LAPORAN PRAKTIKUM PEMROGRAMAN

BERBASIS OBJEK

MODUL 6 POLYMORPHISM

Oleh :

Alif Alpian Sahrul Muharom

(20102007)

Dosen:

Agus Priyanto, M.Kom

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

PURWOKERTO

2022

I. TUJUAN

a. Mengerti prinsip polimorfisme dalam bahasa C++ dan Java
b. Mengerti tentang prinsip polimorfisme dan pemakaiannya dalam membentuk suatu kelas.

II. TOOL

1. Apache NetBeans IDE 13
2. Java SE Development Kit 18

III. DASAR TEORI

Polimorfisme terbagi menjadi dua suku kata yaitu, *Poly* yang berarti banyak dan *Morfisme* yang berarti bentuk. Dalam ilmu sains, Polimorfisme (*polymorphism*) adalah sebuah prinsip dalam biologi di mana organisme atau spesies memiliki banyak bentuk serta tahapan (*stages*). Prinsip tersebut diterapkan juga pada bahasa Java.

Polimorfisme dalam OOP merupakan sebuah konsep OOP di mana *class* memiliki banyak "bentuk" *method* yang berbeda, meskipun namanya sama. Maksud dari "bentuk" adalah isinya yang berbeda, namun tipe data dan parameternya berbeda.

Polimorfisme juga dapat diartikan sebagai teknik *programming* yang mengarahkan kamu untuk memprogram secara general daripada secara spesifik. Contohnya kita memiliki tiga class yang berbeda yaitu: "Kelinci", "Kucing", dan "Sapi". Di mana ketiga *class* tersebut merupakan turunan dari *class* "Hewan".

Polimorfisme pada Java memiliki 2 macam yaitu diantaranya:

1. Static Polymorphism (Polimorfisme statis).
2. Dynamic Polymorphism (Polimorfisme dinamis).

Perbedaan keduanya terletak pada cara membuat polimorfisme. Polimorfisme statis menggunakan *method overloading,* sedangkan polimorfisme dinamis menggunakan *method overriding*.

IV. GUIDED

1. BINATANG

Binatang.java

```
package main.java.com.Alpian.Pertemuan_7.Guided.Binatang;
/**
 *
 * @author
 * Alif Alpian Sahrul Muharom
 * 20102007
 * IF-08-0
 */
public abstract class Binatang {
    private String jenis;

    public Binatang(String jenis) {
        this.jenis=jenis;
    }

    protected abstract void suara();

    public String toString(){
        return "seekor" + jenis;
    }
}
```

Penjelasan:

Kelas Binatang merupakan induk dari kelas Anjing, Burung, Kambing, dan Kucing. Di dalamkelas induk terdapat atribut yang akan digunakan pada semua kelas, yaitu jenis dengan tipe data String. Terdapat juga method abstract bernama suara() pada kelas induk.

Anjing.java

```
package main.java.com.Alpian.Pertemuan_7.Guided.Binatang;
/**
 *
 * @author
 * Alif Alpian Sahrul Muharom
 * 20102007
 * IF-08-0
 */
public class Anjing extends Binatang{
    private String nama;

    public Anjing(String nama) {
        super("Anjing");
        this.nama = nama;
```

```
    }

    public void suara(){
        System.out.println("Mengonggong");
    }

    public String toString(){
        return super.toString()+ "" +nama;
    }
```

Burung.java

```
package main.java.com.Alpian.Pertemuan_7.Guided.Binatang;
/**
 *
 * @author
 * Alif Alpian Sahrul Muharom
 * 20102007
 * IF-08-0
 */
public class Burung extends Binatang{
    private String nama;

    public Burung(String nama) {
        super("Burung");
        this.nama = nama;
    }

    protected void suara() {
        System.out.println("Berkicau");
    }

    public String toString(){
        return super.toString() + "" +nama;
    }
}
```

Kambing.java

```
package main.java.com.Alpian.Pertemuan_7.Guided.Binatang;
/**
 *
 * @author
 * Alif Alpian Sahrul Muharom
 * 20102007
 * IF-08-0
 */
public class Kambing extends Binatang{
```

```
    private String nama;

    public Kambing(String nama) {
        super("Kambing");
        this.nama = nama;
    }

    public void suara(){
        System.out.println("Mengembek");
    }

    public String toString(){
        return super.toString()+ "" +nama;
    }
}
```

