}
```

- 2) Untuk menghasilkan keluaran sebagai berikut:

```
Tipe data char    : G
Tipe data integer : 89
Tipe data byte   : 4
Tipe data short  : 56
Tipe data float   : 4.7333436
Tipe data double  : 4.355453532
```

Maka kode program dilengkapi sebagai berikut:

```
public class LangkapiScript2 {  
  
    public static void main(String[] args) {  
  
        char    a = 'G';  
        int     i = 89;  
        byte    b = 4;  
        short   s = 56;  
        float   f = 4.7333436F;  
        double  d = 4.355453532;  
  
        System.out.println("Tipe data char    : " + a);  
        System.out.println("Tipe data integer : " + i);  
        System.out.println("Tipe data byte   : " + b);  
        System.out.println("Tipe data short  : " + s);  
        System.out.println("Tipe data float  : " + f);  
        System.out.println("Tipe data double : " + d);  
    }  
}
```



## RANGKUMAN

---

Struktur Program Java, mulai dari *Document Section (Bagian Dokumentasi)*, *Package Statement (Pernyataan Paket)*, *Import Statement (Pernyataan Import)*, *Interface Statement (Pernyataan Interface)*, *Class Definition (Pendefinisan Class)* sampai pada *Main Method Definition*.

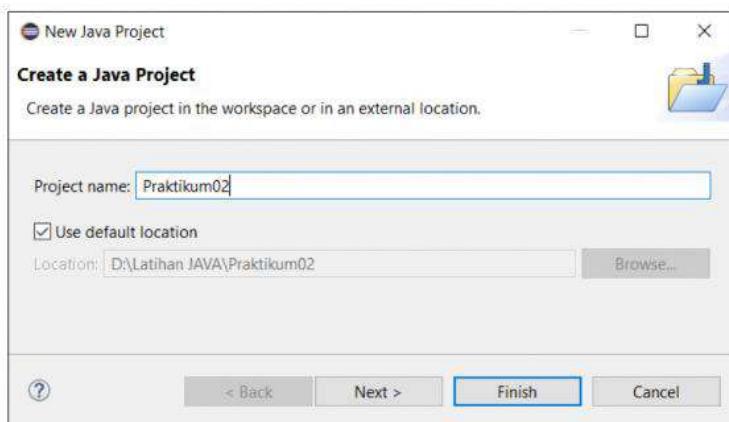
Dalam program Java kita akan mengenal dua jenis tipe data: yang pertama yaitu tipe data *primitive* (primitif) dan yang kedua adalah *reference* (referensi).

**PRAKTIKUM 2.2****Variabel Java**

• Pada Praktikum 2.2 kita akan mempraktekkan penggunaan variabel pada Java, baik variabel lokal, variabel *instance*, maupun variabel *class*.

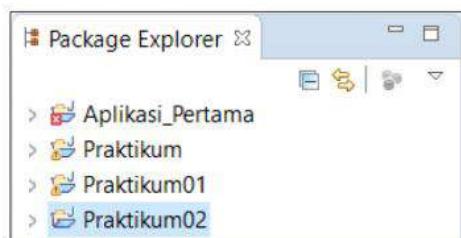
Untuk praktikum pada Modul ini, kita akan membuat *project* baru, dengan langkah-langkah berikut:

1. Klik menu *File* → *New* → *Java Project*.



Gambar 6.27  
Membuat *Project* Baru *Praktikum02*

2. Klik tombol *Finish* untuk membuat *project* *Praktikum02*.



Gambar 6.28  
Daftar *Project* Termasuk *Praktikum02*

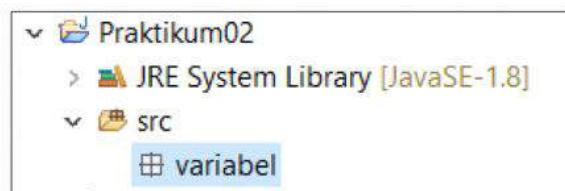
**A. VARIABEL LOKAL**

Variabel lokal dideklarasikan dalam *method*, *constructor*, atau blok. Variabel lokal dibuat saat *method*, *constructor* atau blok mulai dijalankan dan

akan dihapus saat selesai dijalankan. *Modifier* akses tidak dapat digunakan untuk variabel lokal, variabel lokal hanya dapat digunakan di dalam *method*, *constructor* atau blok tempat pendeklarasinya. Tidak ada nilai *default* untuk variabel lokal sehingga variabel lokal harus dideklarasikan dan inisialisasi sebelum digunakan.

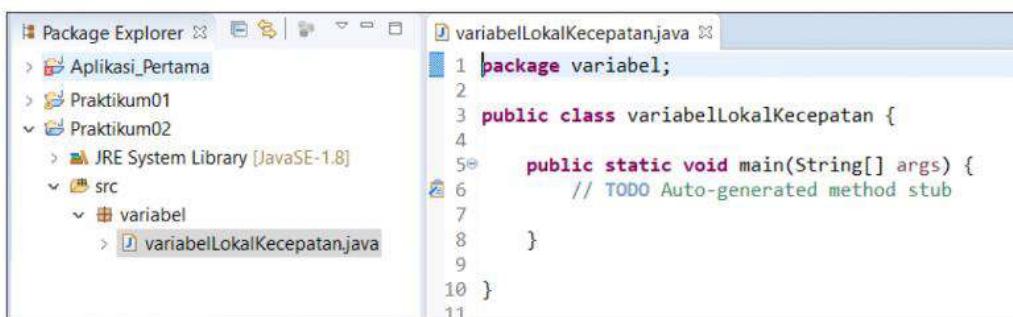
Sekarang kita mempraktekkan pembuatan dan penggunaan variabel lokal dalam Java sebagai berikut:

1. Buat Folder/package dengan nama *variabel* pada folder *src* untuk *project Praktikum02*.



Gambar 6.29  
Folder/Package Variabel

2. Ciptakan *class* baru dalam folder *variabel* dengan nama *class variabelLokalKecepatan.java*, dan jangan lupa membuat *main method*.



Gambar 6.30  
Class variabelLokalKecepatan

3. Buat *method* baru dengan nama *tambahKecepatan* dalam tubuh *class* dan di luar *method main*, berikut kode programnya:

```
public void tambahKecepatan() {
    int cepat = 0;
    System.out.println("Kecepatan Awal : " + cepat);
    cepat = cepat + 40;
```

```
    System.out.println("Penambahan Kecepatan : " + cepat);
}
```

4. Kemudian isi kode program pada *main method* sebagai berikut:

```
variabelLokalKecepatan kecepatan = new variabelLokalKecepatan();
kecepatan.tambahKecepatan();
```

5. Jika dilihat *class variabelLokalKecepatan* sebagai berikut:

```
package variabel;

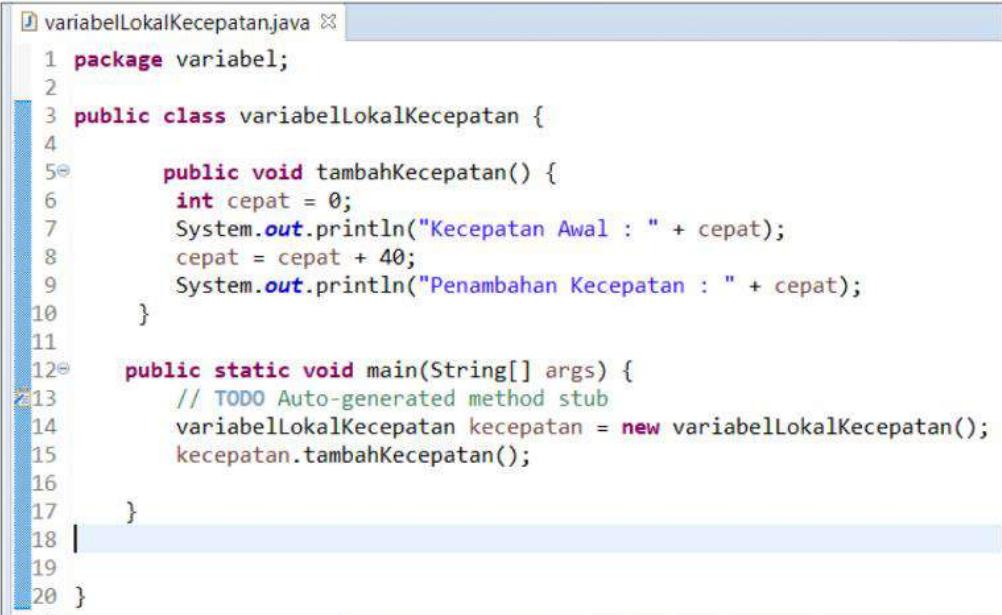
public class variabelLokalKecepatan {

    public void tambahKecepatan() {
        int cepat = 0;
        System.out.println("Kecepatan Awal : " +
cepat);
        cepat = cepat + 40;
        System.out.println("Penambahan Kecepatan : " +
cepat);
    }

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        variabelLokalKecepatan kecepatan = new
        variabelLokalKecepatan();
        kecepatan.tambahKecepatan();

    }
}
```

6. Simpan *class variabelLokalKecepatan*;



```

1 package variabel;
2
3 public class variabelLokalKecepatan {
4
5     public void tambahKecepatan() {
6         int cepat = 0;
7         System.out.println("Kecepatan Awal : " + cepat);
8         cepat = cepat + 40;
9         System.out.println("Penambahan Kecepatan : " + cepat);
10    }
11
12    public static void main(String[] args) {
13        // TODO Auto-generated method stub
14        variabelLokalKecepatan kecepatan = new variabelLokalKecepatan();
15        kecepatan.tambahKecepatan();
16    }
17}
18
19
20
21

```

Gambar 6.31  
Kode Program *Class variabelLokalKecepatan*

7. Kemudian jalankan (*Ctrl+F11*), hasilnya sebagai berikut:

```

Kecepatan Awal : 0
Penambahan Kecepatan : 40

```

Penjelasan:

- Variabel **cepat** adalah sebuah varibel lokal yang didefinisikan dan diinisiasi di dalam *method tambahKecepatan*
- Variabel **cepat** hanya akan bisa digunakan dalam *method tambahKecepatan*. Jika digunakan pada *method* yang lain, maka akan terjadi *error*.
- Pada *class variabelLokalKecepatan* dibuat objek bernama *kecepatan*, pembuatan objek ditandai dengan muncul *keyword new* yang diikuti oleh nama *class (instance)*, fokus pada kode program:

<pre> variabelLokalKecepatan variabelLokalKecepatan(); </pre>	<pre> kecepatan = new </pre>
---	------------------------------

objek *kecepatan* adalah objek *reference* dari *class variabelLokalKecepatan*.

- Objek *kecepatan* bisa memiliki properti dari *class* yang bernama *method tambahKecepatan()*.

Penulisan `kecepatan.tambahKecepatan()` bisa dilakukan untuk memanggil *method* tersebut.

Sekarang Anda boleh menguji ciri-ciri variabel lokal pada program yang baru saja dibuat, misalnya saja kita menguji salah satu ciri-ciri dari variabel lokal yaitu: Pada variabel lokal tidak berlaku *access modifiers*. Untuk mengujinya, silahkan menambahkan *modifier* di depan variabel lokal, misalnya *modifiers protected* kita tambahkan di depan variabel `int cepat = 0;`

```
package variabel;

public class variabelLokalKecepatan {

    public void tambahKecepatan() {
        protected int cepat = 0;
        System.out.println("Kecepatan Awal : " + cepat);
        cepat = cepat + 40;
        System.out.println("Penambahan Kecepatan : " +
        cepat);
    }

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        variabelLokalKecepatan kecepatan = new
        variabelLokalKecepatan();
        kecepatan.tambahKecepatan();

    }
}
```

Simpan kemudian jalankan programnya, kita akan mendapatkan pesan *error* sebagai berikut:

```
Exception in thread "main" java.lang.Error: Unresolved
compilation problem:
    Illegal modifier for parameter cepat; only final
    is permitted
```

```

at
variabel.variabelLokalKecepatan.tambahKecepatan(variabelL
okalKecepatan.java:7)
at
variabel.variabelLokalKecepatan.main(variabelLokalKecepat
an.java:16)

```

Satu-satunya *modifier* yang diijinkan adalah *modifier final*, itu pun dengan beberapa keterbatasan tertentu.

## B. VARIABEL INSTANCE

Variabel *instance* akan dieksekusi apabila objek dieksekusi. Variabel *instance* tidak bisa dideklarasikan dengan *keyword static*. Variabel *instance* dapat digunakan oleh *method*, baik *constructor* atau *method* yang lain. Untuk tipe numerik *default* adalah 0, tipe data *boolean* nilai *default* adalah *false*, dan tipe *reference* adalah *null*.

Untuk lebih melihat secara jelas, praktikan sebagai berikut:

1. Buat *class* baru dalam folder/*package variabel*
2. Berikan nama *class* dengan *variabelInstanceSepeda.java*.
3. Sertakan *main method* dalam *class variabelInstanceSepeda*.
4. Ketik potongan kode program di bawah tepat di bodi *class variabelInstanceSepeda*;

```

//variabel instance ini dapat digunakan oleh
//semua subkelas.
public String namaSepeda;
public int hargaSepeda;

```

5. Buat sebuah *constructor* setelah kode program pada poin 4 sebagai berikut:

```

//variabel namaSepeda dan hargaSepeda
//diinisialisasikan dalam constructor.
//diinisialisasikan dalam constructor.

```

```
public variabelInstanceSepeda(String NamaSepeda, int
HargaSepeda)
{
    namaSepeda = NamaSepeda;
    hargaSepeda = HargaSepeda;
}
```

Catatan:

*Constructor yang dibangun dalam class mengikuti nama class yang membawanya, contoh potongan kode program di atas nama class nya adalah variabelInstanceSepeda maka nama constructornya adalah mengikuti nama class.*

6. Kemudian buat *method* dengan nama *tampilSepeda* untuk mencetak informasi Sepeda;

```
//Method ini menampilkan informasi sepeda.
public void tampilSepeda()
{
    System.out.println("Nama sepeda : "+namaSepeda);
    System.out.println("Harga sepeda : "+hargaSepeda);
}
```

7. Kemudian tambahkan kode program berikut pada *main method* sebagai berikut:

```
variabelInstanceSepeda sepeda =
    new variabelInstanceSepeda("Sepeda Gunung",100000);
sepeda.tampilSepeda();
```

8. Simpan *class* (*Ctrl+S*)
9. Kode program secara keseluruhan:

```
package variabel;

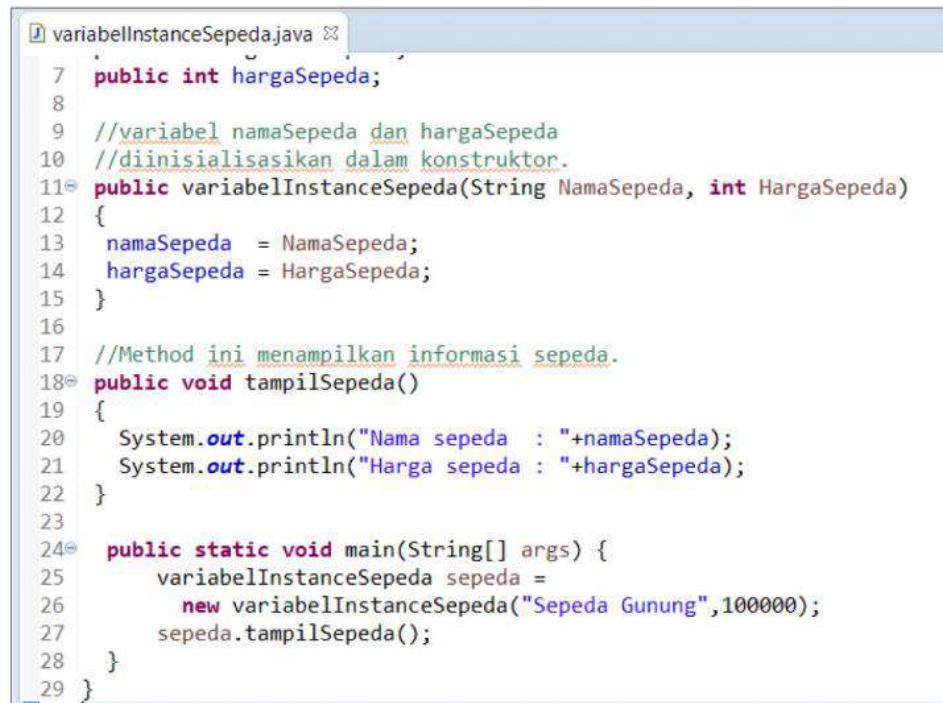
public class variabelInstanceSepeda {
    //variabel instance ini dapat digunakan oleh
```

```
//semua subkelas.  
public String namaSepeda;  
public int hargaSepeda;  
  
//variabel namaSepeda dan hargaSepeda  
//diinisialisasikan dalam constructor.  
public variabelInstanceSepeda(String NamaSepeda, int  
HargaSepeda)  
{  
    namaSepeda = NamaSepeda;  
    hargaSepeda = HargaSepeda;  
}  
  
//Method ini menampilkan informasi sepeda.  
public void tampilSepeda()  
{  
    System.out.println("Nama sepeda : "+namaSepeda);  
    System.out.println("Harga sepeda : "+hargaSepeda);  
}  
  
public static void main(String[] args) {  
    variabelInstanceSepeda sepeda =  
        new variabelInstanceSepeda("Sepeda Gunung",100000);  
    sepeda.tampilSepeda();  
}  
}
```

#### 10. Jalankan Program (*Ctrl+F11*)

```
Nama sepeda : Sepeda Gunung  
Harga sepeda : 100000
```

#### 11. Gambar File class *variabelInstanceSepeda.java*.



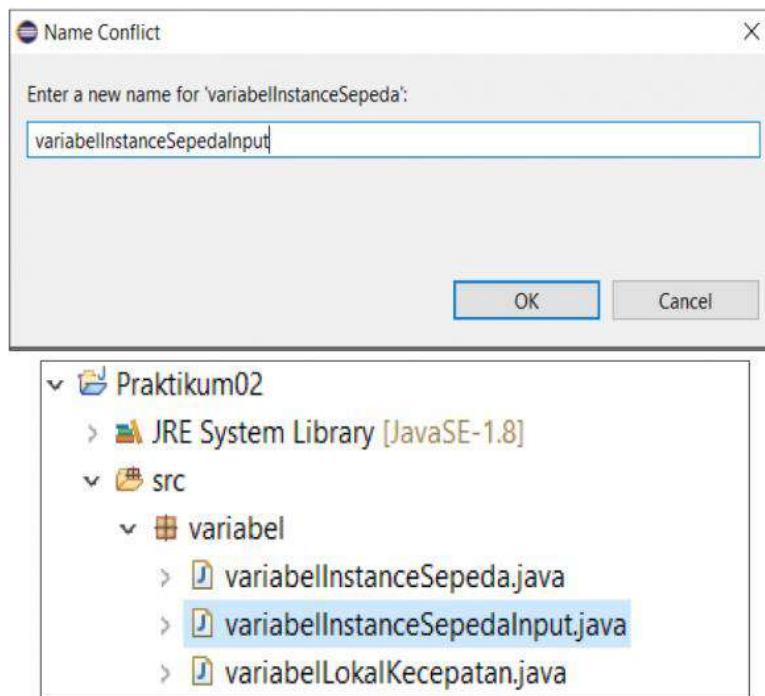
```
variabelInstanceSepeda.java
7  public int hargaSepeda;
8
9  //variabel namaSepeda dan hargaSepeda
10 //diinisialisasikan dalam konstruktur.
11 public variabelInstanceSepeda(String NamaSepeda, int HargaSepeda)
12 {
13     namaSepeda = NamaSepeda;
14     hargaSepeda = HargaSepeda;
15 }
16
17 //Method ini menampilkan informasi sepeda.
18 public void tampilSepeda()
19 {
20     System.out.println("Nama sepeda : "+namaSepeda);
21     System.out.println("Harga sepeda : "+hargaSepeda);
22 }
23
24 public static void main(String[] args) {
25     variabelInstanceSepeda sepeda =
26         new variabelInstanceSepeda("Sepeda Gunung",100000);
27     sepeda.tampilSepeda();
28 }
29 }
```

Gambar 6.32  
Kode Program *Class variabelInstanceSepeda*

Sekarang kita membuat sebuah variasi, Nama Sepeda dan Harga Sepeda kita input melalui *keyboard* dan kemudian ditampilkan ke layar.

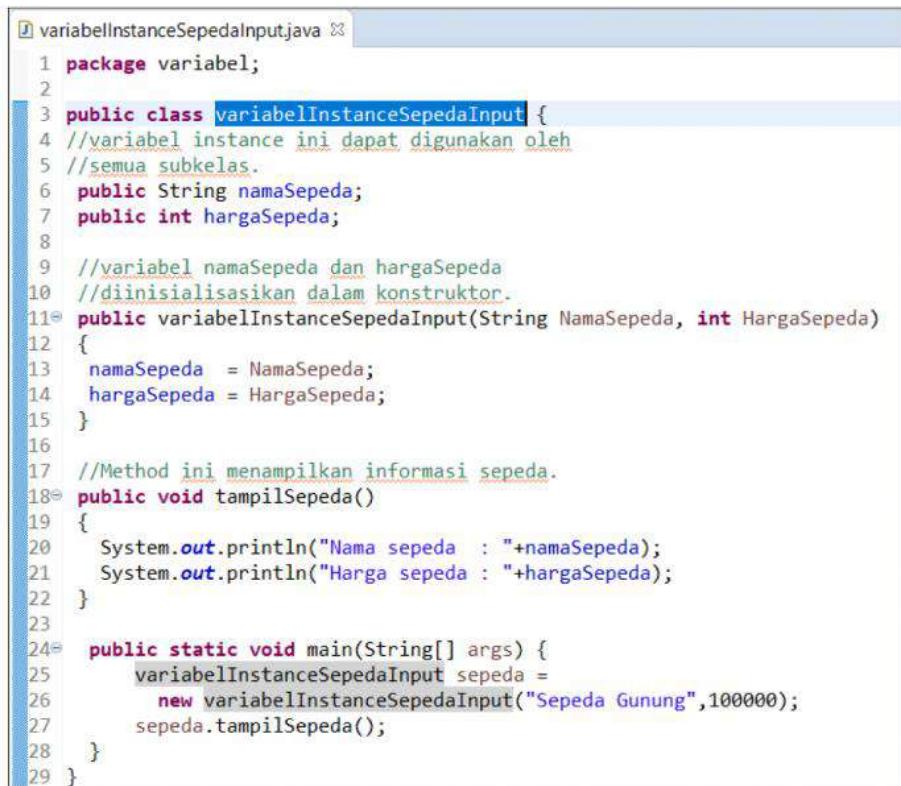
Adapun langkah-langkahnya sebagai berikut:

1. *Copy file class variabelInstanceSepeda.java.*
2. Kemudian paste di *folder/package variabel*.
3. Isi nama *file class* baru dengan nama *variabelInstanceSepedaInput*;



Gambar 6.33  
Proses *Copy-Paste* dari *variabellInstanceSepeda* ke  
*variabellInstanceSepedaInput*

Pada saat mengubah dari *variabellInstanceSepeda.java* ke *variabellInstanceSepedaInput.java* Anda tidak perlu mengubah *constructor* dan objek *reference* yang mengarah ke nama *constructor*, perhatikan gambar di bawah pada baris 11 dan baris 25-26, kode program baris-baris tersebut langsung mengikuti nama *class* yang membungkus dirinya.



```

1 package variabel;
2
3 public class variabelInstanceSepedaInput {
4 //variabel instance ini dapat digunakan oleh
5 //semua subkelas.
6 public String namaSepeda;
7 public int hargaSepeda;
8
9 //variabel namaSepeda dan hargaSepeda
//diinisialisasikan dalam konstruktur.
10 public variabelInstanceSepedaInput(String NamaSepeda, int HargaSepeda)
11 {
12     namaSepeda = NamaSepeda;
13     hargaSepeda = HargaSepeda;
14 }
15
16
17 //Method ini menampilkan informasi sepeda.
18 public void tampilSepeda()
19 {
20     System.out.println("Nama sepeda : "+namaSepeda);
21     System.out.println("Harga sepeda : "+hargaSepeda);
22 }
23
24 public static void main(String[] args) {
25     variabelInstanceSepedaInput sepeda =
26         new variabelInstanceSepedaInput("Sepeda Gunung",100000);
27     sepeda.tampilSepeda();
28 }
29 }
```

Gambar 6.34  
Kode Program Class *variabelInstanceSepedaInput*

4. Sekarang lakukan modifikasi dengan menambahkan kode program berikut di bawah **package variabel;**

```
import java.util.Scanner;
```

5. Modifikasi/tambahkan di tubuh *main method* dari kode program berikut:

```

public static void main(String[] args) {
    variabelInstanceSepeda sepeda =
        new variabelInstanceSepeda("Sepeda Gunung",100000);
    sepeda.tampilSepeda();
}
```

menjadi:

```

public static void main(String[] args) {
    String namaSepeda;
    int hargaSepeda;
    Scanner input = new Scanner(System.in);
    System.out.println("Masukkan Nama Sepeda dan Harga
Sepeda");
    System.out.println("-----");
    System.out.print("Masukkan Nama Sepeda : ");
    namaSepeda = input.next();
    System.out.print("Masukkan Harga Sepeda : ");
    hargaSepeda = input.nextInt();

    variabelInstanceSepedaInput sepeda = new
        variabelInstanceSepedaInput(namaSepeda,hargaSepeda);
    sepeda.tampilSepeda();
}

```

6. Simpan *class*, kemudian coba jalankan program yang baru saja dibuat dengan menekan *Ctrl+F11*.
  - Pada isian *Nama Sepeda* isi dengan *Gunung (enter)*
  - Pada isian *Harga Sepeda* isi dengan *1700000 (enter)*
7. Lihat hasil dari proses poin 6.

```

Masukkan Nama Sepeda dan Harga Sepeda
-----
Masukka Nama Sepeda : Gunung
Masukkan Harga Sepeda : 1700000

Nama sepeda : Gunung
Harga sepeda : 1700000

```

8. Sekarang jalankan ulang programnya, dan berikan isian sebagai berikut:
  - Pada isian *Nama Sepeda* isi dengan *Sepeda Gunung (enter)*

Perhatikan *error* eksepsi yang dihasilkan:

```

Exception in thread "main" java.util.InputMismatchException
at java.util.Scanner.throwFor(Unknown Source)

```

```
at java.util.Scanner.nextInt(Unknown Source)
at java.util.Scanner.nextInt(Unknown Source)
at java.util.Scanner.nextInt(Unknown Source)
at
Variabel.variabelInstanceSepedaInput.main(variabelInstanceSepedaInput.java:38)
```

Perlu diketahui, bahwa penginputan *string* pada terdapat dua *method*, yaitu:

- *Method next()* : khusus untuk input 1 kata saja
- *Method nextLine()* : digunakan untuk lebih dari satu kata

Jadi untuk mengatasi masalah *error* di atas, cukup dengan mengubah kode program:

```
namaSepeda = input.nextLine() ;
```

menjadi

```
namaSepeda = input.nextLine() ;
```

## 9. Simpan dan siap ujicoba

```
Masukkan Nama Sepeda dan Harga Sepeda
```

```
-----  
Masukkan Nama Sepeda : Sepeda Gunung  
Masukkan Harga Sepeda : 1700000
```

```
Nama sepeda : Sepeda Gunung  
Harga sepeda : 1700000
```

10. Untuk kode program secara penuh sebagai berikut:

```
package variabel;

import java.util.Scanner;

public class variabelInstanceSepedaInput {
    //variabel instance ini dapat digunakan oleh
    //semua subkelas.
    public String namaSepeda;
    public int hargaSepeda;

    //variabel namaSepeda dan hargaSepeda
    //diinisialisasikan dalam konstruktor.
    public variabelInstanceSepedaInput(String NamaSepeda,
    int HargaSepeda)
    {
        namaSepeda = NamaSepeda;
        hargaSepeda = HargaSepeda;
    }

    //Method ini menampilkan informasi sepeda.
    public void tampilSepeda()
    {
        System.out.println("\nData hasil Input");
        System.out.println("-----");
        System.out.println("Nama sepeda : "+namaSepeda);
        System.out.println("Harga sepeda : "+hargaSepeda);
    }

    public static void main(String[] args) {
        String namaSepeda;
        int hargaSepeda;
        Scanner input = new Scanner(System.in);
        System.out.println("Masukkan Nama Sepeda dan Harga
        Sepeda");
        System.out.println("-----");
        System.out.print("Masukkan Nama Sepeda : ");
        namaSepeda = input.nextLine();
        System.out.print("Masukkan Harga Sepeda : ");
        hargaSepeda = input.nextInt();

        variabelInstanceSepedaInput sepeda = new
        variabelInstanceSepedaInput(namaSepeda,hargaSepeda);
        sepeda.tampilSepeda();
    }
}
```

### C. VARIABEL *CLASS/STATIC*

Variabel *static* adalah merupakan jenis variabel yang dideklarasikan menggunakan *keyword static* di dalam *class* dan diluar *method*, *constructor* atau *block*. Biasanya dideklarasikan untuk konstanta data (nilai tetap atau tidak berubah), nilai *default* dari numerik adalah 0, *Boolean* adalah *false*, dan *objek referensi* adalah *null*.

Sintaks variabel *class/static* sebagai berikut:

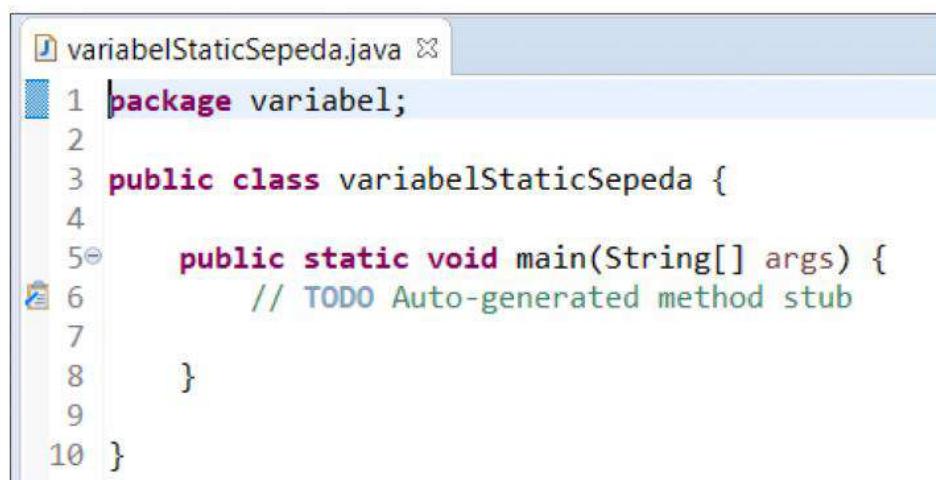
```
[opsional_modifier] static type nama_variabel[;][ = value];
```

Keterangan:

- *Opsional\_modifier* adalah sifat dari variabel
- *type* adalah nama tipe data yang digunakan
- *nama\_variabel* adalah nama variabel yang akan digunakan
- ; adalah memiliki nilai *default* jika langsung menggunakan ;)
- *value* adalah memberikan nilai dari variabel

Untuk praktikumnya, ikuti langkah-langkah praktikum berikut:

1. Buat file *class* baru dengan nama *variabelStaticSepeda.java* pada *folder/package* *variabel* dan sertakan *main method*.



```
variabelStaticSepeda.java
1 package variabel;
2
3 public class variabelStaticSepeda {
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7
8     }
9
10 }
```

Gambar 6.35  
Class *variabelStaticSepeda*

2. Dalam tubuh *class* tambahkan kode program berikut:

```
public static String jenisSepeda;
static int jumlahSepeda;
```

untuk variabel *static jenisSepeda* dengan tipe data *String* diberikan *modifier public* dan untuk variabel *static jumlahSepeda* dengan tipe data *integer modifier*-nya adalah *default*.

3. Dalam tubuh *main method* tambahkan kode program berikut:

```
/*
 * Inisialisasi variabel dan menampilkan
 * jenisSepeda dan jumlahSepeda
 */
jenisSepeda = "Sepeda Gunung";
jumlahSepeda = 4;
System.out.println("Jenis sepeda\t: " + jenisSepeda);
System.out.println("Jumlah sepeda\t: " + jumlahSepeda);
```

4. Kode Program secara lengkap sebagai berikut:

```
package variabel;

public class variabelStaticSepeda {

    // Deklarasi static Variabel
    // modifier public dan default
    public static String jenisSepeda;
    static int jumlahSepeda;

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        /**
         * Inisialisasi variabel dan menampilkan
         * jenisSepeda dan jumlahSepeda
         */
        jenisSepeda = "Sepeda Gunung";
        jumlahSepeda = 4;
```

```

        System.out.println("Jenis sepeda\t: " +
jenisSepeda);
        System.out.println("Jumlah sepeda\t: " +
jumlahSepeda);
    }
}

```

5. Simpan file *class* dan coba jalankan programnya.
6. Hasil dari program sebagai berikut:

```
Jenis sepeda : Sepeda Gunung
Jumlah sepeda : 4
```

Penjelasan:

Di dalam *class variabelStaticSepeda*, dideklarasikan variabel *static*, dengan *modifier publik* dan *default*, proses inisialisasi dan pemanggilan variabel tidak menggunakan objek sama sekali, ini karena penggunaan *keyword static* memberikan penguatan argumentasi bahwa variabel tersebut adalah milik dari *class* itu sendiri.

Untuk memperkaya praktikum ini, kita membuat satu *class* lagi untuk variabel *static* dengan nilai variabel konstanta sebagai berikut:

1. Buatlah *class* dengan nama *variabelStaticSepedaKonstanta* pada folder/package *variabel*
2. Jangan lupa menyertakan (centang) *main method*.
3. Tambahkan dan modifikasi potongan kode program di bawah dalam tubuh *class* akan tetapi di luar *main method*, sebagai berikut:

```

/**
 * variabel hargaSepeda adalah
 * modifier private dan
 * variabel static
 */
private static int hargaSepeda;

// variabel jenisSepeda konstan
public static final String JENISSEPEDA = "Sepeda
Gunung";

```

4. Kemudian dalam tubuh *main method* modifikasi kode program sebagai berikut:

```
hargaSepeda = 12000000;
System.out.println(JENISSEPEDA+" harganya : Rp. " +
hargaSepeda);
```

5. Hingga hasil akhir program sebagai berikut;

```
package variabel;

public class variabelStaticSepedaKonstanta {
    /**
     * variabel hargaSepeda adalah
     * modifier private dan
     * variabel static
     */
    private static int hargaSepeda;

    // variabel jenisSepeda konstan
    public static final String JENISSEPEDA = "Sepeda Gunung";

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        hargaSepeda = 12000000;
        System.out.println(JENISSEPEDA+" harganya : Rp. " +
hargaSepeda);

    }
}
```

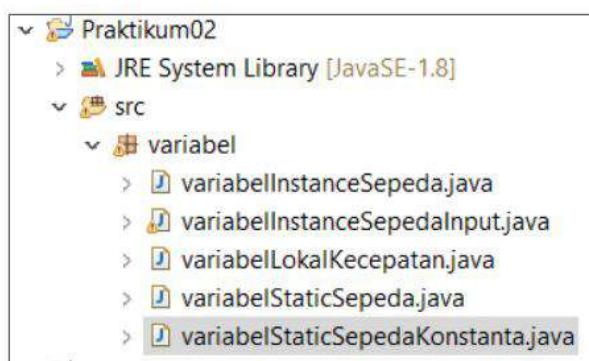
6. Simpan dan jalankan program yang baru saja dibuat, dan hasilnya seperti di bawah:

```
Sepeda Gunung harganya : Rp. 12000000
```

Penjelasan:

Pada baris `public static final String JENISSEPEDA = "Sepeda Gunung";` nilai dari variabel *JENISSEPEDA* diberikan langsung pada saat deklarasi variabel *static*.

Untuk *package variabel* terdapat lima *class* yang telah berhasil kita buat dan praktekkan bersama, untuk memperlancar dalam memahami materi diharapkan , silakan sering mempraktekkan contoh-contoh yang ada.



Gambar 6.36  
*Class variabelStaticSepedaKonstanta*



## LATIHAN

---

Untuk memperdalam pemahaman Anda mengenai materi di atas, kerjakanlah latihan berikut!

- 1) Buatlah program kecil dengan menggunakan variabel lokal untuk menginput data-data berikut:
  - Nama
  - Alamat
  - Email
  - Nomor HP

### *Petunjuk Jawaban Latihan*

- 1) Adapun kode program untuk membuat inputan:
  - Nama
  - Alamat
  - Email

- Nomor HP

Adalah sebagai berikut:

```
import java.util.Scanner;

public class variabelLokalProfileInput {
    public static void main(String[] args) {
        String nama,alamat,email,hp;
        Scanner input = new Scanner(System.in);
        System.out.println("Masukkan Data Anda");
        System.out.println("-----");
        System.out.print("Masukkan Nama : ");
        nama = input.nextLine();
        System.out.print("Masukkan Alamat : ");
        alamat = input.nextLine();
        System.out.print("Masukkan Email : ");
        email = input.nextLine();
        System.out.print("Masukkan No.HP : ");
        hp = input.nextLine();
    }
}
```



## RANGKUMAN

Variabel lokal di deklarasikan dalam *method*, *constructor*, atau blok. Variabel lokal dibuat saat *method*, *constructor* atau blok mulai dijalankan dan akan dihapus saat selesai dijalankan. *Modifier* akses tidak dapat digunakan untuk variabel lokal. Variabel lokal hanya dapat digunakan didalam *method*, *constructor* atau blok tempat pendeklarasinya. Tidak ada nilai default untuk varibel lokal sehingga variabel lokal harus dideklarasikan dan inisialisasi sebelum digunakan.

Variabel *Instance* akan dieksekusi apabila objek dieksekusi. Variabel *Instance* tidak bisa dideklarasikan dengan *keyword static*. Variabel *Instance* dapat digunakan oleh *method*, baik *constructor* atau *method* yang lain. Untuk tipe numerik *default* adalah 0, tipe data *boolean* nilai *default* adalah *false*, dan tipe *reference* adalah *null*.

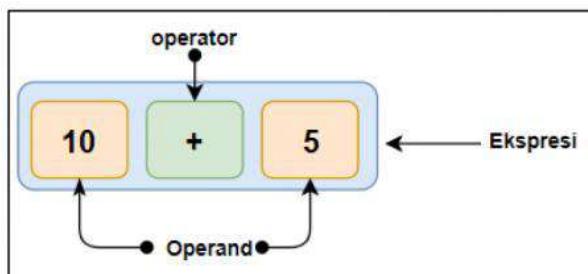
Variabel *static* adalah merupakan jenis variabel yang dideklarasikan menggunakan *keyword static* di dalam *class* dan diluar *method*, *constructor* atau *block*. Biasanya dideklarasikan untuk konstanta data (nilai tetap atau tidak berubah), nilai *default* dari numerik adalah 0, Boolean adalah *false*, dan objek referensi adalah *null*

**PRAKTIKUM 2.3****Operator**

ada Kegiatan Praktikum 2.3 akan dilakukan praktikum untuk mempraktekkan operator-operator pada Java. Pada Java dikenal beberapa istilah seperti ekspresi, operator aritmetika, operator penugasan, operator logika, operator operasional, *unary* dan *ternary*.

**A. OPERATOR****1. Ekspresi**

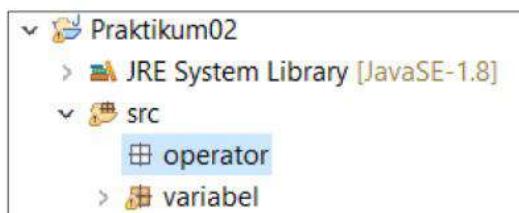
Sebagai pengayaan terhadap teori sebelumnya, di bawah ditampilkan kembali gambar ilustrasi *operator* dan *operand*.



Gambar 6.37  
Ilustrasi Sebuah Operator dengan 2 Operand

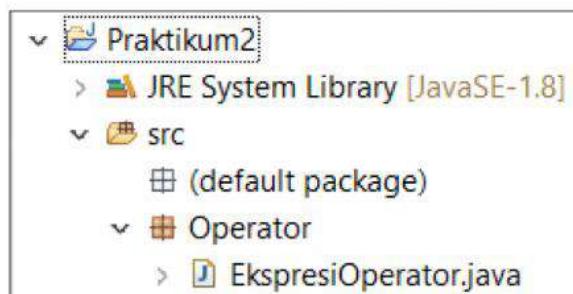
Mari kita mempraktekkan sebuah ekspresi dengan langkah-langkah berikut:

- Buat folder/package dengan nama *operator* pada folder *src*.
- Klik tombol *Finish* untuk membuat folder.



Gambar 6.38  
*Package Operator*

- c. Buat file *class* baru pada *project* Praktikum02 dengan nama *class ekspresiOperator* atau *ekspresiOperator.java* pada *folder operator*, jangan lupa centang opsi *public static void main(String[] args)*.



Gambar 6.39  
Class EkspresiOperator

- d. Pada *main method* ketikkan kode program berikut:

```
int a,b,c;
a = 10;
b = 5;
// a + b adalah ekspresi
// a dan b adalah operand
// tanda + adalah operator
c = a + b;
System.out.println("Hasil 10 + 5 = " + c);
```

- e. Simpan file *class*. Berikut kode program pada *class ekspresiOperator.java* secara utuh:

```
package operator;

public class ekspresiOperator {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int a,b,c;
        a = 10;
        b = 5;
        // a + b adalah ekspresi
        // a dan b adalah operand
        // tanda + adalah operator
```

```

        c = a + b;
        System.out.println("Hasil 10 + 5 = " + c);
    }
}

```

- f. Tekan *F9+F11* untuk menjalankan program.

```
Hasil 10 + 5 = 15
```

## 2. Operator Aritmetika

Jika variabel  $a = 10$  dan variabel  $b = 5$ , jika menggunakan operator aritmetika sebagai berikut:

Operator	Deskripsi	Contoh
$+$ (Penambahan)	Menambahkan nilai di kedua sisi operator	$a + b$ hasilnya 15
$-$ (Pengurangan)	Kurangi <i>operand</i> kanan dari <i>operand</i> kiri	$a - b$ hasilnya 5
$*$ (Perkalian)	Mengalikan nilai di kedua sisi operator	$a * b$ hasilnya 50
$/$ (Pembagian)	Membagi <i>operand</i> kiri dengan <i>operand</i> kanan	$a / b$ hasilnya 2
$\%$ (Modulus)	Membagi <i>operand</i> kiri dengan <i>operand</i> kanan dan memunculkan sisanya	$a \% b$ hasilnya 0

Untuk praktikum operator aritmetika, kita akan membuat kelima operator di atas dalam satu program (*class*) sebagai berikut:

- Buat *class* baru dengan nama *operatorAritmetika.java* pada folder *operator* dan jangan lupa membuat *main method* sekaligus.
- Buka file *class operatorAritmetika.java* kemudian pada *body main method* ketikkan kode program berikut:

```

int a = 10, b = 5, c = 0, d = 15, e = 12, f = 7;
String x = "Algoritma ", y = "dan ", z = "Pemrograman ";

```

```

System.out.println("Operator Aritmetika");
System.out.println("-----");
//operator + dan -
System.out.println("a + b = "+(a + b));
System.out.println("a - b = "+(a - b));

//operator +
//menggabung 3 string dalam 3 variabel.
System.out.println("x + y + z= "+x + y + z);

//operator * dan /
System.out.println("a * b = "+(a * b));
System.out.println("a / b = "+(a / b));

// operator modulo
System.out.println("a % b = "+(a % b));

```

c. Simpan (*Ctrl+S*)

```

package operator;

public class operatorAritmetika {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int a = 10, b = 5, c = 0, d = 15, e = 12, f = 7;
        String x = "Algoritma ", y = "dan ", z = "Pemrograman
        ";

        System.out.println("Operator Aritmetika");
        System.out.println("-----");
        //operator + dan -
        System.out.println("a + b = "+(a + b));
        System.out.println("a - b = "+(a - b));

        //operator +
        //menggabung 3 string dalam 3 variabel.
        System.out.println("x + y + z= "+x + y + z);

        //operator * dan /
        System.out.println("a * b = "+(a * b));
        System.out.println("a / b = "+(a / b));

        // operator modulo
        System.out.println("a % b = "+(a % b));
    }
}

```

- d. Keluaran program sebagai berikut:

```

Operator Aritmatika
-----
a + b = 15
a - b = 5
x + y + z = Algoritma dan Pemrograman
a * b = 50
a / b = 2
a % b = 0

```

### 3. Operator Unary

Jika variabel  $a = -10$  dan variabel  $b = 5$ , jika menggunakan operator aritmetika sebagai berikut:

Operator	Deskripsi	Contoh
$+$ ( <i>Unary plus</i> )	Memberikan nilai positif.	$+a$ hasilnya $-10$ (tetap, karena $+$ (positif) diadu dengan $-$ (negatif) maka hasilnya tetap $-$ (negatif))
$-$ ( <i>Unary minus</i> )	Memberikan nilai negatif.	$-a$ hasilnya $10$ (negatif diadu dengan negatif, menghasilkan positif)
$++$ ( <i>Increment</i> )	Menambahkan nilai dengan 1. Ada dua variasi operator <i>increment</i> , yaitu: <ul style="list-style-type: none"> <li>• <i>Post-Increment</i>: Nilai pertama digunakan dulu dan kemudian ditambah</li> <li>• <i>Pre-Increment</i>: Nilai ditambah dulu baru kemudian digunakan.</li> </ul>	$++a$ hasilnya $-9$
$--$ ( <i>Decrement</i> )	Mengurangi nilai dengan 1. Ada dua variasi operator <i>decrement</i> , yaitu:	$--b$ hasilnya $4$

Operator	Deskripsi	Contoh
	<ul style="list-style-type: none"> <li>• <i>Post-Decrement</i>: Nilai pertama digunakan dulu dan kemudian dikurangi</li> <li>• <i>Pre-Decrement</i>: Nilai dikurangi dulu baru kemudian digunakan.</li> </ul>	
!	Membalikkan nilai Logic	!kondisi hasilnya <i>true</i>

Untuk praktikum ikuti langkah-langkah berikut:

- Buat *class* baru pada folder *operator* dengan nama *operatorUnary* dan jangan lupa membuat *main method*.
- Buka file *class operatorUnary.java*.
- Pada tubuh *main method* ketikkan kode program berikut:

```

int a = 10, b = 5, c = 0, d = 15, e = 12;
boolean kondisi = true;

// operator pre-increment
// a = a+1 dan kemudian c = a;
c = ++a;
System.out.println("Nilai c (++a) = " + c);

// operator post-increment
// c=b kemudian b=b+1
c = b++;
System.out.println("Nilai c (b++) = " + c);

// operator pre-decrement
// d=d-1 kemudian c=d + 1
c = --d + 1;
System.out.println("Nilai c (--d) = " + c);

// operator post-decrement
// c=e kemudian e=e-1
c = --e;
System.out.println("Nilai c (--e) = " + c);

// Membalik kondisi
System.out.println("Nilai !kondisi = " + !kondisi);

```

d. Program secara lengkap:

```

package operator;

public class operatorUnary {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int a = 10, b = 5, c = 0, d = 15, e = 12;
        boolean kondisi = true;

        // operator pre-increment
        // a = a+1 dan kemudian c = a;
        c = ++a;
        System.out.println("Nilai c (++a) = " + c);

        // operator post-increment
        // c=b kemudian b=b+1
        c = b++;
        System.out.println("Nilai c (b++) = " + c);

        // operator pre-decrement
        // d=d-1 kemudian c=d + 1
        c = --d + 1;
        System.out.println("Nilai c (--d) = " + c);

        // operator post-decrement
        // c=e kemudian e=e-1
        c = --e;
        System.out.println("Nilai c (--e) = " + c);

        // Membalik kondisi
        System.out.println("Nilai !kondisi = " + !kondisi);
    }
}

```

e. Hasil program setelah dijalankan:

```

Nilai c (a++) = 11
Nilai c (b++) = 5
Nilai c (--d) = 15
Nilai c (--e) = 11
Nilai !kondisi = false

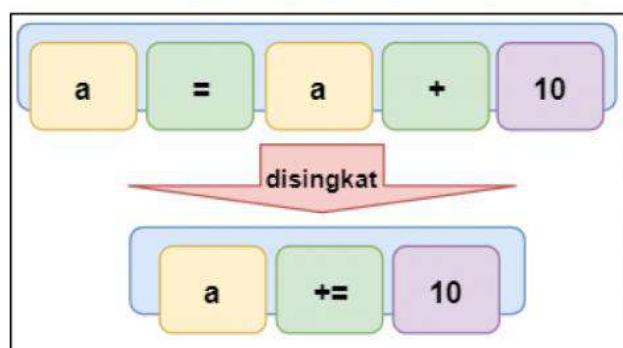
```

Penjelasan:

- `c = ++a;` : variabel a ditambah dulu dengan 1, karena nilai awal variabel `a = 10`, maka hasilnya adalah 11, `c= ++a` sama dengan `c= 1 + a;`
- `c = b++;` : variabel b ditambah 1, nilai awal variabel `b = 5`, maka nilai setelah ditambah 1 adalah 6 (`c=b++` sama dengan `c=b + 1`, bukan `c = 1 + b`);
- `c = --d + 1;` : variabel d dikurangi 1 lalu kemudian ditambah 1 atau `d=d-1` lalu `c = d + 1`, nilai awal `d = 15`, karena ada proses `-d` maka nilai `d = 14`, baru kemudian `c = d + 1` hasilnya adalah 15.
- Begitu juga dengan operasi-operasi yang lain.
- Sedangkan untuk tanda `!` (seru), adalah membalikkan logika, jika nilai awal adalah nilai *true*, maka pada saat nilai tersebut disertakan tanda `!` maka akan bernilai *false*.

#### 4. Operator Penugasan

Operator penugasan digunakan untuk memasukkan nilai ke dalam variabel apa pun. Operator ini dilambangkan dengan “`=`”. Pada lingkungan Java ada beberapa operator penugasan, Pada pemrograman Java sebuah ekspresi operasi dapat dipersingkat, lihat pada contoh gambar di bawah:



Gambar 6.40  
Ilustrasi Operator Normal ke Operator Singkat

Untuk lebih detil melihat operator penugasan, silahkan pelajari/cermati Tabel berikut :

Operator	Deskripsi	Contoh <b>a = 10 dan b = 5</b>
=	Memasukkan nilai pada <i>operand</i> disebelah kanan ke dalam <i>operand</i> disebelah kiri	c = a + b hasilnya 15
+=	Menambahkan <i>operand</i> kiri dengan <i>operand</i> kanan dan kemudian menugaskannya ke variabel di sebelah kiri.	a += 1 hasilnya 11 b += 1 hasilnya 6
-=	Mengurangi <i>operand</i> kiri dengan <i>operand</i> kanan dan kemudian menugaskannya ke variabel di sebelah kiri.	a -= 1 hasilnya 9 b -= 1 hasilnya 4
*=	Mengalikan <i>operand</i> kiri dengan <i>operand</i> kanan dan kemudian menugaskannya ke variabel di sebelah kiri.	a *= 2 hasilnya 20 b *= 2 hasilnya 10
/=	Membagi <i>operand</i> kiri dengan <i>operand</i> kanan dan kemudian menugaskannya ke variabel di sebelah kiri	a /= 2 hasilnya 5
%=	Menetapkan modulus <i>operand</i> kiri dengan <i>operand</i> kanan dan kemudian menugaskannya ke variabel di sebelah kiri.	a %= 2 hasilnya 0
^=	Meningkatkan kekuatan <i>operand</i> kiri ke <i>operand</i> kanan dan menugaskannya ke variabel di sebelah kiri.	a ^= 2 hasilnya 100

Untuk praktikum, mari kita mencoba contoh-contoh yang pernah disebutkan pada Modul 4, langkah-langkah praktikum berikut:

- a. Buat file *class* baru pada folder/*package operator* dengan nama *operatorPenugasan.java*, gabung pada *project Praktikum02* pada IDE Eclipse. Jangan lupa mengikutkan *main method*.

The screenshot shows the Eclipse IDE interface. On the left, the Package Explorer view displays a project structure with several packages and source files. One file, 'operatorPenugasan.java', is selected and shown in the code editor on the right. The code editor contains the following Java code:

```

1 package operator;
2
3 public class operatorPenugasan {
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7
8     }
9
10}
11

```

Gambar 6.41  
*Class operatorPenugasan pada Package Operator*

- b. Kemudian pada *body main method* sisipkan kode program berikut:
- Kode program deklarasi variabel dan memberikan nilai awal.

