

**LAPORAN PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK**

**MODUL 5
“POLYMORPHISM”**



Disusun oleh
Bunga Laelatul Muna
NIM : 21102010

**PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
2023**

BAB 1 – TUJUAN PRAKTIKUM

Tujuan dari praktikum instalasi Java (IntelliJ IDEA) adalah untuk memberikan pengetahuan dan keterampilan kepada mahasiswa tentang Polymorphism

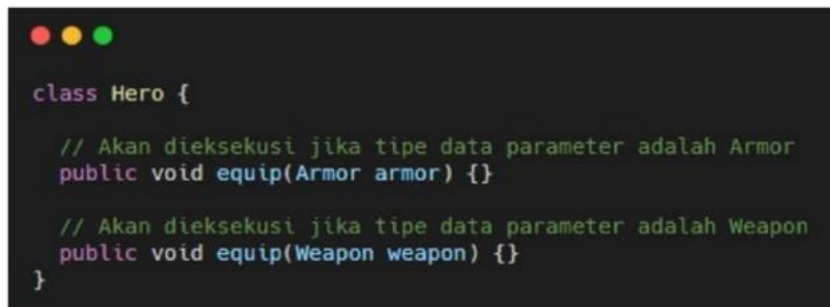
- Mahasiswa dapat memahami tentang konsep utama pbo yaitu polymorphism
- Mahasiswa dapat membuat polimorphism dalam program.

BAB II- DASAR TEORI

Polimorfisme merupakan konsep pembuatan method berbeda dengan nama yang sama. Polimorfisme dibagi menjadi dua, yaitu:

a. Overloading

Overloading adalah ketika di sebuah class ada 2 atau lebih method dengan nama sama namun masing-masing memiliki parameter berbeda. Walaupun namanya sama, program tetap mengerti harus menjalankan method yang mana berdasarkan parameter yang dimasukan saat pemanggilan method.



```
class Hero {  
    // Akan dieksekusi jika tipe data parameter adalah Armor  
    public void equip(Armor armor) {}  
    // Akan dieksekusi jika tipe data parameter adalah Weapon  
    public void equip(Weapon weapon) {}  
}
```

b. Overriding

Overriding adalah ketika sebuah Child class memiliki method dengan nama yang sama dengan Parent class-nya, namun dengan isi yang berbeda (parameter

boleh sama). Nantinya program akan memprioritaskan method milik Child class ini jika ternyata ketahuan Child class memiliki method yang di-Override dari Parent class-nya.

```

class Burung { // Parent class
    /*
     * Jika method ini dipanggil dari object Burung, method
     * yang ini yang akan dieksekusi ketika terbang() dipanggil
     */
    public void terbang() {
        System.out.println("Aku terbang!");
    }
}

class Penguin extends Burung { // Child class
    /*
     * Jika method ini dipanggil dari object Penguin, method
     * yang ini yang akan dieksekusi ketika terbang() dipanggil
     */
    public void terbang() {
        System.out.println("Aku gak bisa terbang :(");
    }
}

```

BAB III – GUIDED

Guided 1

- Source Code (Buah.Java)

```

//Bunga Laelatul Muna
// 21102010

package
com.Bunga_Laelatul_Muna.PBO.Pertemuan5.LatihanKelas;

public class Buah {
    // Atribut
    String nama;
    int jumlah;

    // Constructor
    public Buah() {}
    public Buah(String nama, int jumlah) {
        this.nama = nama;
        this.jumlah = jumlah;
    }

    // Method
    public void showInfo() {
        System.out.println("Nama buah : " + nama);
        System.out.println("Jumlah : " + jumlah);

        extraInfo();
    }

    public void extraInfo() {}
}

