LAPORAN HASIL PRAKTIKUM 07

PEMROMGRAMAN BERBASIS OBJEK

ATHAULLA HAFIZH

244107020030

TI 2A

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

2025

3. Praktikum

Percobaan 1

Karyawan.java

```
public class Karyawan {
    private String nama;
    private String nip;
    private String golongan;
    private double gaji;

    public void setNama(String nama) {
        this.nama = nama;
    }

    public String getNama() {
        return nama;
    }

    public void setNip(String nip) {
        this.nip = nip;
    }

    public String getNip() {
        return nip;
    }

    public void setGolongan(String golongan) {
        this.golongan = golongan;

        // Penentuan gaji berdasarkan golongan
        switch (golongan.charAt(0)) {
            case '1':
                this.gaji = 5000000;
                break;
            case '2':
                this.gaji = 3000000;
```

```
                break;
            case '3':
                this.gaji = 2000000;
                break;
            case '4':
                this.gaji = 1000000;
                break;
            case '5':
                this.gaji = 750000;
                break;
        }
    }

    public String getGolongan() {
        return golongan;
    }

    public void setGaji(double gaji) {
        this.gaji = gaji;
    }

    public double getGaji() {
        return gaji;
    }
}
```

Staff.java

```
public class Staff extends Karyawan {
    private int lembur;
    private double gajiLembur;

    public void setLembur(int lembur) {
        this.lembur = lembur;
    }
```

```
    public int getLembur() {
        return lembur;
    }

    public void setGajiLembur(double gajiLembur) {
        this.gajiLembur = gajiLembur;
    }

    public double getGajiLembur() {
        return gajiLembur;
    }

    // Overloading method getGaji()
    public double getGaji(int lembur, double gajiLembur) {
        return super.getGaji() + (lembur * gajiLembur);
    }

    // Overriding method getGaji()
    @Override
    public double getGaji() {
        return super.getGaji() + (lembur * gajiLembur);
    }

    public void lihatInfo() {
        System.out.println("NIP             : " + this.getNip());
        System.out.println("Nama            : " + this.getNama());
        System.out.println("Golongan        : " +
this.getGolongan());
        System.out.println("Jumlah Lembur   : " + this.getLembur());
        System.out.printf("Gaji Lembur     : %.0f\n",
this.getGajiLembur());
        System.out.printf("Gaji Total      : %.0f\n",
this.getGaji());
    }
}
```

Manager.java

```
public class Manager extends Karyawan {
    private double tunjangan;
    private String bagian;
    private Staff st[];

    public void setTunjangan(double tunjangan) {
        this.tunjangan = tunjangan;
    }

    public double getTunjangan() {
        return tunjangan;
    }

    public void setBagian(String bagian) {
        this.bagian = bagian;
    }

    public String getBagian() {
        return bagian;
    }

    public void setStaff(Staff st[]) {
        this.st = st;
    }

    public void viewStaff() {
        System.out.println("-----------------");
        for (int i = 0; i < st.length; i++) {
            st[i].lihatInfo();
            System.out.println(""); // Memberi spasi antar staff
        }
        System.out.println("-----------------");
    }

    public void lihatInfo() {
```

```
        System.out.println("Manager          : " + this.getBagian());
        System.out.println("NIP              : " + this.getNip());
        System.out.println("Nama             : " + this.getNama());
        System.out.println("Golongan         : " +
this.getGolongan());
        System.out.printf("Tunjangan       : %.0f\n",
this.getTunjangan());
        System.out.printf("Gaji             : %.0f\n",
this.getGaji());
        System.out.println("Bagian           : " + this.getBagian());
        System.out.println("Daftar Staff     : ");
        this.viewStaff();
    }

    // Overriding method getGaji()
    @Override
    public double getGaji() {
        return super.getGaji() + tunjangan;
    }
}
```

