

# 1.0

# PENGATURCARAAN





## 1.0 PENGATURCARAAN

1.1 Strategi  
Penyelesaian  
Masalah

1.2 Algoritma

1.3 Pemboleh  
Ubah, Pemalar  
dan Jenis Data

1.4 Struktur  
Kawalan

1.5 Amalan  
Terbaik  
Pengaturcaraan

1.6 Struktur Data  
dan Modular

1.7  
Pembangunan  
Aplikasi

## 1.1

# STRATEGI PENYELESAIAN MASALAH

1.Keraguan, situasi yang tidak diingini, cabaran dan peluang yang dihadapi dalam kehidupan.

2.Kemahiran membuat keputusan amat diperlukan

## MASALAH

6. PROSES MENGKAJI BUTIRAN SESUATU MASALAH UNTUK MEDAPATKAN PENYELESAIAN

3. 2 format algoritma : Pesudokod cartalir

## PENYELESAIAN MASALAH

4.Pengaturcara perlu menulis sintaks yang spesifik.

5. Sintaks ialah peraturan yang diperlukan oleh komputer untuk melaksanakan arahan

2.Pengaturcara perlu memahami cara penyelesaian masalah dan menterjemahkan menjadi algoritma.

1.Tunjang utama sains komputer

**1.1**

**1.1.1**

## **MENERANGKAN KEPERLUAN PENYELESAIAN MASALAH BERSTRATEGI**



## CIRI PENYELESAIAN MASALAH BERKESAN

KOS

MASA

SUMBER

- KOS**
- Harga yang perlu dibayar untuk memperoleh, mengeluarkan dan menyelenggara.
  - Biasanya wang, masa, tenaga dan perbelanjaan.
  - PROJEK NORMAL – Projek yang disiapkan mengikut masa dan kos yang diperlukan.
  - CRASHING COST – Usaha yang maksimum untuk menyelesaikan projek dalam masa yang terpendek.
  - Kos meningkat apabila tempoh masa menurun.

MASA

- Merujuk kepada projek disiapkan mengikut tempoh masa yang ditetapkan.
- Aktiviti perlaksanaan yang tertunda/lambat akan meningkatkan kos.
- Keperluan menyiapkan projek dalam masa tercepat/terhad juga meningkatkan kos.

SUMBER

- Stok/wang, bahan-bahan mentah, staf dan asset lain yang boleh digunakan oleh organisasi supaya dapat berfungsi dengan efektif.
- Diperlukan untuk menjana hasil atau perkhidmatan.
- CONTOH : Sumber manusia, sumber kewangan.
- Perancangan Sumber – Tanggungjawab pihak pengurusan untuk mendapatkan keputusan yang optimum.
- Perancangan rapi dapat elak pembaziran sumber.
- Kekurangan sumber akan melambatkan masa menyiapkan projek dan meningkatkan kos.
- Penjadualan sumber – elak kelewatian projek.

**1.1****1.1.3**

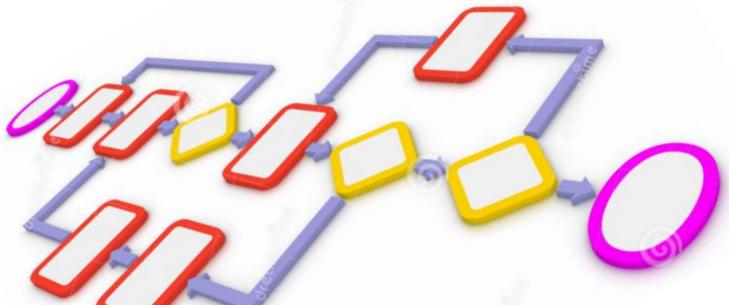
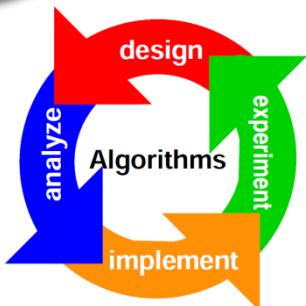
## MENGGUNAKAN PROSES PENYELESAIAN MASALAH



PROSES	AKTIVITI
<b>MENGUMPUL DAN MENGANALISIS DATA</b>	<ul style="list-style-type: none"><li>Mengumpul data tentang punca dan skop masalah.</li><li>Mengenalpasti hal yang berkaitan dengan situasi.</li></ul>
<b>MENENTUKAN MASALAH</b>	<ul style="list-style-type: none"><li>Mengenalpasti masalah utama yang perlu diselesaikan.</li><li>Mengenalpasti masalah seterusnya.</li></ul>
<b>MENJANA IDEA</b>	<ul style="list-style-type: none"><li>Menyenaraikan beberapa idea yang dapat digunakan untuk menyelesaikan masalah.</li></ul>
<b>MENJANA PENYELESAIAN</b>	<ul style="list-style-type: none"><li>Menyenaraikan idea atau langkah semasa merancang penyelesaian.</li></ul>
<b>MENJANA TINDAKAN</b>	<ul style="list-style-type: none"><li>Membuat pilihan terbaik daripada senarai idea yang dibuat.</li></ul>
<b>MELAKSANAKAN PENYELESAIAN</b>	<ul style="list-style-type: none"><li>Menggunakan pelbagai alat dan teknik yangtelah dipilih untuk melaksanakan penyelesaian.</li></ul>
<b>MEMBUAT PENILAIAN</b>	<ul style="list-style-type: none"><li>Penilaian dilaksanakan terhadap langkah penyelesaian.</li></ul>
<b>MEMBUAT PENAMBAHBAIKAN</b>	<ul style="list-style-type: none"><li>Setiap penyelesaian perlu ditambah baik jika ada kekurangan dan mengikut keperluan.</li></ul>

## 1.2

# ALGORITMA



1.Satu set arahan untuk menyelesaikan masalah.

2.Arahan-arahan terperinci yang dapat diikuti oleh pembaca

## ALGORITMA

3.Algoritma dapat dihalusi dengan menambahkan butiran.

### CIRI-CIRI ALGORITMA

Butiran Jelas

Boleh dilaksanakan

Mempunyai batasan.

## 1.2

### 1.2.1

### MENGGUNAKAN ALGORITMA UNTUK MENYATAKAN PENYELESAIAN KEPADA MASALAH

## KONSEP INPUT-PROSES-OUTPUT (IPO) UNTUK PERISIAN KOMPUTER

INPUT



PROSES



OUTPUT

### ANALISIS IPO

- Mengenalpasti data input, proses, untuk mengubah nilai data kepada maklumat dan paparan output maklumat setelah proses.
- Carta Input-Proses-Output(IPO) boleh digunakan untuk menganalisis masalah.

## CARTA IPO

INPUT	* Harus mengenalpasti data yang perlu dibaca daripada pengguna atau persekitaran.
PROSES	* Langkah-langkah ataupun rumusan untuk memproses data input kepada output.
OUTPUT	*Harus mengenalpasti output yang dikehendaki, yakni apa yang perlu dipaparkan pada skrin diakhir aturcara.

## Contoh CARTA IPO

INPUT	Tahun_kelahiran
PROSES	<ol style="list-style-type: none"><li>1. Baca Input Tahun_kelahiran.</li><li>2. Dapatkan tahun semasa daripada sistem komputer, tahun_semasa.</li><li>3. Umur = Tahun_semasa – Tahun_kelahiran</li></ol>
OUTPUT	Umur

# PERWAKILAN ALGORITMA

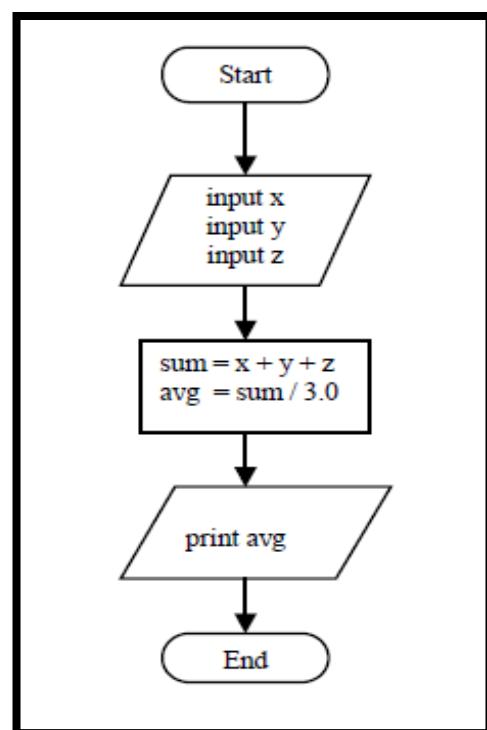
## Pseudokod

Senarai struktur kawalan komputer yang ditulis dalam bahasa pertuturan manusia dan mempunyai nombor turutan.

```
Begin
    input x
    input y
    input z
    sum = x + y + z
    avg = sum / 3.0
    print avg
End
```

## Carta Alir

Alternatif kepada pseudokod, menggunakan simbol grafik untuk mewakili arahan-arahan penyelesaian.



# PSEUDOKOD

- 
- PSEUDOKOD**
- 1. Bukan Bahasa pengaturcaraan komputer.
  - 2. Arahan ditulis dalam Bahasa pertuturan harian.
  - 3. Setiap arahan ialah ungkapan matematik, ungkapan logic, penggunaan struktur kawalan atau penggunaan fungsi komputer.
  - 4. Setiap arahan diletakkan dalam baris baharu yang diberikan nombor siri.

## LANGKAH-LANGKAH MENULIS PSEUDOKOD

- Tulis kenyataan **MULA**.
- Baca **INPUT**.
- Proses data menggunakan ungkapan logik atau matematik.
- Papar **OUTPUT**.
- Tulis kenyataan **TAMAT**.

Pseudocode

```

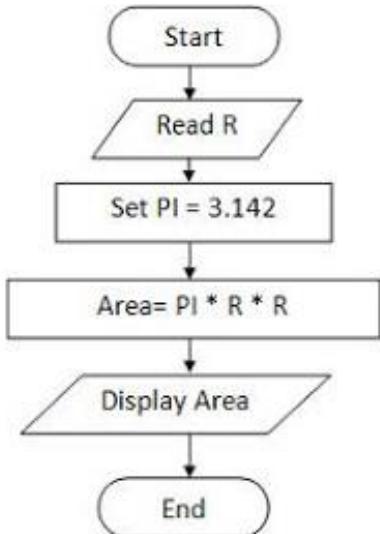
BEGIN
    input hours
    input rate
    pay = hours * rate
    print pay
END
  
```

## 1.2

### 1.2.1

# MENGGUNAKAN ALGORITMA UNTUK MENYATAKAN PENYELESAIAN KEPADA MASALAH

## CARTA ALIR

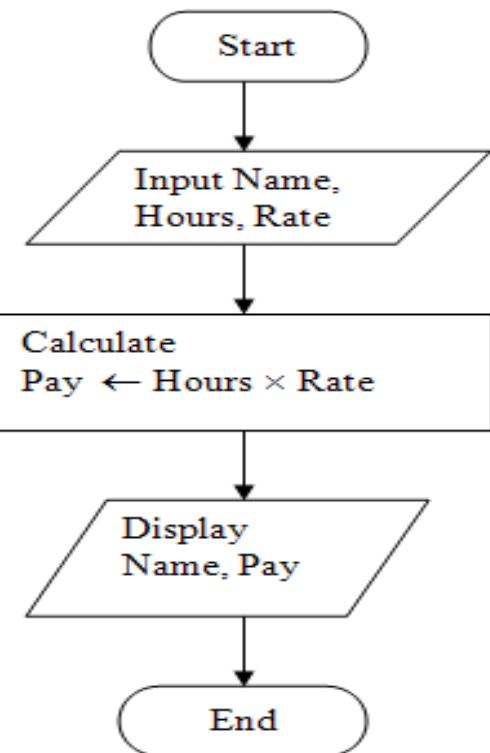


SIMBOL	NAMA NOD	FUNGSI
	Terminal Mula	Permulaan algoritma dalam carta alir.
	Terminal Tamat	Penamat algoritma dalam carta alir.
	Input/Output	Membaca input atau memaparkan output ke skrin
	Proses	Arahan untuk memproses input dalam bentuk ungkapan, memproses fail dan sebagainya.
	Penghubung	Titik sambungan untuk menyambungkan carta alir yang terpisah.
	Syarat	<ul style="list-style-type: none"> <li>Menguji syarat yang terkandung dalam syarat.</li> <li>Terdapat satu anak panah masuk dan 2 anak panah keluar.</li> </ul>
	Aliran aktiviti	Menghubungkan nod-nod untuk menunjukkan aliran proses.

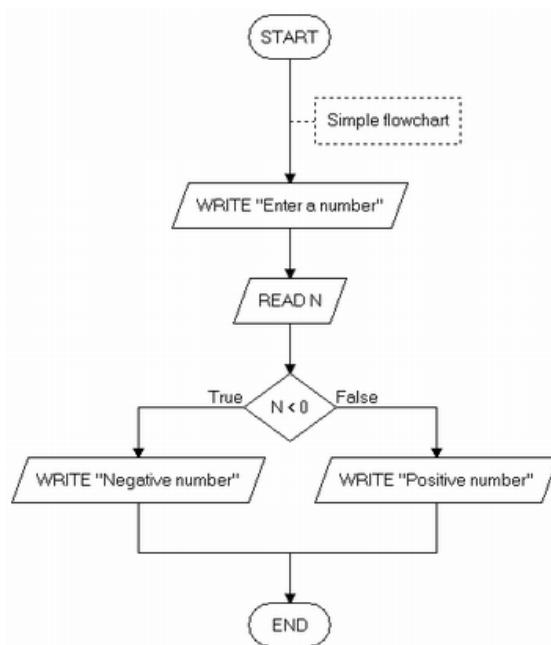
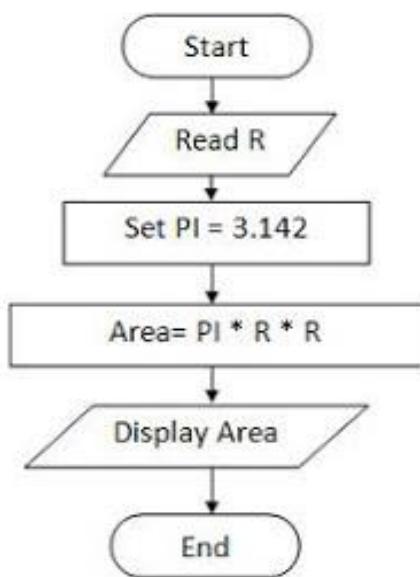
# CARTA ALIR

## LANGKAH-LANGKAH MEMBINA CARTA ALIR

- Lukis nod terminal MULA.
- Lukis garis penghubung.
- Lukis nod input, masukkan butiran seperti umpukan data.
- Lukis garis penghubung.
- Lukis nod proses, Masukkan butiran seperti ungkapan matematik.
- Lukis garis penghubung.
- Sekiranya perlu, lukis nod proses atau nod input lain-lain yang diperlukan,
- Sekiranya tiada, lukis nod terminal tamat.



