

The background is a dark, textured surface with faint, light-colored sketches of various scientific and mathematical concepts. These include a globe in the upper left, a large letter 'V' in the top left, a telescope-like instrument on the left, a stack of books at the bottom left, a cross symbol, an open book with handwritten text at the bottom center, and a percentage sign and other symbols on the bottom right.

Abstract Class

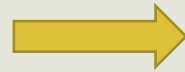
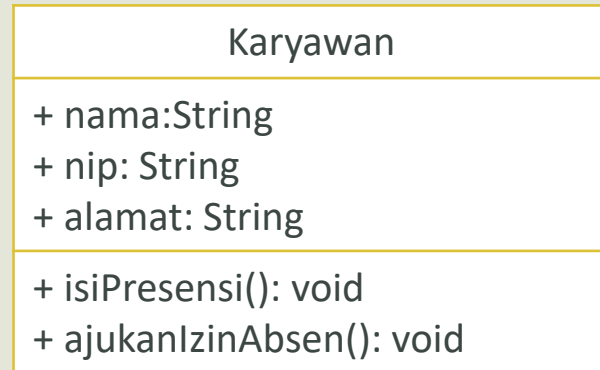
Tim Ajar PBO – JTI Polinema

TUJUAN

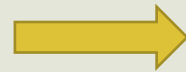
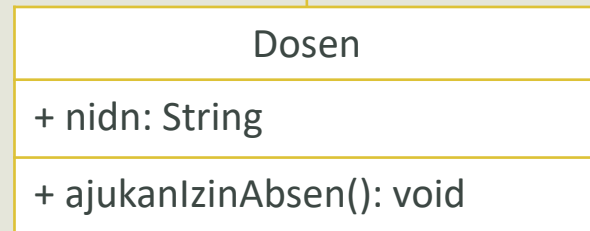
- Mahasiswa mampu memahami konsep abstract class
- Mahasiswa mampu membuat class diagram untuk studi kasus dengan abstract class

OUTLINE

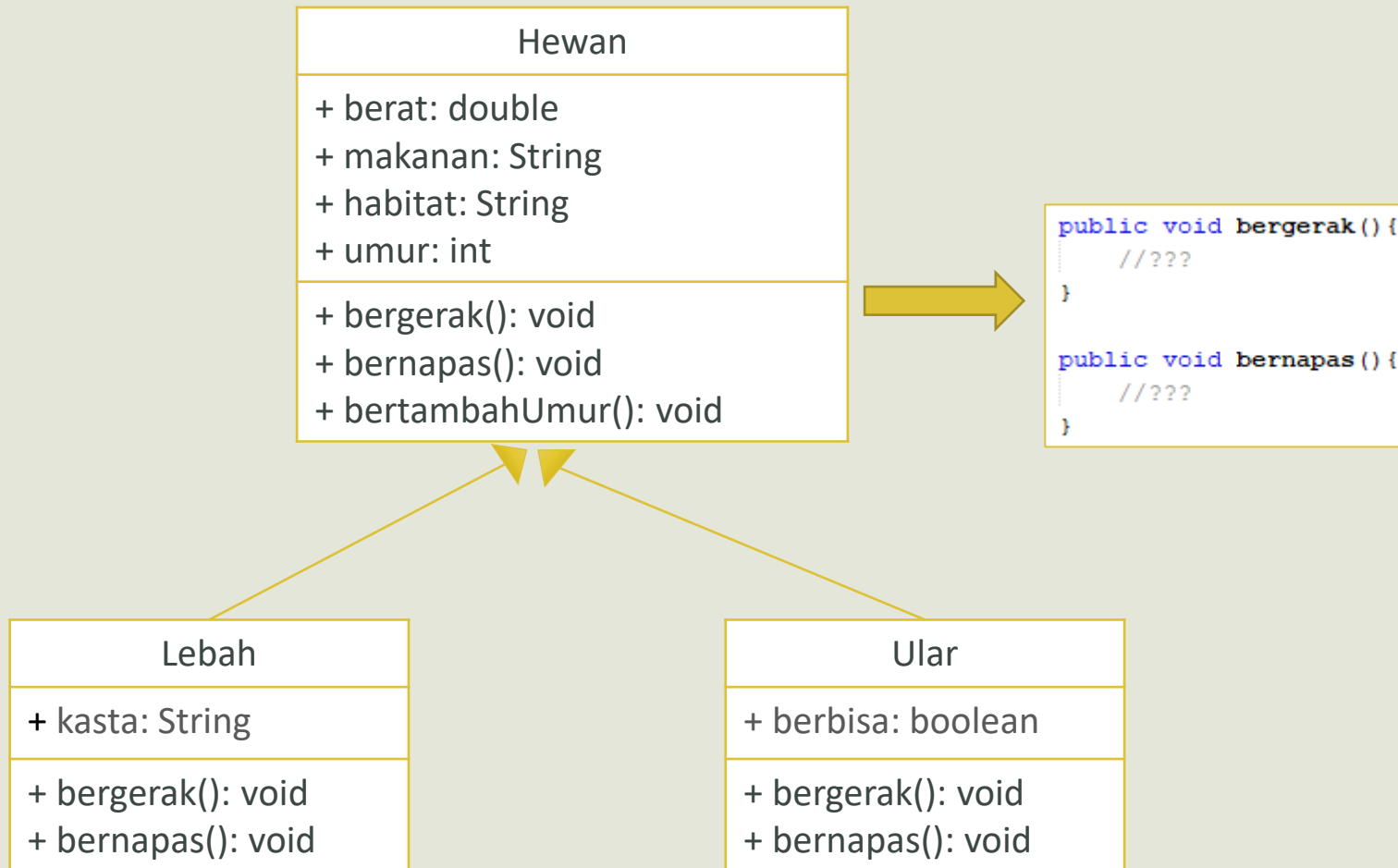
- Studi Kasus
 - Concrete Superclass
 - Abstract Superclass
- Definisi Abstract Class
- Abstract Method
- Syntax
- Notasi dalam Class Diagram
- Penggunaan



```
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");  
}
```