```

int a = 20;
System.out.println(a+=5);

```

mendeklarasikan variabel a dengan tipe data *integer* dan menginisiasinya dengan nilai 10. Untuk operasi  $a+=5$  artinya sama dengan  $20+5 = 25$ .

- Pada baris berikutnya ketikkan kode program berikut:

```

int b = 10;
System.out.println(b-=5);

```

Mendeklarasikan variabel b dengan *integer* dan menginisiasinya dengan nilai 10. Untuk operasi  $b-=5$  artinya sama dengan  $10-5 = 5$ .

- Baris berikutnya ketikkan kode program berikut:

```
int c = 5;
System.out.println(c*=5);
```

Mendeklarasikan variabel c dengan *integer* dan menginisiasinya dengan nilai 5. Untuk operasi  $c*=5$  artinya sama dengan  $5*5 = 25$ .

- Untuk baris berikutnya:

```
float d = 10;
System.out.println(d/=3);
```

Mendeklarasikan variabel d dengan *float* dan menginisiasinya dengan nilai 10. Untuk operasi  $d/=3$  artinya sama dengan  $10/3 = 3.33333$ .

- Baris berikutnya:

```
int e = 10;
System.out.println(e%3);
```

Mendeklarasikan variabel e dengan *integer* dan menginisiasinya dengan nilai 10. Untuk operasi  $d\%3$  artinya sama dengan  $10 \% 3 = 1$  (sisa hasil bagi).

- c. Untuk kode program lengkapnya sebagai berikut:

```
package operator;

public class operatorPenugasan {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int a = 20;
        System.out.println(a+=5);

        int b = 10;
        System.out.println(b-=5);
    }
}
```

```

int c = 5;
System.out.println(c*=5);

float d = 10;
System.out.println(d/=3);

int e = 10;
System.out.println(e%=3);
}
}

```

- d. Simpan *class* dan kemudian jalankan program, keluaran program sebagai berikut:

```

25
5
25
3.3333333
1

```

Praktikum selanjutnya adalah menggabungkan operasi normal dan operasi singkat, dengan langkah-langkah sebagai berikut:

- Buat *class* baru dengan nama *operatorPenugasan2.java*, letakkan pada folder *operator* yang ada di *project Praktikum02* dan jangan lupa membuat *main method*.
- Pada tubuh *main method* ketikkan kode program berikut:

```

//Deklarasi dan INISIASI nilai awal
int a = 20, b = 10, c, d, e = 10, f = 4;

// contoh penugasan sederhana
c = b;
System.out.println("Nilai c = " + c);

// contoh operasi
// penambahan, pengurangan,
// perkalian, dan pembagian
// dengan cara biasa
a = a + 1;
b = b - 1;
e = e * 2;
f = f / 2;
System.out.println("Nilai dengan operand BIASA");
System.out.println("-----");

```

```

System.out.println("Nilai a = " + a);
System.out.println("Nilai b = " + b);
System.out.println("Nilai e = " + e);
System.out.println("Nilai f = " + f);
System.out.println("");
// mengembalikan ke nilai awal seperti pada
// saat INISIASI nilai awal
a = a - 1;
b = b + 1;
e = e / 2;
f = f * 2;

// menggunakan operator penugasan singkat
// yang hasilnya sama dengan operasi biasa
a += 1;
b -= 1;
e *= 2;
f /= 2;
System.out.println("Nilai dengan operand SINGKAT");
System.out.println("-----");
System.out.println("Nilai a = " + a);
System.out.println("Nilai b = " + b);
System.out.println("Nilai e = " + e);
System.out.println("Nilai f = " + f);
System.out.println("");

// menggunakan operator Modulus
f %= 2;
System.out.println("Modulus");
System.out.println("-----");

System.out.println("Nilai f = " + f);

```

c. Kode program seutuhnya:

```

package operator;

public class operatorPenugasan2 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        // Deklarasi dan INISIASI nilai awal
        int a = 20, b = 10, c, d, e = 10, f = 4;

        // contoh penugasan sederhana
        c = b;
        System.out.println("Nilai c = " + c);
    }
}

```

```
// contoh operasi
// penambahan, pengurangan,
// perkalian, dan pembagian
// dengan cara biasa
a = a + 1;
b = b - 1;
e = e * 2;
f = f / 2;
System.out.println("Nilai dengan operand BIASA");
System.out.println("-----");
System.out.println("Nilai a = " + a);
System.out.println("Nilai b = " + b);
System.out.println("Nilai e = " + e);
System.out.println("Nilai f = " + f);
System.out.println("");
// mengembalikan ke nilai awal seperti pada
// saat INISIASI nilai awal
a = a - 1;
b = b + 1;
e = e / 2;
f = f * 2;

// menggunakan operator penugasan singkat
// yang hasilnya sama dengan operasi biasa
a += 1;
b -= 1;
e *= 2;
f /= 2;
System.out.println("Nilai dengan operand
SINGKAT");
System.out.println("-----");
System.out.println("Nilai a = " + a);
System.out.println("Nilai b = " + b);
System.out.println("Nilai e = " + e);
System.out.println("Nilai f = " + f);
System.out.println("");

// menggunakan operator Modulus
f %= 2;
System.out.println("Modulus");
System.out.println("-----");

System.out.println("Nilai f = " + f);

}
```

d. Hasil dari program sebagai berikut:

```

Nilai c = 10
Nilai dengan operand BIASA
-----
Nilai a = 21
Nilai b = 9
Nilai e = 20
Nilai f = 2

Nilai dengan operand Singkat
-----
Nilai a = 21
Nilai b = 9
Nilai e = 20
Nilai f = 2

Modulus
-----
Nilai f = 0

```

## 5. Operator Relasional

Operator Relasional adalah operator yang digunakan untuk memeriksa hubungan seperti kesetaraan, lebih besar dari pada, kurang dari, nilai baliknya adalah *boolean*.

Operator	Deskripsi	Contoh <b>a = 2 dan b = 3</b>
$==$	Sama dengan: Memeriksa apakah nilai dari dua <i>operand</i> sama atau tidak, jika ya maka kondisi menjadi benar ( <i>true</i> )	a == b hasilnya <i>false</i> (2 == 3)
$!=$	Tidak sama dengan: Memeriksa apakah nilai dari dua <i>operand</i> sama atau tidak, jika nilai tidak sama maka kondisi menjadi benar ( <i>true</i> )	a != b hasilnya <i>true</i> (2 != 3)
$<$	Lebih kecil: memeriksa apakah nilai <i>operand</i> kiri lebih kecil dari nilai <i>operand</i> kanan,	a < b hasilnya <i>true</i> (2 < 3)

Operator	Deskripsi	Contoh $a = 2$ dan $b = 3$
	jika ya maka kondisi menjadi benar ( <i>true</i> )	
$\leq$	Lebih kecil sama dengan: Memeriksa apakah nilai <i>operand</i> kiri kurang dari atau sama dengan nilai <i>operand</i> kanan, jika ya maka kondisi menjadi benar ( <i>true</i> )	$a \leq b$ hasilnya <i>true</i> ( $2 \leq 3$ )
$>$	Lebih besar: Memeriksa apakah nilai <i>operand</i> kiri lebih besar dari nilai <i>operand</i> kanan, jika ya maka kondisi menjadi benar ( <i>true</i> ).	$a > b$ hasilnya <i>false</i> ( $2 > 3$ )
$\geq$	Lebih besar sama dengan: Memeriksa apakah nilai <i>operand</i> kiri lebih besar dari atau sama dengan nilai <i>operand</i> kanan, jika ya maka kondisi menjadi benar ( <i>true</i> )	$a \geq b$ hasilnya <i>false</i> ( $2 \geq 3$ )

Untuk lebih memahami mari kita praktekkan sebagai berikut:

- Buat *class* baru dengan nama *operatorRelasional.java*, letakkan pada folder *operator* yang ada di *project Praktikum2* dan jangan lupa membuat *main method*.
- Pada tubuh *main method* ketikkan kode program berikut:

```

int a = 2, b = 3;
String x = "Relasional", y = "relasional";
int ar[] = { 1, 2, 3 };
int br[] = { 1, 2, 3 };
boolean kondisi = true;

//Penggunaan operator relasional
System.out.println("Penggunaan operator relasional");
System.out.println("-----");
System.out.println("a == a : " + (a == b));
System.out.println("a == b : " + (a == b));

```

```

System.out.println("a < b : " + (a < b));
System.out.println("a <= b : " + (a <= b));
System.out.println("a > b : " + (a > b));
System.out.println("a >= b : " + (a >= b));
System.out.println("a != b : " + (a != b));

// Untuk Array tidak bisa di bandingkan
// menggunakan operator relasional
// nilai akan menghasil false
System.out.println("x == y : " + (ar == br));

System.out.println("kondisi==true :" + (kondisi ==
true));
System.out.println("\nKeterangan: a=2 dan b=3");
System.out.println("x='Relasional' dan y='relasional'");

```

c. Untuk kode program seutuhnya:

```

package operator;

public class operatorRelasional {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int a = 2, b = 3;
        String x = "Relasional", y = "relasional";
        int ar[] = { 1, 2, 3 };
        int br[] = { 1, 2, 3 };
        boolean kondisi = true;

        //Penggunaan operator relasional
        System.out.println("Penggunaan operator
relasional");
        System.out.println("-----");
        System.out.println("a == a : " + (a == b));
        System.out.println("a == b : " + (a == b));
        System.out.println("a < b : " + (a < b));
        System.out.println("a <= b : " + (a <= b));
        System.out.println("a > b : " + (a > b));
        System.out.println("a >= b : " + (a >= b));
        System.out.println("a != b : " + (a != b));

        // Untuk Array tidak bisa di bandingkan
        // menggunakan operator relasional
        // nilai akan menghasil false
        System.out.println("x == y : " + (ar == br));
    }
}

```

```

        System.out.println("kondisi==true :" + (kondisi ==
    true));
        System.out.println("\nKeterangan: a=2 dan b=3");
        System.out.println("x='Relasional' dan
    y='relasional'");
    }
}

```

- d. Simpan, jalankan programnya dan lihat hasilnya:

```

Penggunaan operator relasional
-----
a == a : false
a == b : false
a < b : true
a <= b : true
a > b : false
a >= b : false
a != b : true
x == y : false
kondisi==true :true

Keterangan: a=2 dan b=3
            x='Relasional' dan y='relasional'

```

Penjelasan:

- Pada setiap operasi hanya membandingkan, apakah hubungan keduanya selaras atau tidak.
- Misalnya: Relasional  $a == a$  atau  $2 == 2$  nilainya *true* (secara logika benar)
- Relasional  $a == b$  atau  $2 == 3$  nilainya *false* (karena seharusnya yang bernilai benar adalah  $2 < 3$ )
- Relasional  $a < b$  atau  $2 < 3$  nilainya *true*, karena secara logika itu benar, begitu juga dengan yang lainnya.
- Sedangkan untuk relasional  $x == y$  atau “Relasional” == “relasional” bernilai *false*, membedakan huruf kecil dan huruf besar, walau dalam penyebutan sama bunyinya.

## 6. *Operator Logika*

Mirip dengan Operator Relasional yang membandingkan data, hanya saja Operator Logika adalah operator yang membandingkan bisa lebih dari dua

data. Data yang dibandingkan adalah data *boolean*. Misalnya  $10 > 9$  hasilnya *true*, hasil nilai *true* tersebutlah yang dibandingkan dengan nilai *boolean* yang lain. Untuk Operator Logika digunakan dua operator yaitu logika AND dan logika OR. Perhatikan Tabel di bawah:

Operator	Deskripsi	Contoh
<code>&amp;&amp;</code>	Menghasilkan hubungan logika DAN (AND), lihat poin kebenaran dari logika AND sebagai berikut: 1. <i>true</i> AND <i>true</i> = <i>true</i> 2. <i>false</i> AND <i>false</i> = <i>true</i> 3. <i>true</i> AND <i>false</i> = <i>false</i> 4. <i>false</i> AND <i>true</i> = <i>false</i>	$10 > 9 \&\& 9 > 8$ hasilnya <i>true</i>
<code>  </code>	Menghasilkan hubungan logika ATAU (OR), lihat poin kebenaran dari logika OR sebagai berikut: 1. <i>true</i> OR <i>true</i> = <i>true</i> 2. <i>false</i> OR <i>false</i> = <i>false</i> 3. <i>true</i> OR <i>false</i> = <i>true</i> 4. <i>false</i> OR <i>true</i> = <i>true</i>	$10 < 9 \&\& 9 > 8$ hasilnya <i>true</i>
<code>!</code>	Menghasilkan hubungan logika NEGASI (NOT), lihat poin kebenaran dari logika NOT sebagai berikut: 1. <code>!(false)</code> = <i>true</i> 2. <code>!(true)</code> = <i>false</i>	<code>!(2 &lt; 3)</code> hasilnya <i>false</i>

Sekarang kita praktekkan dengan langkah-langkah praktikum sebagai berikut:

- Buatlah file *class* dengan nama *operatorLogika.java* pada folder *operator* pada *project Praktikum02*, kemudian jangan lupa menyertakan *main method* pada file *class* tersebut.
- Kemudian pada *main method* ketikkan kode-kode program berikut:

```

//Operator logika &&
System.out.println("Operator logika AND (&& ");
System.out.println("-----");
System.out.print("1. true && true : ");
System.out.println(true && true);
System.out.print("2. true && false : ");
System.out.println(true && false);
System.out.print("3. false && false : ");
System.out.println(false && false);

System.out.println("");

//Operator logika ||
System.out.println("Operator logika OR (||) ");
System.out.println("-----");
System.out.print("1. true || true : ");
System.out.println(true || true);
System.out.print("2. true || false : ");
System.out.println(true || false);
System.out.print("3. false || false : ");
System.out.println(false || false);

System.out.println("");

//Operator logika NOT
System.out.println("Operator logika NOT (!) ");
System.out.println("-----");
System.out.print("1. !(true || true) : ");
System.out.println(!(true || true));
System.out.print("2. !(true || false) : ");
System.out.println(!(true || false));
System.out.print("3. !(false || false) : ");
System.out.println(!(false || false));

```

- c. Kode program secara utuh sebagai berikut:

```

package operator;

public class operatorLogika {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        //Operator logika &&
        System.out.println("Operator logika AND (&& ");

```

```

        System.out.println("-----");
        System.out.print("1. true && true : ");
        System.out.println(true && true);
        System.out.print("2. true && false : ");
        System.out.println(true && false);
        System.out.print("3. false && false : ");
        System.out.println(false && false);

        System.out.println("");

        //Operator logika ||
        System.out.println("Operator logika OR (||) ");
        System.out.println("-----");
        System.out.print("1. true || true : ");
        System.out.println(true || true);
        System.out.print("2. true || false : ");
        System.out.println(true || false);
        System.out.print("3. false || false : ");
        System.out.println(false || false);

        System.out.println("");

        //Operator logika NOT
        System.out.println("Operator logika NOT (!) ");
        System.out.println("-----");
        System.out.print("1. !(true || true) : ");
        System.out.println(!(true || true));
        System.out.print("2. !(true || false) : ");
        System.out.println(!(true || false));
        System.out.print("3. !(false || false) : ");
        System.out.println(!(false || false));

    }

}

```

- d. Simpan dan jalankan aplikasi, dan hasilnya seperti di bawah ini:

```

Operator logika AND (&&)
-----
1. true && true : true
2. true && false : false
3. false && false : false

Operator logika OR (||)
-----
1. true || true : true
2. true || false : true

```

```
3. false || false : false
```

Operator logika NOT (!)

- ```
1. !(true || true) : false
2. !(true || false) : false
3. !(false || false) : true
```

Penjelasan:

- Untuk logika AND atau &&
  - ✓ Nomor 1: *true && true* hasilnya boolean *true*
  - ✓ Nomor 2: *true && false* hasilnya boolean *false*
  - ✓ Nomor 3: *false && false* hasilnya boolean *false*
- Untuk logika OR atau ||
  - ✓ Nomor 1: *true || true* hasilnya boolean *true*
  - ✓ Nomor 2: *true || false* hasilnya boolean *true*
  - ✓ Nomor 3: *false || false* hasilnya boolean *false*
- Untuk NOT atau (!)
  - ✓ Nomor 1: *!(true || true)* hasilnya boolean *false*
  - ✓ Nomor 2: *!(true || false!)* hasilnya boolean *false*
  - ✓ Nomor 3: *!(false || false)* hasilnya boolean *true*

Sekarang kita akan membuat praktikum Operator Logika dengan membandingkan multidata (yang dibandingkan lebih dari 2 data *boolean*). langkah-langkahnya sebagai berikut:

- a. Buatlah file *class* dengan nama *operatorLogikaMulti.java* pada folder *operator* pada *project Praktikum02*, kemudian jangan lupa menyertakan *main method* pada file *class* tersebut.
- b. Kemudian pada *main method* ketikkan kode-kode program berikut:

```
int nilaiA = 25;
    int nilaiB = 40;
    float nilaiC = 40.4f;

    boolean a = nilaiA > nilaiB && nilaiB < nilaiC ||
                nilaiC < nilaiB;
    boolean b = nilaiA < nilaiB || nilaiB < nilaiC &&
                nilaiC < nilaiB;

//Jika dijalankan
```

```

System.out.println("Nilai dari a : "+a); //hasilnya:
false
System.out.println("Nilai dari b : "+b); //hasilnya: true

```

- c. Kode program secara utuh sebagai berikut:

```

package operator;

public class operatorLogikaMulti {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        int nilaiA = 25;
        int nilaiB = 40;
        float nilaiC = 40.4f;

        boolean a = nilaiA > nilaiB && nilaiB < nilaiC ||
                    nilaiC < nilaiB;
        boolean b = nilaiA < nilaiB || nilaiB < nilaiC &&
                    nilaiC < nilaiB;

        //Jika dijalankan
        System.out.println("Nilai dari a : "+a);
        //hasilnya: false
        System.out.println("Nilai dari b : "+b);
        //hasilnya: true
    }
}

```

- d. Simpan dan jalankan programnya; hasilnya sebagai berikut:

```

Nilai dari a : false
Nilai dari b : true

```

Penjelasan:

- **boolean a = nilaiA > nilaiB && nilaiB < nilaiC || nilaiC < nilaiB**

jika potongan kode program di atas diberikan langkah penyelesaiannya maka jadinya seperti berikut:

1. *boolean a = 25 lebih besar 40 AND 40 lebih kecil 40.4 OR 40.4 lebih kecil 40*
  2. *boolean a = false AND true OR false*
  3. *boolean a = ( fasle AND true ) OR false*
  4. *boolean a = false OR false*
  5. *boolean a = false*
- **boolean b = nilaiA < nilaiB || nilaiB < nilaiC && nilaiC < nilaiB**

Langkah penyelesaian program sebagai berikut:

1. *boolean b = 25 lebih kecil 40 OR 40 lebih kecil 40.4 AND 40.4 lebih kecil 40*
  2. *boolean b = true OR true AND false*
  3. *boolean b = true OR (true AND false)*
  4. *boolean b = true OR false*
  5. *boolean b = true*
- Yang harus catatan adalah jika dalam sebuah operasi terdapat logika **&&** maka operasinya harus didahulukan, lalu kemudian setelahnya dari kiri ke kanan, contoh pada *boolean b*.

## 7. *Operator Ternary*

Fungsi dari pada Operator *Ternary* adalah alternatif dari fungsi *if else*, dengan menggunakan operator ini adalah kode program terlihat lebih ringkas. Pada praktikum di bawah ini kita akan membandingkan menggunakan kondisi *if else* dengan Operator *Ternary*, sebagai berikut:

- a. Buatlah file *class* dengan nama *operatorTernary.java* pada folder *operator* pada *project Praktikum02*, kemudian jangan lupa menyertakan *main method* pada file *class* tersebut.
- b. Kemudian pada *main method* ketikkan kode-kode program berikut:

```

int nilai1 = 50;
String keterangan;

//Menggunakan if else
if (nilai1 >= 50)
    keterangan = "lulus seleksi!";
else
    keterangan = "tidak lolos seleksi!";

System.out.println("hasil VAR keterangan dari if-else");
System.out.println("-----");
System.out.println(keterangan+'\n');

```

```
int nilai2 = 50;
//Menggunakan ternary
keterangan = (nilai2 >= 50)? "lulus seleksi!" : "tidak lolos
seleksi!";

System.out.println("hasil VAR keterangan dari Operator
Ternary");
System.out.println("-----");
System.out.println(keterangan+'\n');
```

c. Kode program secara lengkap:

```
package operator;

public class operatorTernary {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int nilai1 = 50;
        String keterangan;

        //Menggunakan if else
        if (nilai1 >= 50)
            keterangan = "lulus seleksi!";
        else
            keterangan = "tidak lolos seleksi!";

        System.out.println("hasil VAR keterangan dari if-else");
        System.out.println("-----");
        System.out.println(keterangan+'\n');

        int nilai2 = 50;
        //Menggunakan ternary
        keterangan = (nilai2 >= 50)? "lulus seleksi!" :
   "tidak lolos
seleksi!";

        System.out.println("hasil VAR keterangan dari Operator
Ternary");
        System.out.println("-----");
        System.out.println(keterangan+'\n');

    }
}
```

- d. Simpan dan jalankan program untuk melihat hasilnya:

```
hasil VAR keterangan dari if-else
-----
lolos seleksi!

hasil VAR keterangan dari Operator Ternary
-----
lolos seleksi!
```

Penjelasan:

- Antar *if else* dan Operator *Ternary* hasilnya sama
- Operator *Ternary* jauh lebih ringkas dari *if else*.
- Jumlah baris jika menggunakan *if else* adalah 4 baris, dan jika menggunakan Operator *Ternary* adalah 1 baris.



## LATIHAN

---

Untuk memperdalam pemahaman Anda mengenai materi di atas, kerjakanlah latihan berikut!

- 1) Ketikkan program di bawah ini, dan apa keluaran dari program?

```
package operator.latihan;

public class latihanOperator1 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        System.out.println("Operator Aritmetika");
        int a = 6 + 7;
        int b = a * 5;
        int c = b / 3;
        int d = (c - a) + 2;
        int e = -d - 5;

        System.out.println(" a = " + a);
        System.out.println(" b = " + b);
        System.out.println(" c = " + c);
        System.out.println(" d = " + d);
        System.out.println(" e = " + e);
    }
}
```

```

System.out.println("Operator Boolean");

boolean op_a = !false;
boolean op_b = true;
boolean op_c = op_a | op_b;
boolean op_d = op_a & op_b;
boolean op_e = op_a ^ op_b;
boolean op_f = (!op_a & op_b) | (op_a & !op_b);
boolean op_g = !op_a;

System.out.println(" op_a          = " + op_a);
System.out.println(" op_b          = " + op_b);
System.out.println(" op_a | op_b   = " + op_c);
System.out.println(" op_a&op_b    = " + op_d);
System.out.println(" op_a^op_b    = " + op_e);
System.out.println(" !op_a&b|op_a&!op_b = " + op_f);
System.out.println(" !op_a         = " + op_g);

}

}

```

### Petunjuk Jawaban Latihan

- 1) Hasil dari program pada Nomor 1 adalah:

```

Operator Aritmetika
a = 13
b = 65
c = 21
d = 10
e = -15
Operator Boolean
op_a          = true
op_b          = true
op_a | op_b   = true
op_a&op_b    = true
op_a^op_b    = false
!op_a&op_b|op_a!op_b = false
!op_a         = false

```



## RANGKUMAN

---

Operator dalam pemrograman digunakan untuk melakukan operasi tertentu, macam-macam operator pada Java adalah: Operator Arimatika, Operator Penugasan, Operator Pembanding, Operator Logika, operator *Ternary*, dan Operator *Unary*.

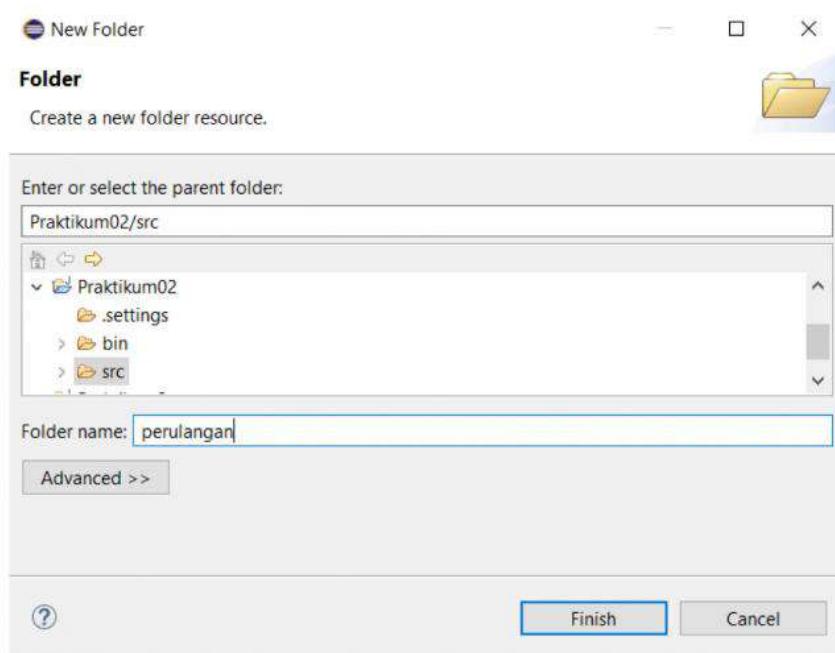
**PRAKTIKUM 2.4****Perulangan dan Kondisi**

• ada kegiatan praktikum 2.4, akan dipraktekkan perulangan dan kondisi, untuk perulangan akan dipraktekkan pernyataan *while*, pernyataan *for*, pernyataan *do..while*, dan untuk kondisi akan dipraktekkan pernyataan *if*, *if..else if*, *if..else if..else*, dan pernyataan *switch case*.

**A. PERULANGAN**

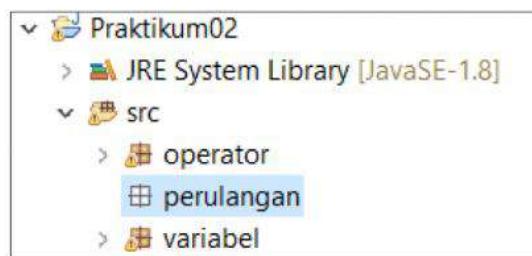
Untuk praktikum Perulangan ini, kita buat folder baru pada *project Praktikum02* pada folder *src* dengan langkah sebagai berikut:

1. Klik kanan folder *src* yang ada pada *project Praktikum02*.
2. Pilih *New → Folder*;



**Gambar 6.41**  
Pembuatan *Package* Perulangan

3. Untuk kolom *Folder name*: isi dengan *perulangan*
4. Klik tombol *Finish* untuk tahap terakhir pembuatan folder.



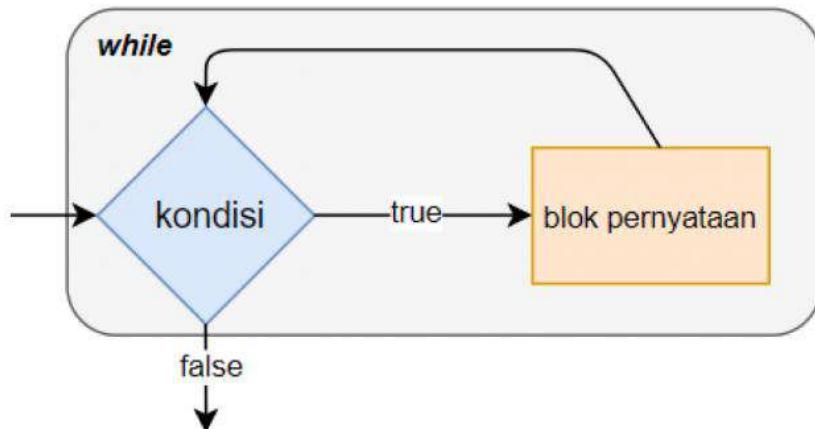
Gambar 6.42  
package perulangan

### 1. Pernyataan *while*

Pernyataan *while* adalah pernyataan dengan aliran kontrol memungkinkan blok program dieksekusi berulang-ulang selama kondisi memenuhi syarat.

Sintaks dan *Flowchart*:

```
while (kondisi boolean)
{
    blok pernyataan
}
```



Gambar 6.43  
*Flowchart Perulangan While*

Praktikum berikut ini adalah mencetak nilai 1 sampai dengan 5, ketika nilai perulangan lebih kecil sama dengan 5 maka perulangan dihentikan. Berikut langkah-langkah praktikum pembuatan programnya:

- a. Buatlah file *class* dengan nama *perulanganWhile.java* pada folder *perulangan* pada *project Praktikum02*, kemudian jangan lupa menyertakan *main method* pada file *class* tersebut.
- b. Kemudian pada *main method* ketikkan kode-kode program berikut:

```
int perulangan = 1;

System.out.println("Pernyataan While");
System.out.println("-----");
//jika kondisi maka blok program
//di eksekusi
while (perulangan <= 5)
{
    System.out.println("nilai perulangan : " +
perulangan);

    // nilai perulangan ditambah 1
    perulangan++;
}
```

- c. Kode program selengkapnya:

```
package perulangan;

public class perulanganWhile {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int perulangan = 1;

        System.out.println("Pernyataan While");
        System.out.println("-----");

        //di bawah ini adalah pernyataan while dengan
        //dengan blok yang dimilik {...}
        //jika kondisi maka blok program
        //di eksekusi
        //ini
        while (perulangan <= 5)
        { //blok awal
            System.out.println("nilai perulangan : " +
perulangan);

            // nilai perulangan ditambah 1
            perulangan++;
        }
    }
}
```

```
    } //blok akhir  
}  
}
```

- d. Hasil dari program di atas:

```
Pernyataan While  
-----  
nilai perulangan : 1  
nilai perulangan : 2  
nilai perulangan : 3  
nilai perulangan : 4  
nilai perulangan : 5
```

Penjelasan:

- **int** perulangan = 1; : Variabel *perulangan* dideklarasikan dengan tipe data *integer* dan diinisiasi dengan nilai 1;
- **while** (perulangan <= 5) : pernyataan *while* langsung melakukan pengecekan syarat, apakah benar nilai variabel perulangan lebih kecil dari 5 atau (*I* <= 5), ternyata benar (*true*), karena nilainya *true*, maka blok program yang ada di bawah pernyataan *while* akan dieksekusi sekali dan mencetak di layar “*nilai perulangan : I*”.
- Di dalam blok *while* terdapat perintah *perulangan++*; yang artinya *perulangan* = *perulangan* + 1; atau setelah melewati kode program tersebut variabel *perulangan* sudah bernilai 2 (*perulangan* = 2).
- Dicek lagi dipernyataan **while** (perulangan <= 5) dengan kondisi (2 <= 5), dan hasilnya *true*. Kemudian dikerjakan lagi blok di bawah pernyataan *while* dan cetakan di layar adalah “*nilai perulangan : 2*”.
- Sampai pada variabel *perulangan* = 6 atau **while** (6 <= 5) akan bernilai *false* dan perulangan dihentikan.

Keterangan:

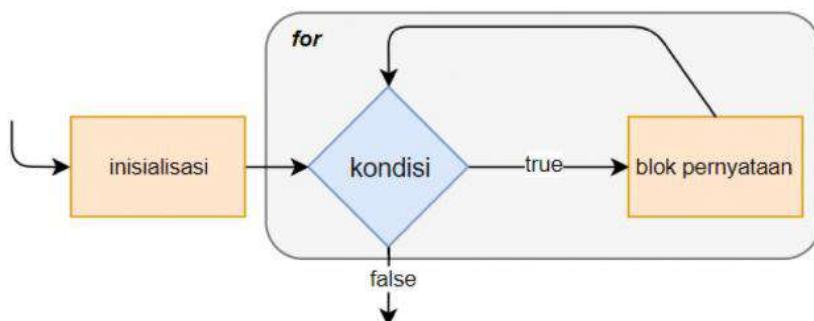
*Perlu diperjelas bahwa untuk perulangan **while**, jika kondisi tidak memenuhi maka tidak ada perulangan yang dilakukan.*

## 2. Pernyataan *for*

Pernyataan *for* menyediakan cara ringkas dalam penulisan struktur perulangan. Berbeda cara dari pernyataan perulangan *while*, pernyataan *for* membutuhkan inisialisasi, kondisi dan penambahan/pengurangan dalam satu baris sehingga menyediakan struktur perulangan yang lebih singkat dan mudah untuk cek kesalahannya.

Sintaks dan Flowchart:

```
for (kondisi inisialisasi; kondisi pengujian;
penambahan/pengurangan)
{
    blok pernyataan
}
```



Gambar 6.44  
Flowchart Perulangan For

Kita akan memberikan contoh dengan mempraktekkan langkah-langkah demi langkah di bawah ini:

- Buatlah file *class* dengan nama *perulanganFor.java* pada folder *perulangan* pada *project Praktikum02*, kemudian jangan lupa menyertakan *main method* pada file *class* tersebut.
- Kemudian pada *main method* ketikkan kode-kode program berikut:

```
System.out.println("Pernyataan For");
System.out.println("-----");
// iterasi for diawali iterasi=2
// dan dijalankan sampai pada
// iterasi <=9
```

```
for (int iterasi = 2; iterasi <= 9; iterasi++)  
{  
    System.out.println("Nilai iterasi :" + iterasi);  
}
```

- c. Kemudian kode lengkap program sebagai berikut:

```
package perulangan;  
  
public class perulanganFor {  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
  
        System.out.println("Pernyataan For");  
        System.out.println("-----");  
  
        /** iterasi for diawali iterasi=2  
         * dan dijalankan sampai pada  
         * iterasi <=9 **/  
        for (int iterasi = 2; iterasi <= 9; iterasi++)  
        {  
            System.out.println("Nilai iterasi :" +  
iterasi);  
        }  
    }  
}
```

- d. Simpan dan jalankan programnya, hasilnya sebagai berikut:

```
Pernyataan For  
-----  
Nilai iterasi :2  
Nilai iterasi :3  
Nilai iterasi :4  
Nilai iterasi :5  
Nilai iterasi :6  
Nilai iterasi :7  
Nilai iterasi :8  
Nilai iterasi :9
```

Penjelasan:

- Pernyataan *for* di awali dengan *for (...)*
- **int iterasi = 2;** : deklarasi variabel *iterasi* dengan menginisialisasi nilai awal variabel *iterasi* = 2.
- **iterasi <= 9;** : Pengujian, selama nilai variabel *iterasi* masih lebih kecil sama dengan 9 atau bernilai *true*, maka akan tetap dilakukan perulangan dan akan berhenti ketika sudah bernilai *false*.
- **iterasi++** : Variabel *iterasi* dalam setiap perulangan terjadi akan ditambah dengan 1 (*iterasi = iterasi + 1*).
- Untuk blok pernyataan adalah pernyataan yang akan dilakukan perulangan.

Mulai Java 5, diperkenalkan cara penyederhanaan sintaks untuk cetak nilai *array*, di bawah ini akan kita praktikkan pencetakan nilai *array* dengan cara lama dan cara baru (Java 5), sebagai berikut:

- Buatlah file *class* dengan nama *perulanganForArray.java* pada folder *perulangan* pada *project Praktikum02*, kemudian jangan lupa menyertakan *main method* pada file *class* tersebut.
- Deklarasikan variabel *array* dengan tipe data *String* dan menginisiasi *array* tersebut dengan beberapa data *string* pada *main method*;

```
String [] Sepeda = {"Gunung", "Santai", "Balap", "Ontel"};
```

- Tetap pada *main method* ketikkan kode-kode program berikut (untuk cetak *array* gaya lama):

```
System.out.println("Jenis Sepeda cetak Array klasik ");
System.out.println("-----");
//Perulangan Array klasik
for (int i = 0; i < Sepeda.length; i++)
{
    System.out.println(Sepeda[i]);
}
```

- Teruskan pada *main method* mengetikkan kode program *array* baru (Java 5) sebagai berikut:

```
        System.out.println("\nJenis Sepeda cetak dengan Array  
Java5");  
        System.out.println("-----  
- ");  
        //Perulangan array yang disederhanakan  
        for(String nama_sepeda : Sepeda ) {  
            System.out.println( nama_sepeda );  
        }
```

- e. Kode program secara utuh di bawah ini:

```
package perulangan;  
  
public class perulanganForArray {  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        String [] Sepeda = {"Gunung", "Santai", "Balap",  
"Ontel"};  
  
        System.out.println("Jenis Sepeda cetak Array klasik  
");  
        System.out.println("-----  
");  
        //Perulangan Array klasik  
        for (int i = 0; i < Sepeda.length; i++)  
        {  
            System.out.println(Sepeda[i]);  
        }  
  
        System.out.println("\nJenis Sepeda cetak dengan  
Array Java5");  
        System.out.println("-----  
----- ");  
        //Perulangan array yang disederhanakan  
        for(String nama_sepeda : Sepeda ) {  
            System.out.println( nama_sepeda );  
        }  
    }  
}
```

f. Simpan program dan coba jalankan, hasilnya:

```
Jenis Sepeda cetak Array klasik
```

```
-----  
Gunung  
Santai  
Balap  
Ontel
```

```
Jenis Sepeda cetak dengan Array Java5
```

```
-----  
Gunung  
Santai  
Balap  
Ontel
```

Penjelasan:

- Perlu dibandingkan adalah potongan program *for* berikut:  
*Klasik*

```
for (int i = 0; i < Sepeda.length; i++)  
{  
    System.out.println(Sepeda[i]);  
}
```

*Penyerderhanaan (Java 5)*

```
for(String nama_sepeda : Sepeda ) {  
    System.out.println( nama_sepeda );  
}
```

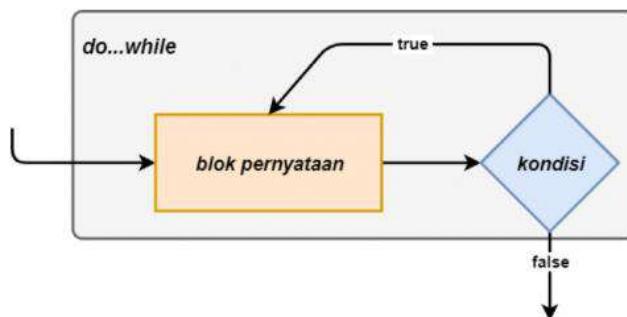
Bisa dilihat dari kedua cara penulisan kedua kode program tersebut di atas, yang satu dengan gaya lama dan sisanya dengan penyederhanaan sintaks namun dari kedua perulangan *for* di atas, menghasilkan keluaran yang sama.

### 3. Pernyataan *do...while*

Fungsi dari pernyataan *do..while* adalah mirip dengan *while*, perbedaan mendasarnya adalah kalau *while* melakukan pengecekan kondisi terlebih dahulu, sedangkan *do..while* minimal mengerjakan *blok pernyataan* sekali lalu kemudian ada pengecekan kondisi.

Sintaks dan *flowchart*:

```
do
{
    blok pernyataan
}
while (kondisi)
```



Gambar 6.45  
Flowchart Perulangan *do..while*

Selama kondisi benar atau *true*, maka *do..while* akan tetap melakukan perulangan, perulangan akan terhenti jika kondisi salah atau *false*. Mari kita praktekkan langkah-langkah berikut:

- Buatlah file *class* dengan nama *perulanganDoWhile.java* pada folder *perulangan* pada *project Praktikum02*, kemudian jangan lupa menyertakan *main method* pada file *class* tersebut.
- Kemudian pada *main method* ketikkan kode-kode program berikut:

```
int angka = 12;
System.out.println("Pernyataan do...while");
System.out.println("-----");
do
```

```
{  
    System.out.println("Nilai angka : " + angka);  
    angka++;  
}  
while (angka < 20);
```

- c. Kode program secara utuh sebagai berikut:

```
package perulangan;  
  
public class perulanganDoWhile {  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        int angka = 12;  
        System.out.println("Pernyataan do...while");  
        System.out.println("-----");  
        do  
        {  
            System.out.println("Nilai angka : " +  
angka);  
            angka++;  
        }  
        while (angka < 20);  
  
    }  
}
```

- d. Simpan dan jalankan programnya, hasilnya sebagai berikut:

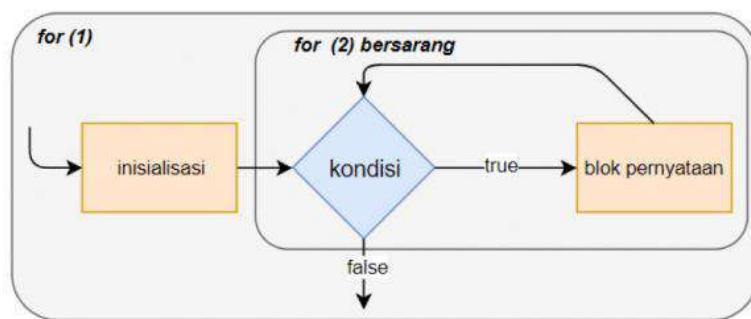
```
Pernyataan do...while  
-----  
Nilai angka : 12  
Nilai angka : 13  
Nilai angka : 14  
Nilai angka : 15  
Nilai angka : 16  
Nilai angka : 17  
Nilai angka : 18  
Nilai angka : 19
```

Penjelasan:

- `int angka = 12;` : deklarasi variabel *angka* sekaligus menginisiasi dengan *angka* = 12.
- `do {...} while ()` : pernyataan *do...while* itu sendiri dan {...} adalah blok program/pernyataan *do...while* itu sendiri.
- Setelah **do** adalah {...} adalah blok yang akan dikerjakan minimal sekali.
- `while (angka < 20);` : *while* dengan kondisi tertentu yang akan di cek, operasi (*angka* < 20) akan dilakukan uji, jika nilainya *true* maka perulangan akan berlanjut ke perulangan berikutnya, dan jika *false* maka perulangan akan dihentikan.

#### 4. Perulangan Bersarang

Perulangan bersarang adalah perulangan yang ada dalam blok pernyataan perulangan, misalnya *for* di dalam *for* atau *for* di dalam *while*.



Gambar 6.46  
Flowchart Perulangan *for* Bersarang

Untuk praktikum berikut ini, akan dibuat *for* dan kemudian dalam program yang sama dibuat *for* dalam *for* sebagai berikut:

- Buatlah file *class* dengan nama *perulanganForDalamFor.java* pada folder *perulangan* pada *project Praktikum02*, kemudian jangan lupa menyertakan *main method* pada file *class* tersebut.
- Kemudian pada *main method* ketikkan kode-kode program berikut untuk perulangan *for* tunggal:

```

// iterasi for diawali iterasi=1
// dan dijalankan sampai pada
// iterasi <=15
System.out.println("For tan for bersarang");

```

```

System.out.println("-----");
for (int iterasi = 1; iterasi <= 10; iterasi++)
{
    System.out.println("*");
}

```

- c. Kemudian tetap dalam blok *main method*, ketikkan kode program berikut untuk *for* di dalam *for* sebagai berikut:

```

System.out.println("\nFor bersarang dalam for");
System.out.println("-----");
for (int iterasi = 1; iterasi <= 15; iterasi++)
{
    for (int iterasi2 = 1; iterasi2 <= iterasi; iterasi2++)
    {
        System.out.print("*");
    }

    System.out.println("");
}

```

- d. Kode program secara lengkap sebagai berikut:

```

package perulangan;

public class perulanganForDalamFor {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        // iterasi for diawali iterasi=1
        // dan dijalankan sampai pada
        // iterasi <=15
        System.out.println("For bersarang");
        System.out.println("-----");
        for (int iterasi = 1; iterasi <= 10; iterasi++)
        {
            System.out.println("*");
        }

        System.out.println("\nFor bersarang dalam for");
    }
}

```

```
System.out.println("-----");
for (int iterasi = 1; iterasi <= 15; iterasi++)
{
    for (int iterasi2 = 1; iterasi2 <= iterasi;
iterasi2++)
    {
        System.out.print("*");
    }
    System.out.println("");
}
}
```

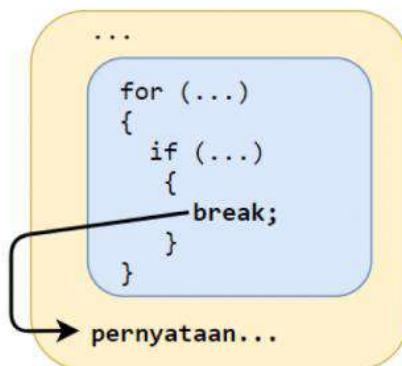
- e. Simpan dan jalankan program, lihat hasilnya sebagai berikut:

Penjelasan:

- Statemen *for* tunggal hanya mencetak \* setiap perulangan *for* sebanyak 10 kali.
- Untuk statemen *for* yang luar akan melakukan perulangan selama 15 kali, sementara untuk blok *for* yang bagian dalam akan terulang sebanyak 15 kali karena *for* yang luar, disamping dia dikerjakan selama 15 kali, *for* bagian dalam juga akan melakukan perulangan selama variabel *iterasi* perulangan *for* pertama.
- Kenapa dia berbentuk tangga, itu disebabkan oleh variabel *iterasi* yang juga mengalami perubahan nilai disetiap perulangan luar.

## 5. Keluar dari Perulangan

Pada pemrograman Java, sebuah perulangan dapat diintervensi pada iterasi yang beroperasi, dalam iterasi perulangan dapat diberhentikan dengan pernyataan *break;*, pernyataan ini dapat digunakan pada 3 pernyataan perulangan.



Gambar 6.47  
Sintaks *break*

Pada praktikum berikut ini adalah kita membuat sebuah inputan didalam sebuah perulangan, jika sebuah kondisi belum memenuhi, maka akan terjadi perulangan dan meminta inputan sampai pengguna memasukkan kondisi yang benar. Adapun langkah-langkah pembuatannya sebagai berikut:

- a. Buatlah file *class* dengan nama *perulanganExit.java* pada folder *perulangan* pada *project Praktikum02*, kemudian jangan lupa menyertakan *main method* pada file *class* tersebut.
- b. Setelah baris *package perulangan;* ketikkan kode program untuk *library I/O* sebagai berikut:

```
import java.util.Scanner;
```

- c. Kemudian pada *main method* ketikkan kode-kode program berikut:

```
//buat objek input
Scanner input = new Scanner(System.in);
//Deklarasi variabel angka
int angka,perulangan=0;
do
{
    perulangan++;
    System.out.print("Perulangan ke "+perulangan+
                     ", Nilai angka [1-9] atau 0 untuk keluar : ");
    angka = input.nextInt();
    if(perulangan==10) break;
}
while (angka > 0);

System.out.println("Anda keluar dari perulangan");
System.out.println("Perulangan selama: "+perulangan+" kali");
```

- d. Kode program secara lengkap:

```
public class perulanganExit {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        //buat objek input
        Scanner input = new Scanner(System.in);
        //Deklarasi variabel angka
        int angka, Perulangan = 0;

        do
        {
            Perulangan++;
            System.out.print("Perulangan ke "+Perulangan+
                            ", Nilai angka [1-9] atau 0 untuk keluar : ");
            angka = input.nextInt();
            if(Perulangan==10) break;
        }
    }
}
```

```

while (angka > 0);

System.out.println("Anda keluar dari perulangan");
System.out.println("Perulangan selama: "+Perulangan+
    " kali");

}
}

```

- e. Simpan *class* dan jalankan program, hasilnya sebagai berikut:

Keluaran di bawah adalah keluaran karena pernyataan *break* (Keluaran Pertama):

```

Perulangan ke 1, Nilai angka [1-9] atau 0 untuk keluar : 12
Perulangan ke 2, Nilai angka [1-9] atau 0 untuk keluar : 2
Perulangan ke 3, Nilai angka [1-9] atau 0 untuk keluar : 3
Perulangan ke 4, Nilai angka [1-9] atau 0 untuk keluar : 4
Perulangan ke 5, Nilai angka [1-9] atau 0 untuk keluar : 6
Perulangan ke 6, Nilai angka [1-9] atau 0 untuk keluar : 6
Perulangan ke 7, Nilai angka [1-9] atau 0 untuk keluar : 6
Perulangan ke 8, Nilai angka [1-9] atau 0 untuk keluar : 8
Perulangan ke 9, Nilai angka [1-9] atau 0 untuk keluar : 9
Perulangan ke 10, Nilai angka [1-9] atau 0 untuk keluar : 2
Anda keluar dari perulangan
Perulangan selama: 10 kali

```

Sementara keluaran di bawah adalah keluaran karena kondisi pada pernyataan *while* memenuhi (Keluaran Kedua):

```

Perulangan ke 1, Nilai angka [1-9] atau 0 untuk keluar : 3
Perulangan ke 2, Nilai angka [1-9] atau 0 untuk keluar : 4
Perulangan ke 3, Nilai angka [1-9] atau 0 untuk keluar : 7
Perulangan ke 4, Nilai angka [1-9] atau 0 untuk keluar : 9
Perulangan ke 5, Nilai angka [1-9] atau 0 untuk keluar : 0
Anda keluar dari perulangan
Perulangan selama: 5 kali

```

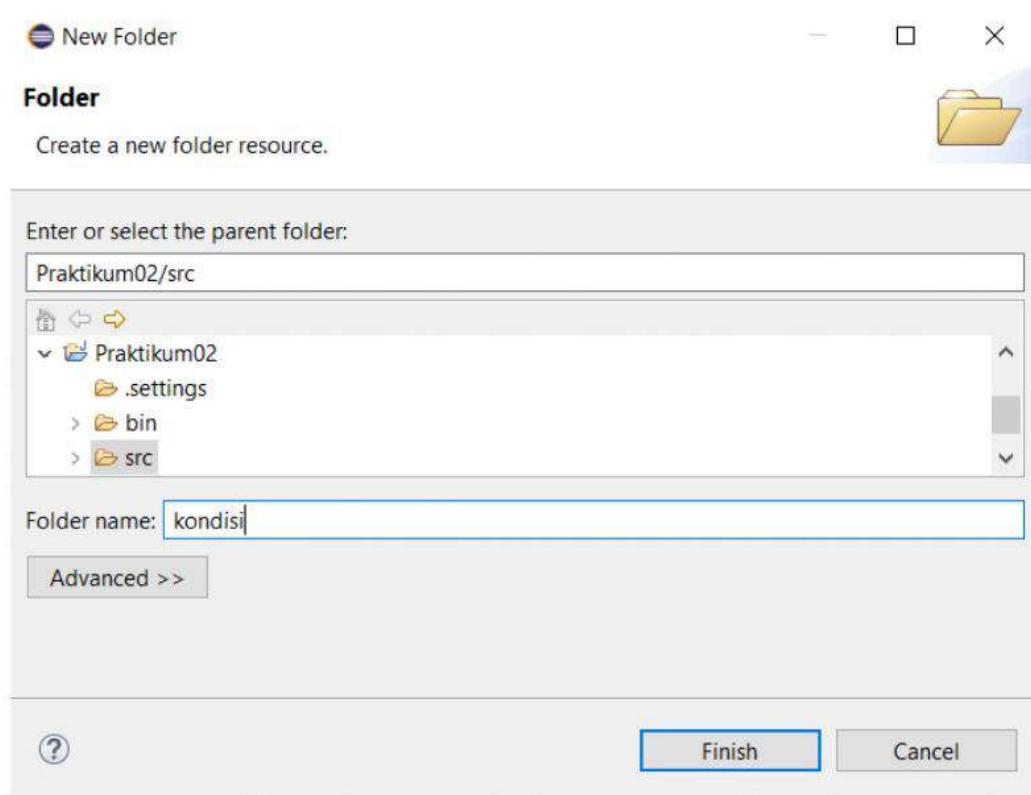
Penjelasan:

- Keluaran pertama, program keluar karena variabel *perulangan* = 10, dan keluar dengan pernyataan *break*. Pernyataan *break* memaksa untuk keluar dari perulangan *do...while*.
- Keluaran kedua, program berhenti karena memang kondisi *do...while* (*angka < 0*) memenuhi kondisi yang ada dengan menginput *angka* = 0.

## B. KONDISI

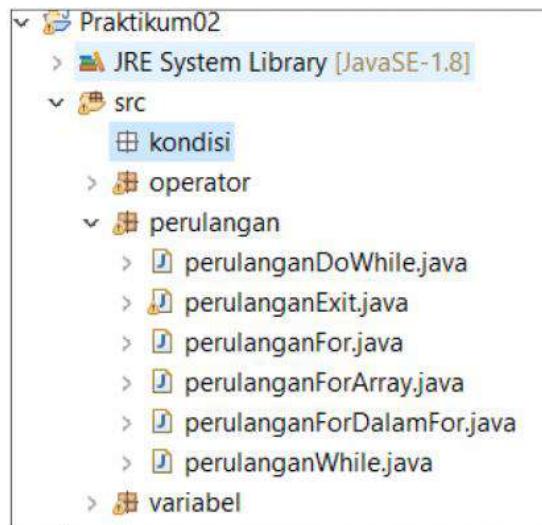
Untuk praktikum kondisi, kita buat folder baru pada *project Praktikum02* pada folder *src* dengan langkah sebagai berikut:

1. Klik kanan folder *src* yang ada pada *project Praktikum2*.
2. Pilih *New → Folder*;



Gambar 6.48.  
Pembuatan Pacakge Kondisi

3. Untuk kolom *Folder name*: isi dengan *kondisi*
4. Klik *Finish* untuk tahap terakhir pembuatan folder.



Gambar 6.49  
*Pacakge Kondisi*

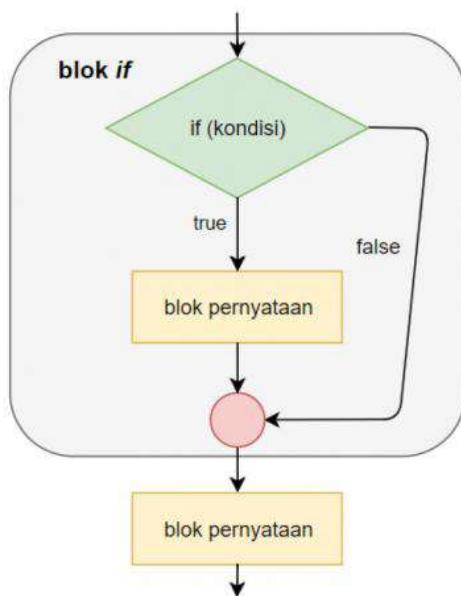
### 1. Pernyataan *if*

Pernyataan *if* adalah pernyataan pengambilan keputusan yang paling sederhana.

Sintaks dan *Flowchart*:

```
if (kondisi)
{
    // Pernyataan untuk dieksekusi jika
    // kondisi benar
}
```

*Flowchart:*



Gambar 6.50  
Flowchart Kondisi if

Setelah pernyataan *if*, ada *kondisi* yang nilainya berupa *boolean*, nilainya akan diperiksa atau dievaluasi nilainya benar (*true*) atau tidak (*false*), jika nilainya benar maka dia akan mengerjakan blok pernyataan di bawahnya.

Untuk praktik kondisi *if*, ikuti langkah-langkah berikut ini:

- Buatlah file *class* dengan nama *kondisiIf.java* pada folder *kondisi* pada *project Praktikum02*, kemudian jangan lupa menyertakan *main method* pada file *class* tersebut.
- Ketikkan program berikut pada *main method*, sebagai berikut:

```

int i = 30;

if (i > 15)
{
    System.out.println("30 > 15 adalah benar");
}
System.out.println("ini sudah diluar if");
  
```

- c. Kode program secara utuh sebagai berikut:

```
package kondisi;

public class pernyataanIf {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int i = 30;

        if (i > 15)
        {
            System.out.println("30 > 15 adalah benar");
        }
        System.out.println("ini sudah diluar if");

    }

}
```

- d. Simpan dan jalankan program, hasilnya sebagai berikut:

```
30 > 15 adalah benar
Ini sudah diluar if
```

Penjelasan:

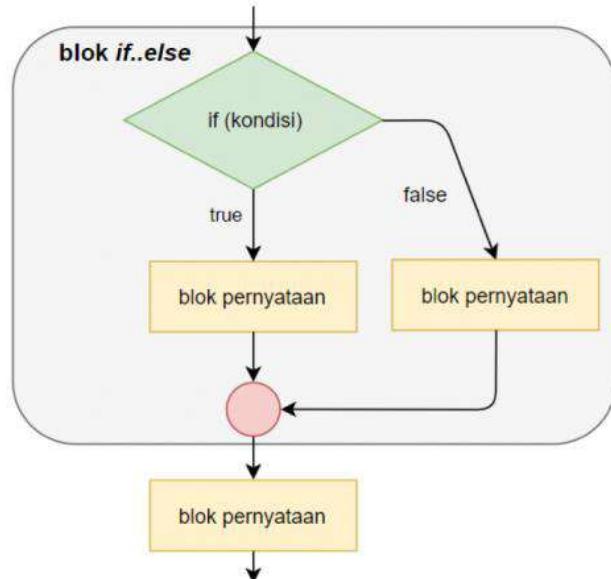
Pada baris `if (i > 15)`, nilai variabel `i` akan diperiksa, karena `nilai = 30` maka dibandingkan dengan operator `>`, operasi `30 > 15` memberikan nilai `true` (bernilai benar secara logika), karena bernilai benar maka blok program di bawahnya (`System.out.println("30 > 15 adalah benar");`) akan dikerjakan.

## 2. Pernyataan `if..else`

Sintaks dan simbol *flowchart*:

```
if (kondisi)
{
    // Pernyataan untuk dieksekusi jika
    // kondisi benar
}
else
{
```

```
// Pernyataan untuk dieksekusi jika
// kondisi salah
}
```



Gambar 6.51  
Flowchart Kondisi *if..else*

Ikuti langkah-langkah berikut untuk mempraktekkan pernyataan *if...else*, sebagai berikut:

- Buatlah file *class* dengan nama *kondisiIfElse.java* pada folder *kondisi* pada *project Praktikum02*, kemudian jangan lupa menyertakan *main method* pada file *class* tersebut.
- Ketikkan kode program berikut pada *main method*, sebagai berikut:

```

int i = 30;

if (i > 40)
{
    System.out.println(i > 40);
}
else
{
    System.out.println(i > 40);
}

System.out.println("ini sudah diluar if..else");
  
```

- c. Kode program secara utuh sebagai berikut:

```
package kondisi;

public class kondisiIfElse {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int i = 30;

        if (i > 40)
        {
            System.out.println(i > 40);
        }
        else
        {
            System.out.println(i > 40);
        }

        System.out.println("ini sudah diluar if..else");

    }
}
```

- d. Simpan dan jalankan program, hasilnya sebagai berikut:

```
False
Ini sudah diluar if
```

Penjelasan:

Setelah pernyataan *if*, ada *kondisi* ( $i > 40$ ) yang nilainya berupa *boolean*, nilainya akan diperiksa atau dievaluasi nilainya apakah benar (*true*) atau tidak (*false*)?, jika nilainya benar maka dia akan mengerjakan blok pernyataan di bawahnya, dan jika salah, maka blok pernyataan yang ada di bawah *else* yang dikerjakan.

Sekarang kita coba praktekkan dengan nilai input dengan ketentuan sebagai berikut:

1. Angka yang di input antara 1 sampai dengan 100;
2. Angka 1 – 50 dianggap tidak lulus;

3. Angka 50 ke atas hingga 100 dianggap lulus.

Sekarang kita praktikkan algoritma sederhana di atas dengan langkah-langkah berikut:

1. Buatlah file *class* dengan nama *kondisiIfElseInput.java* pada folder *kondisi* pada *project Praktikum02*, kemudian jangan lupa menyertakan *main method* pada file *class* tersebut.
2. Setelah baris **package Kondisi;** ketikkan kode program berikut:

```
import java.util.Scanner;
```

3. Ketikkan kode program berikut pada *main method*, sebagai berikut:

```
//membuat objek input
Scanner input = new Scanner(System.in);
int Angka = 0;
System.out.print("Masukkan angka 1-100 : ");

//input nilai kedalam variabel Angka
Angka = input.nextInt();

//kondisi if..else..
if (Angka > 1 && Angka <= 50)
{
    System.out.println("Anda tidak lulus");
}
else
{
    System.out.println("Anda lulus");
}
```

4. Kode program secara utuh sebagai berikut:

```
package kondisi;

import java.util.Scanner;

public class kondisiIfElseInput {
```

```
public static void main(String[] args) {
    // TODO Auto-generated method stub

    //membuat objek input
    Scanner input = new Scanner(System.in);
    int Angka = 0;
    System.out.print("Masukkan angka 1-100 : ");
    //input nilai kedalam variabel Angka
    Angka = input.nextInt();
    //kondisi if..else..
    if (Angka > 1 && Angka <= 50)
    {
        System.out.println("Anda tidak lulus");
    }
    else
    {
        System.out.println("Anda lulus");
    }
}
```

5. Simpan dan jalankan programnya, hasilnya:

*Pengujian 1:*

Input Angka = 30

```
Masukkan angka 1-100 : 30
Anda tidak lulus
```

*Pengujian 2:*

Input Angka = 60

```
Masukkan angka 1-100 : 60
Anda lulus
```

Penjelasan:

Selama Anda menginput angka (numerik) maka program ini tidak akan mengalami masalah, akan tetapi jika Anda menginput lebih kecil 0 atau

lebih besar dari 100, maka yang akan di eksekusi adalah bagian blok *else*. Anda bisa mencobanya dengan memasukkan nilai -1.

