# LAPORAN RESMI
# MODUL IV
# ABSTRACT CLASS DAN ABSTRACT METHOD
# PEMROGRAMAN BERBASIS OBJEK

| | |
|---|---|
| NAMA | : ANISYAFAAH |
| N.R.P | : 220441100105 |
| DOSEN | : FIRMANSYAH ADIPUTRA, ST., M.Cs. |
| ASISTEN | : KUKUH COKRO WIBOWO |
| TGL PRAKTIKUM | : 14 APRIL 2023 |

**Disetujui : 05 Mei 2023**
**Asisten**

**KUKUH COKRO WIBOWO**
**21.04.411.00102**


**LABORATORIUM BISNIS INTELIJEN SISTEM**
**PRODI SISTEM INFORMASI**
**JURUSAN TEKNIK INFORMATIKA**
**FAKULTAS TEKNIK**
**UNIVERSITAS TRUNOJOYO MADURA**

# BAB I
# PENDAHULUAN

## 1.1 Latar Belakang

Pesatnya perkembangan teknologi membuat makin banyak pula aplikasi atau program yang digunakan untuk membantu masyarakat. Bukan hanya aplikasi produktif saja, aplikasi hiburan seperti games juga terus bertumbuhan. Semua aplikasi atau program tersebut dibuat dan dikembangkan menggunakan bahasa pemrograman yang berbeda. Meskipun begitu, cara berpikir atau algoritmanya akan tetap sama.

Salah satu pemrograman yang paling banyak digunakan yaitu Pemrograman Berorientasi Objek. Dalam Pemrograman Berorientasi Objek, kita dituntut untuk memahami dan memetakan masalah ke dalam class - class yang lebih kecil dan simpel agar solusi yang dibuat lebih spesifik. Selanjutnya, class - class tersebut akan saling berkomunikasi dan berkolaborasi untuk memecahkan masalah yang kompleks. Berikut ini adalah contoh pembuatan program dengan menggunakan abstract class dan abstract method.

## 1.2 Tujuan

- Mahasiswa mampu memahami konsep Abstract Class dan abstract method dalam Pemrograman Berorientasi Objek serta mampu mengimplementasikannya.

# BAB II
# DASAR TEORI

## 2.1 Konsep Abstract Class

Abstract class atau kelas abstrak adalah kelas yang terletak diposisi tertinggi dalam hierarki class. Class ini tidak dapat diinstansiasi karena masih bersifat abstrak. Class ini hanya berisi variabel umum dan deklarasi method tanpa detail penggunaannya (abstract method). Selanjutnya class yang menjadi turunan dari abstract class ini yang akan menentukan detail penggunaan method tersebut.

## 2.2 Deklarasi Abstrak Class

- Abstract Class dideklarasikan dengan cara sebagai berikut :

```
public abstract class NamaClassAbstrak
{
  // deklarasi variabel dan abstract method
  // definisi method tidak abstrak
}
```

- Abstract Class digunakan dengan cara sebagai berikut :

```
Public class KelasPengguna extends NamaClassAbstrak
{


}
```

## 2.3 Abstract Method

Abstract method adalah yang dideklarasikan tanpa body method dan digunakan menggunakan kata kunci abstract.

Contoh :

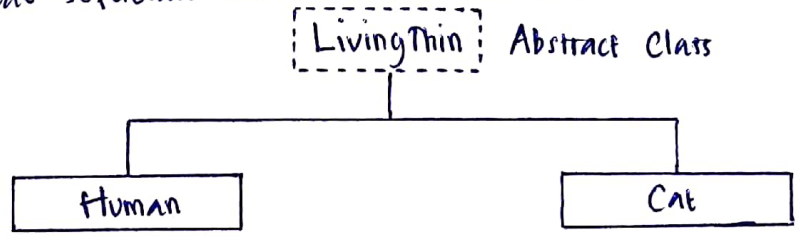
```
public abstract class NamaClassAbstrak
{
  abstract void namaMethod() ;
}
```

Java memiliki aturan - aturan dalam penggunaan method abstrak dan class abstrak sebagai berikut :

- Class yang didalamnya terdapat abstract method harus dideklarasikan sebagai abstract class.
- Abstract class tidak dapat diinstansi, tetapi harus diturunkan
- Sebuah class dapat dideklarasikan sebagai abstract class meskipun class tersebut tidak memiliki abstract method
- Abstract method tidak boleh mempunyai body method dan demikian juga sebaliknya bahwa method yang tidak ditulis body methodnya maka harus dideklarasikan sebagai abstract method.

## Contoh Program

Kita akan membuat superclass bernama LivingThin. Class ini mempunyai method tertentu seperti breath, eat, sleep, dan walk. Akan tetapi, ada beberapa method di dalam superclass yang sifatnya tidak dapat digeneralisasi. Kita ambil contoh, method walk. Tidak semua kehidupan berjalan (walk) dalam cara yang sama. Ambil manusia sebagai misal, kita manusia berjalan dengan dua kaki, dimana kehidupan lainnya seperti kucing. berjalan dengan empat kaki. Akan tetapi, beberapa ciri umum dalam kehidupan sudah biasa, itulah kenapa kita inginkan membuat superclass umum dalam hal ini.




