JURUSAN TEKNOLOGI INFORMASI

Pemrograman Berorientasi Objek

04. Pewarisan (Bagian-1)

Yoppy Yunhasnawa, S.ST., M.Sc.

Topik

1. Pendahuluan
2. Sintaksis, dan Aturan Pewarisan pada Bahasa Pemrograman

Topik #1: Pendahuluan

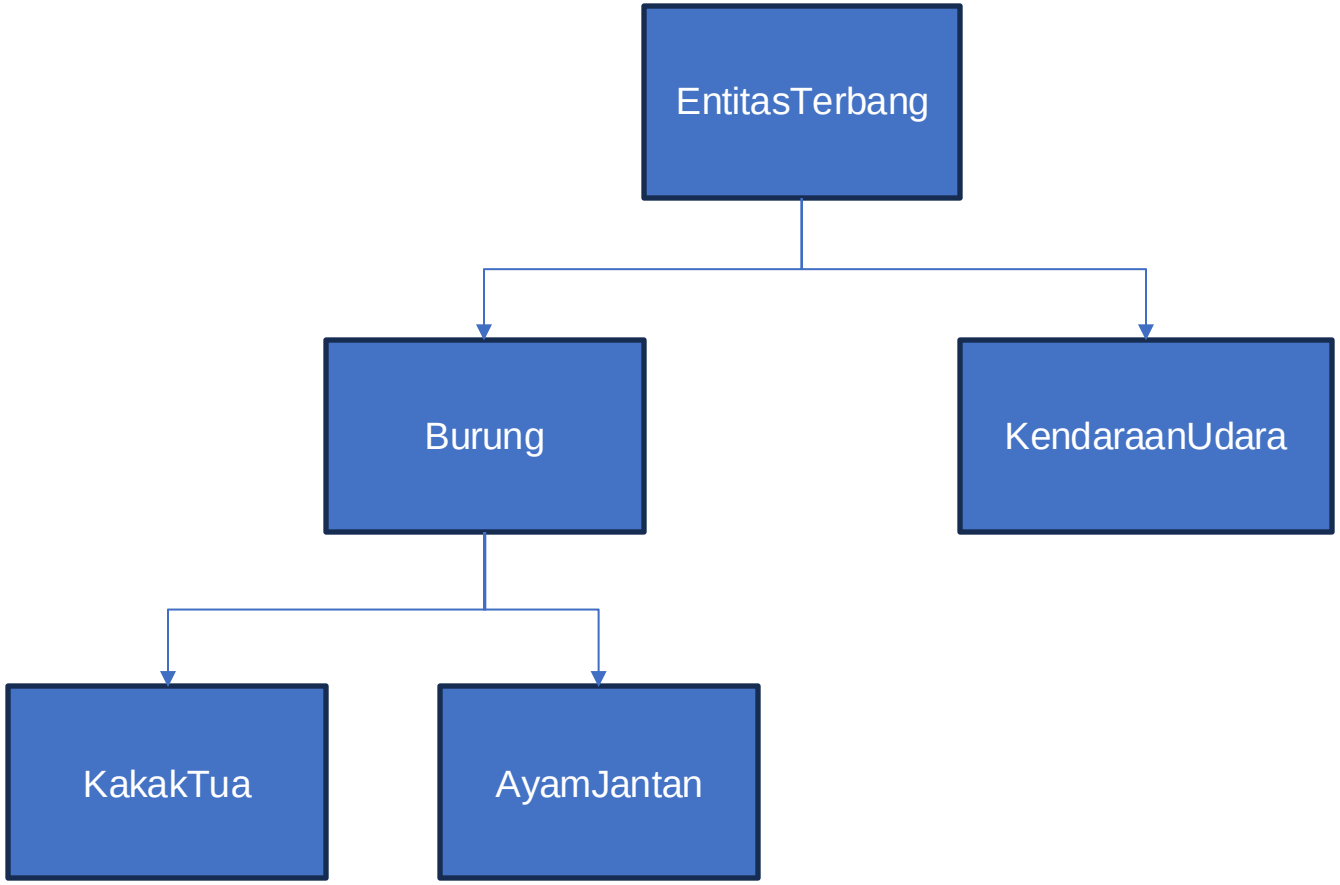
1. Pewarisan

- Pewarisan asal katanya dari *Inheritance* → Adalah konsep dalam OOP yang mengusahakan agar kode dan/atau prinsip kerja yang serupa agar **tidak perlu ditulis/dipahami ulang**.