## CONTOH CARTA ALIR



## STRUKTUR KAWALAN

Struktur Kawalan Urutan

Struktur Kawalan Pilihan

Struktur Kawalan Pengulangan

### STRUKTUR KAWALAN URUTAN

PSEUDOKOD	CARTA ALIR	
Mula Penyataan 1 Penyataan 2 Penyataan 3 Tamat	<pre> graph TD     M([MULA]) --&gt; S1[Penyataan 1]     S1 --&gt; S2[Penyataan 2]     S2 --&gt; S3[Penyataan 3]     S3 --&gt; T([TAMAT])   </pre>	<ul style="list-style-type: none"> <li>Melaksanakan arahan-arahan Komputer satu per satu.</li> <li>Urutan arahan yang betul penting kerana urutan yang berlainan boleh memberikan output yang berlainan.</li> <li>Setiap arahan adalah satu pernyataan algoritma.</li> <li>Urutan algoritma disusun secara linear.</li> </ul>

### STRUKTUR KAWALAN PILIHAN

PSEUDOKOD	CARTA ALIR	
JIKA syarat benar MULA_JIKA BLOK Penyataan 1 TAMAT_JIKA JIKA_TIDAK MULA_JIKA TIDAK BLOK Penyataan 2 TAMAT_JIKA_TIDAK	<pre> graph TD     M([MULA]) --&gt; D{Syarat?}     D -- Benar --&gt; S1[Penyataan 1]     D -- Tidak --&gt; S2[Penyataan 2]     S1 --&gt; T([TAMAT])     S2 --&gt; T   </pre>	<ul style="list-style-type: none"> <li>Memberikan perisian komputer keupayaan untuk membuat keputusan berasaskan syarat yang telah ditentukan oleh pengatur cara.</li> <li>Struktur ini membolehkan arahan-arahan lain komputer dilaksanakan dalam situasi masalah yang berbeza.</li> <li>Ungkapan logik digunakan dalam syarat.</li> <li>Operator Aritmetik digunakan dalam ungkapan logik.</li> </ul>

### STRUKTUR KAWALAN PENGULANGAN

PSEUDOKOD	CARTA ALIR	
SELAGI Syarat MULA_SELAGI BLOK Penyataan TAMAT_SELAGI	<pre> graph TD     M([MULA]) --&gt; D{Syarat?}     D -- Benar --&gt; B[Blok Penyataan]     B --&gt; D     D -- Tidak --&gt; T([TAMAT])   </pre>	<ul style="list-style-type: none"> <li>Mengulang arahan-arahan komputer.</li> <li>Ulangan boleh berlangsung sehingga menerima syarat berhenti atau mencapai bilangan yang ditetapkan.</li> </ul>

## 1.2

### 1.2.3

## MENGUJI DAN MEBAIKI RALAT DALAM ALGORITMA

### PENGUJIAN ALGORITMA

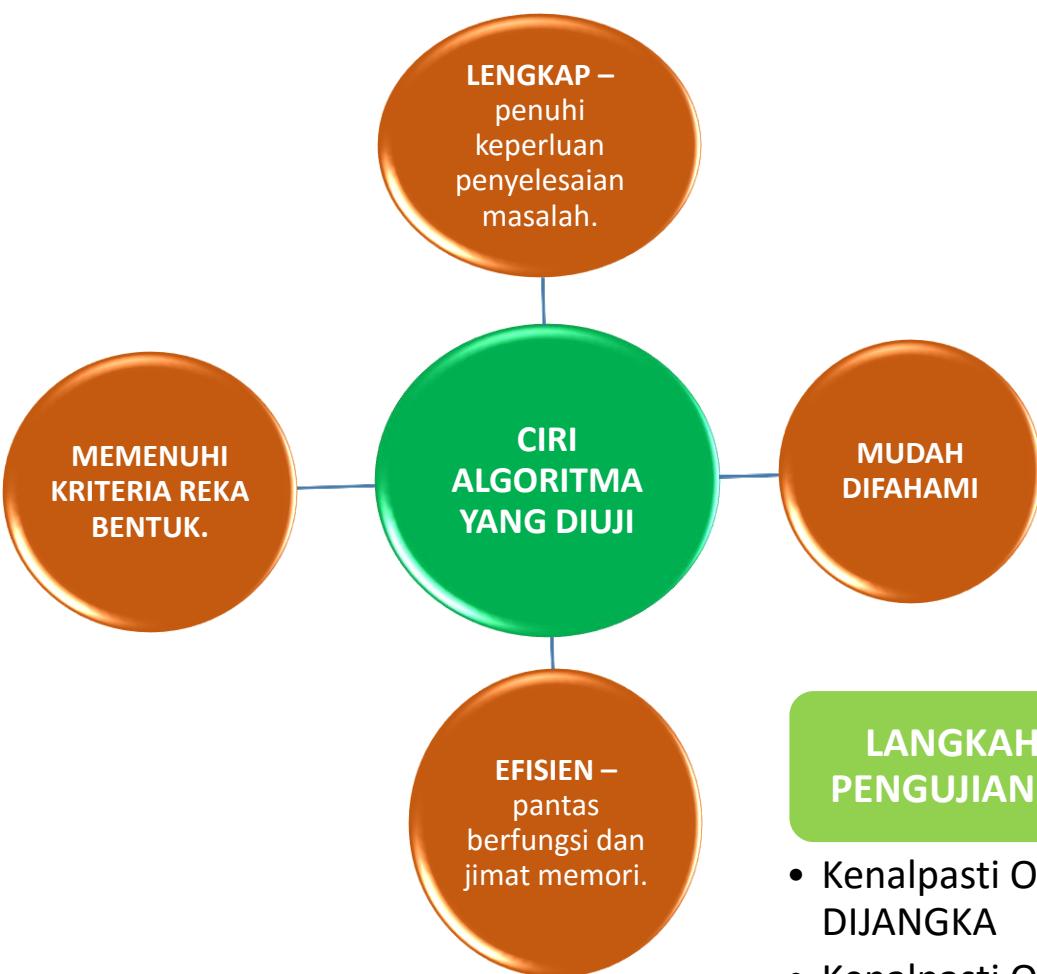
- Algoritma diuji untuk tujuan pembaikan.
- Dibuat sebelum algoritma ditulis sebagai kod komputer.
- MATLAMAT PENGUJIAN - untuk memastikan logik algoritma adalah betul dan memikirkan pembaikan algoritma supaya lebih efisien.

TULIS ALGORITMA

UJI ALGORITMA

PEMBETULAN

PENGATURCARAAN



### LANGKAH-LANGKAH PENGUJIAN ALGORITMA

- Kenalpasti OUTPUT DIJANGKA
- Kenalpasti OUTPUT DIPEROLEH
- Bandingkan OUTPUT DIPEROLEH dengan OUTPUT DIJANGKA.
- Analisis dan baiki algoritma

## JENIS RALAT ALGORITMA

RALAT SINTAKS

RALAT LOGIK

RALAT MASA LARIAN

RALAT SINTAKS

- Tidak wujud dalam algoritma.
- Berlaku kerana **cuai semasa menggunakan Bahasa pengaturcaraan.**
- Biasanya ditemui secara automatik oleh perisian compiler Bahasa pengaturcaraan.
- Ralat algoritma tidak menyebabkan ralat sintaks.

RALAT LOGIK

- Berlaku kerana **perisian** yang dihasilkan **tidak menjalankan fungsi yang sepatutnya, tidak lengkap atau menghasilkan output yang tidak tepat.**
- PUNCA - Ungkapan/formula yang salah, kecuaian, jenis data tidak sesuai, umpukan tidak betul.

RALAT MASA LARIAN

- Ralat yang **timbul apabila aturcara dijalankan.**
- Contoh – Aturcara tidak dapat dimulakan, sangat perlahan atau tidak responsive.
- Boleh dikenalpasti daripada kegagalan output dan paparan amaran dalam aturcara.
- Boleh dikesan melalui reka bentuk algoritma yang tidak efisien atau salah.
- CONTOH – Struktur kawalan tidak betul, pembolehubah tiada nilai, pembahagian dengan sifar, logik syarat salah dalam pengulangan.

**1.2****1.2.4**

## MENGESAN NILAI PEMBOLEH UBAH PADA SETIAP TAHAP DALAM ALGORITMA

### PEMBOLEH UBAH

- Algoritma mengumpuk dan mengubah nilai sesuatu pembolehubah.
- Nilai pembolehubah adalah tidak tetap.
- Setiap baris algoritma mungkin membuat perubahan pada pemboleh ubah tertentu.
- Jadual pemboleh ubah digunakan untuk mengesan nilai pemboleh ubah pada setiap tahap algoritma.

### CONTOH



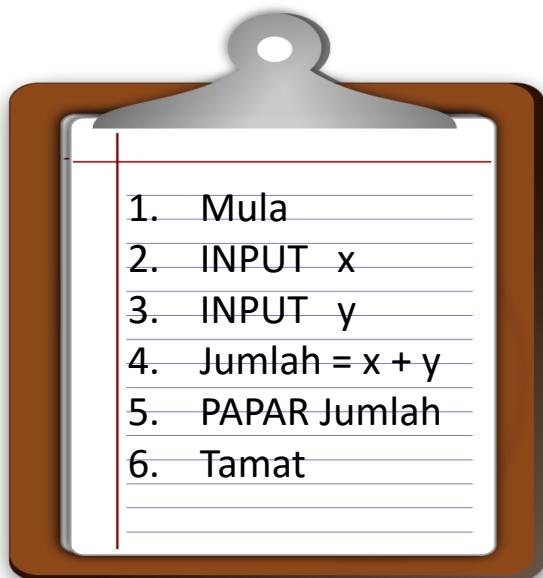
JADUAL BERNOMBOR

NO	z	I/O
1	-	-
2	100	100
3	50	-
4	200	-
5	250	-
6	250	250
7	-	-

## OUTPUT

- Output yang betul bergantung kepada boleh ubah sewaktu algoritma papar dipanggil.
- Membandingkan output dijangka merupakan satu-satunya cara menentukan kesahihan output algoritma.
- Output dijangka ditentukan secara hitungan manual.
- Jika output algoritma adalah betul, output algoritma sepatutnya bersamaan dengan output dijangka.

## CONTOH



**JADUAL BERNOMBOR**

NO	x	y	Jumlah	I/O
1	-	-	-	-
2	12	-	-	-
3	12	88	-	-
4	12	88	$12 + 88 = 100$	
5	12	88	100	Jumlah = 100
6	-	-	-	-

## TERJEMAHAN ALGORITMA

- Setiap baris algoritma yang direka bentuk dapat ditukarkan kepada kod computer.
- Algoritma berbentuk universal.
- Oleh itu, simbol dan perkataan yang digunakan tidak perlu bersandarkan kepada mana-mana Bahasa pengaturcaraan.
- Contoh Bahasa pengaturcaraan ialah Visual Basic (VB), Java,C# dan lain-lain.

## CONTOH

ALGORITMA	KOD KOMPUTER (JAVA)
Mula	Public static void main (String[] args) {
PAPAR “ Nama Pengguna”	System.out.print (“ Masukkan nama pengguna : ”);
INPUT Nama	String nama new java.util. Scanner (System.in) nextLine();
PAPAR “ Apa Khabar”. Nama , “?”	System.out.println (“ Hello ” + nama);
Tamat	;

# PEMBOLEH UBAH, PEMALAR DAN JENIS DATA

- **Pemboleh Ubah** atau **pemalar** dalam Java perlu diisytiharkan sebelum digunakan.
- Dalam proses pengaturcaraan, seorang pengatur cara perlu mengisytiharkan **jenis data** yang diperlukan dalam sesuatu program yang hendak dilaksanakan.

```
public class Class{
    public static void method() {
        ArrayList list = new ArrayList();
        list.add("string");
        anotherMethod("string");
    }
    public static String returnString() {
        return s;
    }
}
```

**Expressions**  
 "string"  
 anotherMethod(...)

```
public class Demo {
    public static void main(String[] args) throws IOException {
        //declare new File and Scanner objects
        File file = new File("input.txt");
        Scanner inputFile = new Scanner(file);
        //loop through txt file
        while(inputFile.hasNext()){
            //read next line
            String line = inputFile.nextLine();
            System.out.print(line);
            //call check method to determine balance
            if(check(line))
                System.out.print("\t--> correct\n");
            else
                System.out.print("\t--> incorrect\n");
        }
        inputFile.close();
    }
}
```

# PEMBOLEH UBAH, PEMALAR DAN JENIS DATA

## PEMBOLEH UBAH

### PEMBOLEH UBAH

- Ruang simpanan sementara untuk nombor , teks dan objek.
- Nilai pemboleh ubah sentiasa berubah semasa berlakunya pemprosesan data dan tidak akan memegang sebarang nilai selepas program tamat.
- Pengatur cara perlu memberikan nama kepada setiap pemboleh ubah yang digunakan untuk menyelesaikan sesuatu masalah dalam program yang dibangunkan.
- Pengatur cara menggunakan nama pemboleh ubah sebagai nama rujukan untuk nilai spesifik pemboleh ubah tersebut.
- Komputer menggunakan nama pemboleh ubah sebagai rujukan untuk mencari nilai pemboleh ubah itu dalam memorinya

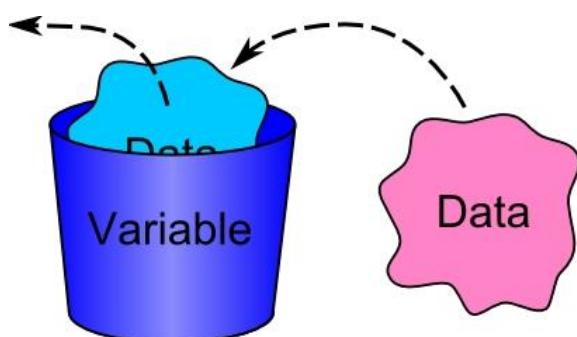
**When you declare a variable the computer:**

- Allocates some **memory** large enough to hold the data
- Assigns that memory block the **identifier** you entered

```
int num1;
int num2;
int result;
```

MEMORY

num1	
result	
num2	



# PEMBOLEH UBAH, PEMALAR DAN JENIS DATA

## PEMALAR

### PEMALAR

- Nilai pemalar adalah tetap dan tidak akan berubah sewaktu proses pengaturcaraan dilaksanakan.
- Digunakan semasa pengatur cara ingin mengisyiharkan nilai yang tidak akan berubah.
- Jenis data yang diisyiharkan mestilah sepadan dengan nilai.

#### When you assign a value with the = sign

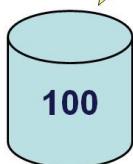
- The value is stored in the memory location for that variable

```
num1 = 10; // assigning
num2 = 7;
result = num1 + num2;
```

MEMORY	
num1	10
result	17
num2	7

Container named  
“Count” holding  
a value 100

count = 100;



Variable\_name = value;

Can be any user  
defined name

Assignment  
operator

Constant

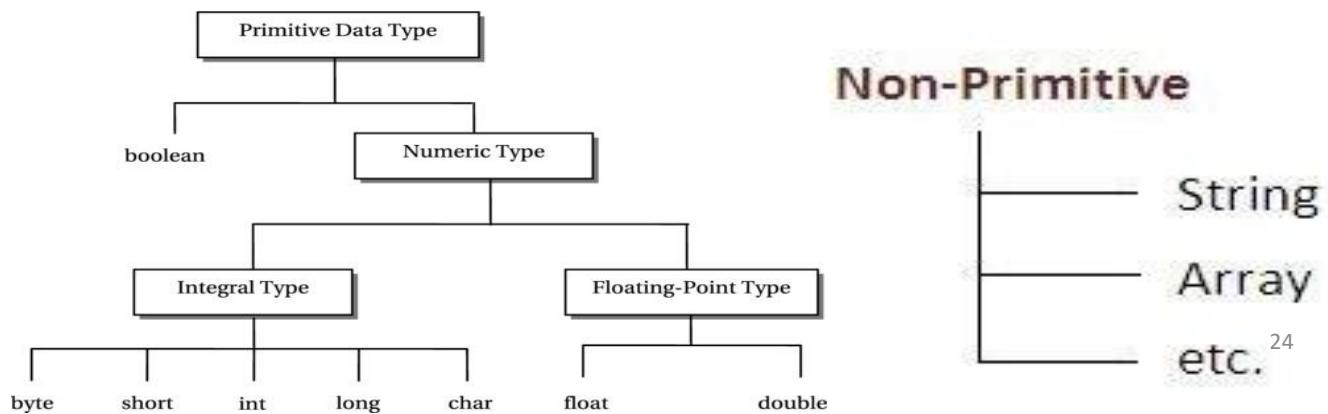
# PEMBOLEH UBAH, PEMALAR DAN JENIS DATA

## JENIS DATA

### JENIS DATA

- Satu set data yang mempunyai nilai dan ciri-ciri yang telah ditetapkan.
- Data akan diproses menghasilkan output.
- Jenis data boleh dikategorikan kepada 2 kelas iaitu data primitive dan data bukan primitive.

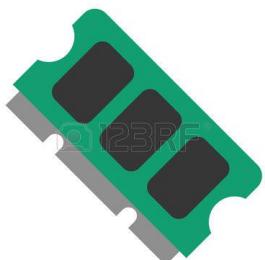
JENIS DATA	CONTOH NILAI		KAPASITI INGATAN KOMPUTER
Integer	Minimum	-2147483648	4 bait
	Maksimum	2147483647	
<i>float</i>	Minimum	-3.4e38	4 bait
	Maksimum	3.4e38	
<i>double</i>	Minimum	-1.7e308	8 bait
	Maksimum	3.4e38	
<i>char</i>	Satu karakter sahaja		2 bait
<i>String</i>	Bermula dari 0 hingga tiada had.		> 10 bait
<i>Boolean</i>	Benar (true)		1 bait
	Palsu (false)		



## 1.3

### 1.3.1

#### MENJELASKAN HUBUNGAN ANTARA JENIS DATA, SAIZ DATA DAN KAPASITI INGATAN KOMPUTER



- Setiap jenis data akan disimpan dalam ingatan komputer.
- Nama pemboleh ubah memainkan peranan yang penting dalam menentukan saiz data dalam ingatan.
- Sistem operasi akan menentukan apa-apa yang boleh disimpan dalam ingatan computer berdasarkan jenis data yang digunakan oleh pemboleh ubah.
- Kepelbagaiannya penggunaan jenis data pada pemboleh ubah dapat menjimatkan ruang pada ingatan komputer.

