# Pemrograman Berorientasi Obyek

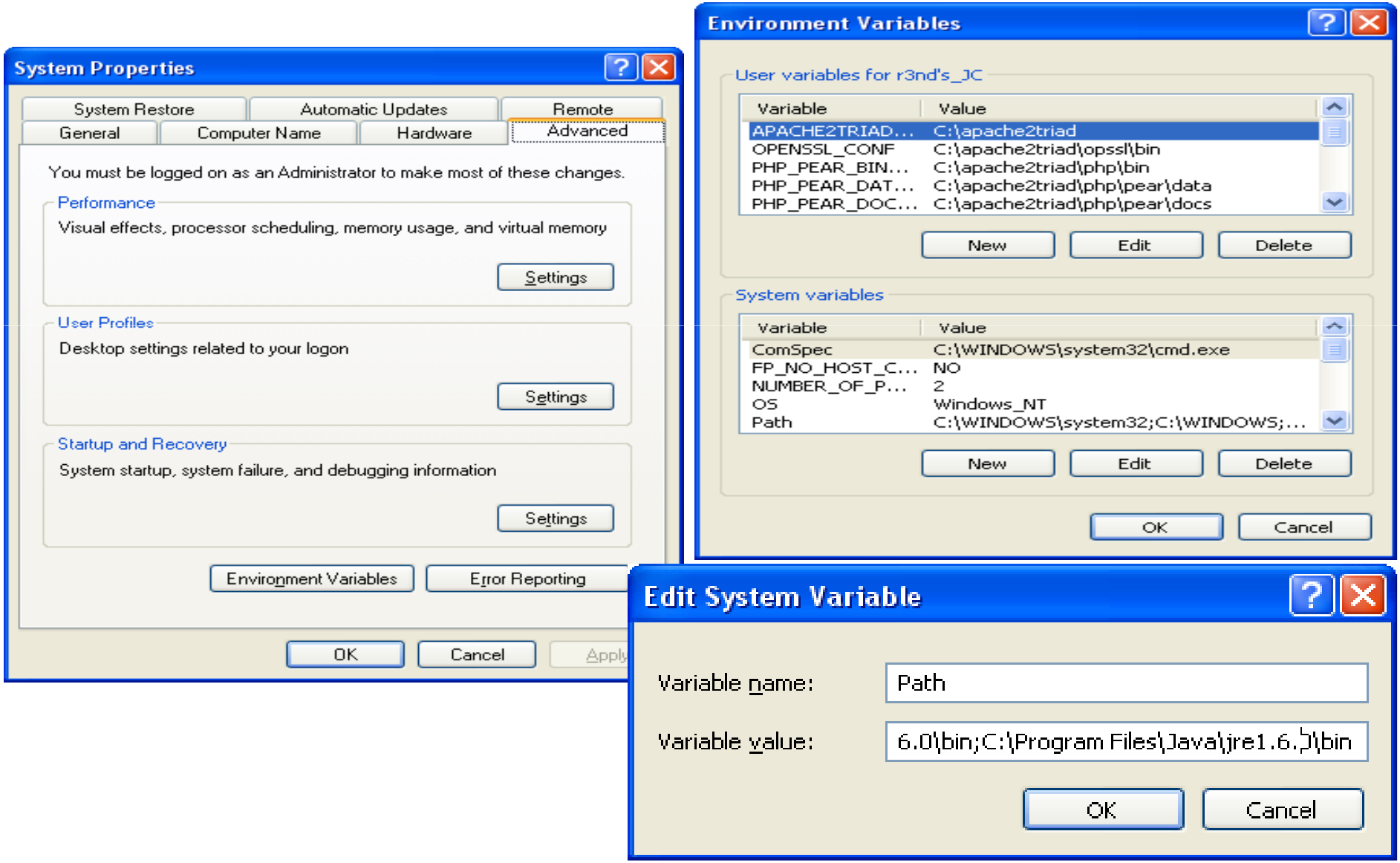
Class dan Object I

Antonius Rachmat C

# REVIEW: Instalasi JDK

- Download JDK for free
- Instalasi biasa
- Set **PATH** dan **JAVA_HOME**
  - set PATH=%PATH%;<your Java\Bin directory>
  - set JAVA_HOME=<your Java directory>
- Bisa juga dilakukan lewat Windows GUI
  - Control Panel > System Properties > Environment Variable

# Set PATH


System Properties
System Restore
Automatic Updates
Remote
General
Computer Name
Hardware
Advanced
You must be logged on as an Administrator to make most of these changes.
Performance
Visual effects, processor scheduling, memory usage, and virtual memory
Settings
User Profiles
Desktop settings related to your logon
Settings
Startup and Recovery
System startup, system failure, and debugging information
Settings
Environment Variables
Error Reporting
OK
Cancel
Environment Variables
User variables for r3nd's_JC
Variable
Value
APACHE2TRIAD...
C:\apache2triad
OPENSSL_CONF
C:\apache2triad\opssl\bin
PHP_PEAR_BIN...
C:\apache2triad\php\bin
PHP_PEAR_DAT...
C:\apache2triad\php\pear\data
PHP_PEAR_DOC...
C:\apache2triad\php\pear\docs
New
Edit
Delete
System variables
Variable
Value
ComSpec
C:\WINDOWS\system32\cmd.exe
FP_NO_HOST_C...
NO
NUMBER_OF_P...
2
OS
Windows_NT
Path
C:\WINDOWS\system32;C:\WINDOWS;...
New
Edit
Delete
OK
Cancel
Edit System Variable
Variable name:
Path
Variable value:
6.0\bin;C:\Program Files\Java\jre1.6.0\bin
OK
Cancel

# REVIEW: Editor

- Notepad
- Notepad++
- Editplus
- Jcreator Lite / Pro
- Netbeans 7.x
- IntType
- Eclipse

