PERTEMUAN 11

KONSEP PEMROGRAMAN BERORIENTASI OBJEK

POKOK BAHASAN

1. Pendahuluan
2. Pengenalan Pemrograman Berorientasi Objek
3. Pengenalan Objek & Class
4. Karakteristik OOP
5. Kelebihan OOP
6. Pemrograman Terstruktur VS Pemrograman Berorientasi Objek

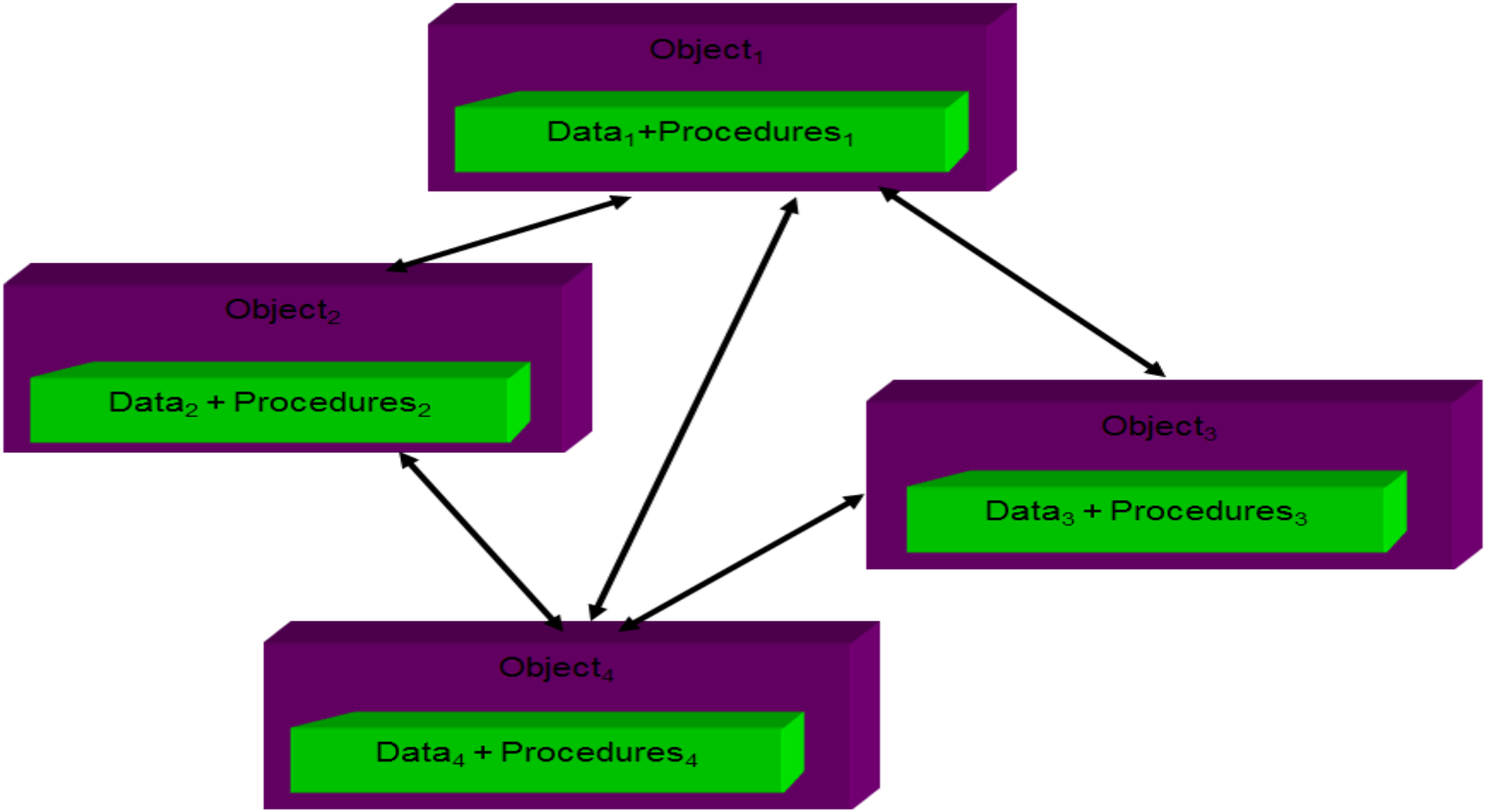
PENDAHULUAN

- Perancangan prosedural memiliki keterbatasan dalam pengembangan sistem yang besar, jaringan dan sistem multi user.
- Meskipun disusun secara terstruktur dan modular, tetap menjadi sangat rumit dan sulit dipahami.
- Kurangnya menyadari pekerjaan yang dilakukan dengan tim pengembang sehingga terjadi duplikasi pada beberapa bagian.
- Teknologi berorientasi objek dapat mengatasi permasalahan di atas serta memberikan fleksibilitas dan ekonomis untuk program sistem jaringan dan multi user

PENGENALAN PEMROGRAMAN BERORIENTASI OBJEK

- Pemrograman berbasis objek (OOP) berdasarkan pada konsep **objek dan interaksinya**.
- Objek dapat menerima pesan *(message)*, mengolah data, dan mengirimkan pesan ke objek lain → membentuk interaksi antar objek.
- Objek bersifat independen: tiap objek dapat dipandang sebagai sebuah entitas mandiri yang memiliki peran atau tanggung jawab tertentu.

