

LAPORAN
WEEK 4 PEMROGRAMAN BERBASIS OBJEK

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

Oleh:

Nama	: Raihana Aisha Az-Zahra
NIM	: 241511056
Kelas	: 2B
Program Studi	: D3 Teknik Informatika
Jurusan	: Teknik Komputer dan Informatika



POLITEKNIK NEGERI BANDUNG
KOTA BANDUNG
2025

Daftar Isi

BAB I PENDAHULUAN.....	3
1. Latar Belakang.....	3
2. Tujuan Praktikum.....	3
3. Manfaat Praktikum.....	3
BAB II HASIL Pengerjaan	5
1. Hasil Pengerjaan Task 1.1.....	5
Hasil Pengujian:.....	6
2. Hasil Pengerjaan Task 1.2.....	7
Hasil Pengujian:.....	9
3. Hasil Pengerjaan Task 1.3.....	10
Hasil Pengujian:.....	10
4. Hasil Pengerjaan Task 2.1.....	11
Instruksi Class Shape:.....	12
Instruksi Class Circle:.....	12
Instruksi Class Rectangle:	12
Instruksi Class Square:	13
Implementasi Program.....	14
Hasil Pengujian:.....	16
5. Implementasi Inheritance, Super, dan Overriding pada Kasus Karyawan	17
BAB III LESSON LEARNED	23

BAB I

PENDAHULUAN

1. Latar Belakang

Pemrograman berorientasi objek (Object-Oriented Programming/OOP) menyediakan berbagai konsep yang membantu dalam membangun perangkat lunak yang terstruktur, efisien, dan mudah dikembangkan. Salah satu konsep fundamentalnya adalah *generalization* dan *specialization*. *Generalization* adalah proses pengabstrakan dari beberapa kelas yang memiliki kesamaan menjadi sebuah kelas induk (*superclass*), sedangkan *specialization* adalah proses membuat kelas turunan (*subclass*) yang lebih spesifik dari kelas induknya.

Kedua konsep tersebut diimplementasikan melalui mekanisme *inheritance* (pewarisan). Dengan *inheritance*, sebuah *subclass* dapat mewarisi atribut dan metode dari *superclass*. Dalam praktiknya, *subclass* juga dapat memodifikasi perilaku metode yang diwarisi melalui metode *overriding*. Selain itu, bahasa pemrograman Java menyediakan kata kunci *super* untuk memanggil konstruktor maupun metode dari *superclass*, sehingga pewarisan dapat dilakukan secara fleksibel.

2. Tujuan Praktikum

- a. Mempelajari dan memahami konsep *generalization* dan *specialization* dalam OOP.
- b. Menjelaskan penerapan *inheritance* pada program berbasis java.
- c. Mengimplementasikan metode *overriding* pada *subclass*.
- d. Memahami fungsi kata kunci *super* untuk mengakses metode dan konstrukto *superclass*.

3. Manfaat Praktikum

- a. Memperoleh pemahaman praktis tentang penerapan *generalization* dan *specialization*.
- b. Mampu menerapkan *inheritance* untuk mengurangi duplikasi kode.

- c. Memahami perbedaan antara metode yang diwariskan dan metode yang dioverride.
- d. Mampu menggunakan super untuk mendukung implementasi yang lebih fleksibel.

BAB II

HASIL Pengerjaan

1. Hasil Pengerjaan Task 1.1

Pada tugas ini, dilakukan modifikasi terhadap kelas Circle sebagai superclass dari Cylinder. Tujuannya adalah untuk menambahkan atribut baru berupa color serta memberikan kemampuan untuk mengatur dan mengambil nilai dari atribut tersebut. Selain itu, ditambahkan juga constructor baru yang dapat menerima parameter radius dan color sekaligus.

Instruksi Utama:

- Tambahkan variabel color dengan tipe data String
- Buat constructor baru Circle(double radius, String color).
- Tambahkan getter dan setter untuk variabel color.

