



Name : Faatihurizki Prasojo

NIM : 244107020142

Class : TI_1H

Laporan Tugas Minggu 9

Pemrograman Berorientasi Objek (PBO)

Percobaan 1

Class Karyawan

```
1 public class Karyawan {  
2  
3     /**  
4      * @param args the command line arguments  
5      */  
6     // public static void main(String[] args) {  
7     // TODO code application logic here  
8     // }  
9     private String nama;  
10    private String nip;  
11    private String golongan;  
12    private double gaji;  
13  
14    public void setNama(String nama)  
15    {  
16        this.nama=nama;  
17    }  
18  
19    public void setNip(String nip)  
20    {  
21        this.nip=nip;  
22    }  
23  
24    public void setGolongan(String golongan)  
25    {  
26        this.golongan=golongan;  
27  
28        switch(golongan.charAt(0)){  
29            case '1':this.gaji=5000000;  
30            break;  
31            case '2':this.gaji=3000000;  
32            break;  
33            case '3':this.gaji=2000000;  
34            break;  
35            case '4':this.gaji=1000000;  
36            break;  
37            case '5':this.gaji=750000;  
38            break;  
39        }  
40    }
```



Name : Faatihurrizki Prasojo

NIM : 244107020142

Class : TI_1H

Class Staff

```
● ● ●

1  public class Staff extends Karyawan {
2      private int lembur;
3      private double gajiLembur;
4
5      public void setLembur(int lembur)
6      {
7          this.lembur=lembur;
8      }
9
10     public int getLembur()
11     {
12         return lembur;
13     }
14
15     public void setGajiLembur(double gajiLembur)
16     {
17         this.gajiLembur=gajiLembur;
18     }
19
20     public double getGajiLembur()
21     {
22         return gajiLembur;
23     }
24
25     public double getGaji(int lembur, double gajiLembur)
26     {
27         return super.getGaji()+lembur*gajiLembur;
28     }
29
30     public double getGaji()
31     {
32         return super.getGaji()+lembur*gajiLembur;
33     }
34
35     public void lihatInfo()
36     {
37         System.out.println("NIP :" +this.getNip());
38         System.out.println("Nama :" +this.getNama());
39         System.out.println("Golongan :" +this.getGolongan());
40         System.out.println("Jml Lembur :" +this.getLembur());
41         System.out.printf("Gaji Lembur :%.0f\n", this.getGajiLembur());
42         System.out.printf("Gaji :%.0f\n",this.getGaji());
43     }
44 }
```



Name : Faatihurizki Prasojo

NIM : 244107020142

Class : TI_1H

Class Manager

```
● ● ●

1  public class Manager extends Karyawan {
2      private double tunjangan;
3      private String bagian;
4      private Staff st[];
5
6      public void setTunjangan(double tunjangan)
7      {
8          this.tunjangan=tunjangan;
9      }
10
11     public double getTunjangan()
12     {
13         return tunjangan;
14     }
15
16     public void setBagian(String bagian)
17     {
18         this.bagian=bagian;
19     }
20
21     public String getBagian()
22     {
23         return bagian;
24     }
25
26     public void setStaff(Staff st[])
27     {
28         this.st=st;
29     }
30
31     public void viewStaff()
32     {
33         int i;
34         System.out.println("-----");
35         for(i=0;i<st.length;i++)
36         {
37             st[i].lihatInfo();
38         }
39         System.out.println("-----");
40     }
41
42     public void lihatInfo()
43     {
44         System.out.println("Manager :" +this.getBagian());
45         System.out.println("NIP :" +this.getNip());
46         System.out.println("Nama :" +this.getNama());
47         System.out.println("Golongan :" +this.getGolongan());
48         System.out.printf("Tunjangan :%.0f\n",this.getTunjangan());
49         System.out.printf("Gaji :%.0f\n",this.getGaji());
50         System.out.println("Bagian :" +this.getBagian());
51         this.viewStaff();
52     }
53
54     public double getGaji()
55     {
56         return super.getGaji()+tunjangan;
57     }
58 }
```