## 1.3

### 1.3.2

#### MEMILIH DAN MENGGUNAKAN JENIS DATA YANG BERSESUAIAN

- Pemilihan dan penggunaan data yang sesuai amat penting supaya aturcara dapat dibangunkan tanpa ralat sintaks.
- Jenis data bagi pemboleh ubah dan pemalar yang menentukan jenis maklumat akan disimpan dalam ruang ingatan yang diperlukan.
- Penggunaan jenis data yang sesuai berfungsi sebagai penanda aras kepada sesuatu pemboleh ubah itu sama ada pemboleh ubah tersebut menyimpan data yang tetap, data yang boleh dikira, huruf, nombor perpuluhan atau data yang mempunyai nilai benar atau palsu.

### JENIS DATA

#### INTEGER

**Int**  
Merangkumi semua nombor yang tidak mempunyai nilai pecahan atau perpuluhan.

#### NOMBOR NYATA

**float, double**  
Terdiri daripada semua nombor yang mempunyai titik perpuluhan atau bahagian pecahan.

#### AKSARA / RENTETAN

**char, string**  
Data dalam bentuk ruang kosong (space), teks, perkataan atau nilai yang mengandungi susunan aksara atau simbol.

#### BOOLEAN

Data dalam bentuk pilihan iaitu memilih salah satu daripada sesuatu yang benar atau palsu.

**1.3**

**1.3.3**

## **MEMBEZAKAN PEMBOLEH UBAH SEJAGAT (GLOBAL) DAN SETEMPAT (LOCAL)**

- Setiap pemboleh ubah dan pemalar mungkin wujud dan akan digunakan untuk keseluruhan aturcara atau hanya bagi satu fungsi.
- Kewujudan pemboleh ubah atau pemalar dikenal sebagai kawasan yang kedua-duanya boleh digunakan secara sah.

### **SKOP PEMBOLEH UBAH**

**PEMBOLEH UBAH SEJAGAT  
(GLOBAL)**

**PEMBOLEH UBAH SETEMPAT  
(LOCAL)**

- Hanya berfungsi dalam aturcara sahaja.
- Apabila tatacara tamat, ruang memori juga tamat.
- **PENGISYTIHARAN** : Diluar di mana-mana fungsi.
- **AKSES** : Boleh diakses dimana-mana fungsi.
- Boleh digunakan hingga ke akhir aturcara.
- Jika pemboleh ubah setempat mempunyai nama yang sama dengan pemboleh ubah sejagat , rujukan hanya dibuat terhadap pemboleh ubah terdekat (setempat).

- Hanya berfungsi dalam subatur cara yang diisytiharkan.
- Digunakan dalam fungsi dimana pemboleh ubah diisytiharkan bermula dari mana pemboleh ubah diisytiharkan dan bila penamat akhir tatacara tersebut.
- **PENGISYTIHARAN** : Dalam sebuah fungsi dalam atur cara.
- **AKSES** : Tidak boleh diakses diluar fungsi itu.
- Tiada masalah jika dua fungsi menggunakan pemboleh ubah tempatan yang sama.

```
1 global variable
2
3 var global = 10;
4
5 function func() {
6
7     var local = 5;
8
9 }
10 local variable
```

# PENGISYTIHARAN NILAI PEMBOLEH UBAH

- Pembelah ubah dikenal sebagai tempat untuk menyimpan data.
- Setiap pembelah ubah dalam *Java* mempunyai jenis yang tertentu yang menentukan saiz dan susun atur memori dan set operasi yang boleh digunakan.
- Semua pembelah ubah perlu diisytiharkan sebelum boleh digunakan.
- Pengisytiharan pembelah ubah perlu dilaksanakan dengan memberikan jenis data dan nama pembelah ubah.

JENIS DATA	NAMA PEMBOLEH UBAH	CONTOH
INTEGER	x	<pre>public class pembelahUbah {     public static void main(String[] args {         int x;     } }</pre>
STRING	nama	<pre>public class pembelahUbah {     public static void main(String[] args {         String nama;     } }</pre>
DOUBLE	s , t, u	<pre>public class pembelahUbah {     public static void main(String[] args {         double s,t,u;     } }</pre>
BOOLEAN	v	<pre>public class pembelahUbah {     public static void main(String[] args {         boolean v;     } }</pre>
CHAR	w	<pre>public class pembelahUbah {     public static void main(String[] args {         char w;     } }</pre>
FLOAT	y	<pre>public class pembelahUbah {     public static void main(String[] args {         float y;     } }</pre>

# 1.3

## 1.3.4

### MENGISYTIHARKAN, MEMULAKAN DAN MENETAPKAN NILAI PADA PEMBOLEH UBAH DAN PEMALAR

## PERMULAAN DAN PENETAPAN NILAI PEMBOLEH UBAH

- Selepas pengisytiharan jenis dan nama pemboleh ubah, nilai kepada pemboleh ubah boleh ditetapkan atau diumpukan.

JENIS DATA	NILAI	CONTOH
INTEGER	10	public class pembolehUbah { public static void main(String[] args { <b>int x = 10</b> ; } }
STRING	hardeep	public class pembolehUbah { public static void main(String[] args { <b>String nama = "hardeep"</b> ; } }
DOUBLE	s = 0.123 t = 1.1 u = s + t	public class pembolehUbah { public static void main(String[] args { <b>double s = 0.123, t = 1.1, u = s + u</b> ; } }
BOOLEAN	true	public class pembolehUbah { public static void main(String[] args { <b>boolean v = true</b> ; } }
CHAR	y	public class pembolehUbah { public static void main(String[] args { <b>char w = 'y'</b> ; } }
FLOAT	342.234f	public class pembolehUbah { public static void main(String[] args { <b>float y = 342.234f</b> ; } }

**1.3**

**1.3.4**

**MENGISYTIHARKAN, MEMULAKAN DAN  
MENETAPKAN NILAI PADA PEMBOLEH  
UBAH DAN PEMALAR**

## **PENGISYTIHARAN DAN PENETAPAN NILAI PEMALAR**

- Pemalar ialah pemboleh ubah yang mempunyai **nilai yang tidak berubah** semasa menjalankan atur cara.

```
final int BILANGAN_HARI_DALAM_SEMINGGU = 7
```

```
final double Pi = 3.14
```

```
Static final double = 4.14
```

**1.3**

**1.3.5**

## MENGGUNAKAN PERNYATAAN UMPUKAN DAN PERNYATAAN ARITMETIK

### CONTOH JENIS PERNYATAAN

#### PERNYATAAN UMPUKAN

- Boleh terdiri daripada satu atau lebih ungkapan yang lain.
- Merujuk “**sama dengan**” atau simbol “**=**” .
- Pernyataan umpukan akan memberikan nilai kepada pembolehubah.

#### PERNYATAAN ARITMETIK

- Boleh terdiri daripada satu atau lebih ungkapan aritmetik.
- Merujuk kepada **operasi aritmetik** (**+, -, \*, /**).
- Unit pemprosesan utama boleh membaca operasi aritmetik dari kiri ke kanan sahaja.
- Jika pengaturcara ingin menjalankan operasi darab terlebih dahulu, penggunaan simbol kurungan () perlu diutamakan.

### CONTOH

```
String x;  
x = "Hello";  
  
double p;  
p = 0.123;  
  
float w;  
w = 561.123f;  
  
int p;  
int q;  
int r;  
p = q =r = 255;
```

### CONTOH

$(30 - 5 + ((2*3) / 3)$

## CONTOH PENYATAAN ARITMETIK DALAM ATURCARA

NOTASI	CONTOH ATURCARA	CONTOH OUTPUT
OPERASI TAMBAH  +	<pre>public class operasiTambah {     public static void main(String[] args {         int i = 10;         int j = 20;         i = i + j;         System.out.println (" Hasil Tambah ialah : " + i );     } }</pre>	Hasil Tambah ialah : 30
OPERASI TOLAK  -	<pre>public class operasiTolak {     public static void main(String[] args {         int i = 30;         int j = 20;         i = i - j;         System.out.println (" Hasil Tolak ialah : " + i );     } }</pre>	Hasil Tolak ialah : 10
OPERASI DARAB  *	<pre>public class operasiDarab {     public static void main(String[] args {         int i = 30;         int j = 20;         i = i * j;         System.out.println (" Hasil Darab ialah : " + i );     } }</pre>	Hasil Darab ialah : 600
OPERASI BAHAGI  /	<pre>public class operasiBahagi {     public static void main(String[] args {         int i = 30;         int j = 2;         i = i / j;         System.out.println (" Hasil Bahagi ialah : " + i );     } }</pre>	Hasil Bahagi ialah : 15

**1.3**

**1.3.6**

**MENULIS ATUR CARA UNTUK  
MEMASUKKAN INPUT DARI PAPAN  
KEKUNCI DAN MEMAPARKAN OUTPUT**

## **ATUR CARA YANG MEMBENARKAN INPUT DIMASUKKAN MENGGUNAKAN PAPAN KEKUNCI**

```
import java.util.Scanner;  
Scanner input = new Scanner (System.in);
```

### **CONTOH #1 (Input dari pengguna)**

```
import java.util.Scanner ;  
public class luasSegiTiga {  
  
    public static void main (String [] args) {  
        Scanner input = new Scanner (System.in);  
        System.out.println ("Sila masukkan nilai tapak : ");  
        int tapak = input.nextInt();  
  
        System.out.println ("Sila masukkan nilai tinggi : ");  
        int tinggi = input.nextInt();  
  
        double luas = (1.0/2) * tapak * tinggi ;  
  
        System.out.println ("Luas segi tiga ialah : " + luas );  
    }  
}
```

**OUTPUT**

Sila masukkan nilai tapak :

**6**

Sila masukkan nilai tinggi :

**4**

Luas segi tiga ialah : **12.0**

## CONTOH #1 (Nilai dalam atur cara)

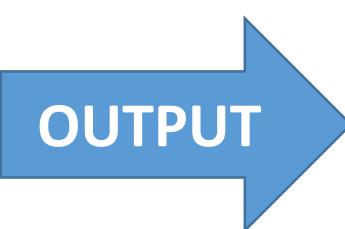
```
public class luasSegiTiga2 {  
  
    public static void main (String [] args) {  
  
        int tapak = 6;  
  
        int tinggi = 4;  
  
        double luas = (1.0/2) * tapak * tinggi ;  
  
        System.out.println ("Luas segi tiga ialah : " + luas );  
    }  
}
```

OUTPUT

Luas segi tiga ialah : **12.0**

## CONTOH #2 (Input dari pengguna)

```
import java.util.Scanner ;  
  
public class infoAnda {  
  
    public static void main (String [] args) {  
  
        Scanner input = new Scanner (System.in);  
        System.out.println ("Siapakan nama anda ? : " );  
        String nama = input.nextInt();  
  
        System.out.println ("Berapakah umur anda ? : " );  
        int umur = input.nextInt();  
  
        System.out.println ("Apakah hobi anda ? : " );  
        String hobi = input.nextInt();  
  
        nama = nama;  
        umur = umur;  
        hobi = hobi;  
  
        System.out.println ("Nama : " + nama );  
        System.out.println ("Umur : " + umur );  
        System.out.println ("Hobi : " + hobi );  
    }  
}
```

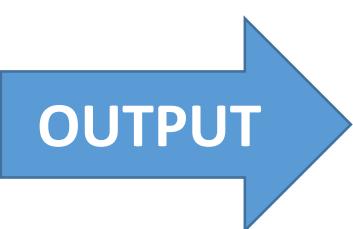


OUTPUT

Siapakan nama anda ? :  
**Haziq**  
Berapakah umur anda ? :  
**7**  
Apakah hobi anda ? :  
**Bermain**  
**Nama : Haziq**  
**Umur : 7**  
**Hobi : Bermain**

## CONTOH #2 (Nilai dalam atur cara)

```
public class infoAnda2 {  
  
    public static void main (String [] args) {  
  
        String nama ;  
        int umur ;  
        String hobi;  
  
        nama = "Haziq";  
        umur = 7;  
        hobi = "Bermain";  
  
        System.out.println ("Nama : " + nama );  
        System.out.println ("Umur : " + umur );  
        System.out.println ("Hobi : " + hobi );  
  
    }  
}
```



OUTPUT

Nama : Haziq  
Umur : 7  
Hobi : Bermain

## 1.4

# STRUKTUR KAWALAN

## STRUKTUR KAWALAN

### Struktur Kawalan Urutan

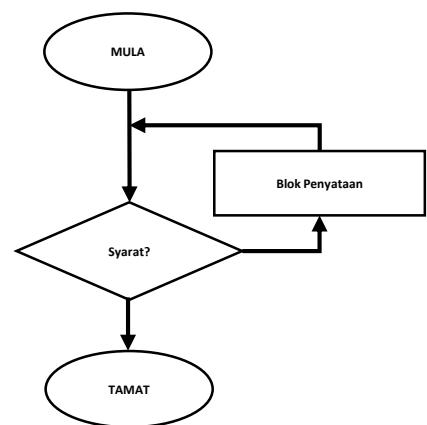
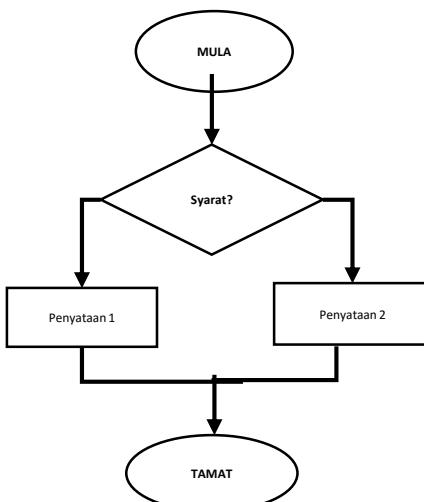
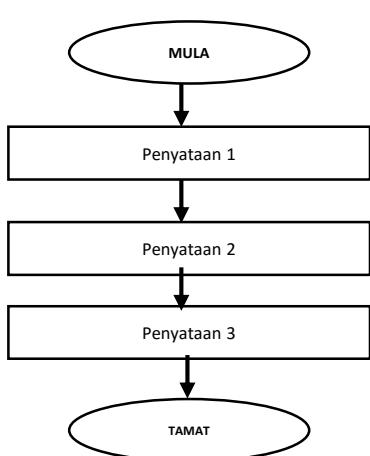
- Tidak bervariasi.
- Hanya mengikut urutan.

### Struktur Kawalan Pilihan

- if – else - if
- Switch-case

### Struktur Kawalan Pengulangan

- For
- While
- Do-while



**1.4**

**1.4.1**

# **STRUKTUR KAWALAN PILIHAN**

## **KAWALAN PILIHAN**



- Mekanisme yang membolehkan keputusan atau pemilihan dibuat secara automatik..
- PERNYATAAN SYARAT BOOLEAN – Digunakan untuk menguji nilai input yang dimasukkan dan ini seterusnya akan menentukan set atau blok arahan yang akan dilaksanakan.
- Syarat boolean membolehkan perbandingan memboleh ubah, sifat objek atau nilai yang dilakukan melalui operator hubungan atau operator logikal.
- Perbandingan ini memberikan keputusan dalam bentuk data jenis boolean.



### **BENTUK STRUKTUR KAWALAN PILIHAN**

**if**

**If-else**

**If-else-if**

**switch-case**

## KAWALAN PILIHAN *if*

- Hanya akan melaksanakan pernyataan-pernyataan tertentu seperti memproses data melalui ungkapan sekiranya syarat adalah benar.

