

1 SEKILAS TENTANG JAVA

Bahasa pemrograman Java pada awalnya dikhususkan untuk aplikasi berbasis Internet, namun sekarang aplikasi-aplikasi Java sudah dapat digunakan untuk tidak hanya berbasis Web saja tetapi dari basis Desktop hingga aplikasi mobile sudah menggunakannya. Java itu **portable** yaitu dapat dijalankan di berbagai **platform** yaitu Windows, Linux, Unix, MacOS, atau SymbianOS (mobile). Kunci dari portabilitas ini adalah keluaran hasil kompilasi Java bukanlah file *executable* melainkan berbentuk **bytecode**. Bytecode inilah yang akan dieksekusi oleh **JRE** (Java Runtime Environment) yang disebut **JVM** (Java Virtual Machine) dan telah diinstall pada setiap sistem operasi yang akan digunakan, sehingga JVM merupakan **interpreter** bagi bytecode.

Sebenarnya eksekusi interpreter lebih lambat dengan bandingkan dengan kompilasi *executable*, maka Sun menyuplai teknologi HotSpot yang menyediakan compiler **JIT** (Just In-Time) untuk *bytecode* dan menjadi bagian JVM untuk mengkompilasi *bytecode* menjadi *executable code* secara **real-time**, sehingga menjalankan aplikasi Java lebih cepat.

Bahasa Java kali pertama dikonsepsikan oleh James Gosling, Patrick Naughton, Chris Warth, Ed Frank, dan Mike Sheridan di Sun Microsystem tahun 1991, dengan nama bahasa programnya "Oak". Tahun 1995 mereka mengunjungi sebuah café kopi lokal dan mengubah namanya menjadi "Java" hingga sekarang.

Bahasa Java secara langsung berhubungan dengan C dan C++, karena Java menurunkan sintaks-nya dari C dan objeknya diadaptasi dari C++, sehingga Java memiliki sifat **case sensitive** (membedakan antara huruf besar maupun kecil).

1.1 Pemrograman Berorientasi Objek

Java merupakan bahasa pemrograman berorientasi objek atau OOP (Object Oriented Programming). Elemen-elemen dari pemrograman objek ini diantaranya adalah **encapsulation, polymorphism** dan **inherit**.

Encapsulation merupakan mekanisme pemrograman yang mengikat data dan program bersama-sama dan mengamankannya dari penyalahgunaan dan interferensi dari luar. Melalui objek, data dan kode dapat menjadi **private, protected** atau **public** bagi objek tersebut. Seperti yang diketahui kode dan data **private** hanya dapat diakses oleh bagian dalam dari objek tersebut sedangkan kode dan data **protected** aksesnya selain dari dalam objek sendiri, dapat diakses oleh objek keturunannya. Kode dan data **public** dapat diakses oleh objek luar.

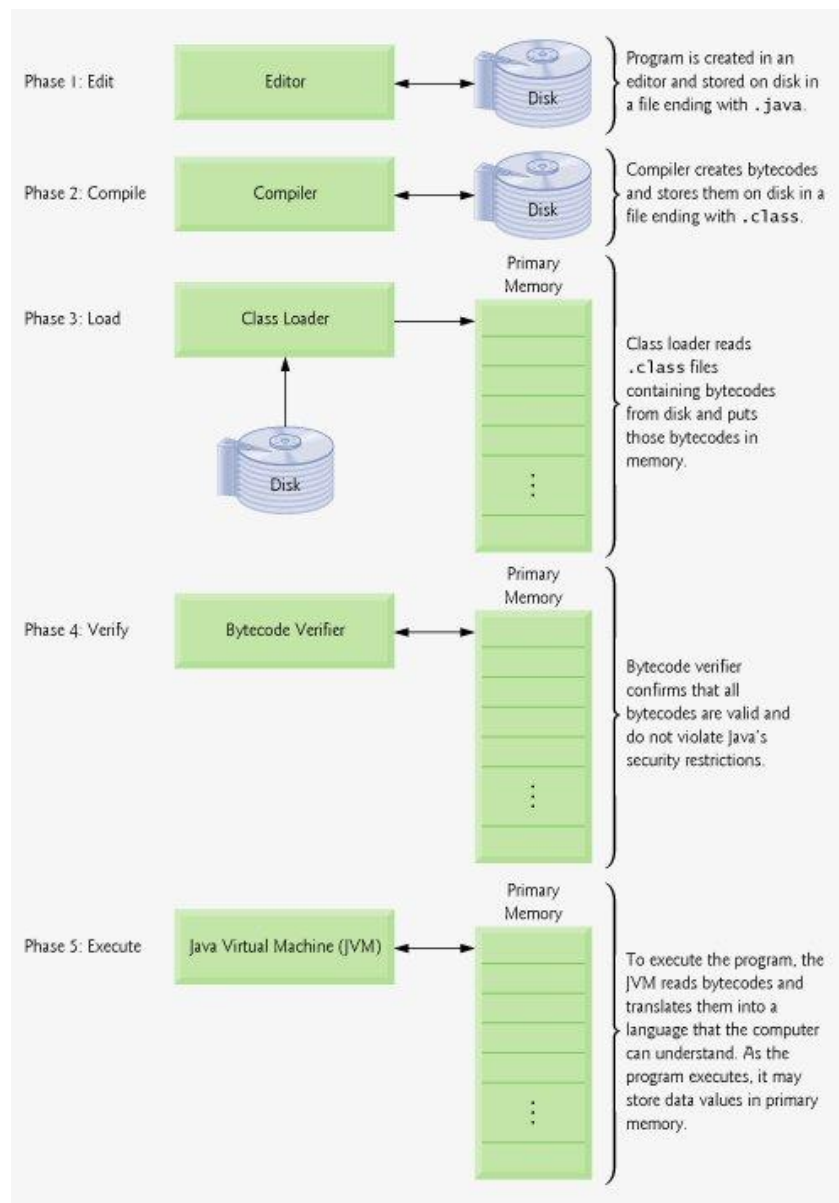
Unit dasar dari encapsulation ini adalah **class**. Class mendefinisikan format dari objek yang akan dibangun sehingga objek merupakan instansi (*instance*) dari class. Kode dan data merupakan anggota (*member*) dari class, dimana data disebut **members variables** atau **instance variables** sedangkan kode yang mengoperasikan data disebut **member methods** atau **methods**. Method dalam java merupakan subrutin atau *function*.

Polymorphism membolehkan satu *interface* mengakses *general class* dalam aksinya. Lebih tepatnya lagi polymorphism merupakan satu *interface* banyak *method*. Misalnya suatu method dengan nama yang sama namun memiliki pengoperasian tipe data yang berbeda.

Inheritance adalah proses dimana suatu objek mendapatkan properti dari objek lain. Konsep ini merupakan **hierarchical classification** yang diperlukan dalam penurunan sifat suatu objek terhadap objek lain yang menjadi orang tuanya (*parent*).

1.2 Java Development Kit (JDK)

Untuk membuat aplikasi Java dibutuhkan JDK yang telah disediakan secara gratis oleh sun Microsystems dan dapat download di www.java.sun.com. Setelah di-install, JDK menyertakan beberapa program penting diantaranya adalah **javac.exe** sebagai *Java compiler* dan **java.exe** sebagai *interpreter* Java.



Gambar 1.1 Lingkungan pemrograman Java

Langkah-langkah pembuatan aplikasi Java adalah sebagai berikut:

Menulis program Java dapat menggunakan teks editor yang telah disediakan oleh masing-masing Sistem Operasi yang digunakan misalnya notepad pada Microsoft Windows atau yang lebih baik lagi menggunakan JCreator hingga Netbean. Contoh programnya di bawah ini.

```
public class Contoh1{
    public static void main(String args[]){
        System.out.print("Halo semua...");
        System.out.println("saya sedang belajar Java nih...!");
        System.out.println("Nggak susah koqq...;-)");
    }
}
```

Simpan dengan nama file yang sama persis (huruf kapitalnya) dengan nama class dan tambahkan ekstension java di akhir file, yaitu Contoh1.java, kemudian kompilasi file teks dengan perintah ini di lokasi folder tempat file tersebut disimpan.

```
C:\>javac Contoh1.java
```

Perintah diatas akan menghasilkan bytecode dengan nama Contoh1.class. Cara menjalankannya bytecode tersebut dengan perintah java dan file bytecode-nya tanpa diikuti ekstension class:

```
C:\>java Contoh1
Halo semua...saya sedang belajar Java nih...!
Nggak susah koqq...;-)
```

1.2.1 Penggunaan Variabel

Variabel merupakan lokasi memori yang telah diberikan nama dan dapat diberikan suatu nilai selama eksekusi program berlangsung. Aturan penamaan variabel sama dengan bahasa-bahasa pemrograman lainnya seperti C atau C++.