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

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

```
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");  
    }  
}
```



```
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

```
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;  
    }  
}
```



Abstract Class

```
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;  
    }  
}
```

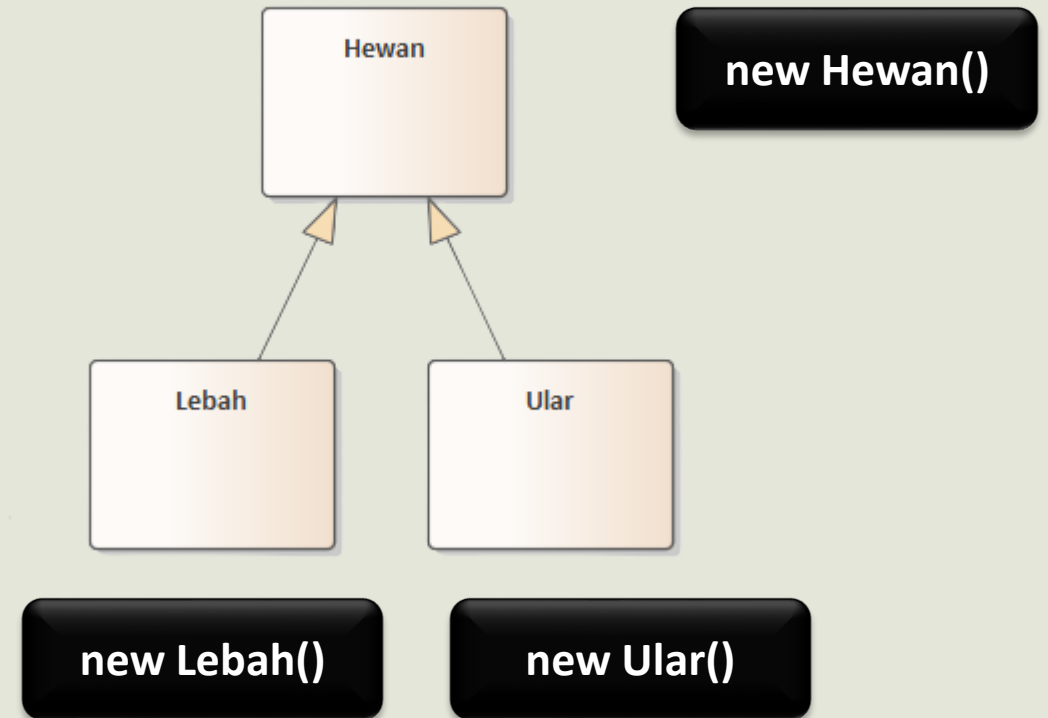
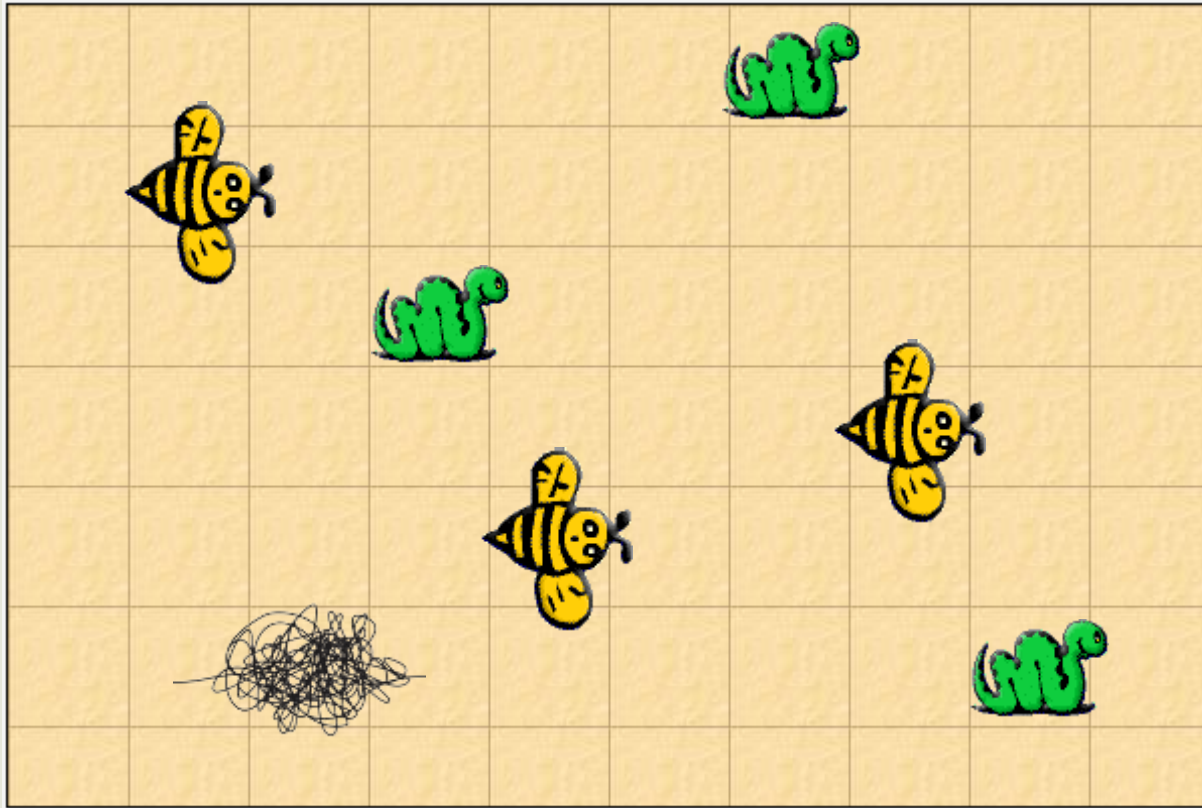


