**[Konsep Dasar OOP pada pemrograman Java](https://hendraenda.wordpress.com/2013/04/01/konsep-dasar-oop-pada-pemrograman-java/)**

Posted on [April 1, 2013](https://hendraenda.wordpress.com/2013/04/01/konsep-dasar-oop-pada-pemrograman-java/) by [hendraenda](https://hendraenda.wordpress.com/author/hendraenda/)

**KONSEP DASAR OOP**

**Pengertian OOP**

**OOP** adalah sebuah metodologi atau cara berpikir dalam melakukan pemrograman dimana pendefinisian tipe data disertai dengan pendefinisian fungsi. Struktur data yang seperti ini disebut dengan istilah *object*. Paradigma pemrograman OOP dapat dilihat sebagai interaksi sebuah *object* dalam melakukan tugasnya.

Segelas teh tentunya memiliki berbagai spesifikasi seperti volume air, rasa, temperatur dan sebagainya. Pada pemrograman, spesifikasi-spesifikasi tersebut merupakan variabel yang dideklarasikan beserta tipe data. Pada OOP, sebuah object Teh juga memiliki *method* berupa fungsi yang dapat dipanggil untuk merubah kelakuan atau spesifikasi teh tersebut. Method minumTeh dapat dideklarasikan sebagai fungsi yang dapat dipanggil untuk mengurangi nilai variabel Volume.

Di samping itu, sebuah *object* dapat memiliki relasi terhadap *object* lainnya. Sebagai contoh, sebuah *object* dapat berupa keturunan dari *objec*t lain.

**Sejarah OOP**

Konsep OOP bermula pada era 1960-an. Sebuah bahasa pemrograman *Simula* memperkenalkan berbagai konsep yang mendasari OOP dengan *SIMULA I* (1962-65) dan *Simula 67* (1967). Kemudian pada tahun 70-an, bahasa pemrograman *Smalltalk* menjadi yang pertama kali disebut *object-oriented*.

Pada tahun 1980-an, dua bahasa pemrograman ADA (US Department of Defense) dan PROLOG (the Japanese “Fifth Generation Computer Project”) dipercayai akan bersaing ketat sebagai bahasa pemrograman yang paling dominan. Namun justru OOP yang menjadi paradigma pemrograman yang paling dominan sampai sekarang. Bahasa pemrograman yang *object-oriented* seperti *C++* pada tahun 80-an menjadi populer. Pada tahun 90-an, bahasa-bahasa pemrograman seperti *Java* mulai menerapkan OOP. Sampai pada 2002, **Microsoft Visual Studio** memperkenalkan bahasa *object-oriented* baru yang diberi nama *C#*. Disusul *VB.NET* yang merupakan penyempurnaan Visual Basic 6.0 yang tidak mendukung OOP.

***Object****–****Oriented Programming*** (OOP) adalah sebuah pendekatan untuk pengembangan / development suatu software dimana dalam struktur software tersebut didasarkan kepada interaksi object dalam penyelesaian suatu proses / tugas. Interaksi tersebut mengambil form dari pesan-pesan dan mengirimkannya kembali antar object tersebut. *Object* akan merespon pesan tersebut menjadi sebuah tindakan / *action* atau metode. Jika kita mencoba melihat bagaimana tugas disekitar kita diselesaikan, kita akan mengetahui bahwa kita berinteraksi dalam sebuah ***object****–****oriented****world*. Jika akan bepergian kita pasti berinteraksi dengan object mobil. Sebagai sebuah object, mobil berisi object-object lain yang berinteraksi untuk melakukan tugasnya membawa kita.