Contoh menggunakan tipe data integer:

```
public class Contoh2{
    public static void main(String args[]){
        int v1;
        int v2;
        int v;

        v1=70;
        v2=30;

        v=v1+v2;
        System.out.println(v1+" + "+v2+" = "+v);
        v=v1-v2;
        System.out.println(v1+" - "+v2+" = "+v);
        v=v1*v2;
        System.out.println(v1+" * "+v2+" = "+v);
        v=v1/v2;
        System.out.println(v1+" / "+v2+" = "+v);
        v=v1%v2;
        System.out.println(v1+" % "+v2+" = "+v);
    }
}
```

```
70 + 30 = 100
70 - 30 = 40
70 * 30 = 2100
70 / 30 = 2
70 % 30 = 10
```

Contoh menggunakan tipe data float atau double:

```
public class Contoh3{
    public static void main(String args[]){
        double v1;
        double v2;
        double v;

        v1=23.34;
        v2=12.223;

        v=v1+v2;
        System.out.println(v1+ " + "+v2+" = "+v);
        v=v1-v2;
        System.out.println(v1+ " - "+v2+" = "+v);
        v=v1*v2;
        System.out.println(v1+ " * "+v2+" = "+v);
        v=v1/v2;
        System.out.println(v1+ " / "+v2+" = "+v);
    }
}
```

```
23.34 + 12.223 = 35.563
23.34 - 12.223 = 11.116999999999999
23.34 * 12.223 = 285.28482
23.34 / 12.223 = 1.90951484905506
```

1.2.2 Pengambilan Keputusan (if Statement) & Operator Relasi

Statement if dalam java digunakan untuk membandingkan suatu ekspresi yang menghasilkan nilai true atau false. Operator-operator relasi yang dapat digunakan adalah sebagai berikut:

Operator	Contoh	Keterangan
==	x == Y	Sama dengan
!=	x != Y	Tidak sama dengan
>	x > Y	Lebih besar
>=	x >= Y	Lebih besar atau sama dengan
<	x < Y	Lebih kecil
<=	x <= Y	Lebih kecil atau sama dengan

Contoh programnya adalah sebagai berikut:

```
import java.util.Scanner;
public class Compare{
    public static void main( String args[] ){

        Scanner input = new Scanner( System.in );
        int n1;
        int n2;

        System.out.print( "Ketik angka integer pertama: " );
        n1 = input.nextInt();

        System.out.print( "Ketik angka integer pertama: " );
        n2 = input.nextInt();
    }
}
```

```
if ( n1 == n2 )
    System.out.printf( "%d == %d\n", n1, n2 );
if ( n1 != n2 )
    System.out.printf( "%d != %d\n", n1, n2 );
if ( n1 < n2 )
    System.out.printf( "%d < %d\n", n1, n2 );
if ( n1 > n2 )
    System.out.printf( "%d > %d\n", n1, n2 );
if ( n1 <= n2 )
    System.out.printf( "%d <= %d\n", n1, n2 );
if ( n1 >= n2 )
    System.out.printf( "%d >= %d\n", n1, n2 );
}
```

Ketik angka integer pertama: 25
Ketik angka integer pertama: 23
25 != 23
25 > 23
25 >= 23

1.2.3 Perulangan Proses (Loop: for statement)

Proses pengulangan menggunakan statement for dalam bahasa java dapat dilihat pada contoh di bawah ini:

```
public class GalToLiter{

    public static void main(String args[] ) {
        double galon, liter;
        int counter;
        counter = 0;

        for(galon = 1; galon <= 100; galon++){
            liter = galon * 3.7854;
            System.out.println(galon + " galon adalah " + liter + " liter.");
            counter++;
            if(counter == 10) {
                System.out.println();
                counter = 0;
            }
        }
    }
}
```

1.0 galon adalah 3.7854 liter.
2.0 galon adalah 7.5708 liter.
3.0 galon adalah 11.356200000000001 liter.
4.0 galon adalah 15.1416 liter.
5.0 galon adalah 18.927 liter.
6.0 galon adalah 22.712400000000002 liter.
7.0 galon adalah 26.4978 liter.
8.0 galon adalah 30.2832 liter.
9.0 galon adalah 34.0686 liter.
10.0 galon adalah 37.854 liter.

11.0 galon adalah 41.6394 liter.
12.0 galon adalah 45.424800000000005 liter.
13.0 galon adalah 49.2102 liter.
14.0 galon adalah 52.9956 liter.
15.0 galon adalah 56.781 liter.
16.0 galon adalah 60.5664 liter.
17.0 galon adalah 64.3518 liter.
18.0 galon adalah 68.1372 liter.
19.0 galon adalah 71.9226 liter.
20.0 galon adalah 75.708 liter.
...
91.0 galon adalah 344.4714 liter.
92.0 galon adalah 348.2568 liter.
93.0 galon adalah 352.04220000000004 liter.
94.0 galon adalah 355.8276 liter.
95.0 galon adalah 359.613 liter.

96.0 galon adalah 363.39840000000004 liter.
97.0 galon adalah 367.1838 liter.
98.0 galon adalah 370.9692 liter.
99.0 galon adalah 374.7546 liter.
100.0 galon adalah 378.54 liter.

1.3 Latihan

1.3.1 Permainan FizzBuzz

```
public class FizzBuzz {
    public static void main(String[] args){
        for(int i = 1; i <= 100; i++) {
            if (((i % 5) == 0) && ((i % 7) == 0))
                System.out.print("fizzbuzz");
            else if ((i % 5) == 0)
                System.out.print("fizz");
            else if ((i % 7) == 0)
                System.out.print("buzz");
            else
                System.out.print(i);
            System.out.print(" ");
        }
        System.out.println( );
    }
}
```

1 2 3 4 fizz 6 buzz 8 9 fizz 11 12 13 buzz fizz 16 17 18 19 fizz buzz 22 23 24 fizz 26 27 buzz
29 fizz 31 32 33 34 fizzbuzz 36 37 38 39 fizz 41 buzz 43 44 fizz 46 47 48 buzz fizz 51 52 53
54 fizz buzz 57 58 59 fizz 61 62 buzz 64 fizz 66 67 68 69 fizzbuzz 71 72 73 74 fizz 76 buzz 78
79 fizz 81 82 83 buzz fizz 86 87 88 89 fizz buzz 92 93 94 fizz 96 97 buzz 99 fizz

1.3.2 FizzBuzz dengan Switch

```
public class FizzBuzzSwitch {
    public static void main(String[] args) {
        for(int i = 1; i <= 100; i++) {
            switch(i % 35) {
                case 0:
                    System.out.print("fizzbuzz ");
                    break;
                case 5:
                case 10:
                case 15:
                case 20:
                case 25:
                case 30:
                    System.out.print("fizz ");
                    break;
                case 7:
                case 14:
                case 21:
                case 28:
                    System.out.print("buzz ");
                    break;
                default:
                    System.out.print(i + " ");
                    break;
            }
        }
        System.out.println( );
    }
}
```

1 2 3 4 fizz 6 buzz 8 9 fizz 11 12 13 buzz fizz 16 17 18 19 fizz buzz 22 23 24 fizz 26 27 buzz
29 fizz 31 32 33 34 fizzbuzz 36 37 38 39 fizz 41 buzz 43 44 fizz 46 47 48 buzz fizz 51 52 53
54 fizz buzz 57 58 59 fizz 61 62 buzz 64 fizz 66 67 68 69 fizzbuzz 71 72 73 74 fizz 76 buzz 78
79 fizz 81 82 83 buzz fizz 86 87 88 89 fizz buzz 92 93 94 fizz 96 97 buzz 99 fizz

