Fakultas Informatika
School of Computing
Telkom University

Pemrograman Berorientasi Objek

Kelas Abstrak dan Antarmuka

Abstrak

Metode Abstrak

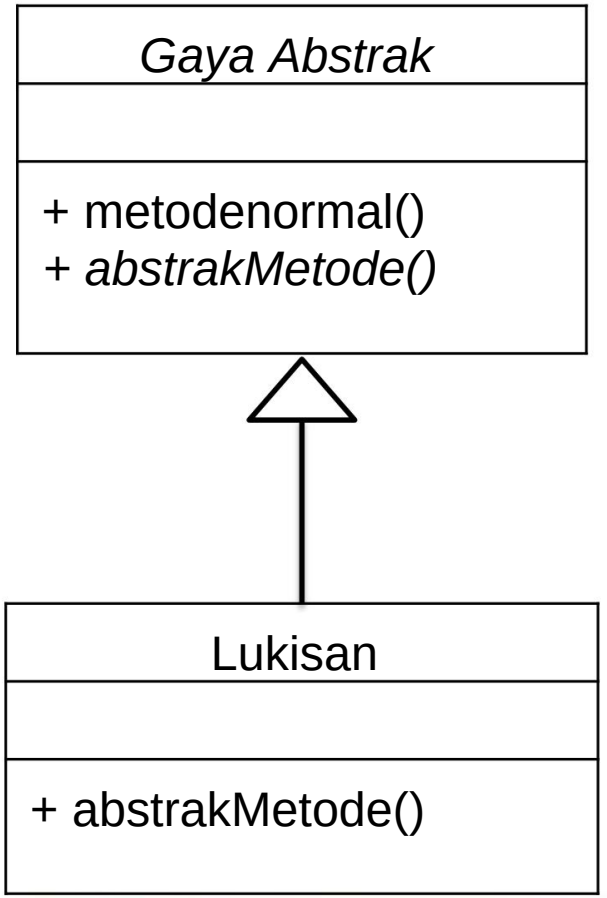
- Metode tanpa implementasi atau perilaku spesifik
- Diserahkan ke kelas anak untuk mengimplementasikannya sendiri
  - –Kelas induk hanya mendeklarasikan nama metode
- Kelas anak **harus** mengimplementasikan atau menentukan metode
  - –Pengesampingan penuh

Kelas Abstrak

- Kelas dengan setidaknya satu metode abstrak harus dideklarasikan sebagai kelas abstrak
- Kelas abstrak tidak dapat diwujudkan
  - –Karena kelas abstrak memiliki metode abstrak, metode yang belum diimplementasikan. Suatu instance harus memiliki semua metode yang sudah diimplementasikan
- Kelas abstrak akan memastikan anak menerapkan metode abstrak yang dideklarasikan
- Gunakan "extends" seperti pewarisan

Diagram Kelas Abstrak

- Diagram kelas dari sebuah abstrak
  - Nama kelas miring
  - Nama metode miring jika abstrak
- Anak memperluas kelas induk abstrak
  - Anak harus mengimplementasikan semua metode abstrak
  - Atau anak akan dinyatakan abstrak juga



Contoh

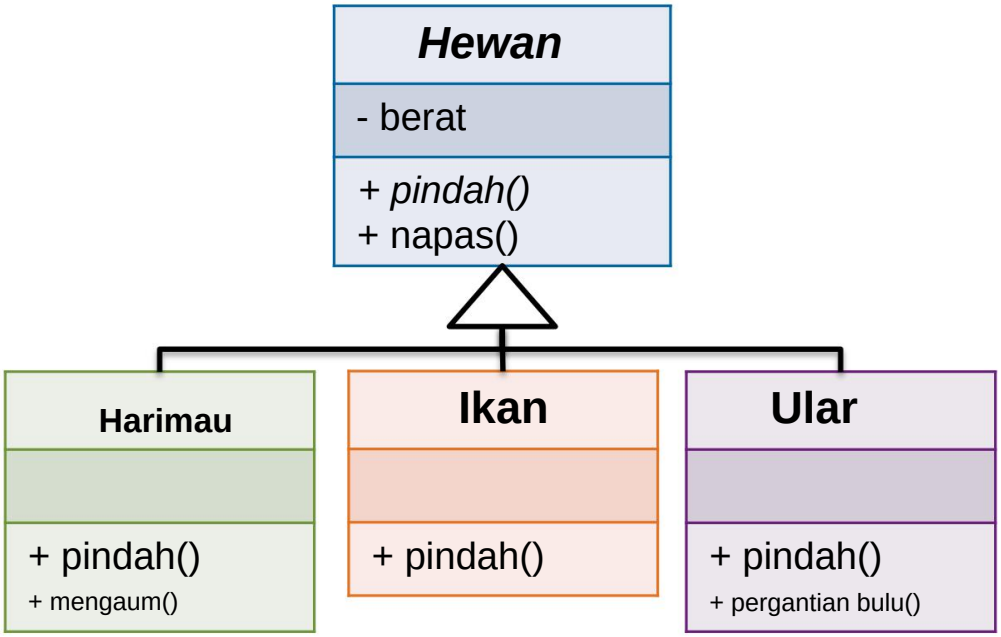
```
kelas abstrak publik AbstractParent{

    publik String toString(){ return "ini adalah
        kelas Induk";
    }

    abstrak publik void abstractMethod();

}
```

```
kelas publik Anak memperluas AbstractParent{

    publik void abstrakMetode(){
        // mengimplementasikan metode abstrak
    }

}
```