CARTA ALIR	SINTAKS
<pre> graph TD     M([MULA]) --&gt; D{Adakah Syarat benar?}     D -- Benar --&gt; L[Laksana kenyataan ini jika benar]     L --&gt; T([TAMAT])     D --&gt; T   </pre>	<p>If (&lt;syarat Boolean&gt;) {      &lt;Arahan-arahan jika benar&gt;      }</p> <p><b>CONTOH</b></p> <pre> If ( umur &gt; 20 ) {     System.out.println( "Anda layak mengundi. " ); }</pre>

# KAWALAN PILIHAN *if-else*

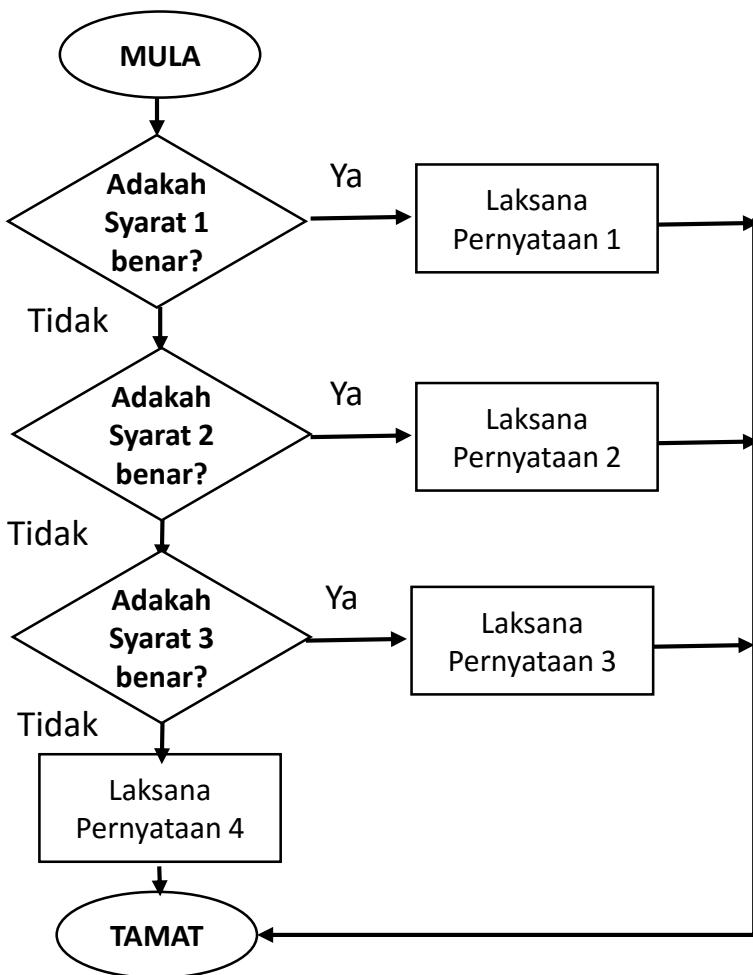
- Digunakan untuk membuat keputusan dalam sesuatu arah cara.
- Menunjukkan hasil Boolean – Ya (Benar) atau Tidak (Palsu).
- Pernyataan susulan bergantung kepada hasil Boolean tersebut.

CARTA ALIR	SINTAKS
<pre> graph TD     M([MULA]) --&gt; D{Adakah Syarat benar?}     D -- Palsu --&gt; L1[Laksana pernyataan ini jika palsu]     D -- Benar --&gt; L2[Laksana pernyataan ini jika benar]     L1 --&gt; T([TAMAT])     L2 --&gt; T   </pre>	<pre> <i>If (&lt;syarat Boolean&gt;) {</i>   &lt;Arahan-arahan jika benar&gt; } <i>else</i>   &lt;Arahan-arahan jika palsu&gt; }   </pre>
	<p><b>CONTOH</b></p> <pre> If ( umur &gt; 20 ) {   System.out.println ( "Anda layak mengundi. " ); } <i>else</i>   System.out.println ( "Anda tidak layak mengundi. " ); }   </pre>

# KAWALAN PILIHAN *if-else-if*

- Untuk membuat keputusan yang lebih kompleks.
- Mencuba syarat Boolean yang baharu sekiranya syarat terdahulu menghasilkan keputusan palsu.
- Sekiranya syarat Boolean menghasilkan keputusan benar, pernyataan akan dilaksanakan. Syarat Boolean lain tidak akan diuji.

## CARTA ALIR



## SINTAKS

```

if (<syarat Boolean1_benar>)
{
  Pernyataan01

} else if (<syarat Boolean2_benar>)
{
  Pernyataan02

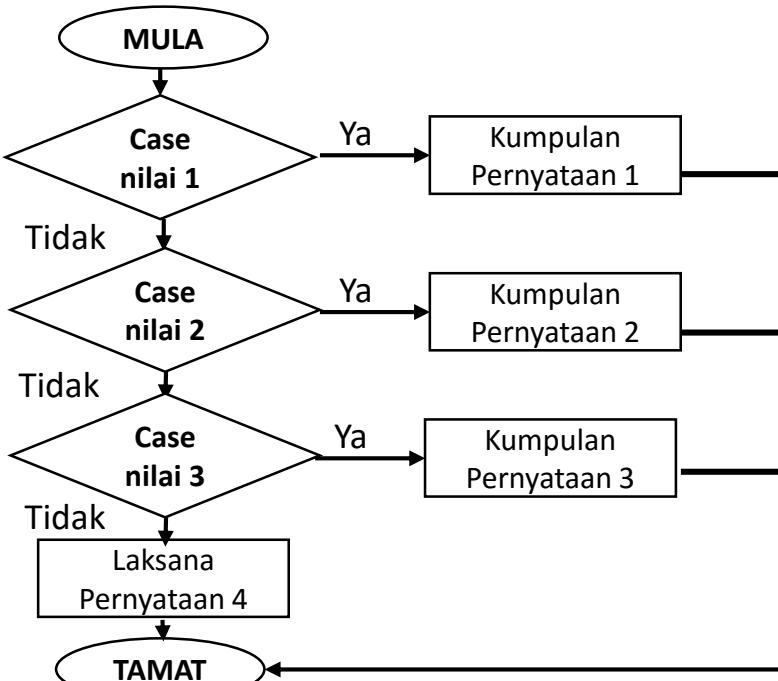
} else if (<syarat Boolean3_benar>)
{
  Pernyataan03

} else
{
  Pernyataan04
}
  
```

# KAWALAN PILIHAN *switch-case*

- Untuk mengatasi kekurangan penggunaan *if-else-if* (pernyataan perlu diulang banyak kali dan boleh mengelirukan pengguna).
- Struktur ini lebih mudah difahami.
- Ujian Switch* : ungkapan nombor, abjad atau rentetan.
- Case* : Mengandungi nilai yang akan dipadankan dengan ujian switch.
- Break* : Digunakan sebagai arahan untuk keluar dari blok *switch*. Jika ungkapan *break* tidak disertakan, pernyataan selepas *break* akan dilaksanakan.
- Default* : Kadang-kadang dimasukkan sebagai langkah tambahan. Pernyataan *default* akan dilaksanakan jika ujian *switch* tidak bersamaan dengan mana-mana nilai *case*.

## CARTA ALIR



## SINTAKS

```

switch (ujian) {
  case : nilai1 {
    Kumpulan Pernyataan 1
    break ;
  }
  case : nilai2 {
    Kumpulan Pernyataan 2
    break ;
  }
  case : nilai3 {
    Kumpulan Pernyataan 3
    break ;
  }
  default : {
    Kumpulan Pernyataan 4
  }
}
  
```

# 1.4

## 1.4.2

### MENULIS ATUR CARA MENGUNAKAN STRUKTUR KAWALAN PILIHAN DENGAN MENGGABUNGKAN OPERATOR HUBUNGAN DAN OPERATOR LOGICAL

## OPERATOR

### OPERATOR HUBUNGAN

### OPERATOR LOGIKAL

- Digunakan untuk membandingkan dua nilai bagi menghasilkan keputusan Boolean.

#### OPERATOR HUBUNGAN

`==` sama dengan

`!=` tidak sama dengan

`>` lebih besar daripada

`>=` lebih besar daripada atau sama dengan

`<` Kurang daripada

`<=` Kurang atau sama dengan

- Digunakan untuk menghasilkan beberapa ungkapan Boolean bagi menghasilkan syarat yang lebih kompleks.

#### OPERATOR LOGIKAL

AND

OR

NOT

#### Operator Logikal AND

- Digunakan apabila dua atau lebih syarat Boolean perlu digabungkan dan **kesemua syarat perlu benar**.
- Ditulis dengan simbol “`&&`”.

#### Operator Logikal OR

- Digunakan apabila dua atau lebih syarat Boolean perlu digabungkan dan **hanya salah satu syarat perlu benar**.
- Ditulis dengan simbol “`||`”.

#### Operator Logikal NOT

- Menukarkan nilai Boolean kepada lawannya.
- Ditulis dengan simbol “`!`”.

x	NOT x
TRUE	FALSE
FALSE	TRUE

**1.4**

**1.4.2**

**MENULIS ATUR CARA MENGUNAKAN  
STRUKTUR KAWALAN PILIHAN DENGAN  
MENGGABUNGKAN OPERATOR HUBUNGAN  
DAN OPERATOR LOGICAL**

# **CONTOH ATURCARA OPERATOR HUBUNGAN**

<b>CONTOH ATURCARA</b>	<b>OUTPUT</b>
<pre>public class contoh31a {      public static void main (String [] args) {         int nombor = 15;          if (nombor &gt; 0)             System.out.println ("Nombor ini adalah integer positif" );          else             System.out.println ("Nombor ini adalah bukan integer positif" );     } }</pre>	Nombor ini adalah integer positif
<pre>public class contoh31b {      public static void main (String [] args) {         int nombor = -7;          if (nombor &gt; 0)             System.out.println ("Nombor ini adalah integer positif" );          else if (nombor == 0)             System.out.println ("Nombor ini adalah sifar" );          else             System.out.println ("Nombor ini adalah integer negatif" );     } }</pre>	Nombor ini adalah integer negatif

# 1.4

## 1.4.2

### MENULIS ATUR CARA MENGUNAKAN STRUKTUR KAWALAN PILIHAN DENGAN MENGGABUNGKAN OPERATOR HUBUNGAN DAN OPERATOR LOGICAL

## CONTOH ATURCARA OPERATOR LOGIKAL

CONTOH ATURCARA	OUTPUT
<pre>public class contoh32 {      public static void main (String [] args) {         int markah = 55;          if (markah &gt;= 0 &amp;&amp; markah &lt;= 100)             System.out.println ("Markah yang dimasukkan adalah sah." );          else             System.out.println ("Markah yang dimasukkan adalah tidak sah" )     } }</pre>	Markah yang dimasukkan adalah sah.
<pre>public class contoh33 {      public static void main (String [] args) {         boolean malam = true;         boolean hujan = false;          if (malam    hujan)             System.out.println ("Angkat baju" );     } }</pre>	Angkat baju
<pre>public class contoh34 {      public static void main (String [] args) {         boolean lulus;         int markah = 83;          if (markah &gt;= 40)             lulus = true;         else             lulus = false;         if (!lulus)             System.out.println ("Anda perlu mengulangi ujian" );         else             System.out.println ("Anda lulus" );     } }</pre>	Anda lulus

**1.4**

**1.4.2**

**MENULIS ATUR CARA MENGUNAKAN  
STRUKTUR KAWALAN PILIHAN DENGAN  
MENGGABUNGKAN OPERATOR HUBUNGAN  
DAN OPERATOR LOGICAL**

## **PENGGABUNGAN OPERATOR HUBUNGAN DAN OPERATOR LOGIKAL DALAM STRUKTUR KAWALAN PILIHAN**

- Operator hubungan dan operator logikal boleh digabungkan dalam struktur kawalan pilihan.
- CONTOH : Markah  $\geq 0 \ \&\& \text{ Markah} \leq 100$

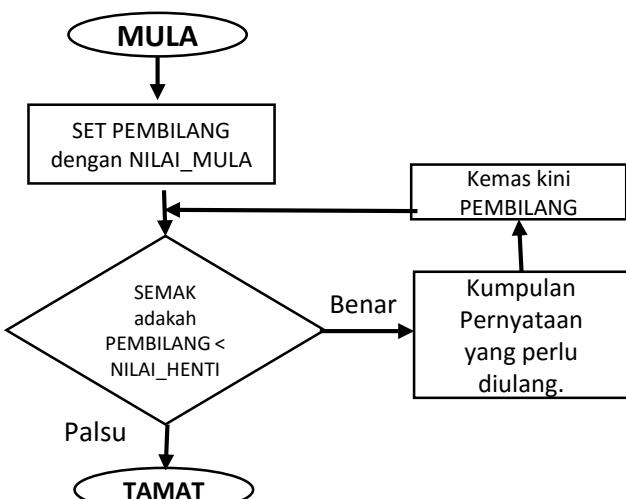
<b>CONTOH ATURCARA</b>	<b>OUTPUT</b>
public class contoh35 {  public static void main (String [] args) { double celcius = 39 , Fahrenheit = 97 ;  if (celcius > 39    Fahrenheit > 98.6) System.out.println ("Anda mungkin demam panas." );  } }	Anda mungkin demam panas.

## KAWALAN ULANGAN

### Ulangan berdasarkan pembilang (For)

- Untuk bilangan tertentu.
- Ditentukan oleh pemboleh ubah pembilang yang bermula dengan nombor indeks tertentu seperti 0 dan 1.
- Nombor indeks akan ditambah secara automatik pada akhir blok pernyataan.
- Penambahan dibuat setiap kali blok kenyataan telah diulang dan akan berlanjut sehingga syarat Boolean berulang menjadi tidak benar.

### CARTA ALIR



### SINTAKS

```

for ( pemula ; penamat ; penambah)
{
< Blok pernyataan yang perlu diulang>
}
  
```

### Ulangan berdasarkan syarat (While, Do-While)

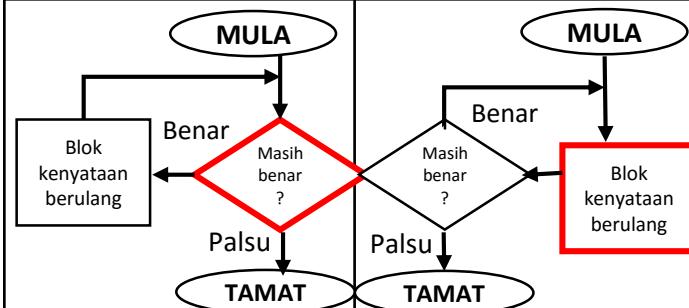
#### While

- Membuat ujian terlebih dahulu ke atas input.
- Jika memenuhi syarat, blok arahan dalam gelung akan dilaksanakan.

#### Do-While

- Membuat ujian selepas blok arahan dalam gelung dilaksanakan.

### CARTA ALIR



### SINTAKS

```

While (<Syarat Boolean >)
{
<Blok kenyataan  
berulang>
}
  
```

```

Do
<Blok kenyataan  
berulang>

Loop While
(<Syarat Boolean >)
  
```

# CONTOH ATURCARA *for*

CONTOH ATURCARA	OUTPUT
<pre>public class contoh36 {     public static void main (String [] args) {         int i ;         for ( i = 1 ; i &lt;=100 ; i+=1) {             System.out.print ("i" );         }     } }</pre>	12345678910.
<pre>public class contoh37 {     public static void main (String [] args) {         int jumlah = 0 ;         for ( i = 2 ; i &lt;=100 ; i+=1) {             jumlah = jumlah + i ;         }         System.out.println ("Jumlah : " + jumlah );     } }</pre>	5049
<pre>public class contoh38 {     public static void main (String [] args) {         for ( i = 0 ; i &lt;= 30 ; i+=1) {             if ((i % 2) == 2)                 System.out.print ( i + "," );         }     } }</pre>	1,3,5,7,9,11,15,17,19,23,25,27,29
<pre>public class contoh39 {     public static void main (String [] args) {         double baki = 500.0;         for ( i = 0 ; i &lt;= 5 ; i+=1) {             baki = baki + (0.1 * baki);         }         System.out.println ( " Baki 5 tahun : " + baki) ;     } }</pre>	Baki 5 tahun : 805.255

**1.4****1.4.3**

## MENERANGKAN ALIRAN STRUKTUR KAWALAN ULANGAN

# CONTOH ATURCARA *while*

CONTOH ATURCARA	OUTPUT
<pre>public class contoh40 {      public static void main (String [] args) {         int n = 5 ;         while ( n &gt; 0 ) {             System.out.println (n + “ , ” );             n = n-1;         }     } }</pre>	5,4,3,2,1
<pre>public class Module1 {      public static void main (String [] args) {         Scanner scanner = new Scanner (System.in);         int nom = scanner.nextInt ();          while (nom &gt; 0 ) {             nom - = 1;             System.out.println ( nom + “ ” );         }     } }</pre>	
<pre>public class contoh42 {      public static void main (String [] args) {         String strPassword = new String ();         Scanner scanner = new Scanner (System.in);         final String RekodLaluanRahsia = “Pisang” ;          while (!strPassword.equal (RekodLaluanRahsia) ) {             System.out.print ( “ Sila masukkan password : ” );             strPassword = scanner.next();             System.out.print ( )         }     } }</pre>	Sila masukkan password : Pisang

# CONTOH ATURCARA *do-while*

CONTOH ATURCARA	OUTPUT
<pre>public class contoh43 {     public static void main (String [] args) {         int no = 1 ;         do {             System.out.println (no + " X 3 = " + no * 3) ;             no = no + 1;         } while ( n &lt;= 12)     } }</pre>	$1 \times 3 = 3$ $2 \times 3 = 6$ $3 \times 3 = 9$ $4 \times 3 = 12$ $5 \times 3 = 15$ $6 \times 3 = 18$ $7 \times 3 = 21$ $8 \times 3 = 24$ $9 \times 3 = 27$ $10 \times 3 = 30$ $11 \times 3 = 33$ $12 \times 3 = 36$
<pre>public class contoh44 {     public static void main (String [] args) {         double no ;         double sum;         int counter = 1;         Scanner scanner = new scanner (System.in);         do {             no = scanner.nextDouble();             sum += no ;             counter = counter + 1;         } while ( counter &lt;= 5)         System.out.println (" The total is " + sum) ;     } }</pre>	6 7 9 3 6 The total is 31.0
<pre>public class contoh45 {     public static void main (String [] args) {         string input ;         Scanner scanner = new scanner (System.in);         int no1;         int no2;         do {             no1 = scanner.nextInt();             no2 = scanner.nextInt();             System.out.println (" Beza = " + Math.abs (no1 -no2)) ;             System.out.println ("Taip YA untuk teruskan");             input = scanner.next();         } while ( input.equal("YA"));     } }</pre>	5 7 Beza = 2 Taip YA untuk teruskan YA 5 9 Taip YA untuk teruskan t

## 1.4

### 1.4.4

MENULIS ATUR CARA MENGGUNAKAN  
STRUKTUR KAWALAN ULANGAN YANG  
MELIBATKAN - **OPERATOR INCREMENT**  
**DAN DECREMENT**

## OPERATOR **INCREMENT** (++) DAN **DECREMENT** (--)

- Lazimnya digunakan sebagai pembilang.
- **Operator Increment (++)** : penambahan nilai boleh ubah bagi bilangan nombor tertentu.
- **Operator Decrement (--)** : mengurangkan nilai boleh ubah bagi bilangan nombor tertentu.