Implementasi Program Sebelum Mengerjakan Instruksi dari Task:


```
1 public class Circle{
2     private double radius;
3     private String color;
4
5     // Default constructor
6     public Circle(){
7         radius = 1.0;
8         color = "red";
9     }
10
11    // Constructor with radius (2nd constructor)
12    public Circle(double r){
13        radius = r;
14        color = "red";
15    }
16
17    // Returns the radius
18    public double getRadius(){
19        return radius;
20    }
21
22    // Returns the area of this Circle instance
23    public double getArea(){
24        return radius*radius*Math.PI;
25    }
26
27    // Return a self-descriptive string of this instance in the form of Circle[radius=?,color=?]
28    public String toString() {
29        return "\nCircle[radius=" + radius + " color=" + color + " ]";
30    }
31 }
32 }
```

Implementasi Instruksi Task 1.1	Penjelasan
---------------------------------	------------

<p>Tambahkan variabel color dengan tipe data String</p>  <pre> 1 public class Circle{ 2 private double radius; 3 private String color; 4 } </pre>	<p>Variabel color ditambahkan sebagai atribut baru dari kelas Circle</p>
<p>Buat constructor baru</p>  <pre> 1 /* Constructor with radius and color (task 1.1) 2 public Circle(double r, String c) { 3 radius = r; 4 color = c; 5 } </pre>	<p>Constructor Circle(double r, String c) memungkinkan objek Circle dibuat langsung dengan radius dan warna tertentu</p>
<p>Getter dan Setter untuk variable color</p>  <pre> 1 /* Getter for color (task 1.1) 2 public String getColor() { 3 return color; 4 } 5 6 /* Setter for color (task 1.1) 7 public void setColor(String c) { 8 color = c; 9 } </pre>	<p>Getter dan Setter untuk color berfungsi mengambil dan mengubah nilai warna setelah objek dibuat.</p>

Hasil Pengujian:

TestCircle.java



```

1  public class TestCircle{
2      public static void main(String[] args) {
3          Circle c1 = new Circle(2.0, "Blue");
4          System.out.println(c1.toString());
5          System.out.println("Old Color: "+c1.getColor());
6
7          c1.setColor("Green");
8          System.out.println(c1.toString());
9          System.out.println("Updated Color: "+c1.getColor());
10     }
11 }

```

Output di terminal

```
PS D:\Semester 3\PBO Praktek\W2\tasks\task-1> cd ".\src\testCircle.java" ; if ($?) { java TestCircle }

Circle[radius=2.0 color=Blue]
Old Color: Blue

Circle[radius=2.0 color=Green]
Updated Color: Green
```

2. Hasil Pengerjaan Task 1.2

Pada Task 1.2, konsep inheritance diterapkan dengan membuat kelas Cylinder yang merupakan subclass dari kelas Circle. Kelas Cylinder mewarisi semua atribut dan method dari Circle, seperti radius, color, serta method `getArea()` dan `toString()`. Namun, kelas ini menambahkan atribut baru yaitu height serta method `getVolume()` untuk menghitung volume tabung berdasarkan luas alas lingkaran dan tinggi. Dengan demikian, task ini menunjukkan bagaimana konsep generalisasi–spesialisasi dan pewarisan kode (inheritance) dapat dimanfaatkan untuk memperluas fungsionalitas kelas tanpa harus menulis ulang atribut dan method yang sudah ada di superclass.

Instruksi Utama:

- Subclass Cylinder mewarisi method `getArea()` dari superclass Circle
- Override method `getArea()` di dalam subclass Cylinder agar menghitung luas permukaan tabung dengan rumus: $\text{Surface area} = 2 \pi \times \text{radius} \times \text{height} + 2 \times \text{base-area}$
- Perbaiki method `getVolume()` agar tetap menggunakan base area (dari superclass Circle), bukan luas permukaan tabung