Dalam program ini ada,

1. Abstract class yaitu
   - Abstract class LivingThin
2. Subclass yaitu
   - Class Human
   - Class Cat

1 Class yang digunakan untuk menjalankan program yaitu
- Class Run

Membuat abstract class LivingThin :

```
public abstract class LivingThin
{
   public void breath() {
      System.out.println ("Makhluk hidup bernapas ... ");
   }
   public void eat() {
      System.out.println ("Makhluk Hidup Makan ... ");
   }
   /**
    * abstract method walk
    * Kita ingin method ini di-overridden oleh subclass
    */
   public abstract void walk();
}
```

Ketika class meng-extend class abstract LivingThin, diberikan untuk override method abstract walk() seperti gambar dibawah ini.

```
public class Human extends LivingThin
{
   // membuat method
   public void name() {
      System.out.println ("Manusia");
   }
   // mengoverride method
   public void walk() {
      System.out.println ("Manusia berjalan menggunakan dua kaki");
   }
}
```

Jika class Human tidak dapat override method walk, kita akan menemui pesan berikut ini.

Human is not abstract and does not override abstract method walk() in LivingThin

Sama seperti class Human, class Cat juga ketika meng-extend class abstract LivingThin maka harus mengoverride method walk jika tidak akan terjadi error.

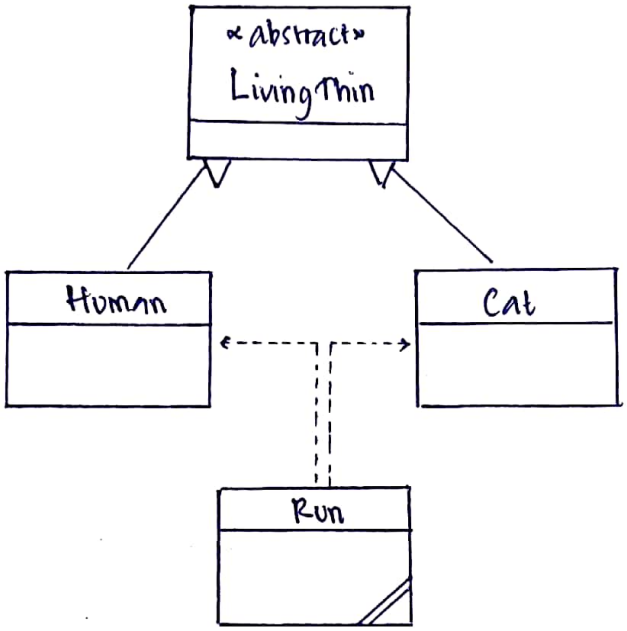
```
public class Cat extends LivingThin
{
  //method
  public void name() {
    System.out.println("Kucing");
  }
  //mengoverride method
  public void walk() {
    System.out.println("Kucing berjalan menggunakan empat kaki");
  }
}
```

Membuat class bernama Run yang digunakan untuk menjalankan program dengan membuat Objek dari masing-masing class kemudian memanggil semua methodnya seperti pada gambar di bawah ini.

```
public class Run
{
  public static void main() {
    // membuat objek human dari class Human
    Human Human = new Human();
    // membuat objek kucing dari class kucing
    Cat cat = new Cat();
    human.name();
    human.eat();
    human.breath();
    human.walk();
```

```
        Cat.name();
        Cat.eat();
        Cat.breath();
        Cat.walk();
    }
}
```

Setelah selesai membuat semua classnya maka akan seperti gambar di bawah ini.




Running Program
Manusia
Makhluk hidup pasti makan...
Makhluk hidup pasti bernafas...
Manusia berjalan menggunakan dua kaki
Kucing
Makhluk hidup pasti makan...
Makhluk hidup pasti bernafas...
Kucing berjalan menggunakan empat kaki

# BAB III
# IMPLEMENTASI

## 3.1 Soal

Buatlah aplikasi penjumlahan, pengurangan, perkalian dan pembagian menggunakan konsep abstract class dan polimorfisme dengan memasukkan parameter bilangan A = 6.5 dan bilangan B = 0.5!

## 3.2 Jawaban

a) Hasil

```
PENJUMLAHAN
6.5 + 0.5 = 7.0
--------------------

PENGURANGAN
6.5 - 0.5 = 6.0
--------------------

PERKALIAN
6.5 * 0.5 = 3.25
--------------------

PEMBAGIAN
6.5 / 0.5 = 13.0
--------------------

BUILD SUCCESSFUL (total time : 0 seconds)
```

b) Source Code

1) OperasiBilanganAbs

```
package Project4;
  abstract class OperasiBilanganAbs {
    protected double a,b,c;
    protected abstract void set_A (double A);
    protected abstract void set_B (double b);
    protected abstract void set_C ();
```

```
    protected abstract double get_A();
    protected abstract double get_B();
    protected abstract double get_C();
    protected abstract void tampil();
}
```