### OPERATOR INCREMENT (++)

UNGKAPAN	MAKNA	CONTOH
i += 1	i = i + 1	while ( <syarat Boolean> ) { <Blok kenyataan berulang> <Kemaskini nilai dalam syarat> }
i += 2	i = i + 2	
i += 3	i = i + 3	

### OPERATOR DECREMENT (--)

UNGKAPAN	MAKNA	CONTOH
i -= 1	i = i - 1	while ( <syarat Boolean> ) { <Blok kenyataan berulang> <Kemaskini nilai dalam syarat> }
i -= 2	i = i - 2	
i -= 3	i = i - 3	

# Math.random [ ]

- Math.random [ ] ialah subaturcara java untuk menjana nombor secara rambang.
- Menggunakan waktu sistem sebagai nilai benih untuk memulakan penjanaan nombor secara rambang.
- CONTOH :  

$$(\text{int}) (\text{Math.random()} * 10) + 1 \quad (\text{nombor rambang } 1 \text{ hingga } 10)$$

## MENJANA 20 NOMBOR RAMBANG BAGI DADU

### CONTOH ATURCARA

```
public class contoh46 {
    public static void main (String [] args) {
        int i ;
        Scanner scanner = new Scanner (System.in);
        Boolean flag = true;
        do {
            for (i = 1; i <=20 ; i++ ) {
                System.out.println ((int) (Math.random() * 6) + 1) + " ";
            }
            System.out.println () ;
            System.out.println (" -----");
            System.out.println (" Taip ya untuk teruskan, tidak untuk henti");
            if (Scanner.next().equals("ya")) {
                flag = true;
            } else {
                flag = false;
            }
        } while ( flag)
    }
}
```

# BENDERA BOOLEAN

- Satu pemboleh ubah Boolean digunakan untuk mengawal ulangan.
- Pengguna ditanya untuk meneruskan atur cara itu lagi atau tidak.

## MENJANA 20 NOMBOR RAMBANG BAGI DADU

### CONTOH ATURCARA

```
public class contoh47 {
    public static void main (String [] args) {
        int i;
        int diceNo;
        int count1 =0, count2 =0, count3 =0, count4 =0, count5 =0, count6 =0;
        Scanner scanner = new Scanner (System.in);
        Boolean flag = true;
        do {
            for (i = 1; i <=20 ; i++) {
                diceNo = (int) (Math.random() * 6 + 1);
                System.out.println ( diceNo + " " );
                Switch (diceNo) {
                    case 1 : count1++; break;
                    case 2 : count2++; break;
                    case 3 : count3++; break;
                    case 4 : count4++; break;
                    case 5 : count5++; break;
                    case 6 : count6++; ;
                }
                System.out.println ();
                System.out.println (" * Dice Number 1 = " + count1 + "%");
                System.out.println (" * Dice Number 2 = " + count2 + "%");
                System.out.println (" * Dice Number 3 = " + count3 + "%");
                System.out.println (" * Dice Number 4 = " + count4 + "%");
                System.out.println (" * Dice Number 5 = " + count5 + "%");
                System.out.println (" * Dice Number 6 = " + count6 + "%");
                System.out.println ();
                System.out.println ("-----");
                System.out.println (" Taip ya untuk teruskan, tidak untuk henti");
                if (Scanner.next().equals("ya")) {
                    flag = true;
                else {
                    flag = false;
                }
            } while ( flag)
        }
    }
}
```

# PEMBILANG

- Digunakan untuk membuat pengiraan dalam penyelesaian masalah.

## KEKERAPAN NOMBOR DADU DALAM 100 LAMBUNGAN

### CONTOH ATURCARA

```

import java.util.Scanner;
public class dadurambang {
    public static void main (String [] args) {
        int i;
        int diceNo;
        int Kira1 =0, Kira2 =0, Kira3 =0, Kira4 =0, Kira5 =0, Kira6 =0;
        double percen1 = 0 , percen2 = 0 , percen3 = 0 , percen4 = 0 , percen5 = 0 , percen6 = 0;
        int jumlahKiraan = 0;
        Scanner scanner = new Scanner (System.in);
        Boolean flag = true;
        do {
            for (i = 1; i <=20 ; i++ ) {
                diceNo = (int) (Math.random() * 6 + 1);
                switch (diceNo) {
                    case 1 : Kira1++ ; break;
                    case 2 : Kira2++ ; break;
                    case 3 : Kira3++ ; break;
                    case 4 : Kira4++ ; break;
                    case 5 : Kira5++ ; break;
                    case 6 : Kira6++ ;
                }
            }
            jumlahKiraan = Kira1 + Kira2 + Kira3 + Kira4 + Kira5 + Kira6;

            percen1 = (double) Kira1/jumlahKiraan * 100;
            percen2 = (double) Kira2/jumlahKiraan * 100;
            percen3 = (double) Kira3/jumlahKiraan * 100;
            percen4 = (double) Kira4/jumlahKiraan * 100;
            percen5 = (double) Kira5/jumlahKiraan * 100;
            percen6 = (double) Kira6/jumlahKiraan * 100;

            System.out.println () ;
            System.out.println (" Nombor 1 dadu = " + percen1 + " %");
            System.out.println (" Nombor 2 dadu = " + percen2 + " %");
            System.out.println (" Nombor 3 dadu = " + percen3 + " %");
            System.out.println (" Nombor 4 dadu = " + percen4 + " %");
            System.out.println (" Nombor 5 dadu = " + percen5 + " %");
            System.out.println (" Nombor 6 dadu = " + percen6 + " %");

            System.out.println () ;
            System.out.println ("-----");
            System.out.println (" Taip ya untuk teruskan");
            if (!scanner.next().equals("ya")) {
                flag = true;
            }
        } while ( flag )
    }
}

```

## 1.5

# AMALAN TERBAIK PENGATURCARAAN

- **AMALAN TERBAIK** : Teknik atau methodologi yang telah dibuktikan melalui suatu pengalaman atau kajian yang boleh dipercayai, untuk mendapatkan hasil yang diinginkan.
- Satu komitmen untuk menggunakan semua pengetahuan dan teknologi yang ada untuk memastikan keberhasilan yang baik.
- **AMALAN TERBAIK PENGATURCARAAN** : Apabila pengaturcara dapat mempraktikkan amalan-amalan yang biasa diikuti untuk menghasilkan atur cara yang baik.

## INDEN YANG KONSISTEN

- Inden yang konsisten membuatkan kod atur cara mudah dibaca dan difahami oleh pengguna lain.
- Cara menulis inden konsisten dari mula hingga akhir kod.

## JENIS DATA

- Pilih jenis data yang bersesuaian supaya saiz pemboleh ubah tidak terlampau kecil atau besar dan memulihara sumber.

## PEMBOLEH UBAH YANG BERMAKNA

- Tidak bermula dengan nombor. CTH : 1cara
- Tiada ruang kosong antara perkataan. Gunakan underscore atau rapatkan perkataan. CTH : cara\_1 / cara1.
- Tidak sama dengan kata kekunci. CTH : integer, double.
- Penggunaan huruf besar dan huruf kecil. CTH : caraPertama tidak sama dengan CaraPertama.
- Nama yang bermakna dan mudah difahami. Penggunaan singkatan tidak digalakkan.
- Tidak menggunakan perkataan rezab khas. CTH : print,value.

## KOMEN

- Ditulis dengan jelas dalam dua hingga tiga baris pendek untuk menerangkan fungsi kod dan memenuhi ruang lajur pengekodan.

## JENIS RALAT ALGORITMA

RALAT SINTAKS

RALAT LOGIK

RALAT MASA LARIAN

RALAT SINTAKS

- Kesalahan tatabahasa seperti salah ejaan atau tata tanda.
- Penggunaan objek atau aksara yang tidak dikenali.

RALAT LOGIK

- Berlaku apabila atur cara tidak berfungsi seperti yang diingini.
- Jarang atau tidak dapat dikesan oleh pengkompi.
- Hanya pengaturcara yang boleh mengesan melalui output yang dihasilkan.
- Pengatur cara perlu memeriksa semua aspek output projek.

RALAT MASA LARIAN

- Ralat yang ditemui ketika aturcara yang sedang berjalan terganggu akibat beberapa faktor.
- Berlaku sekiranya pengatur cara cuba melaksanakan operasi aritmetik yang mustahil.

**1.5**

**1.5.2**

## **MENGESAN, MENGENALPASTI, MENTERJEMAH MESEJ RALAT DAN MEMBAIKI RALAT**

Semak semula atur cara pada bahagian pengisytiharan

Pastikan semua tatahanda ditaip dengan lengkap

Pastikan nama boleh ubah yang diisyiharkan adalah sama dengan nama yang akan dipanggil balik dalam atur cara (semak ejaan & penggunaan huruf besar/kecil)

Baiki ralat yang dikenalpasti

**1.5**

**1.5.3**

## **MENGENALPASTI NILAI BAGI PEMBOLEH UBAH PADA BAHAGIAN TERTENTU ATUR CARA**

<b>PEMBOLEH UBAH</b>	<b>INPUT</b>	<b>OUTPUT</b>
Item Pemboleh Ubah		
Nilai (Data Pemboleh Ubah)		

**1.5****1.5.4**

## MENGHASILKAN ATUR CARA YANG MUDAH DIBACA DENGAN MENGGUNAKAN GAYA YANG BAIK



KOMEN	<ul style="list-style-type: none"><li>Penanda yang dibuat oleh pengatur cara untuk setiap atur cara yang di bina.</li><li>Dalam Java , kod komen perlu mengikut sintaks yang ditetapkan untuk mengelak ralat sintaks.</li></ul> <table border="1"><thead><tr><th>JENIS KOMEN</th><th>HURAIAN</th></tr></thead><tbody><tr><td>//</td><td>Pengkompil mengabaikan semua teks bermula dengan // hingga teks terakhir ayat yang sama.</td></tr><tr><td>/* */</td><td>Pengkompil mengabaikan semua teks yang berada dalam /* hingga ke */ walaupun pada baris berlainan.</td></tr><tr><td>/** */</td><td>Komen dokumentasi. Pengkompil mengabaikan komen ini sama seperti komen /*.</td></tr></tbody></table>	JENIS KOMEN	HURAIAN	//	Pengkompil mengabaikan semua teks bermula dengan // hingga teks terakhir ayat yang sama.	/* */	Pengkompil mengabaikan semua teks yang berada dalam /* hingga ke */ walaupun pada baris berlainan.	/** */	Komen dokumentasi. Pengkompil mengabaikan komen ini sama seperti komen /*.
JENIS KOMEN	HURAIAN								
//	Pengkompil mengabaikan semua teks bermula dengan // hingga teks terakhir ayat yang sama.								
/* */	Pengkompil mengabaikan semua teks yang berada dalam /* hingga ke */ walaupun pada baris berlainan.								
/** */	Komen dokumentasi. Pengkompil mengabaikan komen ini sama seperti komen /*.								
PEMBOLEH UBAH YANG BERMAKNA	<ul style="list-style-type: none"><li>Nama pemboleh ubah yang mempunyai ejaan yang ringkas dan bermakna.</li></ul>								
INDEN	<ul style="list-style-type: none"><li>Merujuk kepada cara penulisan atur cara yang memudahkan pembacaan.</li><li>Pembacaan atur cara akan dimulakan dengan inden iaitu barisan teks berada di beberapa kedudukan aksara ke dalam, dari jidar kiri atau kanan halaman.</li></ul>								

## CONTOH

```

/* Program ringkas untuk mengira Luas Segitiga
Langkah 1: Baca Tapak
Langkah 2: Baca Tinggi
Langkah 3: Hitung Luas
Langkah 4: Papar Luas
*/
public class LuasSegitiga {
    public static void main (String[] args){
        /**Pengisytiharan pemboleh ubah input dan output*/
        int Tapak = 6;
        int Tinggi = 4;
        double Luas;

        // Proses yang terlibat dalam penghitungan
        // luas segitiga
        Luas = (1.0 / 2) * Tapak * Tinggi;

        //Paparan output
        System.out.println ("Luas Segitiga ialah : " + Luas);
    }
}

```

Komen

Komen

Komen

Inden

```

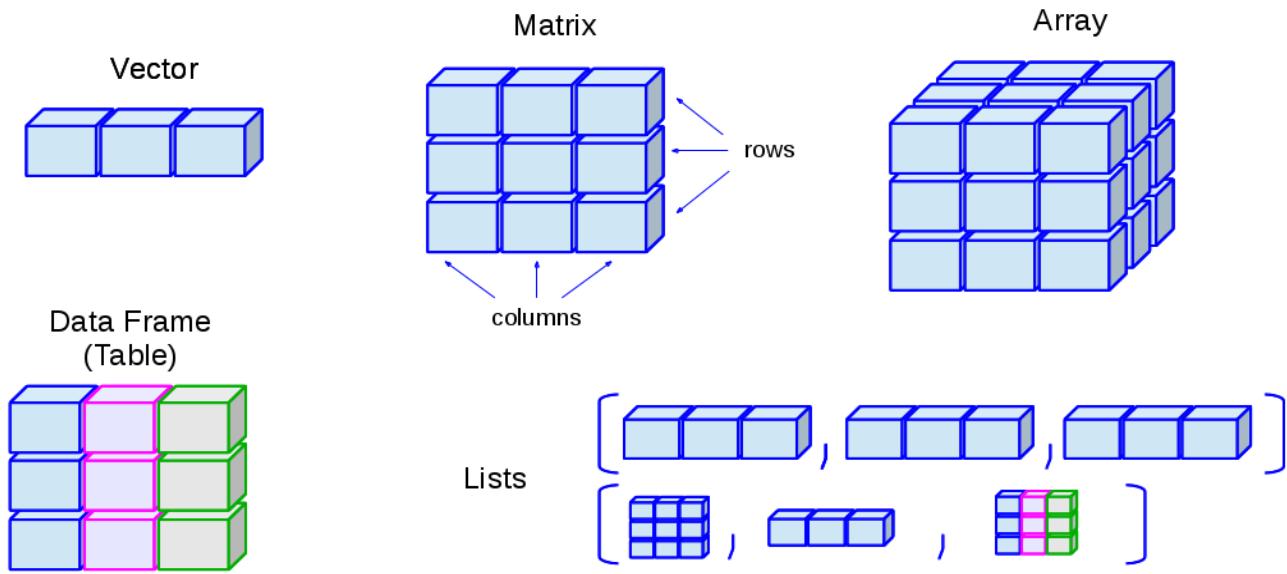
/* Program menghitung luas segitiga*/
public class LuasSegitiga {
    public static void main (String[] args){
        /**Pengisytiharan pemboleh ubah input dan output*/
        int Tapak = 6;
        int Tinggi = 4;
        double Luas;

        // Proses yang terlibat dalam penghitungan
        // luas segitiga
        Luas = (1.0 / 2) * Tapak * Tinggi;

        //Paparan output
        System.out.println ("Luas Segitiga ialah : "
+ Luas);
    }
}

```

Pemboleh ubah yang bermakna



- **STRUKTUR DATA** : Satu kaedah tertentu untuk menyimpan secara tersusun data-data dalam ingatan supaya senang dicapai untuk diproses menjadi maklumat mengikut kehendak pengguna.
- Data boleh disusun dalam bentuk **tatasusunan (Array)** dan **vector (vector)**, **senarai pautan (linked list)**, **timbunan (stack)** dan **giliran (queue)**.
- Gunakan struktur yang sistematis untuk pemboleh ubah dan arahan semasa membangunkan atur cara.
- Pemboleh ubah boleh dipecahkan kepada “*kumpulan-kumpulan kecil*” yang dipanggil **TATASUSUNAN**.
- Arahan-arahan komputer juga boleh dipecahkan kepada *kumpulan-kumpulan kecil* yang dipanggil **FUNGSI**.
- Apabila diperlukan sahaja, kumpulan yang berkaitan akan dipanggil.

