Abstract Class

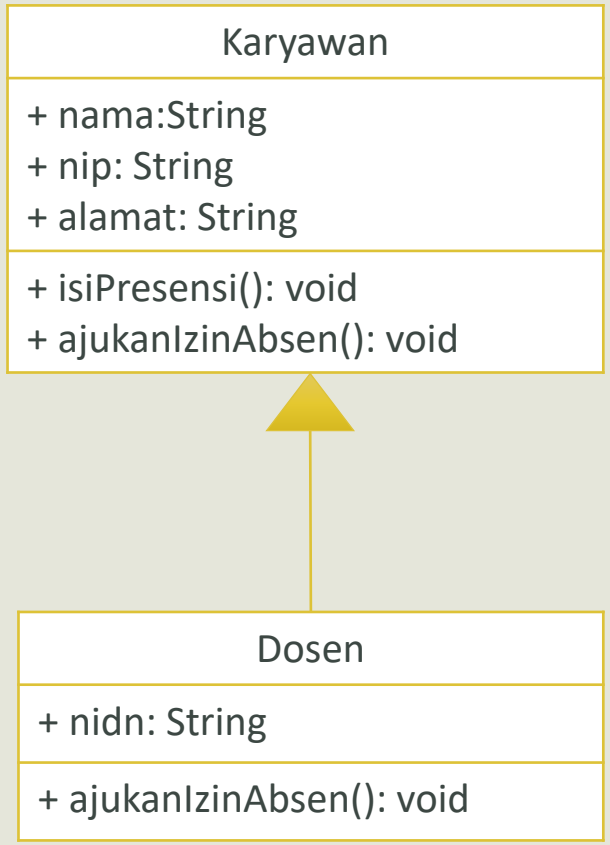
Tim Ajar PBO – JTI Polinema

TUJUAN

- Mahasiswa mampu memahami konsep abstract class
- Mahasiswa mampu membuat class diagram untuk studi kasus dengan abstract class

OUTLINE

- Studi Kasus
  - Concrete Superclass
  - Abstract Superclass
- Definisi Abstract Class
- Abstract Method
- Syntax
- Notasi dalam Class Diagram
- Penggunaan




```
public void ajukanIzinAbsen(){
    System.out.println("Menghubungi Biro Kepegawaian");
}
```

```
public void ajukanIzinAbsen(){
    System.out.println("Menghubungi Biro Kepegawaian");
    System.out.println("Menginfokan kepada ketua kelas");
}
```

DEFINISI

- *Abstract class* merupakan class yang **tidak dapat diinstansiasi namun dapat di-*extend***.
- Kegunaan:
  - Sebagai **generalisasi** sublass atau guideline untuk subclass
- Karakteristik:
  - **Selalu** dideklarasikan dengan menggunakan keyword '`abstract class`'
  - **Dapat** memiliki properties dan methods seperti *concrete class*
  - Class yang memiliki abstract method harus dideklarasikan sebagai abstract class. Tetapi suatu class bisa dideklarasikan sebagai abstract tanpa memiliki abstract method

ABSTRACT METHOD

- Method yang hanya dideklarasikan tetapi tidak memiliki implementasi (*body*)
- *Abstract method* dibuat dengan keyword ***abstract***
- Abstract method hanya menjelaskan apa saja yang bisa dilakukan oleh sebuah class namun tidak ada detail bagaimana cara melakukannya

SYNTAX

- Untuk mendeklarasikan abstract class:
  - **<modifier> abstract class** <NamaClass>
- Untuk mendeklarasikan abstract method:
  - **<modifier> abstract <return_type>** <namaMethod>();
- Contoh:

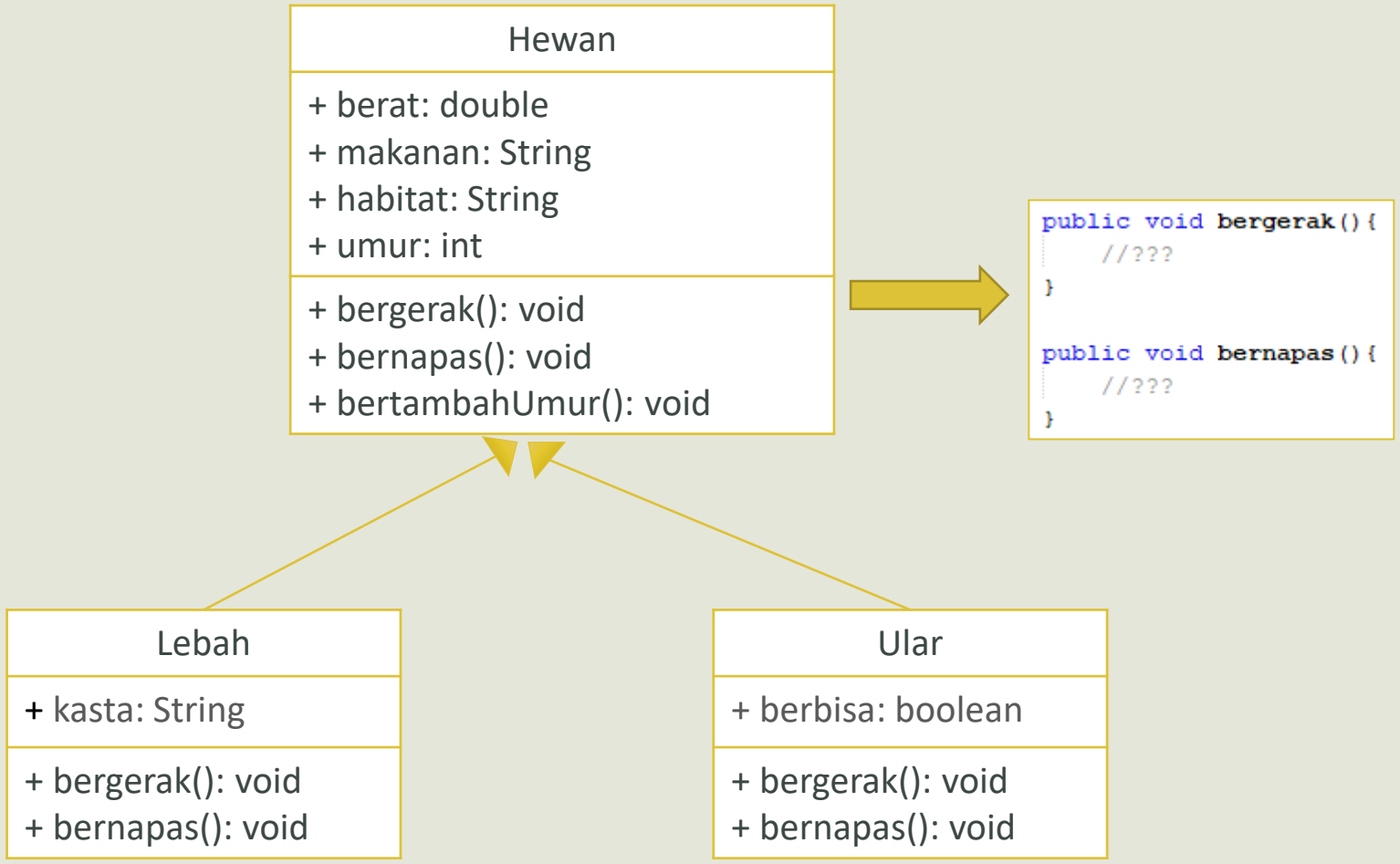
```
public abstract class Hewan {
    public abstract void bergerak();
    public abstract void bernapas();
}
```

Penggunaan Abstract Class

- Abstract class tidak dapat diinstansiasi (tidak dapat dibuat objectnya)
- Baris kode berikut akan memunculkan compilation error "*Hewan is abstract; cannot be instantiated*"

```
Hewan hewan1 = new Hewan();
```

- Untuk menggunakan *abstract class*, dibuat concrete class yang meng-extend *abstract class* tersebut
  - concrete class menggunakan extends keyword
  - concrete class harus mengimplementasi semua abstract method
- Class yang menge-extend abstract class tetapi tidak mengimplementasi seluruh abstract method nya maka harus dideklarasikan sebagai abstract class juga



Method yang belum dapat diimplementasikan sebaiknya dideklarasikan sebagai ***abstract method***.

Akibatnya class-nya menjadi ***abstract class***

```java
public class Ular extends Hewan {
    public boolean berbisa;

    public void bergerak() {
        System.out.println("Otot-otot di bagian samping tubuh ular berkontraksi, sehingga tubuh ular menjadi memanjang");
        System.out.println("tot-otot di bagian samping tubuh ular berelaksasi, sehingga tubuh ular kembali.");
    }

    public void bernapas() {
        System.out.println("Melebarkan tulang rusuk untuk menciptakan ruang hampa");
        System.out.println("Menghirup udara melalui hidung -> tenggorokan -> trakea -> paru-paru");
        System.out.println("Menyempitkan tulang rusuk, mendorong udara keluar");
    }
}
```

```java
public class Lebah extends Hewan {
    public String kasta;

    public void bergerak() {
        System.out.println("Mengepakkan sayap mereka ke depan dan ke belakang");
        System.out.println("Tekanan udara di bagian atas lebah lebih rendah dibanding tekanan udara di sekitarnya");
        System.out.println("Tubuh lebah terangkat ke atas");
    }

    public void bernapas() {
        System.out.println("Sebuah katup di setiap spirakel lebah terbuka dan menyedot udara segar");
        System.out.println("Udara ditransfer melalui lengan trakea ke dalam kantung udara");
        System.out.println("Kantung udara berkontraksi, memaksa udara melewati trakeol dan masuk ke jaringan lebah");
        System.out.println("karbon dioksida dipompa kembali melalui trakeol dan keluar melalui spirakel");
    }
}
```