```
Masukka angak 1-100 : -1  
Anda lulus
```

Untuk mengatasi hal di atas, tidak ada jalan lain kecuali mengatasinya dengan pernyataan *if..else if..else*.

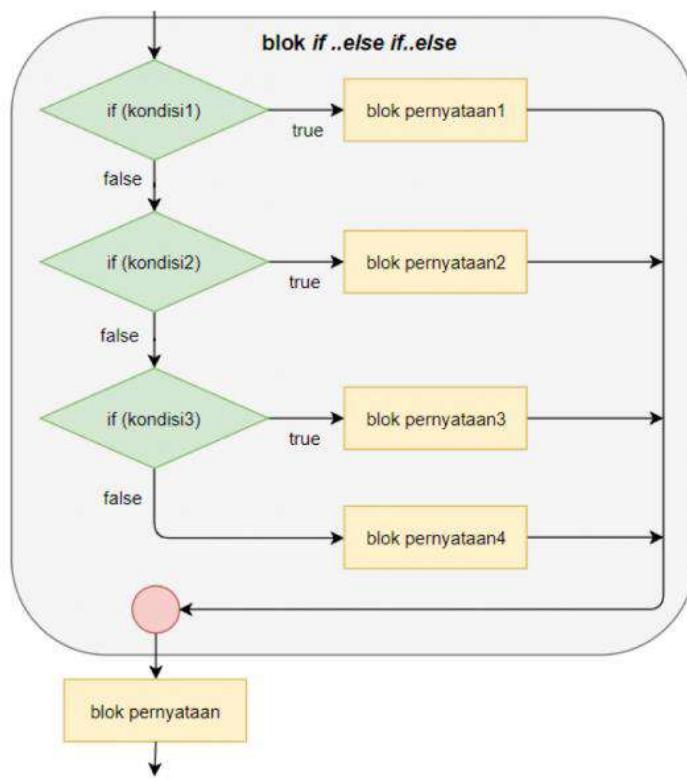
### 3. Pernyataan *if..else if..else*

Pernyataan *if..else if..else* adalah pernyataan yang digunakan untuk multi kondisi. Jika terdapat lebih dari 2 kondisi, maka pernyataan lebih cocok untuk digunakan.

Sintaks dan simbol *flowchart*:

```
if (kondisi1)
{
    // Pernyataan untuk dieksekusi jika
    // kondisi benar
}
else if (kondisi2)
{
    // Seleksi kedua setelah
    // kondisi1 tidak memenuhi
}
Else
{
    //kondisi salah
}
```

*Flowchart:*



Gambar 6.52.  
Flowchart Kondisi *if..else if..else*

Sekarang mari kita gunakan pernyataan *if..else if..else* pada praktikum berikut ini:

- Buatlah file *class* dengan nama *kondisiIfElseIfElse.java* pada folder *kondisi* pada *project Praktikum02*, kemudian jangan lupa menyertakan *main method* pada file *class* tersebut.
- Ketikkan kode program berikut pada *main method*, sebagai berikut:

```

int x = 20;

if (x == 10)
    System.out.println("x adalah 10");
else if (x == 15)
    System.out.println("x adalah 15");
else if (x == 20)
    System.out.println("x adalah 20");
else
    System.out.println("x tidak 20");
    
```

- c. Kode program utuh sebagai berikut:

```

package kondisi;

public class kondisiIfElseifElse {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int x = 20;

        if (x == 10)
            System.out.println("x adalah 10");
        else if (x == 15)
            System.out.println("x adalah 15");
        else if (x == 20)
            System.out.println("x adalah 20");
        else
            System.out.println("x tidak 20");

    }

}

```

- d. Simpan dan jalankan program, lihat hasilnya:

```
x adalah 20
```

#### 4. Pernyataan *if bersarang*

Pernyataan *if* yang bersarang adalah *if* yang terdapat dalam blok pernyataan. Pada pemrograman Java memungkinkan hal ini bisa digunakan.

Sintaks dan simbol *Flowchart*:

```

if (kondisi)
{
    If (kondisi) //bersarang
    {
        ...
    }
    Else
    {
}

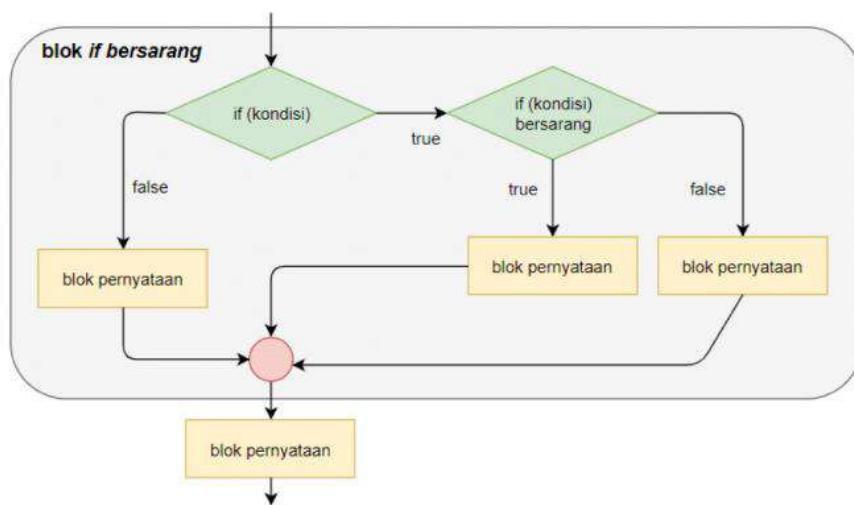
```

```

    }
}
else
{
    // Pernyataan untuk dieksekusi jika
    // kondisi salah
}

```

*Flowchart:*



Gambar 6.53  
Flowchart Kondisi *if* Bersarang

Mari kita praktekkan untuk kondisi *if* bersarang dalam sebuah *if*, ikuti langkah-langkah berikut:

- Buatlah file *class* dengan nama *kondisiIfbersarang.java* pada folder *kondisi* pada *project Praktikum02*, kemudian jangan lupa menyertakan *main method* pada file *class* tersebut.
- Setelah baris **package** *Kondisi*; ketikkan kode program berikut:

```
import java.util.Scanner;
```

- c. Ketikkan kode program berikut pada *main method*, sebagai berikut:

```
//buat objek input
Scanner input = new Scanner(System.in);
int x;
System.out.println("If Bersarang dalam If");
System.out.println("-----");
System.out.print("Masukkan nilai x : ");
x = input.nextInt();

if (x >= 13)
{
    // if pertama yang bersarang
    if (x == 13)
        System.out.println("variabel x = 13");

    //if kedua yang bersarang
    if (x <= 14)
        System.out.println("variabel x <= 14");
    else
        System.out.println("variabel x > 14");
}
else
{
    System.out.println("Nilai x < 13");
}
```

- d. Kode program secara keseluruhan:

```
package kondisi;

import java.util.Scanner;

public class kondisiIfbersarang {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        //buat objek input
        Scanner input = new Scanner(System.in);
        int x;
        System.out.println("If Bersarang dalam If");
        System.out.println("-----");
        System.out.print("Masukkan nilai x : ");
        x = input.nextInt();
```

```

if (x >= 13)
{
    // if pertama yang bersarang
    if (x == 13)
        System.out.println("variabel x = 13");

    //if kedua yang bersarang
    if (x <= 14)
        System.out.println("variabel x <= 14");
    else
        System.out.println("variabel x > 14");
}
else
{
    System.out.println("Nilai x < 13");
}

}

```

- e. Simpan program  
f. Jalankan program, Uji coba 1: masukkan x = 12, hasilnya:

```

If Bersarang dalam If
-----
Masukkan nilai x : 12
Nilai x < 13

```

Ketika memasukkan nilai x = 12, maka program yang dijalankan adalah:

```

else
{
    System.out.println("Nilai x < 13");
}

```

- g. Jalankan program, Uji coba 2: masukkan  $x = 13$ , hasilnya:

```
If Bersarang dalam If
-----
Masukkan nilai x : 13
Nilai x = 13
Variabel x <= 14
```

Ketika memasukkan nilai  $x = 13$ , maka program yang dijalankan adalah:

```
if (x == 13)
    System.out.println("variabel x = 13");
```

- h. Jalankan program, Uji coba 3: masukkan  $x = 14$ , hasilnya:

```
If Bersarang dalam If
-----
Masukkan nilai x : 14
Nilai x = 14
Variabel x <= 14
```

Ketika memasukkan nilai  $x = 14$ , maka program yang dijalankan adalah:

```
if (x <= 14)
    System.out.println("variabel x <= 14");
```

- i. Jalankan program, Uji coba 3: masukkan  $x = 15$ , hasilnya:

```
If Bersarang dalam If
-----
Masukkan nilai x : 15
Variabel x > 14
```

Ketika memasukkan nilai  $x = 15$ , maka program yang dijalankan adalah:

```
else  
    System.out.println("variabel x > 14");
```

## 5. Pernyataan switch case

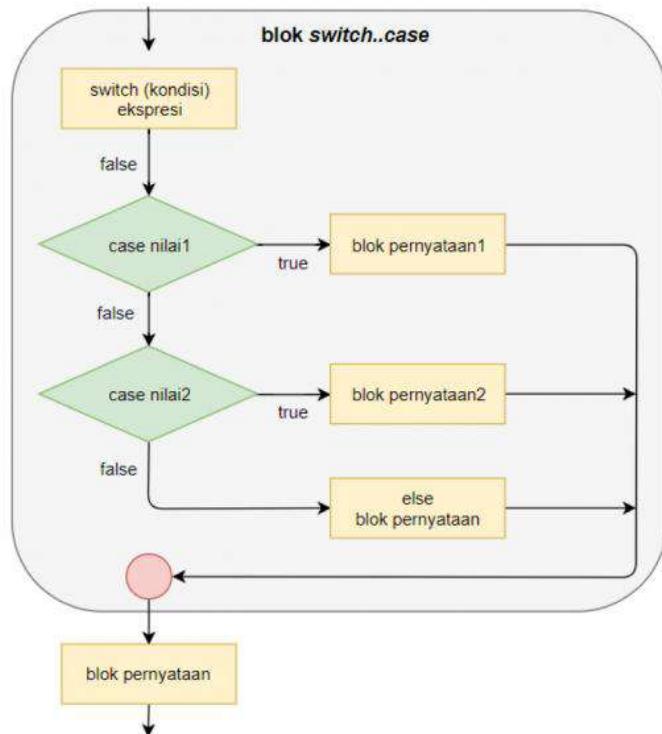
Pernyataan *switch...case* adalah pernyataan cabang dengan multi-arah. Pernyataan ini menyediakan cara mudah untuk mengirimkan eksekusi ke berbagai blok pernyataan berdasarkan nilai ekspresi.

Untuk ekspresi dapat berupa tipe *byte*, *short*, *int*, *char* atau *enumeration*. Dan dimulai dengan JDK7, ekspresi juga bisa dari tipe *String*. Nilai tidak boleh sama pada dua atau lebih *case*. Pernyataan *default* bersifat opsional. Pernyataan *break* digunakan di dalam saklar untuk mengakhiri rangkaian pernyataan. Pernyataan *break* bersifat opsional. Jika dihilangkan, eksekusi akan berlanjut ke *case* berikutnya.

Sintaks dan simbol *Flowchart*:

```
switch (kondisi)  
{  
    case value1:  
        pernyataan1;  
        break;  
    case nilai2:  
        pernyataan2;  
        break;  
    .  
    .  
    case nilaiN:  
        pernyataanN;  
        break;  
    default:  
        statementDefault;  
}
```

*Flowchart:*



Gambar 6.54  
*Flowchart Kondisi switch..case*

Untuk latihan praktikum kali ini kita akan membuat praktikum dengan memadukan antara *do..while* dengan *switch..case*, dengan langkah-langkah berikut:

- Buatlah file *class* dengan nama *kondisiSwitchCase.java* pada folder *kondisi* pada *project Praktikum02*, kemudian jangan lupa menyertakan *main method* pada file *class* tersebut.
- Setelah baris **package** *Kondisi*; ketikkan kode program berikut:

```
import java.util.Scanner;
```

- Tambahkan kode program berikut pada blok *main method*:

```
Scanner input = new Scanner(System.in);
int angka;
```

```
do
{
    System.out.print("Masukkan angka [1-9] : ");
    angka = input.nextInt();
    switch (angka)
    {
        case 1:
            System.out.println("Angka Satu");
            break;
        case 2:
            System.out.println("Angka Dua");
            break;
        case 3:
            System.out.println("Angka Tiga");
            break;
        case 4:
            System.out.println("Angka Empat");
            break;
        case 5:
            System.out.println("Angka Lima");
            break;
        case 6:
            System.out.println("Angka Enam");
            break;
        case 7:
            System.out.println("Angka Tujuh");
            break;
        case 8:
            System.out.println("Angka Delapan");
            break;
        case 9:
            System.out.println("Angka Sembilan");
            break;
        default:
            System.out.println("Anda memilih keluar");
    }
}
while (angka > 0);
```

- d. Kode program selengkapnya:

```
package kondisi;

import java.util.Scanner;
```

```
public class kondisiSwitchCase {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner input = new Scanner(System.in);
        int angka;

        do
        {
            System.out.print("Masukkan angka [1-9] : ");
            angka = input.nextInt();
            switch (angka)
            {
                case 1:
                    System.out.println("Angka Satu");
                    break;
                case 2:
                    System.out.println("Angka Dua");
                    break;
                case 3:
                    System.out.println("Angka Tiga");
                    break;
                case 4:
                    System.out.println("Angka Empat");
                    break;
                case 5:
                    System.out.println("Angka Lima");
                    break;
                case 6:
                    System.out.println("Angka Enam");
                    break;
                case 7:
                    System.out.println("Angka Tujuh");
                    break;
                case 8:
                    System.out.println("Angka Delapan");
                    break;
                case 9:
                    System.out.println("Angka Sembilan");
                    break;
                default:
                    System.out.println("Anda memilih keluar");
            }
        }
        while (angka > 0);
    }
}
```

- e. Simpan dan jalankan program.

```
Masukkan angka [1-9] : 2
Angka Dua
Masukkan angka [1-9] : 4
Angka Empat
Masukkan angka [1-9] : 6
Angka Enam
Masukkan angka [1-9] : 8
Angka Delapan
Masukkan angka [1-9] : 9
Angka Sembilan
Masukkan angka [1-9] : 1
Angka Satu
Masukkan angka [1-9] : 5
Angka Lima
Masukkan angka [1-9] : 0
Anda memilih keluar
```

Penjelasan:

- Jika Anda menginput angka = 0, maka program akan berhenti, akan tetapi selama Anda menginput angka antara 1 sampai dengan 9, maka dia akan meminta input terus dengan perulangan *do..while*.
- Setiap Anda menginput angka antara 1 – 9, setiap angka yang Anda input akan direspon oleh *switch..case*. Misal Anda menginput angka 1, maka *switch case* akan merespon dengan keluaran “*Angka Satu*”.

**LATIHAN**

Untuk memperdalam pemahaman Anda mengenai materi di atas, kerjakanlah latihan berikut!

- 1) Buatlah program untuk menampilkan sebagai berikut:

|    |                                                                                                                                           |
|----|-------------------------------------------------------------------------------------------------------------------------------------------|
| a. | 1<br>1 2<br>1 2 3<br>1 2 3 4<br>1 2 3 4 5                                                                                                 |
| b. | 1<br>1 3<br>1 3 5<br>1 3 5 7<br>1 3 5 7 9                                                                                                 |
| c. | 1 2 3 4 5 6 7 8 9 10<br>1 2 3 4 5 6 7 8 9<br>1 2 3 4 5 6 7 8<br>1 2 3 4 5 6 7<br>1 2 3 4 5 6<br>1 2 3 4 5<br>1 2 3 4<br>1 2 3<br>1 2<br>1 |

*Petunjuk Jawaban Latihan*

- 1) Jawaban:
- a. Untuk menampilkan:
- 1  
1 2  
1 2 3  
1 2 3 4  
1 2 3 4 5

diperlukan perulangan dalam perulangan, atau *for* dalam *for*, dan tidak harus *for* dalam *for*, juga bisa diselesaikan *for* dalam *while* atau *do..while*. misalnya saja kita gunakan *for* dalam *for* dengan kode program sebagai berikut:

```
public class printAngkaSampai5 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        for (int i = 1; i <= 5; i++)
        {
            for (int j = 1; j <= i; j++)
            {
                System.out.print(j);
            }
            System.out.println("");
        }
    }
}
```

atau menggunakan *for* dalam *while* sebagai berikut:

```
public class printAngkaSampai5 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int m = 1;
        while (m <= 5)
        {
            for (int n = 1; n <= m; n++)
            {
                System.out.print(n);
            }
            System.out.println("");

            m++;
        }
    }
}
```

- b. Untuk menampilkan:

```
1  
1 3  
1 3 5  
1 3 5 7  
1 3 5 7 9
```

Hampir sama dengan bagian a, juga menggunakan perulangan, hanya saja menampilkan bilangan ganjil saja. Maka sebelum ditampilkan perlu ada kondisi *if..else* dan operator modulus. Berikut penyelesaiannya:

```
public class tampilanAngkaGanjil1ke10 {  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        for (int i = 1; i <= 10; i++)  
        {  
            if(i % 2 == 1)  
            {  
                for (int j = 1; j <= i; j++)  
                {  
                    if(j % 2 == 1)  
                    {  
                        System.out.print(j);  
                    }  
                }  
                System.out.println("");  
            }  
        }  
    }  
}
```

- c. Sedangkan untuk menampilkan:

```
1 2 3 4 5 6 7 8 9 10  
1 2 3 4 5 6 7 8 9  
1 2 3 4 5 6 7 8
```

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

Berikut kode programnya:

```
public class test {  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        for (int i = 10; i >= 1; i--)  
        {  
            for(int j=1; j <=i; j++)  
            {  
  
                System.out.print(j+" ");  
            }  
  
            System.out.println("");  
        }  
        System.out.println("\n\n");  
    }  
}
```



## RANGKUMAN

---

Perulangan pada Java dikenal dengan pernyataan *while*, *for* dan *do..while*. dan masing-masing perulangan mempunyai karakter masing-masing.

Untuk kondisi, Java memberikan beberapa pilihan untuk digunakan, yaitu *if*, *if..else*, *if..else if*, *if..else if..else*, dan *switch..case*.

# Array, Method, dan Eksepsi

Kani, M.Kom.



## PENDAHULUAN

---

Pada Bahasa Pemrograman Java terdapat ribuan fungsi/*method* yang disimpan kedalam *library-library* Java, terkadang kita mengetikkan kode program namun kita tidak sadar bahwa itu adalah *method* yang dibangun oleh pengembang Java. Begitu banyak *method* dibangun untuk memanjakan *programmer* dalam menyelesaikan masalah. Pada modul ini akan dibahas masalah tersebut dengan mengkolaborasi dengan tiga sub-pokok bahasan yaitu *array*, *method* dan eksepsi.

Setelah mempelajari modul ini, diharapkan Mahasiswa dapat:

1. menerangkan pengertian *Array*;
2. membedakan dan mengenali antar *Array* satu dimensi dan *array* multidimensi;
3. melatih contoh-contoh program kecil tentang *Array*;
4. menerangkan pengertian dan menunjukkan cara membuat *method*;
5. menerangkan *method* dengan sebuah *return*;
6. menerangkan *method* dengan menggunakan *void*;
7. melatih contoh-contoh program kecil menyangkut pembuatan *method* dan memanggil *method*;
8. menerangkan pengertian eksepsi, *error* dan *bugs*;
9. menerangkan bagaimana cara penanganan eksepsi;
10. menerangkan dan mengenali berbagai tipe-tipe error pada pemrograman java;
11. melatih contoh-contoh program kecil tentang *try..catch*;
12. melatih contoh-contoh program kecil tentang *try..catch* banyak;
13. melatih contoh-contoh program kecil tentang *try..catch* bersarang;
14. melatih contoh-contoh program kecil tentang *try..catch..finally*;
15. Memahami contoh-contoh program kecil tentang *throw* dan *throws*.

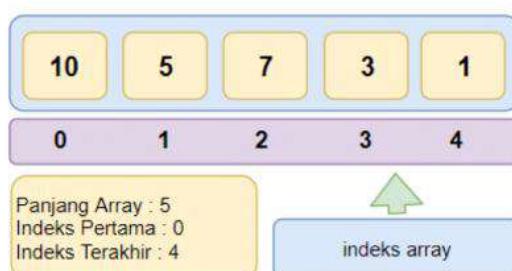
## KEGIATAN BELAJAR 1

### *Array*

ada pemrograman Java, *array* adalah objek yang digunakan untuk menyimpan sejumlah data, elemen-elemen pada *array* ditandai dengan indeks. Elemen dari *array* adalah juga variabel dan masing-masing indeks yang unik, yang panjang pada saat *array* dibuat. Indeks dari setiap *array* akan dimulai dengan indeks ke-0. Jadi jika *array* terdapat 5 data, maka indeks dari *array* tersebut adalah 4, atau berisi 6 data, maka indeks *array* tersebut adalah 5. *Array* dapat digunakan satu dimensi, dua, tiga, empat, dan seterusnya tergantung kebutuhan dari pengguna.

Berikut beberapa poin penting tentang *array* pada bahasa pemrograman Java, sebagai berikut:

1. Pada Java semua *array* dialokasikan secara dinamis.
2. Pada Java adalah objek, dengan demikian dapat mengetahui jumlah anggota.
3. Variabel *array* dideklarasikan seperti variabel lain dengan menyebutkan nama *array* setelah tipe data.
4. Indeks pada *array* dimulai dari 0.
5. Pada Java, *array* dapat dibuat *static*, variabel lokal atau menjadi parameter pada sebuah *method*.



**Gambar 7.1**  
**Ilustrasi Sebuah Array**

#### A. **ARRAY SATU DIMENSI DAN MENGALOKASIKAN MEMORI**

Ada dua cara untuk mendeklarasikan variabel untuk *array* satu dimensi.

Sintaksis:

```
tipe nama_var_array[];
```

atau

```
tipe[] nama_var_array;
```

Deklarasi *array* memiliki dua komponen, yaitu: tipe data dan variabel. Tipe data adalah tipe data dari *array*. Tipe data yang diperbolehkan pada *array* adalah tipe data primitif, seperti tipe data *char*, *float*, *double*, dan lain-lain atau juga bisa menggunakan objek dari *class*.



Gambar 7.2  
Ilustrasi Sebuah Satu Dimensi

Ketika sebuah *array* dideklarasikan, sesungguhnya baru membuat struktur *array*. Untuk benar-benar membuat *array* bisa digunakan maka kita harus memberikan lokasi memori ke *array*.

Sintaksis:

```
nama_var_array = new tipe[besar_alokasi];
```

Atau

```
tipe nama_var_array[];
nama_var_array = new tipe[besar_alokasi_mem];
```

Atau menggabungkan keduanya

```
tipe [] nama var array = new tipe[besar alokasi mem];
```

Catatan:

- Unsur-unsur dalam *array* yang dialokasikan oleh *new* secara otomatis diinisialisasi ke 0 (untuk tipe integer), *false* (untuk boolean), atau *null* (tipe referensi).
  - Untuk mendapatkan *array* diperlukan proses deklarasi variabel dan mengalokasikan memori.

Contoh 7.1:

Keluaran:

```
Daftar Array dalam ArrayInt
-----
Array pada indeks ke-0 : 5
Array pada indeks ke-1 : 7
Array pada indeks ke-2 : 4
Array pada indeks ke-3 : 8
Array pada indeks ke-4 : 9
```

Contoh 7.2: Menampilkan *array* dengan menggunakan pernyataan *for* lebih singkat.

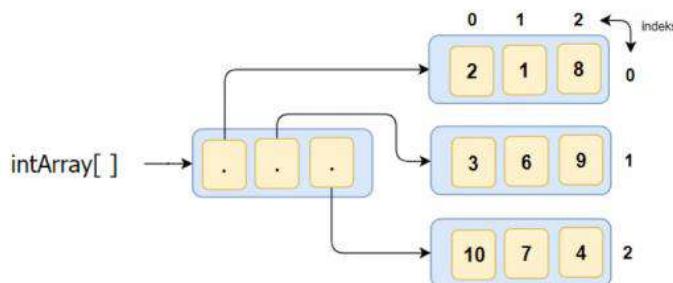
Keluaran antara contoh 7.1 dengan contoh 7.2 adalah sama, perbedaan dari keduanya bisa dilihat pada contoh 7.2 dengan kode program bercetak tebal.

## B. ARRAY MULTIDIMENSI

Setelah mematangkan pengetahuan tentang *array* satu dimensi. Pada bagian ini dibahas *array* multidimensi, *array* multidimensi adalah pengembangan *array* bentuk satu dimensi. *Array* multidimensi adalah variabel yang menyimpan sekumpulan data yang memiliki tipe sama dan elemen yang diakses melalui banyak indeks, biasanya digunakan untuk matriks.

Sintaksis:

```
tipe [] nama_var_array = new tipe[m][m]; //2D
tipe [] nama_var_array = new tipe[m][m][m]; //3D
```



Gambar 7.3  
Ilustrasi Sebuah *Array Multidimensi*

Contoh 7.3:

```
class multiDimensional
{
    public static void main(String args[])
    {
        // deklarasi array 2D
        int array2D[][] = { {2, 1, 8},
                           {3, 6, 9},
                           {1, 7, 4} };
        System.out.println("Tampilkan Array 2D");
    }
}
```

```

        System.out.println("-----");
        // tampilkan array 2D array
        for (int i=0; i < 3 ; i++)
        {
            for (int j=0; j < 3 ; j++)
                System.out.print(array2D[i][j] + "    ");

            System.out.println();
        }
    }
}

```

Keluaran:

```

Tampilkan Array 2D
-----
2   1   8
3   6   9
1   7   4

```

### C. PASSING ARRAY KE METHOD

Pada pemrograman Java dimungkinkan untuk melakukan *passing array* ke sebuah *method* seperti objek lainnya. Hal ini terkadang diperlukan untuk mentransfer sejumlah data antar *method*.

Contoh 7.4:

```

//Passing Array ke sebuah method
public class PassArraykeMethod
{
    public static void main(String args[])
    {
        int x[] = { 10, 20, 30};
        System.out.println(x[0]);

        tampil(x);
        System.out.println(x[0]);
    }

    //Method tampil
    public static void tampil(int y[])
    {
        System.out.println(y[0]);
    }
}

```

```

    y[0] = 100;
}
}

```

Penjelasan:

- `int x[] = { 10, 20, 30};` adalah mendeklarasikan *array* dengan tipe data *integer* sekaligus menginisiasi.
- `System.out.println(x[0]);` cetak ke layar *array* *x[0]* nilai yang dicetak adalah nilai *10*.
- `tampil(x);` adalah memanggil ***method tampil*** dengan parameter *array* *y[]* dengan tipe data *integer*, jika diperhatikan pada *method* tersebut melakukan cetak ke layar dengan perintah `System.out.println(y[0]);`. Mencetak *array* *x[0]* dan hasilnya adalah nilai *10*.
- Tetap pada ***method tampil*** baris `y[0] = 100;;`, perintah tersebut memperbarui nilai *array* *x[0]* yang tadinya bernilai 10 menjadi 100.
- Setelah pekerjaan ***method tampil*** selesai, program akan mengerjakan perintah di bawah kode program `tampil(x);` yang tidak lain adalah perintah untuk cetak ke layar yaitu: `System.out.println(x[0]);` dan mencetak nilai *array* *x[0]* yang baru yaitu *100*.

Keluaran:

```

10
10
100

```

#### D. MENGAMBIL NILAI *ARRAY* DARI *METHOD*

Kata lain dari sub judul di atas adalah bahwa *method* dapat mengembalikan nilai apa yang diminta.

Contoh 7.5:

```

// mengembalikan nilai permintaan
// array dari method

class BalikanNilaiDariMethod
{

```

```

public static void main(String args[])
{
    //Meminta data dari method
    int arrayInt[] = ReturnMethod();
    System.out.println("Data semua dari Method
ReturnMethod");
    System.out.println("-----");
    for (int x:arrayInt)
        System.out.print(x+" ");
}

public static int[] ReturnMethod()
{
    return new int[]{1,2,3,4,5,6,7,8,8,10};
}
}

```

Penjelasan:

- `int arrayInt[] = ReturnMethod();` adalah mendeklarasikan *array* dengan tipe data *integer* sekaligus mengisi data dari **method ReturnMethod()**.
- Sementara **method ReturnMethod()** dibuat dan untuk merespon permintaan *method* lain dengan kode program berikut:

```
return new int[]{1,2,3,4,5,6,7,8,8,10};
```

Keluaran:

```
Data semua dari Method ReturnMethod
-----
1 2 3 4 5 6 7 8 8 10
```

## E. COPY ARRAY

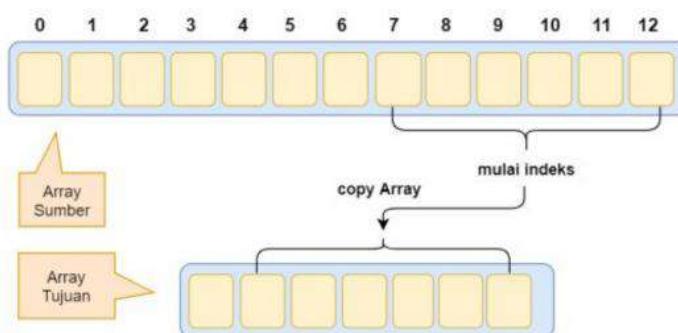
Pada *class system* ada sebuah method yang bernama *arraycopy* yang dapat dipergunakan untuk mengcopy dari suatu *array* ke *array* yang lain.

Sintaksis:

```
System.arraycopy(array_sumber,
                int_ambil_indeks,
                array_tujuan,
                int_sisip_awal_indeks,
                int_sisip_akhir_indeks);
```

Keterangan:

- `array_sumber` adalah nama variabel *array* yang ingin di-*copy* atau *array sumber*.
- `int_ambil_indeks` adalah harus angka, angka yang memberikan informasi kepada sistem bahwa *array* yang ingin di-*copy* mulai dari indeks sekitan.
- `array_tujuan` adalah *array tujuan*, antara *array sumber* dan *array tujuan* tidak harus sama panjangnya.
- `int_sisip_awal_indeks` adalah harus angka, angka yang menentukan pada indeks berapa mulai disisipkan pada *array tujuan*.
- `int_sisip_akhir_indeks` adalah harus angka, angka yang menentukan pada indeks terakhir disisipkan pada *array tujuan*.



Gambar 7.4  
Ilustrasi Copy Array

Contoh 7.6:

```
//Copy Array
class copyArray {
    public static void main(String[] args) {
```

```

char[] arraySumber = { 'S', 'e', 'p', 'e', 'd', 'a', ' ', 
                      'S', 'a', 'n', 't', 'a', 'i' };
//char[] arrayTujuan = new char[7];
char[] arrayTujuan = { 'L', 'a', 'g', 'i', ' ', 
                      'K', 'o', 'd', 'i', 'n', 'g' };
//cetak arraySumber
System.out.println(new String(arraySumber));

//cetak arrayTujuan sebelum disisip
System.out.println(new String(arrayTujuan));

//Proses copy sebagian data arraySumber
//dan menyisipnya ke arrayTujuan
System.arraycopy(arraySumber, 7, arrayTujuan, 5, 6);

//Cetak arrayTujuan setelah menyisipan data
System.out.println(new String(arrayTujuan));
}
}

```

Keluaran:

```

Sepeda Santai
Lagi Koding
Lagi Santai

```

## F. KLONING ARRAY

Pada pemrograman Java telah menyediakan fasilitas untuk mengkloning *array*, kloning *array* dapat dilakukan pada *array* satu dimensi maupun multidimensi.

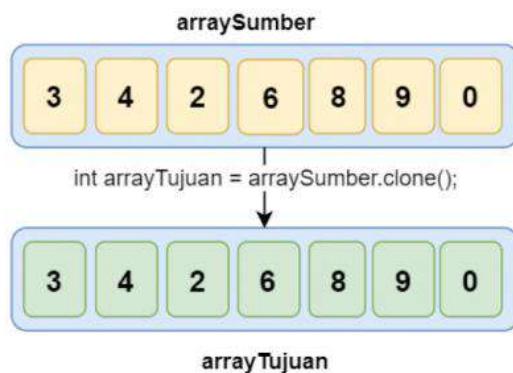
### 1. Kloning Array Satu Dimensi

Adapun perintah atau sintaksis untuk kloning *array* satu dimensi sebagai berikut:

```
tipe arrayTujuan[ ] = arraySumber; //1D
```

atau

```
arrayTujuan[ ] = arraySumber; //1D
```



Gambar 7.5  
Ilustrasi Kloning Array

Contoh 7.7:

```
//Kloning Array 1 Dimensi
class KloningArray1D
{
    public static void main(String args[])
    {
        int arraySumber[] = {1,2,3,4,5,6,7,8};
        int arrayTujuan[] = {10,12,13,14,15,16,17,18};

        System.out.println("Nilai arraySumber");
        System.out.println("-----");
    });

    for (int x:arraySumber)
        System.out.print(x+" ");

    System.out.println('\n');

    System.out.println("Nilai      arrayTujuan      sebelum
kloning");
    System.out.println("-----");
    for (int x:arrayTujuan)
        System.out.print(x+" ");

    System.out.println('\n');
    System.out.print("Bandingkan arraySumber[0] vs
arrayTujuan[0] = ");
    //bandingkan array arraySumber dan arrayTujuan
    //pada indeks 0 sebelum kloning
    System.out.println(arraySumber[0] == arrayTujuan[0]);
}
```

```

//Proses kloning
arrayTujuan = arraySumber.clone();

System.out.println('\n');
System.out.println("Nilai      arrayTujuan      setelah
kloning");
System.out.println("-----");
for (int x:arrayTujuan)
    System.out.print(x+" ");

System.out.println('\n');
System.out.print("Bandingkan arraySumber[0] vs
                arrayTujuan[0] = ");
//bandingkan array arraySumber dan arrayTujuan
//pada indeks 0 setelah kloning
System.out.println(arraySumber[0] == arrayTujuan[0]);

}
}

```

Keluaran:

```

Nilai arraySumber
-----
1 2 3 4 5 6 7 8

Nilai arrayTujuan sebelum kloning
-----
10 12 13 14 15 16 17 18

Bandingkan arraySumber[0] vs arrayTujuan[0] = false

Nilai arrayTujuan setelah kloning
-----
1 2 3 4 5 6 7 8

Bandingkan arraySumber[0] vs arrayTujuan[0] = true

```

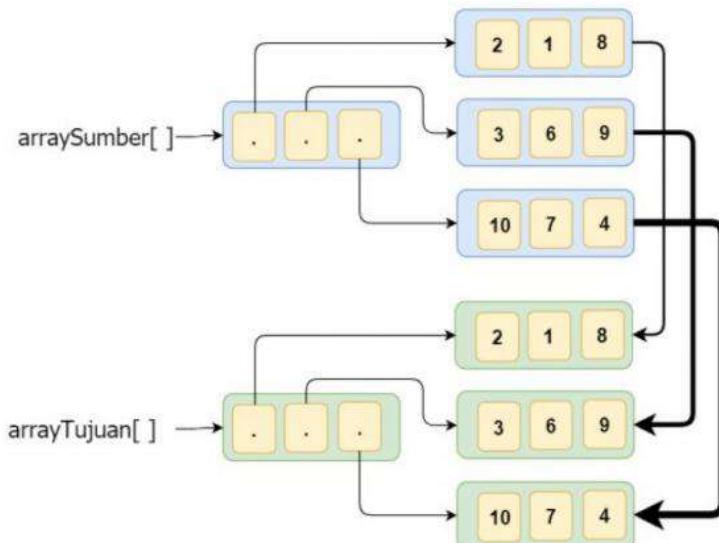
## 2. Kloning Array Multidimensi

Adapun perintah atau sintaksis untuk kloning *array* multidimensi sebagai berikut:

```
tipe arrayTujuan[][] = arraySumber; //2D
```

atau

```
arrayTujuan[][] = arraySumber; //2D
```



Gambar 7.6  
Ilustrasi Kloning Array Multidimensi

Contoh 7.8:

```
//Kloning Array Multidimensi
class KloningArrayMultidimensi
{
    public static void main(String args[])
    {
        int arraySumber[][] = {{1,2,3},{4,5,6},{7,8,9}};
        int arrayTujuan[][] = {{0,0,0},{0,0,0},{0,0,0}};

        //dibandingkan sebelum kloning
        System.out.println("dibandingkan sebelum kloning");
        System.out.println("-----");
        System.out.println(arraySumber[0][1] == arrayTujuan[0][1]);
        System.out.println(arraySumber[1] == arrayTujuan[1]);

        //proses kloning
        arrayTujuan = arraySumber.clone();

        System.out.println('\n');
        //dibandingkan setelah kloning
    }
}
```

```
System.out.println("dibandingkan sebelum kloning");
System.out.println("-----");
System.out.println(arraySumber[0][1] == arrayTujuan[0][1]);
System.out.println(arraySumber[1] == arrayTujuan[1]);

}
```

### Keluaran:

```
dibandingkan sebelum kloning
-----
false
false

dibandingkan setelah kloning
-----
true
true
```



LATIHAN

Untuk memperdalam pemahaman Anda mengenai materi di atas, kerjakanlah latihan berikut!

- 1) Sebutkan poin-poin penting yang harus diketahui lebih awal sebelum mengaplikasikan *array* pada pemrograman Java?
- 2) Pada pemrograman Java dikenal dengan *array* satu dimensi dan *array* multidimensi, jelaskan kedua jenis *array* tersebut dan berikan contoh masing-masing.

### Petunjuk Jawaban Latihan

- 1) Poin-poin penting sebelum mengaplikasikan *array* pada pemrograman Java sebagai berikut:
  - a) Pada Java semua *array* dialokasikan secara dinamis;
  - b) Pada Java adalah objek, dengan demikian dapat mengetahui jumlah anggota.

- c) Variabel *array* dideklarasikan seperti variabel lain dengan menyebutkan nama *array* setelah tipe data.
  - d) Indeks pada *array* dimulai dari 0
  - e) Pada Java, *array* dapat dibuat *static*, variabel lokal atau menjadi parameter pada sebuah *method*.
- 2) *Array* satu dimensi adalah sederetan variabel yang bertipe sama, contoh:

```
class arrayTotalRatarata {
    public static void main(String[] args) {
        int[] nilaiarr = {2, -9, 0, 5, 12, -25, 22, 9, 8, 12};
        int total = 0;
        Double ratarata;

        for (int nilai: nilaiarr) {
            total += nilai;
        }

        int panjangArr = nilaiarr.length;
        ratarata = ((double)total / (double)panjangArr);
        System.out.println("Total      = " + total);
        System.out.println("Rata-Rata = " + ratarata);
    }
}
```

*Array* multidimensi adalah *array* yang memiliki lebih dari satu dimensi atau *array ke array*, contoh:

```
class Testarray3{
    public static void main(String args[]){
        //declaring and initializing 2D array
        int arr[][]={{1,2,3},{2,4,5},{4,4,5}};
        //printing 2D array
```

```

for(int i=0;i<3;i++){
    for(int j=0;j<3;j++){
        System.out.print(arr[i][j]+" ");
    }
    System.out.println();
}
}
}

```

Contoh untuk *array* multidimensi diambil dari laman <https://www.javatpoint.com/array-in-java>.



**RANGKUMAN**

---

*Array* adalah objek yang dapat menyimpan data, elemen-elemen pada *array* ditandai dengan indeks, data yang tersimpan dalam *array* dapat berupa tipe primitif. Beberapa pengetahuan yang mesti diketahui mengenai *array* pada Java, yaitu: *Array* dialokasikan secara dinamis; jumlah anggota pada *array* dapat diketahui; variabel *array* dideklarasikan seperti variabel lainnya; indeks pada *array* dimulai dari 0; *array* dapat berupa *array* satu dimensi dan juga dapat berupa *array* multidimensi.

*Array* memiliki dua komponen, yaitu: tipe data dan variabel, tipe data yang diperbolehkan pada *array* adalah tipe data primitif dan objek referensi. Beberapa fitur lain yang bisa digunakan pada *array* adalah sebuah *array* dapat di-copy ke *array* yang lain, selain itu fitur kloning *array* juga sangat memungkinkan untuk digunakan pada pemrograman Java.



**TES FORMATIF 1**

---

Pilihlah satu jawaban yang paling tepat!

- 1) Indeks sebuah *array* selalu dimulai dari angka....
  - A. -1
  - B. 0
  - C. 1
  - D. 2

- 2) Dua komponen yang harus disertakan ketika membuat *array* pada Java adalah....

- A. Tipe Data dan Variabel
- B. Tipe data dan alokasi memori
- C. Alokasi memori dan data
- D. Data dan penyertaan *method*

- 3) Perhatikan deklarasi *array* berikut:

```
int nilai[];
```

kode program di atas memberikan informasi bahwa....

- A. Mendeklarasikan *array* satu dimensi
- B. Mendeklarasikan *array* dua dimensi
- C. Mendeklarasikan *array* tiga dimensi
- D. A, B, dan C benar

- 4) Perhatikan deklarasi *array* berikut:

```
int nilai[];
```

Untuk bisa digunakan *array* di atas harus mengalokasikan memori dengan kode program....

- A. nilai = new single[4];
- B. int nilai = new single[4];
- C. nilai = new int[];
- D. nilai = new int[4];

- 5) Perhatikan potongan kode program berikut:

```
int array2D[][] = { {2, 1, 8},  
                    {3, 6, 9},  
                    {1, 7, 4} };
```

Untuk menampilkan nilai data *array* dengan nilai 3 adalah dengan cara....

- A. System.out.print(array2D[0][0]);
- B. System.out.print(array2D[0][1]);
- C. System.out.print(array2D[1][0]);
- D. System.out.print(array2D[1][1]);

6) Perhatikan *script* berikut:

```
int array2D[][] = { {2, 1, 8},  
                    {3, 6, 9},  
                    {1, 7, 4} };  
  
BooleanLogic = array2D[1][1] > array2D[1][0];  
...
```

Nilai dari variabel BooleanLogic adalah....

- A. *true*
- B. *false*
- C. 4
- D. 7

7) Perhatikan *script* berikut:

```
System.arraycopy(arraySumber, 7, arrayTujuan, 5, 6);
```

Fungsi utama dari kode program di atas adalah....

- A. Kloning data *array*
- B. Mengambil data dari *Method*
- C. Copy data dari *array* lain
- D. Copy data dari *array* dirinya sendiri

8) Perhatikan *script* berikut:

```
int array2D[][] = { {2, 1, 8},  
                    {3, 6, 9},  
                    {1, 7, 4} };  
  
nilai = array2D[2][1];
```

Nilai dari variabel nilai adalah....

- A. 3
- B. 6
- C. 9
- D. 7

9) Pada *array* pemrograman Java untuk *array* dua dimensi, maka pernyataan:

*Nilai[x][y];*

Data  $x$  dan  $y$  adalah....

- A. Baris dan kolom
- B. Kolom dan baris
- C. Semuanya baris
- D. Semuanya kolom

10) Pernyataan kloning untuk satu dimensi dengan dua dimensi adalah....

- A. Sama
- B. Berbeda
- C. Berbeda untuk beberapa hal
- D. Sama untuk beberapa hal

Cocokkanlah jawaban Anda dengan Kunci Jawaban Tes Formatif 1 yang terdapat di bagian akhir modul ini. Hitunglah jawaban yang benar. Kemudian, gunakan rumus berikut untuk mengetahui tingkat penguasaan Anda terhadap materi Kegiatan Belajar 1.

$$\text{Tingkat penguasaan} = \frac{\text{Jumlah Jawaban yang Benar}}{\text{Jumlah Soal}} \times 100\%$$

Arti tingkat penguasaan: 90 - 100% = baik sekali

80 - 89% = baik

70 - 79% = cukup

< 70% = kurang

Apabila mencapai tingkat penguasaan 80% atau lebih, Anda dapat meneruskan dengan Kegiatan Belajar 2. **Bagus!** Jika masih di bawah 80%, Anda harus mengulangi materi Kegiatan Belajar 1, terutama bagian yang belum dikuasai.

**KEGIATAN BELAJAR 2*****Method***

*M*ethod adalah sekumpulan pernyataan yang dikelompokkan bersama untuk melakukan tugas-tugas tertentu. Banyak pekerjaan yang secara tidak sadar kita selesaikan dengan cara memanggil suatu *method* di pemrograman Java, dalam bahasa pemrograman yang lain misalnya pada bahasa pemrograman Pascal, *method* disebut dengan *function* atau *procedure*. Ribuan *method* yang dibangun didalam bahasa pemrograman untuk memperkaya fitur-fitur bahasa pemrograman itu sendiri.

**A. MEMBUAT METHOD**

Adapun sintaksis dari membuat *method* adalah sebagai berikut:

```
public static tipe_return namaMethod(tipe x, tipe y, tipe z)
{
    blok pernyataan
}
```

Keterangan:

- `public static` adalah *modifier*;
- `tipe_return` adalah jenis nilai yang dikembalikan oleh *method*
- `namaMethod` adalah nama *method* itu sendiri
- `tipe x, tipe y, tipe z` adalah parameter dengan deklarasi variabel dan tipe datanya.

Sekarang kita akan membuat contoh sebuah *method* untuk mencari tahu nilai yang dimasukkan melalui parameter *method* apakah ganjil atau genap.

Contoh 7.9: membuat *method*

```
public static int CekGanjilGenap(int nilai) {
    nilai = nilai % 2;
    return nilai;
}
```

Keterangan:

- Nama *method* adalah `CekGanjilGenap`
- Paramater adalah `nilai` dengan tipe data integer;
- Setelah melakukan operasi variabel `nilai` di `return`.

## B. MEMANGGIL METHOD

*Method* yang sudah dibuat tidak akan pernah digunakan jika tidak dipanggil, secanggih apapun *method* yang dibuat. Pemanggilan *method* cukup sederhana, hanya dengan menyebutkan nama *method*. Berikut contoh pemanggilan *method*.

Contoh 7.10: memanggil dan kemudian nilai balikan *method* ditampung dalam variabel.

```
//Membuat method menentukan bilangan ganjil/genap
class pemanggilanMethod {
    public static void main(String[] args) {
        int x,y = 0;
        y = 3;
        x = cariNilaiGanjilGenap(y);

        if (x == 1)
            System.out.println("Angka "+y+" adalah Ganjil");
        else
            System.out.println("Angka "+y+" adalah Genap");
    }

    public static int cariNilaiGanjilGenap(int nilai) {
        nilai = nilai % 2;
        return nilai;
    }
}
```

Keluaran:

Angka 3 adalah Ganjil

Untuk melihat lebih jelas antara tubuh *method* dan pemanggilan *method* pada gambar berikut:

```
//Membuat method menentukan bilangan ganil/genap
class pemanggilMethod {
    public static void main(String[] args) {
        int x,y = 0;
        y = 3;
        //memanggil method
        x = cariNilaiGanjilGenap(y); panggil method

        if (x == 1)
            System.out.println("Angka "+y+" adalah Ganjil");
        else
            System.out.println("Angka "+y+" adalah Genap");
    }

    public static int cariNilaiGanjilGenap(int nilai) {
        nilai = nilai % 2;
        return nilai; nilai yang dikembalikan
    }
} method
```

Gambar 7.6  
Method dan Pemanggilan Method

### C. MEMBUAT *METHOD* TIDAK MENGEMBALIKAN NILAI

Contoh *method* sebelumnya adalah *method* yang bisa mengembalikan nilai. Ada cara pada pemrograman Java agar sebuah *method* tidak mengembalikan nilai, yaitu dengan menggunakan *method void*. Dengan menyertakan *keyword void* pada di depan nama *method*, maka *method* tidak akan mengembalikan nilai.

Sintaks:

```
public static void namaMethod(tipe x, tipe y, tipe z)
{
    blok pernyataan
}
```

Contoh 7.11:

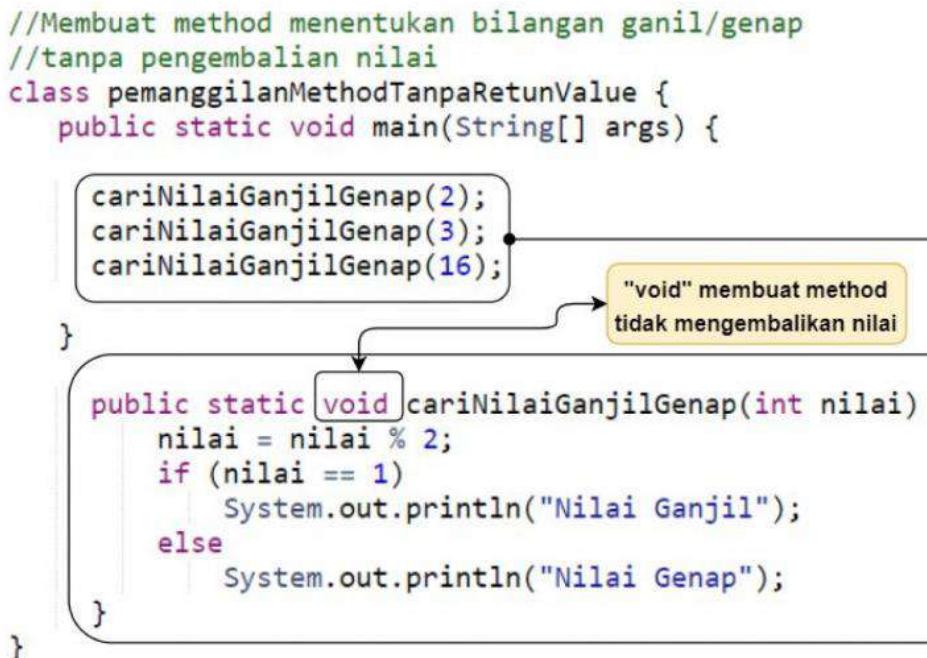
```
//Membuat method menentukan bilangan ganjil/genap
//tanpa pengembalian nilai
class pemanggilanMethodTanpaReturnValue {
    public static void main(String[] args) {

        cariNilaiGanjilGenap(2);
        cariNilaiGanjilGenap(3);
        cariNilaiGanjilGenap(16);

    }

    public static void cariNilaiGanjilGenap(int nilai) {
        nilai = nilai % 2;
        if (nilai == 1)
            System.out.println("Nilai Ganjil");
        else
            System.out.println("Nilai Genap");
    }
}
```

Alur sederhana program untuk penyertaan *void* pada *method* untuk tidak mengembalikan nilai dari *method*.



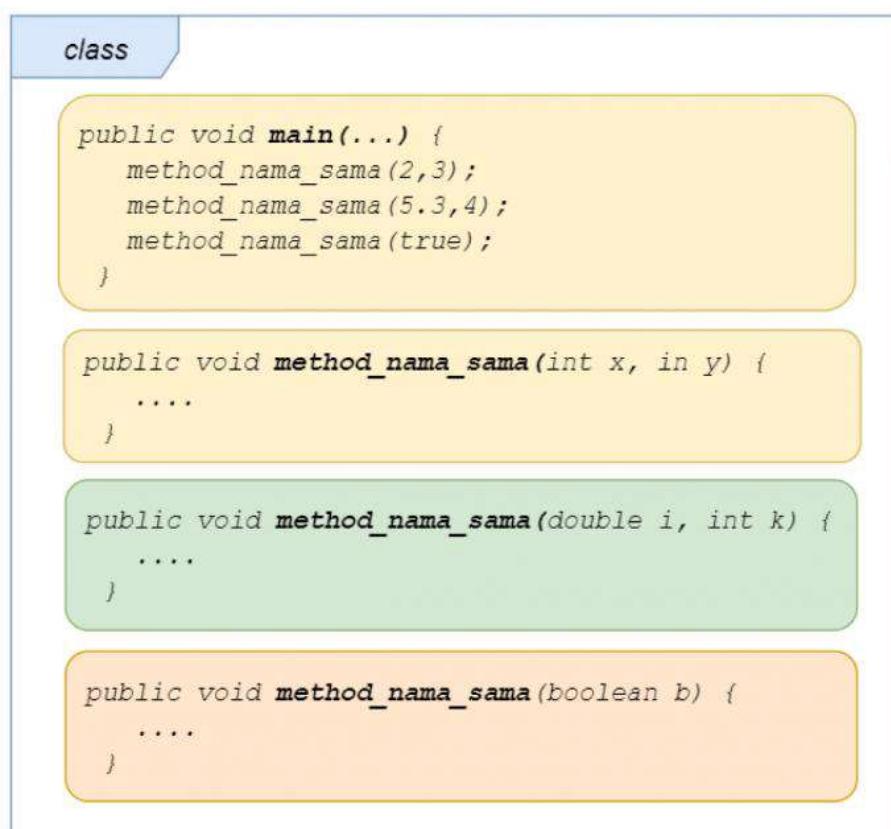
Gambar 7.6  
Method yang Tidak Mengembalikan Nilai

Keluaran:

```
Nilai Genap
Nilai Ganjil
Nilai Genap
```

## D. METHOD OVERLOADING

Salah satu kemampuan Java yang cukup unik adalah kemampuan sebuah *class* memiliki nama *method* yang sama, tapi yang membedakan adalah parameternya, *method* seperti ini disebut dengan *Method Overloading*. Perbedaan parameter pada *method overloading* mencakup: Jumlah parameter, Tipe data parameter, dan urutan dari tipe data parameter.



Gambar 7.7  
*Method Overloading*

Contoh 7.12:

```
// Method Overloading
public class methodOverloading {
```

```
// Driver code
public static void main(String args[]) {

    //Panggil method cariLuas
    System.out.println(cariLuas(32,2));
    System.out.println(cariLuas(5,3));
}

// Overloaded cari luas Segi Tiga
public static double cariLuas(double alas, int tinggi) {
    return (1/2 * (alas * tinggi));
}

// Overloaded cari luas Persegi Panjang
public static int cariLuas(int panjang, int lebar) {
    return (panjang * lebar);
}
```

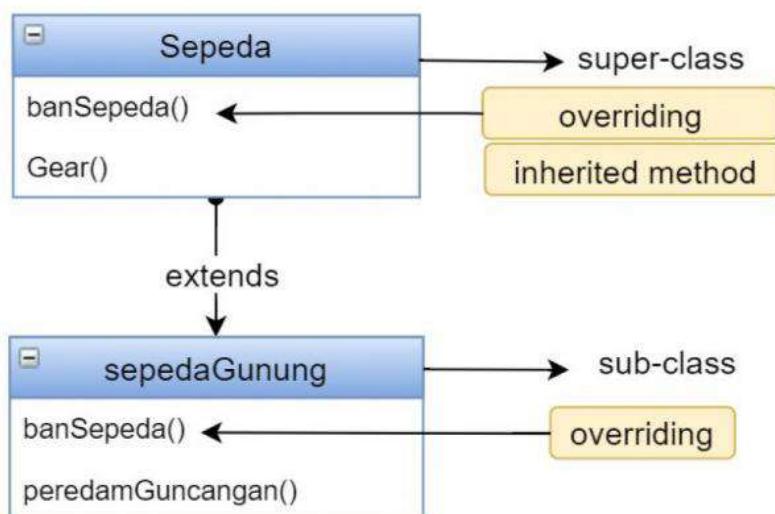
Keluaran:

64  
15

### E. Method Overriding

*Method Overriding* adalah bertujuan untuk meng-*inheritance* (mewariskan) suatu *method* dari *super-class* ke *sub-class*, ketika *super-class* memiliki *method* dan kemudian *method* tersebut dideklarasikan di *sub-class* maka itulah yang sebut meng-*extends class* lain.

Anggaphlah objek sepeda, sepeda adalah *super-class*, sepeda dapat diwariskan kepada menjadi *Sepeda Gunung*, *Sepeda Santai* atau yang lainnya, kesemuanya mempunyai prototipe (*blueprint*) yang sama, walau setelah melihat kegunaannya memiliki perbedaan dan ciri-ciri masing-masing, Misalnya: *Sepeda Gunung* dan *Sepeda Santai*, kedua sepeda ini memiliki ban sepeda yang mengikuti dasar velg yaitu bulat, akan tetapi melihat area yang akan dijelajahi, maka *Sepeda Santai* memiliki ban yang lebih halus karena lebih banyak digunakan di area yang lebih datar dan tak bergelombang seperti aspal, sedangkan *Sepeda Gunung* agak lebih kasar karena akan digunakan pada area yang tidak beraspal, *Sepeda Gunung* rata-rata didesain untuk tahan dan elastis terhadap guncangan.



**Gambar 7.8**  
*Method Overriding*

Contoh 7.13:

```

// method overriding

// Super Class
class Sepeda
{
    void banSepeda() {
        System.out.println("Ban Sepeda lebih kecil");
    }

    void gear() {
        System.out.println("Semua punya Gear");
    }
}

// Sub (Inherited) class
class sepedaGunung extends Sepeda
{
    @Override
    void banSepeda() {
        System.out.println("Ban Sepeda bergigi tebal dan kasar");
    }

    void gear() {
        System.out.println("Memiliki 18-30 gear yang bisa dipindah-
pindah");
    }

    void peredamGuncangan() {
    }
}
  
```

```

        System.out.println("peredam      terhadap      guncangan      saat
berkendaraan");
    }
}

// Class Utama
class methodOverriding
{
public static void main(String[] args)
{
    System.out.println("Dari Super Class");
    System.out.println("-----
-----");
    // Super Class (SupC)
    Sepeda objek_SupC1 = new Sepeda();
    objek_SupC1.banSepeda();

    Sepeda objek_SupC2 = new Sepeda();
    objek_SupC2.gear();

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

    System.out.println("Dari Sub Class");
    System.out.println("-----
-----");
    // Sub Class (SubC)
    Sepeda objek_SubC1 = new sepedaGunung();
    objek_SubC1.banSepeda();

    Sepeda objek_SubC2 = new sepedaGunung();
    objek_SubC2.gear();

    System.out.println("Sub Class tdk menggunakan method super-
class");
    System.out.println("-----
-----");
    sepedaGunung objek_SubC3 = new sepedaGunung();
    objek_SubC3.peredamGuncangan();
}
}

```

Penjelasan program:

- *class Sepeda* adalah *super-class* yang memiliki *method-method* dasar dari sepeda, yaitu **banSepeda** dan **Gear**.
- pada *class sepedaGunung* melakukan *extends* ke *class Sepeda*, artinya bahwa semua *method* yang ada pada *class Sepeda* boleh digunakan/ditulis ulang oleh *class sepedaGunung*.

- *sepedaGunung* adalah *sub-class* yang juga memiliki method yang di extends dari *super-class* *Sepeda* yaitu *banSepeda* dan *Gear*. Namun ada *method* yang baru pada *sub-class* yang tidak diturunkan oleh *super-class*, yaitu *method peredamGuncangan* alasannya adalah tidak semua sepeda memiliki karakter *method* tersebut.
- Kode program *Sepeda objek\_SupC1 = new Sepeda();* adalah mendeklarasikan dan memanggil *method* di *super-class* pada *method* utama di *class* utama.
- Kode program *Sepeda objek\_SubC1 = new sepedaGunung();* adalah mendeklarasikan dan memanggil *method* di *sub-class* yang meng-extends *method Sepeda*.
- Kode program *sepedaGunung objek\_SubC3 = new sepedaGunung();* adalah mendeklarasikan dan memanggil *method* di *sub-class* tanpa ada di *super-class*.

Keluaran:

Dari Super Class

Ban Sepeda lebih kecil  
Semua punya Gear

Dari Sub Class

Ban Sepeda bergigi tebal  
Memiliki 18-30 gear yang bisa dipindah-pindah

Sub Class tdk menggunakan method super-class

peredam terhadap guncangan saat berkendaraan

## F. STATIC METHOD TURUNAN DIBAYANGI STATIC METHOD DASAR

Jika nama *method* statis *class sub-class* (turunan) sama dengan nama *method* statis pada *class super-class*, maka *method* menjadi bayangan.

Contoh 7.14:

```
// Super-Class
class Sepeda {
    static void banSepeda() {
```

```

        System.out.println("Ban sepeda lebih kecil");
    }
}

//Sub-Class
class sepedaGunung extends Sepeda {
    static void banSepeda() {
        System.out.println("Ban sepeda lebih kecil dan bergigi");
    }
}
//Class Utama
public class Main {
    public static void main(String args[]) {
        Sepeda objek_sepeda = new sepedaGunung();
        objek_sepeda.banSepeda(); // prints A.fun()
    }
}

```

Penjelasan:

- *Super-class* dan *sub-class* memiliki *method* statis yang sama yaitu *banSepeda*, dan class *sepedaGunung* melakukan *extends* ke class *sepeda*.
- Pada class pada bagian *method* utama, yang dideklarasi dari objek adalah class *sepedaGunung*, akan tetapi karena *method* statisnya sama jadi yang dicetak adalah yang ada di *super-class*.

Keluaran:

Ban sepeda lebih kecil



## LATIHAN

---

Untuk memperdalam pemahaman Anda mengenai materi di atas, kerjakanlah latihan berikut!

- 1) Apa yang dimaksud dengan *method* mengembalikan nilai dan *method* tidak mengembalikan nilai?
- 2) Apa yang dimaksud dengan *method overloading* dan *method overriding*?
- 3) Buatlah sebuah program sederhana menggunakan *method* untuk memanggil *method* yang lain.

*Petunjuk Jawaban Latihan*

- 1) *Method* mengembalikan nilai adalah *method* yang jika dipanggil dipanggil membawa nilai jika dipanggil. *Method* tidak mengembalikan nilai adalah ketika method dipanggil tidak mengembalikan nilai balik. Ciri-ciri dari *method* tidak mengembalikan nilai ada *keyword void* didepan nama *method*.
- 2) *Method overloading* adalah kemampuan *method* dipanggil dengan nama yang sama akan tetapi mampu menyesuaikan diri ketika dipanggil berdasarkan parameter pemanggilnya (lihat sub bagian D.). *Method overriding* adalah proses pewarisan *method* dari *super-class* ke *sub-class*.
- 3) Contoh kode program memanggil *method* dari *method*, sebagai berikut:

```
public class ExampleMinNumber {

    public static void main(String[] args) {
        int a = 11;
        int b = 6;
        int c = minFunction(a, b);
        System.out.println("Minimum Value = " + c);
    }

    /** returns the minimum of two numbers */
    public static int minFunction(int n1, int n2) {
        int min;
        if (n1 > n2)
            min = n2;
        else
            min = n1;

        return min;
    }
}
```

Sumber: [https://www.tutorialspoint.com/java/java\\_methods.htm](https://www.tutorialspoint.com/java/java_methods.htm)

Keterangan:

*Method minFunction* dipanggil dari *method* dengan perintah:

```
int c = minFunction(a, b);
```

perintah-perintah seperti ini sangat dimungkinkan pada pemrograman Java.



### RANGKUMAN

---

*Method* adalah sekumpulan pernyataan yang dikelompokkan bersama untuk melakukan tugas-tugas tertentu. Pada pemrograman lain *method* biasa disebut dengan *function* atau *procedure*. Pada *method* jika tidak mau mengembalikan nilai, cukup menyertakan pernyataan *void* di awal nama *method*. *Method* yang sudah jadi harus tetap dipanggil agar bisa digunakan.

Salah satu kemampuan Java yang cukup unik adalah kemampuan sebuah *class* memiliki nama *method* yang sama, tapi yang membedakan adalah parameternya yang disebut dengan *Method overloading*. Sedangkan *Method overriding* adalah bertujuan untuk meng-*inheritance* (mewariskan) suatu *method* dari *super-class* ke *sub-class*, ketika *super-class* memiliki *method* dan kemudian *method* tersebut dideklarasikan di *sub-class* maka itulah yang sebut meng-*extends class* lain.



### TES FORMATIF 2

---

Pilihlah satu jawaban yang paling tepat!

- 1) Perhatikan *method* di bawah ini...