1.3.3 Factorial

```
public class Factorial {
    public static int factorial_iteration(int x) {
        if (x < 0)
            throw new IllegalArgumentException("x harus >= 0");
        int fact = 1;
        for(int i = 2; i <= x; i++)
            fact *= i;
        return fact;
    }

    public static long factorial_recursive(long x) {
        if (x < 0)
            throw new IllegalArgumentException("x harus >= 0");
        if (x <= 1)
            return 1;
        else
            return x * factorial_recursive(x-1);
    }

    public static void main(String[] args){
        int x=10;

        System.out.println("Hasil factorial (Iterasi) dari "+ x + factorial_iteration(x));
        System.out.println("Hasil factorial (Recursive) dari "+ x + factorial_recursive(x));
    }
}
```

```
Hasil factorial (Iterasi) dari 103628800
Hasil factorial (Recursive) dari 103628800
```

1.3.4 Factorial Input Argumen (Handling Exceptions)

Dependeces Class : Factorial
Argumen : Bilangan integer;

```
public class FactorialInput {
    public static void main(String[] args) {
        try {
            int x = Integer.parseInt(args[0]);
            System.out.println(x + "! = " + Factorial.factorial_recursive(x) );
        }
        catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Anda harus menentukan nilai argumennya.!");
            System.out.println("Usage: java FactorialInput <number>");
        }
        catch (NumberFormatException e) {
            System.out.println("Nilai argumen harus integer");
        }
        catch (IllegalArgumentException e) {
            System.out.println("Bad argument: " + e.getMessage( ));
        }
    }
}
```

```
C:\ >java FactorialInput 35
35! = 6399018521010896896
```

1.3.5 Factorial Input Scanline

Dependeces Class : Factorial

```
import java.io.*;

public class FactorialInput2 {
    public static void main(String[] args) throws IOException {
```

```

BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
for(;;) {
    System.out.print("FactInput> ");
    String line = in.readLine();
    if ((line == null) || line.equals("quit")) break;
    try {
        int x = Integer.parseInt(line);
        System.out.println(x + "! = " + Factorial.factorial_iteration(x));
    }
    catch(Exception e) {
        System.out.println("Invalid Input");
    }
}
}

```

```

FactInput> 13
13! = 1932053504
FactInput> 35
35! = 0
FactInput> quit

```

1.3.6 RotasiText (StringBuffer)

```

import java.io.*;

public class RotasiText {
    public static void main(String[] args) throws IOException {
        BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
        for(;;) {
            System.out.print("> ");
            String line = in.readLine();
            if ((line == null) || line.equals("quit"))
                break;
            StringBuffer buf = new StringBuffer(line);
            for(int i = 0; i < buf.length(); i++)
                buf.setCharAt(i, rotate(buf.charAt(i)));
            System.out.println(buf);
        }

        public static char rotate(char c) {
            if ((c >= 'A') && (c <= 'Z')) {
                c += 8;
                if (c > 'Z')
                    c -= 26;
            }
            if ((c >= 'a') && (c <= 'z')) {
                c += 8;
                if (c > 'z')
                    c -= 26;
            }
            return c;
        }
    }
}

```

```

> Saya Keren Sekali
Aigi Smzmv Amsitq
> quit

```

1.3.7 Number Sorter

```

public class SortNumber {
    public static void sort(double[] nums) {
        for(int i = 0; i < nums.length; i++) {
            int min = i;
            for(int j = i; j < nums.length; j++) {
                if (nums[j] < nums[min]) min = j;
            }
            double tmp;
            tmp = nums[i];
            nums[i] = nums[min];

```

```
        nums[min] = tmp;
    }
}

public static void main(String[] args) {
    double[] nums = new double[10]; // Create an array to hold numbers
    for(int i = 0; i < nums.length; i++) // Generate random numbers
        nums[i] = Math.random() * 100;
    sort(nums); // Sort them
    for(int i = 0; i < nums.length; i++) // Print them out
        System.out.println(nums[i]);
}
}
```

```
17.799983300050627
19.78444473241019
23.897653475054614
49.87040389469707
54.70877117747368
74.81467947567296
84.84605429261437
89.86695373619493
91.09181123951096
96.45387349543292
```

1.3.8 Nilai Prima Terbesar

```
import java.io.*;

public class Sieve {
    public static void main(String[] args) throws IOException{
        BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
        for(;;) {
            System.out.print("Integer> ");
            String line = in.readLine();
            if ((line == null) || line.equals("quit")) break;
            try {
                int max = Integer.parseInt(line);
                boolean[] isprime = new boolean[max+1];
                for(int i = 0; i <= max; i++)
                    isprime[i] = true;
                isprime[0] = isprime[1] = false;
                int n = (int) Math.ceil(Math.sqrt(max));

                for(int i = 0; i <= n; i++) {
                    if (isprime[i])
                        for(int j = 2*i; j <= max; j = j + i)
                            isprime[j] = false;
                }
                int largest;
                for(largest = max; !isprime[largest]; largest--);
                System.out.println("Nilai prima terbesar yang lebih kecil atau sama
dengan " + max + " adalah " + largest);
            }
            catch(Exception e) {
                System.out.println("Invalid Input");
            }
        }
    }
}
```

```
Integer> 10
Nilai prima terbesar yang lebih kecil atau sama dengan 10 adalah 7
Integer> 19
Nilai prima terbesar yang lebih kecil atau sama dengan 19 adalah 19
Integer> quit
```

2 Tipe Data dan Operator

2.1 Tipe Data Primitif

Java memiliki dua kategori tipe data, yaitu tipe data berorientasi objek dan sederhana. Tipe data berorientasi objek didefinisikan oleh class, sedangkan yang sederhana merupakan tipe data dasar yang dimiliki oleh Java sebanyak delapan buah tipe data sederhana yang dikenal dengan sebutan tipe data primitif. Tipe data primitif tersebut adalah sebagai berikut:

Tipe data	Keterangan
boolean	Menyajikan nilai true/false
byte	8-bit integer
char	Karakter
double	Floating-point presisi ganda
float	Floating-point presisi tunggal
int	Integer (bil. Bulat)
long	Long Integer
short	Short integer

2.1.1 Integer

Java mendefinisikan tipe data integer terdiri dari **byte**, **short**, **int** dan **long** dengan masing-masing jangkauan nilainya adalah sebagai berikut:

Tipe	Lebar bit	Jangkauan nilai
byte	8	-128 to 127
short	16	-32,768 hingga 32,767
int	32	-2,147,483,648 hingga 2,147,483,647
long	64	-9,223,372,036,854,775,808 hingga 9,223,372,036,854,775,807

Contoh:

```
public class Inchi {
    public static void main(String[] args){
        long ci,im;

        im=5280 * 12;
        ci=im*im*im;

        System.out.println("Ada "+ci+" kubik inchi dalam mil kubik.");
    }
}
```

```
Ada 254358061056000 kubik inchi dalam mil kubik.
```

2.1.2 Floating-Point

Tipe data ini dalam Java terdiri dari float dan double, masing-masing mempunyai lebar bit yang berbeda, yaitu float (32 bit) dan double (64 bit). Umumnya tipe data double sering digunakan oleh fungsi-fungsi dalam Java.

Contoh:

```
public class Hypotenusa {
    public static void main(String args[]){
        double x,y,z;

        x=8;
        y=5;
        z=Math.sqrt(x*x+y*y);
        System.out.println("Nilai Hypotenusa adalah " +z);
    }
}
```

```
Nilai Hypotenusa adalah 9.433981132056603
```

2.1.3 Karakter

Tidak seperti bahasa-bahasa pemrograman lainnya yang memiliki tipe data char dengan lebar 8-bit, tetapi Java menggunakan *unicode* (karakter yang digunakan oleh berbagai bahasa di dunia misalnya arab, kanji dll) untuk setiap karakter-nya, sehingga lebar bit tiap karakter adalah 16-bit.

Contoh:

```
public class ContohChar {
    public static void main(String[] args){
        char c;

        c='H';
        System.out.println("C berisi "+c);
        c++;
        System.out.println("Sekarang C berisi "+c);
        c=89;
        System.out.println("C berisi "+c);
    }
}
```

```
C berisi H
Sekarang C berisi I
C berisi Y
```

2.1.4 Tipe Boolean

Tipe boolean menyajikan nilai true/false. Contoh penggunaan tipe boolean adalah sebagai berikut:

```
public class ContohBoolean {
    public static void main(String[] args){
        boolean b;

        b=true;
        System.out.println("b bernilai "+b);
        b=false;
        System.out.println("b bernilai "+b);
    }
}
```

```

        if(b)
            System.out.println("b pasti bernilai true");
        else
            System.out.println("b pasti bernilai false");

        System.out.println("7 < 9 adalah "+(7 < 9));
    }

```

```

b bernilai true
b bernilai false
b pasti bernilai false
7 < 9 adalah true

```

2.1.5 Literal (Konstanta)

Literal merujuk pada nilai yang tetap yang disajikan dan dapat dimengerti oleh manusia, misalnya angka 100 atau karakter 'b' (diapit tanda petik sedangkan string diapit kutip). Secara default penulisan angka pada Java adalah integer tetapi jika menginginkan tipe long, gunakan literal L di akhir angka tersebut, misalnya 12L (12 nilai long integer).