Class/Concrete Class

```java
public class Hewan {
    public double berat;
    public String makanan;
    public String habitat;
    public int umur;

    public void bergerak(){
        //???
    }

    public void bernapas(){
        //???
    }

    public void bertambahUmur(){
        this.umur += 1;
    }
}
```

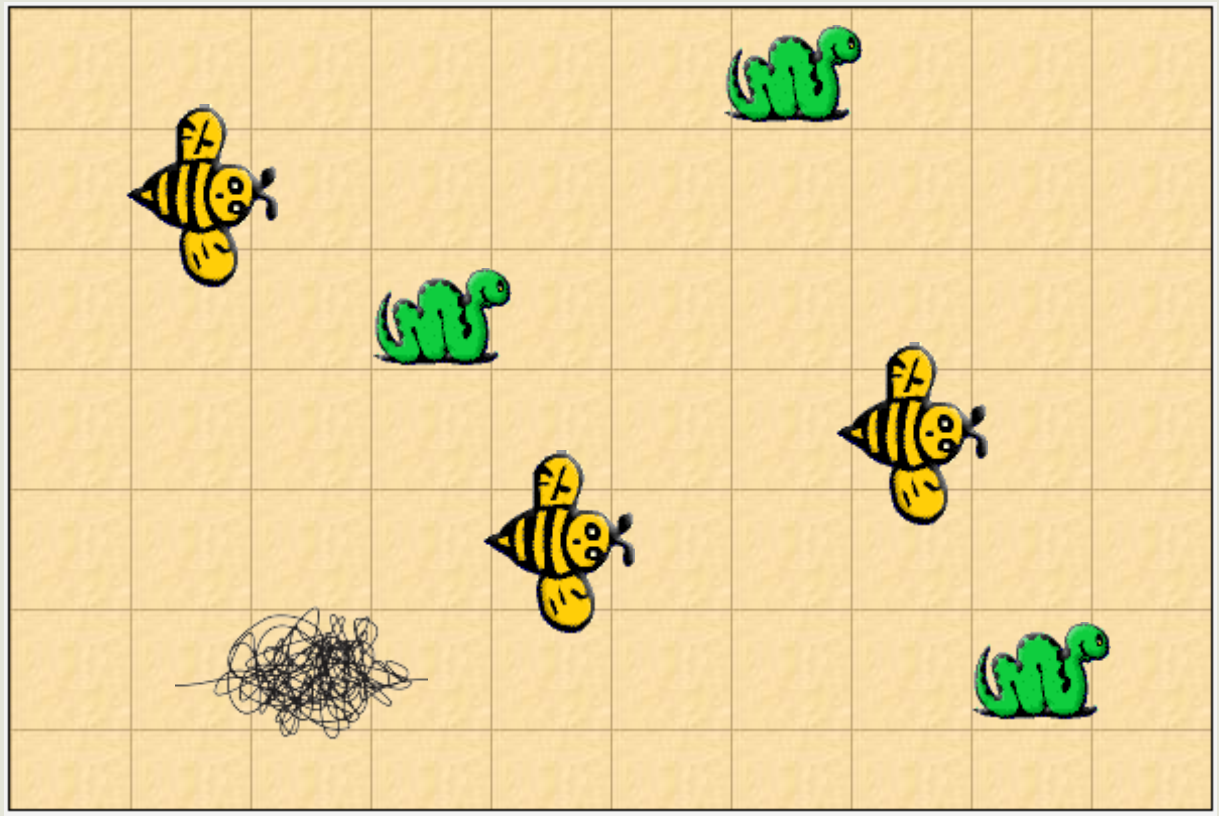
```java
public static void main(String[] args) {
    Hewan h = new Hewan();
}
```