Kapan menggunakan/membuat kelas Abstrak

- Penerapan beberapa metode terlalu bervariasi untuk setiap anak
- Tidak ada objek konkret/aktual untuk kelas induk
  - Tidak ada objek seperti itu
  - Induk tidak perlu diwujudkan
- Buat kondisi bahwa kelas anak harus menerapkan beberapa metode tertentu

Contoh: Hirarki Hewan



Harimau, ikan, dan ular adalah hewan dan mereka semua bisa bergerak, tetapi perilaku 'bergerak' mereka sangat berbeda

Kita tidak perlu menentukan perilaku dalam kelas hewan, biarkan kelas anak mendefinisikan dirinya sendiri

pandangan lain:
Kami memastikan setiap hewan dapat bergerak

Antarmuka

Antarmuka

- Tipe kelas khusus yang mendefinisikan sekumpulan prototipe metode
- Untuk menyampaikan aturan atau mekanisme ke kelas lain
- Sebuah interface hanya dapat berisi – atribut

  publik, metode abstrak – atribut

  publik, statis, final

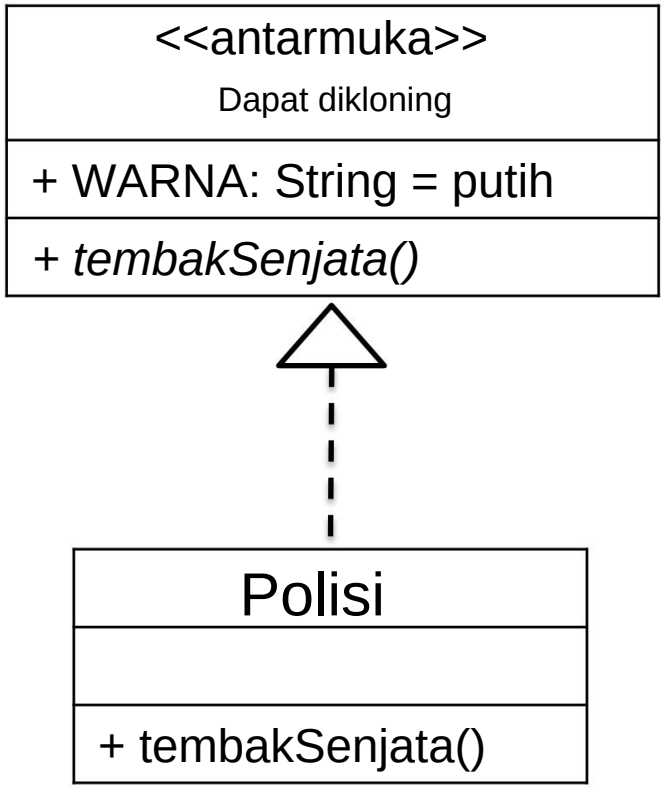
  – Tidak ada konstruktor

Antarmuka

- Gunakan kata kunci "implements"
- Kelas A
  - –Hanya dapat “memperluas” satu kelas, yaitu hanya satu superkelas
  - –Dapat "menerapkan" beberapa antarmuka
- Antarmuka tidak memiliki bidang instans
  - – mereka hanya sekumpulan metode yang harus dimiliki oleh objek melaksanakan

Diagram Kelas Antarmuka

- Diagram kelas dari sebuah antarmuka
  - Tandai <<antarmuka>>



- Ketika kelas A mengimplementasikan Antarmuka I, diagram tersebut dilambangkan sebagai