PENGENALAN PEMROGRAMAN BERORIENTASI OBJEK (lanjutan)



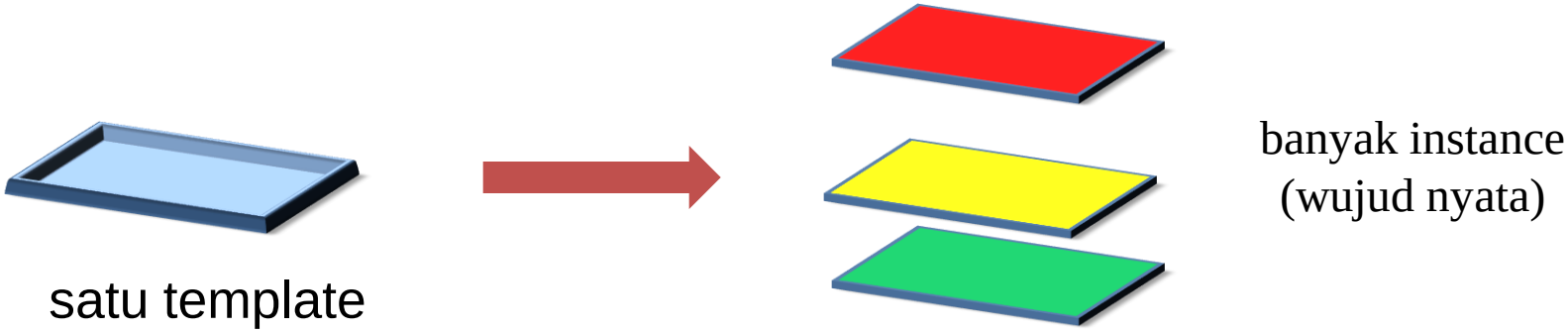
APAKAH OBJEK ?

- **Objek** adalah representasi sebuah entitas yang memiliki makna tertentu yang menjadi perhatian si pemandang.
- Segala sesuatu yang ada di dunia adalah **objek.** Cth : Manusia, Bunga, Hewan, Mobil, Meja, Kursi, Sepeda, Kereta, Pesawat terbang, dll.
- Setiap sistem terdiri dari objek-objek (sistem juga termasuk objek).
- Evaluasi & pengembangan sistem disebabkan oleh interaksi antara objek-objek di dalam atau di luar sistem.

APAKAH KELAS ?

- Merupakan template untuk membuat obyek.

  merupakan prototipe/blue prints yang mendefinisikan variable-variabel dan method –method secara umum.
- Objek (***instances)*** merupakan hasil instansiasi dari suatu kelas, proses pembentukan obyek dari suatu class disebut dengan ***instantiation***.



CONTOH KELAS

Sebuah kelas (Konsep)

Beberapa objek dari kelas yang sama

Sebuah **objek (Realisasi)**

Rekening Bank Udin
Saldo: Rp5.257.000

Rekening Bank Ali
Saldo :Rp13.245.069

Rekening Bank Susi
Saldo : Rp366.891.833

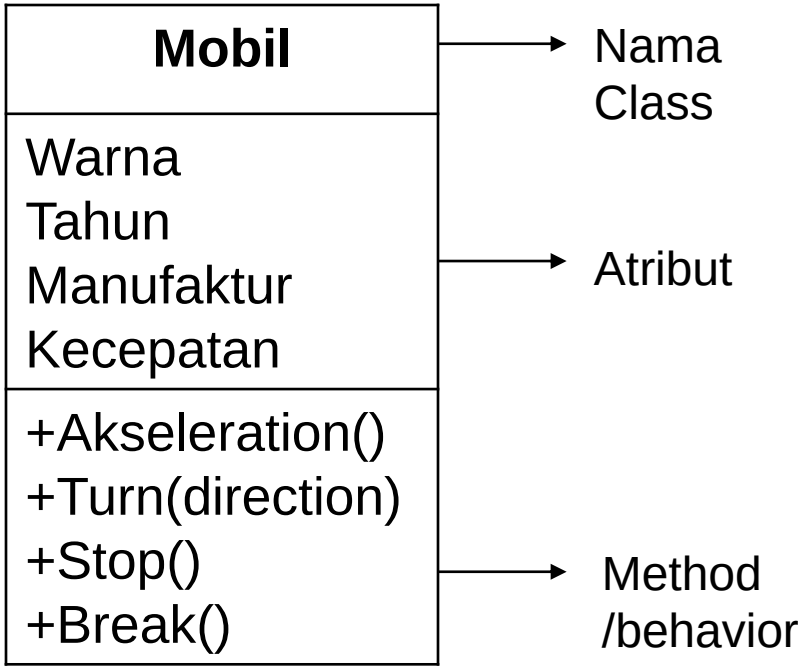
CONTOH KELAS (lanjutan)

Class Mobil

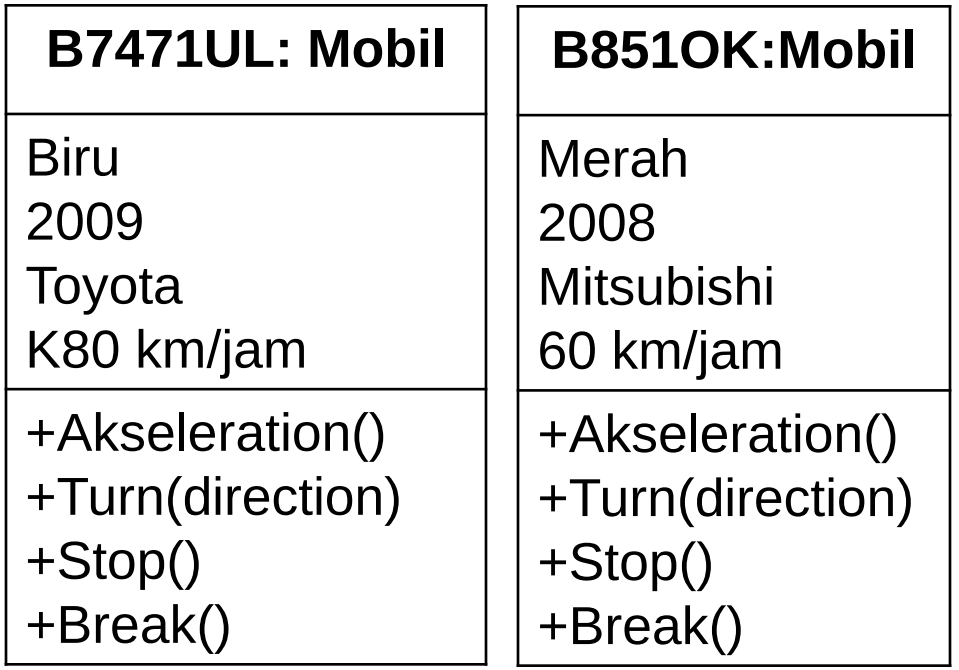
Class Mobil		B7471UL	B851OK
Variabel Instance	Warna	Biru	Merah
	Tahun	2009	2008
	Manufaktur	Toyota	Mitsubishi
	Kecepatan	80 km/jam	60 km/jam
Method Instance	Akseleration() Stop() Break() Turn(direction)		