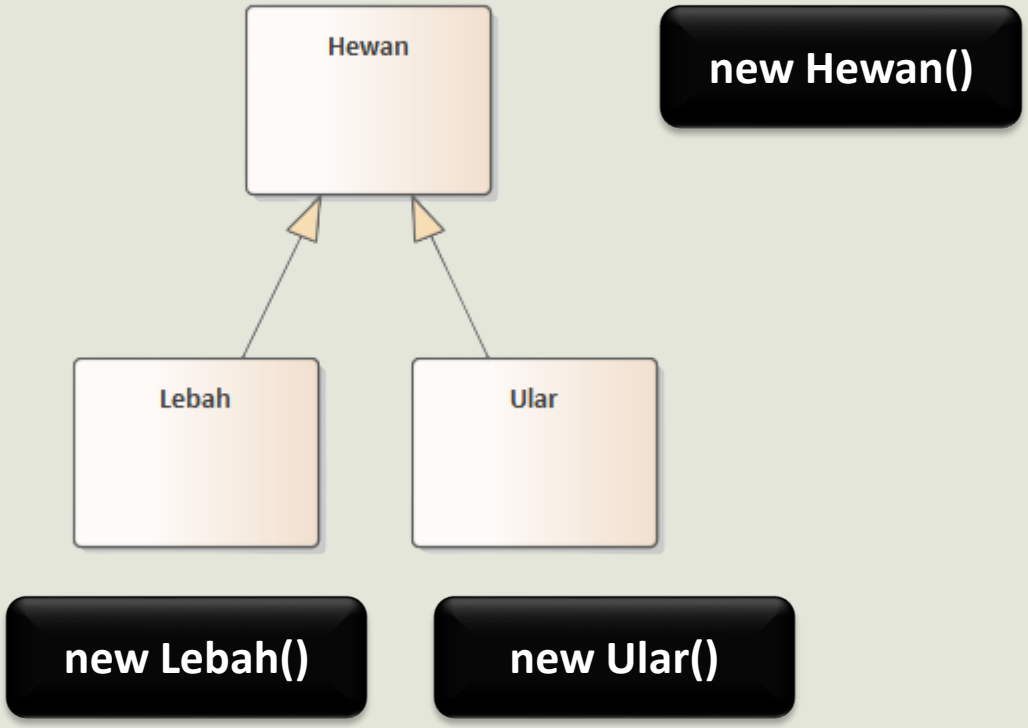
Abstract Class

```java
public abstract class Hewan {
    public double berat;
    public String makanan;
    public String habitat;
    public int umur;

    public abstract void bergerak();
    public abstract void bernapas();

    public void bertambahUmur(){
        this.umur += 1;
    }
}
```

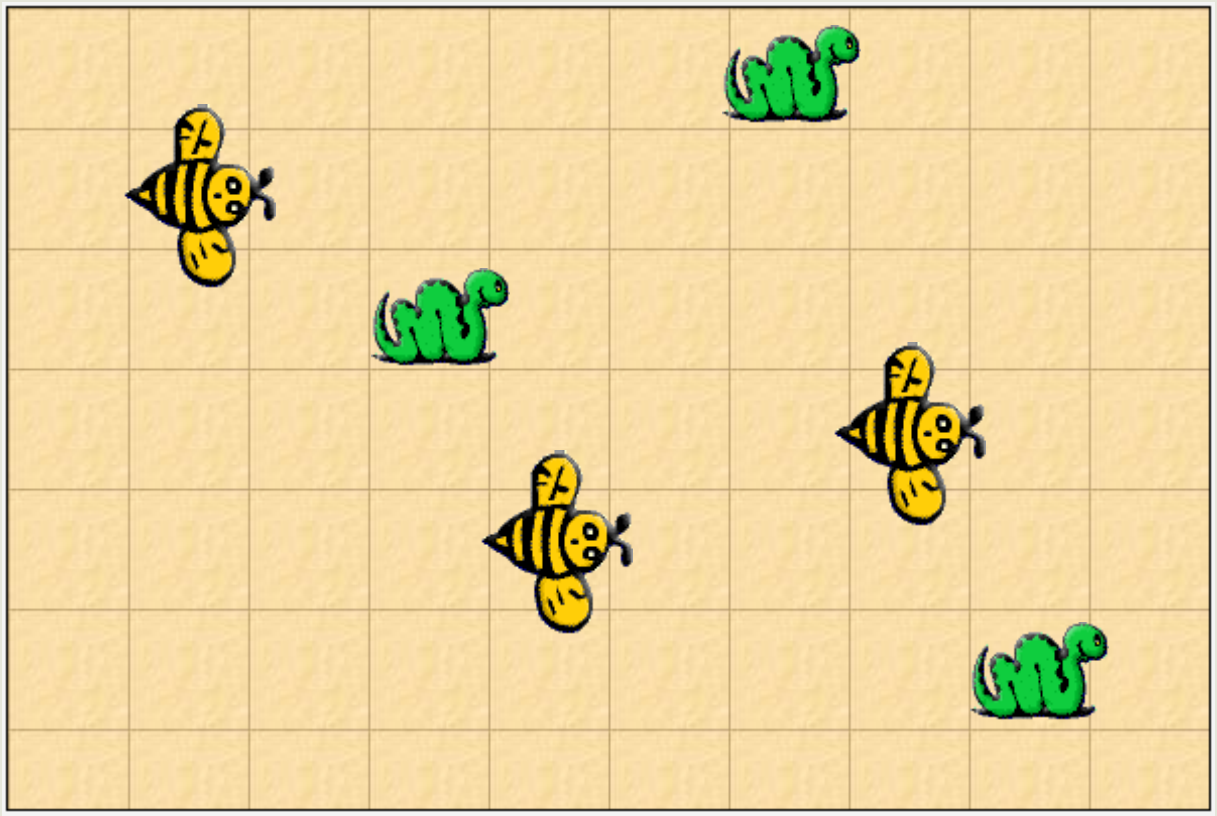
```java
public static void main(String[] args) {
    Hewan h = new Hewan();
}
```