Utama.java

```
public class Utama {
    public static void main(String[] args) {
        System.out.println("Program Testing Class Manager & Staff");
        System.out.println("=====================================\n"
);

        // Pembuatan manajer
        Manager man[] = new Manager[2];
        man[0] = new Manager();
        man[0].setNama("Tedjo");
        man[0].setNip("101");
        man[0].setGolongan("1");
        man[0].setTunjangan(5000000);
        man[0].setBagian("Administrasi");
```

```
        man[1] = new Manager();
        man[1].setNama("Atika");
        man[1].setNip("102");
        man[1].setGolongan("1");
        man[1].setTunjangan(2500000);
        man[1].setBagian("Pemasaran");

        // Pembuatan staff untuk manager 1
        Staff staff1[] = new Staff[2];
        staff1[0] = new Staff();
        staff1[0].setNama("Usman");
        staff1[0].setNip("0003");
        staff1[0].setGolongan("2");
        staff1[0].setLembur(10);
        staff1[0].setGajiLembur(10000);

        staff1[1] = new Staff();
        staff1[1].setNama("Anugrah");
        staff1[1].setNip("0005");
        staff1[1].setGolongan("2");
        staff1[1].setLembur(10);
        staff1[1].setGajiLembur(55000);
        man[0].setStaff(staff1);

        // Pembuatan staff untuk manager 2
        Staff staff2[] = new Staff[3];
        staff2[0] = new Staff();
        staff2[0].setNama("Hendra");
        staff2[0].setNip("0004");
        staff2[0].setGolongan("3");
        staff2[0].setLembur(15);
        staff2[0].setGajiLembur(5500);

        staff2[1] = new Staff();
        staff2[1].setNama("Arie");
        staff2[1].setNip("0006");
```

```
        staff2[1].setGolongan("4");
        staff2[1].setLembur(5);
        staff2[1].setGajiLembur(100000);

        staff2[2] = new Staff();
        staff2[2].setNama("Mentari");
        staff2[2].setNip("0007");
        staff2[2].setGolongan("3");
        staff2[2].setLembur(6);
        staff2[2].setGajiLembur(20000);
        man[1].setStaff(staff2);

        // Cetak informasi dari manager + staffnya
        man[0].lihatInfo();
        System.out.println("\n");
        man[1].lihatInfo();
    }
}
```

Output

```
Program Testing Class Manager & Staff
=====================================

Manager         : Administrasi
NIP             : 101
Nama            : Tedjo
Golongan        : 1
Tunjangan       : 5000000
Gaji            : 10000000
Bagian          : Administrasi
Daftar Staff    :
-----------------
NIP             : 0003
Nama            : Usman
Golongan        : 2
Jumlah Lembur   : 10
Gaji Lembur     : 10000
Gaji Total      : 3100000

NIP             : 0005
Nama            : Anugrah
Golongan        : 2
Jumlah Lembur   : 10
Gaji Lembur     : 55000
Gaji Total      : 3550000

-----------------
```

```
Manager         : Pemasaran
NIP             : 102
Nama            : Atika
Golongan        : 1
Tunjangan       : 2500000
Gaji            : 7500000
Bagian          : Pemasaran
Daftar Staff    :
----------------
NIP             : 0004
Nama            : Hendra
Golongan        : 3
Jumlah Lembur   : 15
Gaji Lembur     : 5500
Gaji Total      : 2082500

NIP             : 0006
Nama            : Arie
Golongan        : 4
Jumlah Lembur   : 5
Gaji Lembur     : 100000
Gaji Total      : 1500000

NIP             : 0007
Nama            : Mentari
Golongan        : 3
Jumlah Lembur   : 6
Gaji Lembur     : 20000
Gaji Total      : 2120000

----------------
```

Latihan

```java
public class PerkalianKu {

 void perkalian(int a, int b){

  System.out.println(a * b);

 }

 void perkalian(int a, int b, int c){

  System.out.println(a * b * c);

 }

 public static void main(String args []){

  PerkalianKu objek = new PerkalianKu();

  objek.perkalian(25, 43);
  objek.perkalian(34, 23, 56);
 }
}
```

4.1 Dari source coding di atas terletak di manakah overloading?

Overloading terjadi pada **method perkalian**.

```
void perkalian(int a, int b){ ... }
void perkalian(int a, int b, int c){
```

Terdapat dua method dengan nama perkalian yang sama dalam satu kelas. Namun, daftar parameternya berbeda; yang satu memiliki dua parameter, dan yang lainnya memiliki tiga parameter. Inilah yang disebut **overloading**.

4.2 Jika terdapat overloading ada berapa jumlah parameter yang berbeda?

Terdapat **2** jumlah parameter yang berbeda.

```
void perkalian(int a, int b) // 2 parameter
void perkalian(int a, int b, int c) // 3 parameter
```

Method pertama memiliki 2 parameter (int a, int b), sedangkan method kedua memiliki 3 parameter (int a, int b, int c).

```
public class PerkalianKu {

 void perkalian(int a, int b){

  System.out.println(a * b);

 }

 void perkalian(double a, double b){

  System.out.println(a * b);

 }

 public static void main(String args []){

  PerkalianKu objek = new PerkalianKu();

  objek.perkalian(25, 43);
  objek.perkalian(34.56, 23.7);
 }
}
```

4.3 Dari source coding di atas terletak di manakah overloading?

Overloading terjadi pada **method perkalian**.

```
void perkalian(int a, int b){ ... }
void perkalian(double a, double b){ ... }
```

Kedua method memiliki nama yang sama (perkalian) tetapi tipe data parameternya berbeda. Method pertama menerima dua int, sedangkan yang kedua menerima dua double.