***Object****–****oriented****programs* terdiri dari objects yang berinteraksi satu sama lainnya untuk menyelesaikan sebuah tugas. Seperti dunia nyata, users dari software programs dilibatkan dari logika proses untuk menyelesaikan tugas. Contoh, ketika kamu mencetak sebuah halaman diword processor, kamu berarti melakukan inisialisasi tindakan dengan mengklik tombol printer. Kemudian kamu hanya menunggu respon apakah job tersebut sukses atau gagal, sedangkan proses terjadi internal tanpa kita ketahui. Tentunya setelah kamu menekan tombol printer, maka secara simultan object tombol tersebut berinteraksi dengan object printer untuk menyelesaikan job tersebut.

***Object-Oriented* vs. *Procedural***

*Procedural programming* merupakan paradigma pemrograman selain OOP yang berdasarkan pada konsep *procedure call*. *Procedure* merupakan fungsi yang akan dijalankan secara berantai ketika program aktif baik oleh program itu sendiri atau *procedure* lain. Contoh bahasa pemrograman yang termasuk kedalam paradigma ini diantaranya adalah *Pascal, Delphi, C++, ALGOL, Fortran*, dll.

TYPE Teh = RECORD  
                   NamaTeh:       STRING;  
                   VolumeTeh:     REAL;  
                   WarnaTeh:      ColourType;  
                   …  
         END;

Selanjutnya proses pemasukkan nilai variabel dan fungsi meminum teh dilakukan seperti berikut:

VAR Tehku: Teh;

BEGIN  
   …  
   (\* Inisialisasi atribut teh \*)  
   Tehku.NamaTeh := “Sosro”;  
   Tehku.VolumeTeh := 1.00;  
   …  
   …  
   (\* Meminum teh \*)  
   Tehku.VolumeTeh := Tehku.VolumeTeh – 0.1;

Proses ini tidak berlaku pada pemrograman berbasis object, sebab struktur data Teh hanya mempunyai tipe-tipe variabelnya. Pada OOP, perubahan sebuah object dilakukan dengan pemanggilan method yang dideklarasikan sebelumnya. Seperti yang disebutkan sebelumnya, method minumTeh dapat dideklarasi dan kemudian dipanggil untuk mengurangi Volume teh tersebut.

**Mengapa menggunakan OOP?**

Mengapa OOP dibangun dalam sebuah paradigma yang luas untuk menyelesaikan masalah bisnis? Bahasa prosedural mengatur program dalam mode barisan linier yang bekerja dari atas ke bawah. Dengan kata lain, program adalah kumpulan dari tahapan yang dijalankan setelah yang lain berjalan. **Programming** tipe ini bekerja dengan baik untuk program kecil yang berisi code relative sedikit, tetapi pada saat program menjadi besar, mereka cenderung susah untuk di-*manage* dan di-*debug*. Dalam usaha untuk me-*manage* program, struktur programming diperkenalkan cara untuk mem-*break down* code-code tersebut melalui *functions* dan *procedures*.

Ini adalah sebuah langkah perbaikan, namun pada saat program dijalankan dalam sebuah fungsi bisnis yang kompleks dan berinteraksi dengan sistem lain, maka kelemahan dari struktur metodologi programming muncul kepermukaan meliputi:

Program menjadi lebih susah untuk dimaintain. Fungsi yang tersedia, susah untuk diubah tanpa harus mempengaruhi fungsi sistem secara keseluruhan.

Programming tidak baik untuk team development. Programmers harus mengetahui setiap aspek bagaimana program itu bekerja dan tidak menyebabkan terisolasi usaha mereka atas aspek yang lain dari sistem.

Butuh usaha yang keras untuk menterjemahkan *Business Models*dalam *programming models*.

Mungkin dapat bekerja dengan baik pada saat terisolasi tapi tidak pada saat terintegrasi dengan sistem lain.