# TATASUSUNAN

- **TATASUSUNAN** ialah pemboleh ubah yang membolehkan koleksi beberapa nilai data (elemen) dalam satu-satu masa dengan menyimpan setiap elemen dalam ruang memori berindeks.
- Pemboleh ubah ialah slot memori yang telah dikhaskan untuk menyimpan data.
- Kebiasaanya, pemboleh ubah mudah cuma menyimpan satu nilai data dalam satu-satu masa.

## Pengisytiharan Tatasusunan

```
jenisData [ ] namaTatasusunan;  
namaTatasusunan = new jenisData [ saizTatasusunan];
```

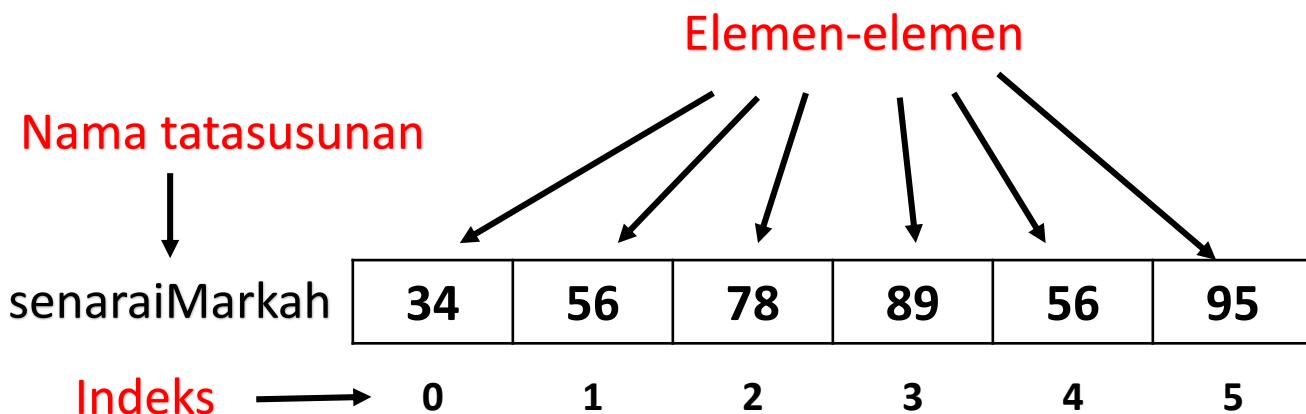
**Contoh :**    int [ ] senaraiMarkah ;  
              senaraiMarkah = new int [ 6 ];

## Pengumpukan Tatasusunan

- Pengisytiharan tatasusunan menyediakan ruang memori yang masih kosong.
- Nilai perlu diberikan melalui proses pengumpukan.
- Selepas diisytihar, nilai diumpuk dengan memanggil elemen –elemen tatasusunan satu-persatu.

**Contoh :**

```
senaraiMarkah [ 0 ] = 34 ;
senaraiMarkah [ 1 ] = 56 ;
senaraiMarkah [ 2 ] = 78 ;
senaraiMarkah [ 3 ] = 89 ;
senaraiMarkah [ 4 ] = 56 ;
senaraiMarkah [ 5 ] = 95 ;
```



## Pengumpukan Nilai Awal Tatasusunan

### Contoh :

```
int senaraiMarkah [ ] = { 34,56,78,89,56,95};
```

- Umpukan dibuat ketika melakukan pengisytiharan.
- Saiz dalam tatasusunan tidak perlu dimasukkan dalam tanda [ ].
- Saiz tatasusunan ditentukan secara automatik berdasarkan bilangan data dalam kurungan { }.
- Semua data yang hendak disimpan ialah satu jenis yang sama.

**Perbezaan Struktur Memori antara  
Pemboleh Ubah Mudah dengan Memori Tatasusunan.**

### Pemboleh Ubah Mudah

```
int markah1 = 56, markah2 = 78, markah3 = 34;
```

### Tatasusunan

```
int markah [ ] = { 56,78,34};
```

# 1.6

## 1.6.1

### MENERANGKAN STRUKTUR TATASUSUNAN (ARRAY) SATU DIMENSI

## ATURCARA YANG MENGGUNAKAN TATASUSUNAN

### CONTOH ATURCARA

```
package perisianSaya;
public class MyClass {
    public static void main (String [] args) {

        String [] senaraiNama = new String[4];

        senaraiNama [0] = "Adam";
        senaraiNama [1] = "Alia";
        senaraiNama [2] = "Wong";
        senaraiNama [3] = "Devi";

        int [ ] senaraiUmur = {16,17,16,17};

        double [] senaraiTinggi = {182.1,172.5,173.2,175.0}

        System.out.println (" NAMA\tUmur\tTinggi(cm)");

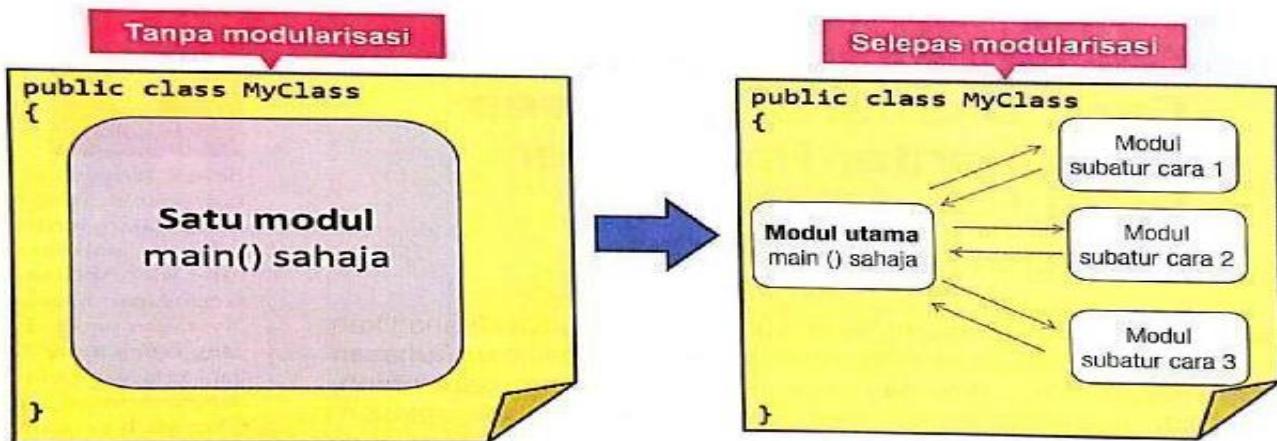
        for (int i = 0; i<4;i++) {
            System.out.print (senaraiNama [i] + "\t");
            System.out.print (senaraiUmur [i] + "\t");
            System.out.print (senaraiTinggi [i] + "\t");

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

# 1.6

## 1.6.2

### MENGGUNAKAN SUBATUR CARA DAN MEMAHAMI KONSEP MENGHANTAR PARAMETER KE SUBATUR CARA DAN MENGBALIKAN DATA

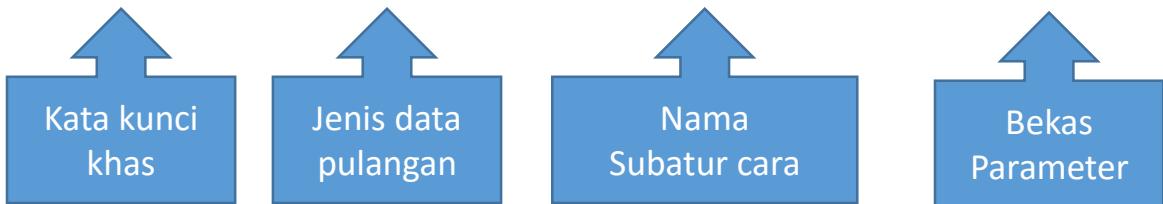


- Fail kod komputer yang panjang adalah sukar ditulis, dibaca, diulangkaji atau dikemaskini.
- Oleh itu, baris-baris kod komputer yang berkait boleh dihimpunkan dalam satu **modul**.
- Dengan itu, kod komputer yang panjang dapat dibahagi-bahagikan kepada modul-modul.
- Setiap modul adalah lebih pendek dan mengkhususkan kepada tujuan tertentu.
- Modul-modul ini dipanggil **subatur cara**, struktur untuk himpunan kod komputer.



## KOMPONEN *HEADER* SUBATUR CARA

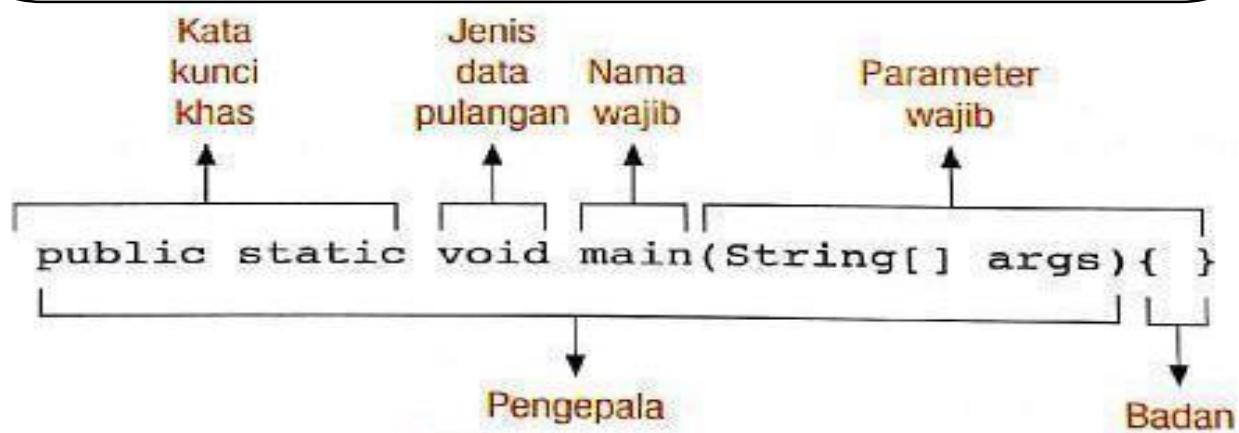
`static void subAtur01 ()`



<b>Kata Kunci Khas</b>	<ul style="list-style-type: none"> <li>Kata kunci <i>static</i> diletakkan dihadapan nama subatur cara.</li> <li>Tanpa kata kunci ini, subatur cara tidak dapat digunakan secara langsung.</li> <li>Tanpa static, subatur cara memerlukan penghasilan objek sebelum data digunakan.</li> </ul>
<b>Jenis Data Pulangan</b>	<ul style="list-style-type: none"> <li>Subatur cara biasanya akan memulangkan hasil setelah badan subatur cara selesai memproses data.</li> <li>Jenis data pulangan di <i>header</i> bergantung kepada jenis data yang ingin dipulangkan oleh <i>body</i>.</li> <li>Ini termasuk <i>int, double, string</i> dan <i>char</i>.</li> <li>Jika tidak ada keperluan memulangkan data, gunakan kata kunci <b>void</b>.</li> </ul>
<b>Nama Subatur Cara</b>	<ul style="list-style-type: none"> <li>Diberikan oleh pengatur cara.</li> <li>Mestilah bermula dengan huruf (biasanya huruf kecil) .</li> <li>Boleh mengandungi angka tetapi bukan symbol.</li> </ul>
<b>Bekas Parameter</b>	<ul style="list-style-type: none"> <li>Simbol () digunakan jika parameter kosong.</li> <li>Nama parameter akan dikepulkan jika bekas menerima parameter.</li> <li>CONTOH : (int kuantiti)</li> </ul>

# SUBATUR CARA **main ( )**

- Subatur cara wajib dengan nama **main ()**.
- boleh wujud tanpa subatur cara yang lain.
- mengandungi baris pertama pernyataan yang mesti dilaksanakan oleh komputer.
- mengandungi baris terakhir pernyataan yang mesti dilaksanakan oleh komputer.
- *Header* subatur cara jarang diubah.
- Pernyataan-pernyataan dalam subatur cara **main ()** akan menentukan sifat atur cara.
- Pernyataan-pernyataan ini seharusnya ditulis berdasarkan algoritma yang telah diuji.



<b>Kata Kunci Khas</b>	<ul style="list-style-type: none"> <li>• Public membolehkan subatur cara diakses dari mana-mana kod projek</li> <li>• Semua subatur cara mempunyai static supaya dapat digunakan secara langsung tanpa objek..</li> </ul>
<b>Jenis Data Pulangan</b>	<ul style="list-style-type: none"> <li>• Kata kunci <b>void</b> digunakan kerana tidak memulangkan data.</li> </ul>
<b>Nama Subatur Cara</b>	<ul style="list-style-type: none"> <li>• Nama wajib ialah <b>main</b>.</li> </ul>
<b>Bekas Parameter</b>	<ul style="list-style-type: none"> <li>• Bekas parameter mesti mengandungi parameter tatasusunan string dengan nama “args”.</li> </ul>

# SUBATUR CARA main()

```

1 package perisianSaya;
2 public class MyClass {
3
4     public static void main(String[] args){
5         System.out.println("Hello Malaysia.");
6     }
7
8 }
9
10

```

Subatur cara main()

C:\WINDOWS\system32\cmd.exe

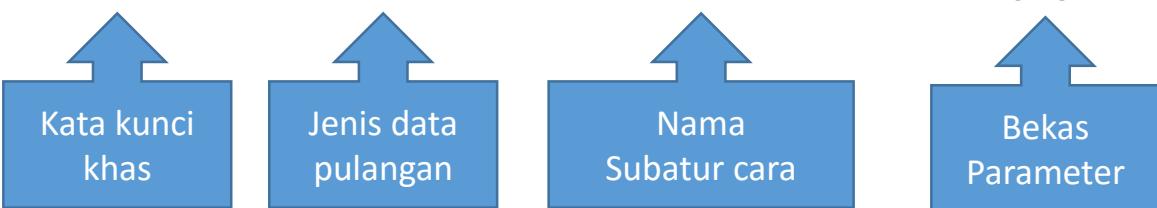
D:\Java Projects\HelloMalaysia\dist>java -jar HelloMalaysia.jar

Hello Malaysia.

Output

# SUBATUR CARA LAIN

## static void subAtur01()



- Pengatur cara boleh menulis subatur cara yang lain – dipanggil **petakrifan subatur cara**.
- Subatur cara lain adalah serupa dengan **main()** tetapi lebih ringkas.

## MEMANGGIL SUBATUR CARA main ( )

- Subatur cara boleh menggunakan subatur cara lain.
- Tujuannya supaya kod pernyataan dalam subatur cara lain turut dilaksanakan.
- Hubungan dua subatur cara – *pemanggil* dan *dipanggil*.
- *Pemanggil* memanggil nama *subatur cara dipanggil* dalam badan subatur cara badan pemanggil.

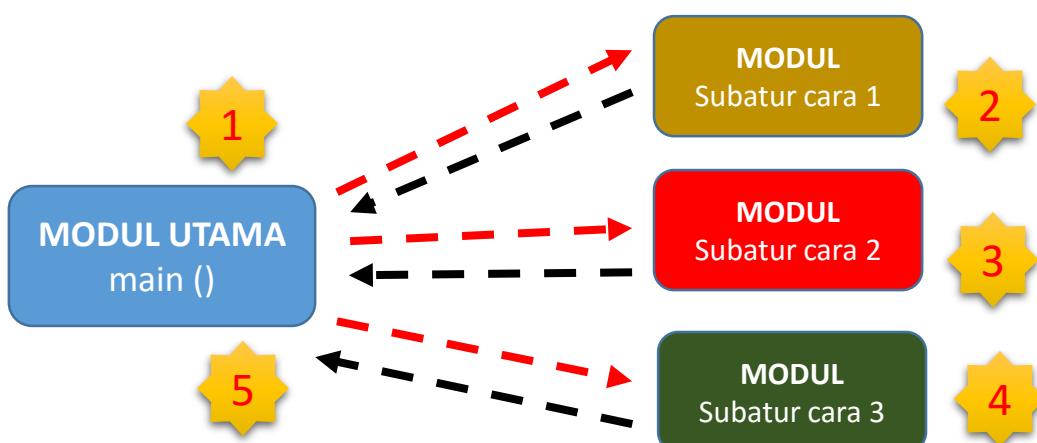
Subatur cara PEMANGGIL

panggil

Subatur cara DIPANGGIL

## MEMANGGIL SUBATUR CARA DARIPADA main ( )

- Kebiasaannya, subatur cara main () menggunakan subatur cara-subatur cara yang lain.
- Apabila main() memerlukan bantuan subatur cara lain untuk proses tertentu , kawalan dipindahkan kepada subatur cara tersebut.
- Setelah subatur cara tersebut selesai, kawalan dikembalikan kepada subatur cara main () .