```
public static void main(String[] args) {  
    Hewan h = new Hewan();  
}
```

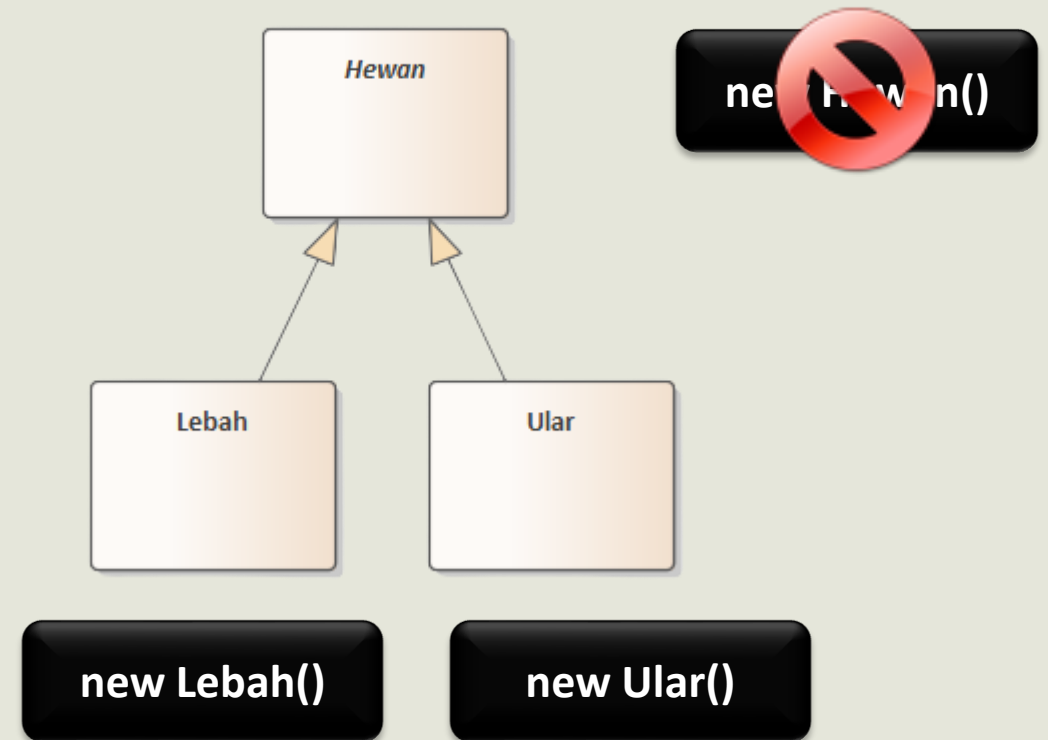
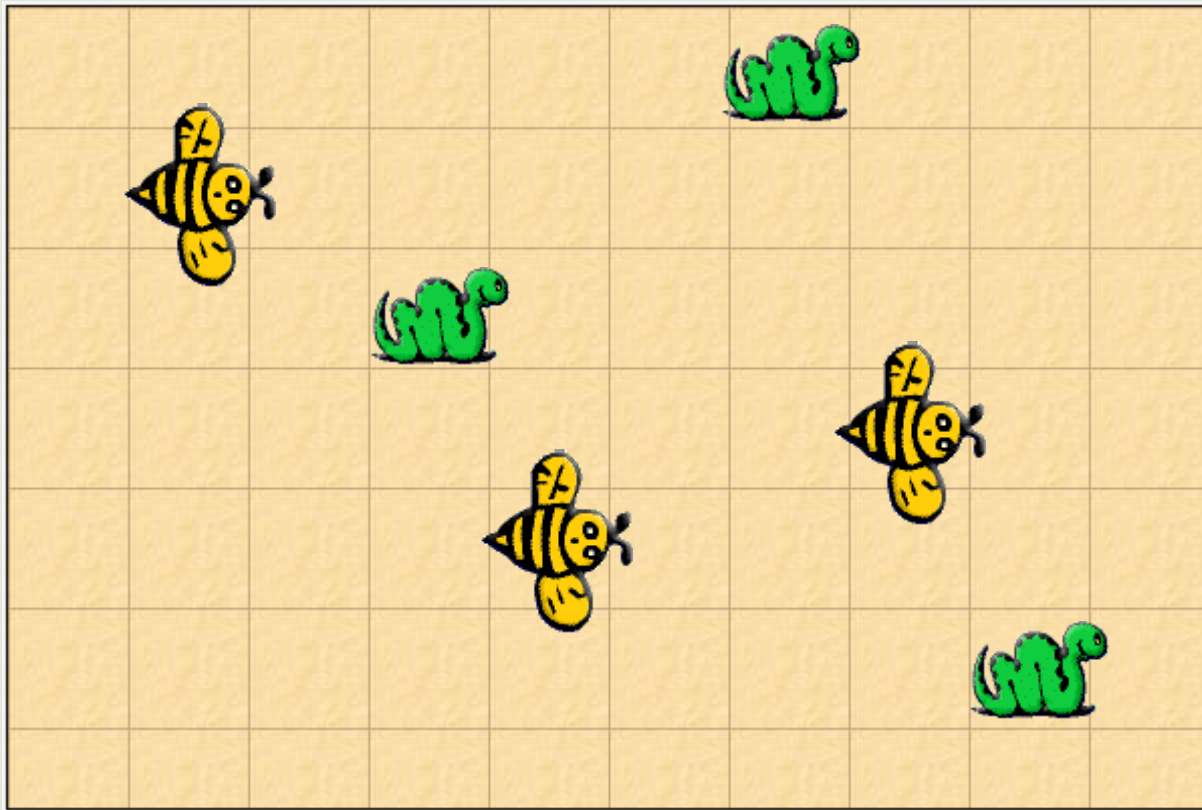