# PBO

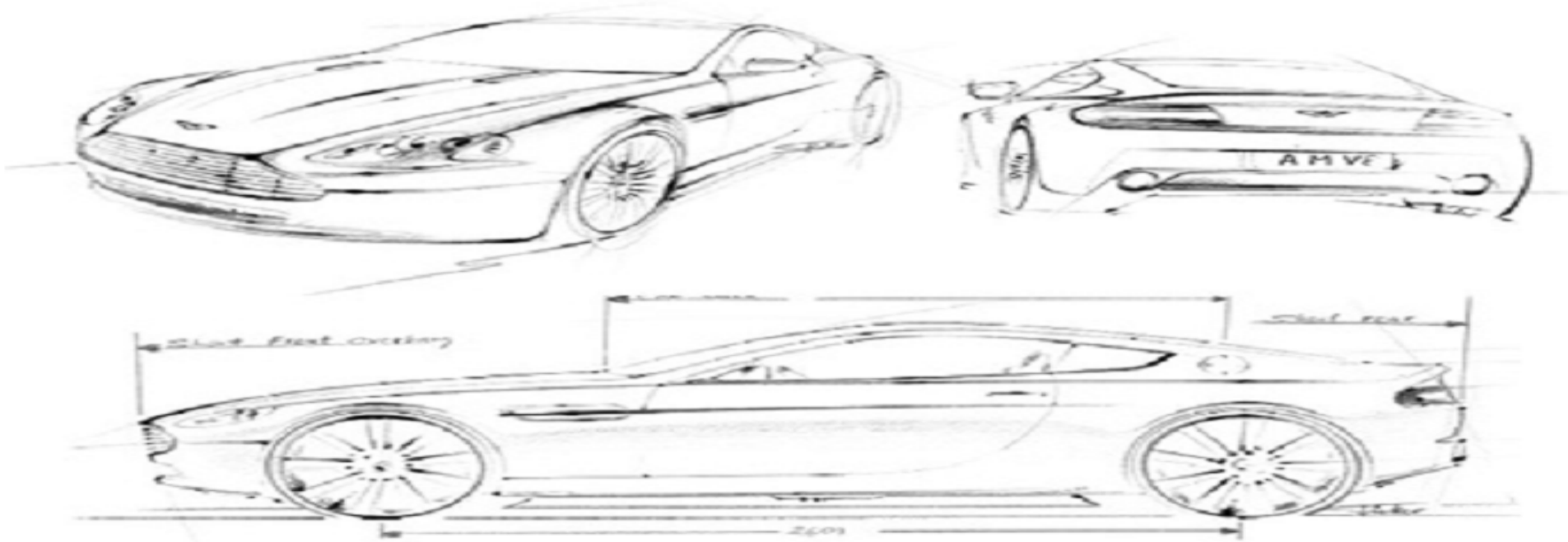
- Semua masalah akan dianggap sebagai "**benda**" atau dikenal sebagai "**obyek**"
- "Obyek" => sesuatu yang nyata, memiliki bentuk, memiliki **atribut** yang melekat padanya, dan memiliki **tingkah laku** yang dapat dilakukannya
- Untuk mengeneralisasi "**rancangan obyek**" digunakan istilah "**class**"

# Class

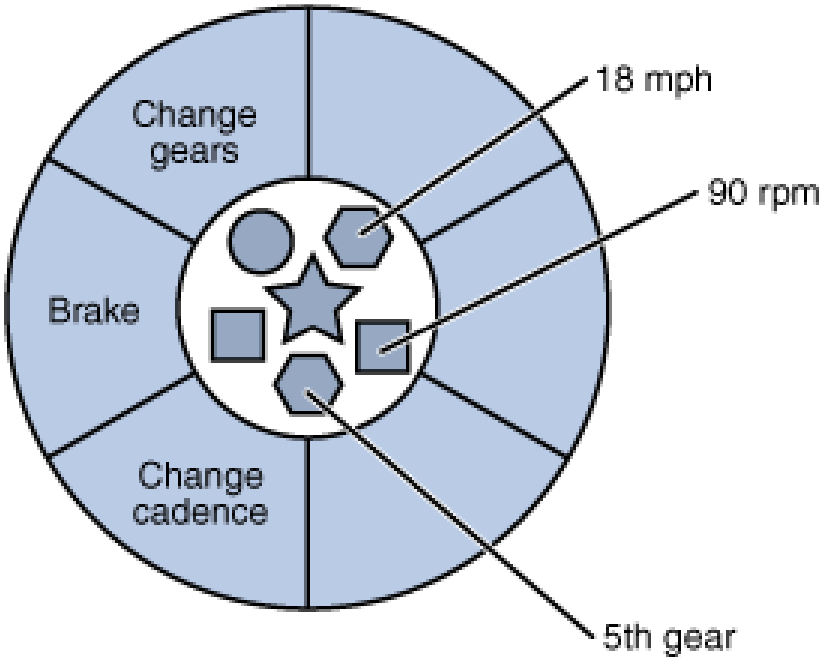
- Suatu **template** (rancangan) yang dapat digunakan untuk menggambarkan suatu “obyek” yang bersifat umum untuk semua “obyek” tersebut
- Contoh class:
  - Class mobil
  - Class hewan
  - Class komputer
  - Class televisi
- Biasanya berupa **kata benda** dan bersifat umum

# Object

- **Bentuk nyata** dari suatu class
- Bukan lagi merupakan template (rancangan)
- Bersifat khusus (spesifik)
- Contoh object:
  - Mobil Sedan
  - Mobil Jip
  - Komputer laptop
  - Komputer desktop
  - Harimau
  - Kucing



# Class




- Contoh: **class sepeda**
- Ada 3 **atribut/state/variabel** kelas = speed, rpm, dan gear.
- **Nilai/value** untuk atribut speed= 18 mph
- Ada 3 **services / methods / behavior**:
  - Ubah gigi
  - Rem
  - Ubah irama berkendara / kecepatan

# Berbagai macam **jenis** objek

- **Concrete**: manusia, mobil, ponsel
- **Intangible***: keluarga, kualitas, ide, account
- **Roles**: dokter, pasien, manager, admin
- **Judgements**: gaji tinggi, pintar
- **relational**: partnership, pernikahan, ownership
- **events**: penjualan, system crash
- **displayable**: string, integer, image

*** Incorporeal property that is saleable though not material, cannot be seen, handled, smelled**

# Class & Object

**Contoh Class lain**

- Rancangan sebuah **flashdisk**:
  - Ada port USB
  - Ada rangkaian penyimpan data (flash storage)
  - Apa bisa dilakukan flashdisk? (read and write)

**Contoh Obyek**

- Object: Flashdisk Kingston 8 GB, 4 GB, 2 GB
- Object: Flashdisk SanDisk 8 GB, 4 GB, 2 GB

# Class Flashdisk

Flashdisk
---
-versi_usb
-kapasitas
-warna
-merk
-kec. write
-kec read
---
+write()
+read()

