

**LAPORAN PRAKTIKUM**  
**PEMROGRAMAN BERORIENTASI OBJEK**  
**KE - 13 (Polymorphism)**



Nama : Azzahra Rachmadia Mumtaz

NRP : 3124600095

Kelas: D4 IT D

Dosen Pengajar : Grezio Arifiyan Primajaya S.Kom, M.Kom

**PROGRAM STUDI D4 TEKNIK INFORMATIKA POLITEKNIK**  
**ELEKTRONIKA NEGERI SURABAYA (PENS) TAHUN 2025**

## A. Tujuan Pembelajaran

1. Memahami dan menerapkan konsep polimorfisme dalam pemrograman
2. Memahami proses terjadinya Virtual Method Invocation
3. Memahami dan menerapkan polymorphic arguments dalam pemrograman
4. Memahami penggunaan instanceof dan cara melakukan casting object

## B. Tugas Pendahuluan

1. Apakah yang dimaksud dengan polimorfisme ?
2. Jelaskan proses terjadinya Virtual Method Invocation !
3. Apakah yang dimaksud dengan polymorphic arguments ?
4. Apakah kegunaan kata kunci instanceof ?

### Jawaban

1. Polimorfisme artinya "banyak bentuk" adalah kemampuan sebuah objek (dari subclass) untuk diperlakukan seolah-olah ia adalah objek dari superclass-nya.
2. Virtual Method Invocation (VMI) adalah proses saat program berjalan (runtime), di mana Java secara otomatis memilih versi method override yang benar untuk dijalankan. Pilihan ini berdasarkan tipe objek asli, bukan tipe referensi variabelnya.
3. Polymorphic Arguments adalah parameter pada sebuah method yang dideklarasikan dengan tipe superclass, tetapi saat dipanggil dapat menerima argumen berupa objek dari subclass mana pun.
4. Operator instanceof digunakan untuk memeriksa tipe data sebuah objek pada saat runtime. Ini mengembalikan true jika objek tersebut adalah instans dari kelas yang ditentukan (atau turunan dari kelas itu), dan false jika bukan.

## C. Percobaan

### 1. Percobaan 1 :

#### a. Listing Program

```
class Parent {
    int x = 5;
    public void info(){
        System.out.println("Ini class Parent");
    }
}

class Child extends Parent {
    int x = 10;
    public void info(){
        System.out.println("Ini class Child");
    }
}

public class Tes {
    public static void main(String[] args){
        Parent tes = new Child();
        System.out.println("Nilai x = " + tes.x);
        tes.info();
    }
}
```

#### b. Output

## D. Latihan

### 1. Latihan 1 :

#### a. Listing Program

```
class Child extends Parent {
}

public class Tes {
    public static void main(String[] args){
        Parent tes = new Child();
        System.out.println("Nilai x = " + tes.x);
        tes.info();
    }
}
```

## 1. Base Class

```
PS C:\Praktikum PBO\Praktikum\Bab 13\Latihan 2> cat .\Base.java
class Base {
    int i = 99;

    Base(){
        amethod();
    }

    public void amethod(){
        System.out.println("Base amethod()");
    }
}
```

## 2. Derived Class

```
PS C:\Praktikum PBO\Praktikum\Bab 13\Latihan 2> cat .\Derived.java
public class Derived extends Base {
    int i = -1;

    public static void main(String[] args){
        Base b = new Derived();
        System.out.println(b.i);
        b.amethod();
    }

    public void amethod(){
        System.out.println("Derived.amethod()");
    }
}
```

### b. Output

```
PS C:\Praktikum PBO\Praktikum\Bab 13\Latihan 2> java .\Derived.java
Derived.amethod()
99
Derived.amethod()
```

## 3. Latihan 3 :

### a. Listing Program

#### 1. Parent

```
class Parent {
    protected void method1(){
        System.out.println("Parent's method1 ()");
    }

    public void method2(){
        System.out.println("Parent's method2 ()");
        method1();
    }
}
```

#### 2. Child

```
PS C:\Praktikum PBO\Praktikum\Bab 13\Latihan 3> cat *.java
class Child extends Parent{
    public void method1(){
        System.out.println("Child's method1 ()");
    }

    public static void main(String[] args){
        Parent p = new Child();
        p.method2();
    }
}
```