```
public static int CekGanjilGenap(int nilai) {
    nilai = nilai % 2;
    return nilai;
}
```

Pada *method* *public static* adalah....

- A. Nama *Method*
- B. *Modifier*
- C. variabel
- D. Tipe data

- 2) Supaya nilai suatu *method* tidak perlu dikembalikan, maka nama didepan *method* harus menyertakan *keyword*....
- A. *while*
  - B. *return*
  - C. *void*
  - D. *break*
- 3) Jika sebuah contoh *method* sebagai berikut....

```
public static double contoh(int nilai, boolean logik, double  
    kecepatan)  
{  
    ...  
    return x;  
}
```

Maka cara memanggil *method* tersebut yang tepat adalah....

- A. ...  
y = contoh(2.5, false, 2);  
...
  - B. ...  
y = contoh(2.5, 3, 2);  
...
  - C. ...  
y = contoh(2.5, 4, false);  
...
  - D. ...  
y = contoh(2, false, 2.5);  
...
- 4) Aturan pembuatan sebuah *method* dari sisi penempatan *method* dalam sebuah *class* adalah....
- A. *Method* harus diluar *class*
  - B. *Method* boleh berada di bawah atau di atas *method* utama
  - C. *Method* harus di atas *method* utama
  - D. *Method* harus di bawah *method* utama
- 5) Perhatikan potongan kode program berikut:

```
public class Contoh {  
  
    // Driver code  
    public static void main(String args[]) {
```

```
    ...
}

public static double cariNilaiX(double alas, int tinggi) {
    return (...);
}

public static int cariNilaiX (int panjang, int lebar) {
    return (...);
}
```

Method *cariNilaiX* adalah *method* yang dikenal dengan *method*....

- A. *Out of memory*
  - B. *Overriding*
  - C. *Overloading*
  - D. *Constructor*
- 6) Sebuah *method* yang sudah dideklarasikan pada *super-class* dan kemudian kembali dideklarasikan pada *sub-class* disebut dengan istilah....
- A. *Overriding*
  - B. *Overload*
  - C. *Overtime*
  - D. *Runtime*
- 7) Sebuah *method* yang dipanggil dari *class* lain, dapat dilakukan dengan berbagai macam cara, contoh yang salah dalam memanggil sebuah *method* adalah sebagai berikut....
- A. ...  
a = nilaiX(2,2);  
...
  - B. ...  
a = nilaiX{2,2};  
...
  - C. ...  
If (nilaiX(2,2) == true) {  
 ...  
}  
...
  - D. ...  
If (nilaiX(2,2) == true) {  
 ...  
}  
...

- 8) Contoh kode program dengan nilai *return* yang benar adalah sebagai berikut....
- A. 

```
public static void nilaix(int nilai) {  
    return nilaix;  
}
```
  - B. 

```
public static boolean nilaix(int nilai) {  
    ...  
    return 5;  
}
```
  - C. 

```
public static int nilaipii(int nilai) {  
    return 3.14;  
}
```
  - D. 

```
public static boolean nilaix(int nilai) {  
    return true;  
}
```
- 9) Pernyataan *void* pada sebuah *method* memberikan pengertian....
- A. Sebuah *method* tidak perlu mengembalikan nilai balik
  - B. Sebuah *method* dapat dipergunakan pada *class* lain
  - C. Sebuah *method* perlu mengembalikan nilai balik
  - D. Sebuah *method* perlu mengembalikan pernyataan *return* pada tubuh *method*
- 10) Jika *method* dideklarasikan dengan tipe data *double*, maka nilai baliknya harus....
- A. *double*
  - B. *int*
  - C. *boolean*
  - D. A, B, dan C benar

Cocokkanlah jawaban Anda dengan Kunci Jawaban Tes Formatif 2 yang terdapat di bagian akhir modul ini. Hitunglah jawaban yang benar. Kemudian, gunakan rumus berikut untuk mengetahui tingkat penguasaan Anda terhadap materi Kegiatan Belajar 2.

$$\text{Tingkat penguasaan} = \frac{\text{Jumlah Jawaban yang Benar}}{\text{Jumlah Soal}} \times 100\%$$

Arti tingkat penguasaan: 90 - 100% = baik sekali

80 - 89% = baik

70 - 79% = cukup

< 70% = kurang

Apabila mencapai tingkat penguasaan 80% atau lebih, Anda dapat meneruskan dengan Kegiatan Belajar 3. **Bagus!** Jika masih di bawah 80%, Anda harus mengulangi materi Kegiatan Belajar 2, terutama bagian yang belum dikuasai.

**KEGIATAN BELAJAR 3****Eksepsi (*Exception*)**

**S**ebuah *bugs* adalah istilah umum yang digunakan untuk memberi label kesalahan, gagal, *error* dalam bahasa pemrograman. *Bugs* dan *error* adalah sebuah keniscayaan dalam membuat program sama halnya manusia yang tidak luput dari kesalahan. *Bugs* dan *error* adalah hal yang sering muncul meskipun program tersebut dibuat oleh *programmer* handal. Untuk menghindari *bugs* dan *error* dalam pemrograman ada yang disebut dengan istilah Eksepsi atau dalam Bahasa Inggris adalah *Exception*.

Eksespsi adalah peristiwa yang menyebabkan program menjadi tidak normal, pemicu dari eksepsi bias terjadi karena masukan data yang tidak valid, file yang dimaksud dalam kode program tidak ada, memori yang tak didefiniskan dan hal ini terjadi pada saat *Runtime Error* atau *error* yang terjadi pada saat program dieksekusi oleh *interpreter*.

Eksepsi dapat ditangani oleh setiap bahas pemrograman dengan menerapkan penanganan eksepsi, fitur ini adalah fitur yang paling penting dalam bahasa pemrograman, tak terkecuali bahasa pemrograman Java, fitur tersebut berjalan atau menangani pada saat *runtime error* sehingga jalannya aplikasi dapat dipertahankan, terlebih lagi jika lagi sebuah kesalahan dapat ditangani dan mengeluarkan sebuah pesan kesalahan yang komunikatif.

**A. KATEGORI *ERROR* DALAM PEMROGRAMAN JAVA**

Ada tiga kategori *error* dalam Bahasa Pemrograman yang harus diketahui oleh calon-calon *programmer* java, yaitu: *Compilation Error*, *Runtime Error*, dan *Logic Error*.

**1. Compilation Error**

*Compilation Error* adalah *error* yang terjadi pada saat program dikompilasi, *compiler* Java akan memunculkan Kesalahan Sintaksis, Kesalahan Semantik, atau Kesalahan *Cascading*.

**a. Kesalahan Sintaksis**

Kesalahan *Sintaksis* adalah kesalahan penulisan program yang tidak sesuai dengan aturan Bahasa pemrograman Java. Sebagai contoh, bahasa

pemrograman Java pada aturannya menyatakan bahwa bahwa setiap pernyataan harus diakhiri dengan titik koma (;).

Contoh 7.15:

```
//Program Error
class Sintaksis
{
    public static void main(String[] args)
    {
        System.out.println("Belajar Java")
    }
}
```

Program di atas jika dikompilasi maka akan terjadi *error* seperti berikut:

```
prog.java:6: error: ';' expected
                    System.out.println("Belajar Java")
   ^
1 error
```

Ditemukan *error* pada baris ke-6, kemudian Java memberikan detail kesalahan yang terjadi, yaitu: kurang titik koma (;), dan lebih jauh dari itu Java mampu menunjukkan di baris dan kolom mana *error* terjadi, yaitu dengan memberikan tanda (^). Sehingga kesalahan-kesalahan yang muncul begitu komunikatif dengan *programmer*, dan jika kode program di atas disempurnakan dengan menambahkan titik koma dibelakang setiap pernyataan.

Contoh 7.16. program yang benar

```
//Program Error
class Sintaksis
{
    public static void main(String[] args)
    {
        System.out.println("Belajar Java");
    }
}
```

Keluaran:

Belajar Java

Kesalahan lain yang sering terjadi pada kesalahan Sintaksis adalah kesalahan dalam mengeja variabel atau *method*, harus dingatkan lagi bahwa variabel ataupun nama *method* pada Java menganut sistem *case sensitive*, yaitu huruf kecil dan huruf besar dibedakan walaupun dalam pengucapan sama. (*Sepeda* dengan *sepeda* adalah berbeda pada pemrograman Java).

Contoh 7.17:

```
//Program Error
class Sintaksis
{
    public static void main(String[] args)
    {
        int a,b,c;
        c = a * b;
        System.out.println("Cetak C : "+C);
    }
}
```

Perhatikan pesan kesalahan yang muncul sebagai berikut:

```
prog.java:8: error: cannot find symbol
                    System.out.println("Cetak C : "+C);
   ^
symbol:   variable C
location: class Sintaksis
1 error
```

Pesan kesalahan menunjukkan tidak adanya simbol “C”, yang biasanya mengartikan bahwa Anda salah menuliskan nama variabel, nama *method*, atau *keyword*.

#### b. Kesalahan Semantik

Sangat penting untuk diperhatikan, bahwa meskipun program yang dibuat mungkin benar secara sintaks, tapi boleh jadi kompiler menemukan beberapa kesalahan semantik, misalnya: kesalahan karena tak menginisiasi variabel.

Contoh 7.18:

```
//Program Error
class Sintaksis
{
    public static void main(String[] args)
    {
        int a,b,c;
        c = a * b;
        System.out.println("Cetak c : "+c);
    }
}
```

Program ketika dikompilasi, sistem kompiler akan mengeluarkan pesan kesalahan berikut:

```
prog.java:7: error: variable a might not have been initialized
      c = a * b;
              ^
prog.java:7: error: variable b might not have been initialized
      c = a * b;
              ^
2 errors
```

Ada dua kesalahan yang ditunjukkan oleh kompiler, yaitu semua pada baris ke-7, dengan pesan kesalahan bahwa variabel *a* dan *b* belum diinisiasi. Namun kenapa variabel *c* tidak terdapat kesalahan? Itu disebabkan karena variabel *c* hanya menjadi penampung nilai operasi. Dalam pemrograman Java, variabel lokal tidak mempunyai nilai awal, jadi harus diinisiasi pada saat variabel dideklarasikan, kecuali variabel tersebut hanya menjadi penampung variabel lain, seperti variabel *c* di atas.

#### c. Kesalahan *Cascading*

Kesalahan *Cascading* tidak seperti kedua kesalahan sintaksis dan semantik, namun perlu pembahasan untuk dipahami. Terkadang pesan kesalahan membingungkan *programmer*, kompiler mampu mengeluarkan pesan kesalahan, akan tetapi pesan kesalahan tersebut memberikan ambiguitas. Perhatikan contoh berikut dengan sedikit mengubah *keyword for* menjadi *fo* (kesalahan penulisan *keyword*):

Contoh 7.19:

```
//Program Error
class Sintaksis
{
    public static void main(String[] args)
    {
        fo ( int i = 0; i < 4; i++ )
        {
            System.out.println("Sepeda baru");
        }
    }
}
```

Pesan kesalahan dari program di atas adalah sebagai berikut:

```
prog.java:6: error: '.class' expected
    fo ( int i = 0; i < 4; i++ )
               ^
prog.java:6: error: illegal start of type
    fo ( int i = 0; i < 4; i++ )
               ^
prog.java:6: error: not a statement
    fo ( int i = 0; i < 4; i++ )
               ^
prog.java:6: error: ';' expected
    fo ( int i = 0; i < 4; i++ )
               ^
4 errors
```

Terdapat 4 kesalahan, akan tetapi semua yang ditunjuk bukan kesalahan sintaks *keyword*, kenapa? 4 kesalahan tersebut tidaklah salah, semua sudah memenuhi aturan Java, untungnya kompiler hanya menunjukkan kesalahan pada baris ke-6 saja, jadi *programmer* tidak beranjak dari baris tersebut. Padahal sudah jelas kesalahannya adalah kata *fo*. Akan tetapi kompiler tidak menunjuk kata *fo* pada pesan kesalahan.

Dari kejadian di atas akhirnya kita menjadi mengerti bahwa kesalahan *keyword* pada suatu pernyataan tertentu seperti *for* tidak akan dibenahi oleh kompiler secara spesifik, akan tetapi dia fokus kepada kesalahan format penulisan pernyataan setelahnya. Contoh di atas sebagai contoh kongkrit

bahwa *fo* tak ditandai, melainkan pernyataan setelahnya yang dianggap tidak memenuhi aturan Java.

## 2. *Runtime Error*

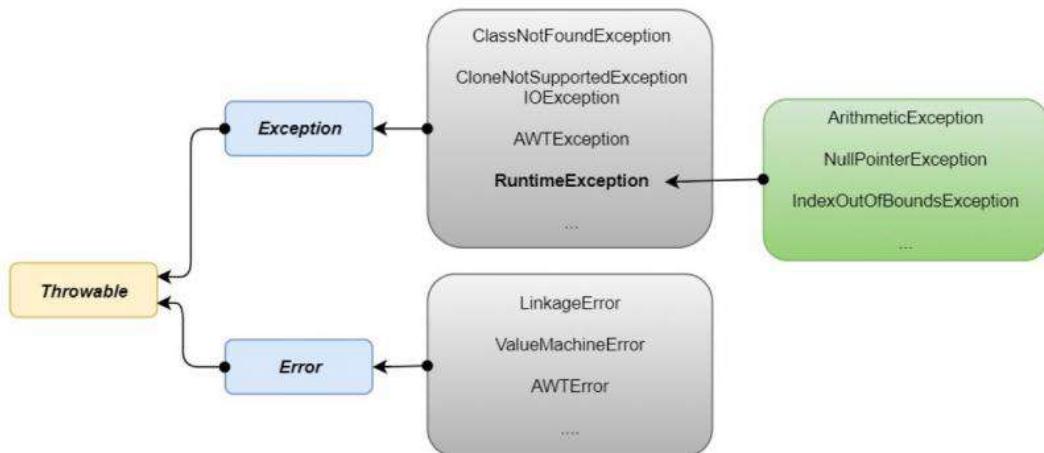
*Runtime Error* adalah kesalahan yang tidak ditampilkan pada saat mengkompilasi program, bahkan pada program yang dikompilasi dengan suksespun tak memberikan pesan kesalahan selama kriteria pemasukan dan proses masih memenuhi. *Programmer* harusnya mampu berpikir dan menduga kesalahan apa yang akan terjadi jika diberikan *input* yang tidak wajar atau tidak sesuai dengan tipe data sehingga terjadi kesalahan operasi. Dalam pemrograman, kondisi ini dipahami sebagai Eksepsi.

## 3. *Logic Error*

Dalam pemrograman komputer, kesalahan adalah *bug* dalam program yang menyebabkannya program beroperasi tidak normal atau *crash* pada kondisi tertentu. Kesalahan menghasilkan keluaran yang tidak sesuai atau tidak, atau yang lainnya. Kesalahan logika dilakukan dalam bahasa yang dikompilasi dan diinterpretasikan. Tidak seperti program dengan kesalahan sintaks atau semantik, program dengan kesalahan adalah program yang *valid* dalam bahasa pemrograman, akan tetapi menghasilkan keluaran yang tidak sesuai yang dinginkan.

## B. TIPE-TIPE EKSEPSI

Pada pemrograman Java, eksepsi adalah sebuah objek dari *sub-class* yang turunkan dari *class Throwable* yang terdapat pada paket *java.lang.object* yang terdapat dalam kumpulan *library* pemrograman Java.



Gambar 7.9  
Percabangan *Throwable* Eksepsi

### 1. *Class Exception* (eksepsi)

*Class* Eksepsi memiliki beberapa *sub-class*, misalnya: *arithmeticException* (kesalahan aritmetika), *NullPointerException* (penggunaan referensi *null* yang tidak valid), *IndexOutOfBoundsException* (beberapa indeks pada *array* di luar batas), dan masih banyak yang lainnya.

Jenis-jenis *error* yang disebutkan di atas dapat diantisipasi dengan menyisipkan pernyataan penanganan eksepsi.

### 2. *Class Error*

Kerusakan yang terjadi pada sistem internal yang bersifat fatal maka dapat dilihat pada *class error*. Beberapa *sub-class* yaitu: *LinkageError*, *ValueMachineError* dan lain-lain.

## C. EKSESPI CHECKED DAN UNCHECKED

### 1. *Checked exception*

*Checked exception* adalah sebuah eksepsi yang diperiksa pada saat kompilasi, artinya pesan kesalahan sudah bisa diprediksi sebelum terjadi, penanganan ini bisa menggunakan *throws*, misalnya pada program memberikan perintah untuk membuka nama file dan dari folder tertentu, akan tetapi pada kenyataan file tersebut tidak ada, maka akan terjadi pesan kesalahan yang harusnya kesalahan tersebut bisa ditangani lebih awal. Contoh *sub-class* pada kategori ini adalah *ClassNotFoundException* dan *CloneNotSupportedException*.

## 2. *Unchecked exception*

*Unchecked exception* adalah eksepsi yang tidak diperiksa, ada dua macam kelas yang ada pada exception ini yaitu *Runtime exception* dan *Error*. *Runtime exception* adalah *exception* yang muncul yang kemunculannya tidak bisa dihindari oleh pemrograman dan sering terjadinya ketika desain program dan atau kesalahan program itu sendiri. *Error* sejatinya bukan *exception*, namun sering muncul di luar kendali pemakai. Beberapa *Runtime Exception* pada class Eksepsi termasuk dalam kelompok ini adalah contohnya: *ArithmetricException* dan *IndexOutOfBoundsException*.

## D. BAGAIMANA EKSEPSI TERJADI?

Jika eksepsi terjadi ketika *programmer* belum mengantisipasi kesalahan yang ada, akibatnya adalah eksekusi program akan dihentikan dan pesan kesalahan yang dihasilkan sistem ditampilkan kepada pengguna. Sebagai contoh lihat pengecualian sistem yang dihasilkan di bawah ini:

Contoh 7.20:

```
//Class Sepeda
public class Sepeda {
    public static void main(String args[]) {
        String sepeda[] = new String[4];

        sepeda[0] = "Sepeda Santai";
        sepeda[1] = "Sepeda Gunung";
        sepeda[2] = "Sepeda Balap";
        sepeda[3] = "Sepeda Ontel";

        System.out.println(sepeda[4]);
    }
}
```

Keluaran:

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException
: 4
at Sepeda.main(Sepeda.java:10)
```

Penjelasan:

- *Java Run Time* menampilkan eksepsi yang disebut dengan *Array Index Out of Bounds*.
- Kesalahan di atas dapat disebut sebagai kesalahan *checked exception*, kenapa? Karena kesalahan tersebut sudah bisa diprediksi akan mengalami kesalahan, *programmer* sudah harus mengetahui bahwa indeks *array* selalu dimulai dari *0*, jika *programmer* menulis *sepeda[4]* sementara pada saat inisiasi hanya memesan *4* tempat yang dimulai dari *0*, maka indeks paling tinggi harusnya *3* atau maksimal *System.out.println(sepeda[3]);*

## E. PENANGANAN EKSEPSI DENGAN *TRY..CATCH*

Kesalahan-kesalahan pada program sangat mungkin terjadi, namun kesalahan tersebut dapat ditangani agar mampu memberikan deskripsi singkat kepada pemakai tentang kesalahan tersebut. Untuk menangani kesalahan tersebut dapat menggunakan *try..catch*.

Sintaks:

```
try {
    // Pernyataan yang mungkin atau dicurigai
    // akan mengalami error
}
catch( TipeException VariabelException)
{
    //blok pernyataan yang dijalankan
    //jika terjadi error
}
```

Penjelasan:

- *try* menyatakan bahwa dalam blok pernyataan dapat terjadi suatu eksepsi dan jika kemungkinan itu terjadi, maka jalankan blok program dalam blok *catch* sesuai dengan tipe eksepsi yang terjadi.
- *Instance variable VariableException* dari *class Exception* dapat digunakan sebagai referensi/data informasi untuk mengetahui penyebab terjadinya eksepsi tersebut.

Contoh 7.21:

```
//Menjebak eksepsi try..catch
public class eksepsiTryCatch {
    public static void main(String args[]) {
        String sepeda[] = new String[4];

        sepeda[0] = "Sepeda Santai";
        sepeda[1] = "Sepeda Gunung";
        sepeda[2] = "Sepeda Balap";
        sepeda[3] = "Sepeda Ontel";

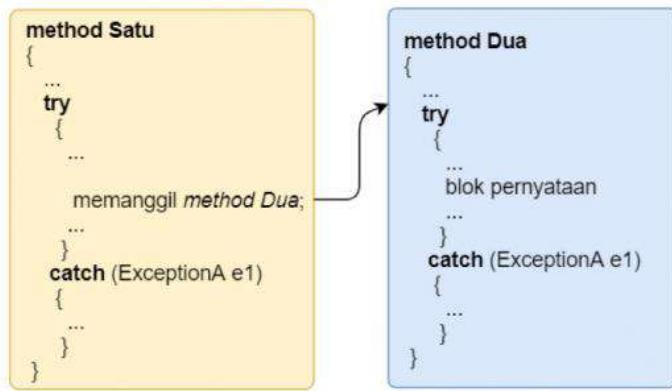
        try {
            System.out.println(sepeda[4]);
        }
        catch (Exception e)
        {
            System.out.println("Awas! Ada kesalahan alamat memori");
        }
    }
}
```

Keluaran:

```
Awas! Ada kesalahan alamat memori
```

## F. PENANGANAN EKSEPSI DENGAN *TRY..CATCH* BERSARANG

Pada dasarnya *try..catch* standar dengan *try..catch* bersarang adalah proses penanganan eksepsi yang sama, hanya saja dalam *try..catch* bersarang terdapat *try..catch* lagi dan tidak melihat apakah itu pada *method* yang sama atau pada *method* yang berbeda.



Gambar 7.10  
*try..catch Bersarang*

Cara kerja dari *try..catch* bersarang adalah mengantisipasi terjadinya *error* pada *try..catch method Satu*, lalu kemudian masuk pada *try..catch method Dua*.

Contoh 7.22:

```

//Menjebak eksepsi try..catch bersarang
public class eksepsiTryCatch {
    public static void main(String args[]) {
        boolean hasil_boolean = false;
        int jumlahSepeda[] = new int[4];
        jumlahSepeda[0] = 3;
        jumlahSepeda[1] = 4;
        jumlahSepeda[2] = 1;
        jumlahSepeda[3] = 3;

        try {
            hasil_boolean
MethodTryCatch(jumlahSepeda[1],jumlahSepeda[3]);
            System.out.println("Hasil Boolean : "+hasil_boolean);
        }
        catch (Exception e)
        {
            System.out.println("Jebak kesalahan dari method utama");
            System.out.println("Awas! Ada kesalahan alamat memori");
        }
    }

    //Method
    public static boolean MethodTryCatch(int a, int b)
    {
        boolean boolean_nilai = false;
        int jumlahTerjual[] = new int[2];
    }
}

```

```

jumlahTerjual[0] = 3;
jumlahTerjual[1] = 4;

try
{
    if (jumlahTerjual[0] > jumlahTerjual[2])
    {
        boolean_nilai = true;
    }

}

catch (Exception e2)
{
    System.out.println("Jebak kesalahan Dari sub-method");
    System.out.println("Awas! terjadi kesalahan");
}

return boolean_nilai;
}
}

```

Keluaran:

```

Jebak kesalahan Dari sub-method
Awas! terjadi kesalahan
Hasil Boolean : false

```

Penjelasan:

Kesalahan terjadi akibat `if (jumlahTerjual[0] > jumlahTerjual[2])`, karena diisi dengan 2, sementara pada saat pemesanan alokasi memori hanya 2, jadi seharusnya indeksnya hanya *0* dan *1*. Karena terjadi kesalahan memasukkan nomor indeks maka jebakan eksepsi berjalan pada `catch (exception e2)`.

## G. PENANGANAN EKSEPSI DENGAN TRY..CATCH BANYAK

Satu *try* tidak menutup hanya pada satu kondisi *catch* saja, akan tetapi sangat mungkin untuk menggunakan *catch* dalam satu *try*. Banyaknya *catch* sangat tergantung banyaknya tipe eksepsi yang akan ditangani dengan cara menspesifikasi tipe eksepsi secara detil.

Sintaksis:

```
try {

    // Pernyataan yang mungkin atau dicurigai
    // akan mengalami error
}
catch( TipeException_x VariabelException)
{
    //blok pernyataan yang dijalankan
    //jika terjadi error
}
catch( TipeException_y VariabelException)
{
    //blok pernyataan yang dijalankan
    //jika terjadi error
}
catch( TipeException_z VariabelException)
{
    //blok pernyataan yang dijalankan
    //jika terjadi error
}
```

Alur kerja dari *catch* banyak contoh di atas adalah pertama eksepsi melempar *try*, kemudian akan ditangkap oleh *catch* yang pertama, kedua, dan seterusnya, tergantung tipe dan masalah yang terjadi.

Contoh 7.23:

```
//Menjebak eksepsi try..catch Banyak
public class eksepsiTryCatch {
    public static void main(String args[]) {
        String sepeda[] = new String[3];
        double pembagian = 0;
        sepeda[0] = "Sepeda Santai";
        sepeda[1] = "Sepeda Gunung";
        sepeda[2] = "Sepeda Balap";

    try {
        System.out.println(sepeda[3]);
    }
}
```

```
pembagian = 10/0;
}
catch (ArrayIndexOutOfBoundsException e)
{
    System.out.println("Eksepsi:
ArrayIndexOutOfBoundsException");
    System.out.println("Ada indeks array melampaui batas");
}
catch (ArithmetricException e)
{
    System.out.println("Eksepsi: ArithmetricException");
    System.out.println("Pembagi tidak boleh = 0");
}
}
```

Keterangan:

- Pada program di atas, yang akan dijebak pertama kali adalah kode program `System.out.println(sepeda[3]);`, akibat dari kesalahan ini akan memicu jebakan *catch* yang pertama dengan keluaran sebagai berikut:

Eksepsi: ArrayIndexOutOfBoundsException  
Ada indeks array melampaui batas

- Jika pada `System.out.println(sepeda[3]);` tidak salah, maka berlanjut memeriksa `pembagian = 10/0;` *catch* kedua akan dipicu untuk dikerjakan, perhatikan keluaran berikut:

```
Sepeda Balap  
Eksepsi: ArithmeticException  
Pembagi tidak boleh = 0
```

## H. PENANGAN EKSEPSI DENGAN *TRY..CATCH..FINALLY*

*Keyword finally* memberikan konsep bahwa apapun yang terjadi, maka blok pernyataan yang ada pada blok *finally* harus tetap dijalankan.

Sintaksis:

```
try {
    // Pernyataan yang mungkin atau dicurigai
    // akan mengalami error
} finally
{
    //blok pernyataan yang dijalankan
    //jika terjadi error
}
```

atau

```
try {
    // Pernyataan yang mungkin atau dicurigai
    // akan mengalami error
} catch( TipeException VariabelException)
{
    //blok pernyataan yang dijalankan
    //jika terjadi error
} finally
{
    //blok pernyataan tetap dijalankan
}
```

Contoh 7.24:

```
//Menjebak eksepsi try..catch..finally
public class eksepsiTryCatchFinally {
    public static void main(String args[]) {
        String sepeda[] = new String[4];

        sepeda[0] = "Sepeda Santai";
        sepeda[1] = "Sepeda Gunung";
        sepeda[2] = "Sepeda Balap";
        sepeda[3] = "Sepeda Ontel";

        try {
            System.out.println(sepeda[4]);
        }
```

```
        }
    catch (Exception e)
    {
        System.out.println("Awas! Ada kesalahan alamat memori");
    }
    finally
    {
        System.out.println("Finally : tetap dikerjakan");
    }
}
```

## I. PENANGANAN EKSEPSI *THROW*

*Keyword throw* digunakan untuk melempar eksepsi dan dibuat secara tersendiri oleh *programmer*, misalnya suatu program untuk memasukkan nilai, namun program tidak memberikan tanda akan terjadi *error*, dikarenakan memang tidak berpotensi terjadi *error*.

Contoh 7.25:

```
public class latihanThrow {
    public static void main(String[] args) {
        String masukan = "latihanThrow";
        try
        {
            if(masukan.equals("latihanThrow"))
            {
                throw new RuntimeException("Throw dijalankan");
            }
            else
            {
                System.out.println(masukan);
            }
        }
        catch (RuntimeException e)
        {
            System.out.println("Eksepsi: RuntimeException");
            System.out.println(e);
        }
    }
}
```

Keluaran:

```
Eksepsi: RuntimeException
java.lang.RuntimeException: Throw dijalankan
```

Dengan pernyataan *throw*, program akan melempar eksepsi untuk kemudian ditampung oleh *catch* di program utama.

### J. PENANGANAN EKSEPSI *THROWS*

Pada dasarnya eksepsi selalu dijebak dengan *catch* pada blok pernyataan dimana eksepsi tersebut dilepaskan. Tanpa *catch*, program tidak bisa dikompilasi. Untuk menangani kasus seperti di atas, program dapat menghindari tanggungjawab untuk menjebak eksepsi tersebut, dengan cara mendeklarasikan nama *class* atau nama *method*. Untuk memberi tahu kompiler bahwa program dapat meneruskan eksepsi, maka diperlukan pernyataan *throws* yang memberi tahu hal tersebut.

Sintaksis:

```
class nama_class throws Exception
{
    ...
}
```

atau

```
public int mana_method () throws Exception
{
    ...
}
```

Contoh 7.26:

```
//Penggunaan Throws
public class PenggunaanThrows {
    void methodX() throws ArithmeticException
{
```

```

        throw new ArithmeticException("Salah perhitungan");
    }

    void methodY() throws ArithmeticException
    {
        methodX();
    }

    void methodZ(){
        try
        {
            methodY();
        }
        catch(ArithmeticException e)
        {
            System.out.println("Eksepsi: ArithmeticException");
        }
    }

    public static void main(String[] args) {
        PenggunaanThrows objek = new PenggunaanThrows ();
        objek.methodZ();
        System.out.println("Selesai");
    }
}

```

Keluaran:

Eksepsi: ArithmeticException  
Selesai



LATIHAN

---

Untuk memperdalam pemahaman Anda mengenai materi di atas, kerjakanlah latihan berikut!

- 1) Pada *Compilation Error* terdapat 3 kesalahan, sebutkan dan jelaskan secara singkat?
- 2) Apa yang dimaksud dengan *runtime error*?
- 3) Apa yang dimaksud dengan *logic error*?
- 4) Carilah 3 contoh program dari internet tentang *try..catch*, setiap contoh tidak boleh diambil dari laman yang sama. Kemudian kompilasi dan tampilkan hasilnya.

### Petunjuk Jawaban Latihan

- 1) Pada *compilation error* terdapat 3 kesalahan, yaitu:
  - Kesalahan sintaksis: kesalahan penulisan program yang tidak sesuai dengan aturan Bahasa pemrograman Java. Sebagai contoh, bahasa pemrograman Java pada aturannya menyatakan bahwa setiap pernyataan harus diakhiri dengan titik koma (,).
  - Kesalahan Semantik: kesalahan makna atau logika, misalnya kesalahan dalam menginisiasi variabel.
  - Kesalahan *cascading*: kesalahan yang tidak menunjuk kesalahan terjadi, akan tetapi menunjuk yang lain, yang terkadang memuat *programmer* bingung.
- 2) *Runtime error* adalah kesalahan yang tidak ditampilkan pada saat kompilasi program, tapi kesalahan yang akan muncul ketika program sudah dijalankan. Misalnya salah mengantisipasi masukan dari *keyboard*, harusnya pengguna menginput angka, tapi pengguna malah menginput huruf.
- 3) *Logic error* adalah *bug* dalam program sehingga program tidak berjalan normal pada kondisi tertentu.
- 4) Contoh try..catch:
  - Laman: [https://www.w3schools.com/java/java\\_try\\_catch.asp](https://www.w3schools.com/java/java_try_catch.asp)

```
public class MyClass {  
    public static void main(String[ ] args) {  
        try {  
            int[] myNumbers = {1, 2, 3};  
            System.out.println(myNumbers[10]);  
        } catch (Exception e) {  
            System.out.println("Something went wrong.");  
        }  
    }  
}
```

Keluaran:

```
Something went wrong
```

- Laman: <https://www.javatpoint.com/try-catch-block>

```
public class TryCatchExample3 {  
  
    public static void main(String[] args) {  
        try  
        {  
            int data=50/0; //may throw exception  
            // if exception occurs,  
            //the remaining statement will not execute  
            System.out.println("rest of the code");  
        }  
        // handling the exception  
        catch(ArithmetricException e)  
        {  
            System.out.println(e);  
        }  
    }  
}
```

Keluaran:

```
java.lang.ArithmetricException: / by zero
```

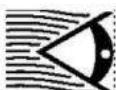
- Laman: <https://www.tutorialspoint.com/try-catch-throw-and-throws-in-java>.

```
import java.io.*;  
public class Demo {
```

```
public static void main(String args[]) {  
    try  
    {  
        int a[] = new int[5];  
        System.out.println("Access element eighth :" + a[7]);  
    }  
    catch (ArrayIndexOutOfBoundsException e)  
    {  
        System.out.println("Exception thrown :" + e);  
    }  
    System.out.println("Out of the block");  
}  
}
```

Keluaran:

```
Exception thrown :java.lang.ArrayIndexOutOfBoundsException: 7  
Out of the block
```



RANGKUMAN

Eksepsi adalah peristiwa yang menyebabkan program menjadi tidak normal, pemicu dari eksepsi bisa jadi disebabkan masukan data yang tidak valid, file yang disebutkan dalam program tidak ada, atau mungkin karena indeks pada *array* terlalu besar, kejadian seperti ini terjadi pada saat *runtime error*.

Ada tiga kategori *error* dalam Bahasa Pemrograman Java, yaitu: *Compilation Error*, *Runtime Error*, dan *Logic Error*. *Compilation Error* adalah *error* yang terjadi pada saat program dikompilasi, mungkin *compiler* Java menemukan Kesalahan Sintaksis, Kesalahan Semantik, atau Kesalahan *Cascading*. *Runtime Error* adalah kesalahan yang tidak ditampilkan pada saat mengkompilasi program, bahkan pada program yang dikompilasi dengan suksespun tak memberikan pesan kesalahan selama kriteria pemasukan dan proses masih memenuhi. Dalam pemrograman komputer, kesalahan adalah *bug* dalam program yang menyebabkan operasi tidak berjalan normal atau *crash* pada situasi tertentu. Kesalahan logika adalah juga kesalahan *bug* yang bisa jadi menghasilkan keluaran yang tidak sesuai yang diinginkan.

Ada dua Tipe Eksepsi, yaitu: *Class Exception* dan *Class Error*. *Class Eksepsi* memiliki beberapa *sub-class*, misalnya: *arithmeticException* (kesalahan aritmetika), *NullPointerException* (penggunaan referensi *null* yang tidak valid), *IndexOutOfBoundsException* (beberapa indeks pada array diluar batas), dan masih banyak yang lainnya, sedangkan *Class Error* adalah kerusakan yang terjadi pada sistem internal yang bersifat fatal maka dapat dilihat pada *class error*. Beberapa *sub-class* yaitu: *LinkageError*, *ValueMachineError* dan lain-lain.

Penanganan eksepsi dapat ditangani dengan beberapa cara, yaitu dengan *try..catch*, *try..catch* bersarang, *try..catch..finally*, *throw* dan *throws*.



### TES FORMATIF 3

---

Pilihlah satu jawaban yang paling tepat!

- 1) Penanganan eksepsi berjalan pada saat *runtime error*, maksudnya adalah....
  - A. Agar proses pada aplikasi dapat berjalan normal seolah tidak ada masalah
  - B. Agar aplikasi dapat dikompilasi dengan cepat
  - C. Agar program tak perlu dikompilasi lagi
  - D. Agar program bisa ditangani dan memunculkan pesan kesalahan yang membuat aplikasi berhenti
- 2) Tiga kategori *Error* dalam bahasa pemrograman adalah....
  - A. *Compilation Error*, *Error Handling*, dan *Logic Error*
  - B. *Compilation Error*, *Bugs*, dan *Logic Error*
  - C. *Compilation Error*, *Runtime Error*, dan *Statement Error*
  - D. *Compilation Error*, *Runtime Error*, dan *Logic Error*
- 3) Yang bukan termasuk *keyword* penanganan eksepsi adalah....
  - A. *finally*
  - B. *throws*
  - C. *try*
  - D. *case*
- 4) Kesalahan Sintaksis termasuk dalam kesalahan kategori....
  - A. *Runtime Error*
  - B. *Compilation Error*

- C. *Bugs*
- D. *Logic Error*

5) Perhatikan contoh program berikut:

prog.java:x: error: variable a might not have been initialized

c = a \* b;  
  ^

Kesalahan di atas adalah kesalahan....

- A. Kesalahan Sintaksis
- B. Kesalahan Semantik
- C. Kesalahan *Cascade*
- D. Semuanya salah

6) Kesalahan yang tidak ditampilkan pada saat mengkompilasi program disebut juga dengan kesalahan....

- A. *Runtime Error*
- B. *Logic Error*
- C. *Sintaksis Error*
- D. *Compiler Error*

7) Eksepsi yang diperiksa pada saat kompilasi, artinya pesan kesalahan sudah bisa diprediksi sebelum terjadi, penanganan ini bisa menggunakan *throws* disebut dengan....

- A. *Exception Handling*
- B. *Checked exception*
- C. *Unchecked exception*
- D. *Runtime Error*

8) Eksepsi yang tidak diperiksa, ada dua macam *class* yang ada pada *exception* ini yaitu *Runtime exception* dan *Error* disebut dengan....

- A. *Exception Handling*
- B. *Checked exception*
- C. *Unchecked exception*
- D. *Runtime Error*

9) Eksepsi handling dengan konsep bahwa apapun yang terjadi, kode program tetap dijalankan pada blok *keyword*-nya adalah....

- A. *try..catch*
- B. *try..catch banyak*
- C. *try..catch..finally*
- D. *throws*

- 10) Ekspresi yang dapat ditangani dengan membangun *method* atau *class* adalah....
- A. *try..catch*
  - B. *try..catch banyak*
  - C. *throw*
  - D. *throws*

Cocokkanlah jawaban Anda dengan Kunci Jawaban Tes Formatif 3 yang terdapat di bagian akhir modul ini. Hitunglah jawaban yang benar. Kemudian, gunakan rumus berikut untuk mengetahui tingkat penguasaan Anda terhadap materi Kegiatan Belajar 3.

$$\text{Tingkat penguasaan} = \frac{\text{Jumlah Jawaban yang Benar}}{\text{Jumlah Soal}} \times 100\%$$

Arti tingkat penguasaan: 90 - 100% = baik sekali

80 - 89% = baik

70 - 79% = cukup

< 70% = kurang

Apabila mencapai tingkat penguasaan 80% atau lebih, Anda dapat meneruskan dengan modul selanjutnya. **Bagus!** Jika masih di bawah 80%, Anda harus mengulangi materi Kegiatan Belajar 3, terutama bagian yang belum dikuasai.

## Kunci Jawaban Tes Formatif

### Tes Formatif 1

- 1) B
- 2) A
- 3) A
- 4) D
- 5) B
- 6) A
- 7) C
- 8) C
- 9) A
- 10) A

### Tes Formatif 2

- 1) B
- 2) C
- 3) D
- 4) B
- 5) C
- 6) A
- 7) B
- 8) B
- 9) B
- 10) A

### Tes Formatif 3

- 1) A
- 2) D
- 3) D
- 4) B
- 5) A
- 6) D
- 7) B
- 8) C
- 9) C
- 10) D

## Daftar Pustaka

- Eckel, B. (2000). *Thinking in Java (2<sup>nd</sup>)*. New Jersey: Prentice Hall.
- Schildt, H. (2007). *The Complete Reference Java (Seventh Edition)*. New York: Mc Graw Hill.
- Flask, R. *Java for Beginners (2<sup>nd</sup>)*.
- Eck, D.J. (2006). *Introduction to programming using java (Version 5.0)*. Departement of Mathematic and Computer Science. Geneva, New York 14456.
- Horstmann, C.S. & Cornell, G. (2013). *Core java volume I fundamental(Ninth Edition)*. New York: Prentice Hall.
- Redko, A. *Adnvanve java preparing you for java mastery*.
- <https://www.jdoodle.com/online-java-compiler>
- <https://jagoankode.blogspot.com/2017/11/contoh-program-throw-dan-throws-pada-java.html>
- <http://www.bahasajava.com/2018/03/mengenal-tipe-exception-pada-java.html>
- <https://halimi1010.wordpress.com/category/j2se/pbo-1/exception-handling/>

# String, String Buffer, dan Math

Kani, M.Kom.



## PENDAHULUAN

---

Dalam pemrograman komputer *string* adalah sederet simbol, dengan menggunakan tipe data *string*. Tipe data string tersebut diperuntukkan untuk menyimpan sekumpulan karakter. Pada biasanya *String* dianggap sebagai tipe data yang sering diimplementasikan sebagai struktur data *array* dari *byte* yang menyimpan urutan *array*, misalnya kata “Java” maka dalam *array* adalah {‘J’, ‘a’, ‘v’, ‘a’}.

Setelah mempelajari modul ini, diharapkan mahasiswa dapat:

1. menjelaskan pengertian *String*;
2. menjelaskan *class string*;
3. menjelaskan *constructor class string*;
4. menyebutkan dan menjelaskan *method-method class string*;
5. menyebutkan cara menggunakan *method-method class string*;
6. melatih contoh-contoh *method-method class string*;
7. menjelaskan *class stringBuffer*;
8. menjelaskan *constructor class stringBuffer*;
9. menyebutkan dan menjelaskan *method-method class stringBuffer*;
10. menyebutkan cara menggunakan *method-method class stringBuffer*;
11. melatih contoh-contoh *method-method class stringBuffer*;
12. menjelaskan *class Math*;
13. menjelaskan *constructor class Math*;
14. menyebutkan dan menjelaskan *method-method class Math*;
15. menyebutkan cara menggunakan *method-method class Math*;
16. melatih contoh-contoh *method-method class Math*.

## KEGIATAN BELAJAR 1

### *String*

*S*tring pada bahasa pemrograman Java merupakan bagian tersendiri. Pada Java, *string* disajikan dengan *class* yang bersifat “final” yang di dalamnya terdapat beberapa utilitas yang dapat digunakan untuk melakukan pengolahan variabel bertipe data “String” tersebut. *Class String* memiliki banyak *constructor* yang memungkinkan membuat objek *String* dan menginisialisasi nilainya dengan menggunakan berbagai macam sumber data yang berbeda.

#### A. CLASS STRING

*String* adalah *class* yang diturunkan dari *java.lang* yang menangani deretan karakter. *Class String* mempunyai sejumlah *method* yang berguna untuk memanipulasi *String* itu sendiri. Misalnya: mengubah huruf kecil menjadi huruf kapital, menggabung karakter atau *String*, memenggal *String* dari depan, tengah atau belakang.

#### B. CONSTRUCTOR CLASS STRING

Pada *class string* terdapat banyak *constructor* yang membangunnya, beberapa *constructor* bisa dilihat sebagai berikut:

| No | Constructor                                                              | Fungsi/Keterangan                                                                                                                    |
|----|--------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------|
| 1  | <i>String()</i>                                                          | Membuat objek <i>String</i> dengan <i>string</i> kosong                                                                              |
| 2  | <i>String(byte[] bytes)</i>                                              | Membuat objek <i>String</i> yang berisi <i>array</i> dari tipe <i>byte</i> dan menggunakan <i>default charset</i>                    |
| 3  | <i>String(byte [] bytes, int offset, int length)</i>                     | Membuat objek <i>String</i> yang berisi suatu <i>sub-array</i> dari tipe <i>byte</i> dan menggunakan <i>default charset</i>          |
| 4  | <i>String(byte [] bytes, int offset, int length, String charsetName)</i> | Membuat objek <i>String</i> yang berisi suatu <i>sub-array</i> dari tipe <i>byte</i> dan menggunakan <i>default charset</i> tertentu |

| No | Constructor                                         | Fungsi/Keterangan                                                                                                                     |
|----|-----------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------|
| 5  | <i>String(byte [] bytes, String charsetName)</i>    | Membuat objek <i>String</i> yang berisi suatu <i>array</i> dari tipe <i>byte</i> dan menggunakan <i>default charset</i> tertentu      |
| 6  | <i>String(char [] value)</i>                        | Membuat objek <i>String</i> yang mempresentasikan deretan karakter yang telah ada pada argumen karakter yang berbentuk <i>array</i> . |
| 7  | <i>String(char [] value, int offset, int count)</i> | Membuat objek <i>String</i> yang berisi suatu karakter dari suatu <i>sub-array</i> dari argumen karakter yang bentuk <i>array</i>     |
| 8  | <i>String(String original)</i>                      | Membuat objek <i>String</i> yang mempresentasikan deretan karakter sebagai suatu argumen, <i>String</i> merupakan salinan             |
| 9  | <i>String(StringBuffer)</i>                         | Membuat objek <i>String</i> baru yang berisi deretan karakter yang telah ada dalam argumen <i>string buffer</i>                       |

Contoh 8.1:

```
//Program String
class contohString
{
    public static void main(String[] args)
    {
        String string1 = "Halo Java";
        String string2 = new String("Gampang belajar Java");

        System.out.println(string1+, "+string2);
    }
}
```

Keluaran:

```
Halo Java, Gampang belajar Java
```

### C. METHOD-METHOD STRING

Beberapa *method* akan dibahas pada sub-bab ini untuk menambah pengetahuan dari *method* yang ada.

#### 1. Method mengetahui Panjang String

Method aksesor yang digunakan untuk mengetahui panjang karakter dalam objek *string* adalah *length()*.

Sintaksis:

```
public int length();
```

Contoh 8.2:

```
//Program mengetahui panjang string
class panjangString
{
    public static void main(String[] args)
    {
        String string1 = "Halo Java, gampang belajar Java";
        int panjangSTR = string1.length();
        System.out.println("Panjang String : "+panjangSTR);
    }
}
```

Keluaran:

```
Panjang String : 31
```

## 2. *Method untuk Menggabung String*

Class *String* menyediakan sebuah *method* untuk menggabung *string* dengan *string* yang lain, adapun *method* tersebut adalah *method concat()*.

Sintaksis:

```
public String concat(String anotherString)
```

Penjelasan:

- *stringX* dan *stringY* adalah objek *string*.
- *concat()* adalah *method* untuk menggabung *string*.

Contoh 8.3:

```
//Program menggabung string
class gabungString
{
    public static void main(String[] args)
    {
        String stringX = "Halo Java, ";
        String stringY = "gampang belajar Java";

        System.out.println(stringX);
        System.out.println(stringY);

        stringX = stringX.concat(stringY);
        System.out.println(stringX);
    }
}
```

Keluaran:

```
Halo Java,  
gampang belajar Java  
Halo Java, gampang belajar Java
```

### 3. *Method charAt*

*Method charAt* digunakan untuk mengembalikan karakter pada indeks yang diberikan. Nomor indeks dimulai dari 0 dan pergi ke n-1, di mana n adalah panjang *string*. Ini mengembalikan *StringIndexOutOfBoundsException* jika nomor indeks yang diberikan lebih besar dari atau sama dengan panjang *string* atau angka negatif.

Sintaksis:

```
Public char charAt(int indeks)
```

Keterangan:

- indeks : adalah indeks karakter yang dikembalikan.

Contoh 8.4:

```
//Program mengembalikan karakter pada  
//indeks tertentu dari String  
class gabungString  
{  
    public static void main(String[] args)  
    {  
        String stringX = "Halo Java, gampang belajar Java";  
        char kar1 = stringX.charAt(5);  
        char kar2 = stringX.charAt(6);  
        char kar3 = stringX.charAt(7);  
        char kar4 = stringX.charAt(8);  
    }  
}
```

```
        System.out.println("Karakter ke-5 :" +kar1);
        System.out.println("Karakter ke-6 :" +kar2);
        System.out.println("Karakter ke-7 :" +kar3);
        System.out.println("Karakter ke-8 :" +kar4);
    }
}
```

Keluaran:

```
Karakter ke-5 :J
Karakter ke-6 :a
Karakter ke-7 :v
Karakter ke-8 :a
```

#### 4. *Method valueOf*

*Method valueOf* adalah *method* untuk mengubah berbagai jenis nilai menjadi *string*. Dengan bantuan *method string valueOf*, Anda dapat mengubah *int* ke *string*, *long* ke *string*, *boolean* ke *string*, *char* ke *string*, *float* ke *string*, *double* ke *string*, *objek* ke *string* dan *array char* ke *string*.

Sintaksis:

```
static String valueOf(boolean b)
```

atau

```
static String valueOf(char c)
```

atau

```
static String valueOf(char[] data)
```

atau

```
static String valueOf(char[] data, int offset, int count)
```

atau

```
static String valueOf(double d)
```

atau

```
static String valueOf(float f)
```

atau

```
static String valueOf(int i)
```

atau

```
static String valueOf(long l)
```

atau

```
static String valueOf(Object obj)
```

Contoh 8.5:

```
//Program penggunaan method valueOf
public class methodValueOf {

    public static void main(String args[]) {
        double d = 102939939.939;
        boolean b = true;
        long l = 1232874;
        char[] array = {'a', 'b', 'c', 'd', 'e', 'f','g' };

        String s = "Nilai balik ";
        System.out.println(s+"valueOf(double) : " + String.valueOf(d));
        System.out.println(s+"valueOf(boolean) : " + String.valueOf(b));
        System.out.println(s+"valueOf(long) : " + String.valueOf(l));
        System.out.println(s+"valueOf(char) : " + String.valueOf(array));
    }
}
```

Keluaran:

```
Nilai balik valueOf(double) : 1.02939939939E8
Nilai balik valueOf(boolean) : true
Nilai balik valueOf(long) : 1232874
Nilai balik valueOf(char) : abcdefg
```

### 5. *Method copyValueOf*

Fungsi inti dari dari *method copyValueOf* adalah menduplikasi isi dari sumber data. Ada dua bentuk dari *method* ini yaitu: 1) Menduplikasi semua isi sumber, 2) Menduplikasi sumber dengan menentukan deretan paling awal dan sejumlah karakter yang diambil.