Antarmuka

- Seperti kelas abstrak, antarmuka tidak dapat diwujudkan
- Kelas yang mengimplementasikan antarmuka harus mengimplementasikan semua metode abstrak
- Kelas abstrak juga dapat mengimplementasikan antarmuka
- Kelas abstrak yang mengimplementasikan antarmuka mungkin tidak mengimplementasikan semua metode abstrak

Antarmuka

- Menerapkan antarmuka memungkinkan kelas menjadi lebih formal tentang perilaku yang dijanjikannya untuk diberikan

- Penggunaan antarmuka praktis memungkinkan pemisahan definisi dan implementasi suatu metode
  - Definisi (kumpulan aturan) disediakan oleh antarmuka
  - Implementasinya disediakan oleh kelas yang mengimplementasikan metode antarmuka

Antarmuka

- Di Java, Interface biasanya diberi nama **“…… - able”** (tapi tidak selalu)
  - –Dapat diserialisasikan
  - –Dapat dijalankan
  - –Dapat dikloning
- Artinya kelas yang mengimplementasikan sebuah interface akan bisa melakukan apa saja sesuai perintah interface tersebut.

Contoh

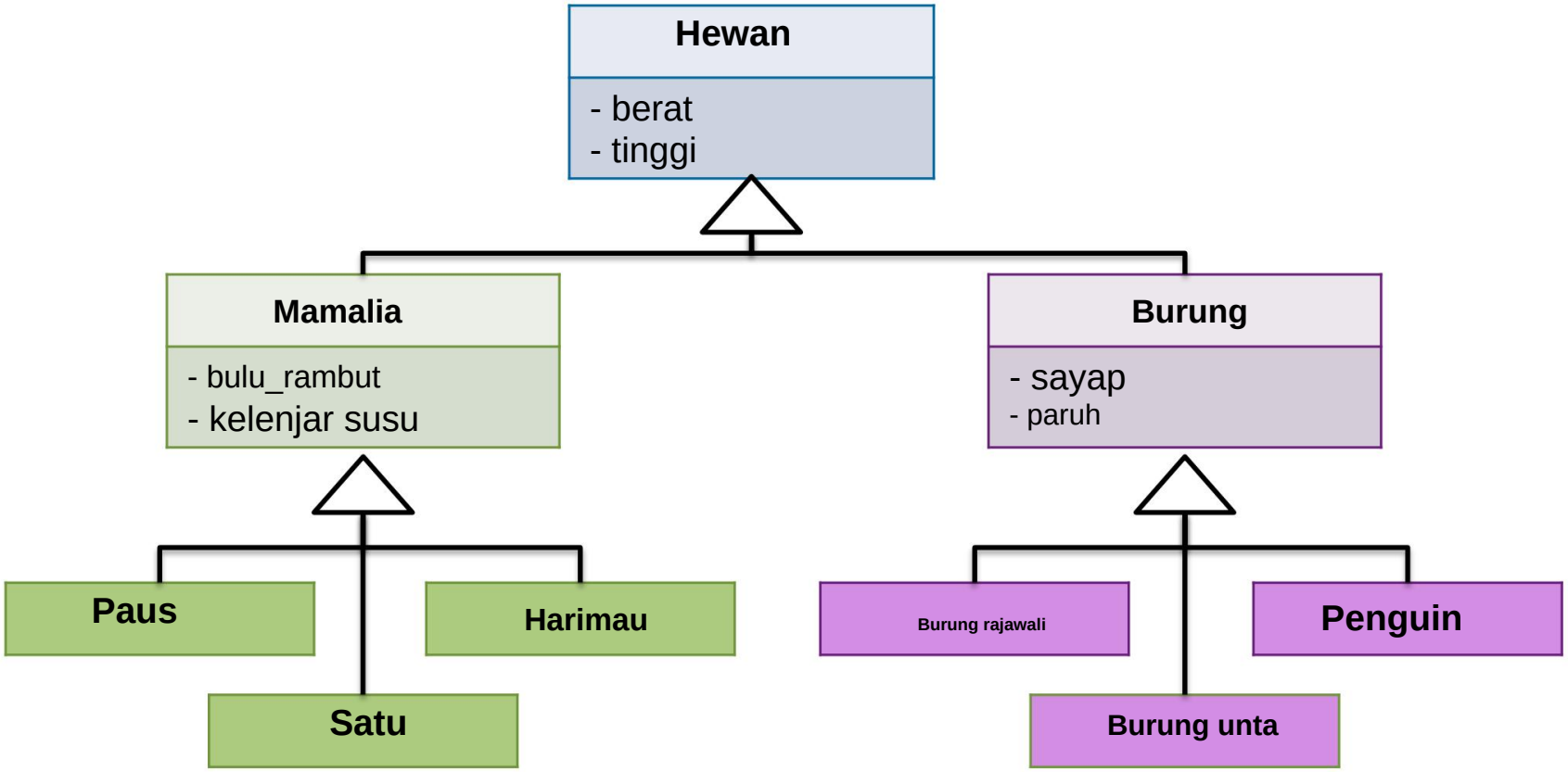
```
antarmuka Instagramable{

    abstrak publik void ambilGambar(); abstrak publik void
    setFilter(); abstrak publik void bagikanFoto();

}
```

```
kelas publik Kitten menerapkan Instagramable{

    publik void takePicture(){ //
        mengimplementasikan metode abstrak
    }

    publik void setFilter(){ //
        mengimplementasikan metode abstrak
    }

    publik void sharePhoto(){ //
        mengimplementasikan metode abstrak
    }
}
```