## b. Output

```
PS C:\Praktikum PB0\Praktikum\Bab 13\Latihan 3> java .\Child.java
Parent's method2 ()
Parent's method1 ()
```

## 4. Latihan 4 :

### Listing Program dan Output



The screenshot shows an IDE with a file named `Perusahaan.java`. The code defines a `Perusahaan` class with a `main` method that creates three `Pegawai` objects: `Manajer` (Budi), `Kurir` (Rudi), and `Pegawai` (Andi). It then calls the `bekerja` method on each. The `Pegawai` class has a `nama` attribute and a `bekerja` method that prints the employee's name and role. The `Manajer` and `Kurir` classes extend `Pegawai` and override the `bekerja` method to include specific roles.

```
1- public class Perusahaan {
2-     public static void main(String[] args) {
3-         Pegawai p1 = new Manajer("Budi");
4-         Pegawai p2 = new Kurir("Rudi");
5-         Pegawai p3 = new Pegawai("Andi");
6-
7-         p1.bekerja();
8-         p2.bekerja();
9-         p3.bekerja();
10-    }
11- }
12-
13-
14- class Pegawai {
15-     protected String nama;
16-
17-     public Pegawai(String nama) {
18-         this.nama = nama;
19-     }
20-
21-     public void bekerja() {
22-         System.out.print(nama + " bekerja di
23-         kantor");
24-     }
25- }
26-
27- class Manajer extends Pegawai {
28-     public Manajer(String nama) {
29-         super(nama);
30-     }
31-
32-     @Override
33-     public void bekerja() {
34-         super.bekerja();
35-         System.out.print(" sebagai manajer dan
36-         sedang memimpin rapat.");
37-         System.out.println();
38-     }
39- }
40-
41- class Kurir extends Pegawai {
42-     public Kurir(String nama) {
43-         super(nama);
44-     }
45-
46-     @Override
47-     public void bekerja() {
48-         super.bekerja();
49-         System.out.print(" sebagai kurir dan
50-         sedang mengirimkan paket.");
51-         System.out.println();
52-     }
53- }
```

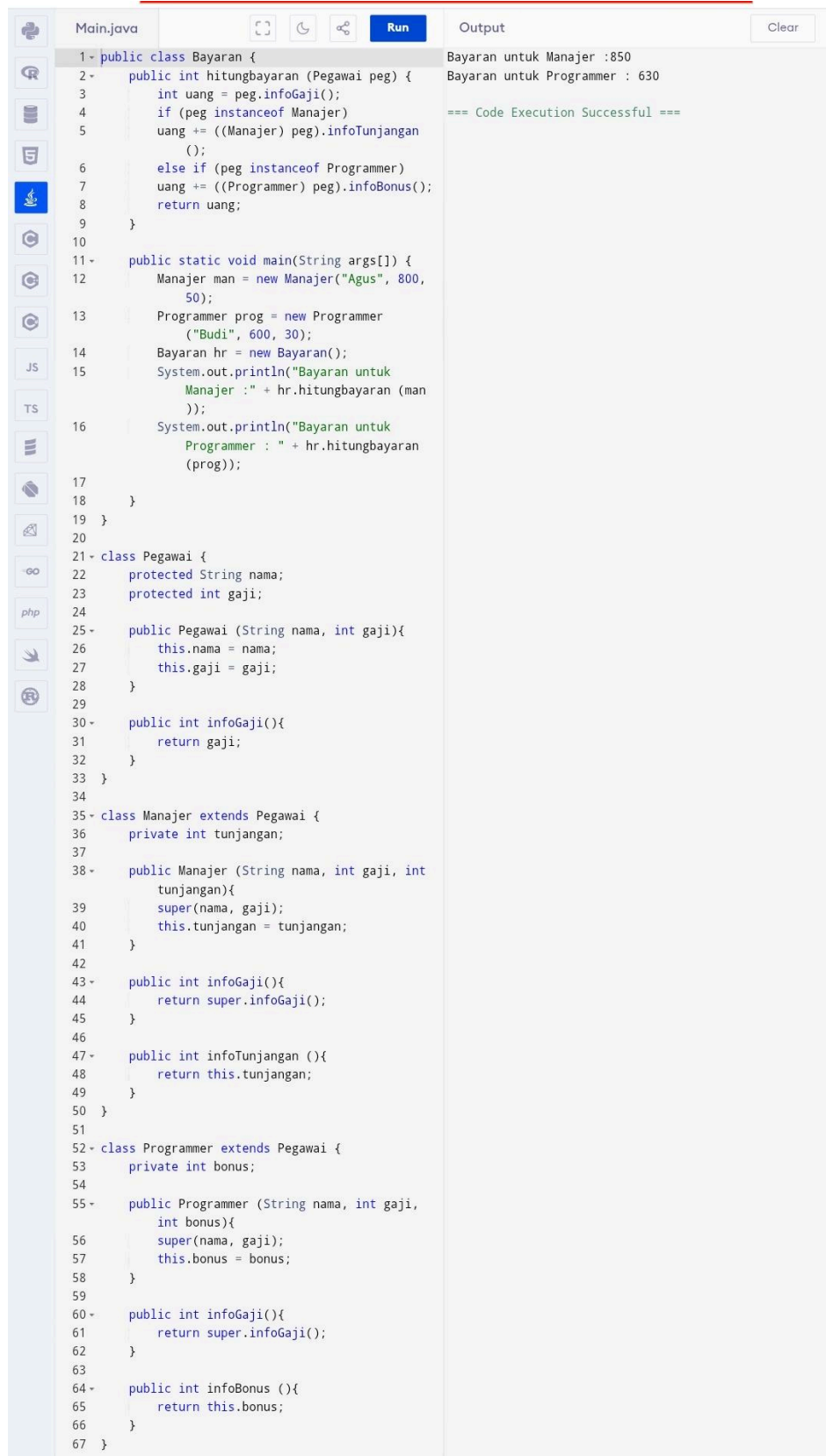
The output window shows the following text:

```
Budi bekerja di kantor sebagai manajer dan sedang
memimpin rapat.
Rudi bekerja di kantor sebagai kurir dan sedang
mengirimkan paket.
Andi bekerja di kantor
=== Code Execution Successful ===
```