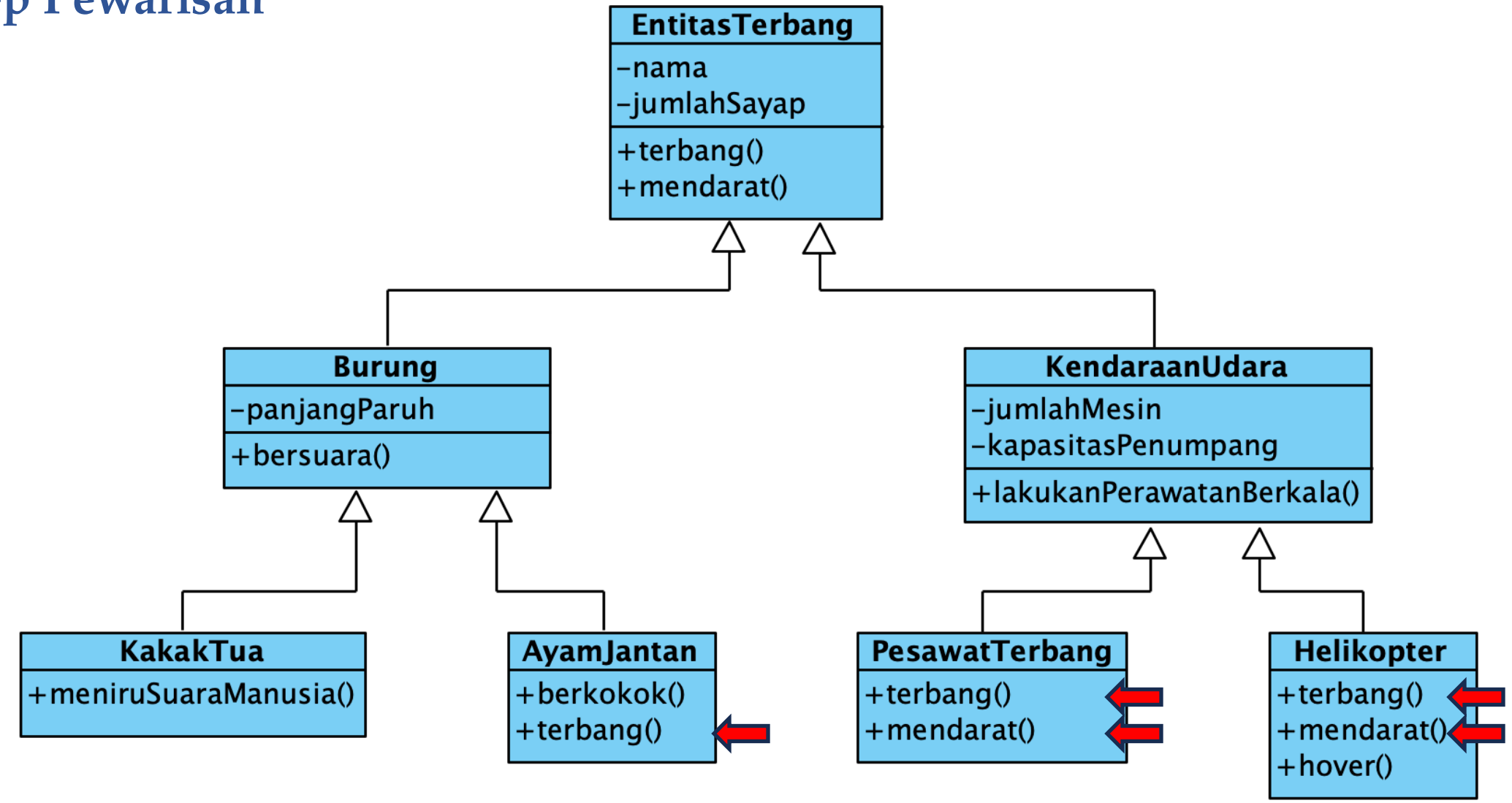
- Manfaat → **DRYS** (*Don't Repeat Your Self)*
  - Meminimalkan jumlah baris kode yang harus dibuat.
  - Menghemat waktu yang dibutuhkan untuk memahami kode.
  - Menjamin tidak terjadi permasalahan pada kode yang sudah dibuat sebelumnya, ketika menambahkan fungsionalitas baru pada program.