```
public static void main(String[] args) {  
    Hewan h = new Hewan();  
}
```

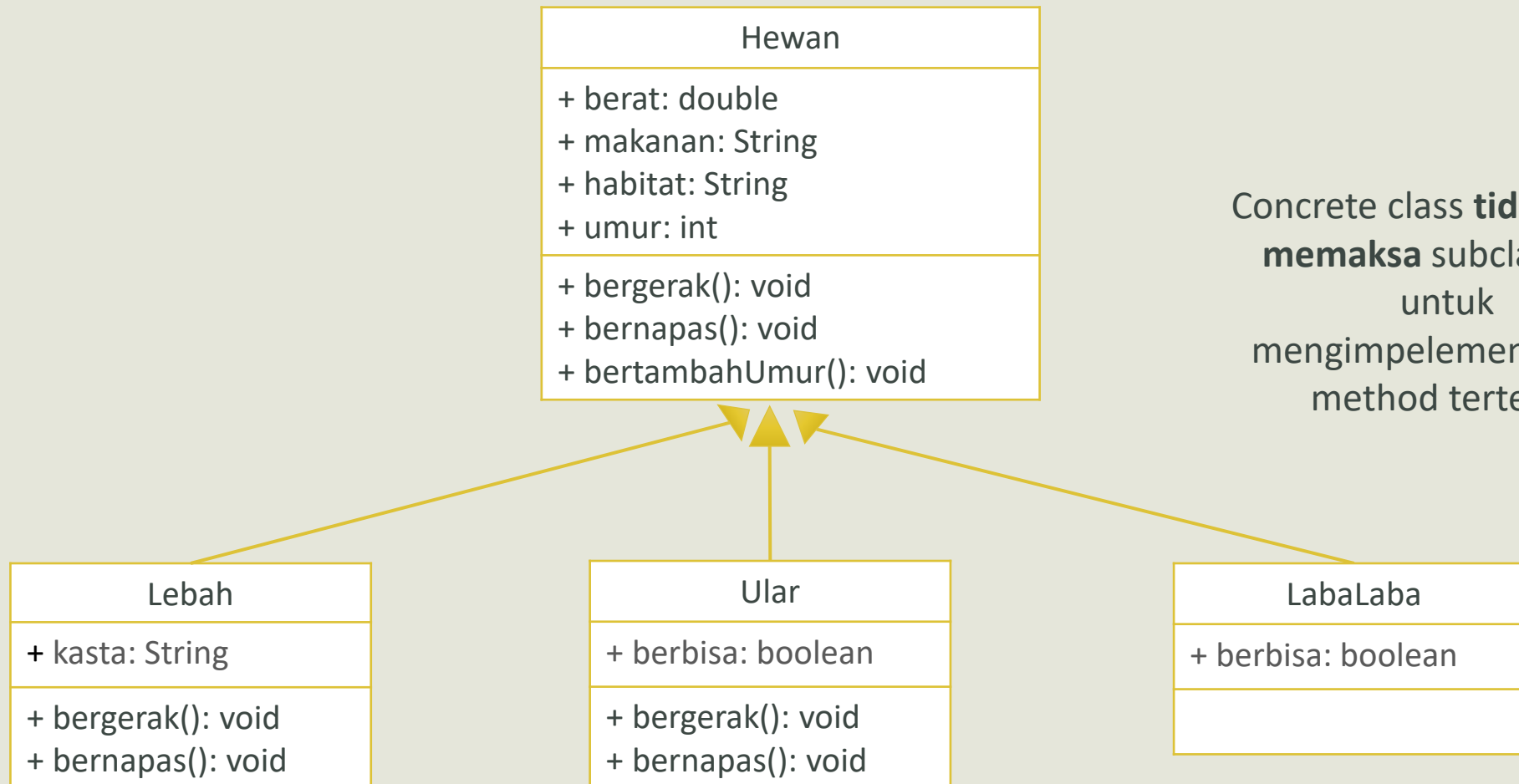

Concrete Superclass



Abstract Superclass

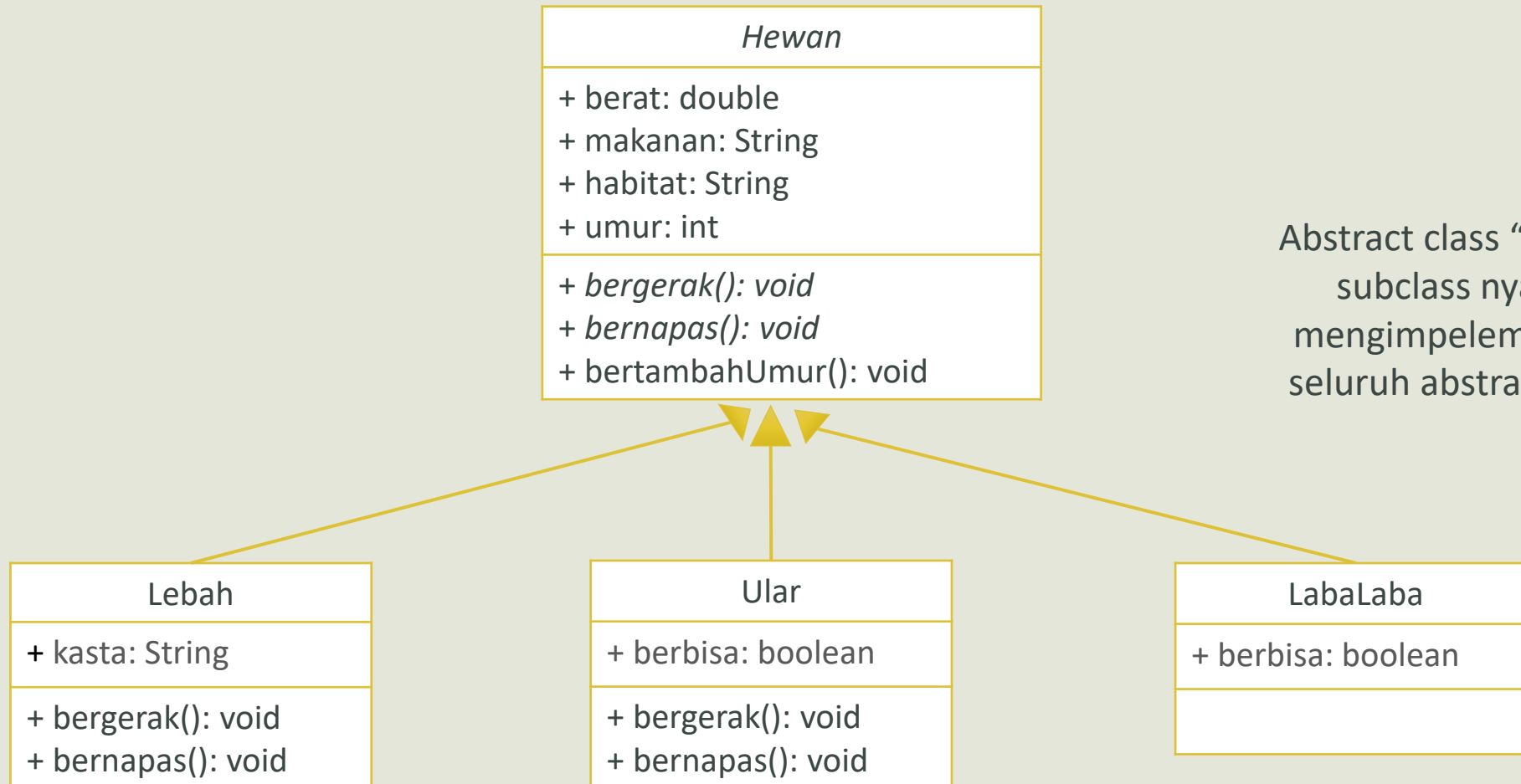


Concrete Superclass