2.1.6 Hexadecimal dan Octal

Penulisan angka hexa dimulai dengan tanda 0x, misalnya 0x2F. Untuk angka octal dimulai dengan angka 0 misalnya 023.

2.1.7 Rangkaian Karakter Escape

Untuk karakter-karakter spesial di luar dari karakter standar misalnya karakter Carriage Return, Tab, Line Feed dan sebagainya menggunakan simbol '\'.

Rangkaian Escape	Keterangan
\'	Single quote
\"	Double quote
\\	Backslash
\r	Carriage return
\n	New line
\f	Form feed
\t	Horizontal tab
\b	Backspace
\ddd	Octal constant (where ddd is an octal constant)
\uxxxx	Hexadecimal constant (where xxxx is a hexadecimal constant)

2.1.8 String Literal

Nilai string pada Java diapit dengan menggunakan kutip (petik ganda). Contoh:

```

public class ContohString {
    public static void main(String[] args){
        System.out.print("Ini merupakan baris pertama, kemudian disusul dengan \nbaris
        kedua!\n");
        System.out.print("Contoh penggunaan \"backslash\" '\\'.\n");
        System.out.print("Tabulasi Tab1\tTab2\tTab3");
    }
}

```

```
}
```

Ini merupakan baris pertama, kemudian disusul dengan baris kedua!
 Contoh penggunaan "backslash" '\\'.
 Tabulasi Tab1 Tab2 Tab3

2.2 Operator Aritmatika

Di bawah ini adalah operator-operator yang digunakan dalam proses penghitungan aritmatika:

Operator	Keterangan
+	Penjumlahan
-	Pengurangan (sebagai tanda minus)
*	Perkalian
/	Pembagian
%	Modulus
++	Increment
--	Decrement

Contoh:

```
class ContohModulus {
    public static void main(String args[]) {
        int iredult, irem;
        double dresult, drem;
        iredult = 10 / 3;
        irem = 10 % 3;
        dresult = 10.0 / 3.0;
        drem = 10.0 % 3.0;
        System.out.println("(Integer) Hasil pembagian dan sisa bagi dari 10 / 3: "
+iredult + " " + irem);
        System.out.println("(Float) Hasil pembagian dan sisa bagi dari 10.0 / 3.0: "
+dresult + " " + drem);
    }
}
```

```
(Integer) Hasil pembagian dan sisa bagi dari 10 / 3: 3 1
(Float) Hasil pembagian dan sisa bagi dari 10.0 / 3.0: 3.3333333333333335 1.0
```

2.3 Operator Logika

Operator relasi mengacu pada hubungan bahwa suatu nilai dapat dimiliki yang lain, sedangkan logika mengacu pada cara dimana nilai *true* dan *false* dapat dihubungkan bersama. Dikarenakan operator relasi menghasilkan nilai *true/false*, biasanya sering dihubungkan dengan operator logika.

Operator Logika:

Operator	Keterangan
&	AND
	OR
^	XOR (exclusive OR)
	Short-circuit OR
&&	Short-circuit AND
!	NOT

Contoh:

```
class ContohRelasi {
    public static void main(String args[]) {
        int i, j;
        boolean b1, b2;

        i = 10;
        j = 11;
        if(i < j)
            System.out.println("i < j");
        if(i <= j)
            System.out.println("i <= j");
        if(i != j)
            System.out.println("i != j");
        if(i == j)
            System.out.println("this won't execute");
        if(i >= j)
            System.out.println("this won't execute");
        if(i > j)
            System.out.println("this won't execute");

        b1 = true;
        b2 = false;
        if(b1 & b2)
            System.out.println("this won't execute");
        if(!(b1 & b2))
            System.out.println("!(b1 & b2) is true");
        if(b1 | b2)
            System.out.println("b1 | b2 is true");
        if(b1 ^ b2)
            System.out.println("b1 ^ b2 is true");
    }
}
```

```
i < j
i <= j
i != j
!(b1 & b2) is true
b1 | b2 is true
b1 ^ b2 is true
```

2.4 Operator Assignment

Operator ini digunakan untuk memberikan suatu nilai ke variabel atau objek. Simbol operator ini adalah "=", bentuk sintaks adalah sebagai berikut:

var = nilai;

ada format lain dalam pemberian nilai yang tidak biasanya misalnya seperti di bawah ini:

```
int x,y,z;
x = y = z = 270;
```

bentuk lain dalam penyingkatan operator *assignment* adalah sebagai berikut:

Operator	Contoh	Persamaan
+=	x += 4	x = x + 4
-=	x -= 4	x = x - 4
*=	x *= 4	x = x * 4
/=	x /= 4	x = x / 4
%=	x %= 4	x = x % 4

&=	x &= 4	x = x & 4
 =	x = 4	x = x 4
^=	x ^= 4	x = x ^ 4

2.4.1 Konversi Nilai dalam Assignment

Konversi nilai dari suatu variabel ke variabel lain dapat dilakukan dengan syarat sebagai berikut:

- Dua tipe data compatible
- Tipe tujuan harus lebih besar jangkauannya dibandingkan tipe sumber.

Misalnya sumber integer ke tujuan float.

2.4.2 Type Casting terhadap tipe yang tidak Compatible

Cast adalah instruksi terhadap compiler untuk mengubah satu tipe ke tipe lainnya. Sintaksnya adalah sebagai berikut:

```
(target-type) expression
```

Contoh:

```
class ContohCast {
    public static void main(String args[]) {
        double x, y;
        byte b;
        int i;
        char ch;
        x = 10.0;
        y = 3.0;
        i = (int) (x / y); // cast double to int
        System.out.println("Keluaran Integer dari x / y: " + i);
        i = 100;
        b = (byte) i;
        System.out.println("Nilai dari b: " + b);
        i = 257;
        b = (byte) i;
        System.out.println("Nilai dari b: " + b);
        b = 88; // ASCII code for X
        ch = (char) b;
        System.out.println("ch: " + ch);
    }
}
```

```
Keluaran Integer dari x / y: 3
Nilai dari b: 100
Nilai dari b: 1
ch: X
```

2.5 Latihan

2.5.1 Menampilkan Tabel Logika

```
class ContohLogika {
    public static void main(String args[]) {
        boolean p, q;
        System.out.println("P\t\tQ\t\tAND\t\tOR\t\tXOR\t\tNOT");
        p = true; q = true;
    }
}
```

```

        System.out.print(p + "\t" + q + "\t");
        System.out.print((p&q) + "\t" + (p|q) + "\t");
        System.out.println((p^q) + "\t" + (!p));
        p = true; q = false;
        System.out.print(p + "\t" + q + "\t");
        System.out.print((p&q) + "\t" + (p|q) + "\t");
        System.out.println((p^q) + "\t" + (!p));
        p = false; q = true;
        System.out.print(p + "\t" + q + "\t");
        System.out.print((p&q) + "\t" + (p|q) + "\t");
        System.out.println((p^q) + "\t" + (!p));
        p = false; q = false;
        System.out.print(p + "\t" + q + "\t");
        System.out.print((p&q) + "\t" + (p|q) + "\t");
        System.out.println((p^q) + "\t" + (!p));
    }
}

```

P	Q	AND	OR	XOR	NOT
true	true	true	true	false	false
true	false	false	true	true	false
false	true	false	true	true	true
false	false	false	false	false	true