Implementasi Program Sebelum Mengerjakan Instruksi dari Task:

```

1  public class Cylinder extends Circle{
2      private double height;
3
4      // Constructor with default color, radius and height
5      public Cylinder(){
6          super(); // call superclass no-arg constructor Circle()
7          height = 1.0;
8      }
9
10     // Constructor with default radius, color but given height
11     public Cylinder(double height){
12         super();
13         this.height = height;
14     }
15
16     // Constructor with default color, but given radius, height
17     public Cylinder(double radius, double height){
18         super(radius); // call superclass constructor Circle(r)
19         this.height = height;
20     }
21
22     // A public method for retrieving the height
23     public double getHeight(){
24         return height;
25     }
26
27     // A public method for computing the volume of cylinder
28     // use superclass method getArea() to get the base area
29     public double getVolume(){
30         return super.getArea() * height;
31     }
32
33 }

```

Implementasi Instruksi Task 1.2	Penjelasan
<p>Override method getArea() di dalam subclass Cylinder</p> <pre> 1 public double getArea(){ 2 double baseArea = super.getArea(); // alas lingkaran 3 double circumference = 2 * Math.PI * getRadius(); 4 return 2 * baseArea + circumference * height; 5 } </pre>	<p>Method ini merupakan implementasi overriding dari getArea() yang ada di kelas Circle. Jika di kelas Circle hanya menghitung luas lingkaran, maka di kelas Cylinder method ini diubah agar menghitung luas permukaan tabung, yaitu dengan menjumlahkan dua kali luas alas lingkaran ($2 * \text{baseArea}$) dan luas selimut tabung ($\text{circumference} * \text{height}$).</p>
<p>Perbaiki method getVolume()</p>	<p>Agar tidak salah menggunakan getArea() yang sudah dioverride di</p>

<pre> 1 public double getVolume(){ 2 return super.getArea() * height; 3 } </pre>	<p>subclass, method ini memanggil <code>super.getArea()</code> dari kelas <code>Circle</code> sehingga yang dihitung adalah luas alas lingkaran, bukan luas permukaan tabung.</p>
--	---

Hasil Pengujian:

TestCylinder.java

```

1 public class TestCylinder{
2     public static void main(String[] args) {
3
4         // Declare and allocate a new instance of cylinder
5         // with default color, radius, and height
6         Cylinder c1 = new Cylinder();
7         System.out.println("Cylinder:"
8             + " radius=" + c1.getRadius()
9             + " height=" + c1.getHeight()
10            + " base area=" + c1.getArea()
11            + " volume=" + c1.getVolume());
12
13        Cylinder c2 = new Cylinder(10.0);
14        System.out.println("Cylinder:"
15            + " radius=" + c2.getRadius()
16            + " height=" + c2.getHeight()
17            + " base area=" + c2.getArea()
18            + " volume=" + c2.getVolume());
19
20        Cylinder c3 = new Cylinder(2.0, 10.0);
21        System.out.println("Cylinder:"
22            + " radius=" + c3.getRadius()
23            + " height=" + c3.getHeight()
24            + " base area=" + c3.getArea()
25            + " volume=" + c3.getVolume());
26    }
27 }

```

Output di Terminal:

```

PS D:\Semester 3\PBO Praktek\W2> cd "d:\Semester 3\PBO Praktek\W2\tasks\task-1\" ; if ($?) {
r }
Cylinder: radius=1.0 height=1.0 base area=12.566370614359172 volume=3.141592653589793
Cylinder: radius=1.0 height=10.0 base area=69.11503837897544 volume=31.41592653589793
Cylinder: radius=2.0 height=10.0 base area=150.79644737231007 volume=125.66370614359172
PS D:\Semester 3\PBO Praktek\W2\tasks\task-1>

```

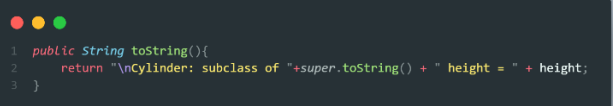
3. Hasil Pengerjaan Task 1.3

Pada task ini, tujuan utamanya adalah membuat method `toString()` di kelas `Cylinder` yang dapat menggantikan (override) method `toString()` milik kelas `Circle`. Dengan overriding ini, objek `Cylinder` akan menampilkan informasi yang lebih lengkap, yakni radius, color, dan juga height. Selain itu, method ini menggunakan `super.toString()` agar tetap menyertakan deskripsi bawaan dari kelas `Circle`, lalu menambahkan detail baru yang menjadi ciri khas dari kelas `Cylinder`, yaitu nilai height.