```

1 package perisiansaya;
2 public class MyClass {
3
4     public static void main(String[] args){
5         hello(); // 1. Panggil
6     }
7
8     static void hello() { // 2. Laksana
9         System.out.println("Hello dunia.");
10    }
11 }
12
13 }
```

Diagram illustrating the execution flow of the Java code:

- 1. Panggil**: The call to the `hello()` method from the `main()` method.
- 2. Laksana**: The definition of the `hello()` method.
- 3. Pulang**: The return point to the `main()` method after the `hello()` method has been executed.

## CONTOH ATURCARA

```

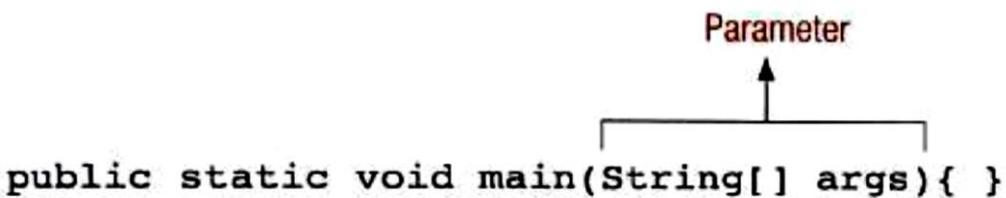
public class contohpg148 {
    static java.util.Scanner
    scanner = new java.util.Scanner (System.in);

    public static void main (String [] args) {
        mintaNama ();
        mintaMarkah();
    }

    public static void mintaNama(){
        String nama;
        System.out.println ("Masukkan Nama : ");
        nama = scanner.nextLine();
        System.out.println ("Terima Kasih" + nama);
    }

    public static void mintaMarkah() {
        int markah;
        System.out.println ("Masukkan Markah : ");
        markah = scanner.nextInt();
        System.out.println ("Markah Anda : " + markah);
    }
}
```

# PARAMETER



- Parameter ataupun argumen ialah pemboleh ubah yang membolehkan subatur cara menerima nilai daripada pemanggil.
- Dengan ini, subatur cara- subatur cara masih dapat berkongsi nilai-nilai pemboleh ubah melalui parameter.
- Parameter rasmi (formal parameter) : merujuk kepada parameter subatur cara.
- Parameter sebenar (actual parameter) : merujuk kepada pemboleh ubah di dalam subatur cara pemanggil.
- Penggunaan parameter perlu diisyiharkan sewaktu pentakrifan subatur cara- subatur cara.
- Jika parameter diperlukan, parameter perlu diisyiharkan dalam kurungan bekas parameter dalam subatur cara.
- Pengisyiharan parameter sama seperti pengisyiharan pemboleh ubah.
- Tiada had untuk bilangan parameter dan turutan parameter bergantung kepada pengatur cara.

TIADA PARAMETER	MENGANDUNG PARAMETER
static void subAtur01 ( ) { }	static void subAtur01 (int x ) { }
static void subAtur02 ( ) { }	static void subAtur02 ( int x ; double y ) { }
static void subAtur03 ( ) { }	static void subAtur03 (int [ ] x ; string z) { }

**1.6**

**1.6.2**

**MENGGUNAKAN SUBATUR CARA DAN MEMAHAMI  
KONSEP MENGHANTAR PARAMETER KE SUBATUR  
CARA DAN MENGEMBALIKAN DATA**

## **CONTOH ATURCARA**

```
static void kuasaDua (int nom) {  
    double jawapan = nom*nom;  
    System.out.print (jawapan);  
}
```

```
static void cariJumlah (int x, int y, int z) {  
    int jawapan = x + y + z;  
    System.out.print (jawapan);  
}
```

```
public static void main (String [ ] args) {  
    cariJumlah (1,2,7);  
}
```

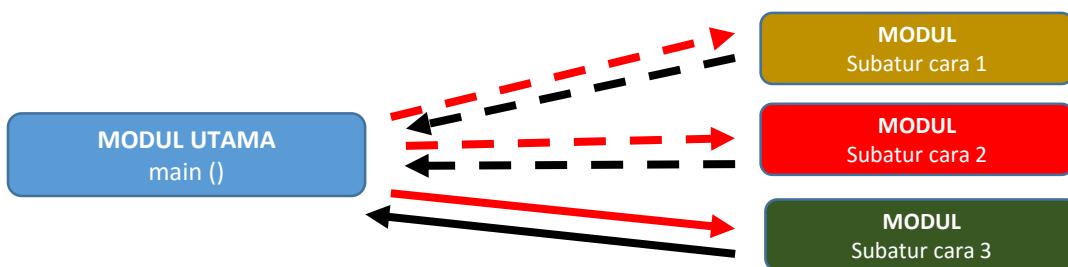
```
static void cariJumlah (int x, int y, int z) {  
    int jawapan = x + y + z;  
    System.out.print (jawapan);  
}
```

```
public static void main (String [ ] args) {  
    paparHarga ("telefon pintar", 1500.0);  
}
```

```
static void paparHarga (string item, double h) {  
    System.out.print (" Harga " + item " adalah " +h);  
}
```

# MENGEMBALIKAN DATA

- Semua subatur cara akan mengembalikan kawalan kepada pemanggil.
- Sesetengah subatur cara bukan sahaja mengembalikan kawalan tetapi juga data.
- Bagi yang memulangkan kawalan sahaja, kata kunci **void** digunakan.
- Sekiranya subatur cara mengembalikan data, baris akhir dalam badan subatur cara mempunyai pernyataan **return** dan data yang ingin dipulangkan kepada pemanggil.
- Nama subatur cara tidak mempunyai void sebagai jenis data pulangan. Sebaliknya, jenis data yang bersesuaian dengan data dipulang diisythiar dalam sintaks.
- Data boleh dipulangkan sebagai hasil ungkapan, nilai dalam boleh ubah, nilai pemalar ataupun nilai data itu sendiri.
- Setiap subatur cara cuma boleh mengembalikan satu jenis data sahaja.
- Nama subatur cara yang memulangkan data biasanya diberikan *prefix get*. Contoh : `getNama`, `getAlamat`, `getTelefon`.



DEFINISI SUBATUR CARA	PENJELASAN
<code>static void subAturcara () {}</code>	Tidak memulangkan data. Cuma kawalan dipulangkan.
<code>static int subAturcara () {}</code>	Memulangkan data jenis integer.
<code>static double subAturcara () {}</code>	Memulangkan data jenis double.
<code>static string subAturcara () {}</code>	Memulangkan data jenis string.

## CONTOH ATURCARA

```
public class contohpg153 {  
    static java.util.Scanner scanner = new java.util.Scanner (System.in);  
  
    public static void main (String[] args){  
        String nama, alamat, telefon;  
  
        nama = getNama();  
        alamat = getAlamat(nama);  
        telefon = getTelefon(nama);  
  
        System.out.println();  
        System.out.println(" Terima Kasih, Sila semak , ");  
        System.out.println(" Nama : " + nama);  
        System.out.println(" Alamat : " + alamat);  
        System.out.println(" Telefon : " + telefon);  
    }  
  
    static String getNama (){  
        System.out.print(" Masukkan nama : ");  
        return scanner.nextLine();  
    }  
  
    static String getAlamat (String n){  
        System.out.print(n + " , Sila Masukkan alamat : ");  
        return scanner.nextLine();  
    }  
  
    static String getTelefon (String n){  
        System.out.print(n + " , Sila Masukkan telefon : ");  
        return scanner.nextLine();  
    }  
}
```

# 1.6

## 1.6.3

### MEMBEZAKAN ANTARA **FUNCTION** DAN **PROCEDURE** PADA SUBATUR CARA

#### FUNGSI

- Data yang telah diproses perlu disimpan dalam badan pemanggil.
- Data yang telah diproses akan digunakan lagi oleh pemanggil.
- Data tersebut mungkin digunakan oleh subatur cara main () atau subatur cara-subatur cara lain.
- Subatur cara digunakan untuk meminta input.
- Subatur cara tidak memaparkan hasil atau menyimpan hasil di mana-mana.

#### PROSEDUR

- Subatur cara digunakan untuk membuat paparan sahaja, seperti mesej kepada pengguna.
- Hasil proses digunakan dalam subatur cara sekali sahaja dan tidak diperlukan lagi.

ASPEK PERBANDINGAN	FUNGSI	PROSEDUR
Persamaan	Mengembalikan kawalan	Mengembalikan kawalan
Perbezaan	<ul style="list-style-type: none"><li>• Mengembalikan data.</li><li>• Badan diakhiri dengan pernyataan <b>return</b> diikuti data yang dipulangkan.</li></ul>	<ul style="list-style-type: none"><li>• Tidak Mengembalikan data.</li><li>• Badan <b>tidak diakhiri</b> dengan pernyataan <b>return</b>.</li></ul>
Jenis data pulangan	int, double , char,string, tatasusunan atau objek java.	void
Sintaks definisi	static jenisData namaFungsi ([jenisData namaParameter]){};	static void namaProsedur ([jenisData namaParameter]){};
Contoh definisi	static int cariJumlah (int x, int y){ int jawapan = x + y; return jawapan; }	static void cariJumlah (int x, int y) { int jawapan = x + y; System.out.print (jawapan); }
Sintaks panggilan	jenisData pembolehUbah; pembolehUbah = namaFungsi ([jenisData namaParameter ]);;	namaProsedur ([jenisData namaParameter ] );
Contoh panggilan	int jumlah = cariJumlah (5,8);	cariJumlah (5,8);

## CONTOH ATURCARA

FUNGSI	PROSEDUR
<pre>static int mintaNombor () {     int nom;     java.util.Scanner sc;     sc = new Java.util.Scanner (System,.in);     nom = sc.nextInt();     Return nom }</pre>	<pre>static void hello () {     System.out.print ("Hello dunia."); }</pre>
<pre>int jumlahNombor ( int x, int y ) {     Int jumlah;     Jumlah = x + y;     Return jumlah; }</pre>	<pre>static void hello (String nama) {     System.out.print ("Hello " + nama ); }</pre>
	<pre>static void cariJumlah (int x, int y ) {     Int jawapan = x + y;     System.out.print (jawapan); }</pre>

**CONTOH ATURCARA**

```
public class contoh61 {  
    static java.util.Scanner scanner = new java.util.Scanner (System.in);  
  
    public static void main (String [] args){  
        String x;  
        x = getNama ();  
        System.out.println ("Salam sejahtera, " + x);  
    }  
  
    static String getNama () {  
        System.out.print ("Masukkan nama : ");  
        return scanner.nextLine();  
    }  
  
public class contoh62{  
    static java.util.Scanner sc = new java.util.Scanner (System.in);  
  
    public static void main (String []args){  
        double nom1,nom2;  
        System.out.println("Masukkan nombor 1: ");  
        nom1 = sc.nextInt();  
        System.out.println("Masukkan nombor 2: ");  
        nom2 = sc.nextInt();  
        System.out.println (Math.max(nom1,nom2) + " adalah lebih besar");  
    }  
}
```

# CONTOH FUNGSI-FUNGSI UTILITI DALAM JAVA

SINTAKS FUNGSI	PENJELASAN
Math.sqrt (double n)	<p>Memulangkan hasil punca kuasa dua untuk nilai n.</p> <p><b>CONTOH :</b> Math.sqrt (100);</p>
Math.floor(double n)	<p>Memulangkan interger paling dekat tetapi kurang atau sama dengan nilai n.</p> <p><b>CONTOH :</b> Math.floor (2.7); [ memulangkan 2] Math.floor (-2.7); [ memulangkan -3]</p>
Math.round (double n)	<p>Memulangkan nombor n stelah dibulatkan kepada interger terdekat.</p> <p><b>CONTOH :</b> Math.floor (2.7); [ memulangkan 3] Math.floor (2.3); [ memulangkan 2]</p>
Math.max (double m , double n)	<p>Memulangkan nombor yang lebih besar antara m dan n</p> <p><b>CONTOH :</b> Math.max(100,10); [ memulangkan 100]</p>
Math.min (double m , double n)	<p>Memulangkan nombor yang lebih kecil antara m dan n</p> <p><b>CONTOH :</b> Math.max(100,10); [ memulangkan 10]</p>

# 1.6

## 1.6.4

### MENULIS ATUR CARA BERMODULAR YANG MENGANDUNGI STRUKTUR TATASUSUNAN

```
1 package perisianSaya;
2 public class MyClass {
3
4     static String nama;
5     static int markah;
6     static String gred;
7     static java.util.Scanner scanner = new java.util.Scanner(System.in);
8
9     public static void main(String[] args) {
10
11         getNama(); 1
12         getMarkah(); 2
13         setGred(); 3
14         paparInfo(); 4
15     }
16 }
```

1

```
static void getNama() {
    System.out.print("Masukkan nama:");
    nama = scanner.nextLine();
}
```

2

```
static void getMarkah() {
    System.out.print("Masukkan markah, " + nama + ":");
    markah = scanner.nextInt();
}
```

3

```
static void setGred() {
    if(markah>=90){gred = "A+"}
    else if(markah>=80){gred = "A";}
    else if(markah>=75){gred = "A-";}
    else if(markah>=70){gred = "B+";}
    else if(markah>=65){gred = "B";}
    else if(markah>=60){gred = "C+";}
    else if(markah>=50){gred = "C";}
    else if(markah>=45){gred = "D";}
    else if(markah>=40){gred = "E";}
    else {gred = "F";}
}
```

4

```
static void paparInfo() {
    System.out.print("Terima kasih, " + nama + ", ");
    System.out.println("Gred untuk " + markah + " adalah " + gred);
}
```

```
1 package perisianSaya;
2 public class MyClass {
3
4     static String nama;
5     static int markah;
6     static String gred;
7     static java.util.Scanner scanner = new java.util.Scanner(System.in);
8
9     public static void main(String[] args){
10
11         getNama();
12         getMarkah();
13         setGred();
14         paparInfo();
15     }
16 }
```

- Atur cara bagi contoh diatas ialah atur cara **console**.
- Pemboleh ubah (*nama*, *gred*, *markah*) diisyihar sebagai **Pemboleh ubah sejagat (global)** – boleh dicapai oleh semua subatur cara.
- Prosedur **main()**, **getNama()**,**getMarkah()**, **setGred()** dan **paparInfo()** dapat membaca dan menulis kepada set pemboleh ubah-pe,boleh ubah yang sama.
- **Prosedur main()** – tidak terdapat banyak pernyataan algoritma kerana pernyataa-pernyataan tersebut telah diletakkan ke dalam subatur cara.
- Memanggil subatur cara harus mengikut urutan logik.

## CONTOH ATURCARA

```
public class contohpg159 {  
    static String nama;  
    static int markah;  
    static String gred;  
    static java.util.Scanner scanner = new java.util.Scanner (System.in);  
  
    public static void main (String[]args){  
        getNama();  
        getMarkah();  
        setGred();  
        paparInfo();  
    }  
  
    static void getNama(){  
        System.out.print ("Masukkan nama : ");  
        nama = scanner.nextLine();  
    }  
  
    static void getMarkah() {  
        System.out.print ("Masukkan markah , " + nama + " : " );  
        markah = scanner.nextInt();  
    }  
  
    static void setGred() {  
        if (markah >=90) {gred = "A+";}  
        else if (markah >=80) {gred = "A";}  
        else if (markah >=75) {gred = "A-";}  
        else if (markah >=70) {gred = "B+";}  
        else if (markah >=65) {gred = "B";}  
        else if (markah >=60) {gred = "C+";}  
        else if (markah >=50) {gred = "C";}  
        else if (markah >=45) {gred = "D";}  
        else if (markah >=40) {gred = "E";}  
        else {gred = "F";}  
    }  
  
    static void paparInfo(){  
        System.out.print ("Terima Kasih," +nama + " . ");  
        System.out.print ("Gred untuk " + markah + " adalah " +gred);  
    }  
}
```

# STRUKTUR TATASUSUNAN DALAM ATUR CARA BERMODULAR

## CONTOH ATURCARA