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

Abstract Superclass



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



DEFINISI

- *Abstract class* merupakan class yang **tidak dapat diinstansiasi** namun dapat di-***extend***.
- Kegunaan:
 - Sebagai **generalisasi** subclass 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

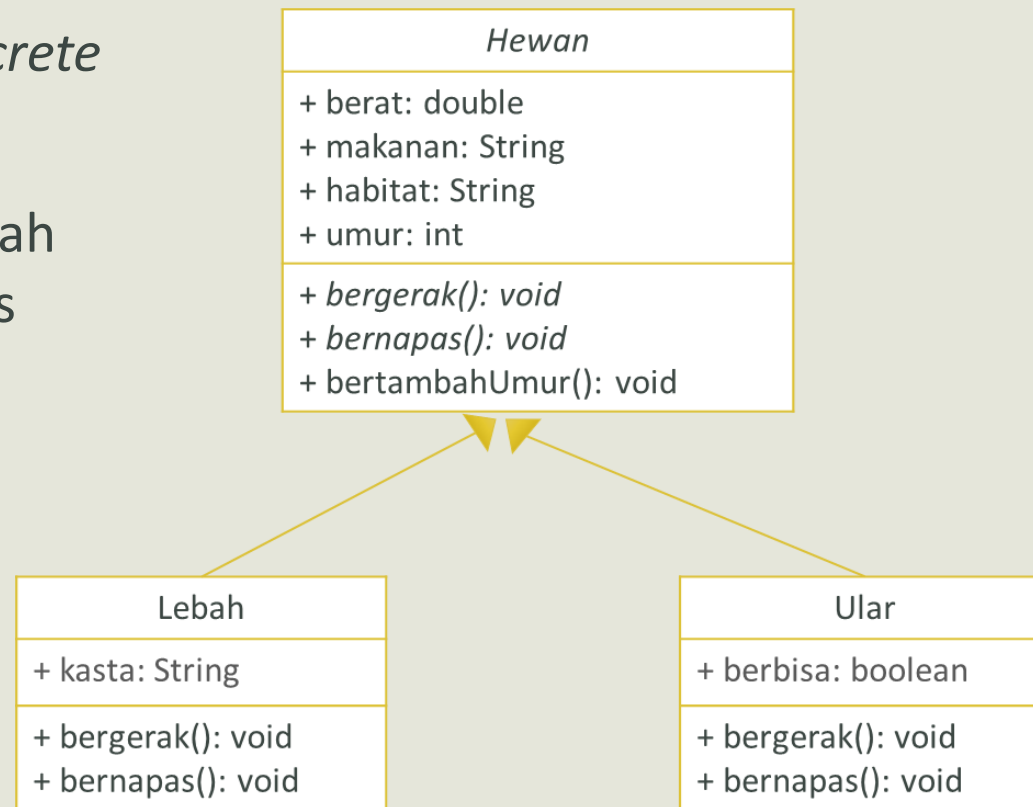
SYNTAX

- 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();  
}
```


