PERTEMUAN 14
ABSTRACT CLASS INTERFACE

A. TUJUAN PEMBELAJARAN

1. Mahasiswa memahami apa itu *abstract class* pada bahasa java oop
2. Mahasiswa memahami apa itu interface pada bahasa java oop
3. Mahasiswa dapat mengetahui aturan yang harus digunakan abstract class dan Interface
4. Mahasiswa dapat mempraktekan kapan penggunaan abstract class dan Interface

B. URAIAN MATERI

Abstract class adalah Sebuah class yang dideklarasikan menggunakan keyword "abstract". Dimana setidaknya mempunyai satu *abstract method* yang tidak mempunyai body (badan program) dan boleh mempunyai method yang non abstract. Dalam pertemuan ini kita akan belajar apa itu abstract class, mengapa kita menggunakannya dan aturan yang harus kita gunakan saat bekerja dengan bahasa program Java.

Contoh dibawah ini adalah abstract class memiliki abstract method yang tidak memiliki body

```
   abstract class hewan{
   //abstract method no body
        public abstract void suara();
}
```

Contoh dibawah ini adalah abstract class memiliki method non abstract atau memiliki *body*

```
Abstract class hewan{
  // method
     abstract void sound();
    void eat(){
        System.out.println ("Kucing  ")
    }
}
```

1. Mengapa menggunakan *abstract class*

Katakanlah kita memiliki class Hewan yang memiliki method suara() dan subclassnya (lihat pertemuan inheritance) seperti Kuda, Kucing, burung dll. Karena suara hewan berbeda dari satu hewan ke hewan lainnya, tidak ada gunanya kalau kita menerapkan metode abstract ini di parent class. karena setiap *child class* harus mengganti method suara untuk memberikan detail implementasinya sendiri, seperti kelas kuda akan bersuara "meringkik" , kelas kucing akan bersuara "meong".

Jadi ketika kita tahu bahwa semua child class hewan harus menimpa *method* ini, method abstract akan menjadi pilihan yang baik karena dengan membuat metode abstract, memaksa semua sub class untuk mengimplementasikan metode ini (jika tidak, kita akan mendapatkan kesalahan kompilasi),

Karena class Hewan memiliki method *abstract* suara(), kita harus mendeklarasikan dengan keyword abstract pada class ini.

Sekarang setiap hewan harus memiliki suara, dengan membuat *method* abstract, mewajibkan child class untuk memberikan detail implementasi metode ini. Dengan cara ini kita pastikan bahwa setiap hewan memiliki suara.

2. Aturan penggunaan abstract class pada bahasa java

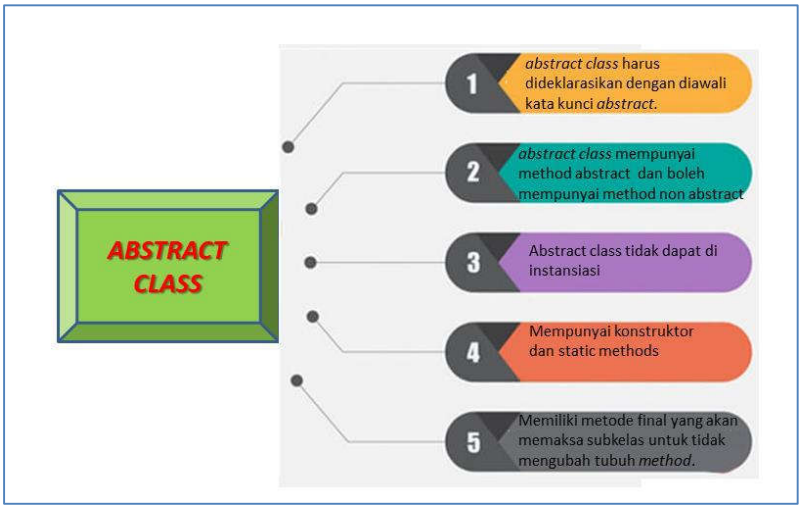
a. *abstract class* harus dideklarasikan dengan diawali kata kunci *abstract.*

b. *abstract class* setidaknya harus mempunyai method abstract (tidak mempunyai body) dan boleh mempunyai method non abstract (mempunyai body atau method konkrit)

c. Abstract class tidak dapat di instansiasi

d. Dapat mempunyai konstruktor dan static methods .

e. Memiliki metode final yang akan memaksa subkelas / *child class* untuk tidak mengubah tubuh *method*.