2.5.2 Parsing Numerik

```

class ParseException extends Exception {
    String errStr; // describes the error

    public ParseException(String str) {
        errStr = str;
    }
    public String toString() {
        return errStr;
    }
}

public class Parser {
    //tipe-tipe token
    final int NONE = 0;
    final int DELIMITER = 1;
    final int VARIABLE = 2;
    final int NUMBER = 3;

    //tipe-tipe syntax error
    final int SYNTAX = 0;
    final int UNBALPARENS = 1;
    final int NOEXP = 2;
    final int DIVBYZERO = 3;

    // indikasi akhir ekspresi.
    final String EOE = "\0";
    private String exp; // refers to expression string
    private int expIdx; // current index into the expression
    private String token; // holds current token
    private int tokType; // holds token's type

    public Parser() {
    }

    // Parser entry point.
    public double evaluate(String expstr) throws ParseException{
        double result;
        exp = expstr;
        expIdx = 0;
        getToken();
        if(token.equals(EOE))
            handleErr(NOEXP); // no expression present
        // Parse and evaluate the expression.
        result = evalExp2();
        if(!token.equals(EOE)) // last token must be EOE
            handleErr(SYNTAX);
        return result;
    }
}

```

```

// Add or subtract two terms.
private double evalExp2() throws ParseException{
    char op;
    double result;
    double partialResult;
    result = evalExp3();
    while((op = token.charAt(0)) == '+' || op == '-') {
        getToken();
        partialResult = evalExp3();
        switch(op) {
            case '-':
                result = result - partialResult;
                break;
            case '+':
                result = result + partialResult;
                break;
        }
    }
    return result;
}

// Multiply or divide two factors.
private double evalExp3() throws ParseException{
    char op;
    double result;
    double partialResult;

    result = evalExp4();
    while((op = token.charAt(0)) == '*' || op == '/' || op == '%') {
        getToken();
        partialResult = evalExp4();
        switch(op) {
            case '*':
                result = result * partialResult;
                break;
            case '/':
                if(partialResult == 0.0)
                    handleErr(DIVBYZERO);
                result = result / partialResult;
                break;
            case '%':
                if(partialResult == 0.0)
                    handleErr(DIVBYZERO);
                result = result % partialResult;
                break;
        }
    }
    return result;
}

// Process an exponent.
private double evalExp4() throws ParseException{
    double result;
    double partialResult;
    double ex;
    int t;
    result = evalExp5();

    if(token.equals("^")) {
        getToken();
        partialResult = evalExp4();
        ex = result;
        if(partialResult == 0.0) {
            result = 1.0;
        } else
            for(t=(int)partialResult-1; t > 0; t--)
                result = result * ex;
    }
    return result;
}

// Evaluate a unary + or -.
private double evalExp5() throws ParseException{
    double result;
    String op;
    op = "";
    if((tokType == DELIMITER) && token.equals("+") || token.equals("-")) {
        op = token;
        getToken();
    }
    result = evalExp6();
}

```

```

        if(op.equals("-")) result = -result;
        return result;
    }

    // Process a parenthesized expression.
    private double evalExp6() throws ParserException{
        double result;
        if(token.equals("(")) {
            getToken();
            result = evalExp2();
            if(!token.equals(")"))
                handleErr(UNBALPARENS);
            getToken();
        }
        else
            result = atom();
        return result;
    }

    // Get the value of a number.
    private double atom() throws ParserException{
        double result = 0.0;
        switch(tokType) {
            case NUMBER:
                try {
                    result = Double.parseDouble(token);
                } catch (NumberFormatException exc) {
                    handleErr(SYNTAX);
                }
                getToken();
                break;
            default:
                handleErr(SYNTAX);
                break;
        }
        return result;
    }
}

// Handle an error.
private void handleErr(int error) throws ParserException{
    String[] err = {
        "Syntax Error",
        "Unbalanced Parentheses",
        "No Expression Present",
        "Division by Zero"
    };
    throw new ParserException(err[error]);
}

// Obtain the next token.
private void getToken(){
    tokType = NONE;
    token = "";
    // Check for end of expression.
    if(expIdx == exp.length()) {
        token = EOE;
        return;
    }
    // Skip over white space.
    while(expIdx < exp.length() && Character.isWhitespace(exp.charAt(expIdx)))
        ++expIdx;
    // Trailing whitespace ends expression.
    if(expIdx == exp.length()) {
        token = EOE;
        return;
    }

    if(isDelim(exp.charAt(expIdx))) { // is operator
        token += exp.charAt(expIdx);
        expIdx++;
        tokType = DELIMITER;
    }
    else if(Character.isLetter(exp.charAt(expIdx))) { // is variable
        while(!isDelim(exp.charAt(expIdx))) {
            token += exp.charAt(expIdx);
            expIdx++;
            if(expIdx >= exp.length())
                break;
        }
        tokType = VARIABLE;
    }
}

```

```

    }
    else if(Character.isDigit(exp.charAt(expIdx))) { // is number
        while(!isDelim(exp.charAt(expIdx))) {
            token += exp.charAt(expIdx);
            expIdx++;
            if(expIdx >= exp.length())
                break;
        }
        tokType = NUMBER;
    }
    else { // unknown character terminates expression
        token = EOE;
        return;
    }
}
// Return true if c is a delimiter.
private boolean isDelim(char c){
    if((" +-/!*%^=()".indexOf(c) != -1))
        return true;
    return false;
}
}

```

File: ParserDemo.Java

Dependeces: Parser

```

import java.io.*;
public class ParserDemo {
    public static void main(String args[]) throws IOException {
        String expr;

        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        Parser p = new Parser();
        System.out.println("Masukan ekspresi kosong untuk berhenti.");
        for(;;) {
            System.out.print("Masukan Ekspresi: ");
            expr = br.readLine();
            if(expr.equals(""))
                break;
            try {
                System.out.println("Hasil: " + p.evaluate(expr));
                System.out.println();
            } catch (ParserException exc) {
                System.out.println(exc);
            }
        }
    }
}

```

```

Masukan ekspresi kosong untuk berhenti.
Masukan Ekspresi: 4*6/3+7
Hasil: 15.0

```

Masukan Ekspresi:

2.6 Latihan (Stream dan File)

2.6.1 Teks I/O

```

import java.io.*;
import java.util.*;

public class TextFile {
    public static void main(String[] args){
        Karyawan[] staff = new Karyawan[3];

        staff[0] = new Karyawan("Jonny Keboy", 75000, 1987, 12, 15);
        staff[1] = new Karyawan("Cecev Gorbacev", 50000, 1989, 10, 1);
        staff[2] = new Karyawan("Gandas Turi", 40000, 1990, 3, 15);
    }
}

```

```
        try{
            PrintWriter out = new PrintWriter("employee.dat");
            writeData(staff, out);
            out.close();

            Scanner in = new Scanner(new FileReader("employee.dat"));
            Karyawan[] newStaff = readData(in);
            in.close();

            for (Karyawan e : newStaff)
                System.out.println(e);
        }
        catch (IOException exception){
            exception.printStackTrace();
        }
    }

    private static void writeData(Karyawan[] employees, PrintWriter out) throws
    IOException{

        out.println(employees.length);
        for (Karyawan e : employees)
            e.writeData(out);
    }

    private static Karyawan[] readData(Scanner in){
        int n = in.nextInt();
        in.nextLine(); // consume newline

        Karyawan[] employees = new Karyawan[n];
        for (int i = 0; i < n; i++){
            employees[i] = new Karyawan();
            employees[i].readData(in);
        }
        return employees;
    }
}

class Karyawan{
    public Karyawan()
        {}

    public Karyawan(String n, double s, int year, int month, int day){
        name = n;
        salary = s;
        GregorianCalendar calendar = new GregorianCalendar(year, month - 1, day);
        hireDay = calendar.getTime();
    }

    public String getName(){
        return name;
    }

    public double getSalary(){
        return salary;
    }

    public Date getHireDay(){
        return hireDay;
    }

    public void raiseSalary(double byPercent){
        double raise = salary * byPercent / 100;
        salary += raise;
    }

    public String toString(){
        return getClass().getName() + "[name=" + name + ",salary=" + salary +
        ",hireDay="+ hireDay + " ]";
    }

    public void writeData(PrintWriter out){
        GregorianCalendar calendar = new GregorianCalendar();
        calendar.setTime(hireDay);
        out.println(name + "|" + salary + "|" + calendar.get(Calendar.YEAR) + "|"
            + (calendar.get(Calendar.MONTH) + 1) + "|"
            + calendar.get(Calendar.DAY_OF_MONTH));
    }
}
```

```

    }

    public void readData(Scanner in){
        String line = in.nextLine();
        String[] tokens = line.split("\\|");
        name = tokens[0];
        salary = Double.parseDouble(tokens[1]);
        int y = Integer.parseInt(tokens[2]);
        int m = Integer.parseInt(tokens[3]);
        int d = Integer.parseInt(tokens[4]);
        GregorianCalendar calendar = new GregorianCalendar(y, m - 1, d);
        hireDay = calendar.getTime();
    }

    private String name;
    private double salary;
    private Date hireDay;
}