Sintaksis menduplikasi semua sumber:

```
public static String copyValueOf(char[] data)
```

Sintaksis menduplikasi sumber dengan menentukan titik awal dan jumlah karakter yang diambil:

```
public static String copyValueOf(char[] data, int offset, int jumlah)
```

Contoh 8.6:

```
//Program penggunaan metod copyValueOf
public class programCopyValueOf {

    public static void main(String args[]) {
        char[] SourceStr1 = {'B', 'e', 'l', 'a', 'j', 'a', 'r'};
        char[] SourceStr2 = {'J', 'a', 'v', 'a'};
        String DestStr3 = "";
        String DestStr4 = DestStr3;
        String DestStr5 = DestStr3;
        //Copy semua sumber
        DestStr3 = DestStr3.copyValueOf(SourceStr1)+" "+
                    DestStr3.copyValueOf(SourceStr2);
        System.out.println("Copy semua \t: " + DestStr3);

        //Copy sebagian sumber
        DestStr4 = DestStr4.copyValueOf(SourceStr1,3,4);
        System.out.println("Copy bagian \t: " + DestStr4);
    }
}
```

Keluaran:

```
Copy semua      : Belajar Java  
Copy bagian    : ajar
```

### 6. *Method compareTo*

*Method compareTo* adalah membandingkan *string* dengan *string* lain.

Sintaksis:

```
int compareTo(Object String)
```

Keterangan:

- Jika nilai balikan *int* adalah *0* maka pembanding dan terbanding sama.
- Jika nilai balikan *int* tidak sama dengan *0* maka pembanding dan terbanding tidak sama
- Jika nilai balikan *int* positif maka pembanding karakternya lebih banyak.
- Jika nilai balikan *int* negatif maka pembanding karakternya lebih sedikit.

Contoh 8.7:

```
//Program Compare String  
public class CompareString {  
  
    public static void main(String args[]) {  
        String comStr1 = "Belajar Java";  
        String comStr2 = new String("Belajar Java");  
        String comStr3 = new String("Belajar Java harus rajin  
praktek");  
  
        //Compare comStr1 vs comStr2  
        int hasilCompare = comStr1.compareTo(comStr2);  
        System.out.println("Compare 1:");  
  
        if(hasilCompare == 0)  
            System.out.println("comStr1 sama dengan comStr2 :  
"+hasilCompare);  
        else  
            System.out.println("comStr1 tidak sama dengan comStr2 "+
```

```

        hasilCompare);

//Compare comStr2 vs comStr3
hasilCompare = comStr2.compareTo(comStr3);
System.out.println("Compare 2:");

if(hasilCompare == 0)
    System.out.println("comStr2 sama dengan comStr3 : "
"+hasilCompare);
else
    System.out.println("comStr2 tidak sama dengan comStr3 : "
"+
                    hasilCompare);

}
}

```

Keluaran:

```

Compare 1:
comStr1 sama dengan comStr2 : 0
Compare 2:
comStr2 tidak sama dengan comStr3 : -20

```

## 7. Method *startsWith*

*Method startsWith* adalah *method* memeriksa apakah *string* ini dimulai dengan awalan yang diberikan. Ini mengembalikan *true* jika *string* ini dimulai dengan awalan yang diberikan, dan jika tidak maka mengembalikan *false*. Lihat ilustrasi di bawah ini:

Ilustrasi 1: untuk *startsWith(String prefix)*:

```

Jika a = "Belajar Java dan JavaScript" dan
b = startsWith("Belajar Java")
maka nilainya true.
karena "Belajar Java" = "Belajar Java"
Jika a = "Belajar Java dan JavaScript" dan
b = startsWith("Belajar Java &")
maka nilainya false.

```

karena "Belajar Java d" <> " Belajar Java &""

Ilustrasi 2: untuk **startsWith(String prefix,int str\_pos)**:

Jika a = "Belajar Java dan JavaScript"  
b = **startsWith("Java",8)**  
maka nilainya *true*.  
Karena "Java" = "Java"  
Jika a = "Belajar JavaScript" dan  
b = **startsWith("Java",7)**  
maka nilainya *false*.  
(" Jav" <> "Java")

Contoh 8.8:

```
//Program penggunaan method StarsWith
public class ProgramMethodStartWith {
    public static void main(String args[]) {
        String ComStringBegin = new String("Belajar Java dan
JavaScript");
        boolean hasilBoolean;

        //Menggunakan startsWith(s)
        hasilBoolean = ComStringBegin.startsWith("Belajar Java");
        System.out.println("Hasil = " + hasilBoolean );

        hasilBoolean = ComStringBegin.startsWith( "Belajar Java &
);
        System.out.println("Hasil = " + hasilBoolean );

        //Menggunakan startsWith(s,i)
        hasilBoolean = ComStringBegin.startsWith("Java",8);
        System.out.println("Hasil = " + hasilBoolean );

        hasilBoolean = ComStringBegin.startsWith("Java",7);
        System.out.println("Hasil = " + hasilBoolean );
    }
}
```



```

        System.out.println("equalStr1 tidak sama dengan
equalStr2 " +
                           hasilequals);

        //Compare comStr2 vs comStr3
        hasilequals = equalStr2.equals(equalStr3);
        System.out.println("Equals 2:");

        if(hasilequals == true)
            System.out.println("equalStr2 sama dengan equalStr3 : "+
                               hasilequals);
        else
            System.out.println("equalStr2 tidak sama dengan
equalStr3 : "+
                           hasilequals);

    }
}

```

Keluaran:

```

Equals 1:
equalStr1 sama dengan equalStr2 : true

Equals 2:
equalStr2 tidak sama dengan equalStr3 : false

```

## 9. *Method substring*

*Method substring* digunakan untuk mendapatkan substring dari String tertentu. Ada dua varian dari *method* ini, yaitu *substring(int indeksAwal)* dan *substring(int indeksAwal, indeksAkhir)*.

Sintaksis:

```
public String substring(int beginIndex)
```

atau

```
public String substring(int IndeksAwal, int IndeksAkhir)
```

Untuk `substring(int indeksAwal)` adalah mengembalikan `substring` mulai dari indeks yang ditentukan (`indeksAwal`) hingga akhir `string`. Misalnya: `str="Belajar Java dan JavaScript"; str.substring(,8)` maka hasilnya "Java dan JavaScript". *Method* ini melempar eksepsi `IndexOutOfBoundsException` jika `indeksAwalnya < 0` dan `> Panjang String`.

Untuk `substring(int indeksAwal, int indekAkhir)` adalah mengembalikan `substring` mulai dari indeks yang diberikan (`indeksAwal`) hingga indeks yang ditentukan (`indekAkhir`). Misalnya: `str="Belajar Java dan JavaScript"; str.substring(8,4)` maka hasilnya "Java". *Method* ini juga melempar eksepsi `IndexOutOfBoundsException` jika `indeksAwalnya < 0` atau `indeksAwal > indeksAkhir` atau `indeksAkhir >` dari Panjang `String`.

Contoh 8.10:

```
//Program substring
public class ContohSubstring {
    public static void main(String args[]) {

        String subStr =
            new String("Belajar Java dan JavaScript menyenangkan
sekali");

        System.out.println("Substring mulai dari indeks 8:");
        System.out.println(subStr.substring(8));

        System.out.println("Substring mulai dari indeks 8 hingga
indeks 28:");
        System.out.println(subStr.substring(8, 28));

    }
}
```

Keluaran:

```
Substring mulai dari indeks 8:
Java dan JavaScript menyenangkan sekali
Substring mulai dari indeks 8 hingga indeks 28:
Java dan JavaScript
```

### 10. *Method toLowerCase*

*Method toLowerCase* adalah *method* yang digunakan untuk merubah *string* menjadi huruf kecil.

Sintaksis:

```
public String toLowerCase()
```

Contoh 8.11:

```
//Program mengubah huruf menjadi huruf kecil
public class lowerCase {

    public static void main(String args[]) {
        String Str = new String("Belajar Java dan JavaScript");

        System.out.print("Hasil : ");
        System.out.println(Str.toLowerCase());
    }
}
```

Keluaran:

```
Hasil : belajar java dan javascript
```

### 11. *Method toUpperCase*

*Method toUpperCase* adalah *method* yang digunakan untuk merubah *string* menjadi huruf kapital.

Sintaksis:

```
public String toUpperCase()
```

Contoh 8.12:

```
//Program mengubah huruf menjadi huruf kapital
public class upperCase {

    public static void main(String args[]) {
        String Str = new String("Belajar Java dan JavaScript");

        System.out.print("Hasil : ");
        System.out.println(Str.toUpperCase());
    }
}
```

Keluaran:

```
Hasil : BELAJAR JAVA DAN JAVASCRIPT
```

## 12. *Method trim*

*Method trim* adalah *method* yang digunakan untuk menghilangkan spasi yang terkandung dalam *string* di awal dan dibelakang.

Sintaksis:

```
public String trim()
```

Contoh 8.13:

```
//Program mengubah trim
public class ProgramTrim{
    public static void main(String args[]){
```

```

        String str = new String("    Belajar Java dan Java Script
");
        System.out.println("String sebelum trim : "+str);
        System.out.println("String setelah trim : "+str.trim());

    }
}

```

Keluaran:

```

String sebelum trim :    Belajar Java dan Java Script
String setelah trim : Belajar Java dan Java Script

```

### 13. Method *isEmpty*

*Method isEmpty* adalah memeriksa apakah suatu *String* kosong atau tidak. Metode ini mengembalikan nilai *true* jika *string* yang diberikan kosong, dan jika tidak akan mengembalikan nilai *false*. Dengan kata lain Anda dapat mengatakan bahwa metode ini mengembalikan nilai *true* jika panjang *string* adalah 0.

Sintaksis:

```
public boolean isEmpty()
```

Contoh 8.14:

```

//Program isEmpty
public class contohisEmpty
{
    public static void main(String args[])
    {
        //string empty-kosong
        String strX="";
        //string tidak empty
        String strY="Belajar Java dan JavaScript";
    }
}

```

```
//prints true  
System.out.print("Hasil : ");  
System.out.println(strX.isEmpty());  
  
//prints false  
System.out.print("Hasil : ");  
System.out.println(strY.isEmpty());  
}  
}
```

Keluaran:

```
Hasil : true
```

```
Hasil : false
```

#### 14. Method *replace*, *replaceFirst*, dan *replaceAll*

Adapun sintaksis dari tiga *method* adalah sebagai berikut:

Sintaksis *replace*:

```
public String replace(char oldChar, char newChar)
```

Keterangan:

- *oldChar* : karakter yang hendak dicari dan diganti.
- *newChar* : karakter pengganti yang akan menimpa *oldChar*.

Sintaksis *replaceFirst*:

```
public String replaceFirst(String regex, String replacement)
```

Keterangan:

- *regex* : merupakan regular *expression string* yang ingin diganti.

- `replacement` : merupakan *substituted string* yang digunakan sebagai pengganti

Sintaksis `replaceAll`:

```
public String replaceAll(String regex, String replacement)
```

Keterangan:

- `regex` : ekspresi reguler untuk *string* yang akan dicocokkan dan akan diganti.
- `replacement` : *String* yang akan menggantikan ekspresi yang ditemukan.

*Method replace* berfungsi mengganti bagian karakter yang ditentukan tapi dengan syarat hanya boleh 1 karakter, misalnya: `str.replace("B", "J")`, maka dianggap salah jika menggunakan 2 karakter seperti: `str.replace("Ba", "Ja")`; adalah sintaks yang salah.

*Method replaceFirst* akan menggantikan *substring* pertama selama ekspresi *string* cocok, misalnya: misalnya: `str.replaceFirst("Belajar", "Mengajar")`, kata “Belajar” akan diganti dengan “Mengajar” selama kata tersebut memenuhi ekspresi atau ditemukan.

*Method replaceAll* hampir sama dengan *method replace* hanya saja yang digunakan adalah berupa *string*.

Contoh 8.15. *method replace*

```
//Program Method Replace
public class contoMethodReplace {

    public static void main(String args[]) {

        String StrRplc = new String("Belajar Java dan JavaScript");

        System.out.println("Sebelum Replace : "+StrRplc);
        System.out.print("Sesudah Replace : " );
        System.out.println(Str.replace('v', 'y'));

        System.out.println("Sebelum Replace : "+StrRplc);
        System.out.print("Sesudah Replace : " );
        System.out.println(Str.replace('S', 's'));
    }
}
```

Keluaran:

```
Sebelum Replace : Belajar Java dan JavaScript
Sesudah Replace : Belajar Jaya dan JayaScript

Sebelum Replace : Belajar Java dan JavaScript
Sesudah Replace : Belajar Java dan Javascript
```

Contoh 8.16. *method replaceFirst*

```
//Program Method ReplaceFirst
public class contoMethodReplaceFirst {

    public static void main(String args[]) {

        String StrRplcFrst =
            new     String("Belajar      Java      dan      JavaScript
menyenangkan");

        System.out.println("Sebelum replaceFirst : "+StrRplcFrst);
        System.out.print("Sesudah replaceFirst : " );

        System.out.println(StrRplcFrst.replaceFirst("(.*JavaScript(.*)"
",
                    "Belajar C++"));

        System.out.println("Sebelum replaceFirst : "+StrRplcFrst);
        System.out.print("Sesudah replaceFirst : " );
        System.out.println(StrRplcFrst.replaceFirst("JavaS",
"Powers"));

    }
}
```

Keluaran:

Sebelum replaceFirst : Belajar Java dan JavaScript menyenangkan

Sesudah replaceFirst : Belajar C++

Sebelum replaceFirst : Belajar Java dan JavaScript menyenangkan

Sesudah replaceFirst : Belajar Java dan PowerScript menyenangkan

Contoh 8.17. *method replaceAll*

```
//Program Method ReplaceAll
public class contoMethodReplaceAll {

    public static void main(String args[]) {

        String StrRplcAll =
            new      String("Belajar      Java      dan      JavaScript
menyenangkan");

        System.out.println("Sebelum replaceAll : "+StrRplcAll);
        System.out.print("Sesudah replaceAll : " );

        System.out.println(StrRplcAll.replaceAll("(.*)JavaScript(.*)",
            "Belajar C/C++"));

    }
}
```

Keluaran:

Sebelum replaceAll : Belajar Java dan JavaScript menyenangkan

Sesudah replaceAll : Belajar C/C++



## LATIHAN

---

Untuk memperdalam pemahaman Anda mengenai materi di atas, kerjakanlah latihan berikut!

- 1) Sebutkan dan jelaskan 3 *method string* yang anda ketahui, dan kemudian berikan contoh program penggunaannya, boleh mengambil contoh dari internet pada laman-laman yang anda ketahui.

### Petunjuk Jawaban Latihan

- 1) Tiga 3 contoh *method string* untuk menjawab pertanyaan sebagai berikut:
  - a. *Method length* berfungsi untuk mendapatkan panjang *string*. *Method* ini mengembalikan nilai jumlah karakter dalam parameter *method*. Contoh:

```
/*
 * This program demonstrates
 * length String methods.
 */
public class StringLengthDemo
{
    public static void main(String[] args)
    {
        String message; // To hold a string
        int n;           // To hold length of String

        message = "The Java Tutorials";
        n = message.length();
        System.out.println("The length of message : " + n);
    }
}
```

Sumber: <http://www.beginwithjava.com/java/inputoutput/string-method.html>.

Keluaran:

```
The length of message : 18
```

- b. *Method compareTo* adalah *method* yang digunakan untuk membandingkan *string* dengan *string* yang lain.

Contoh:

```
public class JavaExample {  
    public static void main(String args[]) {  
        String str1 = "Cow";  
        //This is an empty string  
        String str2 = "";  
        String str3 = "Goat";  
  
        System.out.println(str1.compareTo(str2));  
  
        System.out.println(str2.compareTo(str3));  
    }  
}
```

Sumber: <https://beginnersbook.com/2013/12/java-string-compareto-method-example/>.

Keluaran:

```
3  
-4
```

Penjelasan:

- Baris      `System.out.println(str1.compareTo(str2));`  
membandingkan str1 dan str2 selisihnya adalah 3.
- Baris      `System.out.println(str2.compareTo(str3));`  
membandingkan str2 dengan str3, selisihnya adalah -4.

- a. *Method substring* adalah *method* yang mendapatkan *substring* dari sebuah *string*.

Contoh:

```
public class SubstringTest {  
    public static void main(String[] args) {  
        String testString = "ABCDEFGHIJ";  
        System.out.println(testString.substring(0));  
        System.out.println(testString.substring(1));  
        System.out.println(testString.substring(2));  
        System.out.println(testString.substring(3));  
        System.out.println(testString.substring(4));  
        System.out.println(testString.substring(5));  
        System.out.println(testString.substring(6));  
        System.out.println(testString.substring(7));  
        System.out.println(testString.substring(8));  
        System.out.println(testString.substring(9));  
    }  
}
```

Sumber: <https://javadevnotes.com/java-substring-examples>

Keluaran:

```
ABCDEFGHIJ  
BCDEFGHIJ  
CDEFGHIJ  
DEFGHIJ  
EFGHIJ  
FGHIJ  
GHIJ  
HIJ  
IJ  
J
```

Penjelasan:

- Baris `testString.substring(1)` akan menghasilkan keluaran BCDEFGHIJ, begitu seterusnya.

Kode Program di atas bisa disederhanakan dengan pernyataan *for* menjadi:

```
public class SubstringTest {  
    public static void main(String[] args) {  
        String testString = "ABCDEFGHIJ";  
        int ulang = 0;  
        for (ulang = 0 ; ulang <= 10; ulang++)  
        {  
  
            System.out.println(testString.substring(ulang));  
        }  
    }  
}
```

Hasilnya keluaran sama.



### RANGKUMAN

*String* pada bahasa pemrograman Java menjadi bagian tersendiri, pada Java *string* dipersentasikan dengan *class* yang bersifat “final” yang didalamnya terdapat beberapa utilitas yang dapat digunakan untuk melakukan pengolahan variabel bertipe data “*String*” tersebut.

*Class String* memiliki banyak *constructor* yang langsung bisa dimanfaatkan oleh *programmer* untuk memodifikasi *string*. *Class String* sendiri diturunkan dari *library java.lang*. Beberapa *method-method* penting dari *class String* adalah *valueOf*, *copyValueOf*, *compareTo*, *startsWith*, *subString*, *isEmpty* dan masih banyak yang lainnya.



## TES FORMATIF 1

---

Pilihlah satu jawaban yang paling tepat!

- 1) *Method* yang digunakan untuk mengetahui panjang *string* pada *class string* adalah....
  - A. *length*
  - B. *concat*
  - C. *charAt*
  - D. *valueOf*
  
- 2) Fungsi dari *method concat* adalah....
  - A. Mengganti *string* dengan *string* yang lain
  - B. Menghapus beberapa bagian karakter dari *string*
  - C. Memasukkan *string* kedalam kelompok *string* yang lain
  - D. Menggabung *string* dengan *string* yang lain
  
- 3) Eksepsi *StringIndexOutOfBoundsException* akan mengembalikan suatu pesan kesalahan apabila pada *method charAt* diberikan....
  - A. Nomor indeks yang diberikan lebih besar dari atau sama dengan panjang *string* atau angka negatif
  - B. Nomor indeks yang diberikan lebih kecil dari atau sama dengan panjang *string* atau angka negatif
  - C. Nomor indeks yang diberikan lebih besar dari atau sama dengan panjang *string* atau angka positif
  - D. Nomor indeks yang diberikan lebih kecil dari atau sama dengan panjang *string* atau angka positif
  
- 4) Untuk mengubah berbagai jenis nilai ke bentuk *string* pada Java dapat menggunakan *method string*....
  - A. *trim*
  - B. *valueIn*
  - C. *valueOf*
  - D. *valueString*
  
- 5) *Method compareTo* berfungsi untuk....
  - A. Membandingkan *String* dengan *String* lain
  - B. Membandingkan *String* dengan *float* lain
  - C. Membandingkan *String* dengan *char* lain
  - D. Membandingkan *String* dengan *integer* lain

- 6) *Method* yang memeriksa apakah *string* ini dimulai dengan awalan yang diberikan. Ini mengembalikan *true* jika *string* ini dimulai dengan awalan yang diberikan, dan jika tidak maka mengembalikan *false* adalah....
  - A. *startEnd*
  - B. *startsWith*
  - C. *trim*
  - D. *subString*
- 7) *Method* yang membandingkan *string* dengan *objek string*, jika *string* pembanding sama dengan *string* yang dibandingkan, maka nilai balikannya *true*, dan jika tidak maka nilainya *false* adalah....
  - A. *equals*
  - B. *trim*
  - C. *subString*
  - D. *compareTo*
- 8) Untuk mengubah karakter dari huruf kecil menjadi huruf kapital dapat menggunakan *method*....
  - A. *trim*
  - B. *subString*
  - C. *toLowerCase*
  - D. *toUpperCase*
- 9) Untuk mengubah karakter dari huruf kapital menjadi huruf kecil dapat menggunakan *method*....
  - A. *trim*
  - B. *subString*
  - C. *toLowerCase*
  - D. *toUpperCase*
- 10) *Method* yang digunakan untuk menghilangkan spasi yang terkandung dalam *string* di awal dan dibelakang adalah....
  - A. *trim*
  - B. *subString*
  - C. *toLowerCase*
  - D. *toUpperCase*

Cocokkanlah jawaban Anda dengan Kunci Jawaban Tes Formatif 1 yang terdapat di bagian akhir modul ini. Hitunglah jawaban yang benar. Kemudian, gunakan rumus berikut untuk mengetahui tingkat penguasaan Anda terhadap materi Kegiatan Belajar 1.

$$\text{Tingkat penguasaan} = \frac{\text{Jumlah Jawaban yang Benar}}{\text{Jumlah Soal}} \times 100\%$$

Arti tingkat penguasaan: 90 - 100% = baik sekali

80 - 89% = baik

70 - 79% = cukup

< 70% = kurang

Apabila mencapai tingkat penguasaan 80% atau lebih, Anda dapat meneruskan dengan Kegiatan Belajar 2. **Bagus!** Jika masih di bawah 80%, Anda harus mengulangi materi Kegiatan Belajar 1, terutama bagian yang belum dikuasai.

**KEGIATAN BELAJAR 2*****String Buffer***

*C*lass *StringBuffer* adalah alternatif lain dari *class String*, *class StringBuffer* lebih fleksibel dibanding dengan *class String*. Kemampuan *class* ini adalah dapat memodifikasi berulangkali *string* yang tersimpan di objek *class StringBuffer* yang menggunakan beberapa *constructor* dan *method* yang dimilikinya.

**A. CLASS STRING BUFFER**

*Class StringBuffer* adalah *class* yang diturunkan dari *library* yang ada di objek *java.lang*. Kelebihan dari *class StringBuffer* selain bisa memodifikasi *string* yang ada dalam *buffer string*, fitur ini juga memiliki kapasitas untuk bisa otomatis melebar selama acuan ketentuan Java memenuhi, tidak perlu mengalokasikan *buffer array* internal baru, jika *buffer* internal meluas, maka secara otomatis dibuat lebih besar.

**B. CONSTRUCTOR CLASS STRINGBUFFER**

*Class StringBuffer* memiliki setidaknya tiga *constructor*, *constructor-constructor* tersebut bisa digunakan untuk mengkonstruksi objek *class StringBuffer*. Ketiga *constructor* tersebut yaitu: *StringBuffer()*, *StringBuffer(int length)*, dan *StringBuffer(String str)*.

- *StringBuffer()*: Mengkontruksi *buffer string* kosong (tanpa nilai)
- *StringBuffer(int length)*: Mengkontruksi *buffer string* (tanpa nilai) dengan kapasitas yang disebutkan oleh paramater *length*.
- *StringBuffer(String str)*: Mengkonstruksi *buffer string* dengan paramater *string*.

Contoh 8.18:

```
// Program constructor StringBuffer
public class ConstructorStringBuffer {

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

```

StringBuffer StringBuffer1 = new StringBuffer();
StringBuffer StringBuffer2 = new StringBuffer(60);
StringBuffer StringBuffer3 =
    new StringBuffer("Belajar Java dan JavaScript");

    System.out.println();
    System.out.println("1.  StringBuffer()      :  " + 
StringBuffer1);
    System.out.println("2.  StringBuffer(60)    :  " + 
StringBuffer2);
    System.out.println("3.  StringBuffer(str)  :  " + 
StringBuffer3);
}
}

```

Keluaran:

1. `StringBuffer()` :
2. `StringBuffer(60)` :
3. `StringBuffer(str)` : Belajar Java dan JavaScript

### C. METHOD PADA CLASS STRINGBUFFER

Beberapa *method* yang cukup penting yang terdapat pada *class* `StringBuffer` sebagai berikut:

#### 1. *Method append()*

*Method append* digunakan untuk menambahkan *string* pada *string buffer*

Sintaksis:

```
public StringBuffer append(String S)
```

Contoh 8.19:

```

class StringBufferExample {
    public static void main(String args[])
}

```

```

{
    StringBuffer StringBuffer1 = new StringBuffer("Belajar Java
");
    StringBuffer    StringBuffer2    =    new    StringBuffer("dan
JavaScript");

    //menambahkan string dengan append
    StringBuffer1.append(StringBuffer2);

    System.out.println(StringBuffer1);
}
}

```

Keluaran:

Belajar Java dan JavaScript

## 2. *Method insert()*

*Method insert* digunakan memasukkan *string* yang ditentukan pada posisi yang ditentukan.

Sintaksis:

```
public StringBuffer insert(int offset, String s)
```

Keterangan:

- *offset* : adalah bertipe data *integer*, mulai memasukkan *string* ke *string* berapa *integer*.
- *s* : *String* yang dimasukkan ke *StringBuffer*.

Contoh 8.20:

```

//Program Insert StringBuffer
class contohInsertStringBuffer {
    public static void main(String args[])
    {
        StringBuffer    StringBuffer1    =    new    StringBuffer("Belajar
JavaScript");

```

```
StringBuffer StringBuffer2 = new StringBuffer("Java ");

System.out.print("Sebelum proses Insert: ");
System.out.println(StringBuffer1);
System.out.print("Insert ke-1: ");

StringBuffer1.insert(8, StringBuffer2);

System.out.println(StringBuffer1);
System.out.print("Insert Ke-2: ");

StringBuffer1.insert(13, "dan ");

System.out.println(StringBuffer1);
}
```

Keluaran:

```
Sebelum proses Insert: Belajar JavaScript
Insert ke-1: Belajar Java JavaScript
Insert Ke-2: Belajar Java dan JavaScript
```

### 3. *Method replace()*

*Method replace* digunakan untuk menimpa *string* yang ditentukan dan jumlah *string* yang ditimpas.

Sintaksis:

```
public StringBuffer replace(int start, int end, String S)
```

Keterangan:

- *start* : mulai dari *string* berapa yang mau ditimpas pada *StringBuffer*.
- *end* : sampai *string* ke berapa yang mau ditimpas pada *StringBuffer*.
- *S* : *String* yang akan menimpa.

Contoh 8.21:

```
//Program Replace StringBuffer
public class contohReplaceStringBuffer {

    public static void main(String args[]) {

        StringBuffer StringBuffer1 =
            new StringBuffer("Belajar Java dan JavaScript");

        StringBuffer StringBuffer2 = new StringBuffer("Power ");

        System.out.print("Sebelum ditimpa: ");
        System.out.println(StringBuffer1);

        System.out.print("Sesudah ditimpa: ");

        StringBuffer1.replace(17, 21, "Power");

        System.out.println(StringBuffer1);
    }
}
```

Keluaran:

```
Sebelum ditimpa: Belajar Java dan JavaScript
Sesudah ditimpa: Belajar Java dan PowerScript
```

#### 4. *Method delete()*

*Method delete()* digunakan untuk menghapus satu atau lebih karakter dalam *substring*, parameter yang menyertainya adalah bertipe data *integer*.

Sintaksis:

```
public StringBuffer delete(int start, int end)
```

Keterangan:

- start : indeks awal yang akan mulai dihapus
- end : indeks akhir yang akan dihapus

Eksepsi yang akan dijalankan jika terjadi di luar ketentuan adalah eksepsi *StringIndexOutOfBoundsException*, jika nilai start lebih kecil 0 dan end lebih besar dari panjang string yang ada dalam *buffer string*.

Contoh 8.22:

```
//Program Delete StringBuffer
class contohDeleteStringBuffer {
    public static void main(String args[])
    {
        StringBuffer stringBuffer = new StringBuffer("Belajar
JavaScript");

        System.out.print("Sebelum proses Delete: ");
        System.out.println(stringBuffer);

        System.out.print("Sebelum proses Delete: ");
        stringBuffer.delete(12,18);
        System.out.println(stringBuffer);
    }
}
```

Keluaran:

```
Sebelum proses Delete: Belajar JavaScript
```

Setelah proses Delete: Belajar Java

### 5. *Method reverse()*

*Method reverse* adalah *method* yang digunakan untuk membalikkan urutan karakter, misalnya kata “Java” menjadi “avaJ”. *Method* ini sangat praktis dari pada harus membuat algoritma baru untuk membalikkan kata.

Sintaksis:

```
public StringBuffer reverse()
```

Contoh 8.23:

```
//Program Reverse StringBuffer
class contohReverseStringBuffer
{
    public static void main(String args[]) {

        StringBuffer stringBuffer = new StringBuffer("Belajar
Java");
        System.out.println("Sebelum dibalik : "+stringBuffer);

        stringBuffer.reverse();
        System.out.println("Sesudah dibalik : "+stringBuffer);

    }
}
```

Keluaran:

Sebelum dibalik : Belajar Java

Sesudah dibalik : avaJ rajaleB

## 6. *Method substring*

*Method substring* adalah berfungsi untuk mengembalikan nilai *String* baru yang berisi karakter lanjutan setelah dipotong oleh *method* tersebut. *Substring* dimulai pada indeks yang ditentukan dan meluas ke akhir urutan yang dinginkan.

Sintaksis:

```
public String substring(int start)
```

Penjelasan:

- start : posisi mulai di-*substring*.

Contoh 8.24:

```
public class ContohSubString {  
  
    public static void main(String[] args) {  
  
        StringBuffer stringbuffer = new StringBuffer("Belajar  
Java");  
        System.out.print("Sebelum substring : ");  
        System.out.println(stringbuffer);  
  
        System.out.print("Sesudah substring : ");  
        System.out.println(stringbuffer.substring(3));  
    }  
}
```

Keluaran:

```
Sebelum substring : Belajar Java  
Sesudah substring : ajar Java
```



## LATIHAN

---

Untuk memperdalam pemahaman Anda mengenai materi di atas, kerjakanlah latihan berikut!

- 1) Buatlah program untuk meng-*input* Nama Pelanggan, ada tiga yang di-*input* yaitu: Nama Depan, Nama Tengah dan Nama Belakang, ketiga yang di-*input* tersebut ditampung dalam *array* dan kemudian digabung menjadi 1 menggunakan *method Append()*, adapun output yang diinginkan adalah:

```
Masukkan Data Nama  
-----  
Masukkan Nama Depan      :  
Masukkan Nama Tengah     :  
Masukkan Nama Belakang   :  
-----  
Nama Lengkap Anda Adalah :
```

### Petunjuk Jawaban Latihan

- 1) Menjawab pertanyaan di atas, maka spesifikasi yang harus disiapkan adalah:
  - a. proses input, harus meng-import *java.util.Scanner* untuk *method input*.
  - b. Menyiapkan deklarasi *array* dan alokasi memori, Deklarasi variabel penampung *String Buffer*.
  - c. Proses isi *array* untuk menampung *string* masukan dengan menggunakan pernyataan perulangan (*for*).
  - d. Menggunakan *method.Append()* untuk penggabungan dan dikolaborasi dengan proses perulangan (*for*).

- e. Terakhir adalah menampilkan Nama Lengkap.

```
import java.util.Scanner;
public class inputNamaPel {
    public static void main(String args[]) {
        int urutNama = 3;
        Scanner input = new Scanner(System.in);
        System.out.println("Masukkan Data Nama");
        System.out.println("-----");
        //length = input.nextInt();

        String[] nama_pel = new String[urutNama];
        StringBuffer StringBufferNama = new StringBuffer("");

        for (int ulang = 0; ulang < urutNama; ulang++)
        {
            if (ulang == 0)
            {
                System.out.print("Masukkan Nama Depan : ");
                nama_pel[ulang] = input.nextLine();
            }
            else if (ulang == 1)
            {
                System.out.print("Masukkan Nama Tengah : ");
                nama_pel[ulang] = input.next();
            }
            else
            {
                System.out.print("Masukkan Nama Belakang : ");
                nama_pel[ulang] = input.next();
            }

        }

        input.close();
        System.out.println("-----");
```

```
System.out.print("Nama Lengkap Anda adalah : ");

    for (int ulang = 0; ulang < urutNama; ulang++)
    {
        StringBufferNama.append(nama_pel[ulang] + ' ');
    }

    System.out.print(StringBufferNama);
}
}
```

#### Keluaran:

Jika program dijalankan yang pertama diminta adalah *Masukkan Nama Depan* (misal: *Kani*), kedua *Masukkan Nama Tengah* (misal: *Launggu*), dan *Masukkan Nama Belakang* (misal: *Sempo*), maka keluarannya sebagai berikut:

```
Masukkan Data Nama
-----
Masukkan Nama Depan      : Kani
Masukkan Nama Tengah     : Launggu
Masukkan Nama Belakang   : Sempo
-----
Nama Lengkap Anda adalah : Kani Launggu Sempo
```

#### Penjelasan:

- `import java.util.Scanner;`
- `import java.util.Scanner;` adalah sebuah pustaka java yang berisi class dan method yang digunakan untuk mengambil input-an dari *keyboard*.
- `nama_pel[ulang] = input.nextLine();` adalah baris program untuk menangkap input-an *keyboard* yang kemudian dimasukkan ke *array* dengan indeks tertentu.



## RANGKUMAN

---

*Class StringBuffer* adalah menjadi alternatif lain dari *class String*, karena begitu fleksibel sehingga banyak yang menggunakan dibanding dengan *class String*, kemampuan *class* ini adalah dapat memodifikasi berulangkali *string* yang tersimpan di objek *class StringBuffer* yang menggunakan beberapa *constructor* dan *method* yang dimilikinya.

*Class StringBuffer* adalah *class* yang diturunkan dari *library* yang ada di objek *java.lang*. Kelebihan dari *class StringBuffer* selain dari bisa memodifikasi *string* yang ada dalam *buffer string* adalah memiliki kapasitas yang bisa otomatis melebar selama acuan ketentuan Java memenuhi, tidak perlu mengalokasikan *buffer array* internal baru, karena jika *buffer* internal meluar, maka secara otomatis dibuat lebih besar.

*Class StringBuffer* yang bisa digunakan untuk adalah *StringBuffer()* untuk *buffer String* kosong, *StringBuffer(int length)* untuk kapasitas *string* ditentukan, dan *StringBuffer(String str)* untuk *string* dengan nilai ada berupa *string*.

*Method-method* *StringBuffer* yang bisa digunakan diantaranya: *append()*, *insert()*, *replace()*, *delete()*, dan lain-lain.



## TES FORMATIF 2

---

Pilihlah satu jawaban yang paling tepat!

- 1) Konstruktor *StringBuffer* untuk *buffer string* adalah *constructor*....
  - A. *StringBuffer(int length)*
  - B. *StringBuffer(String str)*
  - C. *StringBuffer()*
  - D. *Append()*
  
- 2) Untuk mem-*buffer string* dengan menentukan panjang *string* pada saat deklarasi adalah *method*....
  - A. *StringBuffer(int length)*
  - B. *StringBuffer(String str)*
  - C. *StringBuffer()*
  - D. *Append()*

- 3) Fungsi dari *method append()* adalah....
  - A. Menambahkan *string* kedalam *buffer string*;
  - B. Menghapus *string* kedalam *buffer string*;
  - C. Menimpa *string* kedalam *buffer string*;
  - D. Menghapus beberapa *string* kedalam *buffer string*;
- 4) Memasukkan *string* yang ditentukan pada posisi tertentu adalah fungsi dari *method*....
  - A. *append*
  - B. *replace*
  - C. *insert*
  - D. *delete*
- 5) *Method StringBuffer* yang digunakan untuk menimpa *string* yang ditentukan dan jumlah *string* yang ditimpak....
  - A. *append*
  - B. *replace*
  - C. *insert*
  - D. *delete*
- 6) *Method* yang berfungsi untuk menghapus satu atau lebih karakter dalam *substring*, parameter yang menyertainya adalah bertipe data *integer* adalah....
  - A. *delete*
  - B. *replace*
  - C. *insert*
  - D. *append*
- 7) Fungsi *method reverse* pada *class StringBuffer* adalah....
  - A. Membalikkan urutan karakter.
  - B. Menghapus karakter
  - C. Mennyisipkan karakter.
  - D. Membuang beberapa karakter.
- 8) *Method* yang berfungsi untuk mengembalikan nilai *String* baru yang berisi karakter lanjutan setelah dipotong oleh *method* tersebut....
  - A. *delete*
  - B. *append*
  - C. *subString*
  - D. *Buffer String*

- 9) Yang bukan termasuk *method* dari *StringBuffer* adalah....
- A. *append*
  - B. *delete*
  - C. *insert*
  - D. *move*
- 10) Tiga Nama konstruktor dari *StringBuffer* adalah....
- A. *append()*, *delete()*, *insert()*
  - B. *insert()*, *delete()*, *subString()*
  - C. *reverse()*, *delete()*, *insert()*
  - D. *StringBuffer()*, *StringBuffer(int length)*, *StringBuffer (String str)*

Cocokkanlah jawaban Anda dengan Kunci Jawaban Tes Formatif 2 yang terdapat di bagian akhir modul ini. Hitunglah jawaban yang benar. Kemudian, gunakan rumus berikut untuk mengetahui tingkat penguasaan Anda terhadap materi Kegiatan Belajar 2.

$$\text{Tingkat penguasaan} = \frac{\text{Jumlah Jawaban yang Benar}}{\text{Jumlah Soal}} \times 100\%$$

Arti tingkat penguasaan: 90 - 100% = baik sekali

80 - 89% = baik

70 - 79% = cukup

< 70% = kurang

Apabila mencapai tingkat penguasaan 80% atau lebih, Anda dapat meneruskan dengan Kegiatan Belajar 3. **Bagus!** Jika masih di bawah 80%, Anda harus mengulangi materi Kegiatan Belajar 2, terutama bagian yang belum dikuasai.

**KEGIATAN BELAJAR 3*****Math***

Pada setiap bahasa pemrograman, operator tambah, kurang, kali, bagi adalah wajib ada, dengan keempat operator tersebut, hampir semua operasi perhitungan matematika terselesaikan, tapi apakah anda mau mengolah dan membuat algoritma sendiri untuk menyelesaikan operasi akar kuadrat, trigonometri dan algoritma? Saya berkeyakinan bahwa itu akan menyita waktu seorang *programmer* untuk membuat *method-method*-nya.

Mulai pada Java SE 6, telah menyediakan *class* untuk menyelesaikan operasi-operasi dasar lainnya, seperti operasi trigonometri, absolut, log, dan lain sebagainya. *Class* tersebut masuk menjadi API pada Java hingga tidak perlu melakukan *download* terlebih dahulu, dan *class* tersebut diberi nama *class Math*.

**A. CLASS MATH**

*Class Math* adalah turunan dari *java.lang* (*java.lang.Math*) berisi *method* untuk melakukan operasi numerik dasar seperti trigonometri, logaritma, akar kuadrat, bahkan eksponensial sekalipun.

Sintaksis pewarisan *java.lang*:

```
public final class Math extends Object
```

**B. METHOD-METHOD PADA CLASS MATH UNTUK MATEMATIKA DASAR**

*Class Math* memiliki banyak *method* untuk menyelesaikan operasi-operasi matematika dasar.

### 1. Method *abs()*

*Method abs()* adalah mengembalikan (menghasilkan) nilai mutlak (absolut) dari berbagai tipe data seperti *integer*, *long*, *float*, *double*. Jika nilai yang diberikan adalah negatif, maka nilai yang akan dikembalikan adalah negasi dari nilai negatif.

Sintaksis:

```
public static int abs(int i)
```

atau

```
public static double abs(double d)
```

atau

```
public static float abs(float f)
```

atau

```
public static long abs(long lng)
```

Keterangan:

- *i, d, f, long* : adalah *argument parameter* yang harus diberi nilai

*Method abs()* ini akan mengembalikan nilai absolut dari *argument* dengan ketentuan berikut:

1. Jika diberikan nilai positif atau nilai negatif sebagai *argument*, maka akan mengembalikan nilai positif.

2. Jika *argumentnya* adalah nilai tak terhingga, maka menghasilkan positif tak terhingga.
3. Jika *argument* sama dengan nilai `integer.MIN_VALUE` atau `Long.MIN_VALUE`, nilai `int` atau nilai `long` yang paling negatif representatif, hasilnya adalah nilai yang sama, yaitu negatif.
4. Jika *argument* nya adala **NaN** (*Not-a-Number*), maka balikannya adalah **NaN**.

Contoh 8.25:

```
// Program menggunakan method abs()
public class contohMethodAbs
{
    public static void main(String args[])
    {
        int intX = 78;
        int intY = -80;
        float singleX = 12;
        double singleY = -54;

        //print nilai absolut
        System.out.println(Math.abs(intX));
        System.out.println(Math.abs(intY));
        System.out.println(Math.abs(Integer.MIN_VALUE));

        System.out.println(Math.abs(singleX));
        System.out.println(Math.abs(singleY));

    }
}
```

Keluaran:

78

80

```
-2147483648
```

```
12.0
```

```
54.0
```

## 2. **Method max()**

*Method max()* adalah *method* yang digunakan untuk mengembalikan nilai terbesar dari dua *argument* yang diberikan. *Argument* yang diperbolehkan adalah *argument* yang bertipe data *int*, *float*, *double*, dan *long*.

Sintaksis:

```
public static int max(int a, int b)
```

atau

```
public static double max(double a, double b)
```

atau

```
public static long max(long a, long b)
```

atau

```
public static float max(float a, float b)
```

Keterangan:

- a : nilai pertama
- b : nilai kedua

Contoh 8.26:

```
// Program menggunakan method max()
public class contohMethodmax
{
    public static void main(String args[])
    {
        int intX = 78;
        int intY = 80;
        float singleX = 12;
        double singleY = 4;

        int intMax = Math.max(intX,intY);
        double singleMax = Math.max(singleX,singleY);
        //print nilai terbesar
        System.out.println("Nilai terbesar Int : "+intMax);
        System.out.println("Nilai terbesar float : "+singleMax);
    }
}
```

Keluaran:

```
Nilai terbesar Int : 80
```

```
Nilai terbesar float : 12.0
```

### 3. **Method min()**

*Method min()* adalah *method* yang digunakan untuk mengembalikan nilai terkecil dari dua *argument* yang diberikan. *Argument* yang diperbolehkan adalah *argument* yang bertipe data *int*, *float*, *double*, dan *long*.

Sintaksis:

```
public static int min(int a, int b)
```

atau

```
public static double min(double a, double b)
```

atau

```
public static long min(long a, long b)
```

atau

```
public static float min(float a, float b)
```

Keterangan:

- a : nilai pertama
- b : nilai kedua

Contoh 8.27:

```
// Program menggunakan method min()
public class contohMethodmin
{
    public static void main(String args[])
    {
        int intX = 78;
```

```
        int intY = 80;
        float singleX = 12;
        double singleY = 4;

        int intMin = Math.min(intX,intY);
        double singleMin = Math.min(singleX,singleY);
        //print nilai terkecil
        System.out.println("Nilai terkecil Int : "+intMax);
        System.out.println("Nilai terkecil float : "+singleMax);
    }
}
```

Keluaran:

```
Nilai terkecil Int : 78
Nilai terkecil float : 4.0
```

#### 4. *Method round()*

*Method round()* adalah *method* yang digunakan untuk pembulatan terdekat, misalnya: 4.71 maka dibulatkan menjadi 5, 4.41 maka dibulatkan menjadi 4, akan tetapi kalau 4.50, maka dibulatkan menjadi 5.

Sintaksis:

```
public static int round(float x)
public static long round(double x)
```

Keterangan:

- x : nilai *floating-point* yang akan dibulatkan ke *integer*.

Contoh 8.28:

```
public class PembulatanDesimalRound {  
    public static void main(String[] args) {  
  
        double doubleX,doubleY;  
        doubleX=4.71;  
        doubleY=4.41;  
        System.out.print("Pembulatan ke terdekat doubleX : ");  
        System.out.println(Math.round(doubleX));  
        System.out.print("Pembulatan ke terdekat doubleY : ");  
        System.out.println(Math.round(doubleY));  
  
    }  
}
```

Keluaran:

```
Pembulatan ke terdekat doubleX : 5
```

```
Pembulatan ke terdekat doubleY : 4
```

## 5. *Method sqrt()*

*Method sqrt()* adalah *method* untuk mengembalikan nilai berupa akar kuadrat suatu angka.

Sintaksis:

```
public static double sqrt(double x)
```

Keterangan:

- x : angka

Contoh 8.29:

```
//Program Mencari Akar Kuadrat (SQRT)
public class methodSQRT {
    public static void main(String[] args) {

        double doubleX,doubleY;
        doubleX=4.71;
        doubleY=8.41;
        System.out.print("Akar Quadrat 4.71 : ");
        System.out.println(Math.sqrt(doubleX));
        System.out.print("Akar Quadrat 8.41 : ");
        System.out.println(Math.sqrt(doubleY));

    }
}
```

Keluaran:

```
Akar Quadrat 4.71 : 2.1702534414210706
```

```
Akar Quadrat 8.41 : 2.9
```

## 6. *Method cbrt()*

*Method cbrt()* adalah *method* untuk mengembalikan nilai berupa akar pangkat tiga suatu angka.

Sintaksis:

```
public static double cbrt(double x)
```

Keterangan:

- x : angka

Contoh 8.30:

```
//Program Mencari Akar Pangkat 3 (CBRT)
public class methodCBRT {
    public static void main(String[] args) {

        double doubleX,doubleY;
        doubleX=4.71;
        doubleY=8.41;
        System.out.print("Akar ^3 dari 4.71 : ");
        System.out.println(Math.cbrt(doubleX));
        System.out.print("Akar ^3 dari 8.41 : ");
        System.out.println(Math.cbrt(doubleY));

    }
}
```

Keluaran:

```
Akar ^3 dari 4.71 : 1.6762558340234293
```

```
Akar ^3 dari 8.41 : 2.0335990575118257
```

## C. **METHOD-METHOD PADA CLASS MATH UNTUK MATEMATIKA LOGARITMIK**

Beberapa *method class* *Math* untuk operasi matematika logaritmik, yaitu: *Log()*, *log10()*, dan *exp()*.

### 1. **Method *log()***

*Method log()* adalah *method* yang digunakan untuk mengetahui nilai logaritmik dari angka apapun, *method* ini mengembalikan nilai logaritma (basis e) dari nilai tipe *single* sebagai parameter.

Sintaksis:

```
public static double Log(double x)
```

Keterangan:

- x : nilai yang dimasukkan oleh user.

Contoh 8.31:

```
//Program Mencari Log
public class methodLog {
    public static void main(String[] args) {

        double doubleX,doubleY;
        doubleX=4.71;
        doubleY=8.41;
        System.out.print("Nilai log dari 4.71 : ");
        System.out.println(Math.Log(doubleX));
        System.out.print("Nilai log dari 8.41 : ");
        System.out.println(Math.Log(doubleY));

    }
}
```

Keluaran:

```
Nilai log dari 4.71 : 1.5496879080283263
```

```
Nilai log dari 8.41 : 2.1294214739848565
```

## 2. Method *log10*

*Method log()* adalah *method* yang digunakan untuk mengetahui nilai logaritmik dari angka basis 10, *method* ini mengembalikan nilai logaritma (basis 10) dari nilai tipe single sebagai parameter.

Sintaksis:

```
public static double Log10(double x)
```

Keterangan:

- x : nilai yang dimasukkan oleh *user*.

Contoh 8.32:

```
//Program Mencari Log10
public class methodLog10 {
    public static void main(String[] args) {

        double doubleX,doubleY;
        doubleX=4.71;
        doubleY=8.41;
        System.out.print("Nilai log10 dari 4.71 : ");
        System.out.println(Math.Log10(doubleX));
        System.out.print("Nilai log10 dari 8.41 : ");
        System.out.println(Math.Log10(doubleY));

    }
}
```

Keluaran:

```
Nilai log10 dari 4.71 : 0.6730209071288962
```

```
Nilai log10 dari 8.41 : 0.9247959957979122
```

## D. METHOD-METHOD PADA CLASS MATH UNTUK MATEMATIKA TRIGONOMETRI

Adapun *method-method* *Math* trigonometri, yaitu: *sin()*, *cos()*, *tan()*, *asin()*, *atan()*, dan *atan()*.

### 1. *Method sin()*

*Method sin()* adalah *method* yang mengembalikan nilai sinus sudut trigonometri, *method* ini mengembalikan nilai antar -1 hingga 1.

Sintaksis:

```
public static double sin(double a)
```

Keterangan:

- a : sudut, dalam radian.

Contoh 8.33:

```
//Program Mencari Sinus
public class methodSin {
    public static void main(String[] args) {

        double doubleX,doubleY;
        doubleX=4.71;
        doubleY=8.41;

        doubleX = Math.toRadians(doubleX);
        doubleY = Math.toRadians(doubleY);

        System.out.print("Nilai sin dari 4.71 : ");
        System.out.println(Math.sin(doubleX));
```

```
        System.out.print("Nilai sin dari 8.41 : ");
        System.out.println(Math.sin(doubleY));

    }
}
```

Keluaran:

```
Nilai sin dari 4.71 : 0.08211245341964743
Nilai sin dari 8.41 : 0.0014347475549568373
```

## 2. *Method cos()*

*Method cos()* adalah *method* yang mengembalikan nilai cosinus sudut trigonometri, *method* ini mengembalikan nilai antar -1 hingga 1.

Sintaksis:

```
public static double cos(double a)
```

Keterangan:

- a : sudut, dalam radian.

Contoh 8.34:

```
//Program Mencari Cosinus
public class methodSin {
    public static void main(String[] args) {

        double doubleX,doubleY;
        doubleX=4.71;
        doubleY=8.41;
```

```
        doubleX = Math.toRadians(doubleX);
        doubleY = Math.toRadians(doubleY);

        System.out.print("Nilai cosinus dari 4.71 : ");
        System.out.println(Math.cos(doubleX));
        System.out.print("Nilai cosinus dari 8.41 : ");
        System.out.println(Math.cos(doubleY));
    }
}
```

Keluaran:

```
Nilai cosinus dari 4.71 : 0.9966230706708561
Nilai cosinus dari 8.41 : 0.9892468215973416
```

### 3. *Method tan()*

*Method tan()* adalah *method* yang mengembalikan nilai tangen sudut trigonometri.

Sintaksis:

```
public static double tan(double a)
```

Keterangan:

- a : sudut, dalam radian.

Contoh 8.35:

```
//Program Mencari Tangen
public class methodTan {
    public static void main(String[] args) {
```

```
double doubleX,doubleY;  
doubleX=4.71;  
doubleY=8.41;  
  
doubleX = Math.toRadians(doubleX);  
doubleY = Math.toRadians(doubleY);  
  
System.out.print("Nilai tangen dari 4.71 : ");  
System.out.println(Math.tan(doubleX));  
System.out.print("Nilai tangen dari 8.41 : ");  
System.out.println(Math.tan(doubleY));  
}  
}
```

Keluaran:

```
Nilai tangen dari 4.71 : 0.08239068092651633
```

```
Nilai tangen dari 8.41 : 0.1478454959243583
```

#### 4. *Method asin()*

*Method asin()* adalah *method()* yang digunakan untuk menghitung *Arc Arconometric* dari sudut. Arc Sin juga disebut sebagai invers dari Sinus. Metode ini mengembalikan nilai antara  $-\pi / 2$  dan  $\pi / 2$ .

Sintaksis:

```
public static double asin(double a)
```

Keterangan:

- a : sudut, dalam radian.

Contoh 8.36:

```
//Program Mencari Arc Sinus
public class methodAsin {
    public static void main(String[] args) {

        double doubleX,doubleY;
        doubleX=0.082;
        doubleY=0.001;

        System.out.print("Nilai asin dari 0.082 : ");
        System.out.println(Math.asin(doubleX));
        System.out.print("Nilai asin dari 0.001 : ");
        System.out.println(Math.asin(doubleY));
    }
}
```

Keluaran:

```
Nilai asin dari 0.082 : 0.08209217383954859
Nilai asin dari 0.001 : 0.0010000001666667416
```

### 5. *Method acos()*

*Method acos()* adalah *method()* yang digunakan untuk menghitung *Arc Arconometric* dari sudut. *Arc cosinus* juga disebut sebagai invers dari cosinus. *Metode* ini mengembalikan nilai 0.0 dan *pi*.

Sintaksis:

```
public static double acos(double a)
```

Keterangan:

- a : sudut, dalam radian.

Contoh 8.37:

```
//Program Mencari Arc Acosinus
public class methodAcosinus {
    public static void main(String[] args) {

        double doubleX,doubleY;
        doubleX=0.082;
        doubleY=0.001;

        System.out.print("Nilai acos dari 0.082 : ");
        System.out.println(Math. acos(doubleX));
        System.out.print("Nilai acos dari 0.001 : ");
        System.out.println(Math. acos(doubleY));
    }
}
```

Keluaran:

```
Nilai acos dari 0.082 : 1.488704152955348
Nilai acos dari 0.001 : 1.56979632662823
```

## 6. *Method atan()*

*Method atan()* adalah *method()* yang digunakan untuk menghitung *Arc Arconometric* dari sudut. Arc tan juga disebut sebagai invers dari Tan. Metode ini mengembalikan nilai antara -pi / 2 dan pi / 2.

Sintaksis:

```
public static double atan(double a)
```

Keterangan:

- a : sudut, dalam radian.

Contoh 8.38:

```
//Program Mencari Arc Atan
public class methodAtan {
    public static void main(String[] args) {

        double doubleX,doubleY;
        doubleX=0.082;
        doubleY=0.001;

        System.out.print("Nilai atan dari 0.082 : ");
        System.out.println(Math.atan(doubleX));
        System.out.print("Nilai atan dari 0.001 : ");
        System.out.println(Math.atan(doubleY));
    }
}
```

Keluaran:

```
Nilai atan dari 0.082 : 0.08181694860365457
```

```
Nilai atan dari 0.001 : 9.999996666668668E-4
```

## E. METHOD-METHOD PADA CLASS MATH UNTUK MATEMATIKA HIPERBOLIK

Adapun *method-method* *Math* trigonometri hiperbolik, yaitu: *sinh()*, *cosh()*, dan *tanh()*.

### 1. Method *sinh()*

*Method sinh()* adalah *method* yang digunakan untuk mengembalikan sinus hiperbolik suatu nilai. Sinus hiperbolik dengan nilai apapun dapat didefinisikan sebagai  $(e^x - e^{-x})/2$ , di mana *e* adalah angka *euler*.

Sintaksis:

```
public static double sinh(double a)
```

Keterangan:

- *a* : angka yang sinusi hiperboliknya harus dikembalikan.

Contoh 8.39:

```
//Program Mencari Sinus Hiperbolik
public class methodSinHiperbolik {
    public static void main(String[] args) {
        double doubleX,doubleY;
        doubleX=30;
        doubleY=25;

        System.out.print("Nilai sinh dari 30 : ");
        System.out.println(Math.sinh(doubleX));
        System.out.print("Nilai sinh dari 25 : ");
        System.out.println(Math.sinh(doubleY));
    }
}
```

Keluaran:

```
Nilai Sinh dari 30 : 5.343237290762231E12
```

```
Nilai sinh dari 25 : 3.600244966869294E10
```

## 2. ***Method cosh()***

*Method cosh()* adalah *method* yang digunakan untuk mengembalikan cosinus hiperbolik suatu nilai. Cosinus hiperbolik dengan nilai apapun dapat didefinisikan sebagai  $(e^x + e^{-x})/2$ , di mana e adalah angka *euler*.

Sintaksis:

```
public static double cosh(double a)
```

Keterangan:

- a : angka yang sinusi hiperboliknya harus dikembalikan.

Contoh 8.40:

```
//Program Mencari Cosinus Hiperbolik
public class methodCosHiperbolik {
    public static void main(String[] args) {

        double doubleX,doubleY;
        doubleX=30;
        doubleY=25;

        System.out.print("Nilai cosh dari 30 : ");
        System.out.println(Math.cosh(doubleX));
        System.out.print("Nilai cosh dari 25 : ");
        System.out.println(Math. cosh (doubleY));
    }
}
```

Keluaran:

```
Nilai cosh dari 30 : 5.343237290762231E12
```

```
Nilai cosh dari 25 : 3.600244966869294E10
```

### 3. *Method tanh()*

*Method tanh()* adalah *method* yang digunakan untuk mengembalikan tangen hiperbolik suatu nilai. tangen hiperbolik dengan nilai apapun dapat didefinisikan sebagai  $((e^x - e^{-x})/2) / ((e^x + e^{-x})/2)$ , di mana e adalah angka *euler* atau kita dapat katakan sebagai  $\tanh(a) = \sinh(a) / \cosh(a)$ .

Sintaksis:

```
public static double tanh(double a)
```

Keterangan:

- a : angka yang sinusi hiperboliknya harus dikembalikan.

Contoh 8.41:

```
//Program Mencari tangenH Hiperbolik
public class methodTangenHHiperbolik {
    public static void main(String[] args) {

        double doubleX,doubleY;
        doubleX=30;
        doubleY=25;