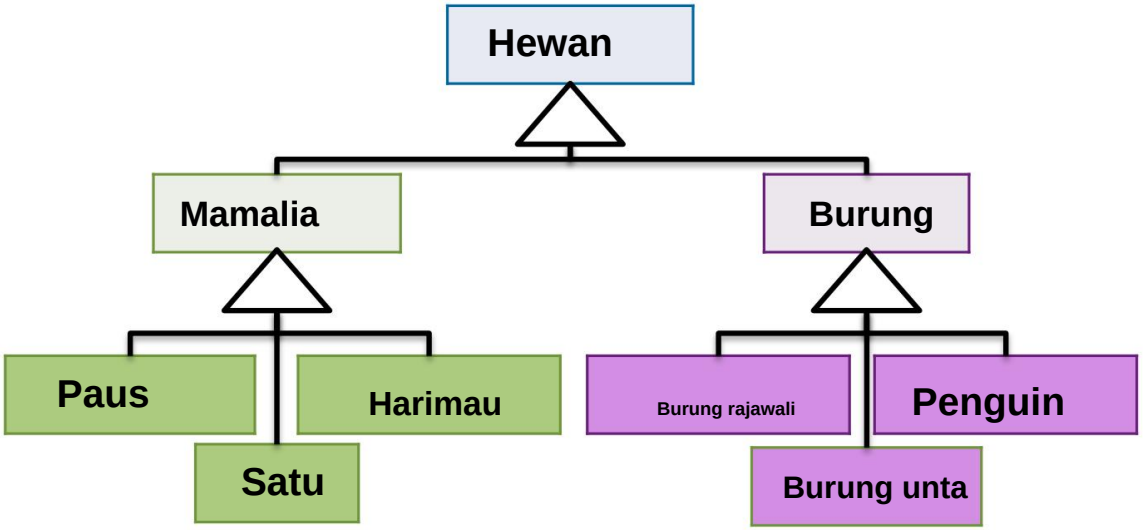
Kapan menggunakan/membuat Antarmuka

- Tentukan beberapa aturan atau mekanisme
- Kelas grup tanpa hubungan pewarisan

Contoh: Hirarki Hewan



Contoh: Hirarki Hewan

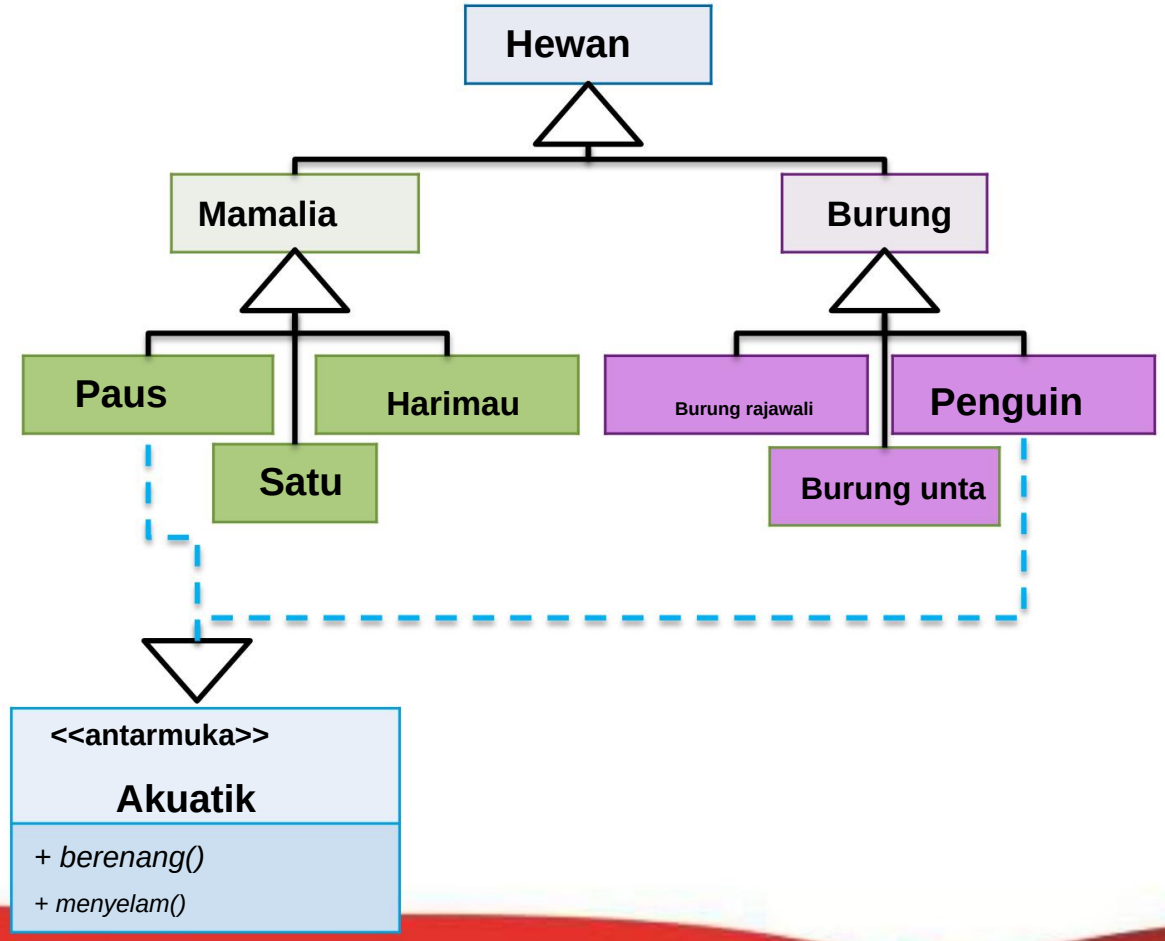


Paus dan penguin merupakan hewan akuatik dan dapat berenang, namun kelelawar, harimau, elang dan burung unta tidak dapat berenang.

kita tidak bisa memasukkan perilaku berenang ke dalam kelas mamalia, burung atau hewan karena tidak semua mamalia, burung dan hewan bisa berenang

Kita juga tidak bisa menempatkan penguin dan paus pada kelas induk yang sama karena keduanya berbeda.