```

- **Source Code (Monokotil.java)**

```
package
com.Bunga_Laelatul_Muna.PBO.Pertemuan5.LatihanKelas;

public class Monokotil extends Buah{
    // Atributr
    String warna;
    int jumlahBiji;

    // Constructor
    public Monokotil(){}
    // Overloading
    public Monokotil(String warna, int jumlahBiji){
        this.warna = warna;
        this.jumlahBiji = jumlahBiji;
    }
    public Monokotil(String nama, int jumlah, String
warna, int jumlahBiji){
        super(nama, jumlah);
        this.warna = warna;
        this.jumlahBiji = jumlahBiji;
    }

    // Method
    // Overriding
    public void extraInfo(){
        System.out.println("Warna : " + warna);
        System.out.println("Jumlah Biji : " + jumlahBiji);
    }
}
```

- **Source Code (Main.Java)**

```
package
com.Bunga_Laelatul_Muna.PBO.Pertemuan5.LatihanKelas;
public class Main {
    public static void main(String[] args){
        // Object
        Monokotil mono = new Monokotil();
        mono.nama = "Alpukat";
        mono.jumlah = 5;
        mono.warna = "Hijau";
        mono.jumlahBiji = 1;

        mono.showInfo();
    }
}
```

- **SS Output**

```
Nama buah : Alpukat  
Jumlah : 5  
Warna : Hijau  
Jumlah Biji : 1
```

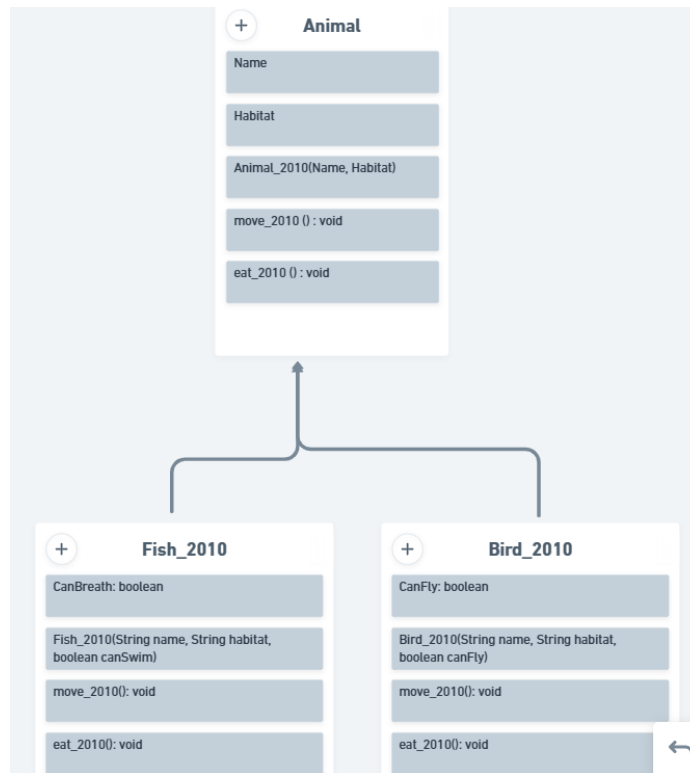
- **Penjelasan**

Ini program Polymorphism yang memiliki 1 parent dan 1 child. Buah sebagai parent memiliki constructor dengan 2 parameter yaitu nama dan jumlah yang kemudian ditampilkan di method showInfo. Lalu pada Monokotil sebagai child. Kelas ini memiliki dua atribut yaitu warna dan jumlahBiji, dan tiga konstruktor yaitu konstruktor tanpa parameter, konstruktor dengan parameter warna dan jumlahBiji, dan konstruktor dengan parameter nama, jumlah, warna, dan jumlahBiji. Kelas Monokotil juga memiliki metode extraInfo() yang meng-overriding metode yang sama pada kelas Buah dan menambahkan informasi mengenai warna dan jumlah biji.

