

## **LAPORAN PRAKTIKUM**

### **STUDI KASUS 1 - 3**

Dibuat untuk memenuhi salah satu tugas mata kuliah Pemrograman Berbasis  
Objek yang diampu oleh Bapak Ardhian Ekawijana



#### **Disusun Oleh:**

NIM : 241511094  
NAMA : ZAHRA ALDILA  
KELAS : 2C  
PROGRAM STUDI : D3 – TEKNIK INFORMATIKA  
JURUSAN : TEKNIK KOMPUTER DAN INFORMATIKA

**Program Studi D-3 Teknik Informatika**

**Jurusan Teknik Komputer dan Informatika Politeknik Negeri Bandung**

**2025**

## DAFTAR ISI

DAFTAR ISI .....	i
DAFTAR GAMBAR .....	ii
A. Pendahuluan.....	1
1. Studi kasus 1 .....	1
2. Studi kasus 2 .....	3
1. Studi kasus 3 .....	5
B. Lesson Learned .....	9

## DAFTAR GAMBAR

Gambar 1. Ouput Studi Kasus 1 .....	3
Gambar 2. Output Studi Kasus 2 .....	5
Gambar 3. Output Studi Kasus 3 .....	7
Gambar 4. Output Studi Kasus 3 Input User .....	8

## A. Pendahuluan

### 1. Studi kasus 1

Studi kasus ini bertujuan untuk memperluas hierarki kelas pegawai dalam sistem penggajian menggunakan konsep inheritance (pewarisan) dan polimorfisme. Sebelumnya sudah ada beberapa jenis karyawan seperti Executive, Employee, Hourly, dan Volunteer. Pada tugas ini ditambahkan satu tipe karyawan baru, yaitu Commission, yaitu karyawan yang dibayar berdasarkan jam kerja seperti Hourly, tetapi juga mendapat komisi dari total penjualan.

Kelas Commission merupakan subclass dari Hourly, sehingga mewarisi semua atribut dan metode yang dimiliki Hourly, namun menambahkan dua atribut baru: totalSales (total penjualan) dan commissionRate (persentase komisi). Metode pay() di-override untuk menambahkan perhitungan komisi ke gaji per jam. Dalam kelas Staff, dua objek Commission ditambahkan ke daftar karyawan (staffList).

Hal ini memperlihatkan polimorfisme, karena meskipun Commission adalah subclass baru, objeknya tetap bisa diproses bersama jenis pegawai lain di dalam array StaffMember[] dan dipanggil dengan metode yang sama, seperti pay().

Menunjukkan pewarisan (extends Hourly) dan atribut baru.

```
public class Commission extends Hourly {  
    private double totalSales;    // total penjualan  
    private double commissionRate; // persentase komisi (mis. 0.2 = 20%)
```

Memanggil konstruktor parent dengan super() dan menambahkan atribut baru.

```
    public Commission(String name, String address, String phone,  
                      String socSecNumber, double rate, double commissionRate)  
    {  
        super(name, address, phone, socSecNumber, rate); // dari Hourly  
        this.commissionRate = commissionRate;  
        this.totalSales = 0;  
    }
```

Menghitung total gaji (upah per jam + komisi penjualan).

```
@Override
public double pay() {
    double payment = super.pay() + (commissionRate * totalSales);
    totalSales = 0; // reset setelah dibayar
    return payment;
}
```

