Konsep PBO

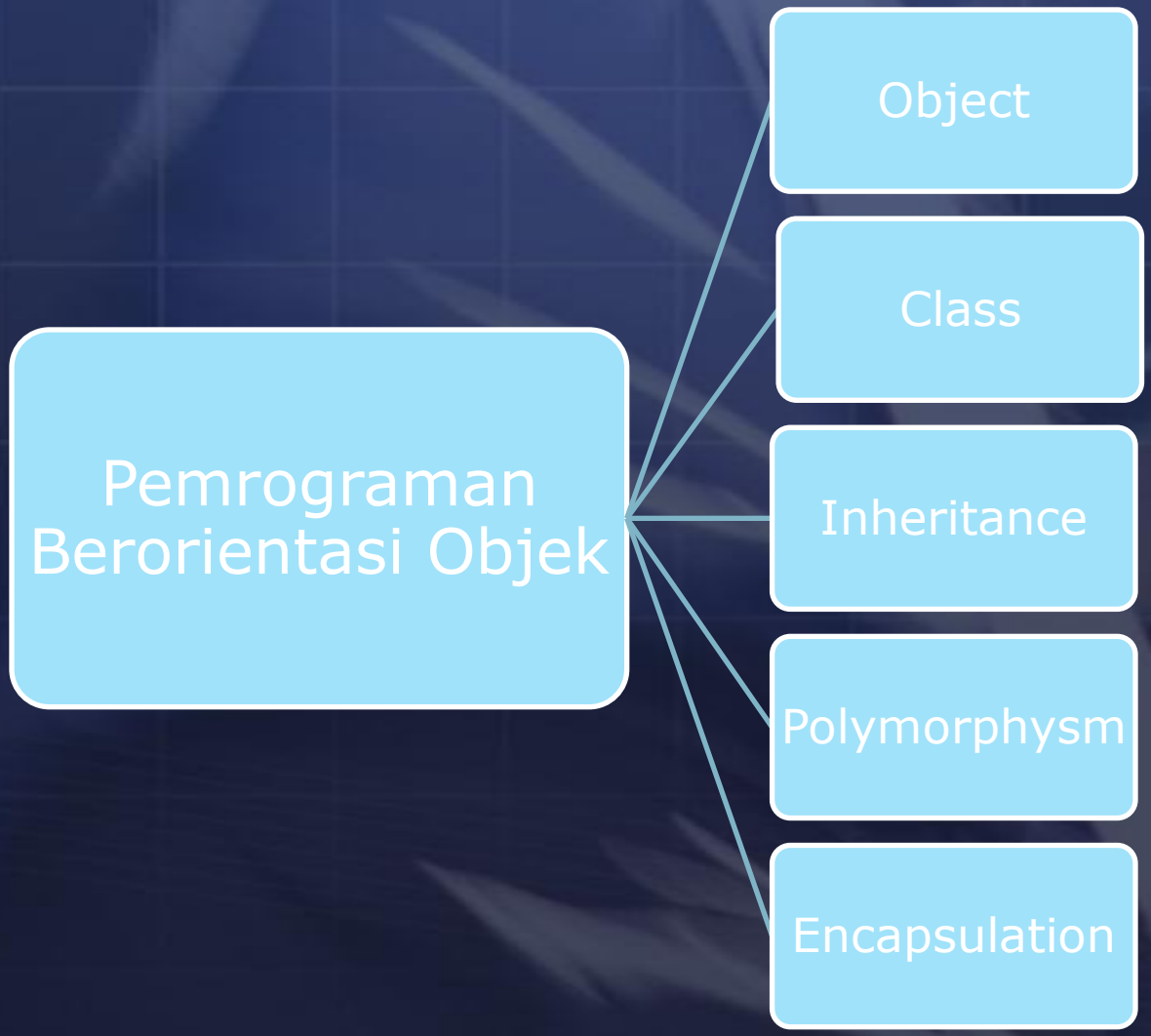
Pemrograman Berorientasi Objek
Dalam Java

Abstraksi

- Pemrograman prosedural
  - Pemrograman yang berorientasi kerja, ct : mengolah data, mencetak dokumen.
- Pemrograman objek
  - Berorientasi benda, ct : mahasiswa, dosen, karyawan. Selanjutnya kerja dilakukan oleh objek-objek tersebut.
  - Misal : dosen mengajar, mahasiswa ujian, karyawan kerja.

- Dalam pemrograman prosedural, yang menjadi orientasi adalah bagaimana melakukan KERJA
- Seperti mengubah data *input*, supaya mendapatkan *output*an.
- Sehingga kita memecah program menjadi beberapa bagian berupa perintah-perintah yang lebih kecil.

- Sedangkan dalam PBO, yang diutamakan adalaH berbasis OBJEK.
- Kita memodelkan sistem menjadi objek-objek.
- Selanjutnya, hubungan antar objek satu dengan objek yang lain.
- Ct : hubungan antara dosen dan mahasiswa. Dosen mengajar mahasiswa, dan mahasiswa memilih dosen untuk kuliahnya.


Pemrograman Berorientasi Objek
Object
Class
Inheritance
Polymorphysm
Encapsulation

OBJEK

- Dua karakteristik objek :
  - Keadaan (state)
    - Untuk menyimpan informasi objek
    - Disebut juga dengan atribut / field.
    - Misal : kucing sebagai objek, bisa mempunyai keadaan. Yaitu lapar/kenyang, nilai berat badan, warna bulu, dll
  - Tingkah laku (behavior)
    - Untuk menentukan kerja apa saja yang dapat dilakukan objek
    - Disebut juga metode (method)
    - Metode objek, misal makan, tidur, dan berak.