Name : Faatihurrizki Prasojo

NIM : 244107020142

Class : TI_1H

Class Utama

```
● ● ●
1  public class Utama {
2
3      public static void main(String[] args)
4      {
5          System.out.println("Program Testing Class Manager & Staff");
6
7          Manager man[] = new Manager[2];
8          Staff staff1[] = new Staff[2];
9          Staff staff2[] = new Staff[3];
10
11         man[0] = new Manager();
12         man[0].setNama("Tedjo");
13         man[0].setNip("101");
14         man[0].setGolongan("1");
15         man[0].setTunjangan(5000000);
16         man[0].setBagian("Administrasi");
17
18         man[1] = new Manager();
19         man[1].setNama("Atika");
20         man[1].setNip("102");
21         man[1].setGolongan("1");
22         man[1].setTunjangan(2500000);
23         man[1].setBagian("Pemasaran");
24
25         staff1[0] = new Staff();
26         staff1[0].setNama("Osman");
27         staff1[0].setNip("0003");
28         staff1[0].setGolongan("2");
29         staff1[0].setLembur(10);
30         staff1[0].setGajiLembur(10000);
31
32         staff1[1] = new Staff();
33         staff1[1].setNama("Anugrah");
34         staff1[1].setNip("0005");
35         staff1[1].setGolongan("2");
36         staff1[1].setLembur(10);
37         staff1[1].setGajiLembur(55000);
38
39         man[0].setStaff(staff1);
40
41         staff2[0] = new Staff();
42         staff2[0].setNama("Hendra");
43         staff2[0].setNip("0004");
44         staff2[0].setGolongan("3");
45         staff2[0].setLembur(15);
46         staff2[0].setGajiLembur(5500);
47
48         staff2[1] = new Staff();
49         staff2[1].setNama("Arie");
50         staff2[1].setNip("0006");
51         staff2[1].setGolongan("4");
52         staff2[1].setLembur(5);
53         staff2[1].setGajiLembur(100000);
54
55         staff2[2] = new Staff();
56         staff2[2].setNama("Mentari");
57         staff2[2].setNip("0007");
58         staff2[2].setGolongan("3");
59         staff2[2].setLembur(6);
60         staff2[2].setGajiLembur(20000);
61
62         man[1].setStaff(staff2);
63
64         man[0].lihatInfo();
65         man[1].lihatInfo();
66     }
67 }
```



Name : Faatihurizki Prasojo

NIM : 244107020142

Class : TI_1H

Output

```
Program Testing Class Manager & Staff
Manager :Administrasi
NIP :101
Nama :Tedjo
Golongan :1
Tunjangan :5000000
Gaji :10000000
Bagian :Administrasi
-----
NIP :0003
Nama :Osman
Golongan :2
Jml Lembur :10
Gaji Lembur :10000
Gaji :3100000
NIP :0005
Nama :Anugrah
Golongan :2
Jml Lembur :10
Gaji Lembur :55000
Gaji :3550000
-----
Manager :Pemasaran
NIP :102
Nama :Atika
Golongan :1
Tunjangan :2500000
Gaji :7500000
Bagian :Pemasaran
-----
NIP :0004
Nama :Hendra
Golongan :3
Jml Lembur :15
Gaji Lembur :5500
Gaji :2082500
NIP :0006
Nama :Arie
Golongan :4
Jml Lembur :5
Gaji Lembur :1000000
```

X ⚡ main* ⚡ ⚡ 0 △ 0 ⓘ 1 ⚡ ⚡ Java: Ready ⚡ Discord RPC



Name : Faatihurizki Prasojo

NIM : 244107020142

Class : TI_1H

Latihan

Class ikan



```
1 public class PerkalianKu {  
2     void perkalian(int a, int b){  
3         System.out.println(a * b);  
4     }  
5  
6     void perkalian(int a, int b, int c){  
7         System.out.println(a * b * c);  
8     }  
9  
10    public static void main(String args []){  
11        PerkalianKu objek = new PerkalianKu();  
12        objek.perkalian(25, 43);  
13        objek.perkalian(34, 23, 56);  
14    }  
15 }
```

Output

```
1075  
43792  
PS C:\User
```

4.1 Dari source coding diatas dimanakah overloading?

= Overloading terjadi pada **dua method** yang bernama **perkalian**.