4.4 Jika terdapat overloading ada berapa tipe data parameter yang berbeda?

Terdapat **2** tipe data parameter yang berbeda.

```
void perkalian(int a, int b) // Tipe data integer
void perkalian(double a, double b) // Tipe data double
```

Method pertama menggunakan tipe data int untuk parameternya, dan method kedua menggunakan tipe data double.

```
  class Ikan{
   public void swim(){
      System.out.println("Ikan bisa berenang");
   }
}
class Piranha extends Ikan{
   public void swim(){
      System.out.println("Piranha bisa makan daging");
   }
}
public class Fish {
    public static void main(String[] args) {
        Ikan a = new Ikan();
      Ikan b = new Piranha();
      a.swim();
      b.swim();
    }
   }
```

4.5 Dari source coding di atas terletak di manakah overriding?

Overriding terletak pada **method swim() di dalam kelas Piranha**.

```
// Method milik Parent Class (Superclass)
public void swim() {
    System.out.println("Ikan bisa berenang");
}

// Method milik Child Class (Subclass) yang meng-override
public void swim(){
    System.out.println("Piranha bisa makan daging");
}
```

Kelas Piranha sebagai *subclass* menuliskan kembali method swim() yang diwarisi dari *superclass* Ikan. Deklarasi method (nama, return type, dan parameter) sama persis, tetapi isinya berbeda.

4.6 Jabarkanlah apabila sourcoding di atas jika terdapat overriding?

Kode tersebut menunjukkan bagaimana *subclass* memodifikasi perilaku yang diwarisi dari *superclass*.

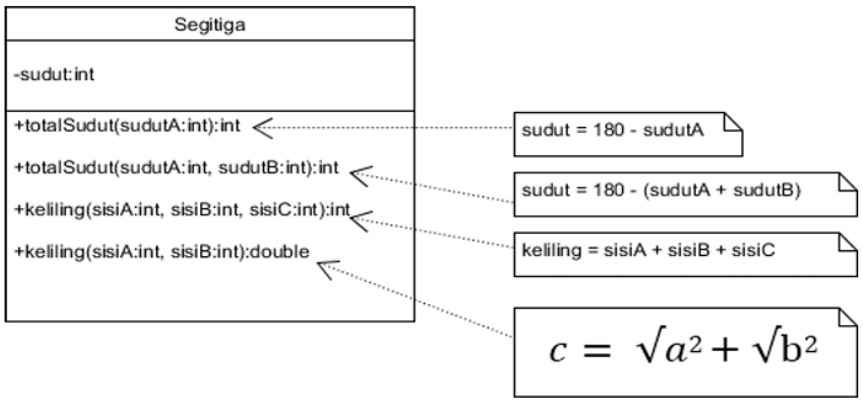
```
Ikan b = new Piranha(); // Objek Piranha, tetapi referensi bertipe Ikan
b.swim(); // Output: "Piranha bisa makan daging"
```

Meskipun variabel b memiliki tipe Ikan, objek yang sebenarnya dibuat adalah new Piranha(). Ketika b.swim() dipanggil, Java secara dinamis menentukan bahwa objek tersebut adalah Piranha, sehingga method swim() yang telah di-*override* di kelas Piranha yang akan dieksekusi, bukan method asli dari kelas Ikan. Ini menunjukkan bahwa *subclass* memiliki tingkah laku yang lebih spesifik.

5. Tugas

5.1 Overloading

Implementasikan konsep overloading pada class diagram dibawah ini :



Segitiga.java

```
public class Segitiga {

    private int sudut;


    // Menghitung besar sudut ketiga jika satu sudut diketahui
(asumsi segitiga sama kaki)

    public int totalSudut(int sudutA) {

        sudut = 180 - (sudutA * 2);

        return sudut;

    }
```

```
    // Menghitung besar sudut ketiga jika dua sudut diketahui
    public int totalSudut(int sudutA, int sudutB) {
        sudut = 180 - (sudutA + sudutB);
        return sudut;
    }

    // Menghitung keliling jika ketiga sisi diketahui
    public int keliling(int sisiA, int sisiB, int sisiC) {
        return sisiA + sisiB + sisiC;
    }

    // Menghitung keliling segitiga siku-siku jika dua sisi (alas
dan tinggi) diketahui
    public double keliling(int sisiA, int sisiB) {
        // Menggunakan Pythagoras untuk mencari sisi miring
(hipotenusa)
        double sisiC = Math.sqrt(Math.pow(sisiA, 2) +
Math.pow(sisiB, 2));
        return sisiA + sisiB + sisiC;
    }
}
```

MainSegitiga.java

```
public class MainSegitiga {
    public static void main(String[] args) {
        Segitiga segitiga = new Segitiga();

        System.out.println("===== MENGGUNAKAN OVERLOADING METHOD
=====");

        // Memanggil method totalSudut dengan 1 parameter
        System.out.println("Total Sudut (jika 1 sudut diketahui): "
+ segitiga.totalSudut(60));

        // Memanggil method totalSudut dengan 2 parameter
        System.out.println("Total Sudut (jika 2 sudut diketahui): "
+ segitiga.totalSudut(60, 30));
```