```

```

Karyawan[name=Jonny Keboy,salary=75000.0,hireDay=Tue Dec 15 00:00:00 ICT 1987]
Karyawan[name=Cecev Gorbacev,salary=50000.0,hireDay=Sun Oct 01 00:00:00 ICT 1989]
Karyawan[name=Gandas Turi,salary=40000.0,hireDay=Thu Mar 15 00:00:00 ICT 1990]

```

2.6.2 Binary I/O

```

import java.io.*;
import java.util.*;

public class RandomFile{
    public static void main(String[] args){

        Employee[] staff = new Employee[3];

        staff[0] = new Employee("Joni Cracker", 75000, 1987, 12, 15);
        staff[1] = new Employee("Harry Muter-muter", 50000, 1989, 10, 1);
        staff[2] = new Employee("Tommy Tester", 40000, 1990, 3, 15);

        try{
            DataOutputStream out = new DataOutputStream(new
            FileOutputStream("employee.dat"));
            for (Employee e : staff)
                e.writeData(out);
            out.close();
            RandomAccessFile in = new RandomAccessFile("employee.dat", "r");

            int n = (int)(in.length() / Employee.RECORD_SIZE);
            Employee[] newStaff = new Employee[n];

            for (int i = n - 1; i >= 0; i--){
                newStaff[i] = new Employee();
                in.seek(i * Employee.RECORD_SIZE);
                newStaff[i].readData(in);
            }
            in.close();

            for (Employee e : newStaff)
                System.out.println(e);
        }
        catch (IOException e){
            e.printStackTrace();
        }
    }
}

class Employee{
    public Employee() {}

    public Employee(String n, double s, int year, int month, int day){
        name = n;
        salary = s;
        GregorianCalendar calendar = new GregorianCalendar(year, month - 1, day);
        hireDay = calendar.getTime();
    }
}

```

```

    public String getName(){
        return name;
    }

    public double getSalary(){
        return salary;
    }

    public Date getHireDay(){
        return hireDay;
    }

    public void raiseSalary(double byPercent){
        double raise = salary * byPercent / 100;
        salary += raise;
    }

    public String toString(){
        return getClass().getName()
            + "[name=" + name
            + ",salary=" + salary
            + ",hireDay=" + hireDay
            + "]";
    }

    public void writeData(DataOutput out) throws IOException{
        DataIO.writeFixedString(name, NAME_SIZE, out);
        out.writeDouble(salary);

        GregorianCalendar calendar = new GregorianCalendar();
        calendar.setTime(hireDay);
        out.writeInt(calendar.get(Calendar.YEAR));
        out.writeInt(calendar.get(Calendar.MONTH) + 1);
        out.writeInt(calendar.get(Calendar.DAY_OF_MONTH));
    }

    public void readData(DataInput in) throws IOException{
        name = DataIO.readFixedString(NAME_SIZE, in);
        salary = in.readDouble();
        int y = in.readInt();
        int m = in.readInt();
        int d = in.readInt();
        GregorianCalendar calendar = new GregorianCalendar(y, m - 1, d);
        hireDay = calendar.getTime();
    }

    public static final int NAME_SIZE = 40;
    public static final int RECORD_SIZE = 2 * NAME_SIZE + 8 + 4 + 4 + 4;

    private String name;
    private double salary;
    private Date hireDay;
}

class DataIO{
    public static String readFixedString(int size, DataInput in) throws IOException{
        StringBuilder b = new StringBuilder(size);
        int i = 0;
        boolean more = true;
        while (more && i < size){
            char ch = in.readChar();
            i++;
            if (ch == 0)
                more = false;
            else b.append(ch);
        }
        in.skipBytes(2 * (size - i));
        return b.toString();
    }

    public static void writeFixedString(String s, int size, DataOutput out) throws
    IOException{
        for (int i = 0; i < size; i++){
            char ch = 0;
            if (i < s.length())
                ch = s.charAt(i);
            out.writeChar(ch);
        }
    }
}

```

```
}  
}
```

```
Employee[name=Joni Cracker,salary=75000.0,hireDay=Tue Dec 15 00:00:00 ICT 1987]  
Employee[name=Harry Muter-muter,salary=50000.0,hireDay=Sun Oct 01 00:00:00 ICT 1989]  
Employee[name=Tommy Tester,salary=40000.0,hireDay=Thu Mar 15 00:00:00 ICT 1990]
```



3 Statement Pengontrol Program

Dalam bagian ini kita akan mempelajari pernyataan-pernyataan yang mengontrol alur program, yaitu percabangan (pemilihan) dan proses pengulangan.

3.1 Statement Percabangan

Percabangan dalam Java menggunakan dua jenis pernyataan yaitu if dan switch.

3.1.1 Statement if

Format percabangan menggunakan statement if adalah:

```
if (ekspresi){
    statement;
}
else
{
    statement lain;
}
```

Contoh:

```
import java.util.*;

public class TebakBilangan {
    public static void main(String[] args){
        int num;
        Scanner scan= new Scanner(System.in);

        System.out.print("Bilangan:> ");
        num=scan.nextInt();

        if(num % 2==0)
            System.out.println("Tebak:> Genap.");
        else
            System.out.println("Tebak:> Ganjil.");
    }
}
```

```
Bilangan:> 8
Tebak:> Genap.
```

3.1.2 Statement switch

Pernyataan switch menyediakan percabangan *multiway*, sehingga program dapat memilih beberapa alternatif. Format switch adalah sebagai berikut:

```
switch(expression) {
case constant1:
    statement sequence
    break;
case constant2:
    statement sequence
    break;
case constant3:
    statement sequence
    break;
...
default:
    statement sequence
}
```

```
public class ContohSwitch {
    public static void main(String args[]) {
        int i;
        for(i=0; i<10; i++)
            switch(i) {
                case 0:
                    System.out.println(i+" adalah nol");
                    break;
                case 1:
                    System.out.println(i+" adalah satu");
                    break;
                case 2:
                    System.out.println(i+" adalah dua");
                    break;
                case 3:
                    System.out.println(i+" adalah tiga");
                    break;
                case 4:
                    System.out.println(i+" adalah empat");
                    break;
                default:
                    System.out.println(i+" adalah lima atau lebih");
            }
    }
}
```

```
1 adalah satu
2 adalah dua
3 adalah tiga
4 adalah empat
5 adalah lima atau lebih
6 adalah lima atau lebih
7 adalah lima atau lebih
8 adalah lima atau lebih
9 adalah lima atau lebih
```

3.2 Statement Pengulangan

Beberapa statement pengulangan yang digunakan Java diantaranya adalah for, while dan do-while.

3.2.1 Statement Loop (for)

Bentuk pernyataan pengulangan dengan menggunakan for adalah:

```
for(initialization; condition; iteration)
{
    statement sequence
}
```

Initialization, biasanya digunakan untuk memberikan nilai awal bagi variabel pengontrol dalam pengulangan.

Condition, merupakan ekspresi boolean yang menentukan apakah loop akan diulang atau tidak.