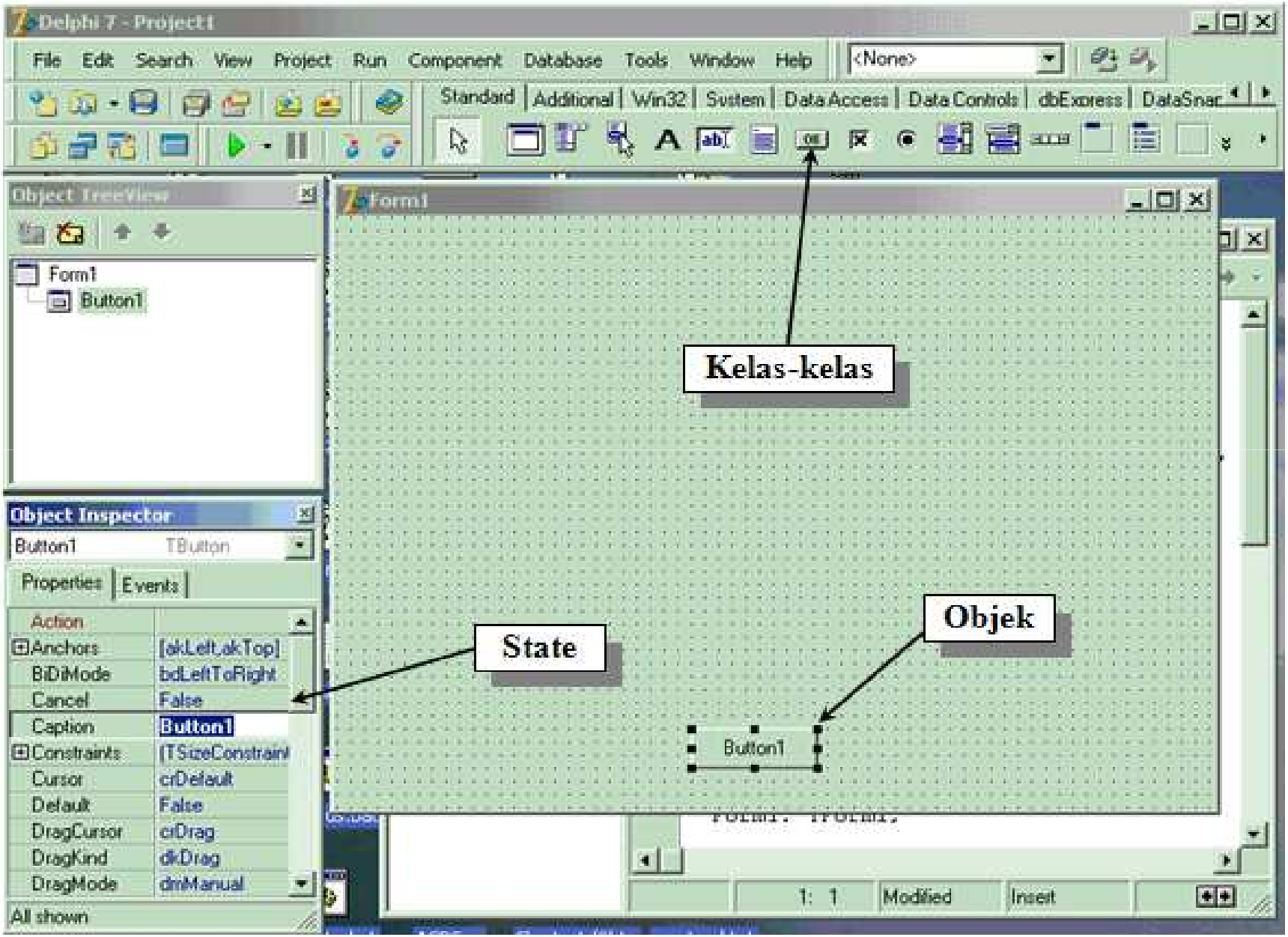
# Contoh lain Class dan Obyek

| Class mobil | | Obyek mobil A | Obyek Mobil B |
|---|---|---|---|
| Variabel Intsance | Nomor Plat | ABC 111 | XYZ 123 |
| | Warna | Biru | Merah |
| | Manufaktur | Mitsubishi | Toyota |
| | Kecepatan | 50 km/h | 100 km/h |
| Method *Instance* | Method Akselerasi | | |
| | Method Belok | | |
| | Method Rem | | |

# Attribute dan Method

- **Attribute**: Ciri pembeda antar obyek
- **Behavior**: Tingkah laku sebuah obyek

| *Obyek* | *Atribut* | *Tingkah Laku* |
|---|---|---|
| Mobil | Tipe dari transmisi<br>manufaktur<br>Warna | Berbelok<br>Mengerem<br>Mempercepat |
| Singa | Berat<br>Warna<br>Lapar atau tidak lapar<br>Jinak atau liar | roaring<br>Tidur<br>Berburu |


Delphi 7 - Project1
File Edit Search View Project Run Component Database Tools Window Help
Standard Additional Win32 System Data Access Data Controls dbExpress
Object TreeView
Form1
Button1
Object Inspector
Button1 TButton
Properties Events
Action
Anchors [akLeft,akTop]
BiDiMode bdLeftToRight
Cancel False
Caption Button1
Constraints (TSizeConstrain
Cursor crDefault
Default False
DragCursor crDrag
DragKind dkDrag
DragMode dmManual
All shown
Form1
Kelas-kelas
State
Objek
Button1
1: 1
Modified
Insert

# Object Creation

- Program terdiri dari ”**kumpulan obyek**”
- Obyek-obyek tersebut di-”**ciptakan**” berdasarkan definisi class yang sudah dibuat sebelumnya
- Setiap obyek harus memiliki tipe (pada Java)
  - Ingat, Java bersifat **strong type** programming languange