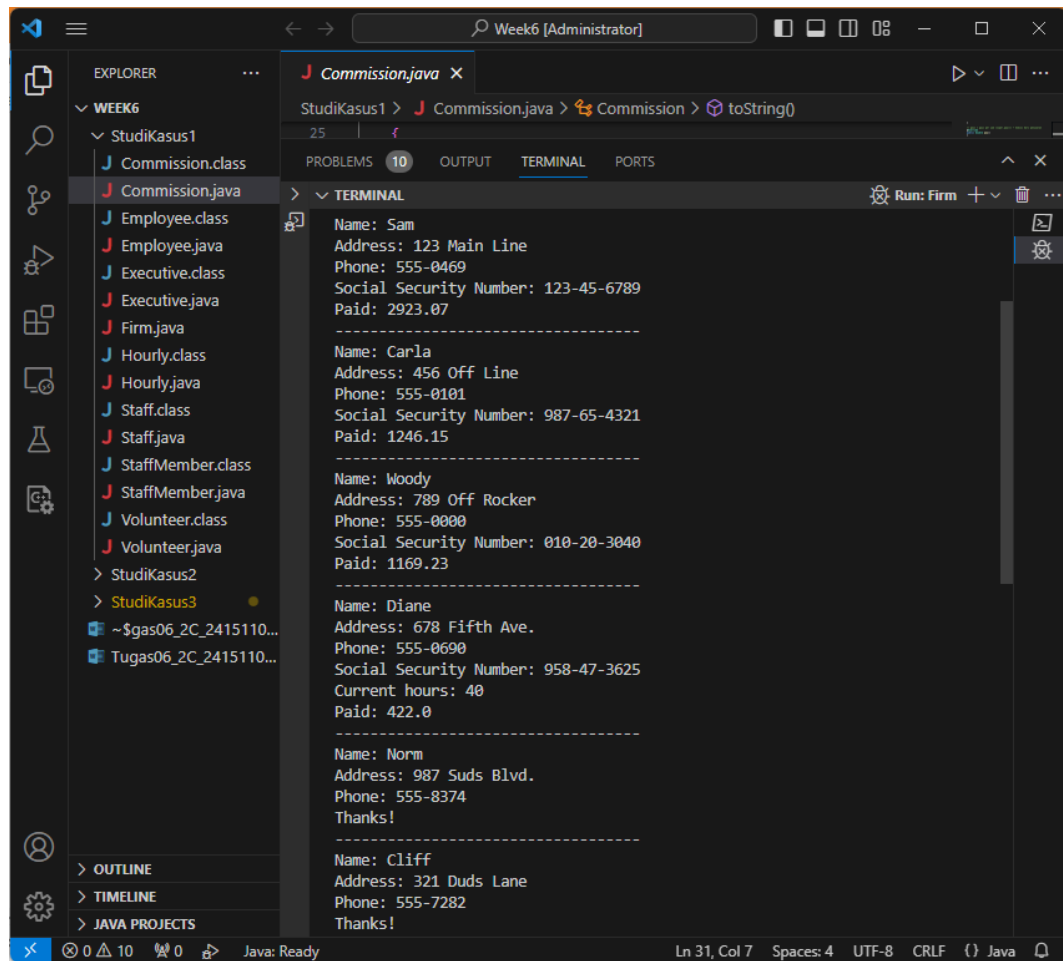
Objek Commission ditambahkan ke array StaffMember[] yang berisi berbagai tipe pegawai.

```
staffList[6] = new Commission("Alex", "101 Sales St", "555-9999",
    "222-11-1111", 6.25, 0.20);
staffList[7] = new Commission("Taylor", "202 Market Rd", "555-8888",
    "333-22-2222", 9.75, 0.15);

((Commission)staffList[6]).addHours(35);
((Commission)staffList[6]).addSales(400);
((Commission)staffList[7]).addHours(40);
((Commission)staffList[7]).addSales(950);
```

Pemanggilan polimorfik di payday().

```
amount = staffList[count].pay(); // memanggil pay() sesuai tipe objek
```



Gambar 1. Ouput Studi Kasus 1

## 2. Studi kasus 2

Program diminta membentuk hierarki kelas Shape sebagai kelas abstrak yang diturunkan oleh Sphere, Rectangle, dan Cylinder.

Setiap kelas turunan memiliki cara tersendiri untuk menghitung luas permukaannya melalui metode area().

Kelas Paint digunakan untuk menghitung jumlah cat yang dibutuhkan berdasarkan luas tiap bentuk dan tingkat cakupan cat (coverage).

Program utama PaintThings kemudian membuat beberapa objek bentuk dan menampilkan berapa banyak cat yang diperlukan untuk mengecat masing-masing objek.

Menunjukkan konsep abstraksi dan pewarisan.

```
public abstract class Shape {
    protected String shapeName;

    public Shape(String shapeName) {
        this.shapeName = shapeName;
    }

    // metode abstrak yang harus diimplementasikan oleh turunan
    public abstract double area();

    @Override
    public String toString() {
        return shapeName;
    }
}
```

Menunjukkan konsep polimorfisme, karena Shape bisa berupa Sphere, Rectangle, atau Cylinder.

```
public double amount(Shape s) {
    System.out.println("Computing amount for " + s);
    return s.area() / coverage;
}
```