<http://santiherliani5g.blogspot.com/2009/01/konsep-oop.html>

Istilah-istilah pada OOP antara lain :

* **Kelas (*Class)***

Kumpulan atas definisi data dan fungsi-fungsi dalam suatu unit untuk suatu tujuan tertentu. Sebagai contoh *‘class of dog*‘ adalah suatu unit yang terdiri atas definisi-definisi data dan fungsi-fungsi yang menunjuk pada berbagai macam perilaku/turunan dari anjing. Sebuah *class* adalah dasar dari modularitas dan struktur dalam pemrograman berorientasi *object*. *Sebuah class* secara tipikal sebaiknya dapat dikenali oleh seorang non-programmer sekalipun terkait dengan domain permasalahan yang ada*,* dan kode yang terdapat dalam sebuah class sebaiknya (relatif) bersifat mandiri dan independen (sebagaimana kode tersebut digunakan jika tidak menggunakan OOP). Dengan modularitas, struktur dari sebuah program akan terkait dengan aspek-aspek dalam masalah yang akan diselesaikan melalui program tersebut. Cara seperti ini akan menyederhanakan pemetaan dari masalah ke sebuah program ataupun sebaliknya.

* **Objek (*Object*)**

Membungkus data dan fungsi bersama menjadi suatu unit dalam sebuah program komputer, objekmerupakan dasar dari modularitas dalam sebuah program komputer berorientasi objek.

* **Abtraksi (*Abstraction*)**

Kemampuan sebuah program untuk melewati aspek informasi yang diproses olehnya, yaitu kemampuan untuk memfokus pada inti. Setiap objek dalam sistem melayani sebagai model dari “pelaku” abstrak yang dapat melakukan kerja, laporan dan perubahan keadaannya, dan berkomunikasi dengan objek lainnya dalam sistem, tanpa mengungkapkan bagaimana kelebihan ini diterapkan. Proses, fungsi atau metode dapat juga dibuat abstrak, dan beberapa teknik digunakan untuk mengembangkan sebuah pengabstrakan.

* **Enkapsulasi (*Encapsulation*)**

Memastikan pengguna sebuah objek tidak dapat mengganti keadaan dalam dari sebuah objek dengan cara yang tidak layak, hanya metode dalam objek tersebut yang diberi ijin untuk mengakses keadaannya. Setiap objek mengakses *interface* yang menyebutkan bagaimana objek lainnya dapat berinteraksi dengannya. Objek lainnya tidak akan mengetahui dan tergantung kepada representasi dalam objek tersebut.

* **Polimorfisme (*Polymorphism*)**

Melalui pengiriman pesan. Tidak bergantung kepada pemanggilan subrutin, bahasa orientasi objek dapat mengirim pesan, metode tertentu yang berhubungan dengan sebuah pengiriman pesan tergantung kepada objek tertentu di mana pesa tersebut dikirim. Contohnya, bila sebuah burung menerima pesan “gerak cepat”, dia akan menggerakan sayapnya dan terbang. Bila seekor singa menerima pesan yang sama, dia akan menggerakkan kakinya dan berlari. Keduanya menjawab sebuah pesan yang sama, namun yang sesuai dengan kemampuan hewan tersebut. Ini disebut polimorfisme karena sebuah variabel tungal dalam program dapat memegang berbagai jenis objek yang berbeda selagi program berjalan, dan teks program yang sama dapat memanggil beberapa metode yang berbeda di saat yang berbeda dalam pemanggilan yang sama. Hal ini berlawanan dengan bahasa fungsional yang mencapai polimorfisme melalui penggunaan fungsi kelas-pertama.

* **Inheritas (*inheritance*)**

Mengatur polimorfisme dan enkapsulasi dengan mengijinkan objek didefinisikan dan diciptakan dengan jenis khusus dari objek yang sudah ada objek-objek ini dapat membagi (dan memperluas) perilaku mereka tanpa haru mengimplementasi ulang perilaku tersebut (bahasa berbasis objek tidak selalu memiliki inheritas.)