Gambar 14. 1 aturan penggunaan abstract class

3. Bentuk Umum pendeklarasian abstract class

```
//Deklarasi menggunakan  keyword abstract
abstract class A{

//dibawah ini  abstract method
   abstract void  MethodAbstract ();

   //dibawah ini method konkrit dengan body
   void  MethodNonAbstract(){
```

```
            //isi body
        }
    }
```

Kelas abstrak menguraikan metode tetapi tidak harus mengimplementasikan semua metode.

Contoh program sederhana Abstract class dengan method abstract

```
    abstract class abstrakhewan {
    // method abstract
        public abstract void suara();
     }
```

```
public class kucingabstrak extends hewan {

    public void suara(){
     System.out.println("Suara kucing mengeong");
   }
    public static void main(String[] args) {
        // TODO code application logic here
        hewan objek=new kucingabstrak();
        objek.suara();

    }

}
```

Output :

```
Suara kucing mengeong
```

Contoh program sederhana Abstract class dengan method abstract dan method non abstract

```
abstract class abstrakhewan {
       // method abstract
    public abstract void suara();
      // method non abstract
    public void suara2(){
      System.out.println(" ini method konkrit dari
parent class");
    }
}
```

```
public class kucingabstrak extends abstrakhewan {
     public void suara(){
     System.out.println(" Suara kucing mengeong");
}
     public static void main(String[] args) {
     // TODO code application logic here
             abstrakhewan objek=new kucingabstrak();
             objek.suara();
                   objek.suara2();
    }
   }
```

Output :

```
Suara kucing mengeong
ini method konkrit dari parent class
```

4. Interface pada java

Interface mirip seperti class tetapi itu bukan class. Interface memiliki method dan attribute atau variabel seperti class tetapi *method* yang dideklarasikan dalam *Interface* secara default *abstract* (hanya method tanpa isi atau body, seperti metohd abstract). variabel yang dideklarasikan dalam *interface* bersifat publik, statis & final secara default.

5. Mengapa menggunakan Interface

Seperti disebutkan di atas *interface* digunakan untuk abstraksi secara penuh. Karena method tidak memiliki isi atau body, metode tersebut harus diimplementasikan oleh class sebelum kita dapat mengaksesnya. Class yang mengimplementasikan *interface* harus mengimplementasikan semua metode *interface* itu. Juga, bahasa pemrograman java tidak memungkinkan kita untuk memperluas lebih dari satu class, Namun kita dapat mengimplementasikan lebih dari satu interface di class kita.

Ada 3 alasan mengapa menggunakan interface

a. Dapat digunakan untuk mencapai abstraksi.
b. Dengan interface , kita dapat mendukung fungsi pewarisan berganda.
c. Dapat digunakan untuk mencapai Objek independen.

6. Bentuk Umum penulisan Interface

```
interface InterfaceKu
{
   /*  Semua    methods  adalah  public  abstract  by
default
    * Seperti yang kita lihat method tidak membpunyai
body
```

```
    */
   public void method1();
   public void method2();
}
```

Contoh program menggunakan *Interface*

Ini adalah bagaimana sebuah class mengimplementasikan sebuah *Interface*. *class* harus menyediakan body dari semua method yang dideklarasikan dalam interface atau dengan kata lain bahwa class harus mengimplementasikan semua method *interface*.

```
public interface printable {
    void method1();
}
```

```
public class demo1 implements printable{
public void method1(){
    System.out.println("Hello apa kabar ini interface
");
}
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        printable objek=new demo1();
        objek.method1();
    }

}
```

Output :

```
     Hello apa kabar ini interface
```

Contoh Interface dengan 2 *method*

```
public interface Interfaceku {

     public void method1();

     public void method2();

}
```

```
  . public class demoInterface implements Interfaceku
{

    public void method1(){

        System.out.println("implementasi          dari
method1");


}

    public void method2(){

        System.out.println("implementasi          dari
method2");

    }

    /**

     * @param args the command line arguments

     */

    public static void main(String[] args) {

        // TODO code application logic here

        Interfaceku objeknya=new demoInterface();

        objeknya.method1();
```

```
        objeknya.method2();
    }


}
```