Contoh: Hirarki Hewan



Di sini kita dapat membuat

Hewan Akuatik
Antarmuka yang mendefinisikan

bahwa hewan yang

mengimplementasikan
antarmuka ini akan dapat

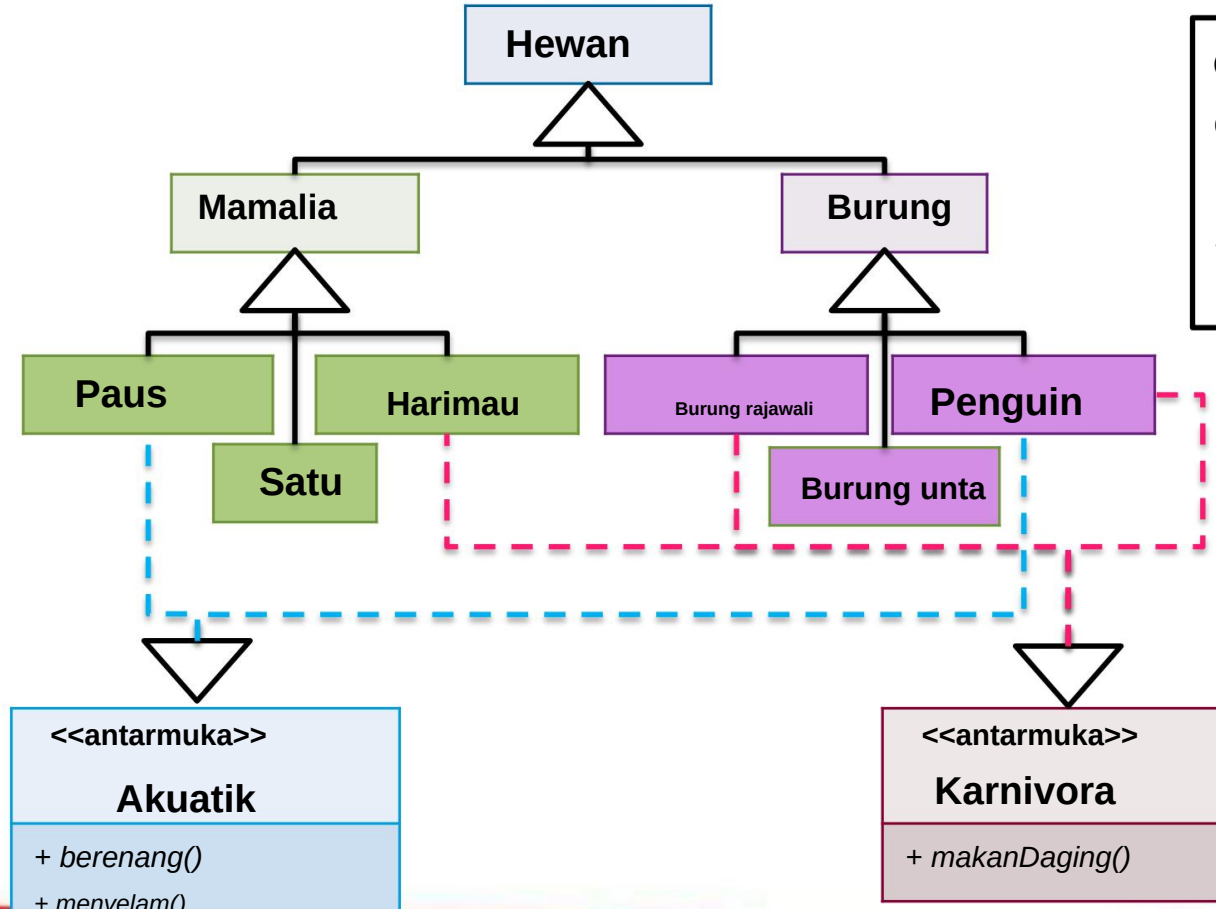
berenang dan menyelam