Dengan menggunakan OOP maka dalam melakukan pemecahan suatu masalah kita tidak melihat bagaimana cara menyelesaikan suatu masalah tersebut (terstruktur) tetapi objek-objek apa yang dapat melakukan pemecahan masalah tersebut. Sebagai contoh anggap kita memiliki sebuah departemen yang memiliki manager, sekretaris, petugas administrasi data dan lainnya. Misal manager tersebut ingin memperoleh data dari bag administrasi maka manager tersebut tidak harus mengambilnya langsung tetapi dapat menyuruh petugas bag administrasi untuk mengambilnya. Pada kasus tersebut seorang manager tidak harus mengetahui bagaimana cara mengambil data tersebut tetapi manager bisa mendapatkan data tersebut melalui objek petugas administrasi. Jadi untuk menyelesaikan suatu masalah dengan kolaborasi antar objek-objek yang ada karena setiap objek memiliki deskripsi tugasnya sendiri.

<http://yul-elfshawol.blogspot.com/2010/12/konsep-dasar-oop.html>

Pemrograman Berorientasi Objek (Object Oriented Programming) merupakan teknik membuat suatu program berdasarkan objek. Pengembangan program OOP dapat dilakukan dengan mudah karena dapat mengurangi duplikasi kode dan mengurangi kesalahan program.

Program Java dapat dibedakan menjadi dua jenis :

– Applet, adalah program yang dibuat dengan Java, dapat diletakkan pada web server dan di akses melalui web browser.

– Aplikasi, adalah program yang dibuat dengan Java yang bersifat umum dan dapat dijalankan secara langsung.

Versi Java yang telah dirilis oleh Sun Microsystems atau yang lebih dikenal dengan J2SDK (Java 2 System Development Kit) antara lain:

1. J2ME (Java Micro Edition)  
2. J2SE ( Java Standard Edition)  
3. J2EE (Java 2 Entrprise Edition)  
dimana urutan dari yang tertinggi adalah J2EE, J2SE, J2ME

Selain J2SDK ada pengembangan yang lain tergolong sebagai IDE (Integrated Development Environment), contohnya adalah Semantic Cafe, Borland Jbuilder.

Untuk mendownload software Java Development Kit (JDK)

* masuk ke dalam situs<http://java.sun.com/javase/downloads/index.jsp>
* pilih paket JDK versi terbaru, klik tombol Download

Pada OOP ada dua istilah yang sering digunakan yaitu class dan objek.

Class berisi kumpulan dari objek-objek, sedangkan objek merupakan sebuah bentuk model yang tergantung dari cara pandang terhadap objek yang dimaksud.

Untuk lebih memudahkan pemahaman tentang class dan objek, sebagai contoh: sebuah class Kereta api mempunyai objek antara lain gerbong, rel, rem, tempat duduk dan yang lainnya. Setiap objek memiliki atribut dan metode. Atribut adalah segala sesuatu yang berhubungan dengan karakteristik dari objek misalnya warna, bentuk atau fasilitas dari Kereta Api tersebut. Sedangkan Metode dari Kereta Api tersebut adalah kecepatan sehingga dapat berjalan dengan cepat di tengah kemacetan. Jadi metode merupakan fungsi atau segala sesuatu yang dapat dilakukan Kereta Api sebagai objek.

contoh :

public class KrtApi {  
int gerbong;  
String fasilitas;  
}

public class KeretaApi {  
public static void main(String[] args)  
{  
// membuat object  
KrtApi kereta=new KrtApi();  
// memanggil atribut dan memberi nilai  
kereta.gerbong= 4;  
kereta.fasilitas= “AC” ;  
System.out.println(“Jumlah Gerbong:”+ kereta.gerbong);  
System.out.println(“Fasilitas :”+ kereta.fasilitas);  
}  
}

Karakteristik Pemrograman Berorientasi Objek  
1. *Abstraction*  
Yaitu kumpulan objek yang saling berinteraksi.  
2. Enkapsulasi  
Yaitu menyembunyikan suatu proses dalam sistem untuk menghindari interferensi dan   
menyederhanakan penggunaan sistem itu sendiri  
3. *Inheritance* (Pewarisan)  
Suatu class dapat mewariskan atribut dan metode kepada class lain.  
4. *Polymorphism*  
Suatu objek dapat memiliki berbagai bentuk sebagai objek dari classnya sendiri.