STRUKTUR KELAS

Contoh kelas mobil



Contoh Objek Mobil



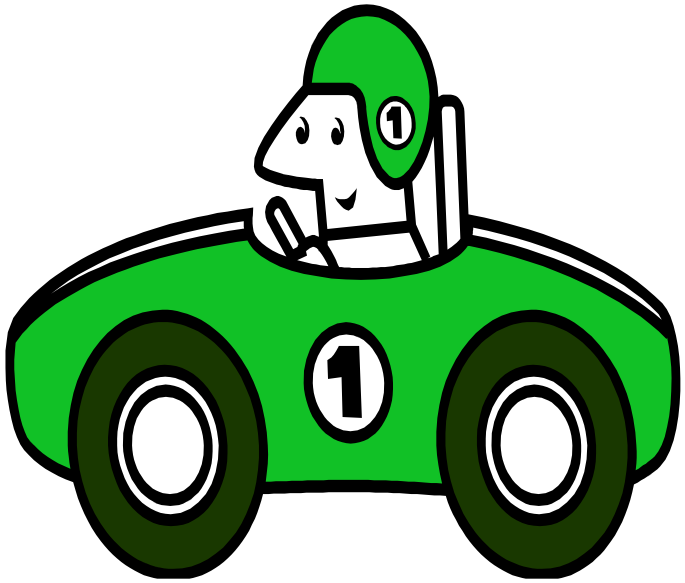
KARAKTERISTIK OBJEK

Attributes :

Warna, Tahun Produksi, Manufaktur, kecepatan

Behavior/Tingkah Laku :

Akseleration(), Turn(direction), Stop(), Break() dll



Identitas :

B7471UL

ATRIBUT

- Atribut adalah data yang membedakan antara objek satu dengan yang lain.
- Contoh atribut mobil : manufaktur, model, warna, jumlah pintu, ukuran engine, kecepatan dll
- Dalam class, atribut disebut sebagai **variabel**.

ATRIBUT (lanjutan)

- Instance variable
  - adalah atribut untuk tiap obyek dari class yang sama.
  - Tiap obyek mempunyai dan menyimpan nilai atributnya sendiri.
  - Jadi tiap obyek dari class yang sama boleh mempunyai nilai yang sama atau beda
- Class variable:
  - adalah atribut untuk semua obyek yang dibuat dari class yang sama.
  - Semua obyek mempunyai nilai atribut yang sama.
  - Jadi semua obyek dari class yang sama mempunyai hanya satu nilai yang value nya sama.

TINGKAH LAKU

- Tingkah laku/behavior adalah hal-hal yang bisa dilakukan oleh objek dari suatu class.
- Behavior dapat digunakan untuk mengubah nilai atribut suatu objek, menerima informasi dari objek lain, dan mengirim informasi ke obyek lain untuk melakukan suatu task.
- Dalam class, behavior disebut juga sebagai **methods**.
- Contoh: mobil

  akseleration

  stop

  turn

  break

TINGKAH LAKU (lanjutan)

- Method adalah serangkaian statements dalam suatu class yang menghandle suatu task tertentu.
- Cara objek berkomunikasi dengan objek lain adalah dengan menggunakan method.

RESPONSIBILITY

- Responsibility adalah sebuah kontrak atau sebuah obligasi dari sebuah class.
- Saat sebuah class dibuat, semua objek dalam class tersebut memiliki keadaan dan tingkah laku yang sama.
- Saat sebuah class dimodelkan, awal yang baik adalah menspesifikasikan responsibilities sesuatu dalam sebuah kamus.
- Teknik seperti CRC card (Class Responsibility Collaboration) dan use case dapat membantu.

KONSTRUKTOR

- Konstruktor adalah proses instansiasi objek dari kelas dilakukan pada operasi khusus atau sekumpulan instruksi.
- Konstruktor menetapkan nilai awal untuk atribut objek baru.
- Konstruktor biasanya memiliki nama yang sama dengan kelasnya.
- Pseudocode untuk instansiasi objek baru

Create nama-objek as new nama-Class()

  - Contoh :

Create mobil as new Mobil()

KONSTRUKTOR (lanjutan)

- Kata NEW menunjukkan pembuatan objek baru.
- Nama kelas diawali dengan huruf besar. Contoh : Mobil, Siswa, Pendaftaran, dll.
- Nama objek diawali dengan huruf kecil. Contoh : mobil, siswa, pendaftaran, dll
- Konstruktor memungkinkan :
  - Memiliki parameter yang menginisilisasi atribut dengan nilai spesifik. Contoh : Create mobil as new Car (”Ford”, ”Falcon”, 4, 300, 6, ”Biru”, 0); atau
  - Tidak memiliki parameter. Objek baru dengan nilai default untuk semua atributnya.

AKSESOR & MUTATOR

Nilai variabel dari objek tersedia untuk semua operasi di dalam objek tersebut, tetapi tersembunyi dari objek luar. Untuk keamanan, operasi publik dikenal istilah aksesor dan mutator, yang membolehkan objek luar untuk mengakses nilai pada atribut.

- Aksesor adalah nama operasi yang mengakses nilai. Nama aksesor dimulai dengan kata GET seperti getPaySlip().
- Mutator adalah nama operasi yang merubah nilai atribut. Operasi mutator memungkinkan objek eksternal untuk mengubah nilai yang tersimpan dalam atribut. Nama mutator diawali dengan kata SET seperti setPayRate().