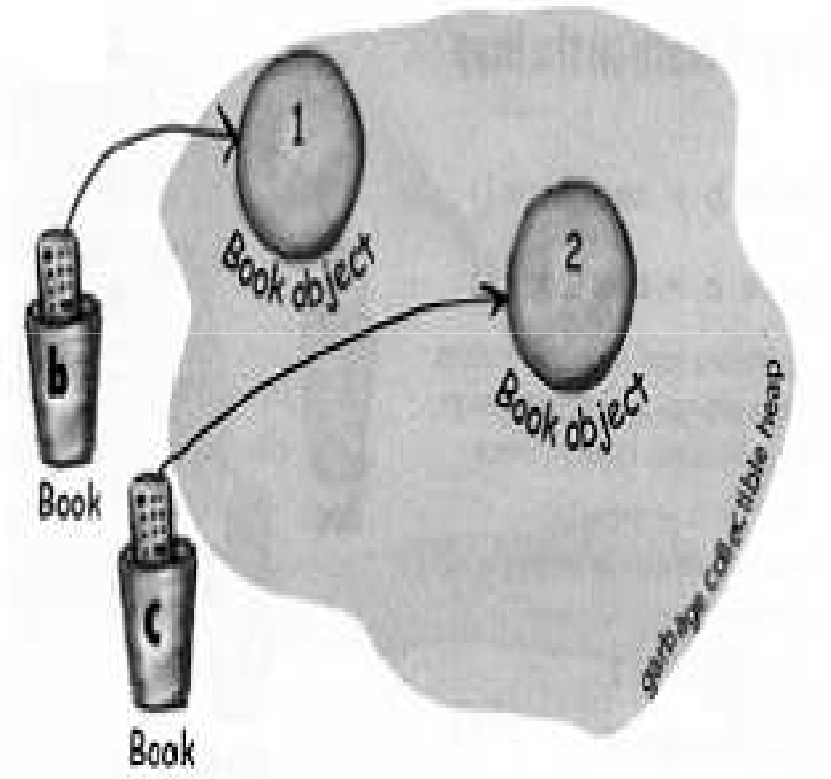
# Object Creation

- Tipe dari obyek ditentukan dari "**cetakannya**"
- Tipe dari obyek ditentukan dari definisi class
- Pembuatan obyek dari definisi class dinamakan **instansiasi** (instansiation)
- Obyek disebut juga sebagai **instance**

# Instansiasi

- Agar suatu class dapat digunakan, obyeknya harus dilakukan **penciptaan** obyek terlebih dahulu
- Biasanya terjadi secara **manual / eksplisit**
- Penciptaan obyek => **instansiasi**
- Proses instansiasi berarti **mengalokasikan semua memory** yang dibutuhkan oleh obyek tersebut termasuk **menginisialisasinya**

# Ilustrasi class dan instansiasi




```
Book b = new Book();
Book c = new Book();
```

*The 2 Book objects are now living on the heap.*

References=2

Objects=2

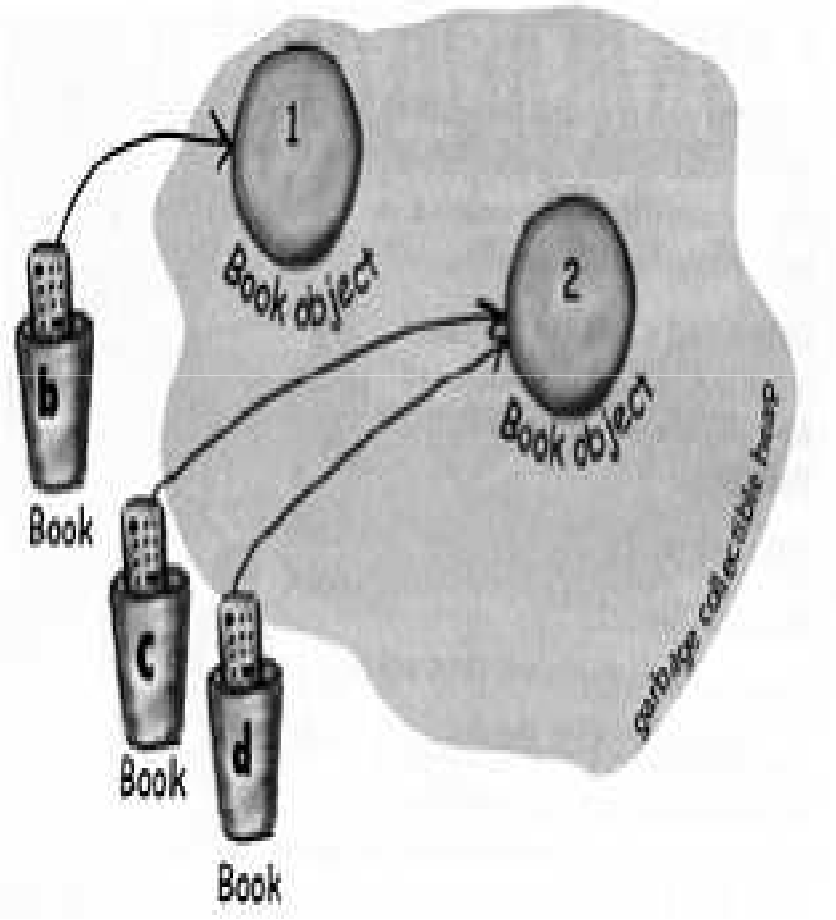
# Referensi dan objects

```
Book d = c;
```

