Modul 8
Abstraction

A) Pokok Bahasan

1. Abstraction
2. Interface

B) Tujuan

1. Mahasiswa Mampu Memahami Konsep Abstraction Pada PBO

C) Dasar Teori

1. Abstraction

   Abstraction adalah konsep menyembunyikan detail implementasi dari suatu method atau class

   a) Abstract Method

   Abstract Method yaitu method yang dideklarasikan, namun tidak memiliki body atau badan

   Abstract Method hanya boleh dideklarasikan pada Abstract Class atau interface

```
Deklarasi :
[access_modifier] abstract [return_type] [nama_method]
([parameter]);

Contoh :
public abstract void katakanHai(String kepada);
```

   b) Abstract Class

   Abstract Class adalah class yang memiliki satu atau lebih abstract method

   Abstract Class tidak dapat dibuat objek secara langsung

   Apabila abstract class di-extends oleh class lain, maka class tersebut wajib mengimplementasikan semua abstract method yang dimiliki oleh abstract class tersebut

```
Deklarasi :
[access_modifier] abstract class [nama_class] {
```

```
    [body_class]
}

Contoh :
abstract class Sapa {
    public abstract void katakanHai(String kepada);
}
class Hai extends Sapa {
    // Wajib diimplementasikan di subclass
    public void katakanHai(String kepada) {
        System.out.println("Hai "+kepada);
    }
}
```

2. Interface

   Interface memiliki kemiripan dengan abstract class, namun interface hanya boleh berisi abstract method, static method, dan static konstanta

   Method pada interface secara otomatis memiliki status abstract tanpa perlu ditambah keyword `abstract`

   Sama seperti abstract class, Interface tidak dapat dibuat menjadi objek secara langsung

   Berbeda dengan class, interface tidak bisa diwariskan tetapi bisa diimplementasikan. Untuk mengimplementasikan interface pada class, gunakan keyword `implements`

   Sebuah class boleh mengimplementasikan lebih dari satu interface

```
Deklarasi :
[access_modifier] interface [nama_interface] {
    [body]
}

Syntax :
[access_modifier] class [nama_class] implements [interfaces] {}

Contoh :
interface ElemenAngin {
    public void tiup();
}
interface TombolOnOff {
    public void on();
    public void off();
}
public class KipasAngin implements ElemenAngin, TombolOnOff {
    // Implementasi dari interface ElemenAngin
    public void tiup() {
        System.out.println("Wussh");
    }
    // Implementasi dari interface TombolOnOff
```

```
            public void on() {
               System.out.println(“Kipas Menyala”);
            }
            public void off() {
               System.out.println(“Kipas Mati”);
            }
        }
```

D) Praktik

Buatlah direktori dengan susunan sebagai berikut

```
/main
    Main.java
    Gundam.java
    OptimusPrime.java
/abstrak
    Kendaraan.java
    Motor.java
/interfes
    Burung.java
    Robot.java
```

file : abstrak/Kendaraan.java

```
package abstrak;

// Abstract Class
public abstract class Kendaraan {
    protected int jumlahBan;
    protected boolean mesinSedangMenyala = false;
    protected int bensin = 0;

    public void apakahMesinSedangMenyala() {
        if(mesinSedangMenyala) {
            System.out.println("Ya");
        } else {
            System.out.println("Tidak");
        }
    }
    // Abstract Method
    public abstract void isiBensin(String tipeBensin);
    public abstract void jalankan();
}
```

file : abstrak/Motor.java

```
package abstrak;

public class Motor extends Kendaraan {
```

```
    public Motor() {
        this.jumlahBan = 2;
    }

    public void isiBensin(String jenisBensin) {
        if (jenisBensin.equals("Pertalite")) {
            this.bensin += 2;
        } else if (jenisBensin.equals("Premium")) {
            this.bensin += 1;
        } else {
            System.out.println("Jenis Bensin tidak diketahui");
        }
    }

    public void jalankan() {
        if (this.nyalakanMesin()) {
            this.gasFull();
        }
    }

    public void gasFull() {
        System.out.println("Motor Jalan laju sekali");
    }

    public boolean nyalakanMesin() {
        if (this.bensin <= 0) {
            this.mesinSedangMenyala = false;
            System.out.println("Bensin Tidak cukup");
            return false;
        }
        this.bensin -= 1;
        this.mesinSedangMenyala = true;
        return true;
    }
}
```

file : interfes/Burung.java

```
package interfes;

public interface Burung {
    void terbang();
    void mendarat();
}
```

file : interfes/Robot.java

```
package interfes;

public interface Robot {
    void serang();
    void keluarkanLaser();
```

```
}
```

file : main/Gundam.java

```
package main;

public class Gundam implements Robot, Burung {
    private int power;
    private int ketinggian;

    public Gundam(int power) {
        this.power = power;
    }

    // Semua abstract method pada interface harus di Implementasikan
    @Override
    public void serang() {
        System.out.println("Gundam menyerang dengan kekuatan "+this.power);
    }
    @Override
    public void keluarkanLaser() {
        this.power++;
        System.out.println("Gundam mengeluarkan Laser");
    }
    @Override
    public void terbang() {
        this.ketinggian += 9999;
        System.out.println("Gundam Terbang menuju tak terbatas dan
melampauinya");
    }
    @Override
    public void mendarat() {
        this.ketinggian = 0;
        System.out.println("Gundam mendarat");
    }

    public int getKetinggian() {
        return this.ketinggian;
    }
}
```

file : main/OptimusPrime.java

```
package main;
import interfes.Robot;

public class OptimusPrime implements Robot {
    public void serang() {
        System.out.println("Optimus menyerang Dengan kekuatan Maximal");
    }
    public void keluarkanLaser() {
        System.out.println("Laser Optimus ketinggalan di kampus");
```

```
    }
}
```

file : main/Main.java

```java
package main;
import abstrak.*;
import interfes.*;

public class Main {

    public static void main(String []args) {
        // Kendaraan motorBapak = new Kendaraan();    // Error
        Motor motorBapak = new Motor();

        motorBapak.jalankan();
        motorBapak.apakahMesinSedangMenyala();

        motorBapak.isiBensin("Premium");
        motorBapak.jalankan();
        motorBapak.apakahMesinSedangMenyala();

        motorBapak.jalankan();
        motorBapak.apakahMesinSedangMenyala();

        Robot robotBapak;

        Gundam zBeta = new Gundam(20);

        robotBapak = zBeta;
        robotBapak.keluarkanLaser();
        robotBapak.serang();

        OptimusPrime optimus = new OptimusPrime();

        robotBapak = optimus;
        robotBapak.keluarkanLaser();
        robotBapak.serang();
    }
}
```