```
public class contoh63 {  
  
    public static void main (String[] args){  
        int[] senaraiNombor = {1,2,3,4,5,6,7,8,9,10};  
        paparSenarai(senaraiNombor);  
    }  
  
    static void paparSenarai (int[] senaraiNombor){  
        System.out.print ("Senarai nonbor dalam subatur cara : ");  
  
        for (int i=0;i<10;i++)  
            {System.out.print (senaraiNombor [i] + " , ");  
        }  
    }  
}
```

- Tatasusunan digunakan sebagai parameter untuk bilangan data yang banyak.
- Pastikan subatur cara mampu menerima parameter tatasusunan.
- Pengisytiharan dilakukan pada kepala subatur cara.
- Struktur kawalan ulangan *for* diperlukan untuk mengumpuk atau mengakses nilai elemen-elemen dalam tatasusunan.
- Elemen-elemen dalam satu-satu tatasusunan boleh diubah terus dari mana-mana subatur cara.
- Oleh itu, pemboleh ubah tatasusunan tidak perlu dikembalikan kepada pemanggil.

# STRUKTUR TATASUSUNAN DALAM ATUR CARA BERMODULAR

## CONTOH ATURCARA

```
public class contoh64 {  
  
    public static void main (String[] args){  
        int[] senaraiNombor = new int[10];  
        setSenaraiRawak (senaraiNombor);  
        System.out.println ("\n\nDalam subatur cara main :");  
        for (int i=0;i<10;i++)  
            {System.out.print (senaraiNombor [i] + " , ");  
         }  
    }  
  
    static void setSenaraiRawak(int[] senaraiNombor){  
        System.out.print ("Dalam subatur cara setSenaraiRawak : ");  
        for (int i=0;i<10;i++){  
            senaraiNombor[i] = (int)(Math.random()*10)+1;  
            {System.out.print (senaraiNombor [i] + " , ");  
         }  
    }  
}
```

- Tatasusunan nombor yang kosong diisyiharkan dalam prosedur **main()**.
- Dari **main()**, panggilan dibuat kepada prosedur **setSenaraiRawak**.
- Dalam prosedur **setSenaraiRawak**, tatasusunan kosong diumpukkan dengan nilai-nilai rawak yang dijanakan oleh **math.random**.
- Nilai-nilai elemen dipaparkan pada kedua-dua subatur cara dan prosedur **main()**.

# PEMBANGUNAN APLIKASI

## KITAR HAYAT PEMBANGUNAN SISTEM Software Development Life Cycle (SDLC)

- Juga dikenali sebagai Kitar Hayat Pembangunan Aplikasi.
- Istilah yang digunakan dalam kejuruteraan sistem dan perisian, sistem maklumat dan pembangunan aplikasi.
- Menjelaskan tentang proses merancang, mereka bentuk, menguji dan mengimplementasi sesuatu aplikasi atau perisian.
- Terdiri daripada satu kitaran fasa berjujukan dan menjadikannya sebagai pelan tindakan yang berkesan kepada pasukan projek.
- SDLC membantu mengesan status bagi penyempurnaan projek tersebut.

### Methodologi Umum SDLC

- **MODEL AIR TERJUN** (Waterfall)
- **MODEL RAD** (Rapid Application Development)
- **MODEL LELARAN** (Iterative)
- **MODEL LINGKARAN** (Spiral)
- **MODEL TANGKAS** (Agile)
- **MODEL HIBRID** (Kombinasi Model)

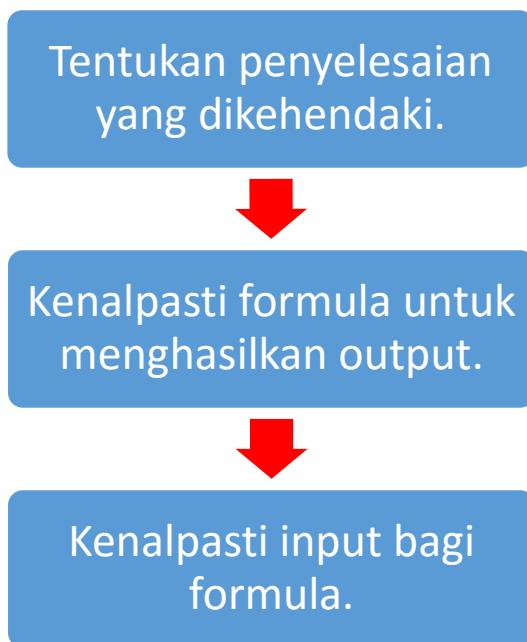


## MODEL AIR TERJUN

- Digunakan sebagai model pembangunan aplikasi memandangkan model ini mirip kepada proses-proses dalam SDLC.
- Merupakan model yang terawal, mudah difahami dan mudah diuruskan.
- Mengandungi 5 fasa.
- Satu fasa perlu diselesaikan sebelum ke fasa seterusnya.
- Maklumat bagi setiap fasa diperlukan untuk fasa yang berikutnya dan tidak boleh berpatah balik.

## Fasa Analisis Masalah

- Proses mengenal pasti keperluan program dan mencari sebab sesuatu program dibina.
- Langkah-langkah sistematik harus dipatuhi untuk menyelesaikan masalah dan penyting untuk kita memahami pernyataan masalah dengan jelas.
- Menggunakan analisis/carta IPO.



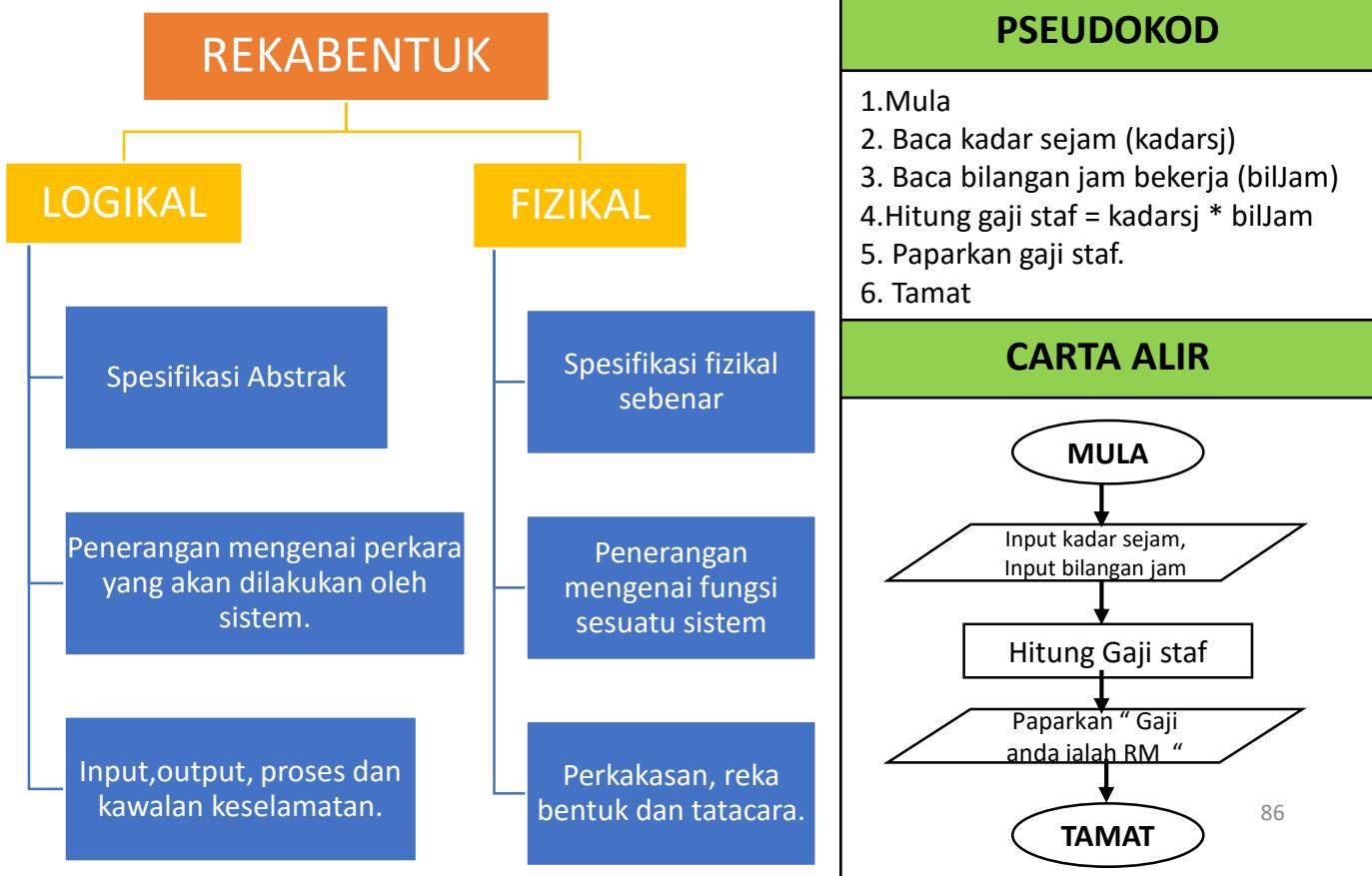
### Contoh CARTA IPO

<b>INPUT</b>	Kadar sejam , Bilangan Jam bekerja
<b>PROSES</b>	<ol style="list-style-type: none"> <li>1. Dapatkan kadar sejam</li> <li>2. Dapatkan bilangan jam bekerja.</li> <li>3. Kirakan Gaji Staf = kadar sejam X bilangan jam bekerja</li> </ol>
<b>OUTPUT</b>	Gaji Staf

# Fasa Reka bentuk Penyelesaian

- Dibuat setelah analisis IPO.
- Fasa ini melihat kepada potensi penyelesaian yang wujud dan menentukan penyelesaian yang efektif dan efisien.
- Membina penyelesaian terbaik.
- **ALGORITMA** : Langkah awal penyelesaian masalah.
- **PSEUDOKOD** : Aturan langkah yang ditulis dalam Bahasa pertuturan.
- **CARTA ALIR** : Perwakilan grafik yang menunjukkan langkah penyelesaian sesuatu masalah dan mempunyai hubung kait antara satu sama lain.

CONTOH



## Fasa Pelaksanaan Penyelesaian

- **TUJUAN :** mengubah reka bentuk kepada program yang akan dipasang pada perkakasan dan bersedia melaksanakan penyelesaian.
- Membina dan menghasilkan sistem yang dapat menyelesaikan masalah yang dihadapi.
- Aktiviti pembangunan aplikasi melibatkan pengekodan.
- **PENGEKODAN :** Memerlukan Bahasa pengaturcaraan seperti Java, Javascript dan sebagainya. – mengubah spesifikasi program kepada kod sumber.
- **PENGKOMPILAN :** Proses menukar kod pengaturcaraan kepada kod boleh laksana (executable).

### CONTOH ATURCARA

```
public class contohpg172 {  
    public static void main (String [] args){  
        int bilJam = 20;  
        double gajistaf, kadarsj;  
        kadarsj = 25.00;  
  
        gajistaf = kadarsj * bilJam;  
  
        System.out.println (" Gaji anda ialah RM " +gajistaf);  
    }  
}
```

## Fasa Uji dan Nyah ralat

- Memastikan semua keperluan dipenuhi
- Memastikan semua pengekodan berfungsi seperti yang dikehendaki.
- Memastikan semua modul boleh berfungsi bila digabungkan.
- Memastikan maklum balas dari pengguna sistem untuk tujuan pembetulan dan penambahbaikan.
- Melibatkan pengguna sepenuhnya di peringkat pembangunan.
- Mengesan ralat yang tercicir.
- Membantu pasukan projek membuat dokumentasi dengan mengesan kesilapan oleh pengguna.
- Menyimpan keputusan ujian sebagai bukti penyempurnaan pembangunan sistem.
- **SEMAKAN KOD (Code Review) :** Dilakukan untuk mengesan ralat. Pengatur cara akan merujuk log yang dipaparkan untuk membetulkan dan membuang ralat yang dikesan.

## Jenis-jenis semakan

JENIS SINTAKS	SIAPA ?	BILA ?
Sendiri	Pengarang	Semasa pengekodan.
Rakan Sebaya	Rakan Sebaya	Selepas tamat modul
Selepas tamat modul.	Pasukan projek yang diketuai oleh pakar Bahasa pengaturcaraan.	Selepas kedua-dua peringkat diatas.

**1.7**

**1.7.1**

## MENGHURAIKAN SETIAP FASA DALAM KITARAN HAYAT PEMBANGUNAN SISTEM (SDLC)

# Jenis pengujian dan perincian

### JENIS PENGUJIAN

**SENDIRI**

**SISTEM**

**INTEGRASI**

**PENERIMAAN**

Memastikan setiap unit dalam sistem yang dibina berfungsi.

Menguji sistem secara keseluruhan selepas ujian atas unit individu.

Memastikan sistem dapat berfungsi dengan sistem sedia ada.

Ujian ini dijalankan semasa ujian integrasi oleh pengguna sistem untuk memastikannya memenuhi kehendak<sup>89</sup> pengguna.

## Contoh

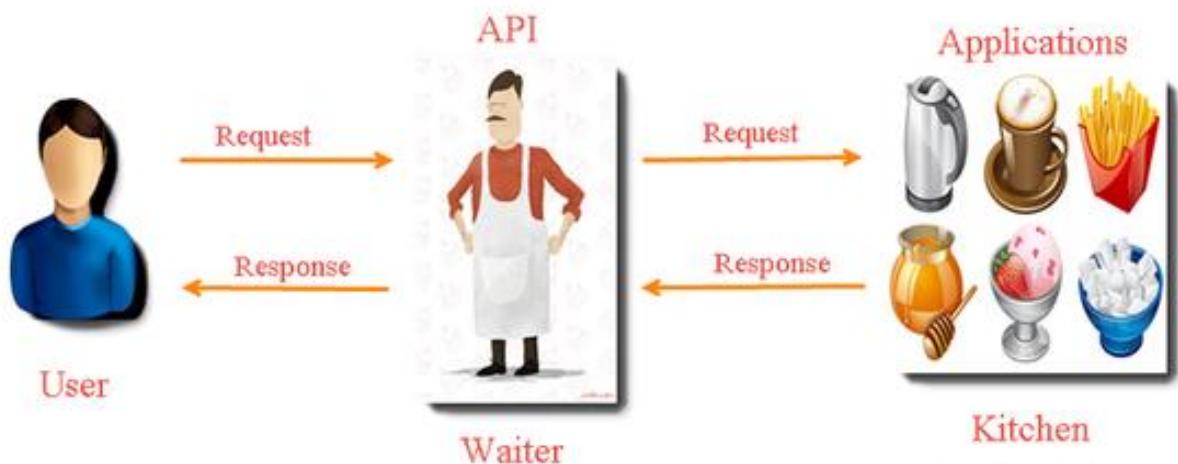
ITEM	AKTIVITI	TANDAKAN (/) ATAU (X)
Ralat Sintaks	Ejaan teks nama boleh ubah	
	Ejaan teks komen	
	Penggunaan objek atau aksara yang tidak dikenali	
	Pengisytiharan jenis data untuk bilangan jam bekerja.	
	Pengisytiharan jenis data untuk gaji staf.	
	Pengisytiharan jenis data kadar sejam	
Ralat Masa Larian	Input pengiraan untuk bilangan jam	
	Input pengiraan untuk kadar sejam	
Ralat Logik	Semak output gaji staf.	

## Application Programming Interface (API)

- Satu set rutin , protocol dan alat untuk membina aplikasi.
- Sesuatu API menentukan bagaimana komponen aplikasi harus berinteraksi.
- API yang baik memudahkan pembangunan aplikasi dengan menyediakan blok pembangunan, di mana pengatur cara komputer akan mencantumkan blok-blok tersebut.
- Contoh : API Google Maps, API Twitter.

### **WHAT IS AN API?**

In computer programming, an **application programming interface (API)** is a set of routines, protocols, and tools for building software applications. An API expresses a software component in terms of its operations, inputs, outputs, and underlying types.



## Fasa Dokumentasi

- Satu proses mengutip dan mengumpulkan data, mengumpulkan maklumat dan ringkasan seperti laporan pengujian yang dijalankan, carta alir, kod atur cara dan juga carta IPO.
- Dokumen-dokumen ini penting untuk rujukan pengguna sistem, pegawai IT dan kakitangan baharu di setiap fasa.
- Dokumentasi yang sepenuhnya bagi fasa projek dari awal pada setiap fasa akan dijadikan rujukan untuk fasa seterusnya.

```
int bilJam = 20; //Pengisytiharan boleh ubah
double gajistaf, kadarsj; //Pengisytiharan boleh ubah
kadarsj = 25.0; // Mendapatkan data kadar bayaran sejam
```

**b** Carta Gantt untuk pembangunan aplikasi bagi mengira gaji staf



**c** Pengujian dan nyah ralat

Nama	Jenis	Penerangan	Catatan
bilJam	integer	Bilangan jam bekerja	Tidak boleh angka negatif
kadarsj	double	Kadar sejam	Dinyatakan dalam RM
gajistaf	double	Hasil darab bilangan jam bekerja dengan kadar sejam	Dinyatakan dalam RM

**d** API

Nama Staf	Input: Bilangan jam bekerja	Input: Kadar Sejam	Output: Gaji Staf	Catatan
A101	20	25.00	500.00	