Contoh Program *Inheritance* (Pewarisan)

public class KeretaEkonomi extends KeretaApi {

public void setTarif(int tarif) {  
System.out.println(“Tarif :”+ tarif);  
}  
}

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

KeretaEkonomi kereta = new KeretaEkonomi();

// Memanggil method di object

System.out.println(“Tarif Kereta Ekonomi jakarta – Bogor”);  
kereta.setTarif(2500);  
}  
}

Tujuan Instruksional Khusus :  
1. Peserta mengetahui aturan penamaan pada identifier.  
2. Peserta mengetahui kata-kata kunci dan tipe-tipe dasar yang ada di Java.  
3. Peserta mengetahui bagaimana melakukan casting dan promotion.  
Teori Dasar :  
1. Token dan Literal  
Kompilator Java melakukan pemindaian file kode sumber menjadi barisan token.  
Token adalah elemen terkecil didalam program yang mempunyai arti bagi kompilator.  
Token-token inilah yang mendefenisikan struktur bahasa (Java).  
Token Java terbagi dalam lima kategori yaitu :  
1. Identifier  
2. Keyword  
3. Literal  
4. Operator  
5. Separator  
Tokenisasi dilakukan dengan mengekstrak stream karakter di file kode  
sumber untuk memperoleh token-token. Kompilator mengenali komentar dan  
whitespace (spasi, tab, dsb) yang segera tidak dipedulikan. Kompilator  
menghilangkan komentar dan whitespace saat melakukan tokenizing. Token yang  
dihasilkan kemudian merangkai satu kalimat, yang kemudian dikompilasi menjadi  
bytecode.  
1. Identifier  
Identifier adalah token yang merepresentasikan nama. Nama ini  
diciptakan oleh pemrogram untuk memberi identitas pada variabel, metode,  
kelas, paket, dan interface pada program.  
Aturan penamaan identifier :  
1. Case sensitive, huruf kapital dan kecil dibedakan.  
2. Semua karakter di identifier adalah signifikan. Nama identifier dapat  
berapapun panjangnya.  
3. Identifier harus dimulai huruf atau garis bawah (\_), atau tanda dollar ($).  
4. Karakter berikutnya dapat berupa huruf atau angka 0-9.  
5. Simbol-simbol seperti ‘+’, spasi, ‘@’ tidak dapat digunakan.  
6. Nama yang merupakan reserved words atau keywords Java tidak boleh  
digunakan sebagai identifier.  
Lembaga Pelatihan Kerja IT LINK TECH Makassar 1  
Modul 2. Dasar Pemrograman Java  
Modul Java Fundamental 2013/2014  
7. Identifier harus satu kata (tanpa spasi dan whitespace).  
2. Variabel  
Variabel menyatakan suatu lokasi di dalam memori komputer yang digunakan  
untuk menyimpan suatu nilai dan nilai yang ada di dalamnya bisa diubah.  
Contoh deklarasi variabel :  
double salary;  
long distance;  
boolean flag;  
int nrDays;  
int i;  
int j;  
Titik koma mengakhiri masing-masing deklarasi. Titik koma diperlukan karena  
deklarasi merupakan satu pernyataan lengkap di Java.  
3. Konvensi  
Nama adalah case-sensitive sehingga BigBox dan bigBox adalah berbeda.  
Sebaiknya setiap nama menggunakan awalan ‘a’ atau ‘an’, sehingga kita tidak  
mempunyai nama yang hanya berbeda dalam huruf kecil dan huruf besar.  
Contoh :  
Box box; sebaiknya Box aBox;  
Airplane airplane; sebaiknya Airplane anAirplane;  
2. Konstanta  
Java memungkinkan pendefenisian konstanta melalui kata kunci final.  
Contoh :  
Final double PI = 3.14;  
Menyatakan bahwa PI adalah konstanta bertipe double dengan nilai 3,14. Kata  
kunci final menyatakan bahwa PI tidak dapat diubah setelah didefenisikan.  
3. Keyword  
Keyword (kata kunci) adalah identifier yang digunakan Java untuk suatu tujuan  
khusus.  
abstract double int super  
boolean else interface switch  
break extends long synchronized  