Kucing.java

```
package main.java.com.Alpian.Pertemuan_7.Guided.Binatang;
/**
 *
 * @author
 * Alif Alpian Sahrul Muharom
 * 20102007
 * IF-08-0
 */
public class Kucing extends Binatang{
    private String nama;

    public Kucing(String nama) {
        super("Kucing");
        this.nama = nama;
    }

    public void suara(){
        System.out.println("Mengeong");
    }

    public String toString(){
        return super.toString()+ "" +nama;
    }

}
```

Penjelasan:

Kelas- kelas di atas merupakan anak dari kelas Binatang. Atribut String jenis tidak perlu ditulis kembali pada kelas-kelas tersebut. Kelas Anjing dan kelas anak lainnya memanggil method abstract dari kelas induk, yaitu method suara(). Namun, di kelas anak tidak sebagai abstract dan memiliki implementasi pada fungsi/method suara().

DemoPaket.java

```
package main.java.com.Alpian.Pertemuan_7.Guided.Binatang;

import java.util.Random;
/**
 *
 * @author
 * Alif Alpian Sahrul Muharom
 * 20102007
 * IF-08-0
 */
public class CobaPolimorfisme {
    public static void main(String[] args) {
        Binatang[] peliharaanku = {
            new Burung("Hantu"),
            new Kucing("Persia"),
            new Anjing("Buldog"),
            new Kambing("Etawa")
        };

        Binatang kesayangan;
        Random pilihan = new Random();
        kesayangan =
peliharaanku[pilihan.nextInt(peliharaanku.length)];
        System.out.println("Binatang Kesayangan Ku : " + kesayangan);

        System.out.print("Suarannya : ");
        kesayangan.suara();
    }
}
```

Penjelasan:

Pada kelas Main (CobaPoli) terdapat library Random yang mana nantinya dapat mengacak output yang akan dikeluarkan. Kelas-kelas anak dipanggil di dalam kelas main sebagai objek dan atribut dari kelas induk (String jenis) digunakan dan diisi dengan String

2. PEGAWAI

Pegawai.java

```
package main.java.com.Alpian.Pertemuan_7.Guided.Pegawai;
/**
 *
 * @author
 * Alif Alpian Sahrul Muharom
 * 20102007
 * IF-08-0
*/
```

```
public class Direktur extends Pegawai {
    private double gajiDirektur;
    private double dividenSaham;
    // Konstruktor kelas Direktur
    public Direktur(String nama, double gajiDirektur, double
dividenSaham) {
        super(nama); // Memanggil konstruktor kelas Pegawai
        setGajiDirektur(gajiDirektur);
        setDividen(dividenSaham);
    }
    // Mengset gaji direktur
    public void setGajiDirektur(double gaji) {
        if (gaji > 0) {
            gajiDirektur = gaji;
        } else {
            gajiDirektur = 0;
        }
    }
    // Mengset hasil pembagian dividen keuntungan saham
    public void setDividen(double dividen) {
        if (dividen > 0) {
            dividenSaham = dividen;
        } else {
            dividenSaham = 0;
        }
    }

    // Method yang mengembalikan nama
    public String nama() {
        return super.getNamaPeg();
    }

    // Method yang mengembalikan jabatan
    public String jabatan() {
        return "Direktur";
    }

    // Method yang mengembalikan besar gaji direktur
    public double gajiPerBulan() {
        return gajiDirektur;
    }

    // Method yang mengembalikan besar dividen saham
    public double labDividen() {
        return dividenSaham;
    }

    // Pengimplementasian/ Pendefinisian method abstract dari kelas
```

```
 Pegawai
     // Method ini mengembalikan besar gaji direktur
     @Override
     public double income() {
         return gajiDirektur + dividenSaham;
     }

}
```

Penjelasan:

Terdapat atribut namaPegawai dengan tipe data String pada kelas induk Pegawai. Terdapat juga constructor, method getter, dan method abstract income() dengan tipe data income() yangtidak memiliki implementasi.