VISIBILITY

- Visibility merupakan kemampuan suatu obyek untuk melihat atau berhubungan dengan obyek lain.
- Atribut dan metoda dapat memiliki salah satu sifat visibility berikut :
  - *Private* (-), tidak dapat dipanggil dari luar *class* yang bersangkutan
  - *Protected* (#), hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya
  - *Public* (+), dapat dipanggil oleh siapa saja

PENGIRIMAN PESAN/MESSAGING

- Objek-objek bekerjasama dengan mengirimkan pesan dari satu objek ke objek lainnya.
- Suatu obyek mengirimkan pesan ke objek lain untuk melakukan sebuah operasi.
- Suatu objek juga dapat menerima pesan dari objek lain untuk melakukan operasi lainnya.
- Kunci dalam pemrograman berorientasi objek adalah bahwa setiap objek itu sendiri bertanggung jawab untuk melaksanakan tugas.

PENGIRIMAN PESAN/MESSAGING (lanjutan)

- Ini termasuk interaksi dan komunikasi dengan benda-benda lainnya.
- Objek mengirim pesan ke objek lainnya. Pesan mungkin menyampaikan informasi tambahan melalui parameter untuk benda-benda penerima.

Contoh Messaging

- Mobil yang diparkir di depan rumah hanya sepotong logam yang dengan sendirinya tidak mampu melakukan aktivitas apapun. Pengemudi harus menyalakan mobil, menggunakan rem, dll
- Objek "pengetik" dapat mengirim pesan "mengubah ukuran (20)" ke objek huruf untuk mengubah ukuran font.

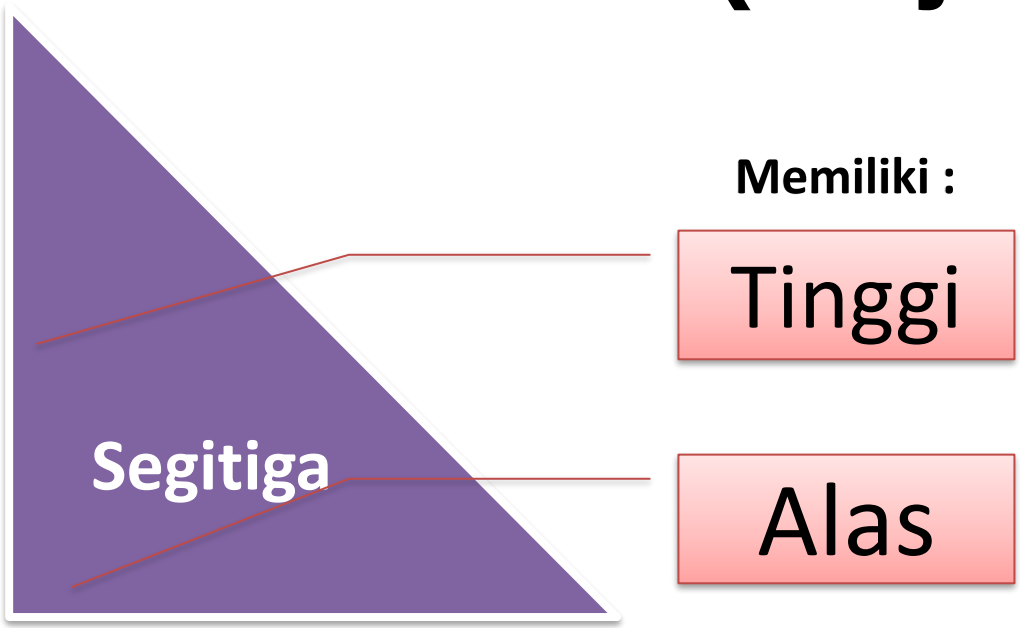
KARAKTERISTIK OOP

- Abstraksi
- Enkapsulasi
- Inheritansi
- Polimorfisme

ABSTRAKSI

- Abstraksi adalah proses penyembunyian kerumitan dari suatu proses untuk permasalahan yang dihadapi.
- Contoh : Orang hanya perlu berpikir bahwa mobil adalah sebuah objek yang telah memiliki prilaku spesifik, yang dapat digunakan sebagai alat transportasi, sehingga dia/mereka tinggal menggunakannya atau mengendarainya tanpa harus mengetahui kerumitan proses yang terdapat di dalam mobil tersebut.

ABSTRAKSI (lanjutan)



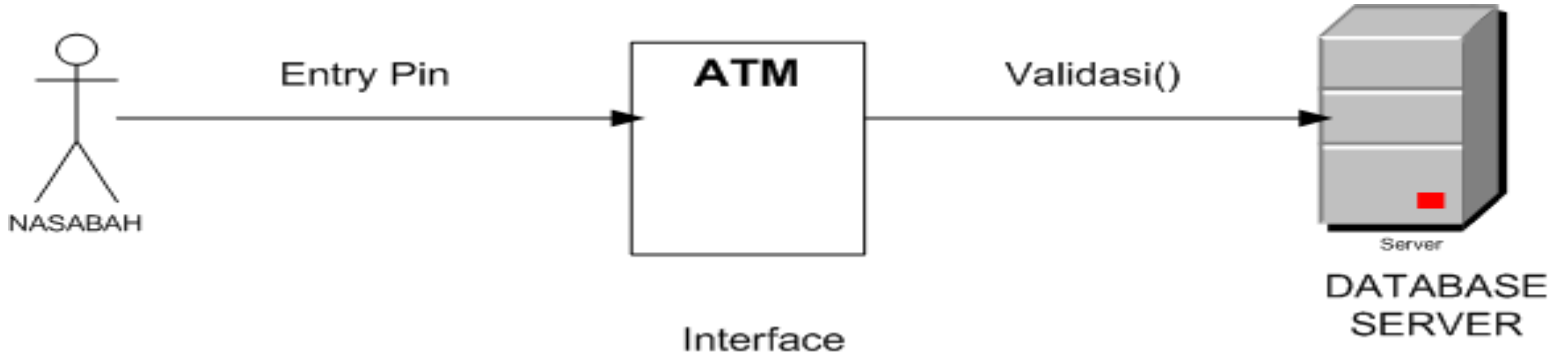
Apa yang anda ingin segitiga lakukan ?