byte false native this  
Lembaga Pelatihan Kerja IT LINK TECH Makassar 2  
Modul Java Fundamental 2013/2014  
byvalue final new threadsafe  
case finally null throw  
catch float package transient  
char for private true  
class goto protected try  
const if public void  
continue implements return while  
default import short  
do instanceof static  
4. Literal  
Literal adalah suatu nilai yang dituliskan pada kode sumber Java. Literal pada  
Java dapat dibedakan menjadi  
- Literal bilangan  
- Literal karakter  
- Literal boolean;  
- Literal string  
Literal Bilangan Integer  
Bilangan Integer dapat mempunyai beberapa format berikut :  
1. Decimal (basis 10) sebagai bilangan biasa tanpa notasi khusus.  
2. Hexadecimal (basis 16) dimulai dengan ox atau oX serupa dengan cara C/C++.  
3. Octal (basis 8), dimulai dengan o.  
Contoh :  
Literal Keterangan  
017 17 oktal  
0x17 17 heksadesimal  
17 17 desimal (basis 10)  
0xA A heksadesimal (=10 desimal)  
Contoh Program :  
//Berkas : LiteralBulat.java  
public class LiteralBulat {  
public static void main (String[] args) {  
System.out.println(17);  
System.out.println(017);  
System.out.println(0x17);  
}  
}  
Lembaga Pelatihan Kerja IT LINK TECH Makassar 3  
Modul Java Fundamental 2013/2014  
Literal Floating-Point  
Floating-point merepresentasikan angka desimal dengan bagian pecahan seperti  
3.142. Bilangan floating point dapat diekspresikan dalam notasi standar atau ilmiah.  
Contoh :  
Bilangan 563.84 juga dapat diekspresikan sebagai 5.6384e2.  
Literal Karakter  
Karakter direpresentasikan satu karakter Unicode tunggal dan muncul dipasangan  
tanda petik tunggal (‘...’). Sedangkan karakter khusus direpresentasikan backslash (\)  
diikuti kode karakter.  
Contoh :  
Karakter spesial \n adalah karakter untuk memaksakan baris baru.  
Tabel berikut memperlihatkan sejumlah karakter yang penulisannya diawali  
dengan simbol \.  
Representasi Deskripsi  
\\ Backslash  
\ Continuation  
\b Backspace  
\r Carriage return  
\f Form feed  
\t Horizontal tab  
\n New line  
\’ Single quote  
\” Double quote  
\udddd Unicode Character  
\ddd Octal Character  
Literal Boolean  
Literal menyediakan tipe boolean dengan dua nilai : true dan false. Masing-masing  
merepresentasikan salah satu dari dua kondisi yang mungkin.  
Lembaga Pelatihan Kerja IT LINK TECH Makassar 4  
Modul Java Fundamental 2013/2014  
Literal Character  
Karakter direpresentasikan satu karakter Unicode tunggal dan muncul dipasangan  
tanda petik tunggal (‘...’). Serupa C/C++, karakter khusus (seperti karakter kendali  
dan karakter yang tidak dapat dicetak) direpresentasikan backslash (\) diikuti kode  
karakter.  
Literal String  
String Literal merepresentasikan banyak karakter dan muncul di dalam pasangan  
tanda petik ganda (“...”).  
Dijava, penanganan literal string berbeda dengan literal-literal lain. Untuk literal string  
akan diciptakan objek kelas String. Java menggunakan pendekatan ini karena cara  
terbaik menangani String adalah sebagai kelas.  
5. Type Data dalam Bahasa Java  
Tipe data Java dibagi dalam dua kategori :  
1. Sederhana  
2. Komposit  
Tipe Data Sederhana  
Tipe data sederhana merupakan tipe inti. Tipe data ini tidak diturunkan dari tipe data  
yang lain. Tipe ini disebut juga tipe data primitif.  
Didalam Java terdapat 8 (delapan) tipe primitif yaitu :  
1. Empat tipe adalah untuk bilangan bulat : byte, short, int, long.  