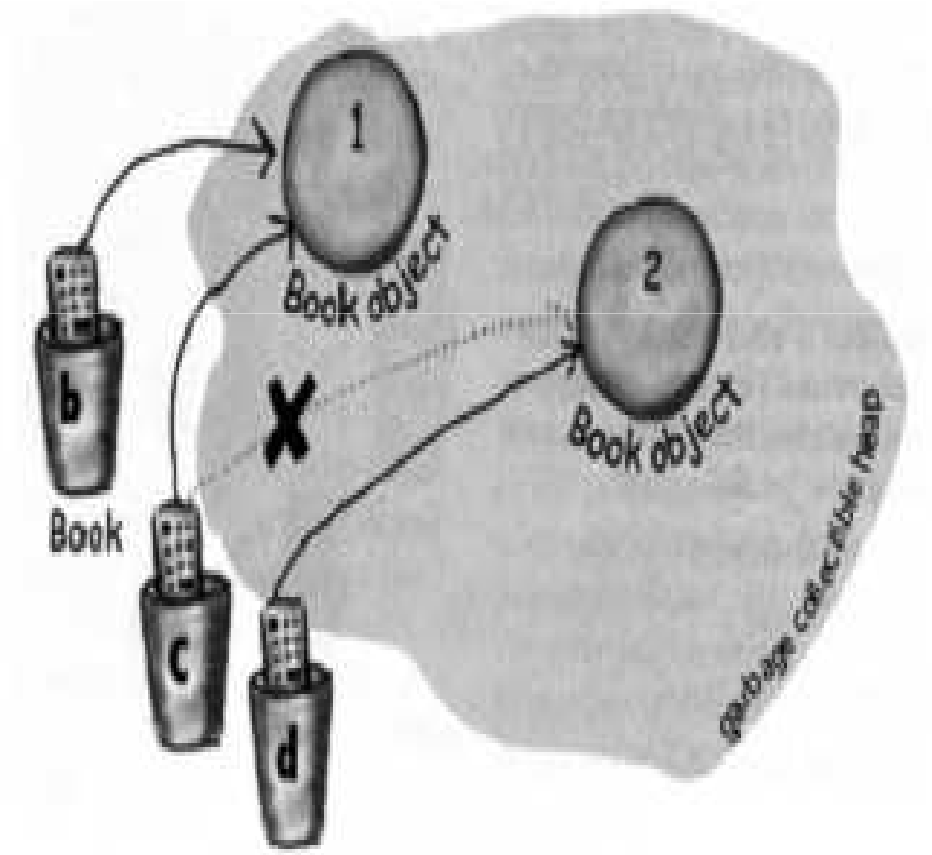
*Both c & d refer to the same object.*

References=3

Objects=2



# Referensi dan objects




```
c = b;
```

*Both b & c refer to the same object.*

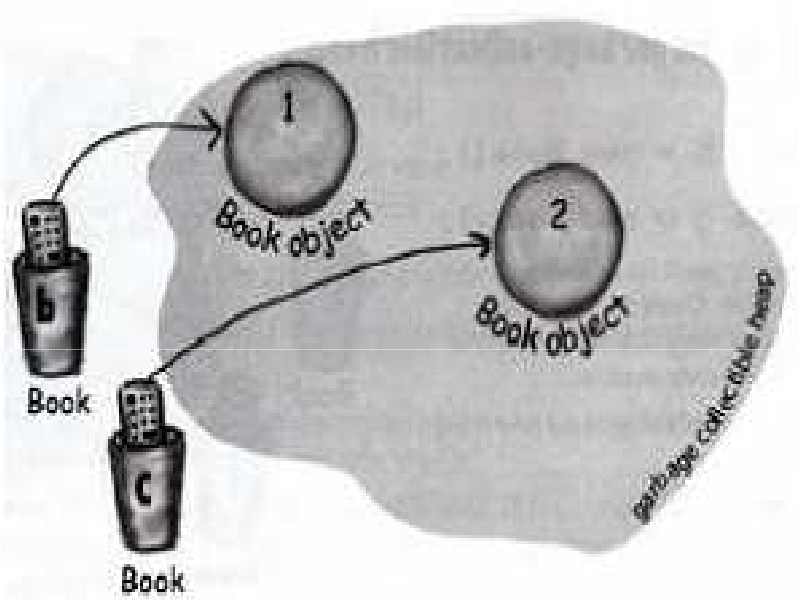
References=3

Objects=2

# Garbage Collector

- Tool yang bertugas membersihkan semua obyek yang sudah tidak terpakai oleh object manapun
- Terjadi secara otomatis oleh Java
- Terdapat dalam sistem JVM

# Ilustrasi kerja Garbage Collector




```
Book b = new Book();
Book c = new Book();
```

*The 2 Book objects are now living on the heap.*

Active References=2

Reachable Objects=2

# Ilustrasi kerja Garbage Collector

```
b = c;
```

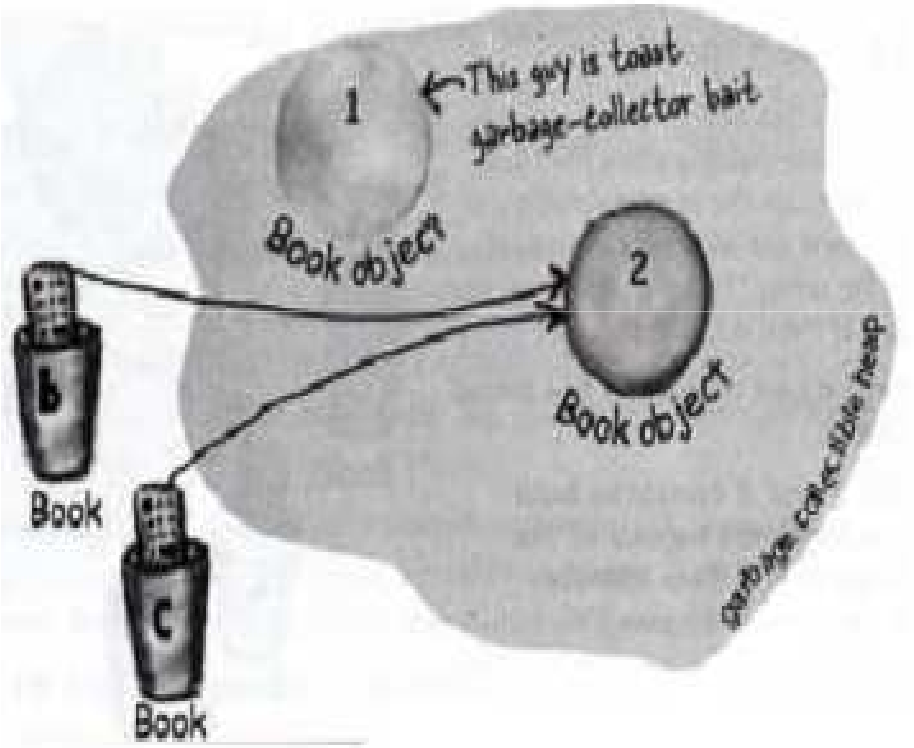
*Both b & c refer to the same object.*

*Object 1 is abandoned & eligible for Garbage Collection (GC).*

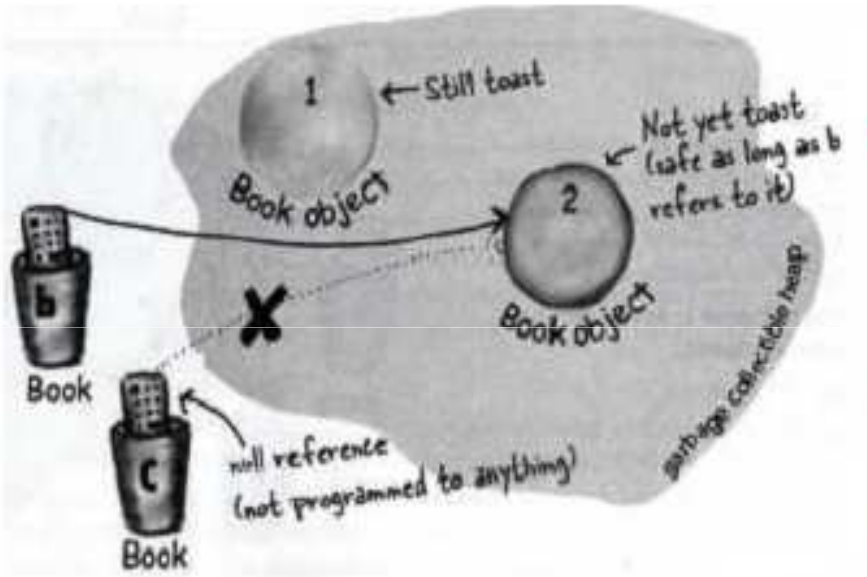
Active References=2

Reachable Objects=1

Abandoned Objects=1



# Ilustrasi kerja Garbage Collector




```
c = null;
```

*Object 2 still has an active reference (b), and as long as it does, the object is not eligible for GC.*