- Hitung Luas
- Hitung Keliling

ENKAPSULASI

Enkapsulasi atau pembungkusan berfungsi untuk melindungi suatu objek dari dunia luar, sehingga seseorang tidak akan mampu merusak objek yang terbungkus. Objek yang terbungkus dalam suatu kelas baik data maupun fungsinya tidak bisa terlihat apalagi dirubah pada saat objek digunakan.

CONTOH ENKAPSULASI

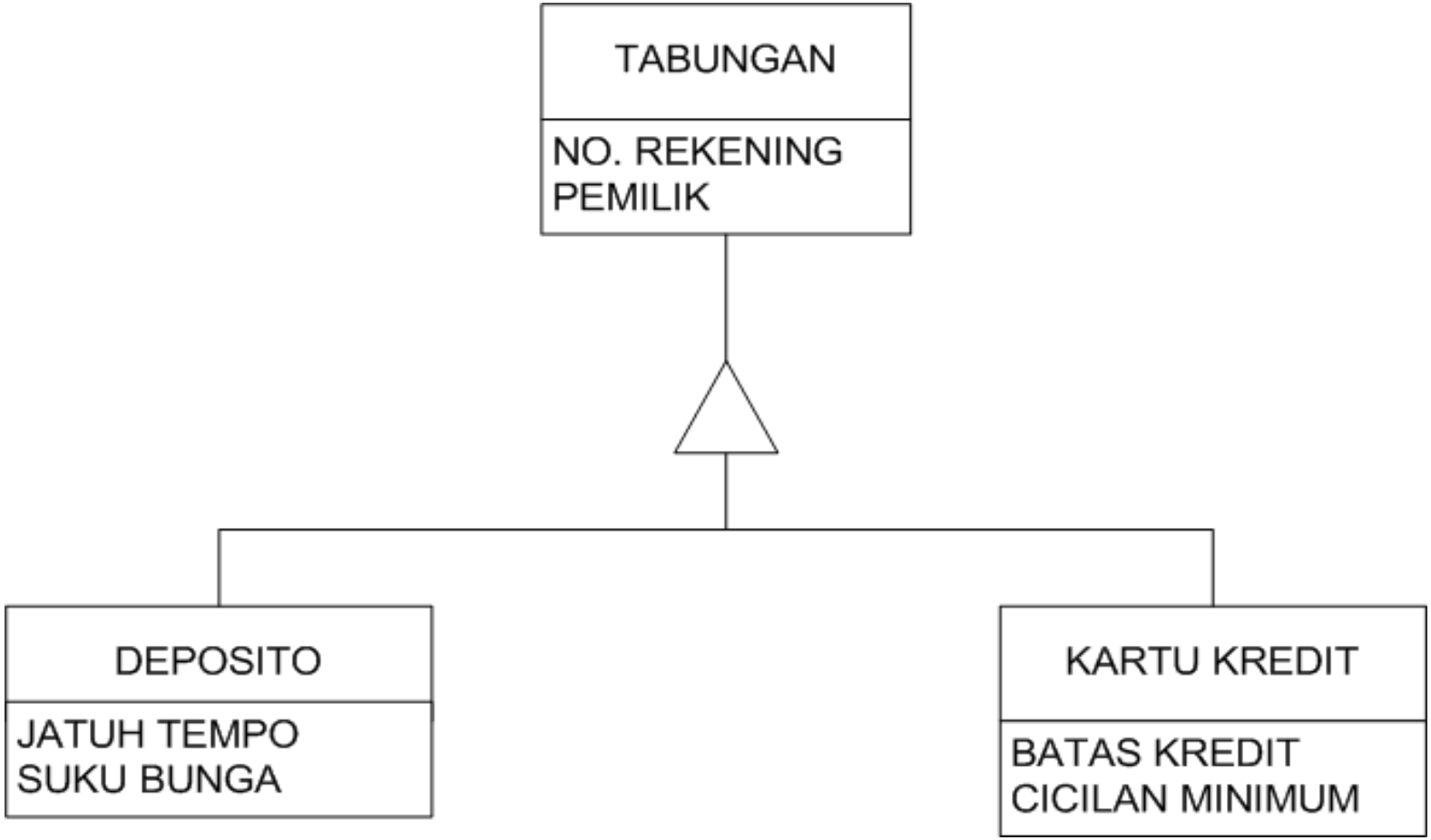


Disini terjadi penyembunyian informasi tentang bagaimana cara kerja pengecekan validitas kartu, kecocokan pin yang dimasukkan, koneksi ke database server, dll, dimana hal-hal tersebut tidak perlu diketahui oleh pengguna tentang bagaimana cara kerjanya.

INHERITANSI

Kelas dapat menurunkan metode-metode dan properti-properti yang dimilikinya pada kelas lain. Kelas yang mewarisi metode dan properti dari objek lain dinamakan kelas turunan. Kelas turunan ini mampu mengembangkan metode sendiri.

CONTOH INHERITANSI



POLIMORFISME

- Polimorfisme dapat diartikan sebagai kemampuan suatu bahasa pemrograman untuk memiliki fungsi-fungsi atau metode yang bernama sama tetapi berbeda dalam parameter dan implementasi kodenya (*overloading*).
- Kelas turunan dapat menggunakan fungsi yang ada pada kelas pewarisnya dan dapat mengimplementasikan kode yang berbeda dari fungsi pewarisnya ini dinamakan *overriding*.