2. Dua untuk tipe angka titik mengambang (floating point) : float dan double.  
3. Satu untuk tipe karakter yaitu char untuk karakter dengan pengkodean  
Unicode: char.  
4. Satu lagi adalah boolean untuk nilai-nilai logika : boolean.  
Tipe Data Komposit  
Tipe data komposit disusun dari tipe sederhana atau tipe komposit lain yang telah  
ada. Tipe ini antara lain : string, array, class, dan interface.  
1. Tipe Data Integer  
Di Java terdapat lima tipe integer, yaitu :  
1. byte  
2. short  
3. int  
Lembaga Pelatihan Kerja IT LINK TECH Makassar 5  
Modul Java Fundamental 2013/2014  
4. long  
5. char  
Tabel 1. Kebutuhan Memori Tipe Integer  
Tipe Ukuran Range  
Byte 8 bit -128 s/d 127  
Short 16 bit -32768 s/d 32767  
Int 32 bit -2147483648 s/d 2147483647  
Long 64 bit -9223372036854775808 s/d  
9223372036854775807  
char 16 bit 0 s/d 65535  
2. Tipe Data Floating-Point  
Bilangan floating-point (titik mengambang), disebut bilangan nyata (real) untuk  
mengevaluasi ekspresi bilangan pecahan. Contoh penggunaannya  
dipenghitungan akar bilangan, sinus, cosinus, dan sebagainya.  
Dua tipe floating-point, yaitu :  
1. Float  
2. Double  
Tabel 2. Kebutuhan Memori Tipe Floating-Point  
Tipe Ukuran Range  
Float 32 bits 3.4e038 s/d 3.4e+038  
Double 64 bits 1.7e-308 s/d 1.7e+308  
3. Tipe Data Boolean  
Java mempunyai tipe sederhana boolean nilai logis. Kita tidak dapat  
mengkonversi tipe bilangan bulat dan nilai boolean. Tipe data boolean  
menyimpan salah satu dari dua nilai, yaitu :  
1. True  
2. False  
4. Tipe Data Character  
Tipe character untuk menyimpan karakter Unicode tunggal. Karena karakter  
Unicode disusun 16-bit, tipe data char adalah 16-bit unsigned integer.  
Lembaga Pelatihan Kerja IT LINK TECH Makassar 6  
Modul Java Fundamental 2013/2014  
Pertama :  
Tanda Petik Tunggal sebagai Penanda Karakter  
Tanda petik tunggal untuk menandai konstanta/literal char.  
Kedua :  
Unicode  
Unicode dirancang untuk menangani semua karakter di dunia, didalam kode 2  
byte. Kode 2 byte memungkinkan 65.536 karakter. Himpunan karakter ASCII  
berada di range 0 sampai 127, dan extended 8 bit disebut ISO-Latin-1 (ISO  
8859-1) di range 0 sampai 255.  
Informasi Unicode lebih lanjut, terdapat di website http://www.unicode.org.  
Beberapa escape untuk karakter-karakter spesial sebagai berikut :  
Tabel 3. Karakter Unicode Spesial  
Barisan Escape Nama Nilai Unicode  
\b Backspace \u0008  
\t Tab \u0009  
\n Linefeed \u000a  
\r Carriage Return \000d  
\” Petik Ganda \u0022  
\’ Petik Tunggal \u0027  
\\ Backslash \u005c  
Lembaga Pelatihan Kerja IT LINK TECH Makassar 7  
Modul Java Fundamental 2013/2014  
Latihan :  
Ketiklah listing program dibawah ini :  
1. Program untuk memberikan nilai pada suatu tipe  
2. Program untuk mencetak nilai default dari tipe dasar  
Lembaga Pelatihan Kerja IT LINK TECH Makassar 8  
Modul Java Fundamental 2013/2014  
4. Program menampilkan bilangan oktal ke format desimal  
Soal:  
Menganalisa batasan maksimum dari suatu tipe.  
Amatilah dan tulislah program berikut ini :  
public class BigInteger {  
public static void main (String args[]) {  
long p = 2147483648;  
}  
}  
Lakukan kompilasi pada file tersebut dan amati pesan kesalahannya. Lakukan analisa  
mengapa bisa terjadi kesalahan padahal batasan nilai maksimum dari suatu bilangan bertipe  
long adalah 263 – (9223372036854775808)?. Kemudian berikanlah solusi yang tepat untuk  
mengatasi persoalan diatas.  
Lembaga Pelatihan Kerja IT LINK TECH Makassar 9