        System.out.print("Nilai tanh dari 30 : ");
        System.out.println(Math.tanh(doubleX));
        System.out.print("Nilai tanh dari 25 : ");
        System.out.println(Math.tanh(doubleY));
    }
}
```

{}

Keluaran:

Nilai tanh dari 30 : 1.0

Nilai tanh dari 25 : 1.0



## LATIHAN

---

Untuk memperdalam pemahaman Anda mengenai materi di atas, kerjakanlah latihan berikut!

- 1) Tentukan penyelesaian soal ini. Contoh soal dari laman:

<https://www.gurusma.id/2016/03/contoh-soal-trigonometri-lengkap.html>.

Berikut print Screen soalnya:

Tentukan nilai dari  $\sin 105^\circ + \sin 15^\circ = \dots \dots \dots$ ?

Jawab:

$$\begin{aligned} & \sin 105^\circ + \sin 15^\circ \\ &= 2 \sin 1/2 (105^\circ + 15^\circ) \cdot \cos 1/2 (105^\circ - 15^\circ) \\ &= 2 \sin 1/2 (120^\circ) \cdot \cos 1/2 (90^\circ) \\ &= 2 \sin 60^\circ \cdot \cos 45^\circ \\ &= 2 \cdot 1/2 \sqrt{3} \cdot 1/2 \sqrt{2} \\ &= 1/2 \sqrt{6} \end{aligned}$$

Buatlah dalam kode program contoh soal di atas.

### Petunjuk Jawaban Latihan

- 1) Petunjuk jawab pada soal di atas terlihat pada gambar, sehingga kita dengan mudah memvalidasi keluaran dari program yang dibuat. Hasil dari

$\sin 105 + \sin 15 = 1,225$ . Jika dikerjakan dalam program Java tentu tidak perlu menguraikan detail seperti pada gambar soal dan jawaban di atas. Adapun program Java untuk mencari nilai  $\sin 105$  dan  $\sin 15$  adalah sebagai berikut:

```
//Program Mencari Sinus
public class methodSin {
    public static void main(String[] args) {

        double doubleX,doubleY;
        doubleX=105.0;
        doubleY=15.0;

        doubleX = Math.toRadians(doubleX);
        doubleY = Math.toRadians(doubleY);

        System.out.print("Nilai sin 105 + sin 15 : ");
        System.out.println(Math.sin(doubleX)+Math.sin(doubleY));
    }
}
```

Keluaran:

```
Nilai sin 105 + sin 15 : 1.225
```



RANGKUMAN

Class *Math* adalah turunan dari *java.lang* (*java.lang.Math*) berisi *method* untuk melakukan operasi numerik dasar seperti trigonometri, logaritma, akar kuadrat, bahkan eksponensial sekalipun.

Beberapa *method* dari *class math* untuk operasi matematika dasar, yaitu: *method abs()*, *method max()*, *method min()*, *method round()*, *method sqrt()*, *method cbrt()*, dan masih banyak *method* lainnya. Untuk operasi matematika logaritmik, yaitu: *Log()*, *log10()*, dan *exp()* serta yang lainnya.

Untuk trigonometri, yaitu: *sin()*, *cos()*, *tan()*, *asin()*, *atan()*, dan *atan()*. Dan untuk trigonometri hiperbolik, yaitu: *sinh()*, *cosh()*, dan *tanh()*.

Untuk mempelajari lebih jauh beberapa materi *method* yang belum disebutkan, anda bisa mengunjungi laman:

[https://www.tutorialspoint.com/java/lang/java\\_lang\\_math.htm](https://www.tutorialspoint.com/java/lang/java_lang_math.htm).



### TES FORMATIF 3

---

Pilihlah satu jawaban yang paling tepat!

- 1) Penyelesaian operasi matematika seperti operasi trigonometri, absolut, log dirilis pada Java versi....
  - A. SE 5
  - B. SE 6
  - C. SE 7
  - D. SE 8
- 2) Sebuah program mengolah data *array* sebagai berikut:

```
...
double[] array = {1.9, 2.9, 4.7, 2.4, 5.8, 1.01}
...
```

keluaran dari program tersebut adalah:

1.01  
5.8

melihat dari keluaran program tersebut, program tersebut menggunakan *method*....

- A. *round()* dan *abs()*
  - B. *min()* dan *max()*
  - C. *max()* dan *min()*
  - D. *abs()* dan *max()*
- 3) Perhatikan angka-angka berikut:  
2.3 dibulatkan menjadi 2  
6.68 dibulatkan menjadi 7  
7.50 dibulatkan menjadi 8

*Method* yang cocok digunakan untuk 3 proses di atas adalah....

- A. *Method cosh()*
  - B. *Method abs()*
  - C. *Method tan()*
  - D. *Method round()*
- 4) Fungsi *method sqrt()* adalah....
- A. Mengembalikan nilai masukan berupa akar pangkat tiga
  - B. Mengembalikan nilai masukan berupa akar quadrat
  - C. Mengembalikan nilai masukan berupa nilai tangen
  - D. Memberikan nilai masukan berupa nilai tertinggi
- 5) *Method* matematika yang digunakan untuk mengembalikan nilai logaritmik log basis e (nilai apapun) adalah....
- A. *log1p()*
  - B. *log10()*
  - C. *exp()*
  - D. *log()*
- 6) *Method* yang mengembalikan nilai antara -1 hingga 1 adalah....
- A. *sin()*
  - B. *cos()*
  - C. *tan()*
  - D. *sinh()*
- 7) Nilai *method asin()* adalah mengembalikan nilai antara....
- A.  $-\pi / 2$  dan  $\pi / 2$
  - B.  $-\pi / 2$  dan  $\pi / 2$
  - C. -1 dan 1
  - D.  $-\pi$  dan  $\pi$
- 8) *Method* trigonometri hiperbolik yang didefinisikan sebagai  $(e^x - e^{-x})/2$  adalah *method*....
- A. *sinh()*
  - B. *asin()*
  - C. *sin()*
  - D. *cos()*

- 9) *Method* yang mendefinisikan  $(e^x + e^{-x})/2$  sebagai suatu nilai hiperbolik adalah *method*....
- A. *sin()* / *cos()*
  - B. *cosh()*
  - C. *acos()*
  - D. *cos()*
- 10) *Method* yang dianggap *vers* dari tangen adalah *method*....
- A. *atan()*
  - B. *tanh()*
  - C. *tan()*
  - D. *cos()*

Cocokkanlah jawaban Anda dengan Kunci Jawaban Tes Formatif 3 yang terdapat di bagian akhir modul ini. Hitunglah jawaban yang benar. Kemudian, gunakan rumus berikut untuk mengetahui tingkat penguasaan Anda terhadap materi Kegiatan Belajar 3.

$$\text{Tingkat penguasaan} = \frac{\text{Jumlah Jawaban yang Benar}}{\text{Jumlah Soal}} \times 100\%$$

Arti tingkat penguasaan: 90 - 100% = baik sekali

80 - 89% = baik

70 - 79% = cukup

< 70% = kurang

Apabila mencapai tingkat penguasaan 80% atau lebih, Anda dapat meneruskan dengan modul selanjutnya. **Bagus!** Jika masih di bawah 80%, Anda harus mengulangi materi Kegiatan Belajar 3, terutama bagian yang belum dikuasai.

## Kunci Jawaban Tes Formatif

### Tes Formatif 1

- 1) A
- 2) D
- 3) A
- 4) C
- 5) A
- 6) B
- 7) A
- 8) D
- 9) C
- 10) A

### Tes Formatif 2

- 1) C
- 2) A
- 3) A
- 4) C
- 5) B
- 6) A
- 7) A
- 8) C
- 9) D
- 10) D

### Tes Formatif 3

- 1) B
- 2) B
- 3) D
- 4) B
- 5) D
- 6) B
- 7) A
- 8) A
- 9) B
- 10) A

## Glosarium

**API** : *API* adalah singkatan dari *Application Programming Interface*, dan memungkinkan *developer* untuk mengintegrasikan dua bagian dari aplikasi atau dengan aplikasi yang berbeda secara bersamaan. *API* terdiri dari berbagai elemen seperti *function*, *protocols*, dan *tools* lainnya yang memungkinkan *developers* untuk membuat aplikasi. Tujuan penggunaan *API* adalah untuk mempercepat proses *development* dengan menyediakan *function* secara terpisah sehingga *developer* tidak perlu membuat fitur yang serupa.

Sumber: <https://www.codepolitan.com/mengenal-apa-itu-web-api-5a0c2855799c8>

## Daftar Pustaka

Eckel, B. (2000). *Thinking in java (2<sup>nd</sup>)*. New Jersey: Prentice Hall.

Schildt, H. (2007). *The complete reference java (Seventh Edition)*. New York: Mc Graw Hill.

Flask, R. *Java for Beginners (2<sup>nd</sup>)*.

Eck, D.J. (2006). *Introduction to programming using java (Version 5.0)*. Departement of Mathematic and Computer Science. Geneva, New York 14456.

Horstmann, C.S. & Cornell, G. (2013). *Core java volume I fundamental(Ninth Edition)*. New York: Prentice Hall.

Redko, A. *Adnvanve java preparing you for java mastery*.

<https://docs.oracle.com/javase/7/docs/api/java/lang/StringBuffer.html>

## Praktikum 3

Kani, M.Kom.



### PENDAHULUAN

---

Modul 9 disebut juga dengan Praktikum 3. Dalam panduan praktikum akan dipraktekkan beberapa topik besar yaitu: *Array*, *Method*, *Eksepsi*, dan algoritma-algoritma *sorting*. Secara khusus kita akan mempelajari, memahami dan mempraktekkan bagaimana algoritma *sorting* yang pernah dibuat dengan cara logika berpikirnya.

Pada kegiatan praktikum ini, diharapkan Mahasiswa dapat mengikuti petunjuk untuk menggunakan:

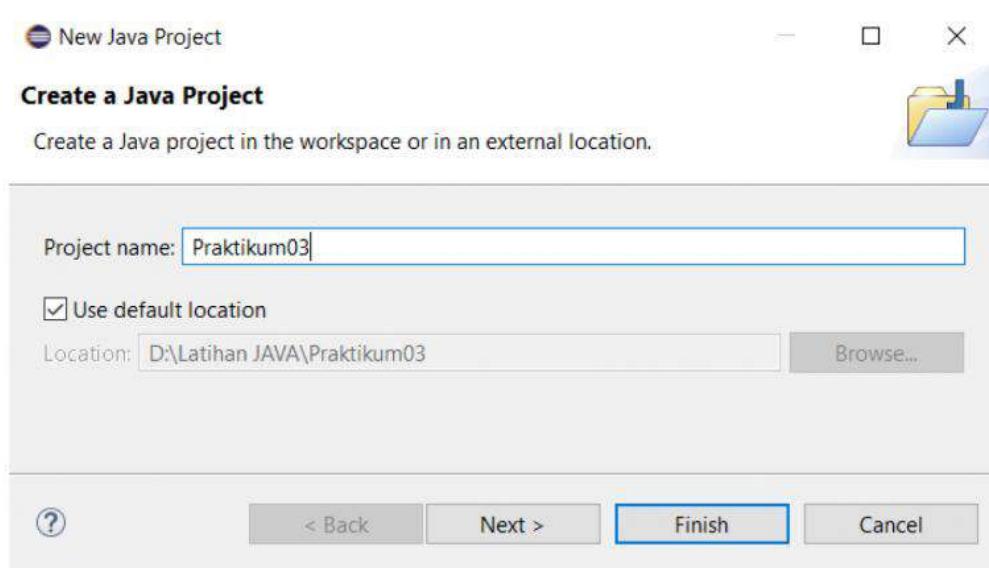
1. *array* satu dimensi;
2. *array* multidimensi;
3. *copy array*;
4. kloning *array*;
5. *Method* dan *method overloading*;
6. *try..catch*;
7. *try..catch* bersarang;
8. *try..catch* multi;
9. *try..catch..finally*;
10. algoritma *Bubble Sort*;
11. algoritma *Selection Sort*;
12. algoritma *Insertion Sort*.

**PRAKTIKUM 3.1*****Array dan Method***

**P**raktikum 3.1 akan membahas 2 topik praktikum yaitu: *Array* dan *Method*. Kedua topik tersebut akan masuk pada projek baru yang diberi nama *Praktikum03*. Dalam projek *Praktikum03* akan memuat 3 Praktikum, yaitu Praktikum 3.1, Praktikum 3.2, dan Praktikum 3.3; namun pada IDE Eclipse hanya akan muncul folder/package setiap topik.

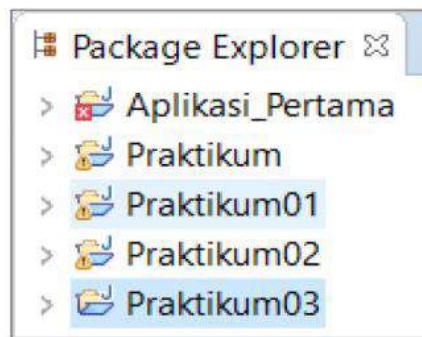
Untuk modul Praktikum 3.1, kita akan buat projek baru dengan nama *Praktikum03* dengan langkah-langkah berikut:

1. Klik menu *File* → *New* → *Java Project*;
2. Pada kolom *Project name*: isi dengan *Praktikum03*



Gambar 9.1  
Pembuatan *Project Praktikum03*

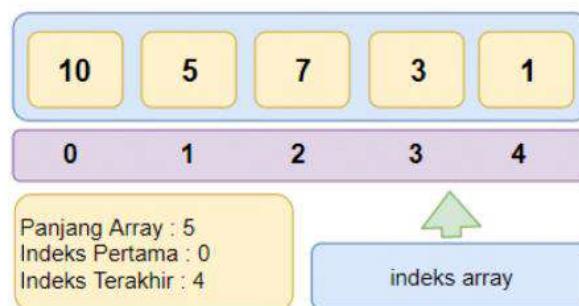
3. Klik tombol *Finish* untuk membuat projek *Praktikum03*
4. Setelah itu akan terbuat *project* baru dengan nama *Praktikum03*, lihat sisi kiri IDE Eclipse bagian *Package Explorer*.



Gambar 9.2  
Daftar Project *Praktikum03*

### A. *ARRAY*

*Array* adalah kumpulan data yang memiliki tipe data yang sama. Setiap data pada *array* memiliki indeks, indeks pada *array* dimulai dari 0, artinya kalau kita memiliki data array [10,5,7,3,1] maka indeksnya adalah 0,1,2,3,4.



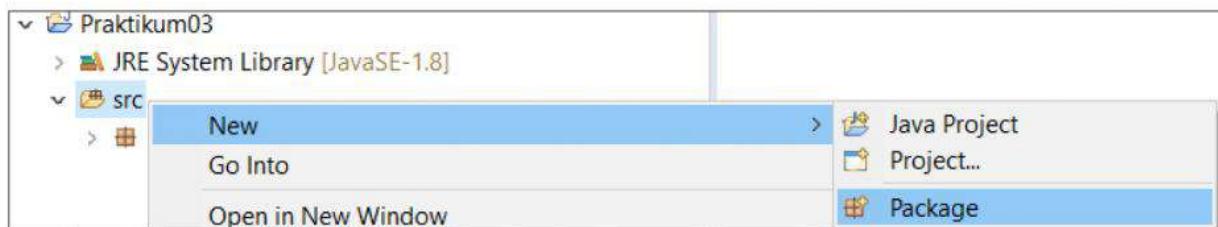
Gambar 9.3  
Ilustrasi Sebuah *Array*

Beberapa pokok bahasan tentang *array* pada bahasa pemrograman Java, adalah sebagai berikut:

- Pada Java semua *array* dialokasikan secara dinamis;
- Pada Java adalah objek; dengan demikian dapat diketahui jumlah anggota.
- Variabel *array* dideklarasikan seperti variabel lain dengan menyebutkan nama *array* setelah tipe data.
- Indeks pada *array* dimulai dari 0
- Pada Java, *array* dapat dibuat *static*, variabel lokal atau menjadi parameter pada sebuah *method*.

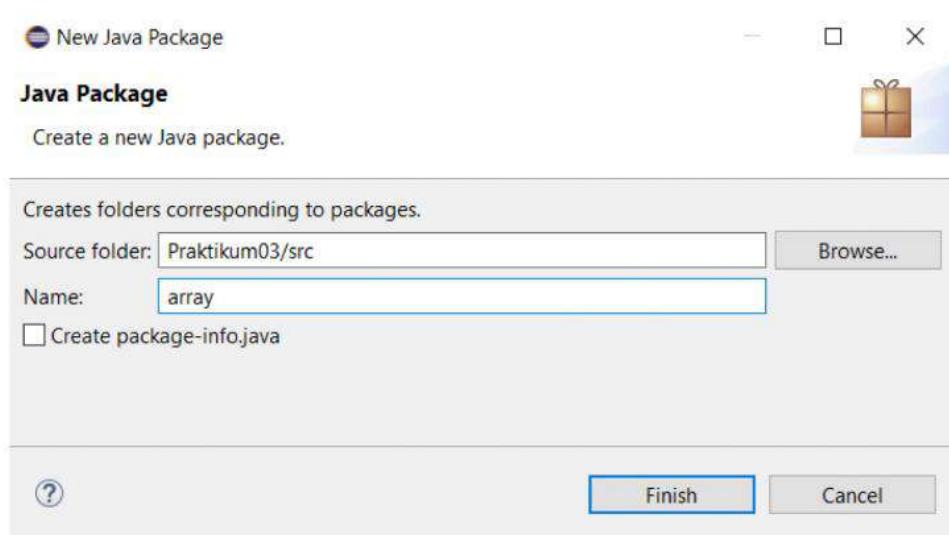
Untuk *Array* kita akan buat folder/package tersendiri pada projek *Praktikum03* dengan langkah-langkah berikut:

- Klik kanan folder *src* → *New* → *Package*



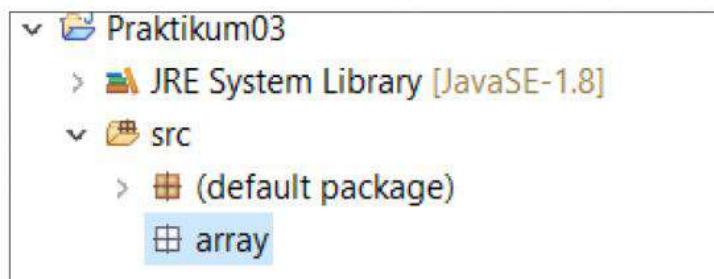
Gambar 9.4  
Pembuatan *Package*

- Kemudian pada window *New Java Package* untuk kolom *Name*: isi dengan *array*.



Gambar 9.5  
Window Pemberian Nama *Package Array*

- c. Klik tombol *Finish* untuk tahap terakhir.



Gambar 9.6  
Package Array yang Berhasil Dibuat

### 1. *Array* satu Dimensi



Gambar 9.7  
Ilustrasi *Array* pada Program

Sintaksis:

```
tipe nama_var_array[];
```

atau

```
tipe[] nama_var_array;
```

- a. *Array* satu Dimensi (menampilkan isi array gaya klasik)

Untuk melihat lebih jelas bagaimana *array* satu dimensi, ikuti langkah-langkah praktikum berikut:

1. Buatlah file *class* dengan nama *arraySatuDimensi.java* pada folder *array* pada projek *Praktikum03*, kemudian jangan lupa menyertakan *main method* pada file *class* tersebut.
2. Ketikkan kode program berikut pada *main method*, untuk deklarasi varabel *array*:

```
// Deklarasi Array integers.  
int[] arrayInt;
```

3. Untuk baris berikutnya adalah ketikkan kode program untuk mengalokasikan memori *array*:

```
// Mengalokasikan memori.  
arrayInt = new int[5];
```

4. Berikutnya kode program untuk menginisiasi *arrayInt* mulai dari indeks 0 hingga indeks 4:

```
// mengisi arrayInt dengan 5 pada indeks 0  
arrayInt[0] = 5;  
  
// mengisi arrayInt dengan 5 pada indeks 1  
arrayInt[1] = 7;  
  
//lainnya  
arrayInt[2] = 4;  
arrayInt[3] = 8;  
arrayInt[4] = 9;
```

5. Untuk baris-baris berikutnya adalah menampilkan *array* dengan pernyataan perulangan *for* sebagai berikut:

```
System.out.println("Daftar Array dalam ArrayInt");  
System.out.println("-----");  
// mengakses data array dengan perulangan for  
for (int i = 0; i < arrayInt.length; i++)  
    System.out.println("Array pada indeks ke-" + i +  
        " : " + arrayInt[i]);
```

6. Kode program secara lengkap sebagai berikut:

```
package array;

class arraySatuDimensi {
    public static void main (String[] args)
    {
        // Deklarasi Array integers.
        int[] arrayInt;

        // Mengalokasikan memori.
        arrayInt = new int[5];

        // mengisi arrayInt dengan 5 pada indeks 0
        arrayInt[0] = 5;

        // mengisi arrayInt dengan 5 pada indeks 1
        arrayInt[1] = 7;

        //lainnya
        arrayInt[2] = 4;
        arrayInt[3] = 8;
        arrayInt[4] = 9;

        System.out.println("Daftar Array dalam ArrayInt");
        System.out.println("-----");
        // mengakses data array dengan perulangan for
        for (int i = 0; i < arrayInt.length; i++)
            System.out.println("Array pada indeks ke-" + i +
                               " : " + arrayInt[i]);
    }
}
```

7. Simpan *class* dan jalankan program  
8. Hasil program sebagai berikut:

```
Daftar Array dalam ArrayInt
-----
Array pada indeks ke-0 : 5
Array pada indeks ke-1 : 7
Array pada indeks ke-2 : 4
Array pada indeks ke-3 : 8
Array pada indeks ke-4 : 9
```

Jika melihat kode program tepat pada potongan kode program berikut:

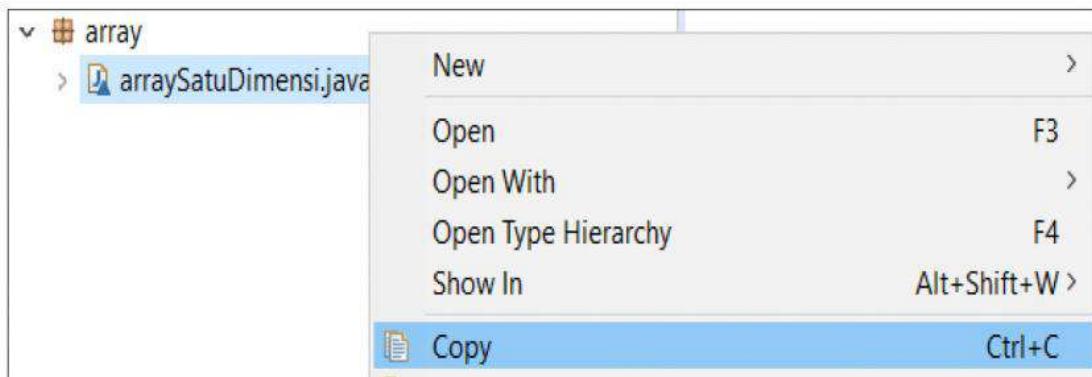
```
for (int i = 0; i < arrayInt.length; i++)
    System.out.println("Array pada indeks ke-" + i +
                        " : " + arrayInt[i]);
```

Potongan kode program tersebut adalah cara menampilkan isi *array* dengan gaya klasik atau lama.

b. *Array satu Dimensi* (menampilkan array dengan gaya sederhana)

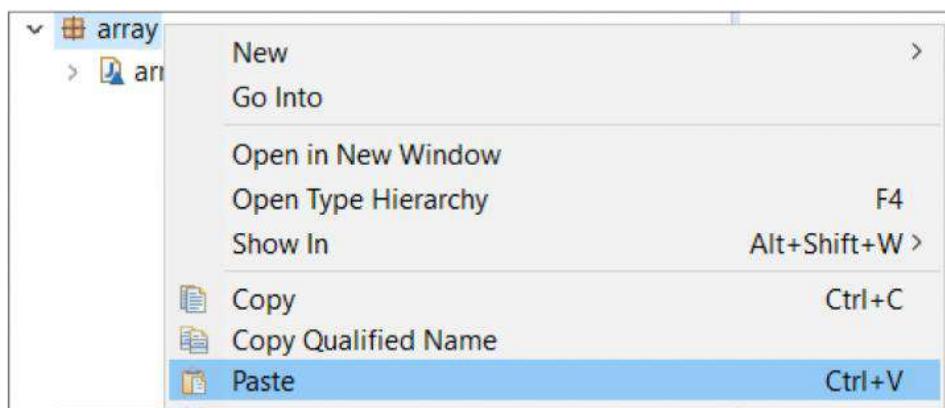
Pada Java telah dikembangkan cara menampilkan isi *array* dengan cara sangat sederhana dan singkat, untuk mempraktekkan lakukan sebagai berikut:

1. Copy file/class java *arraySatuDimensi.java* dengan cara klik kanan tepat pada *class* kemudian pilih menu *copy*.



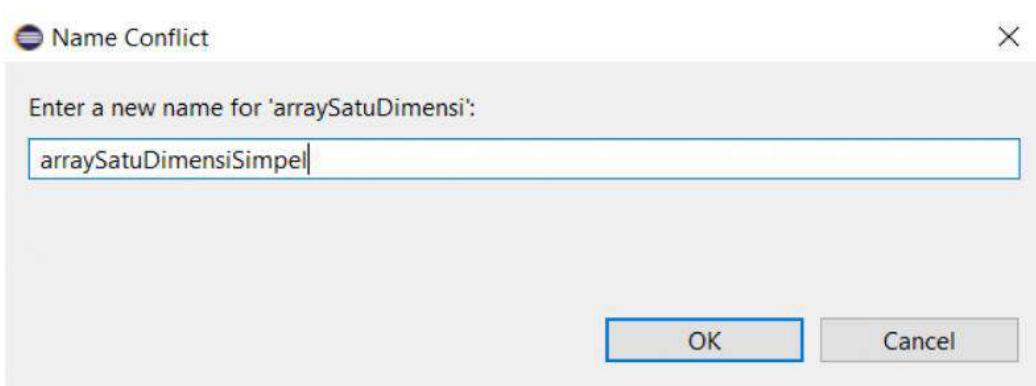
Gambar 9.8  
Copy Class *Arraysatudimensi*

2. Pilih folder/package *array* dan kemudian klik kanan pilih menu *paste*.



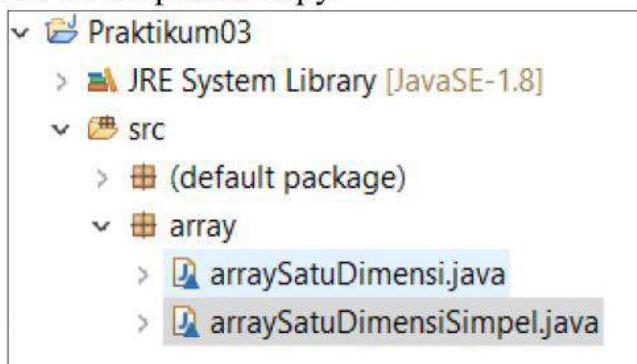
Gambar 9.9  
Paste Class ArraySatudimensi

3. Pada window *NameConflict* ubah nama *arraySatuDimensi* menjadi *arraySatuDimensiSimpel*



Gambar 9.10  
Pemberian Nama Baru Class *arraySatuDimensi* ke *arraySatuDimensiSimpel*

4. Klik tombol *OK* untuk proses copy.



Gambar 9.11  
Class *arraySatuDimensiSimpel*

5. Klik dobel file *class arraySatuDimensiSimpel.java* untuk membuka.

Gambar 9.12  
Kode Program *Class arraySatuDimensiSimpel.java*

6. Sekarang ubah kode program berikut:

```
for (int i = 0; i < arrayInt.length; i++)
    System.out.println("Array pada indeks ke-" + i +
                        " : " + arrayInt[i]);
```

Menjadi:

```
int i=0;
for (int x:arrayInt)
    System.out.println("Array pada indeks ke-" + i++ + " :
    "+ x);
```

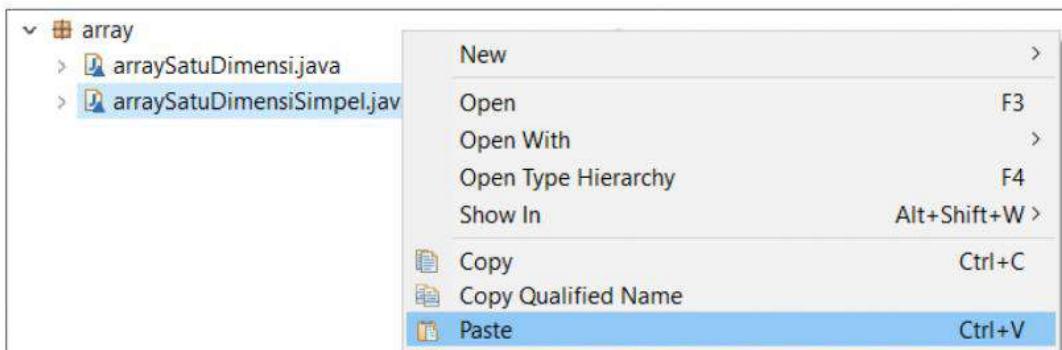
7. Simpan dan jalankan program, hasilnya sama dengan tampilan program *array* sebelumnya.

```
Daftar Array dalam ArrayInt
-----
Array pada indeks ke-0 : 5
Array pada indeks ke-1 : 7
Array pada indeks ke-2 : 4
Array pada indeks ke-3 : 8
```

- c. *Mengurutkan data array satu dimensi dengan class Arrays*

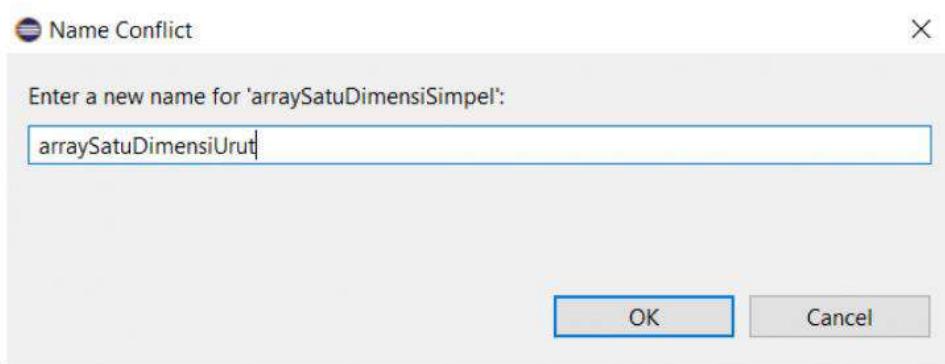
Sekarang kita akan mencoba mengurutkan data *array* dari kecil ke besar dan dari besar ke kecil. Ikuti langkah-langkah berikut:

1. Copy file/class java *arraySatuDimensiSimpel.java* dengan cara klik kanan tepat pada *class* kemudian pilih menu *copy*.
2. Pilih folder/package *array* dan kemudian klik kanan pilih menu *paste*.



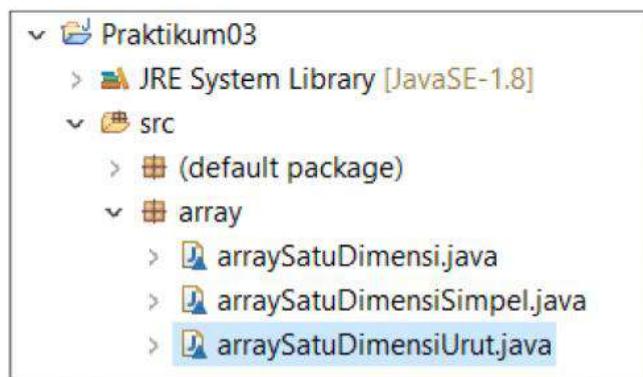
Gambar 9.13  
Menu Paste

3. Pada window *NameConflict* ubah nama *arraySatuDimensiSimpel* menjadi *arraySatuDimensiUrut*



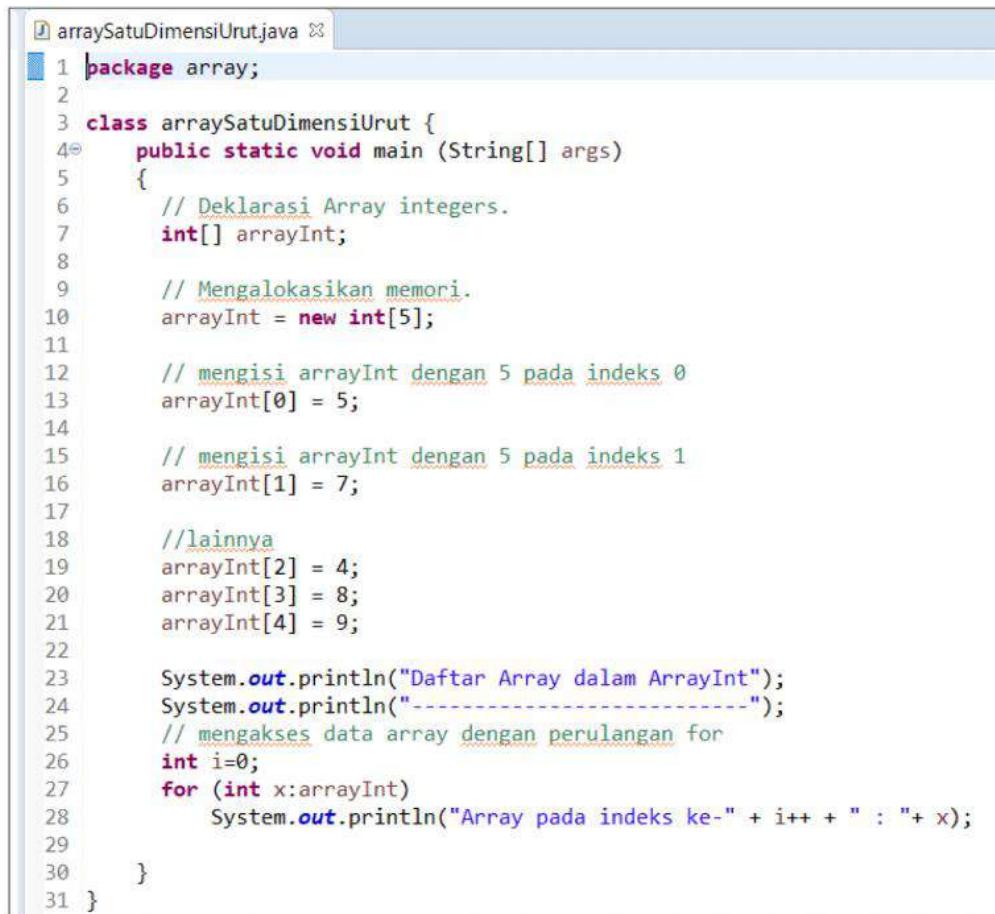
Gambar 9.14  
Pemberian Nama *Class* Baru

4. Klik tombol *OK* untuk proses *copy*.



Gambar 9.15  
*Class* *arraySatuDimensiUrut.java*

5. Klik dobel file *class arraySatuDimensiUrut.java* untuk membuka *class* tersebut.



```

1 package array;
2
3 class arraySatuDimensiUrut {
4     public static void main (String[] args)
5     {
6         // Deklarasi Array integers.
7         int[] arrayInt;
8
9         // Mengalokasikan memori.
10        arrayInt = new int[5];
11
12        // mengisi arrayInt dengan 5 pada indeks 0
13        arrayInt[0] = 5;
14
15        // mengisi arrayInt dengan 5 pada indeks 1
16        arrayInt[1] = 7;
17
18        //lainnya
19        arrayInt[2] = 4;
20        arrayInt[3] = 8;
21        arrayInt[4] = 9;
22
23        System.out.println("Daftar Array dalam ArrayInt");
24        System.out.println("-----");
25        // mengakses data array dengan perulangan for
26        int i=0;
27        for (int x:arrayInt)
28            System.out.println("Array pada indeks ke-" + i++ + " : " + x);
29
30    }
31 }
```

Gambar 9.16  
Kode Program *arraySatuDimensiUrut.java*

6. Lakukan modifikasi sebagai berikut:

- Di bawah *package array;* tambahkan kode program berikut:

```
import java.util.Arrays;
```

Catatan:

*Java.util.Arrays* adalah *class* yang berisi beberapa *method* statis yang dapat digunakan untuk mengisi, menyortir, mencari, dan lain-lain didalam *array*.

- Pada tubuh *main method* tambahkan kode program berikut akhir perintah *for* sebagai berikut:

```
Arrays.sort(arrayInt);
```

```
System.out.println("\nDaftar Array setelah di Sortir");
System.out.println("-----");
// mengakses data array dengan perulangan for
i=0;
for (int x:arrayInt)
System.out.println("Array pada indeks ke-" + i++ + " : " + x);
```

Penjelasan:

- `Arrays.sort(arrayInt);` adalah perintah untuk mengurutkan data *array*
- Kode program setelah kode program poin di atas, adalah perintah untuk menampilkan data *array* setelah di urutkan.

7. Simpan *class* dan coba jalankan, hasilnya sebagai berikut:

```
Daftar Array dalam ArrayInt
-----
Array pada indeks ke-0 : 5
Array pada indeks ke-1 : 7
Array pada indeks ke-2 : 4
Array pada indeks ke-3 : 8
Array pada indeks ke-4 : 9

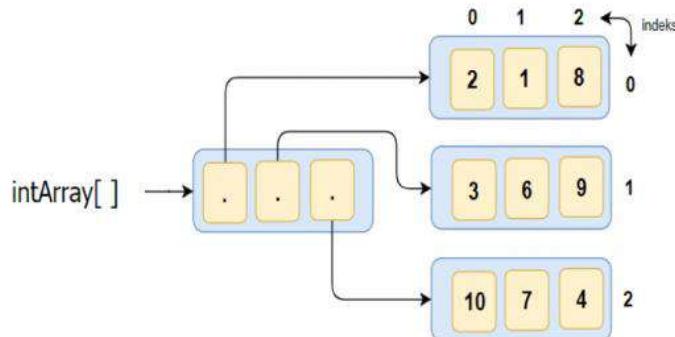
Daftar Array setelah di Sortir
-----
Array pada indeks ke-0 : 4
Array pada indeks ke-1 : 5
Array pada indeks ke-2 : 7
Array pada indeks ke-3 : 8
Array pada indeks ke-4 : 9
```

## 2. Array Multidimensi

*Array multidimensi* adalah perkembangan dari *array* bentuk satu dimensi. *Array multidimensi* adalah variabel yang menyimpan sekumpulan data yang memiliki tipe sama dan elemen yang diakses melalui banyak indeks, biasanya digunakan untuk matriks.

Sintaksis:

```
tipe [] nama_var_array = new tipe[m][m]; //2D
tipe [] nama_var_array = new tipe[m][m][m]; //3D
```



Gambar 9.17  
Ilustrasi Array Multidimensi

Sekarang kita mempraktekkan untuk *array* 2 dimensi dengan langkah-langkah sebagai berikut:

1. Buatlah file *class* dengan nama *arrayDuaDimensi.java* pada folder *array* pada *project Praktikum03*, kemudian jangan lupa menyertakan *main method* pada file *class* tersebut.
2. Ketikkan kode program berikut pada tubuh *main method*:

```
// deklarasi array 2D
int array2D[][] = { {2, 1, 8},
                    {3, 6, 9},
                    {1, 7, 4} };
System.out.println("Tampilkan Array 2D");
System.out.println("-----");

// tampilkan array 2D array
for (int i=0; i < 3 ; i++)
{
    for (int j=0; j < 3 ; j++)
    {
        System.out.print(array2D[i][j] + "   ");
    }
}

System.out.println();
```

```
}
```

3. Kode Program secara utuh sebagai berikut:

```
package array;

public class arrayDuaDimensi {
    public static void main (String[] args)
    {
        // deklarasi array 2D
        int array2D[][] = { {2, 1, 8},
                           {3, 6, 9},
                           {1, 7, 4} };

        System.out.println("Tampilkan Array 2D");
        System.out.println("-----");
        // tampilkan array 2D array
        for (int i=0; i < 3 ; i++)
        {
            for (int j=0; j < 3 ; j++)
            {
                System.out.print(array2D[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

4. Simpan *class* dan jalankan program, hasilnya sebagai berikut:

The screenshot shows a Java application window titled 'arrayDuaDimensi [Java Application]'. The console tab displays the output of a print statement: 'Tampilkan Array 2D' followed by a dashed line and three rows of integers:

```

Tampilkan Array 2D
-----
2 1 8
3 6 9
1 7 4
|
```

Gambar 9.18  
Hasil dari Array Multidimensi

Penjelasan:

- Kode program di bawah adalah mendeklarasikan variabel *array* 2 dimensi sekaligus melakukan inisiasi.

```
// deklarasi array 2D
int array2D[][] = { {2, 1, 8},
                    {3, 6, 9},
                    {1, 7, 4} };
```

- Untuk kode program:

```

for (int i=0; i < 3 ; i++)
{
    for (int j=0; j < 3 ; j++)
    {
        System.out.print(array2D[i][j] + "   ");
    }
    System.out.println();
}
```

- Inti dari program *for* yang bersarang adalah untuk mencetak kolom dan baris dari variabel `array2D[i][j]`.

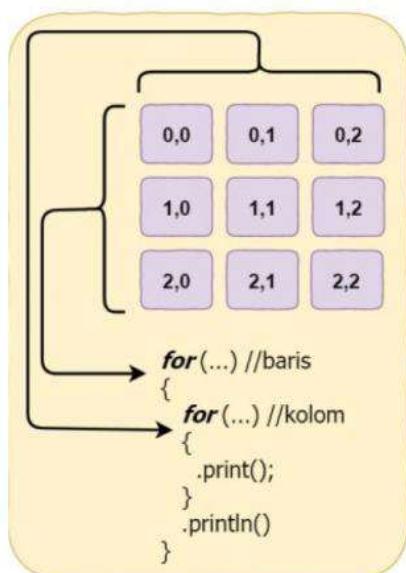
- Untuk *for* yang pertama masih didefinisikan sama halnya dengan *for* yang bersarang.
- Untuk variabel *i* disimpan pada *for* pertama dan untuk variabel *j* disimpan pada *for* kedua, sehingga secara logika bahwa *for* luar akan mencetak baris dan *for* yang kedua akan mencetak kolom.
- Keluaran akan membentuk baris dan kolom karena pada *for* yang kedua dicetak tanpa baris baru (menggunakan perintah *print* bukan *println*):

```
System.out.print(array2D[i][j] + " ");
```

- Kemudian baris baru akan terbentuk setelah perulangan *for* bersarang (kedua) selesai dikerjakan pada setiap perulangan *for* pertama dengan perintah:

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

- Lihat ilustrasi gambar *array* 2 dimensi berikut:



Gambar 9.19  
Array dan Kode Program untuk Menampilkannya

Praktikum berikutnya adalah praktikum *array* 2 dimensi tapi nilainya diinput oleh user, langkah-langkah sebagai berikut:

1. Buatlah file *class* dengan nama *arrayDuaDimensiInput.java* pada folder *array* pada projek *Praktikum03*, kemudian jangan lupa menyertakan *main method* pada file *class* tersebut.
2. Di bawah baris *package array;* tambahkan perintah *import* berikut:

```
import java.util.Scanner;
```

3. Pada tubuh *main method*, tambahkan kode-kode program berikut:
  - Deklarasi *array* dua dimensi dengan pola *array* 3x3.

```
//int array2D[][];  
int[][] array2D = new int[3][3];
```

- Membuat objek input:

```
Scanner input = new Scanner(System.in);
```

- Input nilai *array* dengan *array* 3x3.

```
System.out.println("Input Array 2D");  
System.out.println("-----");  
//input Array 2D  
for (int i=0; i < 3 ; i++)  
{  
    for (int j=0; j < 3 ; j++)  
    {  
        System.out.print("Input nilai  
array["+i+"]["+j+"] = ");  
  
        //proses input nilai array  
        array2D[i][j] = input.nextInt();  
    }  
}
```

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

- Cetak array;

```
System.out.println("Tampilkan Array 2D");
System.out.println("-----");

for (int i=0; i < 3 ; i++)
{
    for (int j=0; j < 3 ; j++)
    {
        System.out.print(array2D[i][j] + "   ");
    }

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

4. Jika dilihat program secara utuh sebagai berikut:

```
package array;

import java.util.Scanner;

public class arrayDuaDimensiInput {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        //int array2D[][];
        int[][] array2D = new int[3][3];
        Scanner input = new Scanner(System.in);

        System.out.println("Input Array 2D");
        System.out.println("-----");
        //input Array 2D
        for (int i=0; i < 3 ; i++)
        {
            for (int j=0; j < 3 ; j++)
            {
                System.out.print("Masukkan nilai
array["+i+"]["+j+"] = ");
                array2D[i][j] = input.nextInt();
            }
        }
    }
}
```

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

    System.out.println("Tampilkan Array 2D");
    System.out.println("-----");

    for (int i=0; i < 3 ; i++)
    {
        for (int j=0; j < 3 ; j++)
        {
            System.out.print(array2D[i][j] + "   ");
        }

        System.out.println();
    }

}
```

5. Simpan dan jalankan program, input berupa angka numerik, hingga hasilnya sebagai berikut:

```
Input Array 2D
-----
Masukkan nilai array[0][0] = 12
Masukkan nilai array[0][1] = 23
Masukkan nilai array[0][2] = 12

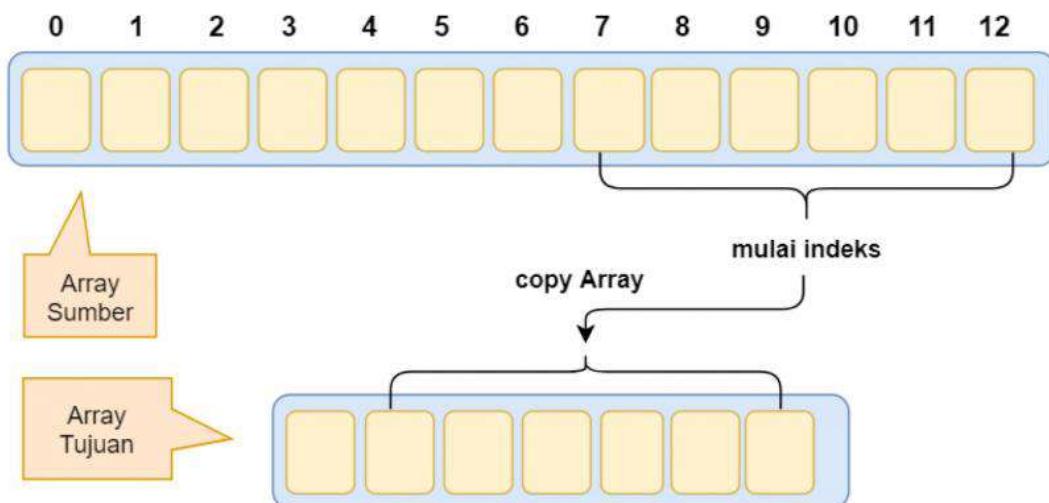
Masukkan nilai array[1][0] = 34
Masukkan nilai array[1][1] = 23
Masukkan nilai array[1][2] = 12

Masukkan nilai array[2][0] = 45
Masukkan nilai array[2][1] = 32
Masukkan nilai array[2][2] = 45

Tampilkan Array 2D
-----
12  23  12
34  23  78
45  32  45
```

### 3. Copy Array

Method yang bernama *arraycopy* ialah *method* yang bisa digunakan untuk mencopy *array* ke lokasi *array* baru.



Gambar 9.21  
Ilustrasi *Copy Array* dari Sumber ke Tujuan

Untuk lebih memahami, ikuti langkah-langkah praktikum berikut:

1. Buatlah file *class* dengan nama *arrayCopy.java* pada folder *array* pada projek *Praktikum03*, kemudian jangan lupa menyertakan *main method* pada file *class* tersebut.
2. Pada tubuh *main method* ketikkan kode-kode program berikut:

```

char[] arraySumber = { 'S', 'e', 'p', 'e', 'd', 'a', ' ', 
                      's', 'a', 'n', 't', 'a', 'i' };
//char[] arrayTujuan = new char[7];
char[] arrayTujuan = { 'L', 'a', 'g', 'i', ' ', 
                      'K', 'o', 'd', 'i', 'n', 
                      'g'};
//cetak arraySumber
System.out.println("Array sumber");
System.out.println("-----");
System.out.println(new String(arraySumber));

//cetak arrayTujuan sebelum disisip
System.out.println("\nArray tujuan sebelum disisip");
System.out.println("-----");

```

```
System.out.println(new String(arrayTujuan));

//Proses copy sebagian data arraySumber
//dan menyisipnya ke arrayTujuan
System.arraycopy(arraySumber, 7, arrayTujuan, 5, 6);

System.out.println("\nArray tujuan setelah disisip");
System.out.println("-----");

//Cetak arrayTujuan setelah menyisipan data
System.out.println(new String(arrayTujuan));
```

1. Kode program secara lengkap:

```
package array;

public class arrayCopy {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        char[] arraySumber = { 'S', 'e', 'p', 'e', 'd',
        'a', ' ', ' ', ' ', ' ', ' ', ' ', 'i' };
        //char[] arrayTujuan = new char[7];
        char[] arrayTujuan = { 'L', 'a', 'g', 'i', ' ', 'K', 'o', 'd', 'i', 'n',
        'g' };
        //cetak arraySumber
        System.out.println("Array sumber");
        System.out.println("-----");
        System.out.println(new String(arraySumber));

        //cetak arrayTujuan sebelum disisip
        System.out.println("\nArray tujuan sebelum
disisip");
        System.out.println("-----");
        System.out.println(new String(arrayTujuan));

        //Proses copy sebagian data arraySumber
        //dan menyisipnya ke arrayTujuan
        System.arraycopy(arraySumber, 7, arrayTujuan,
5, 6);
    }
}
```

```

        System.out.println("\nArray tujuan setelah
disisip");
        System.out.println("-----");
        //Cetak arrayTujuan setelah menyisipan data
        System.out.println(new String(arrayTujuan));

    }
}

```

2. Simpan class dan jalankan program, hasilnya sebagai berikut:

```

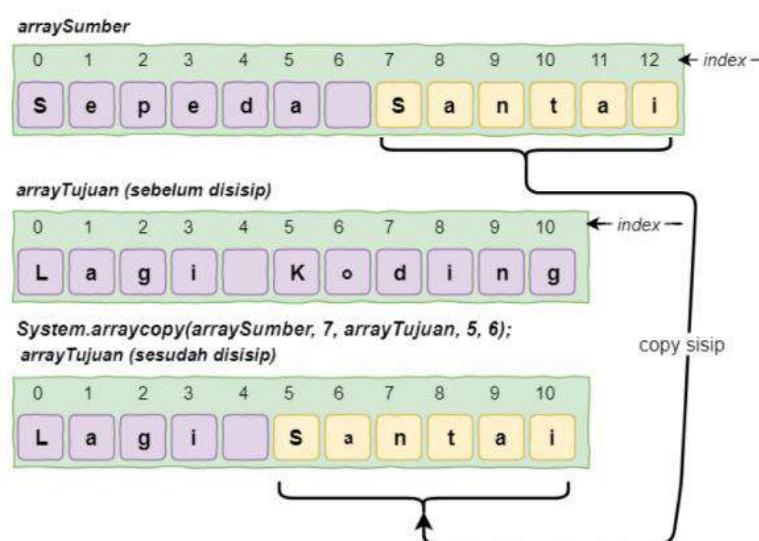
Array sumber
-----
Sepeda Santai

Array tujuan sebelum disisip
-----
Lagi Koding

Array tujuan setelah disisip
-----
Lagi Santai

```

Perhatikan Gambar di bawah sebagai ilustrasi *arraycopy*.

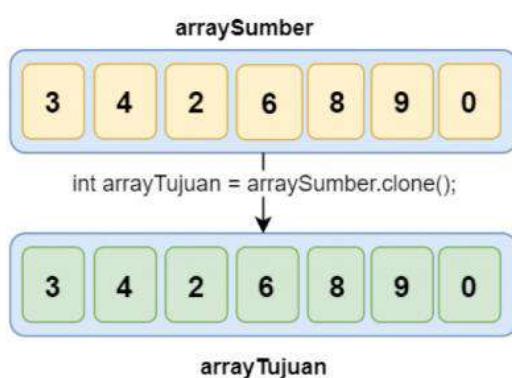


Gambar 9.22  
Ilustrasi Copy Array dari Sumber ke Tujuan

#### 4. Kloning Array

Kloning *array* dapat dilakukan untuk *array* satu dimensi dan juga bisa untuk multi dimensi.

Untuk kloning *array* satu dimensi bisa dilihat ilustrasinya seperti gambar berikut:



Gambar 9.23  
Ilustrasi *Cloning Array* dari Sumber ke Tujuan

Cukup dengan perintah `.clone()`; maka semua isi *array* sumber langsung pindah ke *array* tujuan.

Untuk lebih paham, mari kita praktekkan untuk kloning *array* satu dimensi dengan langkah-langkah berikut:

1. Buatlah file *class* dengan nama *arrayCloningSatuDimensi.java* pada folder *array* pada projek *Praktikum03*, kemudian jangan lupa menyertakan *main method* pada file *class* tersebut.
2. Pada tubuh *main method* ketikkan kode-kode program berikut:

Mendeklarasikan *array* *arraySumber* dan *arrayTujuan* dengan tipe data *integer* sekaligus menginisiasi kedua *array* tersebut:

```
int arraySumber[] = {1,2,3,4,5,6,7,8};  
int arrayTujuan[] = {10,12,13,14,15,16,17,18};
```

Menampilkan isi *array arraySumber*:

```
System.out.println("Nilai arraySumber");
System.out.println("-----");

for (int x:arraySumber)
    System.out.print(x+" ");

System.out.println('\n');
```

Menampilkan isi *array arrayTujuan*:

```
System.out.println("Nilai arrayTujuan sebelum kloning");
System.out.println("-----");

for (int x:arrayTujuan)
    System.out.print(x+" ");
```

Membandingkan *array arraySumber* dengan *arrayTujuan*:

```
System.out.println('\n');
System.out.print("Bandingkan arraySumber[0] vs arrayTujuan[0] =
");

//bandingkan array arraySumber dan arrayTujuan
//pada indeks 0 sebelum kloning
System.out.println(arraySumber[0] == arrayTujuan[0]);
```

Proses kloning data *array arraySumber* ke *arrayTujuan*:

```
//Proses kloning
arrayTujuan = arraySumber.clone();
```

Menampilkan isi *array arrayTujuan*:

```
System.out.println('\n');
```

```

System.out.println("Nilai arrayTujuan setelah kloning");
System.out.println("-----");

for (int x:arrayTujuan)
    System.out.print(x+ " ");

```

Membandingkan kembali isi isi *array arraySumber* dengan *arrayTujuan*:

```

System.out.println('\n');
System.out.print("Bandingkan arraySumber[0] vs arrayTujuan[0] = "
");

//bandingkan array arraySumber dan arrayTujuan
//pada indeks 0 setelah kloning
System.out.println(arraySumber[0] == arrayTujuan[0]);

```

3. Untuk melihat kode program secara utuh sebagai berikut:

```

package array;

public class arrayCloningSatuDimensi {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int arraySumber[] = {1,2,3,4,5,6,7,8};
        int arrayTujuan[] = {10,12,13,14,15,16,17,18};

        System.out.println("Nilai arraySumber");
        System.out.println("-----");

        for (int x:arraySumber)
            System.out.print(x+ " ");

        System.out.println('\n');

        System.out.println("Nilai arrayTujuan sebelum
kloning");
        System.out.println("-----");
        System.out.println("-----");

        for (int x:arrayTujuan)
            System.out.print(x+ " ");
    }
}

```

```

        System.out.println('\n');
        System.out.print("Bandingkan arraySumber[0] vs
arrayTujuan[0] = ");
        //bandingkan array arraySumber dan arrayTujuan
        //pada indeks 0 sebelum kloning
        System.out.println(arraySumber[0] ==
arrayTujuan[0]);

        //Proses kloning
        arrayTujuan = arraySumber.clone();

        System.out.println('\n');
        System.out.println("Nilai arrayTujuan setelah
kloning");
        System.out.println("-----");
        System.out.println("-----");

        for (int x:arrayTujuan)
            System.out.print(x+" ");

        System.out.println('\n');
        System.out.print("Bandingkan arraySumber[0] vs
arrayTujuan[0] = ");
        //bandingkan array arraySumber dan arrayTujuan
        //pada indeks 0 setelah kloning
        System.out.println(arraySumber[0] ==
arrayTujuan[0]);

    }

}

```

4. Simpan *class* dan jalankan programnya, hasilnya sebagai berikut:

```

Nilai arraySumber
-----
1 2 3 4 5 6 7 8

Nilai arrayTujuan sebelum kloning
-----
10 12 13 14 15 16 17 18

Bandingkan arraySumber[0] vs arrayTujuan[0] = false

Nilai arrayTujuan setelah kloning

```

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

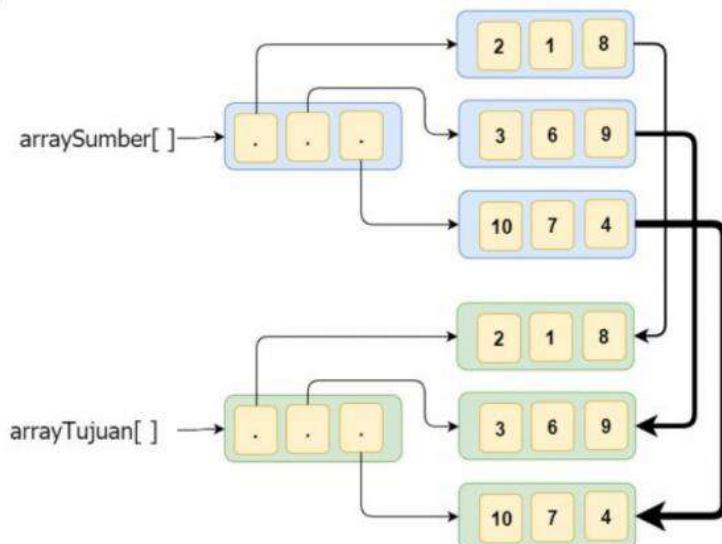
Bandingkan `arraySumber[0]` vs `arrayTujuan[0]` = true

Keterangan:

- Untuk perbandingan data antara `array arraySumber` dengan `arrayTujuan` yang pertama bernilai `false`, karena datanya memang berbeda.
- Untuk perbandingan data antara `array arraySumber` dengan `arrayTujuan` yang kedua bernilai `true`, karena datanya sama setelah proses kloning `array`.