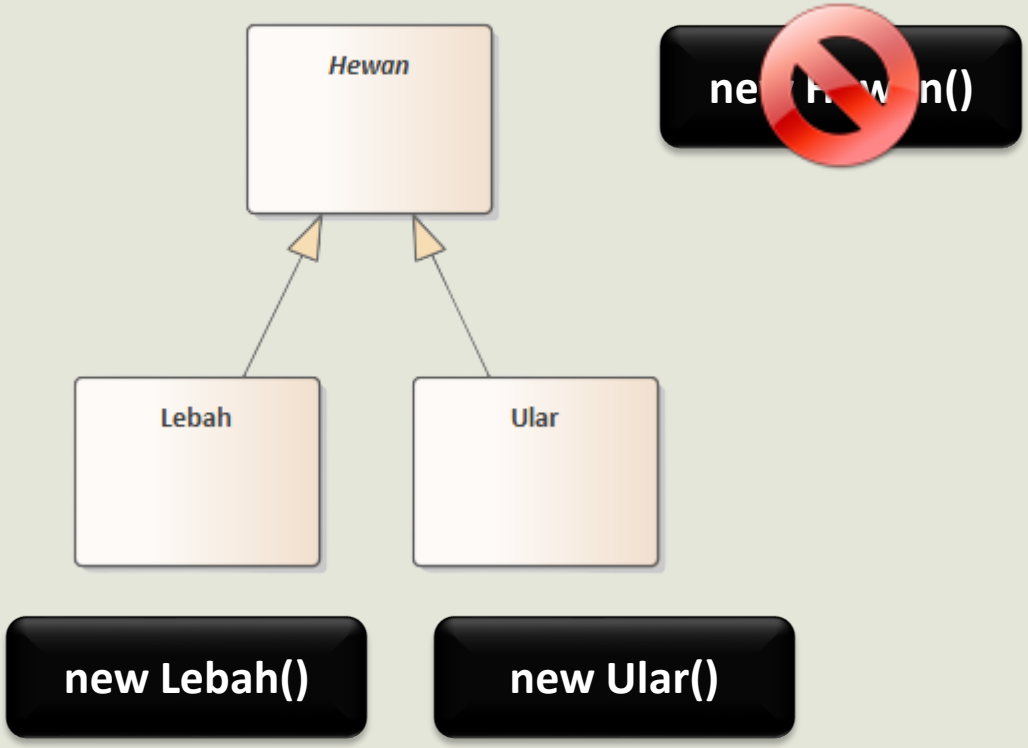
Concrete Superclass



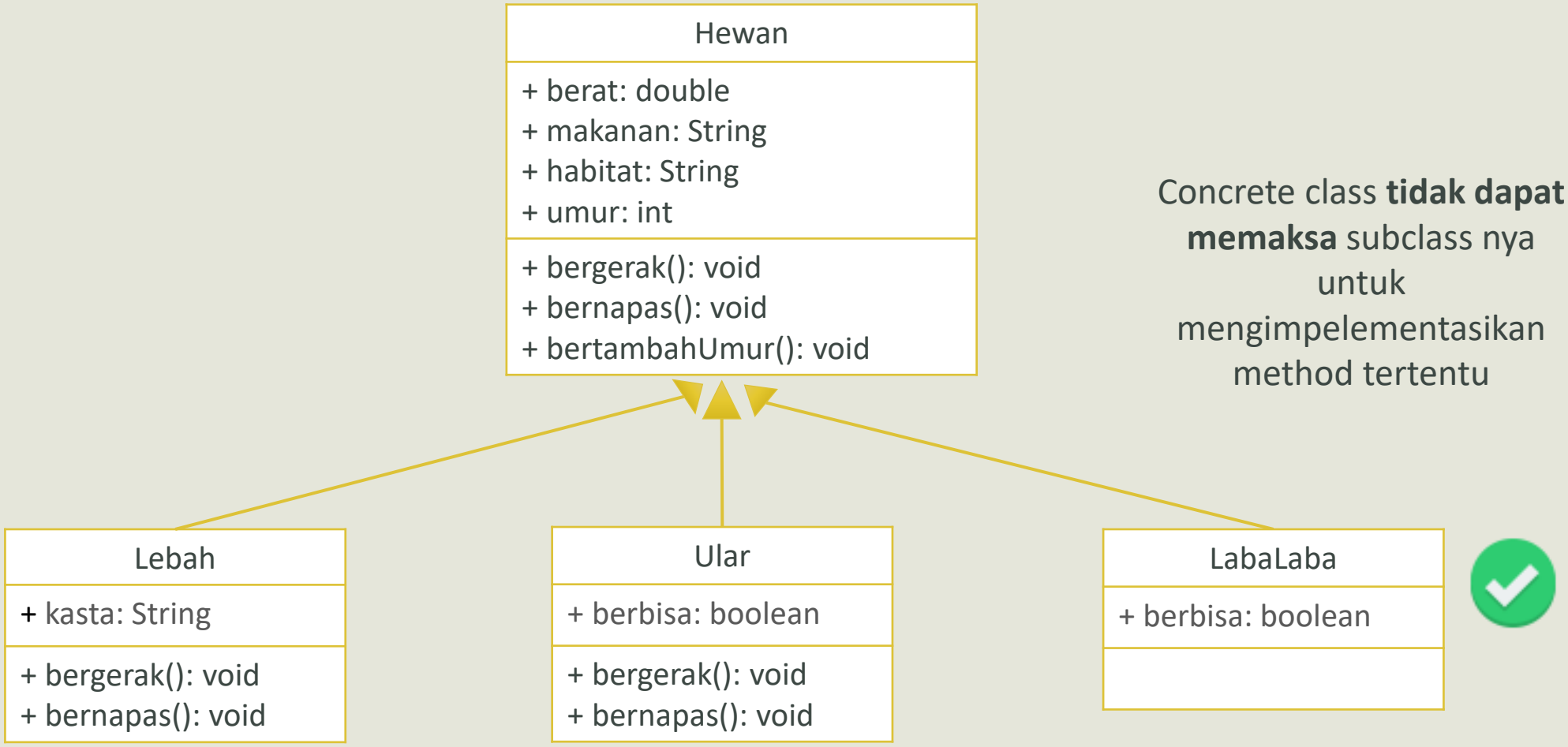


Abstract Superclass