Apa itu konstruktor (constructor) ?

Dalam pemrograman berorientasi objek, kita akan mengenal yang namanya konstruktor.

Konsktruktor adalah method yang pertama kali dijalankan pada saat sebuah objek pertama kali diciptakan.

Sifat konstruktor:  •Nama konstruktor sama dengan nama class

                            •Tidak memiliki nilai balik dan tidak boleh ada kata kunci void

Perbedaan method konstruktor dengan method lainnya pada java adalah nama method konstruktor harus sama dengan nama kelasnya.

Contohnya :

|  |  |
| --- | --- |
| Kode | |
| 1  2  3  4 | class persegiPanjang{  public persegiPanjang(){  }  } |

Lalu apa fungsi konstruktor ?

Konstruktor biasanya digunakan untuk menginisialisasi atau memberikan nilai  (melalui parameter) kepada sebuah objek pada saat objek pertama kali diciptakan.

Contohnya sebagai berikut :

|  |  |
| --- | --- |
| Kode | |
| 1  2  3  4  5  6  7 | class persegiPanjang{  double panjang,lebar;  public persegiPanjang(double p, double l){  this.panjang = p;  this.lebar = l;  }  } |

Untuk menggunakan class diatas, kodenya sebagai berikut :

|  |  |
| --- | --- |
| Kode | |
| 1 | persegiPanjang persegi1  = new persegiPanjang(10, 2); |

Pada saat objek **persegi1** dibuat, kita memberikan 2 nilai untuk konstruktor yang nantinya akan digunakan untuk memberi nilai pada attribut panjang dan lebar di class.

Kesimpulannya, pada saat objek persegi1 dibuat, objek tersebut sudah memiliki nilai panjang dan nilai lebar.

kenapa harus menggunakan konstruktor  ?

Simpel saja, kodenya diatas jadi lebih sederhana.

karena jika tanpa konstruktor kodenya akan jadi seperti ini :

|  |  |
| --- | --- |
| Kode | |
| 1 | persegiPanjang persegi2 = new persegiPanjang(); |

Pada saat objek persegi2 dibuat, objek tersebut belum memiliki nilai  panjang dan lebar. Hanya menjadi sebuah objek yang berasal dari class persegiPanjang saja dan kita harus menambahkan kode berikut agar objek persegi2 mendapat nilai panjang dan lebar :

|  |  |
| --- | --- |
| Kode | |
| 1  2 | persegi2.panjang = 10;  persegi2.lebar = 2; |