## E. Tugas

### 1. Tugas 1 :

#### Listing Program dan output



The screenshot shows an IDE with a file named 'Main.java' and an 'Output' window. The code defines a hierarchy of classes: 'Pegawai' (base class), 'Manajer' (extends Pegawai), and 'Programmer' (extends Pegawai). A 'Bayaran' class calculates the total payment for a 'Pegawai' object, adding salary and bonus/tuition. The main method creates a 'Manajer' and a 'Programmer' object, then uses the 'Bayaran' class to calculate their payments.

```
1 public class Bayaran {
2     public int hitungbayaran (Pegawai peg) {
3         int uang = peg.infoGaji();
4         if (peg instanceof Manajer)
5             uang += ((Manajer) peg).infoTunjangan
6             ();
7         else if (peg instanceof Programmer)
8             uang += ((Programmer) peg).infoBonus();
9         return uang;
10    }
11
12    public static void main(String args[]) {
13        Manajer man = new Manajer("Agus", 800,
14        50);
15        Programmer prog = new Programmer
16        ("Budi", 600, 30);
17        Bayaran hr = new Bayaran();
18        System.out.println("Bayaran untuk
19        Manajer : " + hr.hitungbayaran (man
20        ));
21        System.out.println("Bayaran untuk
22        Programmer : " + hr.hitungbayaran
23        (prog));
24    }
25 }
26
27 class Pegawai {
28     protected String nama;
29     protected int gaji;
30
31     public Pegawai (String nama, int gaji){
32         this.nama = nama;
33         this.gaji = gaji;
34     }
35
36     public int infoGaji(){
37         return gaji;
38     }
39 }
40
41 class Manajer extends Pegawai {
42     private int tunjangan;
43
44     public Manajer (String nama, int gaji, int
45     tunjangan){
46         super(nama, gaji);
47         this.tunjangan = tunjangan;
48     }
49
50     public int infoGaji(){
51         return super.infoGaji();
52     }
53 }
54
55 class Programmer extends Pegawai {
56     private int bonus;
57
58     public Programmer (String nama, int gaji,
59     int bonus){
60         super(nama, gaji);
61         this.bonus = bonus;
62     }
63
64     public int infoGaji(){
65         return super.infoGaji();
66     }
67
68     public int infoBonus (){
69         return this.bonus;
70     }
71 }
```

Output:

```
Bayaran untuk Manajer :850
Bayaran untuk Programmer : 630

=== Code Execution Successful ===
```