BAB IV – UNGUIDED

UnGuided 1

- Diagram



- Source Code(Animal_2010.Java)

```
package com.Bunga_Laelatul_Muna.PBO.Pertemuan5.Tugas;
// Bunga Laelatul Muna
// 21102010

public class Animal_2010 {
    String name;
    String habitat;

    public Animal_2010(String name, String habitat) {
        this.name = name;
        this.habitat = habitat;
    }

    public void move_2010() { System.out.println(name + " bergerak"); }
    public void eat_2010() { System.out.println(name + " sedang makan"); }
}

package com.Bunga_Laelatul_Muna.PBO.Pertemuan5.Tugas;
// Bunga Laelatul Muna
```

```
// 21102010

public class Animal_2010 {
    String name;
    String habitat;

    public Animal_2010(String name, String habitat) {
        this.name = name;
        this.habitat = habitat;
    }

    public void move_2010() {
        System.out.println(name + " bergerak");
    }

    public void eat_2010() {
        System.out.println(name + " sedang makan");
    }
}
```

- Source Code (Bird_2010.java)

```
package com.Bunga_Laelatul_Muna.PBO.Pertemuan5.Tugas;
// Bunga Laelatul Muna
// 21102010

public class Bird_2010 extends Animal_2010 {
    boolean canFly;

    public Bird_2010(String name, String habitat, boolean canFly) {
        super(name, habitat);
        this.canFly = canFly;
    }

    /* === Implementasi Polymorphism ===
    Ini adalah salah satu contoh dari konsep polymorphism,
    di mana objek kelas Bird_2010 dapat merespon metode move_2010()
    dengan cara yang berbeda dari objek kelas Animal_2010.*/

    @Override
    public void move_2010() { //func move di inisialisasi berbeda dg yng ada di animal.
        if (canFly) {
            System.out.println(name + " terbang");
        } else {
            System.out.println(name + " berjalan");
        }
    }
}
```

```
@Override
public void eat_2010() {
    System.out.println(name + " sedang mencari makan");
}

public void chirp_2010() {
    System.out.println(name + " sedang berkicau");
}
}
```

```
package com.Bunga_Laelatul_Muna.PBO.Pertemuan5.Tugas;
// Bunga Laelatul Muna
```

```
// 21102010

public class Bird_2010 extends Animal_2010 {
    boolean canFly;

    public Bird_2010(String name, String habitat, boolean
canFly) {
        super(name, habitat);
        this.canFly = canFly;
    }

    /* === Implementasi Polymorphism ===
    Ini adalah salah satu contoh dari konsep
    polymorphism,
    di mana objek kelas Bird_2010 dapat merespon
    metode move_2010()
    dengan cara yang berbeda dari objek kelas
    Animal_2010.*/
    @Override
    public void move_2010() { //func move di inialisasi
berbeda dg yng ada di animal.
        if (canFly) {
            System.out.println(name + " terbang");
        } else {
            System.out.println(name + " berjalan");
        }
    }

    @Override
    public void eat_2010() {
        System.out.println(name + " sedang mencari
makan");
    }

    public void chirp_2010() {
        System.out.println(name + " sedang berkicau");
    }
}
```