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



Penggunaan Abstract Class

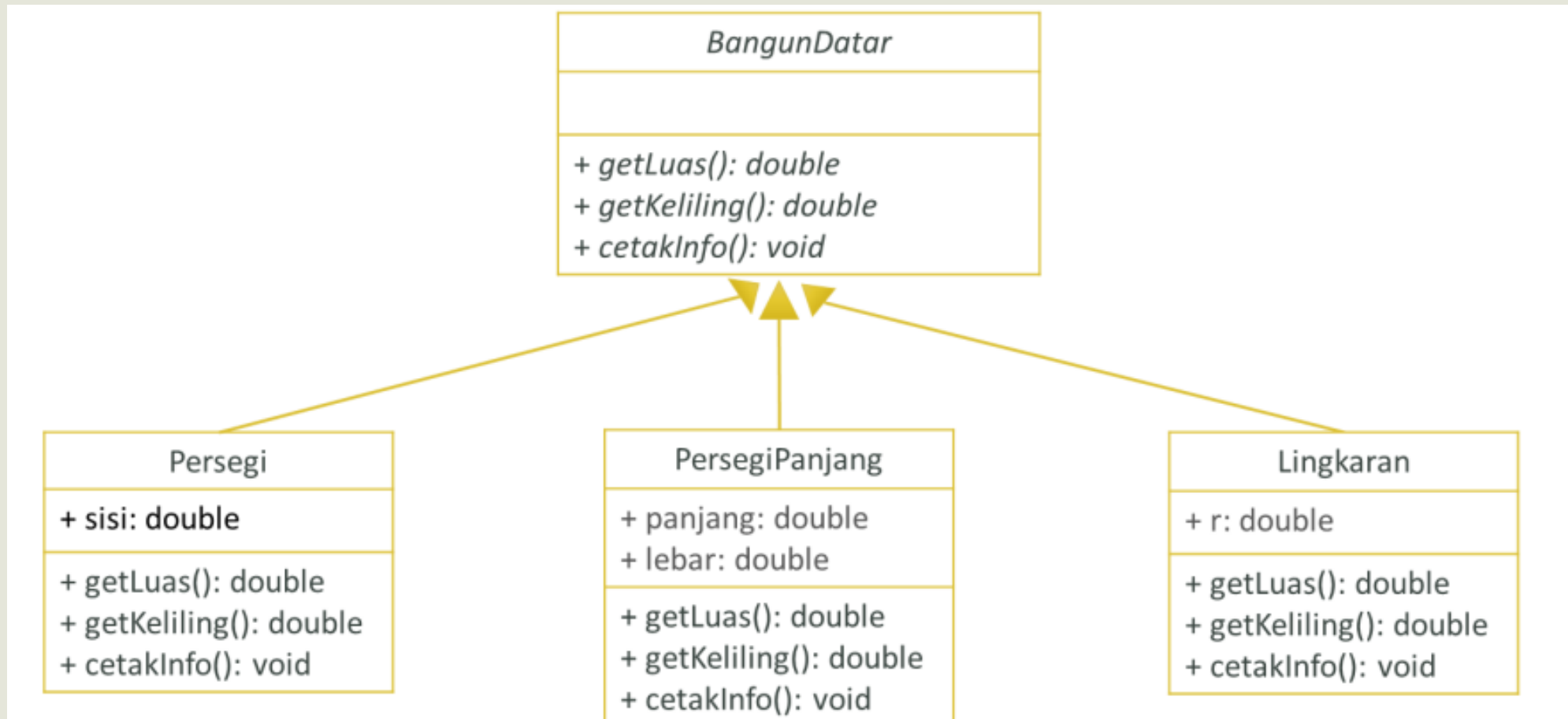
- Abstract class tidak dapat diinstansiasi (tidak dapat dibuat objectnya)
- Baris kode berikut akan memunculkan compilation error “*Hewan is abstract; cannot be instantiated*”

```
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

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



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