2) OperasiPenjumlahan

```
package Project4;
    class OperasiPenjumlahan extends OperasiBilanganAbs {
    @Override
    protected void set_A (double a) {
    this.a = a;
    }
    @Override
    protected void set_B (double b) {
    this.b = b;
    }
    @Override
    protected void set_C() {
    this.c = a + b;
    }
    @Override
    protected double get_A() {
    return a;
    }
    @Override
    protected double get_B() {
    return b;
    }
    @Override
    Protected double get_C() {
    return c;
    }
    @ Override
    Protected void tampil() {
```

```
System.out.println("PENJUMLAHAN");
System.out.println(get_A() + " + " + get_B() + " = " + get_C()
);
System.out.println("--------------------");
}
}
```

3) Operasi Pengorangan

```
package Projecty;
class OperasiPengurangan extends OperasiBilanganAbs {
@Override
Protected void set_A (double a) {
this.A = A;
}
@Override
protected void set_B (double b) {
this.b = b;
}
@Override
protected void set_C () {
this.c = a - b;
}
@Override
Protected double get_A() {
return A;
}
@Overide
protected double get_B() {
return b;
}
@Override
Protected double get_C () {
return C;
}
```

```
@Override
Protected void tampil() {
System.out.println ("PENGURANGAN");
System.out.println (get_A() + "-" + get_B() + "=" + get_C());
System.out.println ("=====================");
}
}
```

4) OperasiPerkalian

```
Package Project4;
class OperasiPerkalian extends OperasiBilanganAbs {
@Override
protected void set_A (double a) {
this.a = a;
}
@Override
Protected void set_B (double b) {
this.b = b;
}
@Override
protected void set_c() {
this.c = a*b;
}
@Override
protected double get_A() {
return a;
}
@Override
protected double get_B() {
return b;
}
@Override
protected double get_c() {
return c;
}
```

```
@Override
Protected void tampil() {
System.out.println("PERKALIAN");
System.out.println(get_A()+" * "+get_B()+" = "+get_c());
System.out.println("----------");
}
}
```

5) OperasiPembagian

```
Package Project4;
class OperasiPembagian extends OperasiBilanganAbs {
@Override
Protected void set_A(double a) {
this.a = a;
}
@Override
protected void set_B(double b) {
this.b = b;
}
@Override
Protected void set_c() {
this.c = a/b;
}
@Override
Protected double get_A() {
return a;
}
@Override
Protected double get_B() {
return b;
}
@Override
Protected double get_c() {
return c;
}
```

```
    @Override
    Protected void tampil() {
    System.out.println ("PEMBAGIAN");
    System.out.println (get_A()+ "/" + get_B () + "=" + get_C());
    System.out.println ("--------------------");
    }
  }
```

6) OperasiBilanganAbs Cetak

```
Package Prosecta;
  Public class OperasiBilanganAbsCetak {
  Private static void Main (OperasiBilanganAbs[] OBA, double
  a, double b) {
    OBA[0] = new OperasiPenjumlahan ();
    OBA[1] = new OperasiPengurangan ();
    OBA[2] = new OperasiPerkalian ();
    OBA [3] = new OperasiPembagian ();

    for (int x = 0; x < OBA.Length; x++) {
    OBA[x].set_A (a);
    OBA[x].set_B (b);
    OBA [x].set_C ();
    OBA [x].tampil ();
    }
    }
    Public static void main (String[] args) {
    OperasiBilanganAbs[] A = new OperasiBilanganAbs [4];
    Main (A, 6.5, 0.5);
    }
  }
```

c) Penjelasan

Pada program abstract class di atas, terdapat beberapa file yang masing-masing berisi perintah operasi aritmatikanya. Selain itu, file untuk atribut dan main filenya juga diletakkan

berbeda. Sehingga pada main filenya hanya berisi permitah untuk memanggil file - file yang telah dibuat untuk ditampilkan outputnya. File - file operator aritmatika menggunakan perintah override dan set, serta get. Operator yang digunakan juga diserupakan dengan nama file atau nama classnya.

# BAB IV
# PENUTUP

## 4.1 Analisa

Dari hasil praktikum, praktikan menganalisa bahwa dalam membuat sebuah aplikasi penjumlahan, pengurangan, perkalian, dan pembagian dengan menggunakan konsep abstract dan polimorfisme dengan parameter maka diperlukan suatu abstract class yang di dalamnya telah diberikan variabel dengan tipe data double. Di dalam class ini tentunya terdapat beberapa perintah seperti perintah set, get, penggunaan inheritance, penggunaan override. Program ini juga menggunakan beberapa operator aritmatika. Nilai yang diproses di dalamnya telah di atur sehingga program bersifat statis.

## 4.2 Kesimpulan

1. Abstract class atau kelas abstrak adalah kelas yang terletak di posisi tertinggi dalam hierarki class dan tidak dapat diinstansiasi karena masih bersifat abstrak.
2. Sebuah class dapat dideklarasikan sebagai abstract class meskipun class tersebut tidak memiliki abstract method.
3. Program yang telah diimplementasikan di atas menggunakan abstract class, inheritance, override, dan operator aritmatika.