Direktur.java

```
package main.java.com.Alpian.Pertemuan_7.Guided.Pegawai;
/**
 *
 * @author
 * Alif Alpian Sahrul Muharom
 * 20102007
 * IF-08-0
*/
public abstract class Pegawai {
    private String namaPeg;

    // konstruktor
    public Pegawai(String namaPeg) {
        this.namaPeg = namaPeg;
    }

    // method (get) untuk mengembalikan nama pegawai
    public String getNamaPeg() {
        return namaPeg;
    }

    // Method abstract ini diwariskan ke semua kelas yang
    // diturunkan dari kelas abstrak ini
    public abstract double income();
    // fungsi abstract, hanya deklarasi, tanpa implementasi
}
```

Penjelasan:

Kelas Direktur merupakan anak dari superclass Pegawai (ditandai dengan penggunaan extendsdan pemanggilan atribut menggunakan super()). Metode set ada dua di dalam kelas ini, yaitu setGajiDirektur dan setDividen yang parameternya menggunakan atribut kelas ini, yaitugajiDirektur dan dividen dengan tipe data double.

Test.java

```
package main.java.com.Alpian.Pertemuan_7.Guided.Pegawai;
/**
 *
 * @author
 * Alif Alpian Sahrul Muharom
 * 20102007
 * IF-08-0
*/
public class Test {
    public static void main(String[] args) {
        Direktur d = new Direktur("Wahyu", 12_000_000.00, 7_500_000.00);
        DecimalFormat digitPresisi = new DecimalFormat("0.00");
        /*Objek referensi dari kelas abstrak pegawai (pgw) merefer objek
        dari kelas Direktur (d) yang diturunkan dari kelas abstrak
Pegawai
        */
        System.out.println("\nDEMO      INHERITANCE,      ENCAPSULATION,
POLYMORPHISM");
        System.out.println("-----------------------------------------
--\n");

        // Mencetak informasi Direktur ke console
        System.out.println("Nama    : " + d.nama() + "\n"
                + "Jabatan  : " + d.jabatan() + "\n"
                + "Gaji  :  "  +  digitPresisi.format(d.gajiPerBulan())  +
"\n"
                + "Dividen   : " + digitPresisi.format(d.labDividen()) +
"\n"
                +  "Total      :  "  +  digitPresisi.format(d.income())  +
"\n");
        System.exit(0);
    }
}
```

Penjelasan:

Pada kelas Main terdapat library yang digunakan untuk membulatkan ke atas output bilangandesimal, yaitu import java.text.DecimalFormat. Objek diisi dengan format String, double, double (tanda (_) pada setiap angka digunakan sebagai pemisah saja agar mempermudah membaca code). Objek kemudian dipanggil pada output dengan objek dari library DecimalFormat agar output bisa dibulatkan.

3. EKSPRESI WAJAH

EkspresiWajah.java

```
package main.java.com.Alpian.Pertemuan_7.Guided.Ekspresi_Wajah;
/**
 *
 * @author
 * Alif Alpian Sahrul Muharom
 * 20102007
 * IF-08-0
 */
public class EkspresiWajah {
    public String respons() {
        return ("Lihat Wajahku ini");
    }
}
```

Penjelasan:

Ekspresi Wajah merupakan kelas induk dari kelas anak pada kelas ini, yaitu Gembira dan Sedih.

Gembira.java

```
package main.java.com.Alpian.Pertemuan_7.Guided.Ekspresi_Wajah;
/**
 *
 * @author
 * Alif Alpian Sahrul Muharom
 * 20102007
 * IF-08-0
 */
public class Gembira extends EkspresiWajah {
    @Override
    public String respons() {
        return ("Ha..ha.. saya lagi senang =)");
    }
}
```

Penjelasan:

Walaupun kelas anak, namun kelas Gembira dan Sedih dapat menggunakan @Override untukmengeset output nilai kembalian (return) yang berbeda dari kelas induknya.