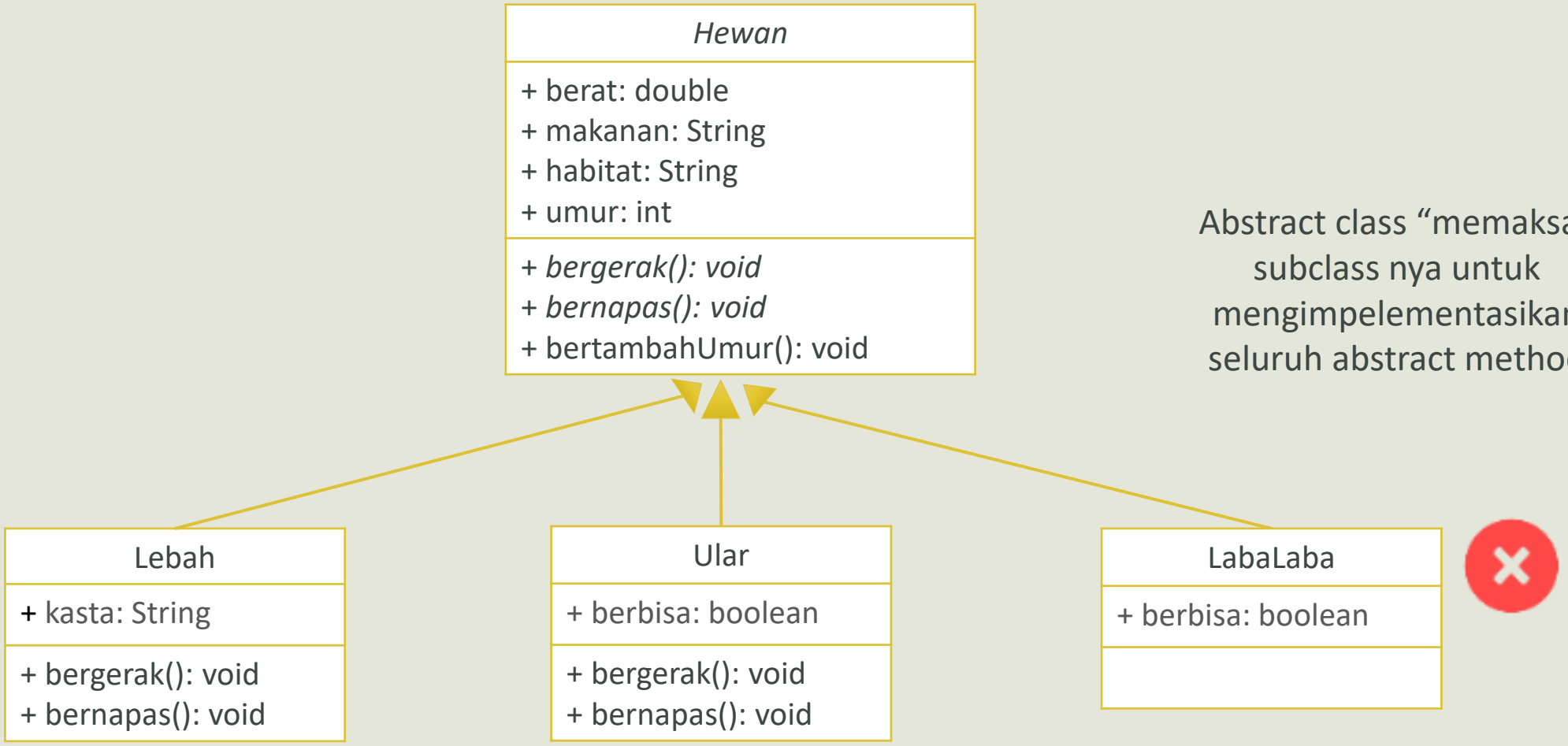


Concrete Superclass



Concrete class **tidak dapat memaksa** subclass nya untuk mengimpelementasikan method tertentu