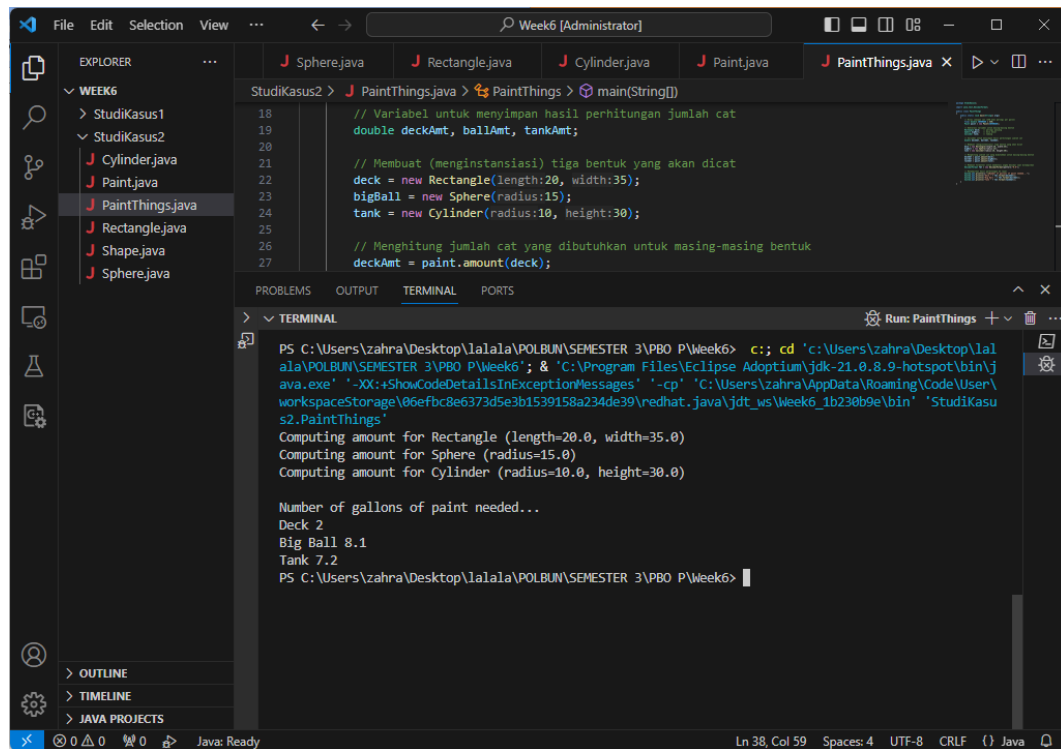
Menunjukkan konsep override dan penerapan rumus luas permukaan.

```
@Override
public double area() {
    // luas permukaan silinder =  $2 * \pi * r * (r + h)$ 
    return 2 * Math.PI * radius * (radius + height);
}
```

Menunjukkan penggunaan polimorfisme dalam praktik.

```
deckAmt = paint.amount(deck);
ballAmt = paint.amount(bigBall);
```

```
tankAmt = paint.amount(tank);
```



```
StudiKasus2 > PaintThings.java > PaintThings > main(String[])
18 // Variabel untuk menyimpan hasil perhitungan jumlah cat
19 double deckAmt, ballAmt, tankAmt;
20
21 // Membuat (menginstansiasi) tiga bentuk yang akan dicat
22 deck = new Rectangle(length:20, width:35);
23 bigBall = new Sphere(radius:15);
24 tank = new Cylinder(radius:10, height:30);
25
26 // Menghitung jumlah cat yang dibutuhkan untuk masing-masing bentuk
27 deckAmt = paint.amount(deck);

PROBLEMS OUTPUT TERMINAL PORTS
> TERMINAL
PS C:\Users\zahra\Desktop\lallala\POLBUN\SEMESTER 3\PBO P\Week6> c:\cd 'c:\Users\zahra\Desktop\lallala\POLBUN\SEMESTER 3\PBO P\Week6'; & 'C:\Program Files\Eclipse Adoptium\jdk-21.0.8.9-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\zahra\AppData\Roaming\Code\User\workspaceStorage\06efbc8e6373d5e3b1539158a234de39\redhat.java\jdt_ws\Week6_1b230b9e\bin' 'StudiKasus2.PaintThings'
Computing amount for Rectangle (length=20.0, width=35.0)
Computing amount for Sphere (radius=15.0)
Computing amount for Cylinder (radius=10.0, height=30.0)

Number of gallons of paint needed...
Deck 2
Big Ball 8.1
Tank 7.2
PS C:\Users\zahra\Desktop\lallala\POLBUN\SEMESTER 3\PBO P\Week6>
```

Gambar 2. Output Studi Kasus 2

## 1. Studi kasus 3

Studi kasus ini bertujuan untuk menunjukkan penerapan polimorfisme dan antarmuka Comparable dalam proses pengurutan objek menggunakan algoritma Selection Sort dan Insertion Sort.