Iteration, adalah ekspresi yang mendefinisikan jumlah dimana pengontrol loop nilainya akan berubah setiap kali pengulangan.

```
class SqrRoot {
    public static void main(String args[]) {
        double num, sroot, rerr;
        for(num = 1.0; num < 100.0; num++) {
            sroot = Math.sqrt(num);
            System.out.println("Akar kuadrat dari " + num + " adalah " + sroot);
            // compute rounding error
            rerr = num - (sroot * sroot);
        }
    }
}
```

```
        System.out.println("Kesalahan pembulatan adalah " + rerrr);
        System.out.println();
    }
}
```

```
Akar kuadrat dari 1.0 adalah 1.0
Kesalahan pembulatan adalah 0.0

Akar kuadrat dari 2.0 adalah 1.4142135623730951
Kesalahan pembulatan adalah -4.440892098500626E-16

Akar kuadrat dari 3.0 adalah 1.7320508075688772
Kesalahan pembulatan adalah 4.440892098500626E-16

Akar kuadrat dari 4.0 adalah 2.0
Kesalahan pembulatan adalah 0.0
...
```

3.2.2 Statement Loop (while)

Bentuk pengulangan proses yang lain dalam Java adalah menggunakan pernyataan while. Format sintaksnya adalah:

```
while(condition) statement;
```

statement dapat berupa pernyataan tunggal atau pernyataan dalam blok. Sedangkan *condition* mendefinisikan kondisi yang mengontrol alur pengulangan atau berupa ekspresi Boolean yaitu pengulangan akan berlanjut selagi kondisi itu benar.

```
class Power {
    public static void main(String args[]) {
        int e;
        int result;
        for(int i=0; i < 10; i++) {
            result = 1;
            e = i;
            while(e > 0) {
                result *= 2;
                e--;
            }
            System.out.println("2 pangkat " + i + " adalah " + result);
        }
    }
}
```

```
2 pangkat 0 adalah 1
2 pangkat 1 adalah 2
2 pangkat 2 adalah 4
2 pangkat 3 adalah 8
2 pangkat 4 adalah 16
2 pangkat 5 adalah 32
2 pangkat 6 adalah 64
2 pangkat 7 adalah 128
2 pangkat 8 adalah 256
2 pangkat 9 adalah 512
```

3.2.3 Statement Loop (do-while)

Pernyataan loop menggunakan do-while mempunyai format sintaks sebagai berikut:

```
do {
    statements;
} while(condition);
```

Pola alur do-while hampir sama dengan while hanya saja format ini akan melakukan pengulangan terlebih dahulu baru kemudian memeriksa kondisi apakah harus dilanjutkan atau tidak.

```
class GuesK {
    public static void main(String args[])
        throws java.io.IOException {
        char ch, answer = 'K';
        do {
            System.out.println("Ada huruf antara A dan Z.");
            System.out.print("Anda dapat menebaknya: ");

            do {
                ch = (char) System.in.read();
            } while(ch == '\n' | ch == '\r');
            if(ch == answer)
                System.out.println("*** BENAR ***");
            else {
                System.out.print("...Maaf, Salah ");
                if(ch < answer)
                    System.out.println("abjadnya terlalu rendah");
                else
                    System.out.println("abjadnya terlalu tinggi");
                System.out.println("Coba lagi!\n");
            }
        } while(answer != ch);
    }
}
```

```
Ada huruf antara A dan Z.
Anda dapat menebaknya: j
..Maaf, Salah abjadnya terlalu tinggi
Coba lagi!

Ada huruf antara A dan Z.
Anda dapat menebaknya: K
** BENAR **
```

3.2.4 Statement Break

Statement break digunakan untuk menghentikan alur pengulangan atau keluar dari blok program, misalnya:

```
class ContohBreak {
    public static void main(String args[]) {
        for(int i=0; i<3;i++) {
            System.out.println("Outer loop count: " + i);
            System.out.print(" Inner loop count: ");
            int t = 0;
            while(t < 100) {
                if(t == 10) break;
                System.out.print(t + " ");
                t++;
            }
            System.out.println();
        }
        System.out.println("Loop selesai.");
    }
}
```

```
Outer loop count: 0
Inner loop count: 0 1 2 3 4 5 6 7 8 9
Outer loop count: 1
Inner loop count: 0 1 2 3 4 5 6 7 8 9
Outer loop count: 2
Inner loop count: 0 1 2 3 4 5 6 7 8 9
Loop selesai.
```

Break dapat juga digunakan sebagai pengganti dari statemen *goto* yang melompat pada suatu label, misalnya:

```
class ContohBreakGoto {
    public static void main(String args[]) {
        int i;
        for(i=1; i<4; i++) {
            SATU: {
                DUA: {
                    TIGA: {
                        System.out.println("\nnilai i adalah " + i);
                        if(i==1)
                            break SATU;
                        if(i==2)
                            break DUA;
                        if(i==3)
                            break TIGA;
                        System.out.println("tidak akan dicetak");
                    }
                    System.out.println("Sesudah blok tiga.");
                }
                System.out.println("Sesudah blok dua.");
            }
            System.out.println("Sesudah blok satu.");
        }
        System.out.println("Sesudah for.");
    }
}
```

```
nilai i adalah 1
Sesudah blok satu.

nilai i adalah 2
Sesudah blok dua.
Sesudah blok satu.

nilai i adalah 3
Sesudah blok tiga.
Sesudah blok dua.
Sesudah blok satu.
Sesudah for.
```

3.2.5 Statemen continue

Statemen ini merupakan kebalikan dari *break*, statemen *continue* digunakan untuk melanjutkan pengulangan secara langsung, misalnya:

```
public class ContohContinue {
    public static void main(String args[]) {
        int i;

        for(i = 0; i<=10; i++) {
            if((i%2) != 0)
                continue;
            System.out.println(i);
        }
    }
}
```

```
0
2
4
6
8
10
```

3.3 Latihan

3.3.1 Membuat menu bantuan

```

class Help {
    public static void main(String args[]) throws java.io.IOException {
        char choice;
        for(;;) {
            do {
                System.out.println("Bantuan untuk:");
                System.out.println(" 1. if");
                System.out.println(" 2. switch");
                System.out.println(" 3. for");
                System.out.println(" 4. while");
                System.out.println(" 5. do-while");
                System.out.println(" 6. break");
                System.out.println(" 7. continue\n");
                System.out.print("Pilih satu (q utk keluar): ");
                do {
                    choice = (char) System.in.read();
                } while(choice == '\n' | choice == '\r');
            } while( choice < '1' | choice > '7' & choice != 'q');
            if(choice == 'q') break;
            System.out.println("\n");
            switch(choice) {
                case '1':
                    System.out.println("format if:\n");
                    System.out.println("if(condition) statement;");
                    System.out.println("else statement;");
                    break;
                case '2':
                    System.out.println("format switch:\n");
                    System.out.println("switch(expression) {");
                    System.out.println(" case constant:");
                    System.out.println(" statement sequence");
                    System.out.println(" break;");
                    System.out.println(" // ...");
                    System.out.println("}");
                    break;
                case '3':
                    System.out.println("format for:\n");
                    System.out.println("for(init; condition; iteration)");
                    System.out.println(" statement;");
                    break;
                case '4':
                    System.out.println("format while:\n");
                    System.out.println("while(condition) statement;");
                    break;
                case '5':
                    System.out.println("format do-while:\n");
                    System.out.println("do {");
                    System.out.println(" statement;");
                    System.out.println("} while (condition);");
                    break;
                case '6':
                    System.out.println("format break:\n");
                    System.out.println("break; or break label;");
                    break;
                case '7':
                    System.out.println("format continue:\n");
                    System.out.println("continue; or continue label;");
                    break;
            }
            System.out.println();
        }
    }
}

```

```

Bantuan untuk:
1. if
2. switch
3. for
4. while
5. do-while

```

- 6. break
- 7. continue

Pilih satu (q utk keluar): 5

3.3.2 Menampilkan Daftar Folder

```
import java.io.*;
public class FindFolder{
    public static void main(String[] args){

        if (args.length == 0)
            args = new String[] { ".." };
        try{
            File pathName = new File(args[0]);
            String[] fileNames = pathName.list();

            for (int i = 0; i < fileNames.length; i++){
                File f = new File(pathName.getPath(), fileNames[i]);
                if (f.isDirectory()){
                    System.out.println(f.getCanonicalPath());
                    main(new String[] { f.getPath() });
                }
            }
        }
        catch (IOException e){
            e.printStackTrace();
        }
    }
}
```

```
G:\DOKUMEN\Adobe
G:\DOKUMEN\Adobe\After Effects CS3
G:\DOKUMEN\Adobe\Premiere Elements
G:\DOKUMEN\Adobe\Premiere Elements\7.0
```