Active References=1

Null References=1

Reachable Objects=1

Abandoned Objects=1

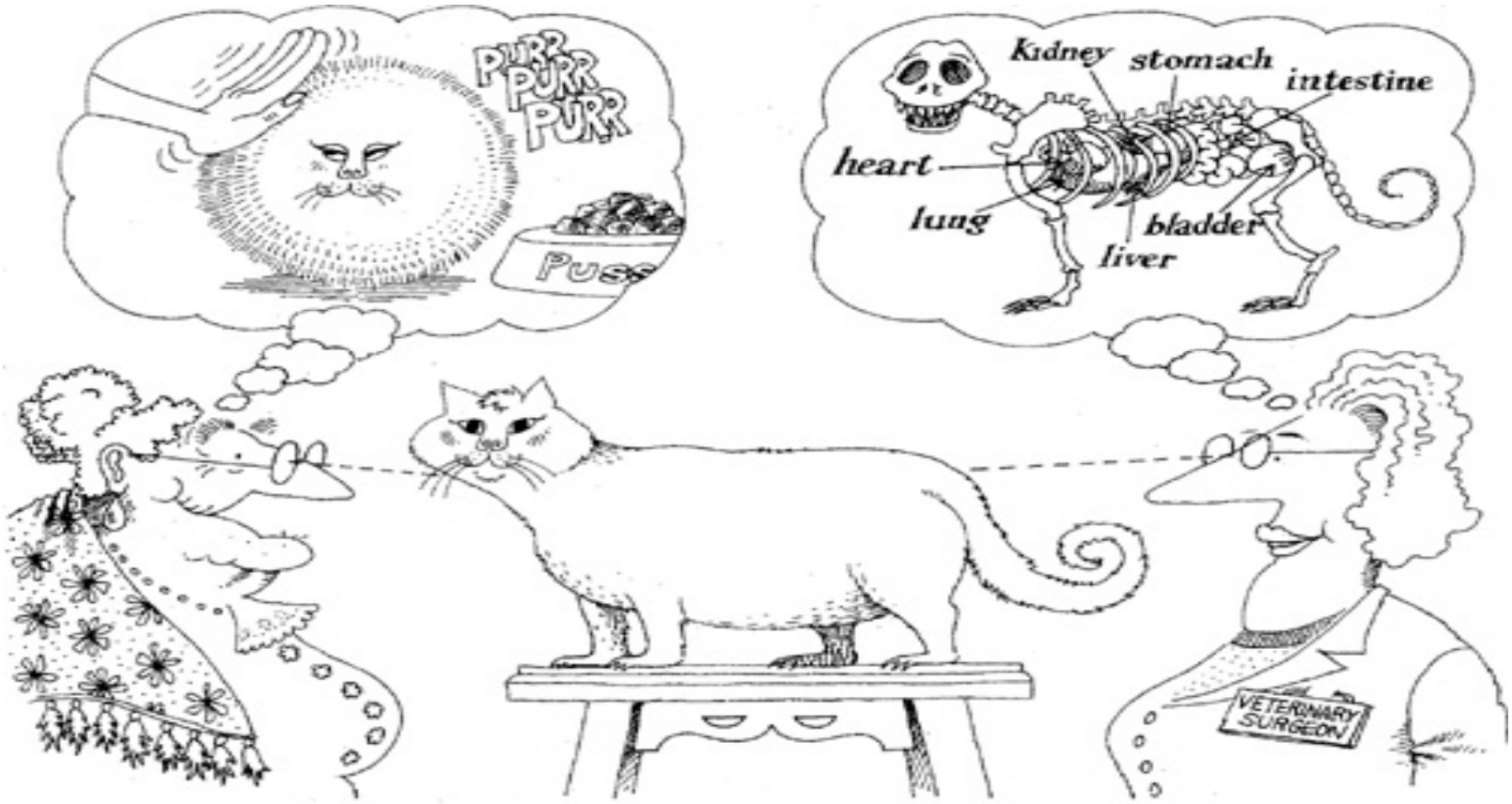
# Konsep Utama OOP

- Abstraction
- Encapsulation and Data Hiding
  - Modularity
- Inheritance / Hierarchy
- Polymorphism

# Abstraksi

- Masalah besar dipecah menjadi masalah masalah kecil
- Fokus pada karakteristik yang esensial dari suatu obyek, relatif pada perspektif yang melihatnya
- Mendefinisikan class yang mampu mengabstraksikan/mendefinisikan suatu obyek dan mampu melakukan kegiatan, mengubah state, dan berkomunikasi dengan obyek lain pada sistem
  - Membuat **class** yg terdiri dari **atribut** dan **method**