Dalam program ini, kelas Sorting berisi dua metode pengurutan (selectionSort dan insertionSort) yang bekerja pada array berisi objek Comparable. Dengan cara ini, algoritma yang sama bisa dipakai untuk berbagai tipe data, seperti angka (Integer), string (String), maupun objek buatan (Salesperson).

Kelas Salesperson mengimplementasikan antarmuka Comparable dan mengatur metode compareTo() untuk menentukan urutan berdasarkan total penjualan (menurun) dan nama (descending secara alfabet) jika nilai penjualan sama. Hal ini memperlihatkan bagaimana objek yang berbeda dapat dibandingkan menggunakan satu antarmuka yang sama — inti dari polimorfisme.



Program WeeklySales menjadi driver yang membuat (atau membaca) data beberapa Salesperson, kemudian mengurutkannya dan menampilkan hasil ranking penjualan.

Menunjukkan bagaimana objek dibandingkan dengan logika sendiri.

```
// Urut berdasarkan totalSales menurun;
// jika sama, nama diurutkan descending juga.
public int compareTo(Object other) {
    Salesperson o = (Salesperson) other;
    if (this.totalSales != o.totalSales)
        return this.totalSales - o.totalSales;
    int byLast = o.lastName.compareTo(this.lastName);
    if (byLast != 0) return byLast;
    return o.firstName.compareTo(this.firstName);
}
```

Menunjukkan algoritma bisa bekerja untuk semua objek Comparable, bukan hanya angka.

```
// Versi descending
while (position > 0 && key.compareTo(list[position - 1]) > 0) {
    list[position] = list[position - 1];
    position--;
}
list[position] = key;
```

Menunjukkan perbedaan versi dinamis dari data hardcoded.

```
System.out.print("Masukkan jumlah salesperson: ");
int n = scan.nextInt();
scan.nextLine();

for (int i = 0; i < n; i++) {
    System.out.println("\nData ke-" + (i+1));
    System.out.print("Nama depan: ");
    String first = scan.nextLine();
    System.out.print("Nama belakang: ");
    String last = scan.nextLine();
    System.out.print("Total penjualan: ");
```

```

int sales = scan.nextInt();
scan.nextLine();

salesStaff[i] = new Salesperson(first, last, sales);
}

```

The screenshot shows the Eclipse IDE with the following components:

- EXPLORER:** A project named 'WEEK6' is expanded, showing sub-projects 'StudiKasus1', 'StudiKasus2', and 'StudiKasus3'. Under 'StudiKasus3', several Java files are listed, including 'WeeklySales.java' which is currently selected.
- EDITOR:** The 'WeeklySales.java' file is open, showing the following code:
 

```

package StudiKasus3;

public class WeeklySales
{
    public static void main(String[] args)
    {
        Salesperson[] salesStaff = new Salesperson[10];

        salesStaff[0] = new Salesperson(first:"Jane", last:"Jones", sa
        salesStaff[1] = new Salesperson(first:"Daffy", last:"Duck", sa
        salesStaff[2] = new Salesperson(first:"James", last:"Jones", sa
        salesStaff[3] = new Salesperson(first:"Dick", last:"Walter", sa
      
```
- TERMINAL:** The terminal shows the command to run the program and its output:
 