Kemudian kita bisa membuat

paus dan penguin menerapkan

perilaku antarmuka Akuatik

Contoh: Hirarki Hewan



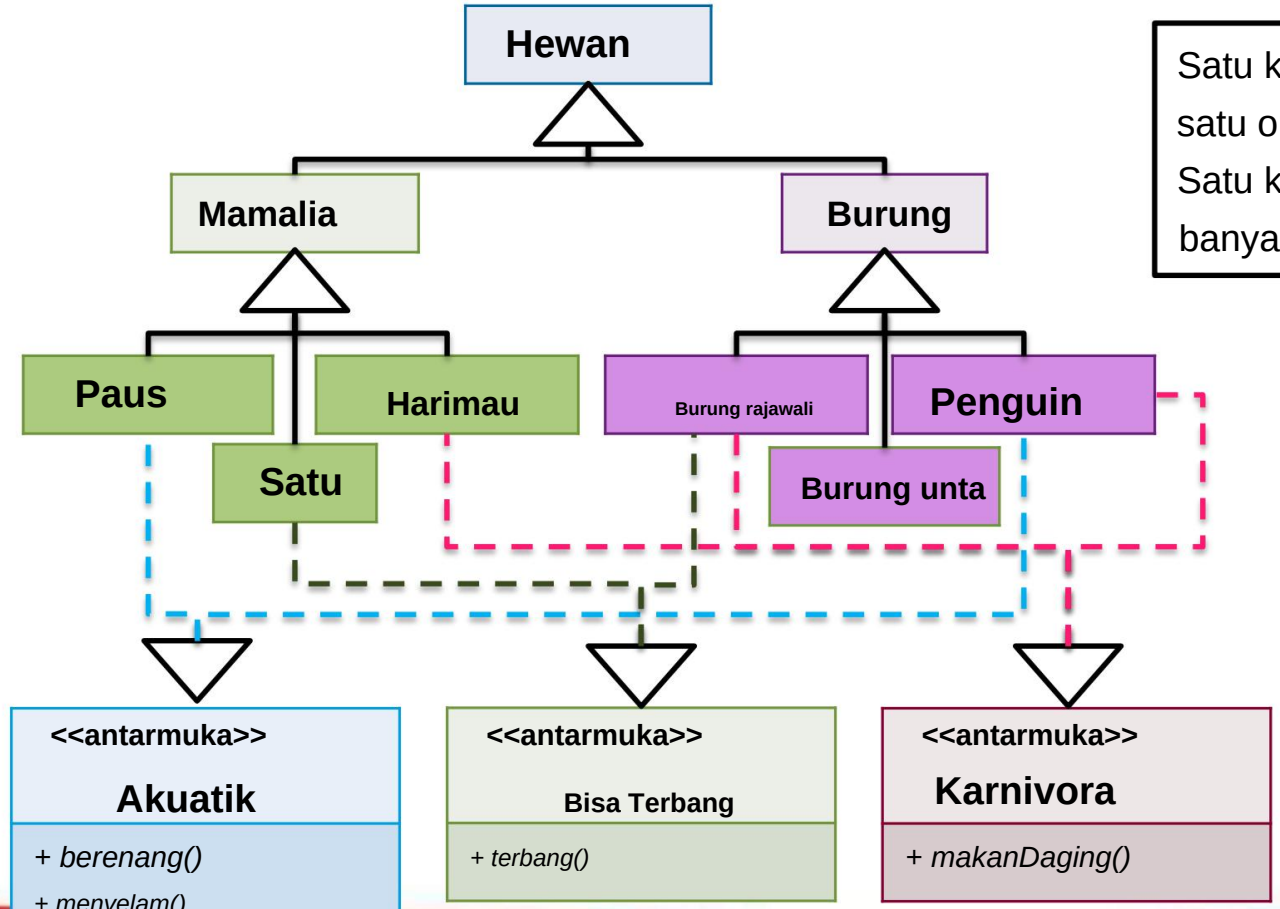
Contoh lain, harimau,
elang, dan penguin adalah
karnivora,
sedangkan sisanya tidak.