Output :

```
implementasi dari  method1
implementasi dari  method2
```

7. Perbedaan antara Abstract class dan interface pada Bahasa Java

Kita telah membahas abstract class dan interface. Berikut ini adalah tabel perbedaan antara *abstract class dan interface :*

Tabel 14. 1 Perbedaan Abstract dan Interface

No	*Abstract class*	*Interface*
1	abstract class hanya dapat meng exentend satu class atau satu class abstract pada satu waktu	*Interface* dapat meng *extend* sejumlah interface pada satu waktu
2	abstract class dapat meng extend method abstract dan method non abstract	*Interface* hanya dapat mengextend abstact *method*
3.	Pada *abstract class* wajib menggunakan keyword "abstract" pada saat dideklarasikan method sebagai abstrak	Pada interface penggunaan keyword "abstract" adalah opsional saat dideklarasikan method tersebut sebagai abstrak
5	*Abstract class* dapat memiliki method abstract sebagai protect dan public	Interface hanya dapat memiliki method abstrak public
6	*abstract class* dapat mempunyai variable static, final atau *static final*	interface hanya dapat mempunyai variable public static final (constant)

1. **Contoh program untuk perbedaan antara abstract class dan interface**

abstract class hanya dapat meng exentend satu class atau satu class abstract pada satu waktu

```
class Example1{
   public void display1(){
      System.out.println("display1 method");
   }
}
abstract class Example2{
   public void display2(){
      System.out.println("display2 method");
   }
}
abstract class Example3 extends Example1{
   abstract void display3();
}
```

```
class Example4 extends Example3{
   public void display3(){
      System.out.println("display3 method");
   }
}
class Demo{
   public static void main(String args[]){
       Example4 obj=new Example4();
       obj.display3();
   }
}
```

Output:

```
display3 method
```

Interface dapat meng *extend* sejumlah interface pada satu waktu

```
//first interface
interface Example1{
    public void display1();
}
//second interface
interface Example2 {
    public void display2();
}
//This interface is extending both the above interfaces
interface Example3 extends Example1,Example2{
}
```

```
class Example4 implements Example3{
    public void display1(){
        System.out.println("display2 method");
    }
    public void display2(){
        System.out.println("display3 method");
    }
}
```

```
class Demo{
    public static void main(String args[]){
        Example4 obj=new Example4();
        obj.display1();
    }
}
```

Output:

```
display2 method
```

C. LATIHAN/TUGAS

1. Kembangkan contoh program Abstract Class dibawah ini, *method abstract* suara(), method non abstract suara2(). Pada *parrent class*

```
      abstract class abstrakhewan {
      // method abstract
          public abstract void suara();
          public void suara2(){
       System.out.println(" ini method konkrit dari
parent class");
    }
}
```

2. Buat child class **burung.java** meng extend dari abstrakhewan()

   Buat objek untuk memanggil method abstract nya.
   Output yang diharapkan adalah :

```
      Suara Burung mencicit
```

D. REFERENSI

Horstmann Cay S., (2011). *Big Java 4t^{h} Edition ,san jose university , united state Of America*. *RRD jefferson city publishing.*

Deitel Paul , Deitel Harvey, (2012) Java how to program eighth edition, pearson education, Boston Massachusetts , *USA, publishing as prentice hall.*

Rose Cristhoper, (2017), Java Succinctly Part 2, Morrisville, NC 27560, USA, Syncfusion, Inc.

Downey Allen B. , Mayfield Chris, (2017), Think Java, Needham, Massachusetts, USA, Green Tea Press

Hayes Helen, (2021), BeginnersBook.com, https://beginnersbook.com/java-tutorial-for-beginners-with-examples/, di akses pada tanggal 21 November 2021.

Sonoo Jaiswal , (2021) JavaTpoint offers college campus training , https://www.javatpoint.com/java-tutorial, diakses pada tanggal 1 Desember 2021