EMPLOYEE

Has

- NIP
- Nama
- Jabatan

Things to do

- createNew
- getSalary

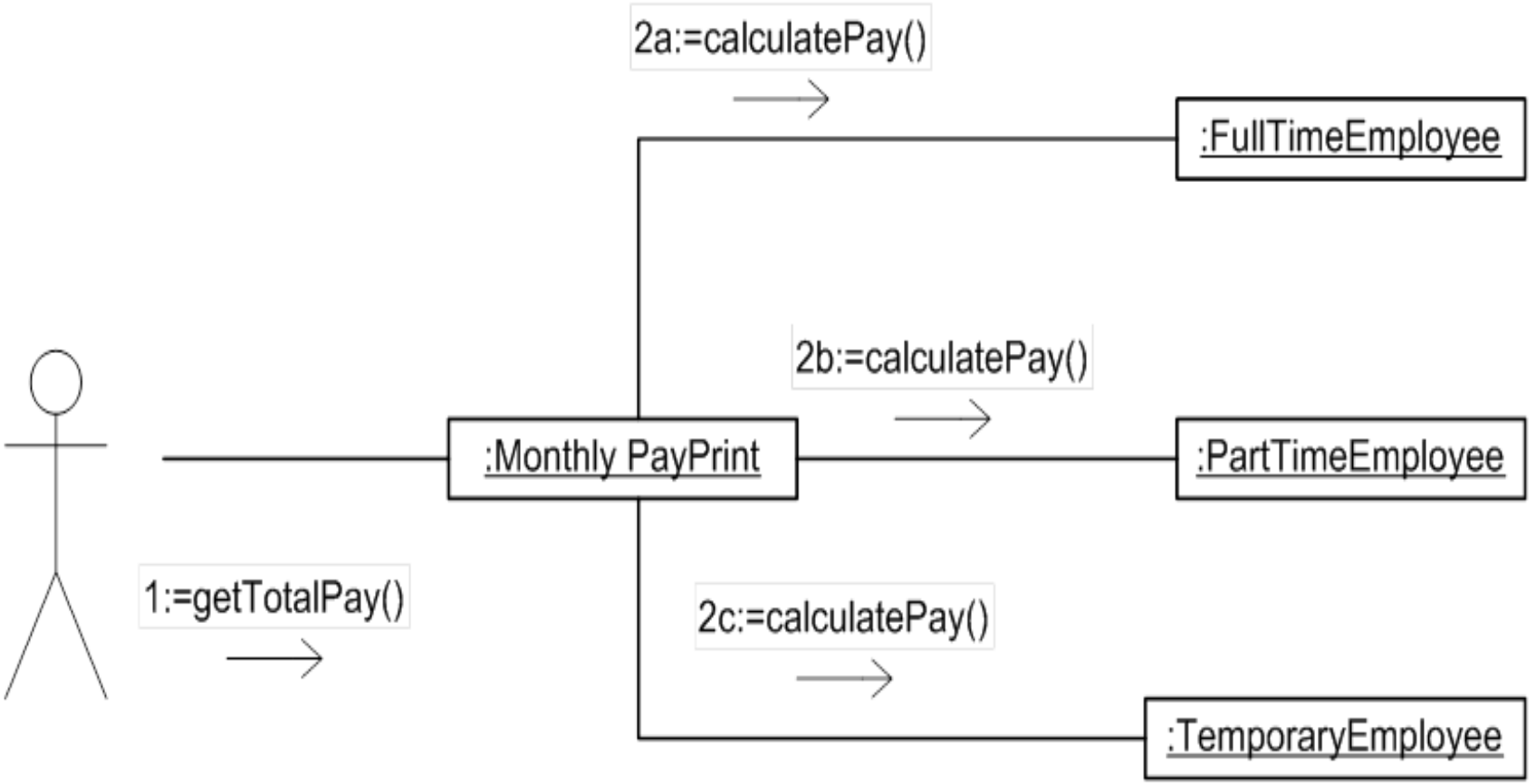
Apakah Mereka Memiliki Gaji Yang Sama

Tetap

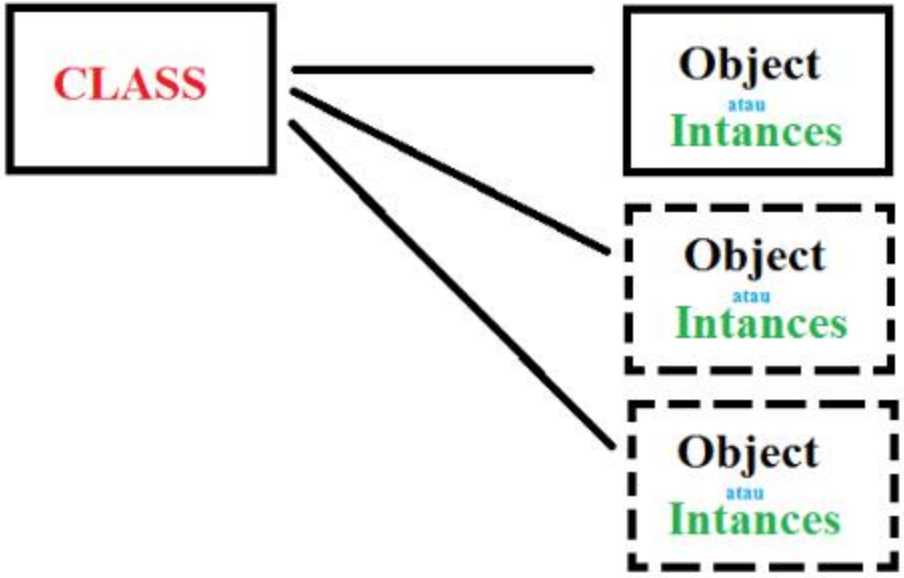
Paruh Waktu

Outsourcing

CONTOH POLIMORFISME

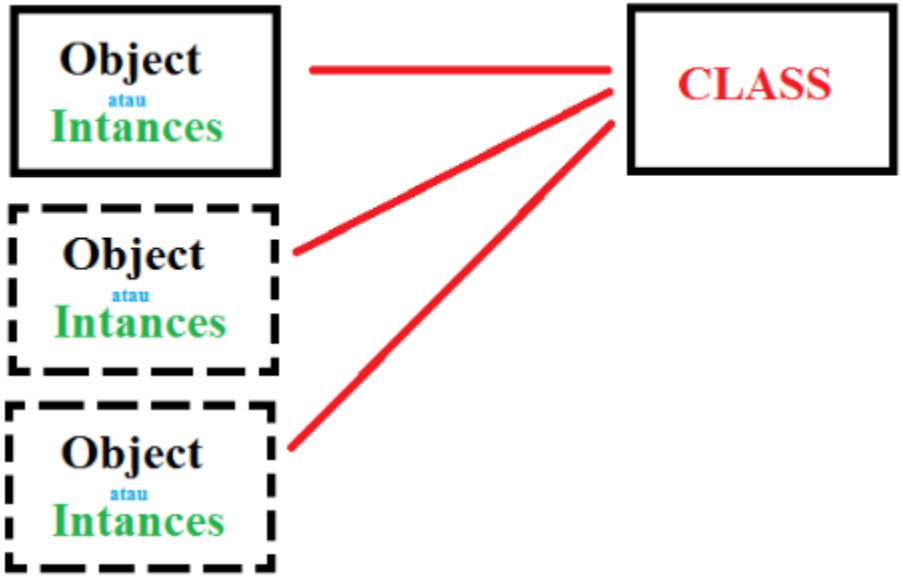


Resume



Class dapat menghasilkan object atau object-object (Instances);

Class = Prototype dari Object atau instances



Kelompok Object –object atau Instances sejenis dapat menjadi Class

Membentuk Objek Baru

Java

```
MyClass myObj1 = new MyClass();
```

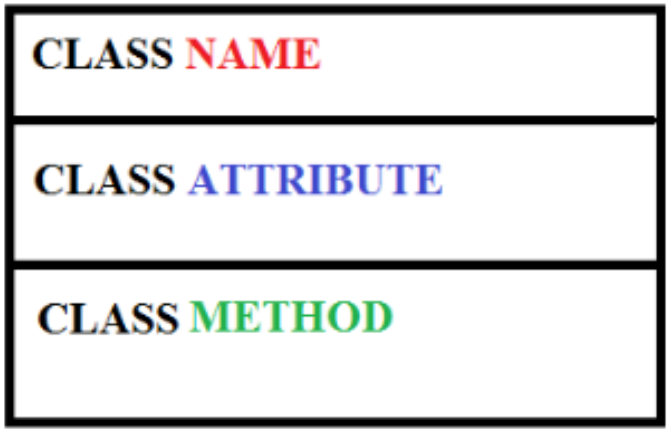
PHP

```
$MyObj1 = new MyClass();
```

C++

```
MyClass myObj1;
```

myObj1 adalah object atau instance baru



Struktur Class terdiri dari 3 bagian

1. Class Name
2. Class Attribut

   Memiliki visibility : **(-)** Private; **(+)** Public; **(#)** Protected
3. Class Method

   Dapat dibentuk dengan operasi Aksesor (get) dan Mutator (set) dari attribut yang ada, dalam java disebut POJO (Plain Old Java Object)

Tetapi tidak semua class harus memiliki 3 bagian, bisa jadi hanya berisi **Class Name dan Attribut** saja atau **Class Name dan Method** saja

POLYMORPHIMS

OVERLOADING

Method Overloading adalah sebuah kemampuan yang membolehkan sebuah class mempunyai 2 atau lebih method dengan nama yang sama, yang membedakan adalah parameternya.

Pada method overloading perbedaan parameter mencakup :

- Jumlah parameter
- Tipe data dari parameter
- Urutan dari tipe data parameter

Method Overloading juga dikenal dengan sebutan Static Polymorphism. Berikut ini contoh Class yang melakukan Overloading.

OVERRIDING

Method overriding merupakan method yang parrent class yang ditulis kembali oleh subclass. Aturan dari method overriding pada Java :

- Parameter yang terdapat pada method overriding di subclass harus sama dengan parameter yang terdapat pada parent class.
- Aturan hak akses (visibility), hak akses method overriding di subclass tidak boleh lebih ketat di bandingkan dengan hak akses method pada parent class.

```
public class ContohOverloading {
public void jumlah (int a, int b){ System.out.println("Jumlah 2
angka ="+ (a + b));