Praktikum berikutnya adalah praktikum kloning data `array` multi dimensi.

Ilustrasi sebagai berikut:



Gambar 9.24  
Ilustrasi Cloning Array dari Sumber ke Tujuan

Untuk praktikum ikuti langkah-langkah berikut:

1. Buatlah file `class` dengan nama `arrayCloningMultiDimensi.java` pada folder `array` pada projek `Praktikum03`, kemudian jangan lupa menyertakan `main method` pada file `class` tersebut.
2. Pada tubuh `main method` ketikkan kode-kode program berikut:

```

int arraySumber[][] = {{1,2,3},{4,5,6},{7,8,9}};
int arrayTujuan[][] = {{0,0,0},{0,0,0},{0,0,0}};

//dibandingkan sebelum kloning
System.out.println("dibandingkan sebelum kloning");
System.out.println("-----");
System.out.println(arraySumber[0][1] ==
arrayTujuan[0][1]);
System.out.println(arraySumber[1] == arrayTujuan[1]);

//proses kloning
arrayTujuan = arraySumber.clone();

System.out.println('\n');
//dibandingkan setelah kloning
System.out.println("dibandingkan setelah kloning");
System.out.println("-----");
System.out.println(arraySumber[0][1] ==
arrayTujuan[0][1]);
System.out.println(arraySumber[1] == arrayTujuan[1]);

```

3. Kode program secara utuh:

```

package array;

public class arrayCloningMultiDimensi {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int arraySumber[][] = {{1,2,3},{4,5,6},{7,8,9}};
        int arrayTujuan[][] = {{0,0,0},{0,0,0},{0,0,0}};

        //dibandingkan sebelum kloning
        System.out.println("dibandingkan sebelum kloning");
        System.out.println("-----");
        System.out.println(arraySumber[0][1] ==
arrayTujuan[0][1]);
        System.out.println(arraySumber[1] ==
arrayTujuan[1]);

        //proses kloning
        arrayTujuan = arraySumber.clone();

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

```

```
//dibandingkan setelah kloning  
System.out.println("dibandingkan setelah kloning");  
System.out.println("-----");  
System.out.println(arraySumber[0][1] ==  
arrayTujuan[0][1]);  
System.out.println(arraySumber[1] ==  
arrayTujuan[1]);  
  
}  
  
}
```

4. Simpan dan jalankan program, hasilnya sebagai berikut:

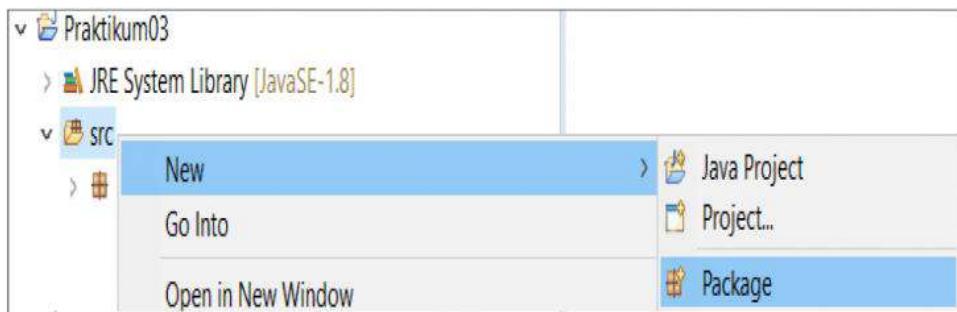
```
dibandingkan sebelum kloning  
-----  
false  
false  
  
dibandingkan setelah kloning  
-----  
true  
true
```

## B. *METHOD*

*Method* adalah sekumpulan pernyataan yang dikelompokkan bersama untuk melakukan tugas-tugas tertentu. Banyak pekerjaan yang secara tidak sadar kita selesaikan dengan cara memanggil suatu *method* di pemrograman Java. Dalam bahasa pemrograman yang lain misalnya pada bahasa pemrograman Pascal, *method* disebut dengan *function* atau *procedure*.

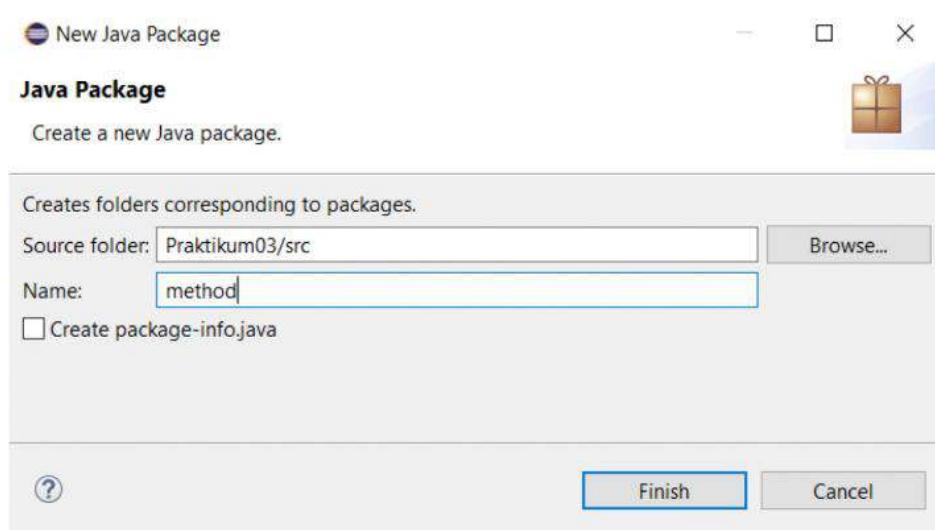
Untuk memisahkan praktikum *method* dengan yang lainnya; mari kita buat folder/package *method* tersendiri pada projek *Praktikum03* dengan langkah-langkah berikut:

1. Klik kanan folder *src* → *New* → *Package*



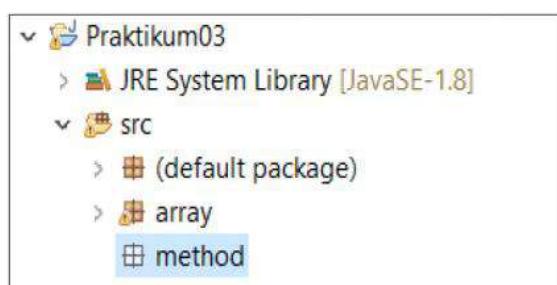
Gambar 9.25  
Menu untuk Membuat Package Baru

2. Kemudian pada window *New Java Package* untuk kolo *Name*: isi dengan *array*.



Gambar 9.26  
Pemberian Nama Package *Method*

3. Klik tombol *Finish* untuk tahap terakhir.



Gambar 9.27  
Package *Method* yang Baru Dibuat

## 1. Membuat dan Memanggil Method

Skema yang kita gunakan dalam praktikum ini adalah membuat class dengan satu proses utuh tanpa memecah masalah menjadi sub *method*, lalu kemudian dari proses yang utuh tadi dipecah masalahnya menjadi sub *method* (sebagian permasalahan dipecahkan pada sub-sub *method*).

Sekarang studi kasusnya adalah keinginan mengetahui data yang diinput melalui keyboard, apakah bilangan ganjil atau genap?

Untuk membuat program tanpa sub *method* dengan langkah-langkah sebagai berikut:

1. Buatlah file *class* dengan nama *mengetahuiAngkaGanjilGenap.java* pada folder *method* pada projek *Praktikum03*, kemudian jangan lupa menyertakan *main method* pada file *class* tersebut.
2. Tambahkan kode program program di bawah *package method*; sebagai berikut:

```
import java.util.Scanner;
```

3. Pada tubuh *main method* ketikkan kode-kode program berikut:

```
Scanner input = new Scanner(System.in);
int x,y;

System.out.print("Masukkan Angka : ");
x = input.nextInt();

y = x % 2;

if (y == 1)
    System.out.println("Angka "+x+" adalah Ganjil");
else
    System.out.println("Angka "+x+" adalah Genap");
```

4. Kode program pada *class* secara utuh sebagai berikut:

```
package method;
```

```

import java.util.Scanner;

public class mengetahuiAngkaGanjilGenap {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner input = new Scanner(System.in);
        int x,y;

        System.out.print("Masukkan Angka : ");
        x = input.nextInt();

        y = x % 2;

        if (y == 1)
            System.out.println("Angka "+x+" adalah
Ganjil");
        else
            System.out.println("Angka "+x+" adalah
Genap");

    }

}

```

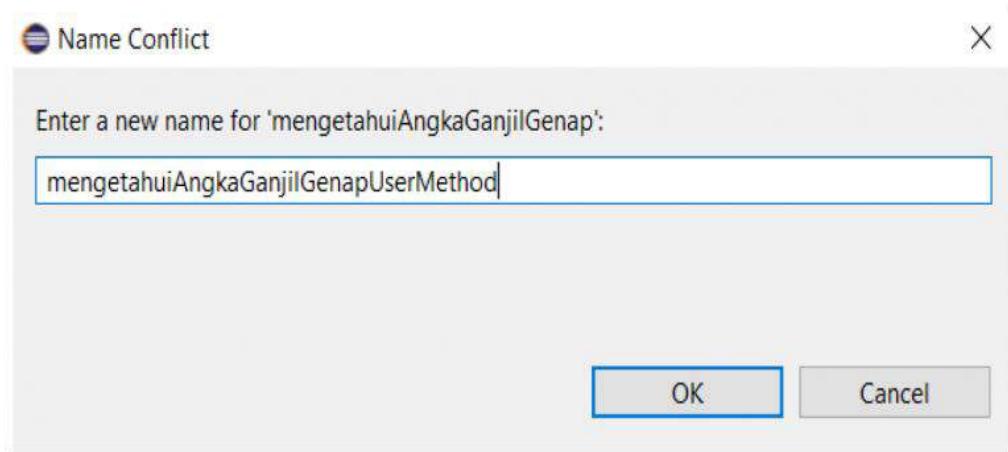
5. Simpan dan jalankan program, hasilnya sebagai berikut:  
Coba Anda masukkan angka 11 pada program tersebut:

Masukkan Angka : 11  
Angka 11 adalah Ganjil

Sekarang program yang sudah kita tadi, kita modifikasi menjadi *method* dan cara memanggilnya. Yang akan kita pecah menjadi sebuah *method* adalah proses penentuan apakah angka yang dimasukkan adalah ganjil atau genap.

Untuk membuatnya, ikuti langkah-langkah berikut:

1. Klik kanan pada file *class mengetahuiAngkaGanjilGenap.java*, kemudian pilih menu *Copy*.
2. Kemudian klik kanan pada folder/package *method* dan pilih menu *Paste*.
3. Pada window *Name Conflict* ganti nama menjadi *mengetahuiAngkaGanjilGenapUseMethod* dan selanjutnya klik tombol *OK*.



Gambar 9.28

Mengganti Nama Class Menjadi *mengetahuiAngkaGanjilGenapUseMethod*

4. Setelah file class *mengetahuiAngkaGanjilGenapUseMethod.java* terbentuk, klik dobel file tersebut sampai terbuka.

```
mengetahuiAngkaGanjilGenapUserMethod.java
1 package method;
2
3 import java.util.Scanner;
4
5 public class mengetahuiAngkaGanjilGenapUserMethod {
6
7     public static void main(String[] args) {
8         // TODO Auto-generated method stub
9         Scanner input = new Scanner(System.in);
10        int x,y;
11
12        System.out.print("Masukkan Angka : ");
13        x = input.nextInt();
14
15        y = x % 2;
16
17        if (y == 1)
18            System.out.println("Angka "+x+" adalah Ganjil");
19        else
20            System.out.println("Angka "+x+" adalah Genap");
21
22    }
23
24 }
```

Gambar 9.29

Kode Program dari Class *mengetahuiAngkaGanjilGenapUseMethod*

5. Sekarang buatlah *method* baru dengan nama *menentukanAngkaGanjilGenap* dan dibuat di dalam *class* utama dan di luar *main method* sebagai berikut (terserah mau simpan sebelum dan sesudah *main method*):

```
Public static String menentukanAngkaGanjilGenap (int x)
{
    int y;
    y = x % 2;

    if (y == 1)
        return "Ganjil";
    else
        return "Genap";
}
```

Penjelasan:

- *Method* yang dibuat di atas adalah *method* berparameter variabel *x* dengan tipe data *integer* (**int** *x*).
  - Nilai balikan dari *method* adalah *String*.
  - *Modifiers* dari *method* adalah *public*.
6. Kemudian lakukan modifikasi pada *main method* sebagai berikut:  
Dari kode program:

```
Scanner input = new Scanner(System.in);
int x,y;

System.out.print("Masukkan Angka : ");
x = input.nextInt();

y = x % 2;

if (y == 1)
    System.out.println("Angka "+x+" adalah Ganjil");
else
    System.out.println("Angka "+x+" adalah Genap");
```

Menjadi:

```
Scanner input = new Scanner(System.in);
```

```
int x;

System.out.print("Masukkan Angka : ");
x = input.nextInt();

System.out.println("Angka "+x+" adalah "+
menentukanAngkaGanjilGenap(x));
```

Keterangan:

- Pada kode program yang bercetak tebal adalah proses pemanggilan *method*.
7. Untuk melihat kode program secara utuh sebagai berikut:

```
package method;

import java.util.Scanner;
public class mengetahuiAngkaGanjilGenapUserMethod {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner input = new Scanner(System.in);
        int x;

        System.out.print("Masukkan Angka : ");
        x = input.nextInt();

        System.out.println("Angka "+x+" adalah "+
            menentukanAngkaGanjilGenap(x));
    }

    //Method baru
    public static String menentukanAngkaGanjilGenap (int x)
    {
        int y;
        y = x % 2;

        if (y == 1)
            return "Ganjil";
        else
            return "Genap";
    }
}
```

8. Simpan modifikasi dan jalankan program, hasilnya sebagai berikut:  
Masukkan angka 11:

```
Masukkan Angka : 11
Angka 11 adalah Ganjil
```

Gambar di bawah ini menampilkan blok antara *main method* dengan *method menentukanAngkaGanjilGenap*.

```

1 package method;
2
3 import java.util.Scanner;
4
5 public class mengetahuiAngkaGanjilGenapUserMethod {
6
7     public static void main(String[] args) {
8         // TODO Auto-generated method stub
9         Scanner input = new Scanner(System.in);
10        int x;
11
12        System.out.print("Masukkan Angka : ");
13        x = input.nextInt();
14
15
16        System.out.println("Angka "+x+" adalah "+ menentukanAngkaGanjilGenap(x));
17    }
18
19
20    public static String menentukanAngkaGanjilGenap (int x)
21    {
22        int y;
23        y = x % 2;
24
25        if (y == 1)
26            return "Ganjil";
27        else
28            return "Genap";
29    }
30
31 }
```

The code shows a Java program with two methods: `main` and `menentukanAngkaGanjilGenap`. The `main` method prompts the user for an integer and prints the result of calling the `menentukanAngkaGanjilGenap` method. The `menentukanAngkaGanjilGenap` method takes an integer `x` and returns a string indicating whether it is odd ('Ganjil') or even ('Genap'). Hand-drawn blue boxes highlight the `main` method block and the `menentukanAngkaGanjilGenap` method block. A callout bubble points from the `menentukanAngkaGanjilGenap` method block to the text 'blok method dan pemanggilan method'.

Gambar 9.30  
Memanggil Sebuah *Method*

Hasil dari kedua program antara *mengetahuiAngkaGanjilGenap.java* dan *mengetahuiAngkaGanjilGenapMethod.java* adalah sama, namun pada program *mengetahuiAngkaGanjilGenapMethod.java* kita gunakan *method*. Mudah-mudahan sampai disini Anda sudah memahami bagaimana cara membuat *method* dan cara memanggilnya.

## 2. Membuat *method* yang tidak mengembalikan nilai

Cukup sederhana agar sebuah *method* tidak mengembalikan nilai, yaitu dengan menambahkan *keyword void* pada awal nama *method*. Untuk contoh, lakukan langkah-langkah berikut:

1. Buatlah file *class* dengan nama *pemanggilanMethodTanpaReturnValue.java* pada folder *method* pada projek *Praktikum03*, kemudian jangan lupa menyertakan *main method* pada file *class* tersebut.
2. Buat *method* baru dalam tubuh *class* dengan nama *method* *cariNilaiGanjilGenap* sebagai berikut:

```
public static void cariNilaiGanjilGenap(int nilai) {
    nilai = nilai % 2;
    if (nilai == 1)
        System.out.println("Nilai Ganjil");
    else
        System.out.println("Nilai Genap");
}
```

Penjelasan:

- Dengan munculnya *keyword void* di awal nama *method* menyebabkan *method* tidak boleh mengembalikan nilai.
3. Pada tubuh *main method* ketikkan kode-kode program berikut:

```
cariNilaiGanjilGenap(2);
cariNilaiGanjilGenap(3);
cariNilaiGanjilGenap(16);
```

4. Program utuh bisa dilihat sebagai berikut:

```
package method;

public class pemanggilanMethodTanpaReturnValue {
    public static void cariNilaiGanjilGenap(int nilai) {
```

```

        nilai = nilai % 2;
        if (nilai == 1)
            System.out.println("Nilai Ganjil");
        else
            System.out.println("Nilai Genap");
    }

public static void main(String[] args) {
    // TODO Auto-generated method stub
    cariNilaiGanjilGenap(2);
    cariNilaiGanjilGenap(3);
    cariNilaiGanjilGenap(16);

}

```

atau

```

package method;
public class pemanggilanMethodTanpaReturnValue { keyword void menyebabkan
method ini tak punya nilai balikan
    public static void cariNilaiGanjilGenap(int nilai) {
        nilai = nilai % 2;
        if (nilai == 1)
            System.out.println("Nilai Ganjil");
        else
            System.out.println("Nilai Genap");
    }

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        cariNilaiGanjilGenap(2);
        cariNilaiGanjilGenap(3);
        cariNilaiGanjilGenap(16);
    }
}

```

Gambar 9.31  
Sebuah Method yang Tidak Perlu Mengembalikan Nilai

5. Simpan dan jalankan program, hasilnya sebagai berikut:

```
Nilai Genap  
Nilai Ganjil  
Nilai Genap
```

### 3. *Method Overloading*

Kemampuan sebuah *class* mempunyai 2 atau lebih *method* dengan nama yang sama dan yang membedakan adalah parameter setiap *method*.

Untuk memahami lebih cepat, praktikkan langkah-langkah berikut:

1. Buatlah file *class* dengan nama *methodOverloading.java* pada folder *method* pada projek *Praktikum03*, kemudian jangan lupa menyertakan *main method* pada file *class* tersebut.
2. Buat *method* yang pertama dengan nama *cariLuas* sebagai berikut:

```
// Overloaded tambahData String  
public static String tambahData(String datanya) {  
    return datanya+ " + empat = delapan";  
}
```

3. Sekarang buat *method* yang kedua dengan nama *method* yang sama seperti di atas (*cariLuas*):

```
// Overloaded tambahData Integer  
public static int tambahData(int panjang, int lebar) {  
    return (panjang + lebar);  
}
```

4. Sekarang pada *main method* ketikkan kode program berikut:

```
//Panggil method tambahData  
System.out.println("1. "+tambahData("empat"));  
System.out.println("2. 4 + 4 = "+tambahData(4,4));
```

5. Untuk kode program secara lengkap sebagai berikut:

```

package method;

public class methodOverloading {

    // Overloaded tambahData String
    public static String tambahData(String datanya) {
        return datanya+ " + empat = delapan";
    }

    // Overloaded tambahData Integer
    public static int tambahData(int panjang, int lebar) {
        return (panjang + lebar);
    }

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        //Panggil method tambahData
        System.out.println("1. "+tambahData("empat"));
        System.out.println("2. 4 + 4 = "+tambahData(4,4));
    }
}

```

6. Simpan dan jalankan program, hasilnya sebagai berikut:

1. Empat + empat = delapan
2. 4 + 4 = 8

Penjelasan:

- *Method tambahData* yang pertama memberikan nilai balik *String* dengan satu parameter bertipe data *String*.
- *Method tambahData* yang kedua adalah memberikan nilai balik *integer* dengan dua parameter bertipe data *integer*.
- Dua *method* dengan nama yang sama akan tetapi memiliki nilai balik yang berbeda dan parameter yang berbeda, akan tetapi bisa digunakan dalam *class* yang sama dan dibuat pada *class* yang sama.

LATIHAN

---

Untuk memperdalam pemahaman Anda mengenai materi di atas, kerjakanlah latihan berikut!

- 1) Sebuah data *array* sebagai berikut:

$a[] = \{20, 23, 40, 31, 56, 67, 100, 49, 57\}$

Buatlah hasilnya dengan tampilan sebagai berikut:

- a. Tampilkan isi *array* apa adanya

20

23

40

31

56

67

100

49

57

- b. Tampilkan isi *array* urut dari terkecil ke terbesar

20

23

31

40

49

56

57

67

100

- c. Tampilan sebelah kiri isi *array* aslinya dan kemudian tampilan sebelah kanan *array* sudah dalam keadaan urut.

20 20

23 23

40 31

31 40

56 49

67 56  
100 67  
49 100

- 2) Sebuah *array* sebagai berikut:

String huruf[] = {"A","B","C","D","E","F","G","H","I","J","L","M","O"}

- a. Tampilkan seperti berikut:

ABCDE  
FGHI  
JKL  
MN  
O

- b. A F K  
B G L  
C H M  
D I N  
E J O

#### Petunjuk Jawaban Latihan

- 1) a Untuk menampilkan *array* seperti berikut:

20  
23  
40  
31  
56  
67  
100  
49  
57

Kode programnya sebagai berikut:

```
public class tampilanArray01 {  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        int a[] = {20, 23, 40, 31, 56, 67, 100, 49,  
57};
```

```
    for (int i=0;i<=8;i++)
    {
        System.out.println(a[i]);
    }
}
```

- b Untuk menampilkan *array* seperti berikut:

20  
23  
31  
40  
49  
56  
57  
67  
100

Kode programnya sebagai berikut:

```
public class tampilanArray02 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int a[] = {20, 23, 40, 31, 56, 67, 100, 49,
57};
        Arrays.sort(a);

        for (int i=0;i<=8;i++)
        {
            System.out.println(a[i]);
        }
    }
}
```

- c Untuk menampilkan *array* seperti berikut:

20 20  
23 23  
40 31  
31 40

56 49  
67 56  
100 67  
49 100

Kode programnya sebagai berikut:

```
import java.util.Arrays;

public class tampilanArray01 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int a[] = {20, 23, 40, 31, 56, 67, 100, 49,
57};
        int b[];
        b = new int[8];
        System.arraycopy(a, 0, b, 0, 8);
        Arrays.sort(b);
        for (int i=0;i<=8;i++)
        {
            System.out.println(a[i] + " " + b[i]);
        }
    }
}
```

- 2) a Untuk menampilkan sebagai berikut:

ABCDE  
FGHI  
JKL  
MN  
O

Kode programnya sebagai berikut:

```
public class arrayHuruf {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        String huruf[] =
        {"A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K", "L", "M", "N", "O"};
        int n = 0;
        for (int i = 1; i <= 5; i++)
        {
            for (int j=1;j <=6-i;j++)
```

```
{  
    System.out.print(huruf[n]);  
    n = n + 1;  
}  
System.out.println("");  
}  
}  
}
```

- b Untuk menampilkan seperti berikut:
- A F K  
B G L  
C H M  
D I N  
E J O

Kode programnya sebagai berikut:

```
public class arrayHuruf1 {  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        String huruf[] =  
  
        {"A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K", "L", "M", "N", "O"};  
        int n = 0;  
        while(n<=4)  
        {  
            System.out.println(huruf[n]+ " "+huruf[n+5]+ "  
            "+huruf[n+10]);  
            n = n + 1;  
        }  
    }  
}
```



## RANGKUMAN

---

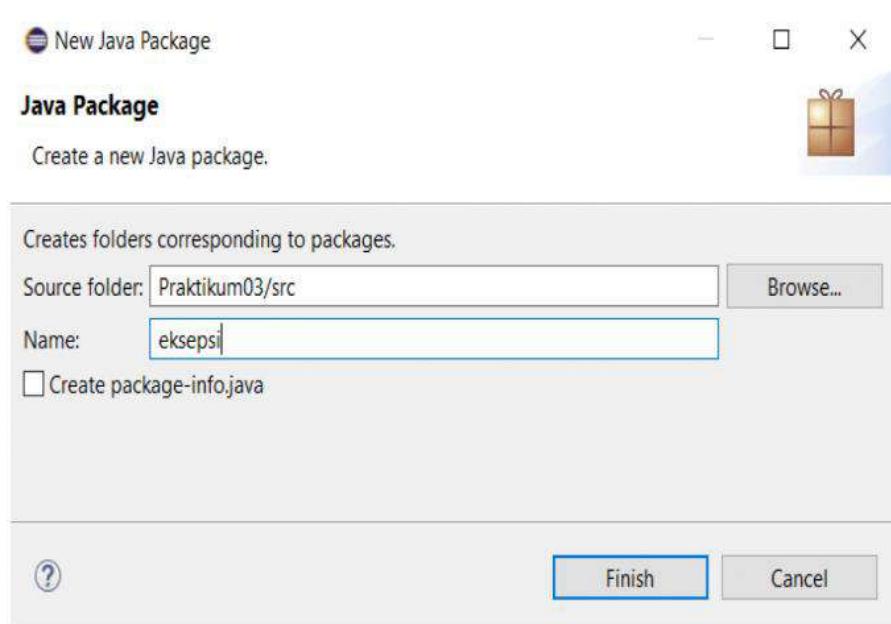
*Array* adalah kumpulan data yang memiliki tipe data yang sama. setiap data pada *array* memiliki indeks, indeks pada *array* dimulai dari 0, artinya kalau kita memiliki data *array*  $[10, 5, 7, 3, 1]$  maka indeksnya adalah 0,1,2,3,4.

*Method* adalah sekumpulan pernyataan yang dikelompokkan bersama untuk melakukan tugas-tugas tertentu. Banyak pekerjaan yang secara tidak sadar kita selesaikan dengan cara memanggil suatu *method* di pemrograman Java, dalam bahasa pemrograman yang lain misalnya pada bahasa pemrograman Pascal, *method* disebut dengan *function* atau *procedure*.

**PRAKTIKUM 3.2****Penanganan Kesalahan**

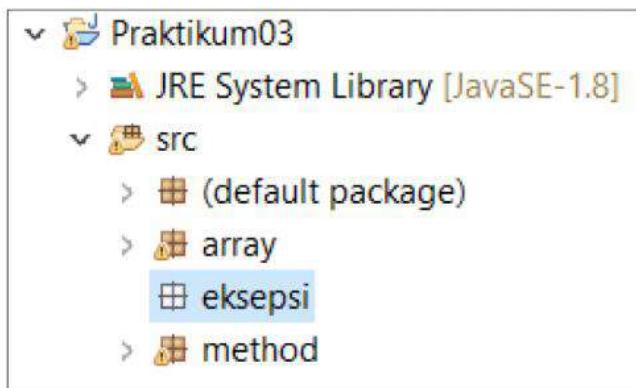
• Pada Praktikum 3.2 akan dipraktekkan sintaks untuk penanganan kesalahan pada Java. Beberapa akan dipraktekkan adalah *try..catch*, *try..catch multi*, *try..catch bersarang*, *try..catch..finally*. Untuk praktikum selanjutnya kita membuat folder/package *eksepsi* dengan langkah-langkah berikut ini:

1. Buat folder/package dengan nama *eksepsi* pada folder *src* dalam projek *Praktikum03*. Isi kolom *name* dengan *eksepsi*



Gambar 9.32  
Membuat Package Eksepsi

2. Klik tombol *Finish* untuk membuat *package*.



Gambar 9.33  
Package Eksepsi Yang Sudah Jadi

### A. EKSEPSI TRY..CATCH

Sintaks:

```
try {
    // Pernyataan yang mungkin atau dicurigai
    // akan mengalami error
}
catch( TipeException VariabelException)
{
    //blok pernyataan yang dijalankan
    //jika terjadi error
}
```

Penjelasan:

- *try* menyatakan bahwa dalam blok pernyataan dapat terjadi suatu eksepsi dan jika kemungkinan itu terjadi, maka jalankan blok program dalam blok *catch* sesuai dengan tipe eksepsi yang terjadi.
- *Instance variable VariableException* dari *class Exception* dapat digunakan sebagai referensi/data informasi untuk mengetahui penyebab terjadinya eksepsi tersebut.

Sekarang ikuti langkah-langkah berikut untuk menguji coba. Pertama kita membuat program tanpa *try..catch* sebagai berikut:

1. Buatlah file *class* dengan nama *eksepsiTryCatch.java* pada folder *eksepsi* pada projek *Praktikum03*, kemudian jangan lupa menyertakan *main method* pada file *class* tersebut.
2. Pada tubuh *main method* ketikkan kode program berikut:

```
String sepeda[] = new String[4];

sepeda[0] = "Sepeda Santai";
sepeda[1] = "Sepeda Gunung";
sepeda[2] = "Sepeda Balap";
sepeda[3] = "Sepeda Ontel";

System.out.println(sepeda[4]);
```

3. Kode program secara utuh sebagai berikut:

```
package eksepsi;

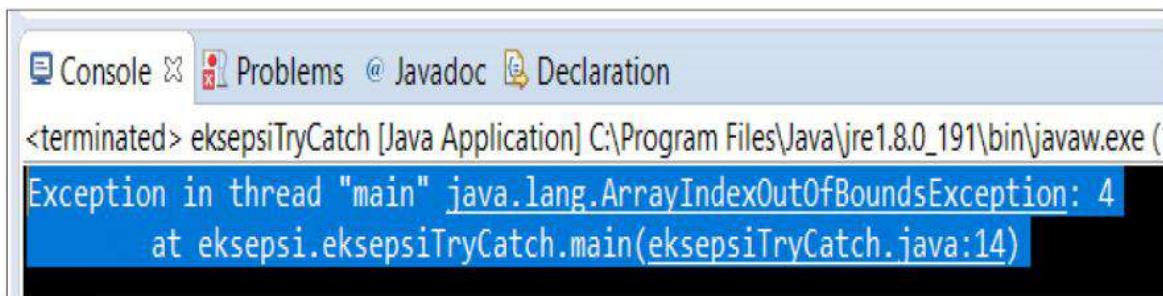
public class eksepsiTryCatch {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        String sepeda[] = new String[4];

        sepeda[0] = "Sepeda Santai";
        sepeda[1] = "Sepeda Gunung";
        sepeda[2] = "Sepeda Balap";
        sepeda[3] = "Sepeda Ontel";

        System.out.println(sepeda[4]);
    }
}
```

4. Simpan program dan jalankan, hasilnya sebagai berikut:



Gambar 9.34  
Contoh *Exception Error*

Penjelasan:

Muncul pesan kesalahan bahwa indek *array* 4 tidak ada. Nah bagaimana menangani kesalahan di atas, ikuti langkah selanjutnya.

5. Ubah bagian:

```
System.out.println(sepeda[4]);
```

Menjadi:

```
try {
    System.out.println(sepeda[4]);
}
catch (Exception e)
{
    System.out.println("Awas! Ada kesalahan akses alamat memori");
}
```

6. Simpan program dan jalankan, hasilnya sebagai berikut:

```
Awas! Ada kesalahan akses alamat memori
```

Penjelasan:

Kesalahan yang terjadi seperti pada langkah 4 diperangkap oleh *try..catch*.

7. Jika kode program `System.out.println(sepeda[4]);` diubah menjadi `System.out.println(sepeda[3]);` maka jalankannya program akan normal.

Sepeda Ontel

8. Program secara lengkap setelah dimodifikasi dengan penambahan eksepsi *try..catch* sebagai berikut:

```
package eksepsi;

public class eksepsiTryCatch {

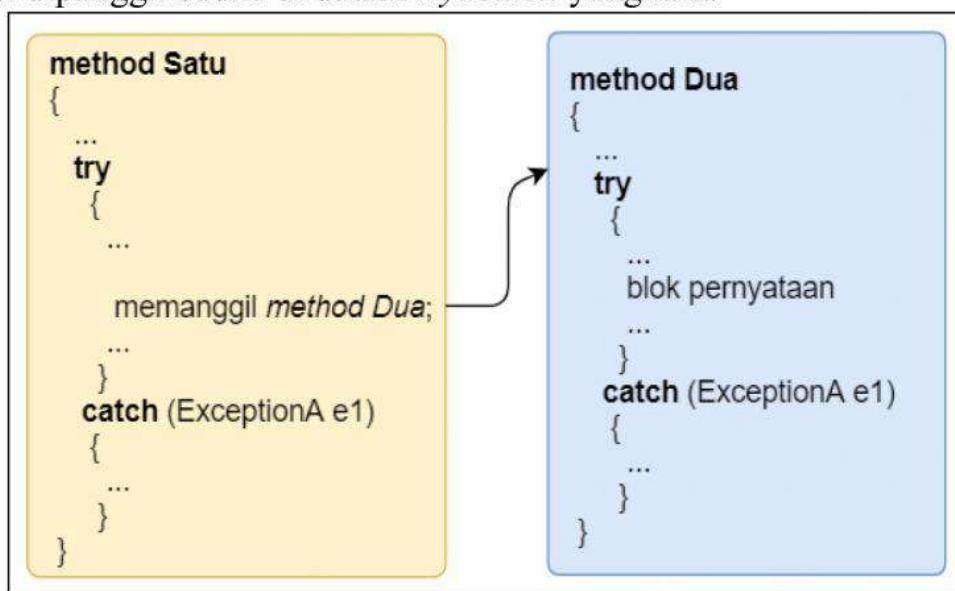
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        String sepeda[] = new String[4];

        sepeda[0] = "Sepeda Santai";
        sepeda[1] = "Sepeda Gunung";
        sepeda[2] = "Sepeda Balap";
        sepeda[3] = "Sepeda Ontel";

        try {
            System.out.println(sepeda[4]);
        }
        catch (Exception e)
        {
            System.out.println("Awas! Ada kesalahan alamat
memori");
        }
    }
}
```

## B. MEMBUAT TRY..CATCH BERSARANG

Eksepsi *try..catch* bersarang adalah *try..catch* yang dipasang dalam *try..catch*. *try..catch* bisa jadi dipasang pada sebuah *method* dan ketika *method* tersebut dipanggil sudah di dalam *try..catch* yang lain.



Gambar 9.35  
Ilustrasi *try except* dalam Sebuah *try except*

Untuk mempraktekkan kita ikuti langkah-langkah berikut:

1. Buatlah file *class* dengan nama *eksepsiTryCatchBersarang.java* pada folder *eksepsi* pada projek *Praktikum03*, kemudian jangan lupa menyertakan *main method* pada file *class* tersebut.
2. Buat *method* baru dengan nama *methodTryCatch* sebagai berikut:

```

//Method try.catch (bersarang)
public static boolean methodTryCatch(int a, int b)
{
    boolean boolean_nilai = false;
    int jumlahTerjual[] = new int[2];
    jumlahTerjual[0] = 3;
    jumlahTerjual[1] = 4;

    try
    {
        if (jumlahTerjual[0] > jumlahTerjual[2])
        {
            boolean_nilai = true;
        }
    }
}

```

```
        }

    }

    catch (Exception e2)
    {
        System.out.println("Jebak kesalahan Dari sub-
method");
        System.out.println("Awas! terjadi kesalahan");
    }

    return boolean_nilai;
}
```

3. Pada tubuh *main method* ketikkan beberapa kode program sekaligus kode program untuk memanggil *method methodTryCatch* sebagai berikut:

```
boolean hasil_boolean = false;
int jumlahSepeda[] = new int[4];
jumlahSepeda[0] = 3;
jumlahSepeda[1] = 4;
jumlahSepeda[2] = 1;
jumlahSepeda[3] = 3;

try {
    hasil_boolean =
MethodTryCatch(jumlahSepeda[1],jumlahSepeda[3]);
    System.out.println("Hasil Boolean :
"+hasil_boolean);
}
catch (Exception e)
{
    System.out.println("Jebak kesalahan dari method
utama");
    System.out.println("Awas! Ada kesalahan alamat
memori");
}
```

4. Kode program secara utuh sebagai berikut:

```
package eksepsi;

public class eksepsiTryCatchBersarang {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        boolean hasil_boolean = false;
        int jumlahSepeda[] = new int[4];
        jumlahSepeda[0] = 3;
        jumlahSepeda[1] = 4;
        jumlahSepeda[2] = 1;
        jumlahSepeda[3] = 3;

        try {
            hasil_boolean =
MethodTryCatch(jumlahSepeda[1],jumlahSepeda[3]);
            System.out.println("Hasil Boolean : "+hasil_boolean);
        }
        catch (Exception e)
        {
            System.out.println("Jebak kesalahan dari method
utama");
            System.out.println("Awas! Ada kesalahan alamat
memori");
        }
    }

    //Method try.catch (bersarang)
    public static boolean methodTryCatch(int a, int b)
    {
        boolean boolean_nilai = false;
        int jumlahTerjual[] = new int[2];
        jumlahTerjual[0] = 3;
        jumlahTerjual[1] = 4;

        try
        {
            if (jumlahTerjual[0] > jumlahTerjual[2])
            {
                boolean_nilai = true;
            }
        }
        catch (Exception e2)
        {
```

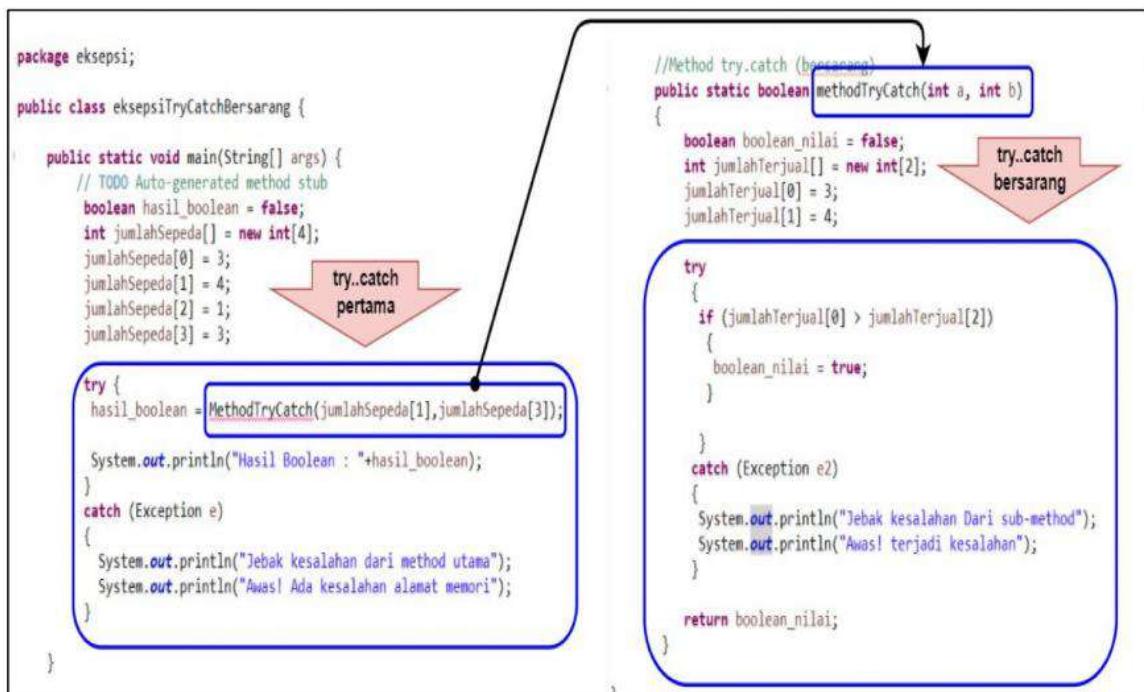
```

        System.out.println("Jebak kesalahan Dari sub-
method");
        System.out.println("Awas! terjadi kesalahan");
    }

    return boolean_nilai;
}
}

```

## 5. Ilustrasi jalannya program terhadap *try..catch* bersarang



Gambar 9.36  
Try except dalam Sebuah try except

## 6. Simpan program dan jalankan, hasilnya sebagai berikut:

```

Jebak kesalahan Dari sub-method
Awas! Terjadi kesalahan
Hasil Boolean : false

```

Penjelasan:

- Untuk hasil atau keluaran yang bertuliskan:

*Jebak kesalahan dari sub-method  
Awas! Terjadi kesalahan*

Di atas adalah keluaran hasil perangkap kesalahan yang dieksekusi karena terjadi kesalahan pada *method methodTryCatch*.

Keterangan:

*Sejatinya contoh program di atas adalah method dengan perangkap kesalahan memanggil method lain dengan juga menggunakan perangkap kesalahan, dimana method dipanggil dalam tubuh try..cath.*

### C. MENGGUNAKAN TRY..CATCH MULTI

Menggunakan banyak *try..catch* sangat dimungkinkan pada Java, selama Anda membuat atau memprediksi kesalahan-kesalahan yang akan terjadi. Untuk praktikum *try..catch* banyak sebagai berikut:

1. Buatlah file *class* dengan nama *eksepsiTryCatchBanyak.java* pada folder *eksepsi* pada projek *Praktikum03*, kemudian jangan lupa menyertakan *main method* pada file *class* tersebut.
2. Pada *main method* ketikkan program berikut:

```

String sepeda[] = new String[3];
double pembagian = 0;
sepeda[0] = "Sepeda Santai";
sepeda[1] = "Sepeda Gunung";
sepeda[2] = "Sepeda Balap";

try {
    pembagian = 10/0;
    System.out.println(sepeda[3]);

}
catch (ArrayIndexOutOfBoundsException e)
{
    System.out.println("Eksepsi:
ArrayIndexOutOfBoundsException");
    System.out.println("Ada indeks array melampaui
batas");
}

```

```

catch (ArithmetricException e)
{
    System.out.println("Eksepsi:
ArithmetricException");
    System.out.println("Pembagi tidak boleh = 0");
}

```

3. Kode program secara utuh sebagai berikut:
4. Simpan program dan jalankan, hasilnya sebagai berikut:

Eksepsi: ArrayIndexOutOfBoundsException  
Ada indeks array melampaui batas

Penjelasan:

- Dengan sistem banyak *try..catch* akan mencegat sesuai urutan *try..catch* yang diberikan. Misalnya yang lebih dulu dicek adalah **ArrayIndexOutOfBoundsException** maka yang akan di cek lebih dulu adalah proses yang berurusan dengan *array*, walau pada dasarnya operasi yang berjalan adalah operasi aritmetika.
- Jika terjadi kesalahan pada:

```

pembagian = 10/0;
System.out.println(sepeda[3]);

```

Maka yang akan diperangkap duluan tergantung pada urutan *catch* yang dipasang. Misalnya yang di atas yang dipasang duluan adalah **ArrayIndexOutOfBoundsException** maka yang ditangani adalah kesalahan *array* lalu kemudian *catch* berikutnya.

- Perangkap *error* yang dibuat sebagai latihan di atas adalah perangkap (eksepsi) khusus atau spesifik pada operasi tertentu. Hal ini membantu user atau *programmer* dalam mendeteksi kesalahan yang terjadi ketika program sudah dikompilasi dan digunakan. Contoh dengan: **ArithmetricException** adalah memang dibuat khusus jika terjadi kesalahan operasi pada operasi-operasi aritmetika.

5. Sekarang ubah kode program berikut:

Dari

```
System.out.println(sepeda[3]);
```

Menjadi:

```
System.out.println(sepeda[2]);
```

6. Simpan program dan jalankan, lihat hasilnya sebagai berikut:

```
Eksepsi: ArithmeticException  
Pembagi tidak boleh = 0
```

Penjelasan:

- Program di atas menjalankan eksepsi `ArithmeticException` spesifik untuk operasi aritmatika
  -
7. Kode program secara utuh sebagai berikut:

```
package eksepsi;

public class eksepsiTryCatchBanyak {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        String sepeda[] = new String[3];
        double pembagian = 0;
        sepeda[0] = "Sepeda Santai";
        sepeda[1] = "Sepeda Gunung";
        sepeda[2] = "Sepeda Balap";

        try {
            pembagian = 10/0;
```

```

        System.out.println(sepeda[2]);

    }
    catch (ArrayIndexOutOfBoundsException e)
    {
        System.out.println("Eksepsi:
ArrayIndexOutOfBoundsException");
        System.out.println("Ada indeks array melampaui
batas");
    }
    catch (ArithmetricException e)
    {
        System.out.println("Eksepsi:
ArithmetricException");
        System.out.println("Pembagi tidak boleh = 0");
    }
}

}

```

#### D. MENGGUNAKAN TRY..CATCH..FINALLY

Ikuti langkah-langkah berikut:

1. Buatlah file *class* dengan nama *eksepsiTryCatchFinally.java* pada folder *eksepsi* pada projek *Praktikum03*, kemudian jangan lupa menyertakan *main method* pada file *class* tersebut.
2. Pada *main method* ketikkan program berikut:

```

// TODO Auto-generated method stub
String sepeda[] = new String[4];

sepeda[0] = "Sepeda Santai";
sepeda[1] = "Sepeda Gunung";
sepeda[2] = "Sepeda Balap";
sepeda[3] = "Sepeda Ontel";

try {
    System.out.println(sepeda[4]);
}
catch (Exception e)
{
    System.out.println("Awas! Ada kesalahan alamat
memori");
}

```

```
    finally
    {
        System.out.println("Finally : tetap
dikerjakan");
    }
```

3. Kode program secara utuh sebagai berikut:

```
package eksepsi;

public class eksepsiTryCatchFinally {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        String sepeda[] = new String[4];

        sepeda[0] = "Sepeda Santai";
        sepeda[1] = "Sepeda Gunung";
        sepeda[2] = "Sepeda Balap";
        sepeda[3] = "Sepeda Ontel";

        try {
            System.out.println(sepeda[4]);
        }
        catch (Exception e)
        {
            System.out.println("Awas! Ada kesalahan akses
alamat memori");
        }
        finally
        {
            System.out.println("Finally : tetap
dikerjakan");
        }

    }
}
```

4. Simpan program dan jalankan, hasilnya sebagai berikut:

```
Awas! Ada kesalahan akses alamat memori
Finally : tetap dikerjakan
```

Penjelasan:

- Perhatikan pada hasil program, walaupun perangkap kesalahan telah ditampilkan, akan tetapi blok *finally* tetap dikerjakan oleh program.

```

try {
    System.out.println(sepeda[4]);
}
catch (Exception e)
{
    System.out.println("Awas! Ada kesalahan alamat memori");
}
finally
{
    System.out.println("Finally : tetap dikerjakan");
}

```

walaupun sudah  
diperangkap oleh  
**catch**, blok **finally**  
akan tetap dikerjakan



## LATIHAN

---

Untuk memperdalam pemahaman Anda mengenai materi di atas, kerjakanlah latihan berikut!

- 1) Apa hasil dari kode program berikut:

```

public class tryExcept {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int a, b;
        try {
            a = 0;
            b = 62 / a;
            System.out.println(b);
            System.out.println("ini adalah blok try..catch");
        }
        catch (ArithmeticException e)
        {
            System.out.println("Anda tidak boleh membagi dengan Nol");
        }
        catch (Exception e)
        {
            System.out.println("Pengecualian");
        }
    }
}

```

```

        }
    finally
    {
        System.out.println("Finally : tetap dikerjakan");
    }
    System.out.println("Akhir dari try..catch..finally");
}
}
}

```

*Petunjuk Jawaban Latihan*

- 1) Hasil dari program:

```

public class tryExcept {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int a, b;
        try {
            a = 0;
            b = 62 / a;
            System.out.println(b);
            System.out.println("ini adalah blok try..catch");
        }
        catch (ArithmeticException e)
        {
            System.out.println("Anda tidak boleh membagi dengan
Nol");
        }
        catch (Exception e)
        {
            System.out.println("Pengecualian");
        }
        Finally
        {
            System.out.println("Finally : tetap dikerjakan");
        }
        System.out.println("Akhir dari try..catch");
    }
}

```

Adalah sebagai berikut:

Anda tidak boleh membagi dengan Nol

Finally : tetap dikerjakan  
Akhir dari try..catch



Untuk menangani *error* di Java, digunakan sebuah *statement* yang bernama *try..catch*. *Statement* tersebut digunakan untuk mengurung eksekusi yang menampilkan *error* dan dapat membuat program tetap berjalan tanpa dihentikan secara langsung. *Error* yang ditangani oleh *try..catch* biasa disebut dengan *exception*.

**PRAKTIKUM 3.3**

## *Review Algoritma, Flowchart dan Kode Program*

Pada praktikum 3.2 kita akan mempraktekkan mulai dari membuat algoritma (*pesudocode*) sampai kepada kode programnya. Jadi kita praktekkan adalah contoh-contoh kasus dan akan kita selesaikan mulai dari *pseudocode*, *flowchart*, dan *kode program*.

### **A. MENGHITUNG TOTAL DATA YANG DIMASUKKAN DARI KEYBOARD**

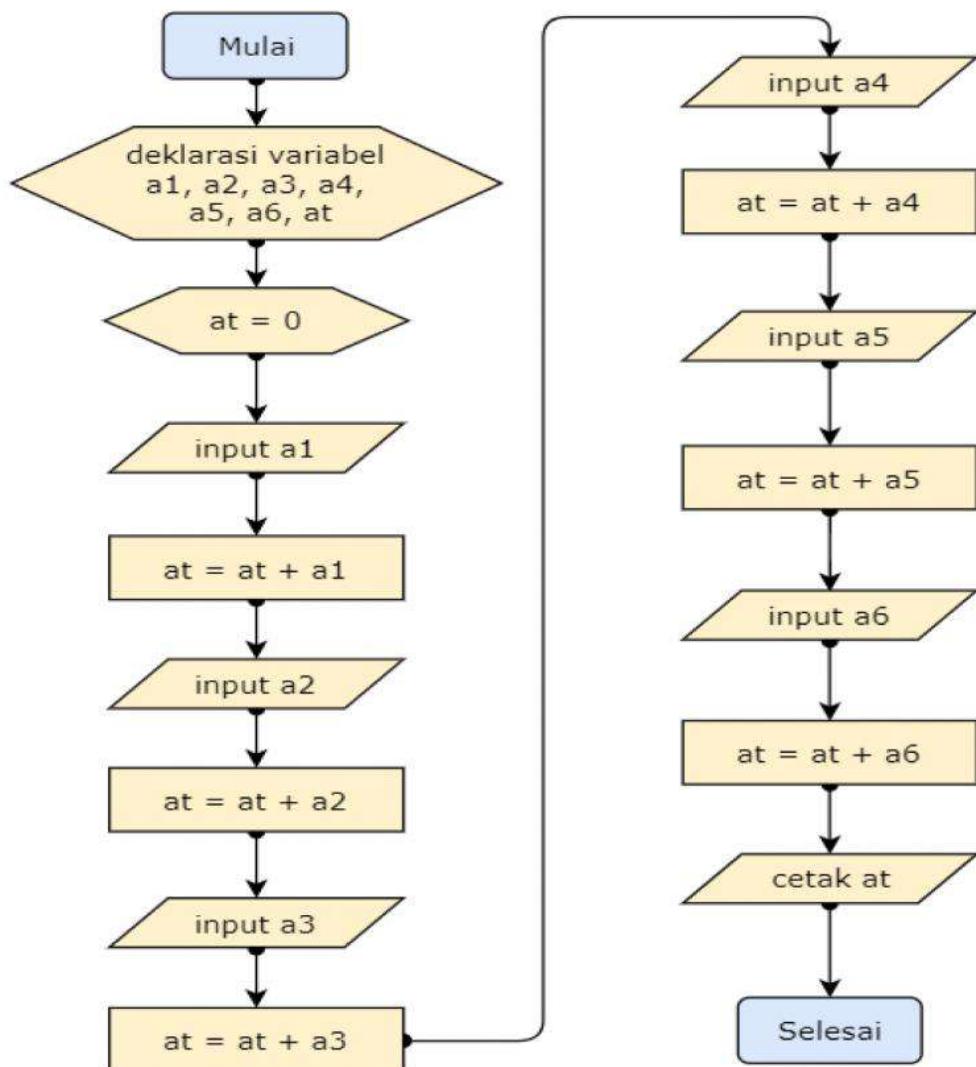
Misalkan nilai bilangan bulat yang dimasukkan adalah 1, 2, 3, 4, 5, dan 6. Nilai tersebut ditampung dalam sebuah variabel untuk menampung jumlah setiap angka yang dimasukkan.

#### **1. Penyelesaian yang tidak optimal**

##### *a. Pseudocode*

1. *Mulai*
2. *Deklarasikan Variabel a1, a2, a3, a4, a5, a6, at*
3. *Inisiasi at ← 0*
4. *Masukkan nilai a1*
5. *at ← at + a1*
6. *Masukkan nilai a2*
7. *at ← at + a2*
8. *Masukkan nilai a3*
9. *at ← at + a3*
10. *Masukkan nilai a4*
11. *at ← at + a4*
12. *Masukkan nilai a5*
13. *at ← at + a5*
14. *Masukkan nilai a6*
15. *at ← at + a6*
16. *Cetak at*
17. *Selesai*

## b. Flowchart



Gambar 9.37  
Flowchart Menghitung Satu Per Satu Sebuah Inputan

## c. Kode Program

```

import java.util.Scanner;

public class inputBilBulat {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int a1,a2,a3,a4,a5,a6,at;
        at = 0;
    }
}
  
```

```

Scanner input = new Scanner(System.in);
System.out.print("Masukkan Nilai a1 : ");
a1 = input.nextInt();
at = at + a1;
System.out.print("\nMasukkan Nilai a2 : ");
a2 = input.nextInt();
at = at + a2;
System.out.print("\nMasukkan Nilai a3 : ");
a3 = input.nextInt();
at = at + a3;
System.out.print("\nMasukkan Nilai a4 : ");
a4 = input.nextInt();
at = at + a4;
System.out.print("\nMasukkan Nilai a5 : ");
a5 = input.nextInt();
at = at + a5;
System.out.print("\nMasukkan Nilai a6 : ");
a6 = input.nextInt();
at = at + a6;

System.out.println("\nMasukkan Nilai at : "
+at);
}

}

```

Jika setiap masukan kita bernilai nilai masing-masing 1, 2, 3, 4, 5, 6 maka hasilnya adalah at = 21.

```

Masukkan Nilai a1 : 1
Masukkan Nilai a2 : 2
Masukkan Nilai a3 : 3
Masukkan Nilai a4 : 4
Masukkan Nilai a5 : 5
Masukkan Nilai a6 : 6

Masukkan Nilai at : 21

```

## 2. Penyelesaian yang optimal

Sebenarnya *flowchart*, *pseudocode*, dan Kode program sebelumnya tidaklah salah, akan tetapi sangat tidak optimal, walau hasil yang dikeluarkan adalah sama. Sekarang kita membuat yang optimal dengan keluaran sama persis yang hanya menggunakan 2 variabel saja dibanding dengan penyelesaian yang pertama menggunakan 7 variabel.