Sedih.java

```
package main.java.com.Alpian.Pertemuan_7.Guided.Ekspresi_Wajah;
/**
 *
 * @author
 * Alif Alpian Sahrul Muharom
 * 20102007
 * IF-08-0
```

```
 */
 public class Sedih extends EkspresiWajah {
     @Override
     public String respons() {
         return ("Hiks..hiks.. =(");
     }
}
```

Ekspresi.java

```
 package main.java.com.Alpian.Pertemuan_7.Guided.Ekspresi_Wajah;
 /**
  *
  * @author
  * Alif Alpian Sahrul Muharom
  * 20102007
  * IF-08-0
 */
 public class Ekspresi {
     public static void main(String[] args) {
         System.out.println("DEMO POLIMORFISME");
         System.out.println("========================");

         EkspresiWajah objEkspresi = new EkspresiWajah();
         Gembira objGembira = new Gembira();
         Sedih objSedih = new Sedih();

         EkspresiWajah[] ekspresi = new EkspresiWajah[3];
         ekspresi[0] = objEkspresi;
         ekspresi[1] = objGembira;
         ekspresi[2] = objSedih;

         System.out.println("Ekspresi[0]:" + ekspresi[0].respons());
         System.out.println("Ekspresi[1]:" + ekspresi[1].respons());
         System.out.println("Ekspresi[2]:" + ekspresi[2].respons());
     }
}
```

Penjelasan:

Seriap kelas memiliki objeknya masing-masing, namun hal tersebut dideklarasikan ulangmenggunakan ekspresi[] untuk mempermudah pemanggilan objek pada output.

V. UNGUIDED

Employee.java

```
package main.java.com.Alpian.Pertemuan_7.Unguided;
/**
 *
 * @author
 * Alif Alpian Sahrul Muharom
 * 20102007
 * IF-08-0
 */
public abstract class Employee {
    protected String nama, nip;
    protected long gajiPokok, komisi, gaji;

    public Employee(String nama, String nip, long gajiPokok, long komisi)
{
        this.nama = nama;
        this.nip = nip;
        this.gajiPokok = gajiPokok;
        this.komisi = komisi;
    }

    public String nama() {
        return nama;
    }

    public String nip() {
        return nip;
    }

    public long gajiPokok() {
        return gajiPokok;
    }

    public long komisi() {
        return komisi;
    }

    public abstract long gaji();
}
```

Penjelasan:

Employee merupakan kelas abstract induk dari SalariedEmployee, CommissionEmployee, dan ProjectPlanner. Di dalam kelas ini terdapat atribut yang digunakan pada kelas anak, sehingga tidak perlu menginputkan atribut yang sama untuk setiap kelas anaknya. Method-method seperti nama(), nip(), gajiPokok(), dan komisi() digunakan untuk mengembalikan nilai.

SalariedEmployee.java

```
package main.java.com.Alpian.Pertemuan_7.Unguided;
/**
 *
 * @author
 * Alif Alpian Sahrul Muharom
 * 20102007
 * IF-08-0
 */
public class SalariedEmployee extends Employee {
    private long upahMingguan;

    public SalariedEmployee(long upahMingguan, String nama, String nip,
long gajiPokok, long komisi) {
        super(nama, nip, gajiPokok, komisi);
        this.upahMingguan = upahMingguan;
    }

    public long getUpahMingguan() {
        return upahMingguan;
    }

    @Override
    public long gaji(){
        return gaji = upahMingguan;
    }

    public void cetakSE() {
        System.out.println("Nama: " + nama);
        System.out.println("NIP: " + nip);
        System.out.println("Upah Mingguan: " + gaji());
    }
}
```

Penjelasan:

SalariedEmployee merupakan salah satu anak dari kelas induk Employee. getUpahMingguan() digunakan untuk mengembalikan nilai upahMingguan yang dideklarasikan lagi ke atribut pajak dikarenakan pada kelas ini, gaji = upahMingguan.

CommissionEmployee.java

```
package main.java.com.Alpian.Pertemuan_7.Unguided;
/**
 *
 * @author
 * Alif Alpian Sahrul Muharom
 * 20102007
 * IF-08-0
*/
public class CommissionEmployee extends Employee{
    private long totalPenjualan;

    public CommissionEmployee(long totalPenjualan, String nama, String
nip, long gajiPokok, long komisi) {
        super(nama, nip, gajiPokok, komisi);
        this.totalPenjualan = totalPenjualan;
    }

    public long getTotalPenjualan() {
        return totalPenjualan;
    }

    @Override
    public long gaji() {
        return gaji = (gajiPokok + (komisi * totalPenjualan));
    }

    public void cetakCE(){
        System.out.println("Nama: " + nama);
        System.out.println("NIP: " + nip);
        System.out.println("Gaji Pokok: " + gajiPokok);
        System.out.println("Total Penjualan: " + totalPenjualan);
        System.out.println("Komisi: " + komisi);
        System.out.println("Gaji: " + gaji());
    }
}
```