- Prinsip utama → Suatu class bisa jadi merupakan turunan class lain.
  - Turunan → Bisa jadi B juga merupakan A, namun A belum tentu B.
  - Contoh:
    - Entitas Terbang (sesuatu yang bisa terbang)
    - Burung termasuk Entitas Terbang
    - Merpati termasuk Burung

Konsep Pewarisan



Konsep Pewarisan



Diskusi

- Pada slide sebelumnya, tahukah Anda mengapa saya meletakkan beberapa anak panah warna merah di beberapa tempat tertentu?

Topik #2: Sintaksis dan Aturan Pewarisan pada Bahasa Pemrograman

2. Sintaksis dan Aturan Pewarisan pada Bahasa Pemrograman

- Untuk membuat suatu class menjadi turunan class lain, digunakan keyword “extends”
- Suatu class anak harus memiliki constructor yang sesuai dengan class induknya.
- Agar property dan/atau method dapat diakses dari class anak, maka *access modifier*-nya harus dibuat **protected**.
- Apabila ada method dari class anak yang berbeda dengan method dari class induknya, maka method induknya harus ditimpa (*override*).
  - Method *override* dituliskan dengan menggunakan ***annotation @Override***

Extends

```
public class Burung extends EntitasTerbang
{
```

- Keyword "**extends**" akan menjadikan class Burung sebagai *subclass* atau *child class* dari class Entitas Terbang.
- Di java, satu class hanya bisa menjadi subclass dari 1 *superclass* saja.
- Di java, class yang menjadi induk dari segala class adalah class **Object**.

Constructor Matching Super

- Class Burung merupakan turunan class Entitas Terbang.
- Sehingga class tersebut harus memanggil constructor class induknya saat constructornya dipanggil.
- Untuk memanggil constructor class induk digunakan keyword "**super**".
  - Diikuti dengan parameter-parameter yang sesuai dengan parameter di constructor class induk.

```
// Constructor
2 usages
public EntitasTerbang(String nama, int jumlahSayap)
{
    this.nama = nama;
    this.jumlahSayap = jumlahSayap;
}
```

```
// Constructor
3 usages
public Burung(String nama, int jumlahSayap) {
    super(nama, jumlahSayap); // <-- Call superclass' constructor
}
```

Protected

```
public class EntitasTerbang
{
    7 usages
    protected String nama;
```

- Agar bisa diakses dari class induknya, maka *access modifier* dari property maupun method dari suatu class induk harus dijadikan minimal **protected**.
- Mengubah access modifier private menjadi protected tidak akan (sangat minim kemungkinan) mempengaruhi kode yang lain.
- Ingat! Jangan jadikan property sebagau **public**. Gunakan getter dan setter saja.

Override

```
public void terbang()
{
    System.out.println(this.nama + " Terbang: Wushhhh!");
}
```

```
@Override // <-- Anotasi (Annotation)
public void terbang()
{
    System.out.println(this.nama + " Terbang: Pek..pek..pek..");
}
```

- Ketika suatu method bawaan dari class induk dirasa tidak cocok (berbeda alur prosesnya) dengan method yang ada di class anak, maka method tersebut dapat ditimpa (override).

- Untuk menimpa suatu method:
  - Tuliskan annotation **@Override**.
  - Buat method dengan *access modifier*, nilai balik, dan nama, serta parameter yang sama persis dengan yang ada di class induknya.

Class "Object"

- Anda sudah mengetahui bahwa suatu class di Java bisa jadi merupakan turunan (subclass/childclass) dari suatu class lain (superclass/parent-class).

- Namun tahukah Anda bahwa **semua** class di Java merupakan turunan dari 1 superclass yang paling "super"?

- Superclass dengan hirarki paling tinggi di Java adalah class yang bernama class "**Object**".
  - Jangan bingung antara "class Object" vs "object".

- Maka dari itu semua class yang kita buat secara otomatis akan mewarisi semua method dan property dari class Object tersebut.

Pertanyaan?

Terima Kasih

Tugas Latihan

- Implementasikanlah class diagram contoh konsep pewarisan pada slide ini!

References

1. -