Sekarang kita membuat yang algoritma yang optimal sebagai berikut:

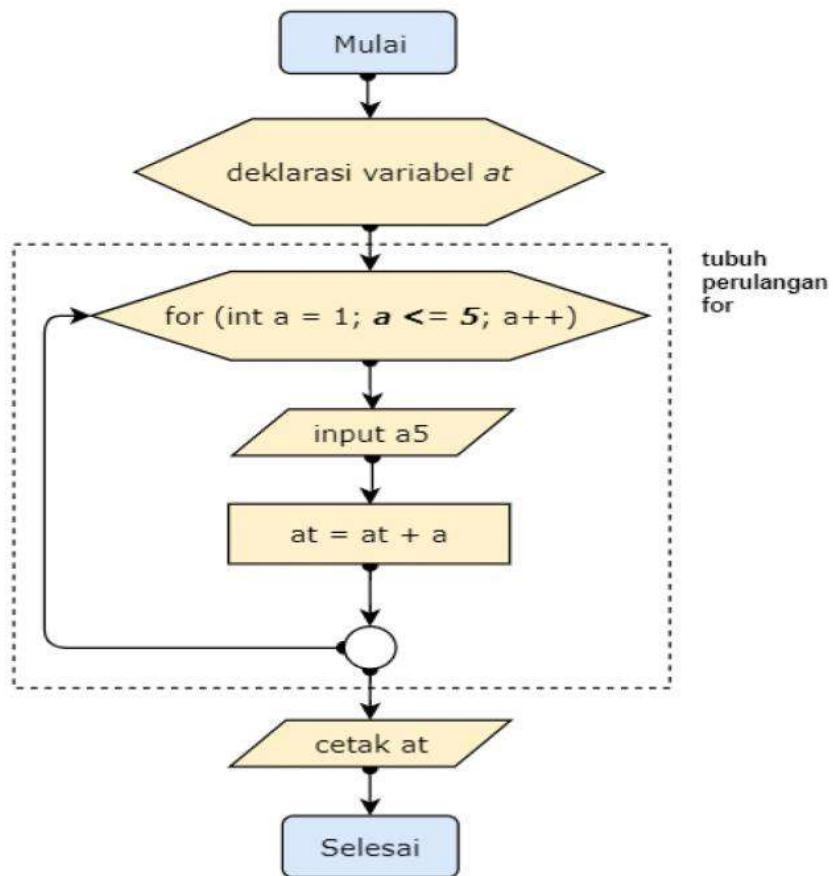
a. *Pseudocode*

1. *Mulai*
2. *Deklarasikan Variabel a, at*
3. *Inisiasi at  $\leftarrow 0$*
4. *for (a  $\leftarrow 1$ ; a  $\leq 5$ ; a++)*  
    4.1. *at  $\leftarrow at + a$*
5. *Cetak at*
6. *Selesai*

Jika dipersingkat lagi:

1. *Mulai*
2. *Int a, at*
3. *at  $\leftarrow 0$*
4. *for (a  $\leftarrow 1$ ; a  $\leq 5$ ; a++)*  
    4.1. *at  $\leftarrow at + a$*
5. *Cetak at*
6. *Selesai*

b. *Flowchart*



Gambar 9.38  
Flowchart Menghitung 5 Angka dengan Menggunakan Perulangan

c. *Kode Program*

```

public class inputBilBulatFor{
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int at = 0;

        Scanner input = new Scanner(System.in);

        for (int a = 1; a <= 6; a++)
        {
            System.out.print("Masukkan Nilai a"+a+
                ": ");
            a = input.nextInt();
            at = at + a;
        }
    }
}
    
```

```

        System.out.println("\nMasukkan Nilai at :
    "+at);
}
}

```

Hasilnya sama sebagai berikut:

```

Masukkan Nilai a1 : 1
Masukkan Nilai a2 : 2
Masukkan Nilai a3 : 3
Masukkan Nilai a4 : 4
Masukkan Nilai a5 : 5
Masukkan Nilai a6 : 6

Masukkan Nilai at : 21

```

Sekarang perhatikan perbedaan algoritma yang tidak optimal dengan yang optimal, perbandingan sebagai berikut :

| Algoritma           | Tak Optimal   | Optimal       |
|---------------------|---------------|---------------|
| <i>Pseudocode</i>   | 17 Baris      | 6 Baris       |
| <i>Flowchart</i>    | 17 Simbol     | 8 Simbol      |
| <i>Kode Program</i> | 27 Baris      | 13 Baris      |
| <b>Keluaran</b>     | Nilai at = 21 | Nilai at = 21 |

#### A. **BUBBLE SORT (PERULANGAN DAN SELEKSI)**

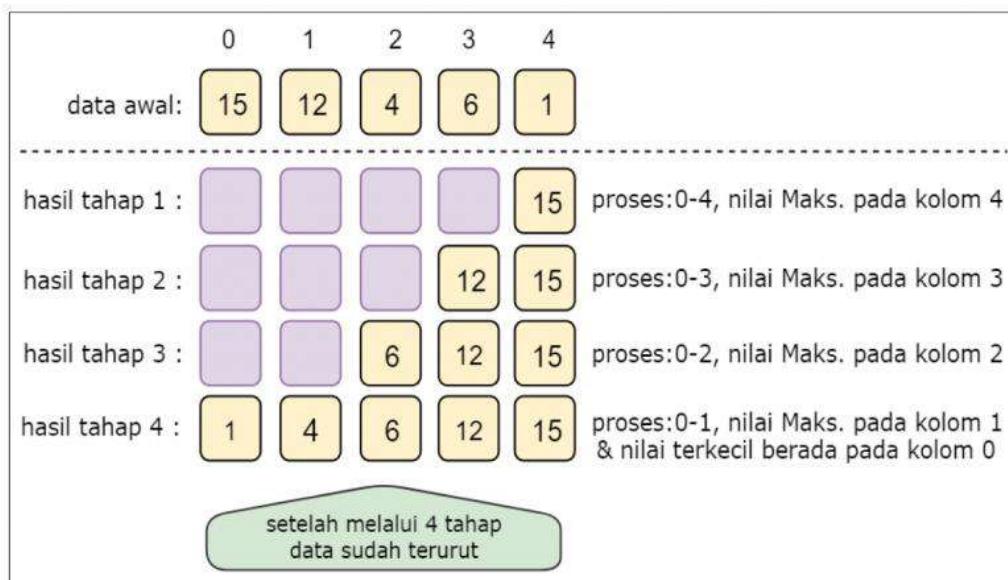
*Buble Sort* adalah algoritma pengurutan data yang menempatkan data terbesar pada bagian bawah setiap tahap. Untuk menghasilkan nilai urut maka yang harus dilakukan adalah:

1. melakukan perulangan dalam perulangan (anggaplah perulangan luar dengan *perulangan1* dan perulangan dalam disebut dengan *perulangan2*), apakah itu *for* dalam *for*, *for* dalam *while* atau *while* dalam *while* atau perulangan lainnya.
2. Untuk perulangan pada *perulangan1* hanya akan mengulang sebanyak  $n$  data.
3. Dalam tubuh perulangan *perulangan2* akan berulang sebanyak  $n-1$  data dan *perulangan2* akan diulang selama  $n$  perulangan, kemudian dibuat

penyeleksian antar data, misal dalam perulangan pertama data yang dibandingkan adalah *data1* dana *data2*, jika *data2* lebih besar dari *data1* maka nilai *data1* ditukar dengan *data2*, begitu seterusnya.

Untuk melihat ilustrasinya di bawah ini disajikan ilustrasi sajian dari algoritma *Buble Sort* langkah demi langkah.

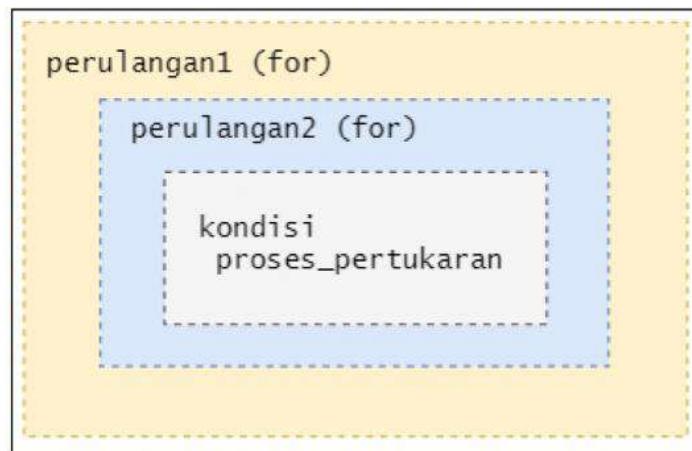
Misal kita punya data awal array sebagai berikut:



Gambar 9.39  
Ilustrasi Algoritma *Bubble Sort*

Skema yang dibuat di atas, dengan 4 tahap sudah mengurutkan data dengan sempurna, jika digambarkan maka terdapat 4 kali perulangan dan untuk setiap perulangan mengerjakan 1 tahap.

Kerangka dasar dari perulangan untuk *Bubble Sort* adalah sebagai berikut:



Gambar 9.40  
Kerangka Dasar Algoritma *Bubble Sort*

Pola *perulangan1 n* dan *perulangan n-1* ini akan berlaku umum untuk perulangan *for* pada sebuah pengurutan data, artinya jika data *array 5* maka akan berlaku *perulangan1 5* dan *perulangan2 5-1*.

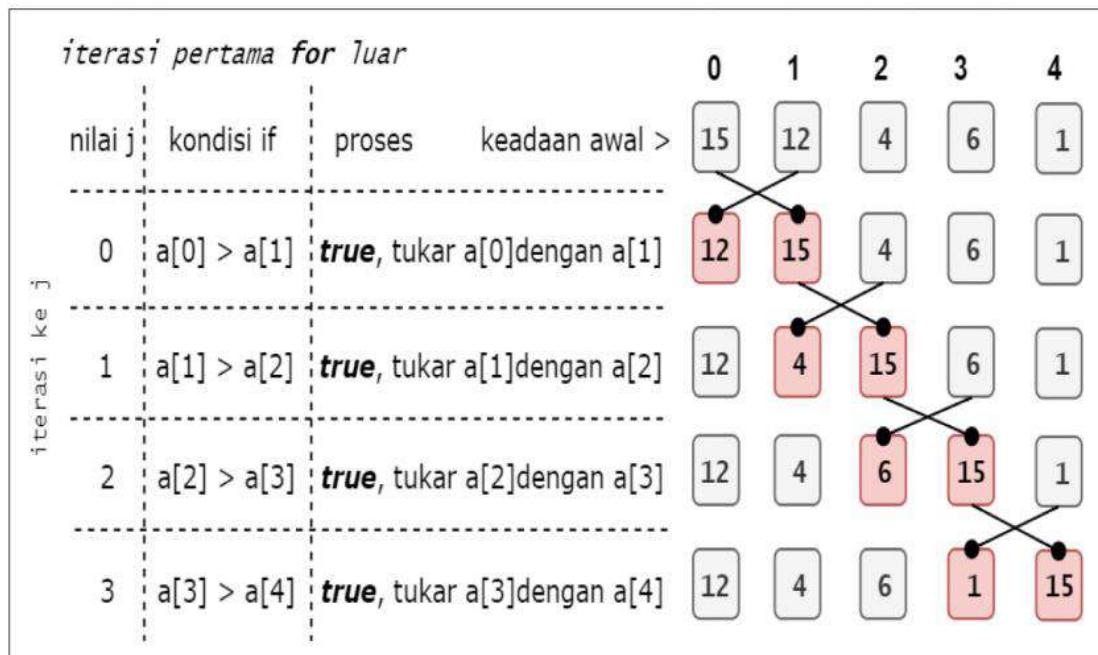
Untuk membuat 4 kali perulangan, gambaran dasar sebagai berikut:



Gambar 9.41  
Kode Program Algoritma *Bubble Sort*

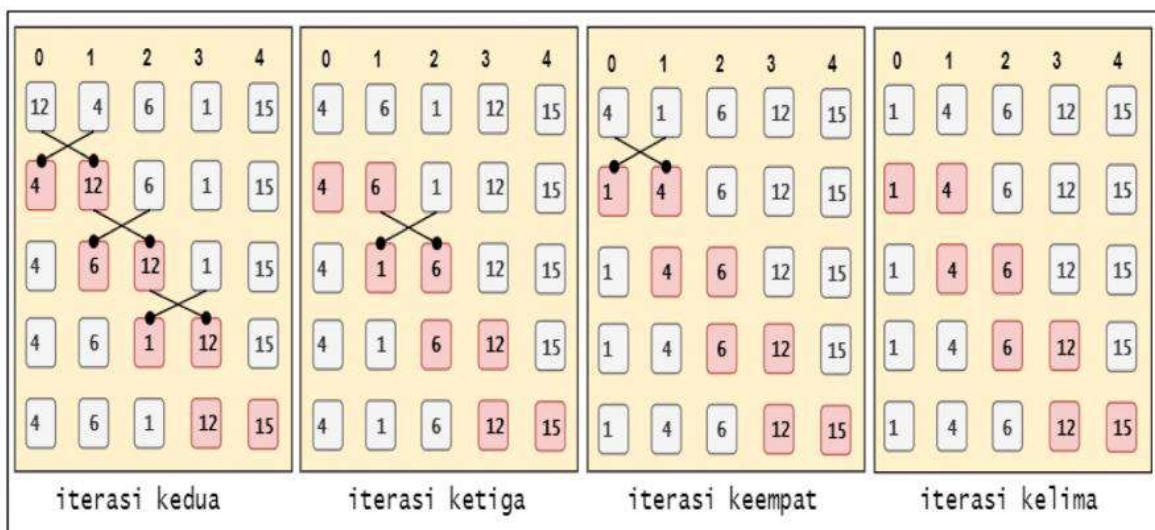
Mari menggambarkan ilustrasi pengurutan data sesuai dengan 2 gambar di atas untuk perulangan pertama untuk *for* luar, dengan ilustrasi untuk *for* dalam (bersarang) sebagai berikut:

- Iterasi Pertama



Gambar 9.42  
Proses Pertukaran Nilai Algoritma *Bubble Sort* pada Iterasi Pertama

- Iterasi berikutnya:

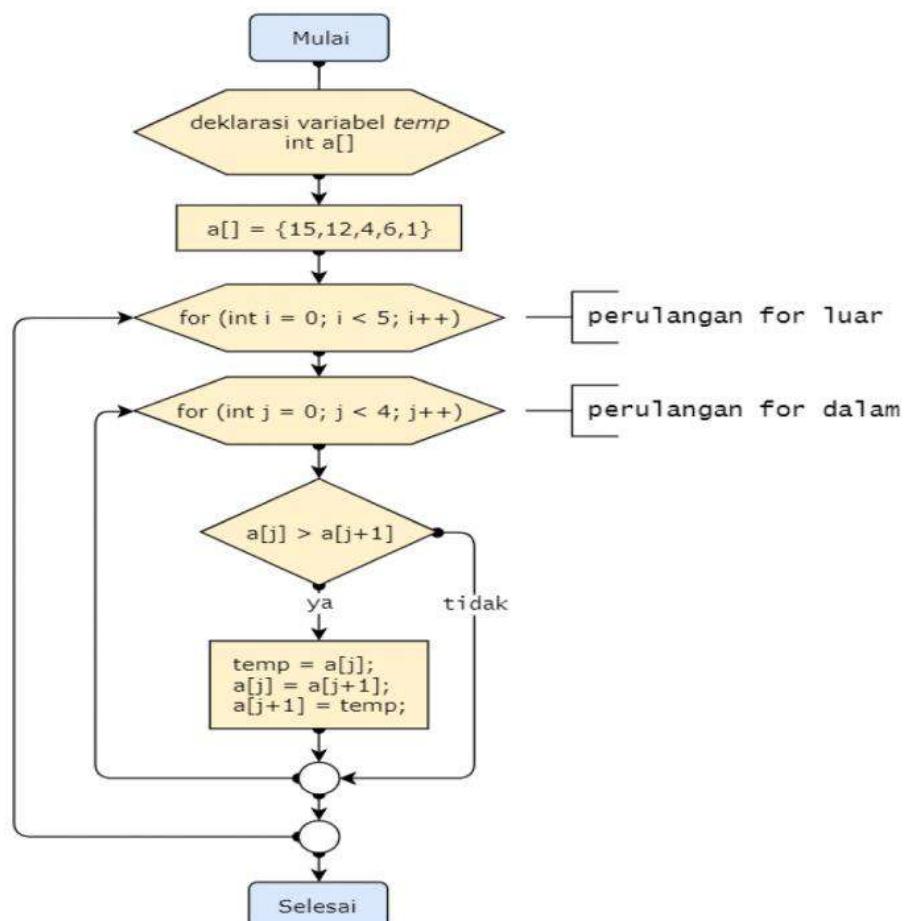


Gambar 9.43  
Proses Pertukaran Nilai Algoritma *Bubble Sort* pada Iterasi Kedua, Ketiga, Keempat, dan Kelima

Sekarang kita buat *pseucode* dari *Bubble Sort* sebagai berikut:

1. Mulai
2. int temp
3. int a[]  $\leftarrow \{15,12,4,6,1\}$
4. for (int i  $\leftarrow 0$ ; i < 5; i++)
  - 4.1. for (int j  $\leftarrow 0$ ; j < 4; j++)
    4111. if (a[j] > a[j+1])
      4112. temp  $\leftarrow a[j]$
      4113. a[j]  $\leftarrow a[j+1]$
      4114. a[j+1]  $\leftarrow temp$
- 4.2. End For (*opsional tanda berakhir for*)
5. End for (*opsional*)
6. Selesai

Untuk *flowchart* sebagai berikut:



Gambar 9.44  
*Flowchart Bubble Sort*

Untuk membuat kode programnya silahkan ketikkan kode program di bawah ini:

```
public class urutkanArray1DBubbleSort {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int a[] = {15,12,4,6,1};
        int x,k;

        System.out.println("Array sebelum diurut");
        for (k = 0; k < a.length; k++)
        {
            System.out.println("Array pada indeks ke-" + k +
                               " : "+ a[k]);
        }

        for (int i = 0; i < 5; i++)
        {
            for (int j = 0; j < 4; j++)
            {
                if (a[j] > a[j+1])
                {
                    x = a[j];
                    a[j] = a[j+1];
                    a[j+1] = x;
                }
            }
        }

        System.out.println("Array sesudah diurut");
        for (k = 0; k < a.length; k++)
        {
            System.out.println("Array pada indeks ke-" + k +
                               " : "+ a[k]);
        }
    }
}
```

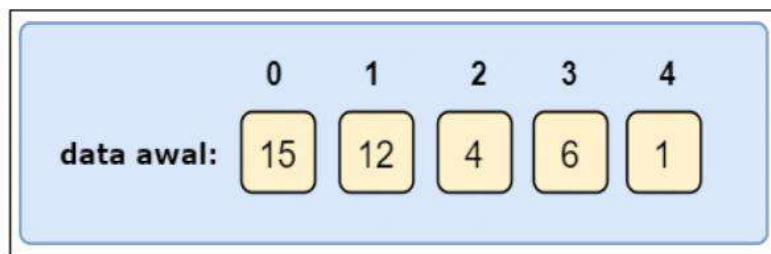
Kode program disertakan cetak sebelum dan sesudah diurutkan, sehingga hasilnya sebagai berikut:

```
Array sebelum diurut
Array pada indeks ke-0 : 15
Array pada indeks ke-1 : 12
Array pada indeks ke-2 : 4
Array pada indeks ke-3 : 6
Array pada indeks ke-4 : 1
Array sesudah diurut
Array pada indeks ke-0 : 1
Array pada indeks ke-1 : 4
Array pada indeks ke-2 : 6
Array pada indeks ke-3 : 12
Array pada indeks ke-4 : 15
```

### C. SELECTION SORT (SELEKSI)

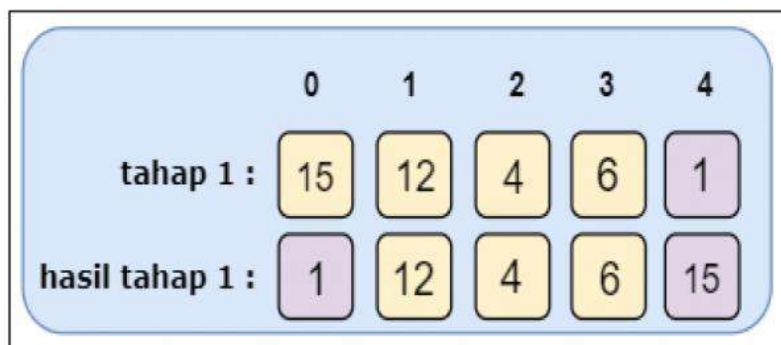
*Selection Sort* adalah algoritma pengurutan data dengan sangat sederhana, algoritma berjalan dimulai dengan menemukan nilai terkecil dalam kumpulan nilai data elemen, kemudian menukaranya dengan elemen pada posisi pertama, kemudian algoritma ini akan mengulangi dengan mencari terkecil kedua dan menukaranya dengan elemen kedua, hingga menemukan urutan yang benar.

Anggaplah kita punya data awal sebagai berikut:



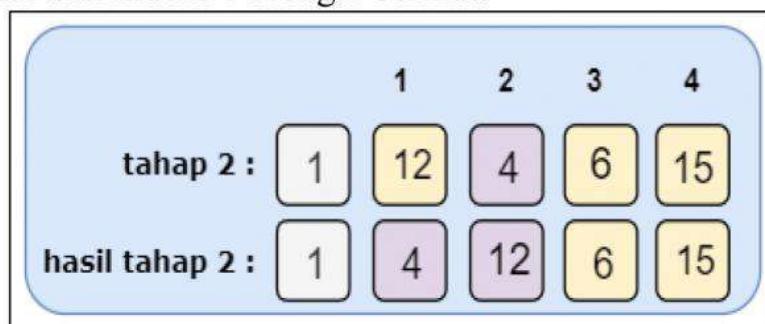
Gambar 9.45  
Data Awal Sebuah Array

Proses Tahap 1 adalah menemukan nilai terkecil pada data awal dengan dimulai dari indeks 0 sebagai berikut:



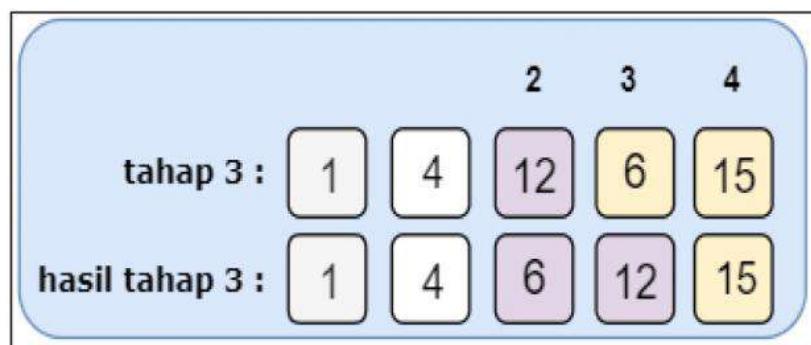
Gambar 9.46  
Tahap 1 dan Hasil Tahap 1

Proses Tahap 2 adalah menemukan nilai terkecil pada data hasil Tahap 1 dengan dimulai dari indeks 1 sebagai berikut:



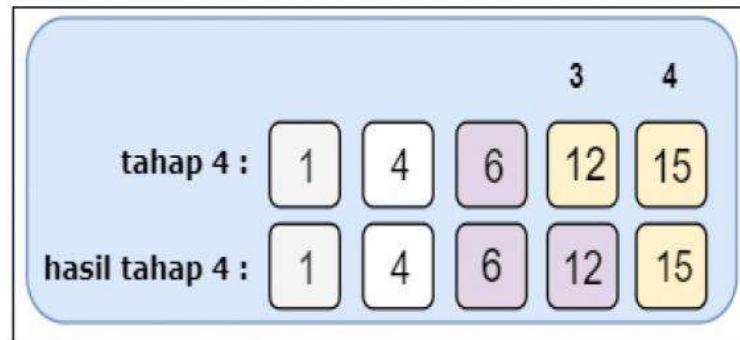
Gambar 9.47  
Tahap 2 dan Hasil Tahap 2

Proses Tahap 3 adalah menemukan nilai terkecil pada data hasil Tahap 2 dengan dimulai dari indeks 2 sebagai berikut:



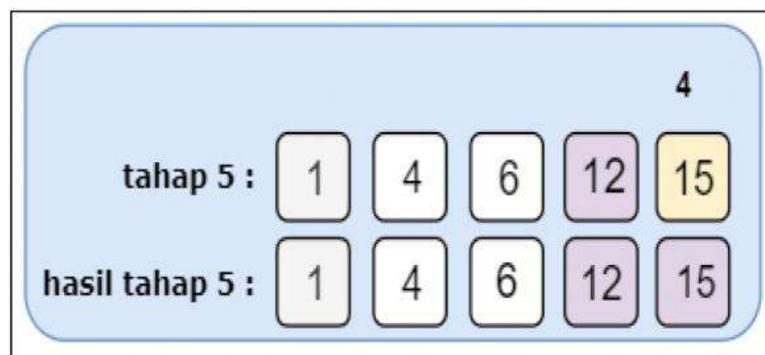
Gambar 9.48  
Tahap 3 dan Hasil Tahap 3

Proses Tahap 4 adalah menemukan nilai terkecil pada data hasil Tahap 3 dengan dimulai dari indeks 3 sebagai berikut:



**Gambar 9.49**  
**Tahap 4 dan Hasil Tahap 4**

Proses Tahap 5 adalah menemukan nilai terkecil pada data hasil Tahap 3 dengan dimulai dari indeks 4 sebagai berikut:

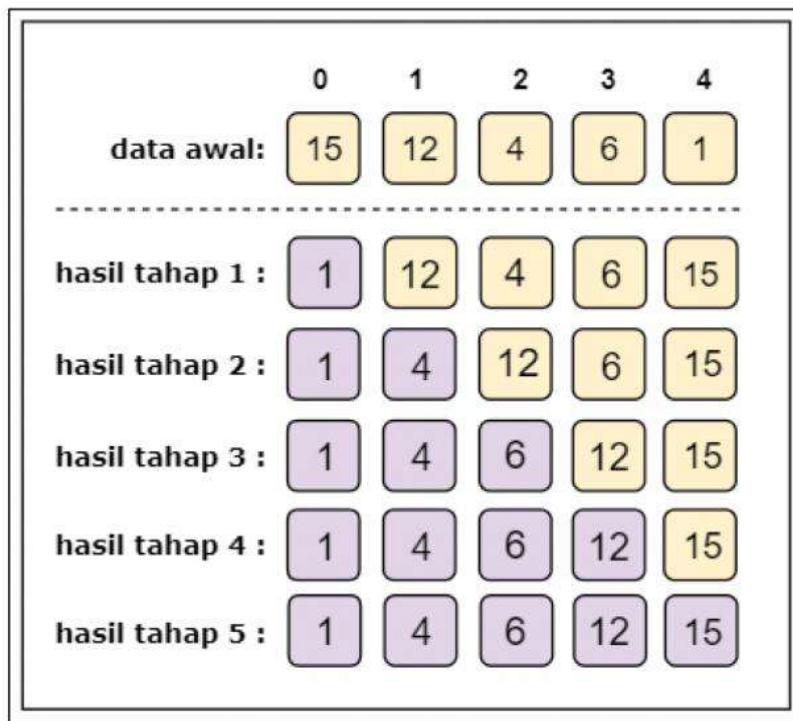


**Gambar 9.50**  
**Tahap 5 dan Hasil Tahap 5**

Catatan:

Jika pada kondisi tertentu menemukan elemen sudah lebih kecil dari pembanding, maka proses pertukaran sudah tidak dilakukan lagi.

Jika dirampingkan pertukarannya maka menghasilkan sebagai berikut:



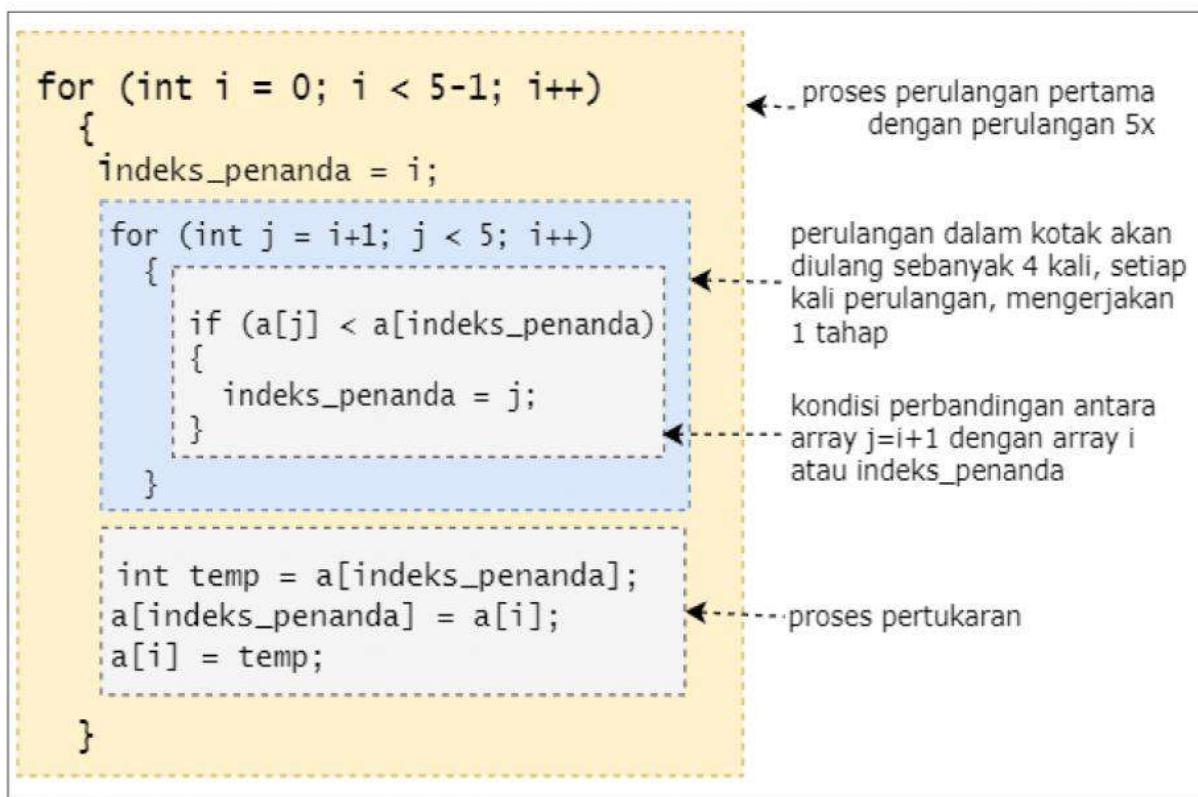
Gambar 9.51  
Data Awal dan Hasil Dari Setiap Tahap

Kerangka dasar dari perulangan untuk *Selection Sort* adalah sebagai berikut:



Gambar 9.52  
Kerangka Dasar dari Algoritma *Selection Sort*

Jika terdapat 5 elemen *array* maka akan selalu ada 4 perulangan . Struktur perulangannya sebagai berikut:



Gambar 9.53  
Kode Program dari Algoritma *Selection Sort*

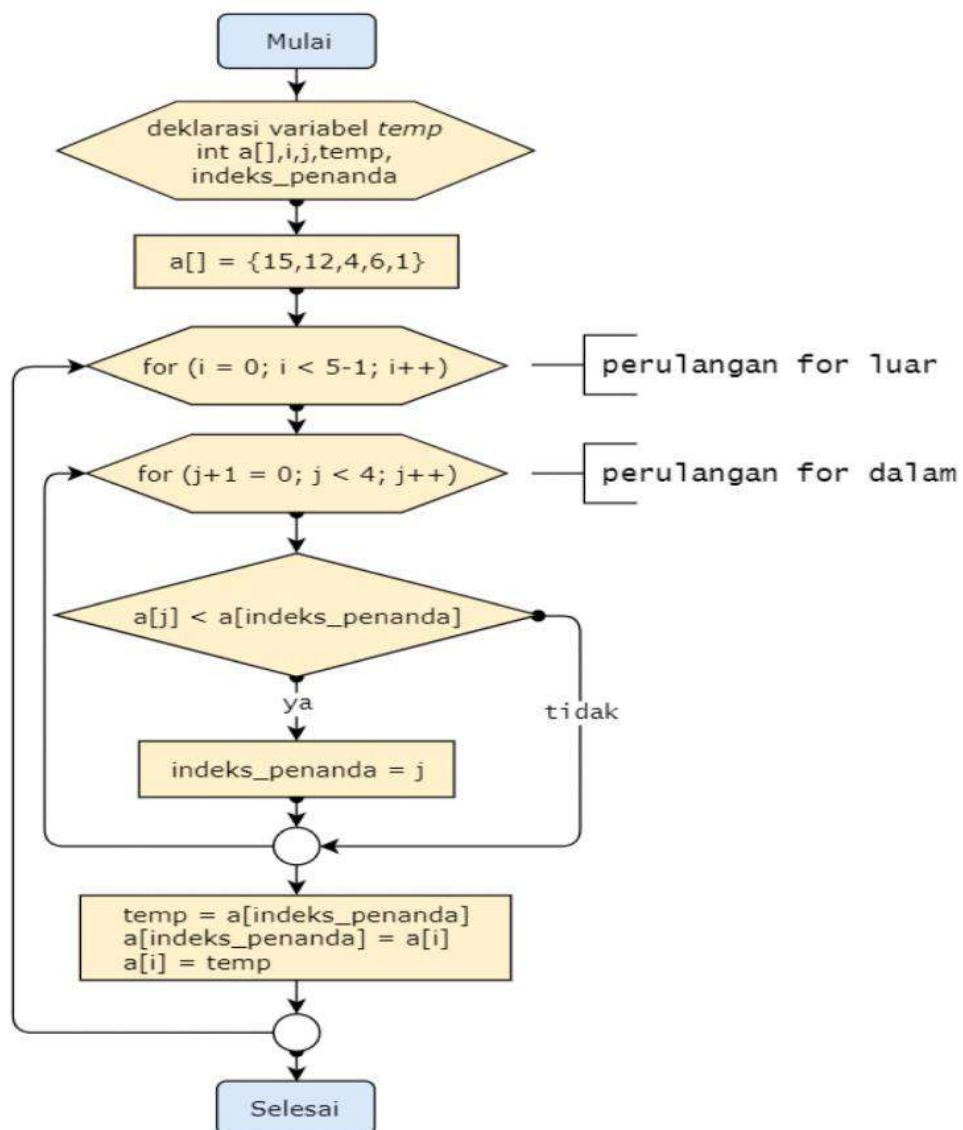
Dengan mengacu penjelasan di atas, sekarang kita buat algoritma *pseudocode* dari algoritma *Selection Sort* sebagai berikut:

1. Mulai
2. int *i,j,indeks\_penanda,temp*
3. int *a[]*  $\leftarrow \{15,12,4,6,1\}$
4. for (int *i* $\leftarrow 0$ ; *i* < 5-1; *i*++)
  - 4.1. *indeks\_penanda*  $\leftarrow i$
  - 4.2. for (int *j* $\leftarrow j+1$ ; *j* < 5; *j*++)
    421. if (*a[j]* > *a[indeks\_penanda]*)
      4211. *indeks\_penanda*  $\leftarrow j$
  - 4.3. End for (*opsional*)
  - 4.3. *temp*  $\leftarrow a[indeks\_penanda]$
  - 4.4. *a[indeks\_penanda]*  $\leftarrow a[i]$
  - 4.5. *a[i]*  $\leftarrow temp$
  - 4.6. End for (*opsional*)

5. End for (*opsional*)

6. Selesai

Kemudian untuk *flowchart*-nya sebagai berikut:



Gambar 9.54  
*Flowchart* dari Algoritma *Selection Sort*

Kode programnya sebagai berikut:

```

public class selectionSort {
    public static void main(String[] args) {
  
```

```

// TODO Auto-generated method stub
int a[] = {15,12,4,6,1};
int i, j,k, indeks_penanda;

//Cetak sebelum diurut
System.out.println("Array sebelum diurut");
for (k = 0; k < a.length; k++)
{
    System.out.println("Array pada indeks ke-" + k +
                       " : "+ a[k]);
}

//Proses pengurutan Selection Sort
for (i = 0; i < 5-1; i++)
{
    indeks_penanda = i;
    for (j = i+1; j < 5; j++)
        if (a[j] < a[indeks_penanda])
    {
        indeks_penanda = j;
    }

    int temp = a[indeks_penanda];
    a[indeks_penanda] = a[i];
    a[i] = temp;
}

//Cetak setelah diurut
System.out.println("Array sesudah diurut");
for (k = 0; k < a.length; k++)
{
    System.out.println("Array pada indeks ke-" + k +
                       " : "+ a[k]);
}
}
}

```

Hasilnya sebagai berikut:

```

sebelum diurut
Array pada indeks ke-0 : 15
Array pada indeks ke-1 : 12
Array pada indeks ke-2 : 4
Array pada indeks ke-3 : 6
Array pada indeks ke-4 : 1
Array sesudah diurut
Array pada indeks ke-0 : 1
Array pada indeks ke-1 : 4

```

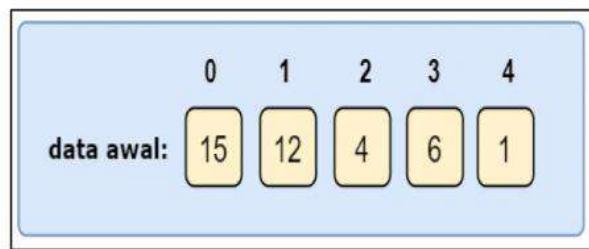
```
Array pada indeks ke-2 : 6
Array pada indeks ke-3 : 12
Array pada indeks ke-4 : 15
```

## D. INSERTION SORT

*Insertion Sort* adalah termasuk algoritma pengurutan sederhana, membandingkan dua elemen data pertama, mengurutkannya, kemudian dilakukan pengecekan elemen data berikutnya satu per satu dan membandingkan data yang telah diurutkan.

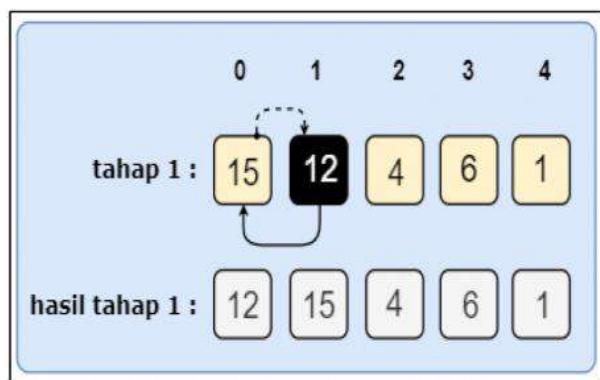
Kita buat ilustrasi sebagai berikut:

- ✓ Data awal:



Gambar 9.55  
Sebuah Data Awal

- ✓ Perulangan Tahap 1:



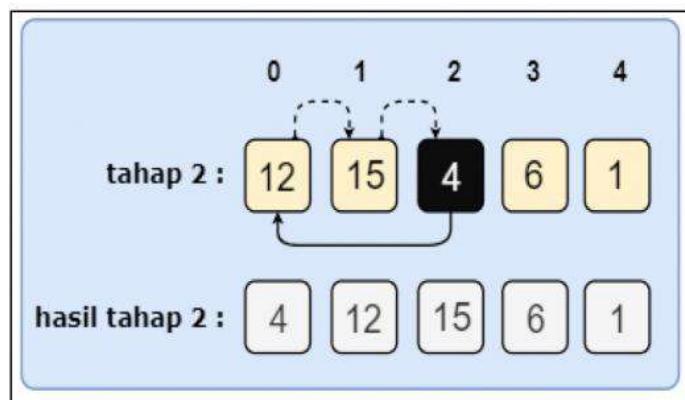
Gambar 9.56  
Tahap 1 dan Hasil Tahap 1

Keterangan:

1. Nilai pada indeks 1 dibandingkan dengan indeks 0

2. Nilai pada indeks 1 lebih kecil dari indeks 0, karena itu nilai indeks 0 ditukar posisi dengan nilai indeks 1.

✓ Perulangan Tahap 2:

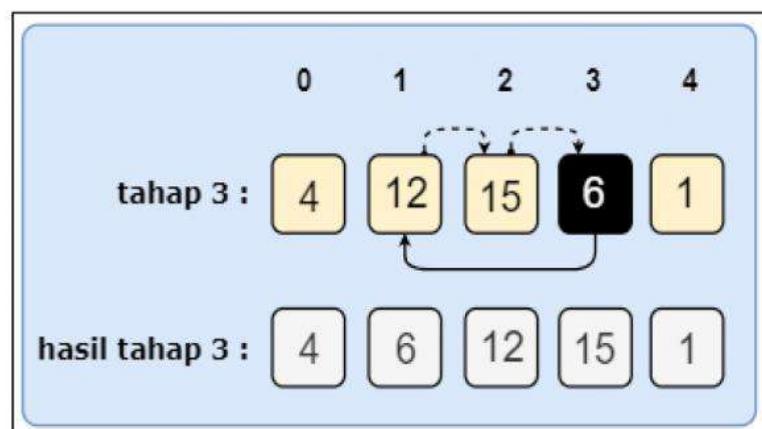


Gambar 9.57  
Tahap 2 dan Hasil Tahap 2

Keterangan:

1. Nilai pada indeks 2 dibandingkan dengan indeks 0 dan 1
2. Nilai pada indeks 2 lebih kecil dari indeks 0 dan 1, karena itu:
  - Nilai indeks 2 → indeks 0
  - Nilai indeks 0 → indeks 1
  - Nilai indeks 1 → indeks 2

✓ Perulangan Tahap 3:

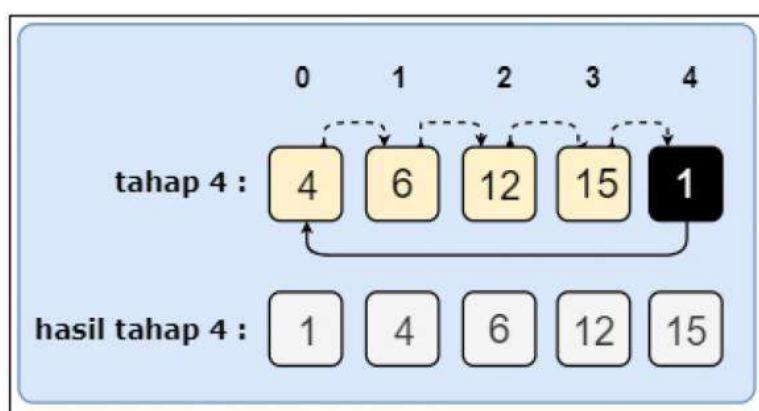


Gambar 9.58  
Tahap 3 dan Hasil Tahap 3

Keterangan:

1. Nilai indeks 3, dibandingkan dengan nilai indeks 0, 1, 2, 3.
2. Nilai pada indeks 3 lebih kecil dari indeks 1 dan 2 akan tetapi lebih besar dari indeks 0, karena itu:
  - Nilai indeks 3 → indeks 1
  - Nilai indeks 1 → indeks 2
  - Nilai indeks 2 → indeks 3
  - Nilai indeks 0 tetap

✓ Perulangan Tahap 4:

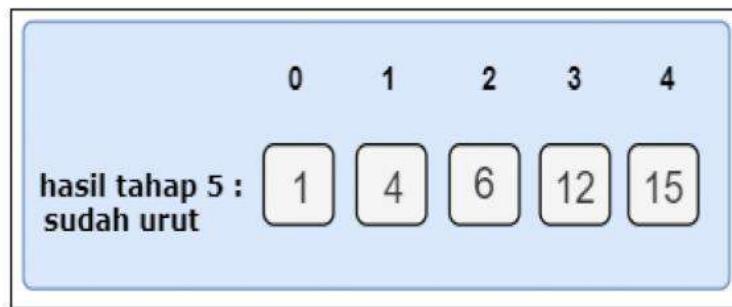


Gambar 9.59  
Tahap 4 dan Hasil Tahap 4

Keterangan:

1. Nilai indeks 4, dibandingkan dengan nilai indeks 0, 1, 2, 3.
2. Nilai pada indeks 4 lebih kecil dari indeks 1, 2, 3 dan 4, karena itu:
  - Nilai indeks 4 → indeks 0
  - Nilai indeks 0 → indeks 1
  - Nilai indeks 1 → indeks 2
  - Nilai indeks 2 → indeks 3
  - Nilai indeks 3 → indeks 4

✓ Perulangan Tahap 5:



Gambar 9.60  
Hasil pengurutan Setelah Tahap 5

Kerangka dasar dari perulangan *for* untuk *Insertion Sort* hampir sama dengan *Bubble Sort*, yang membedakan adalah beberapa kondisi yang tentu sangat berbeda, kerangka sebagai berikut:



Gambar 9.61  
Kerangka Dasar dari Algoritma *Insertion Sort*

Jika terdapat 5 elemen *array* maka akan selalu ada 4 perulangan . jika dibuat struktur perulangannya sebagai berikut:

```

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

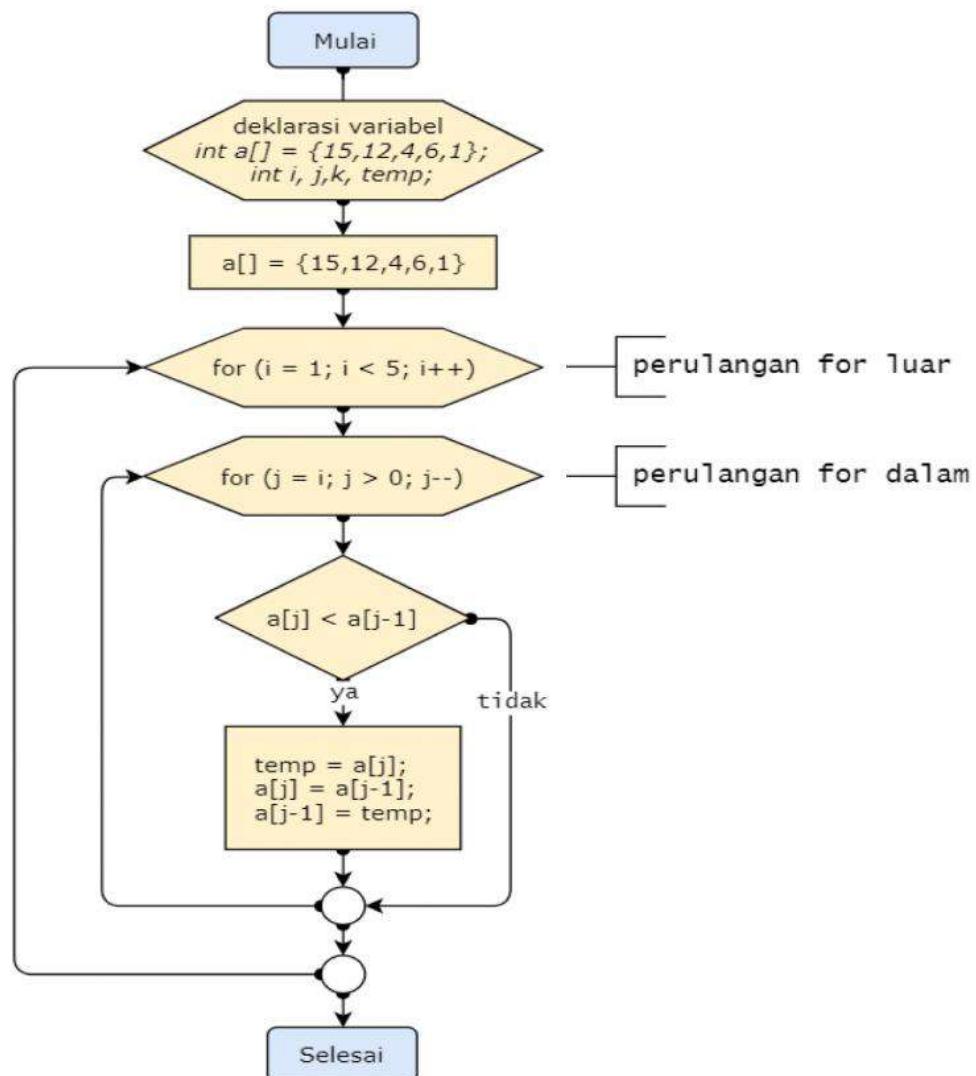
```

Gambar 9.62  
Kode Program dari Algoritma *Insertion Sort*

Sekarang kita coba membuat *pseudocode* dari algoritma *Insertion Sort* sebagai berikut:

1. Mulai
2. int temp
3. int a[]  $\leftarrow \{15,12,4,6,1\}$
4. for (int i  $\leftarrow 0$ ; i < 5; i++)
  - 4.1. for (int j  $\leftarrow i$ ; j > 0; j--)
    411. if(a[j] < a[j-a])
    4111. temp  $\leftarrow a[j]$
    4112. a[j]  $\leftarrow a[j-1]$
    4113. a[j-1]  $\leftarrow temp$
  - 4.2. End For (*opsional tanda berakhir for*)
5. End for (*opsional*)
6. Selesai

Sekarang kita coba membuat *flowchart* dari algoritma *Insertion Sort* sebagai berikut:



Gambar 9.63  
Flowchart Algoritma *Insertion Sort*

Untuk kode program sebagai berikut:

```

public class insertionSort {

    public static void main(String[] args) {
        int a[] = {15,12,4,6,1};
        int i, j,k, temp;

        for (i = 1; i < a.length; i++)
        {
            for(j = i ; j > 0 ; j--)
            {
                if(a[j] < a[j-1])
                {
    
```

```

        temp = a[j];
        a[j] = a[j-1];
        a[j-1] = temp;
    }
}

for (k = 0; k < a.length; k++)
{
    System.out.println("Array pada indeks ke-" + k +
                       " : " + a[k]);
}

}

```

Hasilnya sebagai berikut:

```

Array sesudah diurut
Array pada indeks ke-0 : 1
Array pada indeks ke-1 : 4
Array pada indeks ke-2 : 6
Array pada indeks ke-3 : 12
Array pada indeks ke-4 : 15

```

Di atas hanya dijelaskan 3 algoritma pengurutan data saja, sebenarnya masih ada algoritma pengurutan data yang lainnya seperti: *Shell Sort, Merge Sort, Radix Sort, Quick Sort, Heap Sort*.



## LATIHAN

---

Untuk memperdalam pemahaman Anda mengenai materi di atas, kerjakanlah latihan berikut!

- 1) Buatlah program dengan menggunakan kombinasi *for* dan *while* untuk algoritma *Bubble Sort*.
- 2) Diketahui *array* sebagai berikut:

```
arr[] = {90, 145, 30, 63, 120, 111, 135}
```

Buatlah program untuk mengurutkan *array* tersebut dari besar ke kecil dengan algoritma *Bubble Sort*.

- 3) Diketahui *array* sebagai berikut:

```
arr[] = {90, 145, 30, 63, 120, 111, 135}
```

Buatlah program dengan menggunakan *do..while* untuk algoritma *Selection Sort*.

### *Petunjuk Jawaban Latihan*

- 1) Untuk kombinasi *for* dan *while* sebagai berikut kode programnya:

```
public class bubbleSortForWhile {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int a[] = {15,12,4,6,1};
        int i,j,x,k;

        System.out.println("Array sebelum diurut");
        for (k = 0; k < a.length; k++)
        {
            System.out.println("Array pada indeks ke-" + k +
                               " : "+ a[k]);
        }
        //Algoritma Bubble Sort
        for (i = 0; i < 5; i++)
        {
            j = 0;
            while (j < 4)
            {
                if (a[j] > a[j+1])
                {
                    x = a[j];
                    a[j] = a[j+1];
                    a[j+1] = x;
                }
                j++;
            }
        }

        System.out.println("Array sesudah diurut");
        for (k = 0; k < a.length; k++)
        {
```

- 2) Untuk mengurutkan *array*:

`arr[] = {90, 145, 30, 63, 120, 111, 135}`

Berikut kode programnya:

```
public class bubbleSortDesc {  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        int arr[] = {90, 145, 30, 63, 120, 111, 135};  
        int x,k;  
  
        System.out.println("Array sebelum diurut DESC");  
        System.out.println("-----");  
        for (k = 0; k < arr.length; k++)  
        {  
            System.out.println("Array pada indeks ke-" + k +  
                               " : " + arr[k]);  
        }  
  
        //Algoritma Bubble Sort DESC  
        for (int i = 0; i < arr.length; i++)  
        {  
            for (int j = 0; j < arr.length-1; j++)  
            {  
                //untuk DESC cukup mengubah operator  
                //dari > ke operator <  
                if (arr[j] < arr[j+1])  
                {  
                    x = arr[j];  
                    arr[j] = arr[j+1];  
                    arr[j+1] = x;  
                }  
            }  
        }  
  
        System.out.println("\nArray sesudah diurut DESC");  
        System.out.println("-----");  
    }  
}
```

```

for (k = 0; k < arr.length; k++)
{
    System.out.println("Array pada indeks ke-" + k +
                       " : " + arr[k]);
}
}

```

Hasilnya sebagai berikut:

```

Array sebelum diurut DESC
-----
Array pada indeks ke-0 : 90
Array pada indeks ke-1 : 145
Array pada indeks ke-2 : 30
Array pada indeks ke-3 : 63
Array pada indeks ke-4 : 120
Array pada indeks ke-5 : 111
Array pada indeks ke-6 : 135

Array sesudah diurut DESC
-----
Array pada indeks ke-0 : 145
Array pada indeks ke-1 : 135
Array pada indeks ke-2 : 120
Array pada indeks ke-3 : 111
Array pada indeks ke-4 : 90
Array pada indeks ke-5 : 63
Array pada indeks ke-6 : 30

```

3) Untuk mengurutkan *array*:

```
arr[] = {90, 145, 30, 63, 120, 111, 135}
```

Berikut kode program algoritma *Selection Sort* dengan menggunakan statemen *while*:

```

public class selectionSortDoWhile {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int arr[] = {90, 145, 30, 63, 120, 111, 135};
        int i, j,k, indeks_penanda;

        //Cetak sebelum diurut
    }
}

```

```

System.out.println("Array sebelum diurut");
for (k = 0; k < arr.length; k++)
{
    System.out.println("Array pada indeks ke-" + k +
                       " : " + arr[k]);
}

i = 0;
//Proses pengurutan Selection Sort
do
{
    indeks_penanda = i;
    j = i + 1;

    do
    {
        if (arr[j] < arr[indeks_penanda])
        {
            indeks_penanda = j;
        }

        j++;
    } while (j < arr.length);

    int temp = arr[indeks_penanda];
    arr[indeks_penanda] = arr[i];
    arr[i] = temp;

    i++;
} while (i < arr.length-1);

//Cetak setelah diurut
System.out.println("Array sesudah diurut");
for (k = 0; k < arr.length; k++)
{
    System.out.println("Array pada indeks ke-" + k +
                       " : " + arr[k]);
}
}
}

```

Hasilnya:

```

Array sebelum diurut
Array pada indeks ke-0 : 90
Array pada indeks ke-1 : 145
Array pada indeks ke-2 : 30
Array pada indeks ke-3 : 63
Array pada indeks ke-4 : 120
Array pada indeks ke-5 : 111

```

```
Array pada indeks ke-6 : 135
Array sesudah diurut
Array pada indeks ke-0 : 30
Array pada indeks ke-1 : 63
Array pada indeks ke-2 : 90
Array pada indeks ke-3 : 111
Array pada indeks ke-4 : 120
Array pada indeks ke-5 : 135
Array pada indeks ke-6 : 145
```



## RANGKUMAN

---

Algoritma *Bubble Sort* adalah algoritma pengurutan data yang menempatkan data terbesar pada bagian bawah setiap tahap.

*Selection Sort* adalah algoritma pengurutan data dengan sangat sederhana, algoritma berjalan dimulai dengan menemukan nilai terkecil dalam kumpulan nilai atau elemen, kemudian menukarannya dengan elemen pada posisi pertama, kemudian algoritma ini akan mengulangi dengan mencari terkecil kedua dan menukarannya dengan elemen kedua, hingga menemukan urutan yang benar.

*Insertion Sort* adalah termasuk algoritma pengurutan sederhana, membandingkan dua elemen data pertama, mengurutkannya, kemudian dilakukan pengecekan elemen data berikutnya satu per satu dan membandingkan data yang telah diurutkan.

## Daftar Pustaka

- Eckel, B. (2000). *Thinking in Java (2<sup>nd</sup>)*. New Jersey: Prentice Hall.
- Schildt, H. (2007). *The complete reference java (Seventh Edition)*. New York: Mc Graw Hill.
- Flask, R. *Java for beginners (2<sup>nd</sup>)*.
- Eck, D.J. (2006). *Introduction to programming using java (Version 5.0)*. Departement of Mathematic and Computer Science. Geneva. New York 14456.
- Horstmann, C.S. & Cornell, G. (2013). “Core java volume 1 fundamental(Ninth Edition)”. New York: Prentice Hall.
- Redko, A. *Adnvanve java preparing you for Java Mastery*.

<https://docs.oracle.com/javase/7/docs/api/java/lang/StringBuffer.html>

## Daftar Riwayat Hidup



Nama : Kani, M.Kom.  
Tempat/Tgl. Lahir : Sempang, 14 Juni 1978  
Instansi Afiliasi : Universitas Terbuka  
Fakultas Sains dan  
Teknologi  
Jurusan Sistem Informasi  
Alamat : Pondok Cabe, Pamulang,  
Tangerang Selatan 15418  
Banten – Indonesia

**Riwayat Pendidikan:** Tamat S1 Teknik Informatika FIK-UMI Makassar (2003), Tamat S2 Ilmu Komputer IPB (2017).

**Karya Tulis:** Pemrograman Database Menggunakan Delphi