3.3.3 Tebak-tebak Angka

```
import java.io.*;
import java.util.*;

public class TebakAngka {

    public static void main(String args[]){
        Scanner scan=new Scanner(System.in);
        BufferedReader reader;
        Random rand=new Random();

        System.out.print("Input batasan nilai <min max>: ");
        int min=scan.nextInt();
        int max=scan.nextInt();
        int acakAngka=0;
        while(acakAngka<min)
            acakAngka=rand.nextInt(max);
        int tebak=-1;
        int jumTebak=0;
        reader=new BufferedReader(new InputStreamReader(System.in));

        System.out.println("Coba tebak angka antara "+min+" hingga "+max+" !");
        boolean invalid=false;
        do{
            jumTebak++;
            try{
                System.out.print("Tebakan anda: ");
                String line=reader.readLine();
                if((line==null) || line.equals("quit")) break;
                tebak=Integer.parseInt(line);
            }
            catch(IOException ioe){}
            catch(NumberFormatException nfe){
                System.out.println("Nilai tidak valid (bukan integer)!");
                tebak=-1;
                invalid=true;
            }
        }
    }
}
```

```
        }
        if (tebak>=min && tebak <=max){
            if (tebak==acakAngka)
                System.out.println("Wow benar tebakan anda yang ke-
"+jumTebak+" BENAR SEKALEEE!");
            else if (tebak<acakAngka)
                System.out.println("Tebak angkanya lebih besar lagi!");
            else
                System.out.println("Tebak angkanya lebih kecil lagi!");
        }
        else if (invalid){
            System.out.println("Ingat angkanya antara "+min+" hingga "+max+"
!");
        }
    }while (tebak!=acakAngka);
}
```

```
Input batasan nilai <min max>: 1 10
Coba tebak angka antara 1 hingga 10 !
Tebakan anda: QUIT
Nilai tidak valid (bukan integer)!
Ingat angkanya antara 1 hingga 10 !
Tebakan anda: 5
Tebak angkanya lebih kecil lagi!
Tebakan anda: 4
Wow benar tebakan anda yang ke-3 BENAR SEKALEEE!
```

4 Object, Class, Message dan Method

Dalam pemrograman berorientasi objek, class merupakan template definisi untuk struktur dan pembuatan objek. Jadi objek merupakan instansi dari suatu *class*, sehingga class merupakan kumpulan perencanaan yang menspesifikasikan bagaimana cara untuk membangun suatu objek.

4.1 Object dan Class

Informasi-informasi yang saling berkaitan dan menjelaskan suatu objek disebut atribut objek (*object attribute*). Atribut ini membolehkan suatu objek mempunyai nilai atributnya yang berdiri sendiri. Selain atribut objek juga memiliki suatu tindakan yang disebut dengan *method*. Sehingga method merupakan kumpulan operasi-operasi dari suatu objek yang berupa kode instruksi-instruksi.

Format umum pembuatan class adalah sebagai berikut:

```
class classname {
    // declare instance variables
    type var1;
    type var2;
    // ...
    type varN;
    // declare methods
    type method1(parameters) {
        // body of method
    }
    type method2(parameters) {
        // body of method
    }
    // ...
    type methodN(parameters) {
        // body of method
    }
}
```

4.1.1 Mendefinisikan Class

Perhatikan contoh definisi class kendaraan yang terdiri dari tiga variabel instansi serta pembuatan objek dari kelas tersebut.

```
public class Kendaraan {
    int penumpang; //jumlah orang dalam kendaraan
    int kapasitas_bb; //kapasitas bahan bakar
    int mpg; //konsumsi bahan bakar mil per gallon
}
```

4.1.2 Membuat Objek

Objek dibuat berdasarkan class yang sudah didefinisikan yaitu class Kendaraan.

```
public class ObjekKendaraan {
    public static void main(String[] args){
        Kendaraan minivan = new Kendaraan();
        Kendaraan sportscar = new Kendaraan();

        int range1, range2;

        minivan.penumpang = 7;
        minivan.kapasitas_bb = 16;
        minivan.mpg = 21;
    }
}
```

```

        sportscar.penumpang = 2;
        sportscar.kapasitas_bb = 14;
        sportscar.mpg = 12;

        range1 = minivan.kapasitas_bb * minivan.mpg;
        range2 = sportscar.kapasitas_bb * sportscar.mpg;
        System.out.println("Minivan dapat membawa " + minivan.penumpang +
            " orang dengan jangkauan " + range1);
        System.out.println("Sportscar dapat membawa " + sportscar.penumpang +
            " orang dengan jangkauan " + range2);
    }
}

```

Minivan dapat membawa 7 orang dengan jangkauan 336
 Sportscar dapat membawa 2 orang dengan jangkauan 168

4.1.3 Menambahkan Method kedalam Class

Class yang telah dibuat di atas hanya terdapat variabel instansi saja, sekarang tambahkan beberapa pengoperasian (*method*) kedalam class tersebut.

```

public class Kendaraan {
    int penumpang; //jumlah orang dalam kendaraan
    int kapasitas_bb; //kapasitas bahan bakar
    int mpg; //konsumsi bahan bakar mil per gallon

    String nama=new String();

    int getRange(){
        return mpg*kapasitas_bb;
    }

    double getNumBB(int miles) { //jumlah kebutuhan bahan bakar
        return (double) miles / mpg;
    }
}

```

```

public class ObjekKendaraan {
    public static void main(String[] args){
        Kendaraan minivan = new Kendaraan();
        Kendaraan sportscar = new Kendaraan();
        double galon;
        int jarak=259;

        minivan.nama="Minivan";
        minivan.penumpang = 7;
        minivan.kapasitas_bb = 16;
        minivan.mpg = 21;

        sportscar.nama="Sportscar";
        sportscar.penumpang = 2;
        sportscar.kapasitas_bb = 14;
        sportscar.mpg = 12;

        galon=minivan.getNumBB(jarak);
        System.out.println(minivan.nama+ " membutuhkan " + galon+
            " galon untuk menempuh jarak " + jarak+" mil.");
        galon=sportscar.getNumBB(jarak);
        System.out.println(sportscar.nama+ " membutuhkan " + galon+
            " galon untuk menempuh jarak " + jarak+" mil.");
    }
}

```

Minivan membutuhkan 12.333333333333334 galon untuk menempuh jarak 259 mil.
 Sportscar membutuhkan 21.583333333333332 galon untuk menempuh jarak 259 mil.

4.1.4 Constructor

Sebuah constructor merupakan inisialisasi objek ketika dibuat. Nama constructor sama dengan nama class-nya.

```
public class Kendaraan {
    int penumpang; //jumlah orang dalam kendaraan
    int kapasitas_bb; //kapasitas bahan bakar
    int mpg; //konsumsi bahan bakar mil per gallon
    String nama=new String();

    Kendaraan(String nama,int penumpang, int kapasitas, int mpg){
        this.nama=nama;
        this.penumpang=penumpang;
        this.kapasitas_bb=kapasitas;
        this.mpg=mpg;
    }

    int getRange(){
        return mpg*kapasitas_bb;
    }

    double getNumBB(int miles) { //jumlah kebutuhan bahan bakar
        return (double) miles / mpg;
    }
}
```

```
public class ObjekKendaraan {
    public static void main(String[] args){
        Kendaraan minivan = new Kendaraan("Minivan",7,16,21);
        Kendaraan sportscar = new Kendaraan("Sportscar",2,14,12);
        double galon;
        int jarak=259;

        galon=minivan.getNumBB(jarak);
        System.out.println(minivan.nama+ " membutuhkan " + galon+
            " galon untuk menempuh jarak " + jarak+" mil.");
        galon=sportscar.getNumBB(jarak);
        System.out.println(sportscar.nama+ " membutuhkan " + galon+
            " galon untuk menempuh jarak " + jarak+" mil.");
    }
}
```

```
Minivan membutuhkan 12.33333333333334 galon untuk menempuh jarak 259 mil.
Sportscar membutuhkan 21.58333333333332 galon untuk menempuh jarak 259 mil.
```



5 SUMBER PUSTAKA

Cay S. Horstmann; Gary Cornell. 2000. **Core Java™ 2: Volume I–Fundamentals**. Prentice Hall.

-----, 2008. **Core Java™ Volume II–Advanced Features**. 8th edition. Prentice Hall.

Danny Poo, Derek Kiong, Swarnalatha Ashok. 2008. **Object-Oriented Programming and Java**. 2nd edition. Springer-Verlag. London.

David Flanagan. 2004. **Java Examples in a Nutshell**. 3rd edition. O'Reilly Media Inc. USA.

Herbert Schildt. 2005. **JAVA™: A Beginner's Guide**. 3rd edition. MacGraw Hill. Osborne.

H. M. Deitel - Deitel & Associates. 2004. **Java™ How to Program**. 6th edition. Prentice Hall. USA.

Joseph P. Rusell. 2001. **JAVA Programming for the absolute beginner**. Prima Publishing. USA.