Penjelasan:

Seperti kelas sebelumnya, kelas ini juga merupakan kelas anak. Perhitungan gaji pada kelas ini melibatkan gajiPokok, komisi, dan totalPenjualan. Pertama, komisi dikalikan dengan jumlah dari total penjualan yang telah diinput pada kelas Main. Lalu, hasilnya ditambahkan dengan gaji pokok yang telah diinputkan juga.

ProjectPlanner.java

```
package main.java.com.Alpian.Pertemuan_7.Unguided;
/**
 *
 * @author
 * Alif Alpian Sahrul Muharom
 * 20102007
 * IF-08-0
*/
```

Penjelasan:

Mirip dengan kelas sebelumnya, namun totalPenjualan diganti menggunakan totalHasilProyekdan dikurangi dengan method pajak(). Hasil dari pajak didapatkan dari 5%-nya gajiPokok yangmana nilainya telah ditetapkan pada kelas Main. Setelah itu, gaji dapat dihitung.

Main.java

```
package main.java.com.Alpian.Pertemuan_7.Unguided;
/**
 *
 * @author
 * Alif Alpian Sahrul Muharom
 * 20102007
 * IF-08-0
*/
public class ProjectPlanner extends Employee{
    private long totalHasilProyek, pajak;

    public ProjectPlanner(long totalHasilProyek, String nama, String nip,
long gajiPokok, long komisi) {
        super(nama, nip, gajiPokok, komisi);
        this.totalHasilProyek = totalHasilProyek;
    }

    public long getTotalHasilProyek() {
        return totalHasilProyek;
    }

    public long pajak() {
        return pajak = gajiPokok * 5/100;
    }

    @Override
    public long gaji() {
        return gaji = gajiPokok + (komisi * totalHasilProyek) - pajak();
    }

    public void cetakPP() {
```

```
        System.out.println("Nama: " + nama);
        System.out.println("NIP: " + nip);
        System.out.println("Gaji Pokok: " + gajiPokok);
        System.out.println("Komisi: " + komisi);
        System.out.println("Total Hasil Proyek: " + totalHasilProyek);
        System.out.println("Gaji: " + gaji());
    }
}
```

Penjelasan:

Pada kelas Main terdapat objek-objek referensi dari kelas abstrak Employee. Setelah diinputnilai-nilai yang dibutuhkan, dipanggil method cetak dari ketiga kelas.

VI. KESIMPULAN

Polimorfisme dapat berarti “mempunyai banyak bentuk” sehingga dapat disimpulkan, Polimorfisme adalah kemampuan untuk meminta objek yang berbeda untuk melaksanakan tugas yang sama dan membuat objek tahu bagaimana untuk mencapainya dengan caranya sendiri. Polimorfisme menunjukkan kemampuan untuk menangani dua atau lebih bentuk obyek yang berlainan saat eksekusi berlangsung.

Di dalam Java, class yang memiliki method tanpa definisi atau method kosong dapat kita jadikan kelas abstrak. Pendefinisian kelas abstrak biasanya digunakan untuk polimorfisme di mana subclass dari Class Abstrak yang mendefinisikan method kosong dari Class Abstrak. Class abstrak tidak bisa diinstantiate (di-create menjadi objek), tetapi bisa me-refer objek kongkrit yang Classnya diturunkan dari dirinya.

Abstract Method adalah method yang didefinisikan dengan keyword abstract. Di dalam sebuah kelas abstrak, kita dapat membuat *abstract method*. Abstract method adalah method yangtidak ada implementasinya. Implementasi dari semua abstract method yang dimiliki kelas abstrakdilakukan di kelas konkrit yang di turunkan dari kelas abstrak tersebut. Method Abstrak ini hanyadapat di miliki oleh kelas abstrak. Sebuah kelas konkrit tidak dapat memiliki method abstrak.