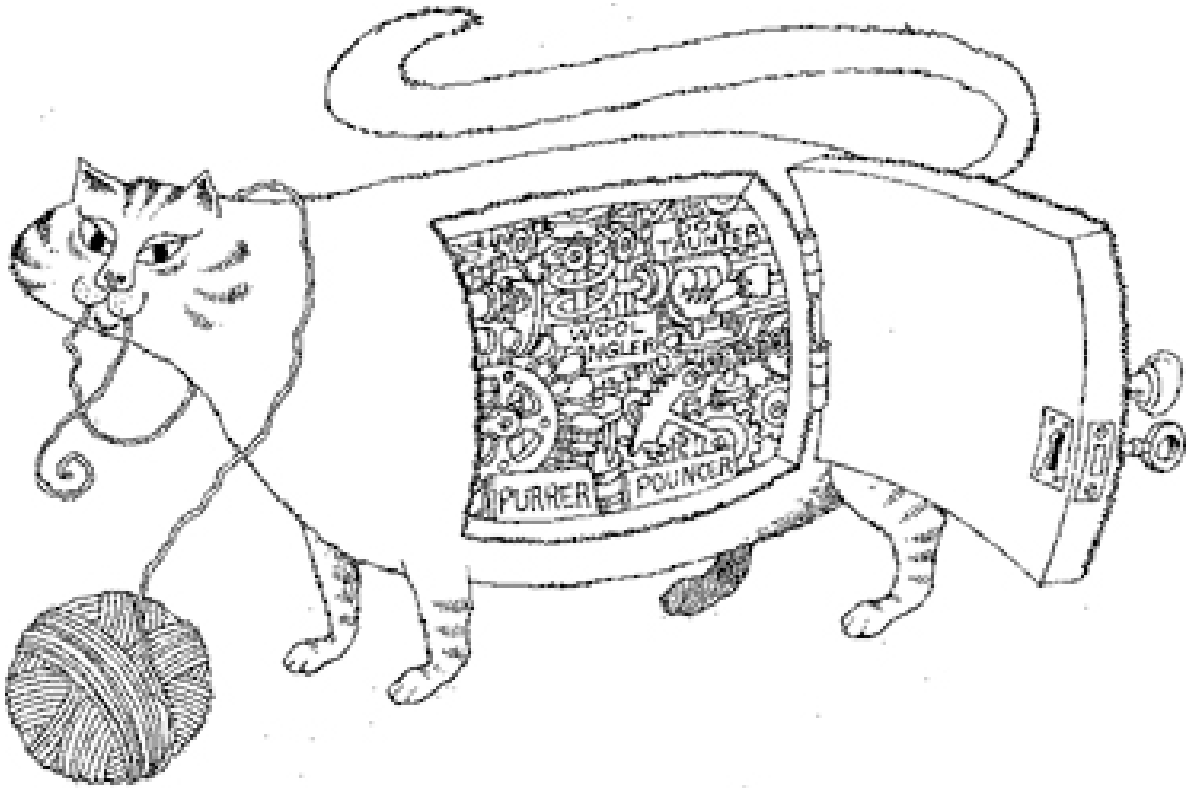
# Prespektif Abstraksi



# Abstraksi

- Pemodelan masalah yang kompleks ke dalam bentuk sederhana, sesuai dengan **tingkatan** yang dibutuhkan
- Contoh nyata: Air Conditioner
- Pengoperasian AC: Turn On, Turn Of, Naikkan suhu, Turunkan suhu, Aktifkan Kipas, Matikan Kipas, Aktifkan timer, Matikan timer, dsb…

# Enkapsulasi



# Encapsulation & Data Hiding

- Object tidak perlu menampilkan seluruh data dan methodnya, hanya yang **dibutuhkan** saja untuk interaksi dengan object lainnya
- Menyembunyikan informasi dan detail implementasi sebuah atribut/method, serta mengatur akses terhadap atribut/method
  - **Hak akses** pada method dan atribut
- Dibuat dalam method-method:
  - **Set** -> mengisi nilai
  - **Get** -> mengambil nilai

# Enkapsulasi vs Abstraksi

- Merupakan dua konsep yang **komplementer** (saling melengkapi)
- Abstraction fokus pada **atribut** dan perilaku (**behavior**) obyek yang dapat dilihat
- Encapsulation fokus pada implementasi (**method**) dari perilaku tersebut beserta perlindungannya (**hak aksesnya**)
- Encapsulation dilakukan dengan cara **information hiding**

# Encapsulation & Information Hiding

Beberapa **keyword akses** untuk encapsulation class

- **Public**: dapat diakses dari semua class
- **Private**: hanya dapat diakses dari dalam class
- **Protected**: hanya dapat diakses dari sub-class
- **Default**: hanya dapat diakses dari class yang berada dalam package yang sama

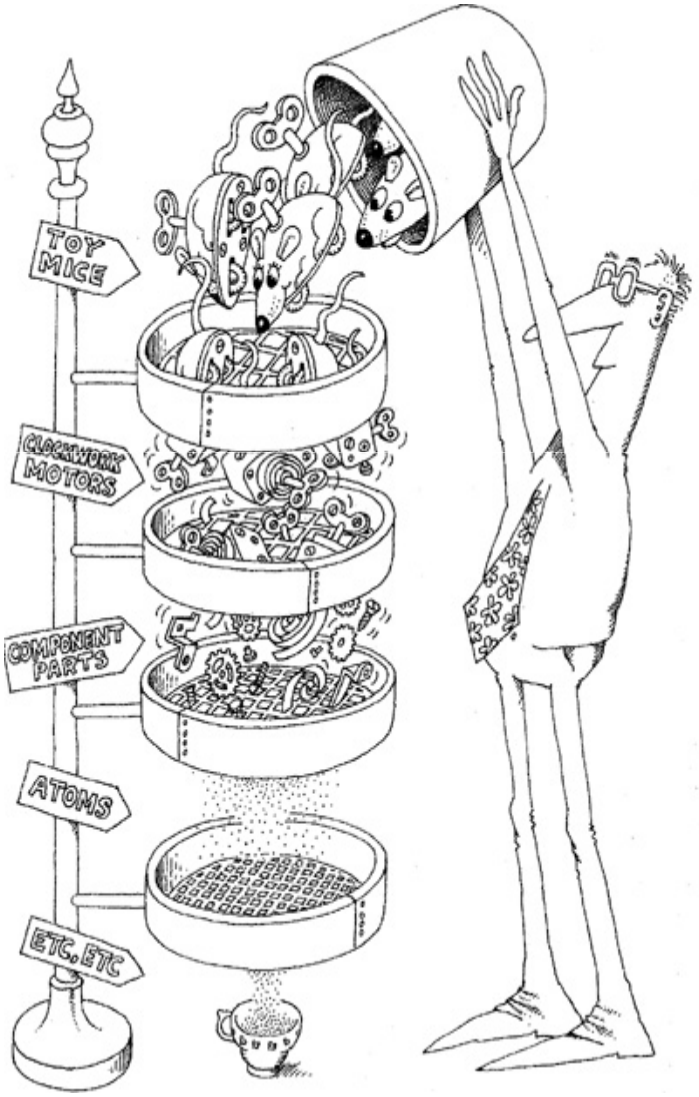
# Encapsulation terdiri dari

- Bagian **Interfaces** (definisi)
  - Dibuat dalam bentuk **method**
  - Mendefinisikan bagaimana membuat instance dari suatu class, dan mendefinisikan operasi-operasi dari object tersebut
  - Sifat : **Public**

- Bagian **Implementation**
  - Sifat : **Private**, disembunyikan,
  - Bisa berupa variabel class

# Modularity

# Inheritance



# Inheritance

- Suatu class dapat **menurunkan** atribut dan method dari class lainnya
- Merupakan **pewarisan** atribut dan method dari class induk ke kelas anak