1. void perkalian(int a, int b): Method yang menerima **dua** parameter (int).
2. void perkalian(int a, int b, int c): Method yang menerima **tiga** parameter (int).

4.2 Jika terdapat overloading ada berapa parameter yang berbeda?

= Terdapat **2 (dua)** jenis daftar parameter yang berbeda, yaitu:

1. Daftar parameter dengan **2 parameter** (int, int).
2. Daftar parameter dengan **3 parameter** (int, int, int).



Name : Faatihurrizki Prasojo

NIM : 244107020142

Class : TI_1H

Class Perkalianku

Output

```
● ● ●
1 public class PerkalianKu {
2
3     void perkalian(int a, int b) {
4         System.out.println(a * b);
5     }
6
7     void perkalian(double a, double b) {
8         System.out.println(a * b);
9     }
10
11    public static void main(String args[]) {
12        PerkalianKu objek = new PerkalianKu();
13        objek.perkalian(25, 43);
14        objek.perkalian(34.56, 23.7);
15    }
16 }
```

```
1075
819.072
PS C:\User:
```

4.3 Dari source coding diatas dimanakah overloading?

1. void perkalian(**int a, int b**): Method yang menerima dua parameter bertipe **integer (int)**.
2. void perkalian(**double a, double b**): Method yang menerima dua parameter bertipe **floating-point (double)**.

4.4 Jika terdapat overloading ada berapa tipe parameter yang berbeda?

= Terdapat **2** jenis daftar tipe parameter yang berbeda:

1. Daftar tipe parameter (**int, int**).
2. Daftar tipe parameter (**double, double**).



Name : Faatihurizki Prasojo

NIM : 244107020142

Class : TI_1H

Class Ikan, Piranha,, Fish

```
● ● ●  
1 class Ikan {  
2     public void swim() {  
3         System.out.println("Ikan bisa berenang");  
4     }  
5 }  
6  
7 class Piranha extends Ikan {  
8     public void swim() {  
9         System.out.println("Piranha bisa makan daging");  
10    }  
11 }  
12  
13 public class Fish {  
14     public static void main(String[] args) {  
15         Ikan a = new Ikan();  
16         Ikan b = new Piranha();  
17  
18         a.swim();  
19         b.swim();  
20     }  
21 }
```

Output

```
Ikan bisa berenang  
Piranha bisa makan daging  
PS C:\Users\fdlpr\Polinema\Se
```

4.5 Dari source coding diatas dimanakah overriding?

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

- Kelas induk (**Ikan**) memiliki method: `public void swim()`.
- Kelas anak (**Piranha**) mendefinisikan ulang method yang sama persis: `public void swim()`.



Name : Faatihurizki Prasojo

NIM : 244107020142

Class : TI_1H

4.6 Jabarkanlah apabila source coding diatas jika terdapat overriding?

= Terdapat *Method Overriding* pada public void swim() karena:

1. **Hubungan Pewarisan:** Kelas Piranha adalah *subclass* dari kelas Ikan (Piranha extends Ikan).
2. **Definisi Ulang:** Kelas Piranha **mendefinisikan ulang** (*override*) *method* swim() yang diwarisi dari Ikan.
 - o *Method* swim() di Ikan mencetak: "Ikan bisa berenang".
 - o *Method* swim() di Piranha mencetak: "Piranha bisa makan daging".

Bagaimana Overriding Bekerja (Dilihat dari main):

Ketika program dijalankan:

- **a.swim():** Variabel a bertipe Ikan dan mengacu pada objek Ikan. Maka, *method* swim() dari kelas **Ikan** dipanggil, menghasilkan *output*: Ikan bisa berenang.
- **b.swim():** Variabel b bertipe Ikan, tetapi mengacu pada objek **Piranha**. Karena *method* swim() telah di-*override* di Piranha, maka *method* swim() dari kelas **Piranha** yang akan dipanggil, menghasilkan *output*: Piranha bisa makan daging.