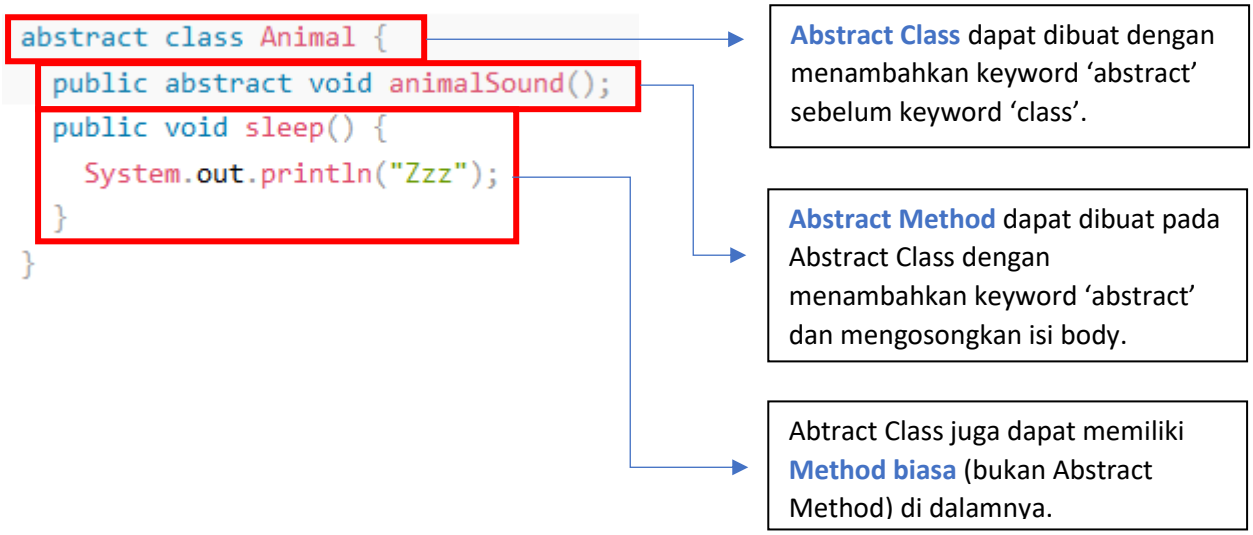
Interface and Abstract

1. Abstract Class and Method

Abstract Class merupakan kelas yang tidak dapat dibuat objeknya. *Abstract Class* masih berbentuk abstrak dan berperan sebagai kerangka dasar bagi kelas turunan (*Inheritance*). Sedangkan *Abstract Method* merupakan *method* yang hanya dapat dibuat pada *abstract class* dan tidak memiliki isi. *Abstract method* dapat diisi melalui kelas turunannya (*subclass*).



Abtract Class tidak dapat dibuat objeknya. Sehingga apabila kita menjalankan code seperti ini:

```
Animal myObj = new Animal();
```

Maka program tersebut ketika dijalankan akan menghasilkan *Error*. *Abstract class* dapat diakses oleh kelas turunannya dengan menggunakan konsep *Inheritance* dan *Polymorphism*.

Berikut contoh implementasi abstract class yang benar:

- Abstract Class

```
abstract class Animal {
  // Abstract method (does not have a body)
  public abstract void animalSound();
  // Regular method
  public void sleep() {
    System.out.println("Zzz");
  }
}
```

- Subclass

```
class Pig extends Animal {
  public void animalSound() {
    // The body of animalSound() is provided here
    System.out.println("The pig says: wee wee");
  }
}
```

Apabila program memiliki main sebagai berikut:

```
public static void main(String[] args) {
  Pig myPig = new Pig(); // Create a Pig object
  myPig.animalSound();
  myPig.sleep();
}
```

Maka program akan menghasilkan output berikut:

```
The pig says: wee wee
Zzz
```

2. Abstract Class vs Interface

Selain menggunakan *Abstract Class*, abstraksi pada Java dapat dibuat juga menggunakan Interface. Struktur *Interface* dapat dilihat pada gambar berikut:

```
interface Animal {
  public void animalSound(); // interface method (does not have a body)
  public void run(); // interface method (does not have a body)
}
```

Pembuatan *Interface* dilakukan menggunakan keyword 'interface' dan tidak menggunakn keyword 'class'. Interface tidak dapat membuat method yang memiliki body. Interface dapat diaskses menggunakan keyword 'implements' dengan contoh sebagai berikut:

```
class Pig implements Animal {
  public void animalSound() {
    // The body of animalSound() is provided here
    System.out.println("The pig says: wee wee");
  }
  public void sleep() {
    // The body of sleep() is provided here
    System.out.println("Zzz");
  }
}
```

Apabila program memiliki main sebagai berikut:

```
public static void main(String[] args) {
  Pig myPig = new Pig();  // Create a Pig object
  myPig.animalSound();
  myPig.sleep();
}
```

Maka program akan menghasilkan ouput berikut:

```
The pig says: wee wee
Zzz
```

Perbedaan Abstract Class dan Interface

Abstract class dan *Interface* meskipun sama – sama bertujuan untuk membuat abstraksi, namun juga memiliki beberapa perbedaan yang dapat dilihat pada table berikut:

Interface	Abstract class
Interface support multiple inheritance	Abstract class does not support multiple inheritance
Interface does'n Contains Data Member	Abstract class contains Data Member
Interface does'n contains Cunstructors	Abstract class contains Cunstructors
An interface Contains only incomplete member (signature of member)	An abstract class Contains both incomplete (abstract) and complete member
An interface cannot have access modifiers by default everything is assumed as public	An abstract class can contain access modifiers for the subs, functions, properties
Member of interface can not be Static	Only Complete Member of abstract class can be Static

Notes:

- Interface dapat melakukan *multiple inheritance* karena *program* dapat meng-*implement* lebih dari satu kelas. Sedangkan dengan menggunakan keyword ‘extend’ hanya dapat menurunkan satu kelas saja.