}

//oveloading perbedaan jumlah parameter
public void jumlah (int a, int b, int c){
System.out.println("Jumlah 3 angka =" + (a + b + c));
}

//overloading perbedaan tipe data parameter
public void jumlah(double a, int b){
 System.out.println("Jumlah 2 angka (double+int) = "+ (a + b));
}
//overloading perbedaan urutan tipe data parameter public void
jumlah (int b, double a){
System.out.println("Jumlah 2 angka (int+double) = "+ (a + b));
}
}
```

Contoh overloading di main program, co adalah object baru dari bentuk class ContohOverloading yang sudah ada

```
public class PenggunaanOverloading {
    public static void main(String[] args) {
     ContohOverloading co  = new ContohOverloading();
              co.jumlah(83,32);
              co.jumlah(34,454,432);
              co.jumlah(34.43,34);
              co.jumlah(28,33.23);
    }
}
```

OVERRIDING

Extends = Turunan; anak (MAMALIA) memiliki sifat parent (MAMALIA)


```
public class Binatang {
  public void bergerak(){
      System.out.println("Binatang bergerak sesuai kemampuannya");
   }

   public void berkembangBiak(){
      System.out.println("Binatang berkembang biak sesuai kemampuannya");
   }
}

public class Mamalia extends Binatang {
//overriding method parent class
public void bergerak(){
      System.out.println("Mamalia bergerak sebagian besar dengan kakinya");
}
public void berlari(){
      System.out.println("Sebagian Mamalia dapat berlari");    }
}
```

```
public class PenggunaanOverriding {
public static void main(String[] args) {
 // TODO Auto-generated method stub
        Binatang b = new Binatang();
        Mamalia m = new Mamalia();
        Binatang bm = new Mamalia();

         b.bergerak();      // Binatang bergerak sesuai kemampuannya
         m.bergerak();      // Mamalia bergerak sebagian besar dengan kakinya
         bm.bergerak();     // Binatang bergerak sesuai kemampuannya + Mamalia bergerak sebagian besar dengan kakinya
          bm.berkembangBiak(); // Binatang berkembang biak sesuai kemampuannya
 }
}
```

bm override object dimana object tersebut memiliki sifat dari parent object
Child object = mamalia
Parent object = binatang