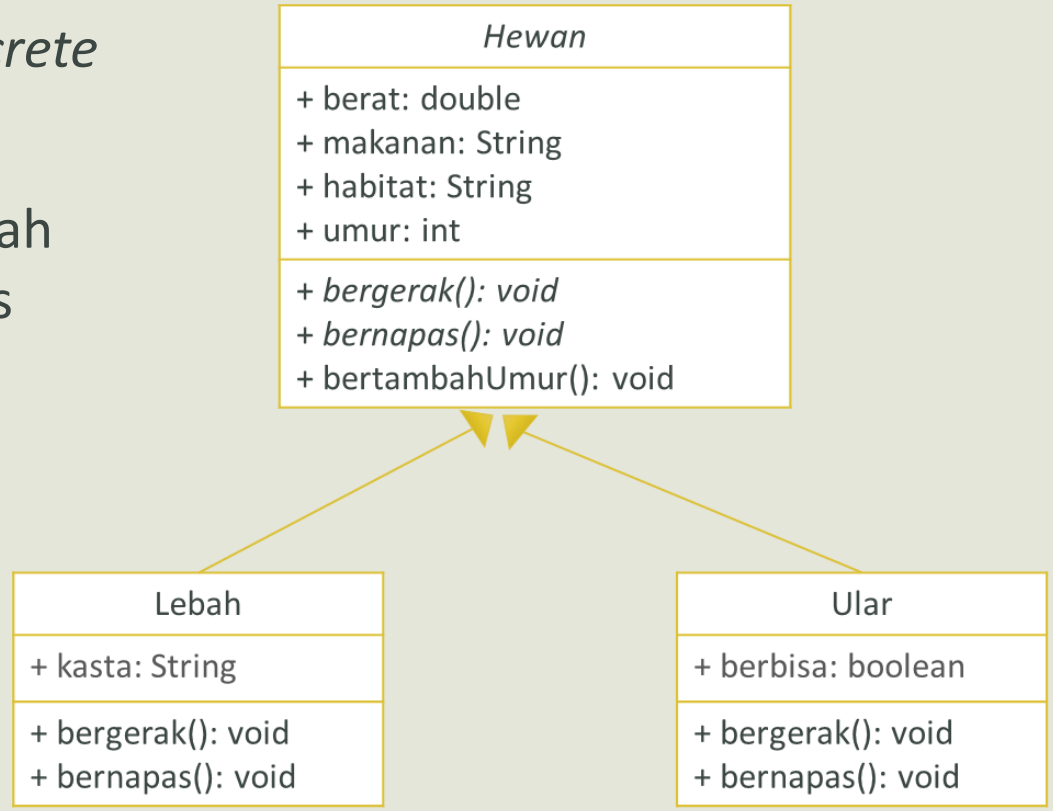
Abstract Superclass



Abstract class “memaksa” subclass nya untuk mengimpelementasikan seluruh abstract method