- **Source Code (Main.Java)**

```
package com.Bunga_Laelatul_Muna.PB0.Pertemuan5.Tugas;
// 🌸 Bunga Laelatul Muna
// 21102010
import java.util.ArrayList;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        ArrayList<Animal_2010> animals = new ArrayList<>();
        Scanner input = new Scanner(System.in);

        while (true) {
            System.out.println("Pilih binatang yang ingin ditambahkan:");
            System.out.println("1. Burung");
            System.out.println("2. Ikan");
            System.out.println("3. Selesai");

            System.out.print("Menu: ");
            int choice = input.nextInt();

            if (choice == 1) {
                Bird_2010 bird = createBird();
                animals.add(bird);
            } else if (choice == 2) {
                Fish_2010 fish = createFish();
                animals.add(fish);
            } else {
                break;
            }
        }

        for (Animal_2010 animal : animals) {
            System.out.println("Nama: " + animal.name);
            System.out.println("Habitat: " + animal.habitat);
            animal.move_2010();
            animal.eat_2010();
            System.out.println();
        }
    }

    public static Bird_2010 createBird() {
        Scanner input = new Scanner(System.in);

        System.out.print("Masukan nama burung: ");
        String name = input.nextLine();

        System.out.print("Masukan habitat burung: ");
        String habitat = input.nextLine();
    }
}
```

```

public static Bird_2010 createBird() {
    Scanner input = new Scanner(System.in);

    System.out.print("Masukan nama burung: ");
    String name = input.nextLine();

    System.out.print("Masukan habitat burung: ");
    String habitat = input.nextLine();

    System.out.print("Bisa terbang? (ya/tidak): ");
    String canFlyStr = input.nextLine();
    boolean canFly;
    if (canFlyStr.equalsIgnoreCase("ya")) {
        canFly = true;
    } else if (canFlyStr.equalsIgnoreCase("tidak")) {
        canFly = false;
    } else {
        System.out.println("Input tidak valid. Mengasumsikan tidak bisa terbang.");
        canFly = false;
    }

    return new Bird_2010(name, habitat, canFly);
}

```

```

public static Fish_2010 createFish() {
    Scanner input = new Scanner(System.in);
    System.out.print("Masukan nama ikan: ");
    String name = input.nextLine();

    System.out.print("Masukan habitat ikan: ");
    String habitat = input.nextLine();

    System.out.print("Bisa bernafas di air? (ya/tidak): ");
    String Canbreat_Str = input.nextLine();
    boolean CanBreath;
    if (Canbreat_Str.equalsIgnoreCase("ya")) {
        CanBreath = true;
    } else if (Canbreat_Str.equalsIgnoreCase("tidak")) {
        CanBreath = false;
    } else {
        System.out.println("Input tidak valid. Mengasumsikan tidak bisa terbang.");
        CanBreath = false;
    }

    return new Fish_2010(name, habitat, CanBreath);
}
}

```

- **SS Output**

```
Pilih binatang yang ingin ditambahkan:
1. Burung
2. Ikan
3. Selesai
Menu: 1
Masukan nama burung: Pipi
Masukan habitat burung: Hutan
Bisa terbang? (ya/tidak): Ya
Pilih binatang yang ingin ditambahkan:
1. Burung
2. Ikan
3. Selesai
Menu: 2
Masukan nama ikan: Fishhi
Masukan habitat ikan: Air Tawar
Bisa bernafas di air? (ya/tidak): ya
Pilih binatang yang ingin ditambahkan:
1. Burung
2. Ikan
3. Selesai
Menu:
```

- **Penjelasan**

Ini program program Polymorphism dengan menu yang mengimplementasi jenis overriding. Dimana di kelas Animal dia memiliki method move_2010 yang menampilkan output ‘bergerak’ namun di kelas fish dan bird method tersebut memiliki output yang berbeda dengan yang ada di animal.

Lalu di main kelas – kelas tsb di panggil dengan menu yang menggunakan perulangan for. Di program main juga ada menggunakan Boolean untuk menjawab pertanyaan dr method move.

- **Alur Program**

Ketika Program di run, yang akan muncul pertama ada pilihan menu untuk option 1 Bird dan Option 2 Fish. Ketika user memilih 1 maka akan langsung ke program di kelas Bird. Di kelas Bird akan muncul pertanyaan yang ada di kelas animal yaitu nama dan habitat kemudian disusul pertanyaan dari kelas burung yaitu method Move_2010(terbang/tidak). Jika user menjawab iya maka akan muncul “Bisa terbang” dan sebaliknya jika tidak “Tidak

Terbang”. Iya/Tidak ini adalah bentuk Boolean yang di konversi ke string menggunakan if else.

Ketika user memilih angka 2 yang muncul adalah pertanyaan dr kelas animal dan juga kelas fish yaitu methode move(Bisa bernafas dalam air atau tidak)

A. REFERENSI

1. PetaniKode
2. Java Documentation
3. Modul 5