TAHAPAN PERANCANGAN PROGRAM BERORIENTASI OBJEK

1. Identifikasi kelas, atribut, responsibility dan operasi
2. Menentukan hubungan antar objek dan kelas
3. Perancangan algoritma untuk operasi menggunakan desain struktur
4. Mengembangkan tes algoritma

KELEBIHAN PEMROGRAMAN BERORIENTASI OBJEK

- Menyediakan struktur modular yang jelas untuk program sehingga bagus digunakan untuk mendefinisikan tipe data abstrak di mana detil implementasinya tersembunyi.
- Mempermudah dalam memelihara dan memodifikasi kode yang sudah ada. Objek yang baru dapat dibuat tanpa mengubah kode yang sudah ada.
- Menyediakan *framework* untuk *library* kode di mana komponen software yang tersedia dapat dengan mudah diadaptasi dan dimodifikasi oleh programmer. Hal ini sangat berguna untuk mengembangkan GUI.
- Resiko kesalahan relative kecil (lebih sedikit mengetik), sintaks juga tidak perlu dihafalkan, karena semuanya sudah disediakan.
- Waktu *debugging* lebih singkat, karena setiap objek tidak perlu di-*debug* setiap kali digunakan

PEMROGRAMAN TERSTRUKTUR VS PEMROGRAMAN BERORIENTASI OBJEK

Pemrograman Terstruktur	OOP
a. Penekanan pada urutan yang harus dikerjakan (algoritma pemecahan masalah)	a. Pendekatan lebih pada data bukanya pada fungsi/prosedur
b. Program berukuran besar dipecah-pecah menjadi program-program yang lebih kecil (Modular)	b. Program besar dibagi ke dalam Objek-objek. Struktur data dirancang dan menjadi karakteristik dari objek-objek
c. Kebanyakan fungsi/prosedur berbagi data global	c. Fungsi-fungsi yang mengoperasikan data tergabung dalam suatu objek yang sama

PEMROGRAMAN TERSTRUKTUR VS PEMROGRAMAN BERORIENTASI OBJEK (lanjutan)

Pemrograman Terstruktur	PBO
d. Data bergerak secara bebas dalam sistem, dari satu fungsi ke fungsi yang lain saling terkait	d. Data tersembunyi dan terlindung dari fungsi/prosedur yang ada di luar
e. Fungsi-fungsi mentransformasi data dari satu bentuk ke bentuk yang lain	e. Objek-objek dapat saling berkomunikasi dengan saling mengirim message satu sama lain
f. Pendekatan adalah pendekatan *top down* (dari atas ke bawah)	f. Pendekatanya adalah *bottom up* (dari bawah ke atas)

SOAL LATIHAN

1. Merupakan template untuk membuat obyek .merupakan prototype/blue prints yang mendefinisikan variable-variabel dan method – method secara umum, merupkan pengertian dari?

A. Sistem
B. Kelas
C. Konsep
D. Objek
E. Atribut

2. Serangkaian statements dalam suatu class yang menghandle suatu task tertentu merupakan pengertian dari?

A. Method

B. Kelas

C. Responsibility

D. Konstruktor

E. Atribut

3. Berikut ini merupakan contoh behavior mobil adalah?

A. Stop, model, warna

B. Turn, stop, break kecepatan

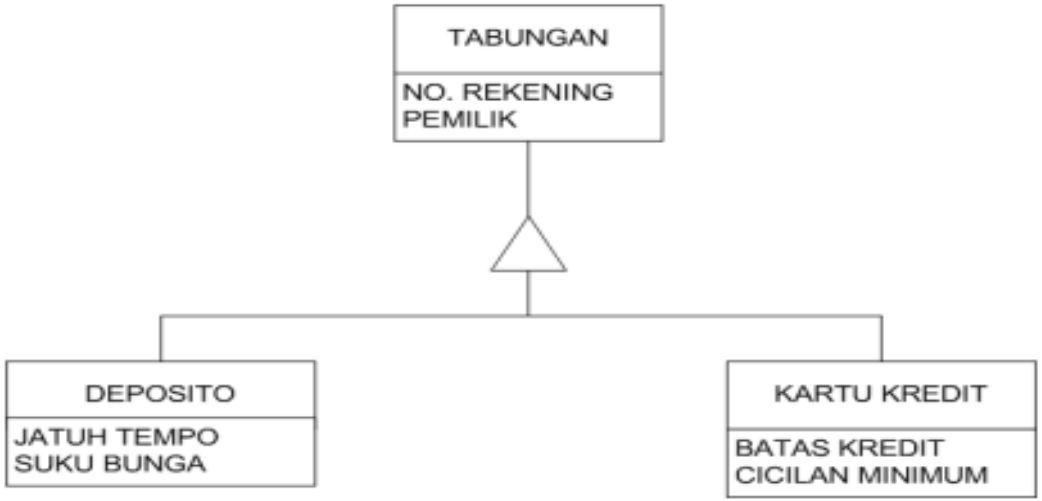
C. Akseleration, manufaktur, model

D. Jumlah pintu,model, kecepatan

E. Akseleration,stop,turn,break

4. Proses penyembunyian kerumitan dari suatu proses untuk permasalahan yang dihadapi adalah?

A. Method
B. Abstraksi
C. Inheritasi
D. Polimorfisme
E. enkapsulasi



5. Bagan di atas merupakan contoh dari?

A. Abstraksi

B. Polimorfisme

C. Enkapsulasi

D. Inheritansi

E. employee