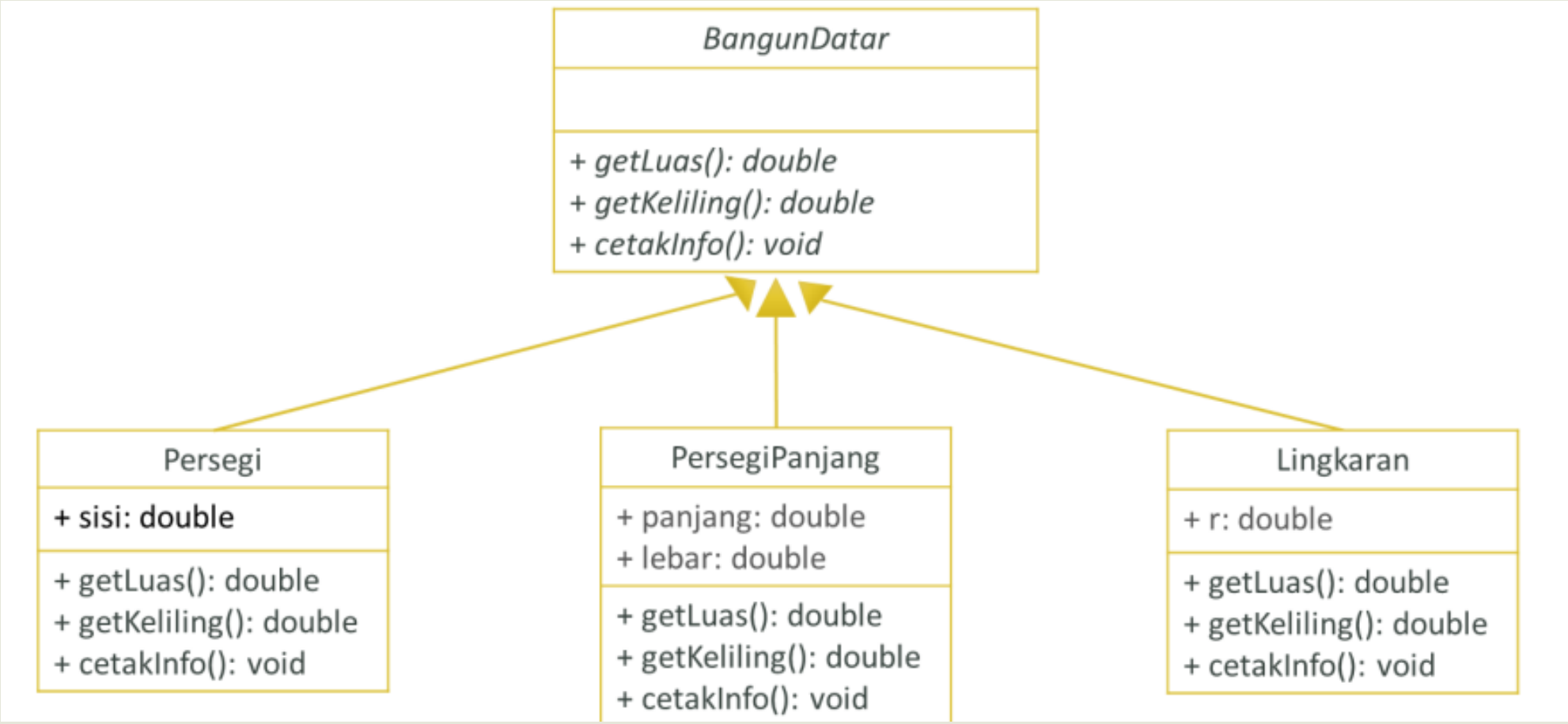
NOTASI CLASS DIAGRAM

- Secara umum sama dengan notasi *concrete class*
- Nama kelas dicetak miring atau ditambah anotasi <<abstract>> di atas nama kelas
- Abstract method juga dicetak miring



Kesimpulan

- Abstract class dapat digunakan untuk mencegah suatu class diinstansiasi atau dibuat objeknya
- Abstract class umumnya digunakan sebagai generalisasi/superclass pada class hierarki.
- Abstract class berlaku sebagai guideline untuk subclass dengan cara memaksa subclass untuk mengimplementasikan abstract method


BangunDatar
+ getLuas(): double
+ getKeliling(): double
+ cetakInfo(): void
Persegi
+ sisi: double
+ getLuas(): double
+ getKeliling(): double
+ cetakInfo(): void
PersegiPanjang
+ panjang: double
+ lebar: double
+ getLuas(): double
+ getKeliling(): double
+ cetakInfo(): void
Lingkaran
+ r: double
+ getLuas(): double
+ getKeliling(): double
+ cetakInfo(): void

Latihan

- Cari sebuah studi kasus dari abstract class kemudian gambarkan UML class diagramnya.
  - Superclass merupakan abstract class dengan minimal 1 concrete method dan 1 abstract method.
  - Terdapat minimal 2 subclass