- Fungsi utama: mengorganisasikan class-class dengan mengumpulkan kesamaan-kesamaan umum / generalisasi (**bottom up**)

# Inheritance

Tujuan Inheritance:

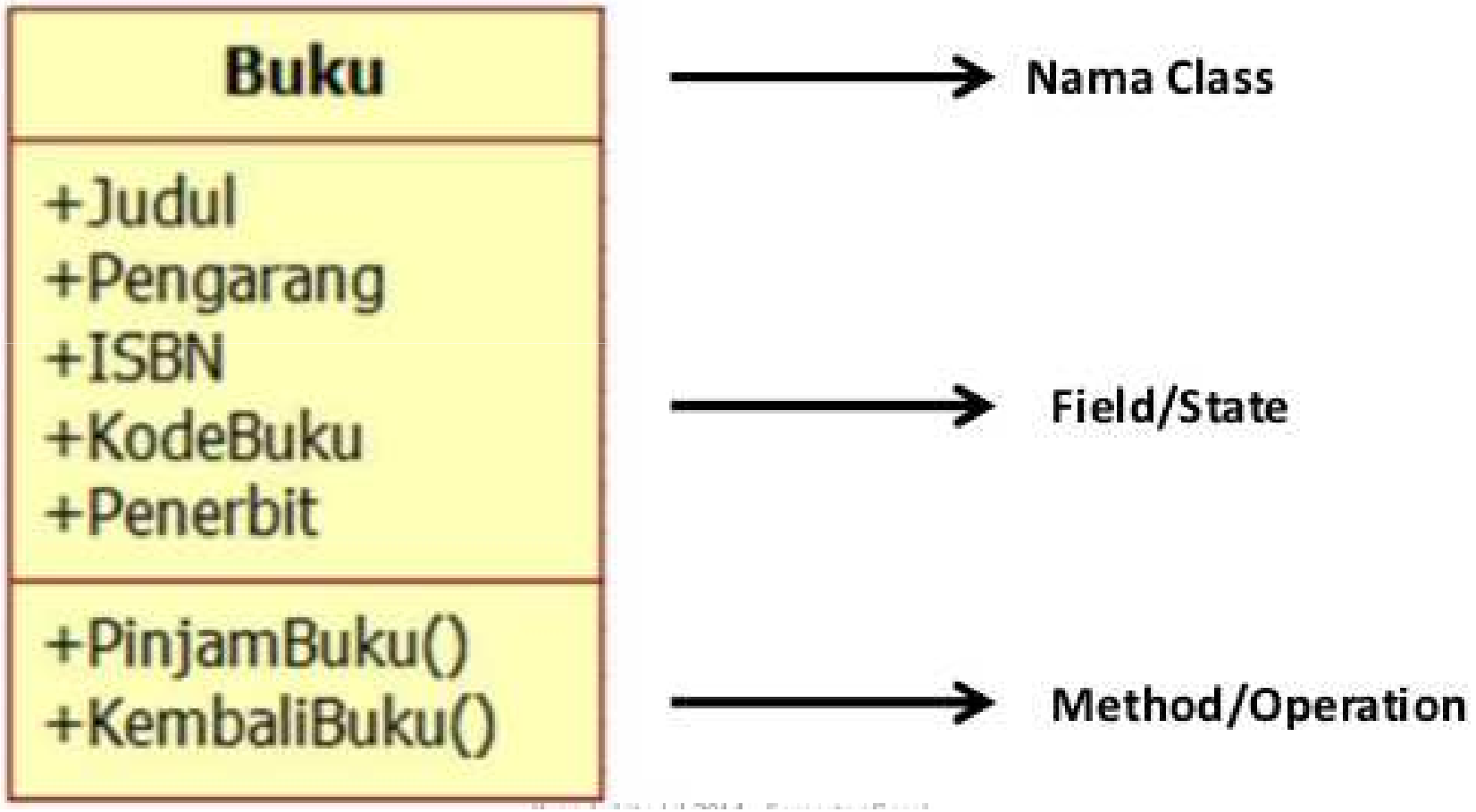
- **Specialization**: object baru memiliki data dan method yang berbeda dari superclassnya (**top down**)
- **Overriding**: method yang diturunkan bisa dideklarasikan ulang pada subclassnya
- **Code Reuse**: method dari superclass bisa langsung digunakan di subclass

# Polymorphism

- Dari bahasa Yunani, artinya "**banyak bentuk**"

- Membuat obyek dari class dasar dapat **berperilaku** seperti obyek lain yang *merupakan turunannya*
- Suatu obyek dari class dapat "**diubah**" menjadi bentuk lain asal merupakan sub classnya

# Class Diagram



# Bentuk umum Class

```
class <Nama_kelas>{
        [<deklarasi_variabel>] //← member variable (1)

        [<konstruktor>] //← constructor (2)

        [<metode>] //← method (3)
}
```

# Contoh Class

```
class Buku{
	String judul, pengarang, isbn, penerbit;

	Buku(){ //ini konstruktor }

	void pinjamBuku(String isbn){
		//isi kode
	}

	void kembalikanBuku(String isbn){
		//isi kode
	}
}
```

# UML

- UML : Unified Modelling Language
- UML untuk perancangan/design
- UML merupakan notasi standar untuk pemodelan pada bahasa pemrograman berorientasi obyek
- UML tidak terikat oleh bahasa pemrograman tertentu

# Jenis UML

- **Use Case Diagram** untuk memodelkan proses bisnis.
- **Conceptual Diagram** untuk memodelkan konsep-konsep yang ada di dalam aplikasi.
- **Sequence Diagram** untuk memodelkan pengiriman pesan (*message*) antar *objects*.
- **Collaboration Diagram** untuk memodelkan interaksi antar *objects*.
- **State Diagram** untuk memodelkan perilaku *objects* di dalam sistem.
- **Activity Diagram** untuk memodelkan perilaku *Use Cases* dan *objects* di dalam *system*.
- **Class Diagram** untuk memodelkan struktur kelas.
- **Object Diagram** untuk memodelkan struktur *object*.
- **Component Diagram** untuk memodelkan komponen *object*.
- **Deployment Diagram** untuk memodelkan distribusi aplikasi.

# Demo class pada Java

# REVIEW: Compile & Run

- Compile: **javac** <namafile.java>
  - Case sensitive
  - Menghasilkan .class
- Run: **java** <namafile>
  - Case sensitive
  - Menghasilkan output
- JAR: **java –jar** <namafile.jar>
  - Menjalankan JAR file

# Next

- Class dan Object (II)