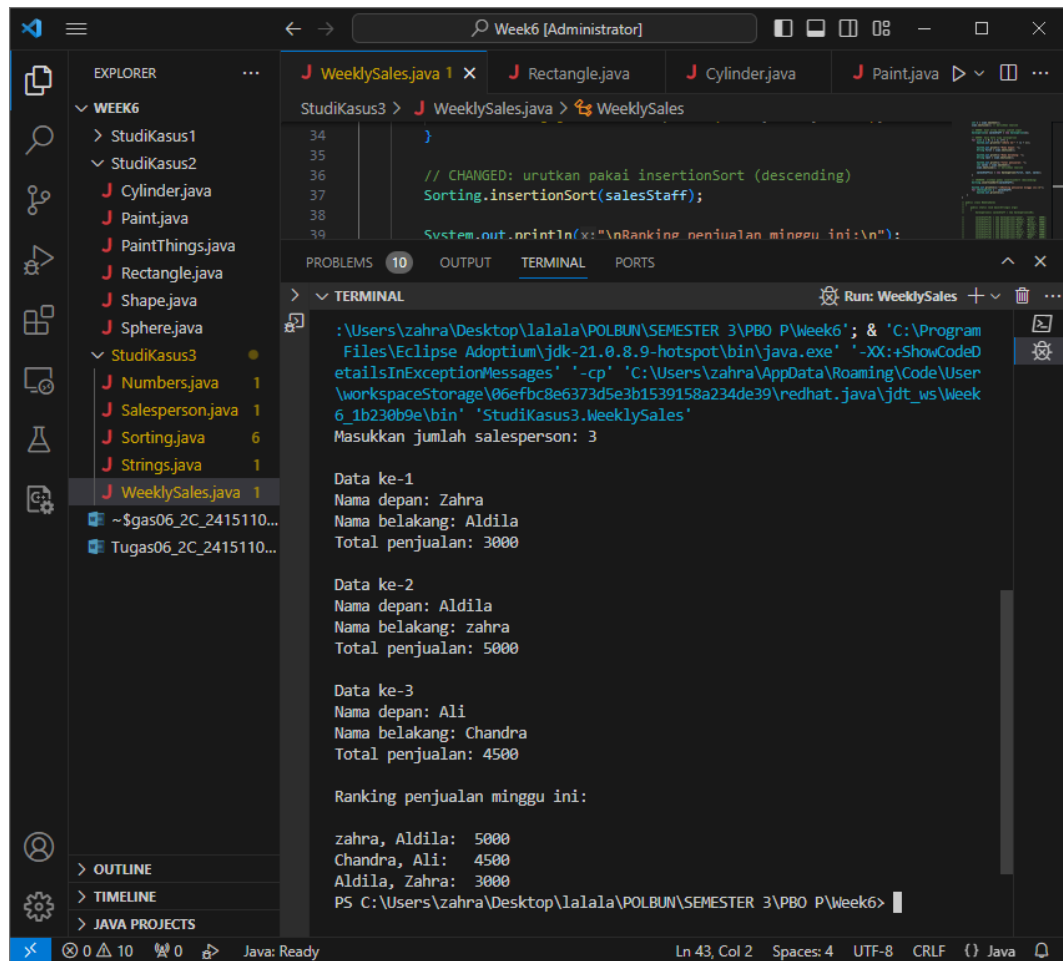
```

PS C:\Users\zahra\Desktop\lalala\POLBUN\SEMESTER 3\PBO P\Week6> & 'C:\Program Files\Eclipse Adoptium\jdk-21.0.8.9-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\zahra\AppData\Roaming\Code\User\workspaceStorage\06efbc8e6373d5e3b1539158a234de39\redhat.java\jdt_ws\Week6_1b230b9e\bin' 'StudiKasus3.WeeklySales'

Ranking of Sales for the Week

Taylor, Harry: 7300
Adams, Andy: 5000
Duck, Daffy: 4935
Black, Jane: 3000
Jones, James: 3000
Jones, Jane: 3000
Smith, Walt: 3000
Doe, Jim: 2850
Walter, Dick: 2800
Trump, Don: 1570
      
```

Gambar 3. Output Studi Kasus 3



Gambar 4. Output Studi Kasus 3 Input User

## **B. Lesson Learned**

Dari praktikum ini saya belajar beberapa hal:

1. Saya belajar bahwa konsep inheritance dan polimorfisme adalah dasar penting dalam pemrograman berorientasi objek, karena memungkinkan satu struktur program digunakan untuk berbagai jenis objek tanpa harus menulis ulang logika yang sama.
2. Dari Studi Kasus 1, saya memahami bagaimana kelas turunan dapat memperluas fungsionalitas kelas induk, seperti pada kelas Commission yang menambahkan fitur komisi tanpa mengubah kelas lain.
3. Dari Studi Kasus 2, saya menyadari bagaimana metode yang sama (amount()) bisa bekerja untuk berbagai bentuk objek (Sphere, Rectangle, Cylinder) hanya dengan memanfaatkan satu kelas abstrak Shape sebagai contoh penerapan polimorfisme.
4. Dari Studi Kasus 3, saya belajar bahwa interface Comparable dapat digunakan untuk membuat objek dapat dibandingkan dan diurutkan, serta bagaimana algoritma seperti insertionSort dan selectionSort dapat berlaku untuk tipe data apa pun.
5. Secara keseluruhan, saya belajar bahwa pendekatan berorientasi objek bukan hanya soal membagi program menjadi banyak kelas, tetapi bagaimana setiap kelas saling berhubungan dengan jelas dan dapat digunakan kembali di berbagai situasi.