ATRIBUT / FIELD

TINGKAH LAKU / METHOD

Warna kulit : Biru Keputih-putihan
Warna Hidung :merah
Menggunakan kalung lonceng

Tidur

Class (Kelas)

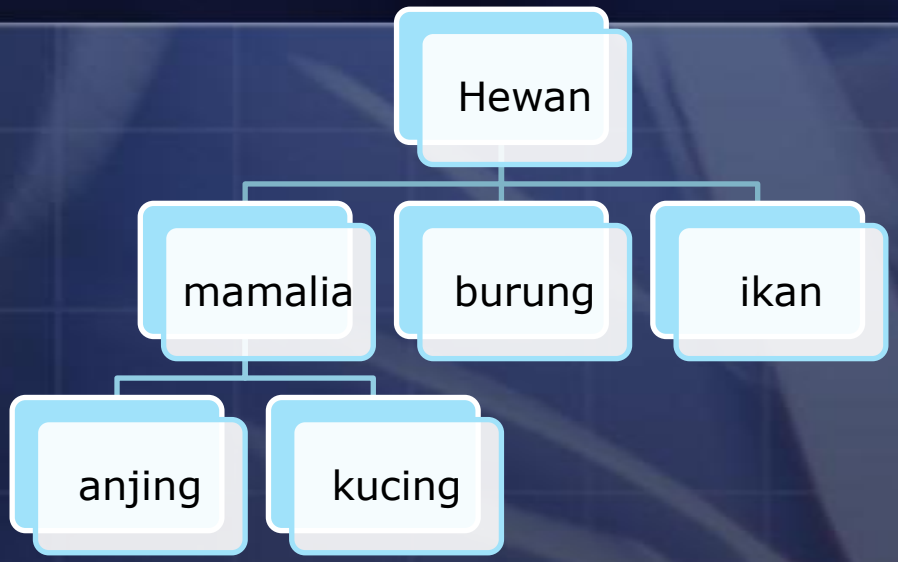
- Setiap program dalam java paling tidak mempunyai satu buah class.
- Class bisa dikatakan sebagai cetak biru atau template objek.
- Class bukanlah objek real, namun konsep objek.
  - Contohnya, dalam game yang anda buat, terdapat dua ekor kucing, bernama *Anggora* dan *Persia*.
  - Maka kucing adalah **class**, sedang Anggora dan Persia adalah **objek** dari tipe kucing.

Pendefinisian (Class)

DemoKucing.java

- Class : Kucing
- Field : nama, berat badan.
  - Dalam deklarasi class, nama dan berat badan kucing tidak ditentukan, karena setiap Kucing bisa mempunyai nama dan berat yang berbeda2.
  - Field akan ditentukan ketika membuat objek kucing. Misal : dibuat objek kucing *bernama* Anggora dg *berat* 5 kg, dan objek kucing lain *bernama* Persia dg *berat* 7 kg.

Inheritance (Pewarisan)



Pewarisan class dalam dunia hewan

- Inheritance : Penurunan sifat dari class.
- Anjing dan kucing mewarisi sifat dari class di atas (*superclass*), yaitu mamalia.

Pewarisan class dalam dunia pendidikan

- Pewarisan dilakukan dengan mendefinisikan class baru, dengan beberapa karakteristik yang diambil dari class lain.
- Karakteristik yang diwariskan terdiri dari field dan metode.

- Class Manusia didefinisikan mempunyai field : *nama*, *TTL*, dan metode *makan()*.
- Artinya, semua class yang diturunkan dari class Manusia (Guru, Dosen, Mahasiswa, Guru SMP, Mahasiswa S2) juga akan mempunyai field *nama*, *TTL*, dan metode *makan()*.

- Penerapan pewarisan telah memudahkan pemahaman sistem serta lebih efisien.
  - Tidak perlu mendefinisikan ulang field dan metode di setiap class.
  - Pengeditan lebih mudah, jika adan perubahan pada class manusia maka class di bawahnya akan mengikuti perubahan tersebut.

Polymorphism

- Kondisi dimana sesuatu mempunyai beberapa bentuk.
- Dalam OOP, penerapan polimorfisme dilakukan menggunakan nama sama, namun implementasi berbeda.
- Contoh : objek mahasiswa dapat melakukan metode mencuci yang berbeda, yaitu metode mencuci motor, mencuci piring, mencuci baju, tentu dilakukan dengan cara yang berbeda.

Polymorphism (Lanjutan)

- Polimorfisme digunakan untuk memudahkan pemrograman karena lebih natural.
- Kita tidak harus menggunakan nama yang berbeda untuk metode yang mirip secara bahasa.
- Untuk contoh sebelumnya, kita cukup menggunakan metode mencuci saja.
- Polimorfisme diterapkan dengan mekanisme
  - Overloading (*signature* yang berbeda pada metode atau konstruktor dg nama sama)
  - Overriding (pendefinisian ulang metode atau konstruktor pada class turunannya/ *subclass*)

Encapsulation

- Implementasi penyembunyian informasi (*information hiding*)
- Tujuannya menyembunyikan informasi data (*field*) objek sehingga tidak terlihat dari luar.
- Informasi tidak diakses sembarangan. Sangat penting untuk keamanan, dan menghindari kesalahan program
- Dilakukan pada class, metode, dan field.
- Penerapannya secara bertingkat menggunakan access modifier yang terdiri atas private, public, dan protected.