Buat Karnivora

Antarmuka, dan membuat hewan karnivora

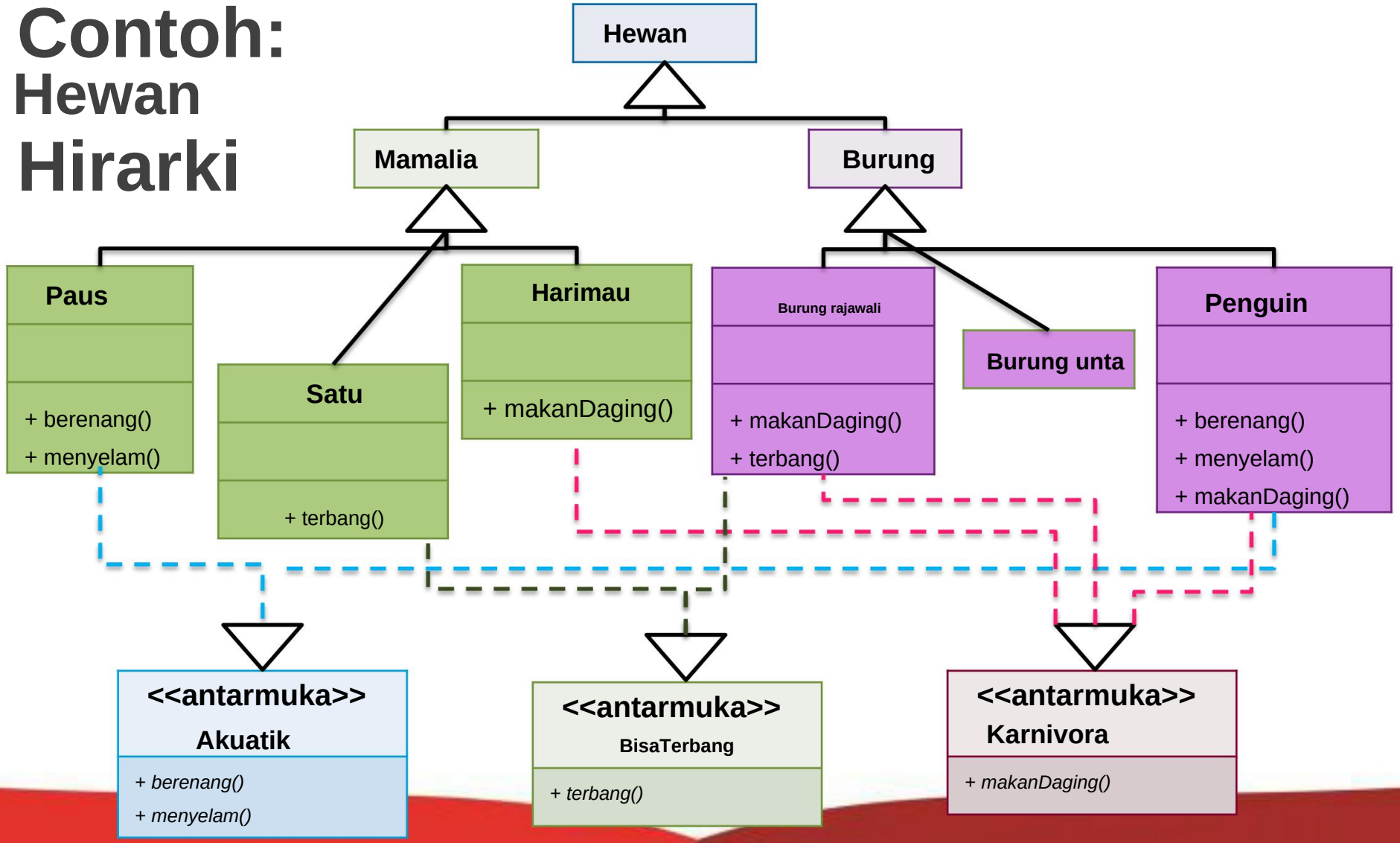
menerapkan perilaku

Contoh: Hirarki Hewan

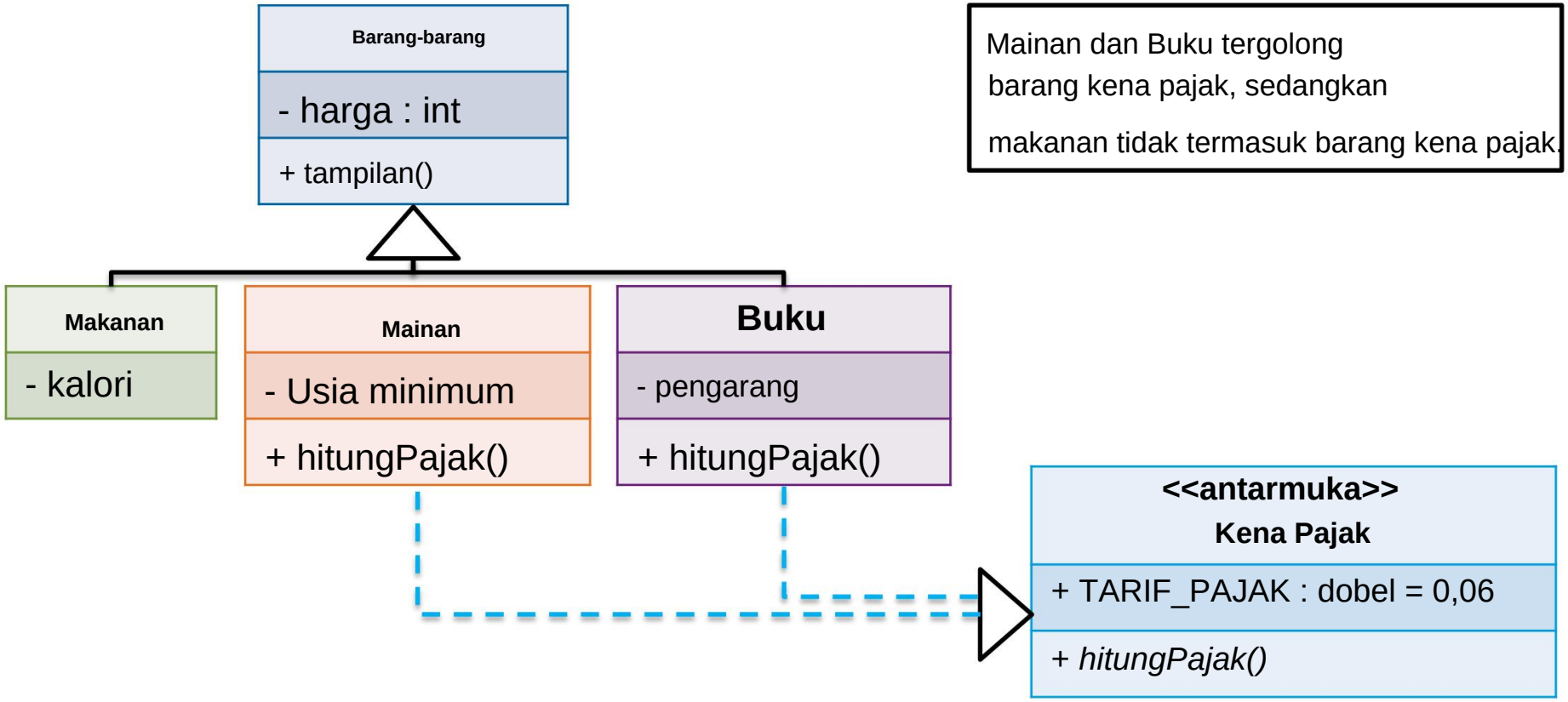


Satu kelas hanya dapat memiliki satu orang tua
Satu kelas dapat mengimplementasikan banyak antarmuka

Contoh:
Hewan
Hirarki



Contoh: Barang Kena Pajak



Pertanyaan?

Fakultas Informatika
School of Computing
Telkom University

TERIMA KASIH