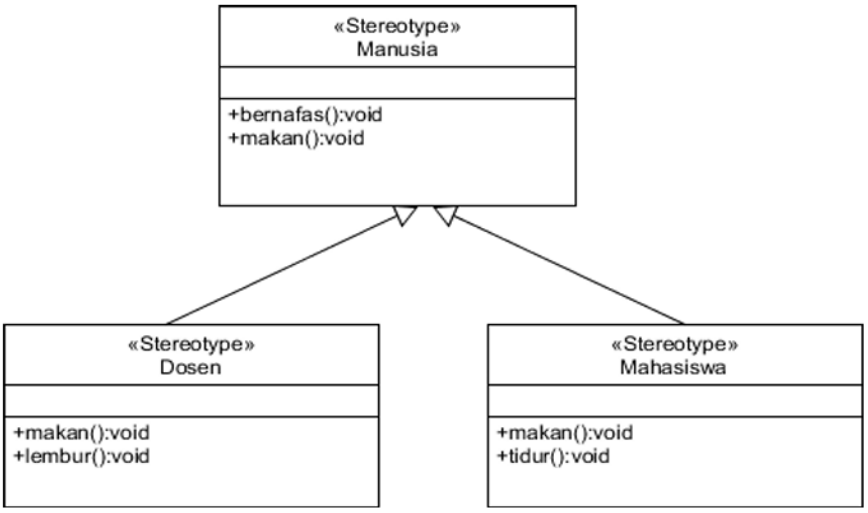
```
        System.out.println("----------------------------------------
");


        // Memanggil method keliling dengan 3 parameter

        System.out.println("Keliling (jika 3 sisi diketahui): " +
segitiga.keliling(10, 12, 14));


        // Memanggil method keliling dengan 2 parameter

        // Hasilnya adalah double

        System.out.printf("Keliling (jika 2 sisi siku-siku
diketahui): %.2f\n", segitiga.keliling(10, 12));

    }

}
```

Output

```
===== MENGGUNAKAN OVERLOADING METHOD =====
Total Sudut (jika 1 sudut diketahui): 60
Total Sudut (jika 2 sudut diketahui): 90
----------------------------------------
Keliling (jika 3 sisi diketahui): 36
Keliling (jika 2 sisi siku-siku diketahui): 37,62
```

5.2 Overriding

Implementasikan class diagram dibawah ini dengan menggunakan teknik dynamic method dispatch :



Manusia.java

```
public class Manusia {
    public void bernafas() {
        System.out.println("Manusia perlu bernafas untuk hidup.");
    }

    public void makan() {
        System.out.println("Manusia juga perlu makan.");
    }
}
```

Dosen.java

```
public class Dosen extends Manusia {
    // Overriding method makan() dari superclass Manusia
    @Override
    public void makan() {
        System.out.println("Dosen makan agar kuat memberikan
nilai.");
    }

    public void lembur() {
        System.out.println("Dosen sering lembur untuk menyelesaikan
pekerjaan.");
    }
}
```

Mahasiswa.java

```
public class Mahasiswa extends Manusia {
    // Overriding method makan() dari superclass Manusia
    @Override
    public void makan() {
        System.out.println("Mahasiswa makan mie instan di akhir
bulan.");
    }
```

```
    public void tidur() {
        System.out.println("Mahasiswa butuh tidur setelah
mengerjakan tugas.");
    }
}
```

MainManusia.java

```
public class MainManusia {
    public static void main(String[] args) {
        // Membuat objek referensi dari superclass
        Manusia manusia;

        System.out.println("===== DYNAMIC METHOD DISPATCH =====");

        // Objek Dosen direferensikan oleh Manusia
        manusia = new Dosen();
        manusia.bernafas(); // Memanggil method dari superclass
        manusia.makan();    // Memanggil method yang sudah di-
override oleh Dosen
        // manusia.lembur(); // Ini akan error karena method
lembur() tidak ada di class Manusia

        System.out.println("-----------------------------------");

        // Objek Mahasiswa direferensikan oleh Manusia
        manusia = new Mahasiswa();
        manusia.bernafas(); // Memanggil method dari superclass
        manusia.makan();    // Memanggil method yang sudah di-
override oleh Mahasiswa
        // manusia.tidur(); // Ini akan error karena method tidur()
tidak ada di class Manusia
    }
}
```

Output

```
===== DYNAMIC METHOD DISPATCH =====
Manusia perlu bernafas untuk hidup.
Dosen makan agar kuat memberikan nilai.
---------------------------------
Manusia perlu bernafas untuk hidup.
Mahasiswa makan mie instan di akhir bulan.
```