Instruksi Utama:

- Tambahkan method `toString()` di dalam kelas `Cylinder`
- Panggil `super.toString()` agar informasi dari kelas `Circle` tetap ditampilkan.
- Tambahkan informasi baru milik `Cylinder`, yaitu height

Implementasi Program:

Implementasi Instruksi Task 1.3	Penjelasan
 <pre>1 public String toString(){ 2 return "\nCylinder: subclass of "+super.toString() + " height = " + height; 3 }</pre>	Kode <code>toString()</code> di kelas <code>Cylinder</code> berfungsi untuk menampilkan deskripsi objek tabung dalam bentuk string yang lebih lengkap. Di dalamnya digunakan <code>super.toString()</code> untuk memanggil deskripsi dari kelas <code>Circle</code> , yaitu radius dan color, lalu ditambahkan informasi khusus milik <code>Cylinder</code> , yaitu height.

Hasil Pengujian:

TestCylinder.java

```

1 public class TestCylinder{
2     public static void main(String[] args) {
3
4         // Declare and allocate a new instance of cylinder
5         // with default color, radius, and height
6         Cylinder c1 = new Cylinder();
7         System.out.println(c1.toString());
8         System.out.println("Cylinder:");
9         + " radius=" + c1.getRadius()
10        + " height=" + c1.getHeight()
11        + " base area=" + c1.getArea()
12        + " volume=" + c1.getVolume());
13
14        Cylinder c2 = new Cylinder(10.0);
15        System.out.println(c2.toString());
16        System.out.println("Cylinder:");
17        + " radius=" + c2.getRadius()
18        + " height=" + c2.getHeight()
19        + " base area=" + c2.getArea()
20        + " volume=" + c2.getVolume());
21
22        Cylinder c3 = new Cylinder(2.0, 10.0);
23        System.out.println(c3.toString());
24        System.out.println("Cylinder:");
25        + " radius=" + c3.getRadius()
26        + " height=" + c3.getHeight()
27        + " base area=" + c3.getArea()
28        + " volume=" + c3.getVolume());
29    }
30 }

```

Output di Terminal:

```

PS D:\Semester 3\PBO Praktek\W2\tasks\task-1> cd "d:\Semester 3\PBO Praktek\W2\tasks\task-1\
a TestCylinder }

Cylinder: subclass of Circle[radius=1.0 color=red] height = 1.0
Cylinder: radius=1.0 height=1.0 base area=12.566370614359172 volume=3.141592653589793

Cylinder: subclass of Circle[radius=1.0 color=red] height = 10.0
Cylinder: radius=1.0 height=10.0 base area=69.11503837897544 volume=31.41592653589793

Cylinder: subclass of Circle[radius=2.0 color=red] height = 10.0
Cylinder: radius=2.0 height=10.0 base area=150.79644737231007 volume=125.66370614359172
PS D:\Semester 3\PBO Praktek\W2\tasks\task-1>

```

4. Hasil Pengerjaan Task 2.1

Pada task ini, tujuan utamanya adalah membuat hierarki class menggunakan konsep inheritance, dimulai dari superclass Shape yang memiliki atribut umum berupa color dan filled. Kemudian dibuat subclass Circle dan Rectangle yang menambahkan atribut khusus masing-masing, serta menghitung luas dan keliling. Dari Rectangle diturunkan lagi subclass Square yang mempertahankan sifat panjang sisi sama dengan cara overriding setter agar nilai width dan length selalu sama. Setiap class juga diminta untuk meng-override method toString() dengan memanfaatkan super.toString() agar informasi dari superclass tetap ditampilkan, lalu menambahkan detail spesifik sesuai bentuknya.

Instruksi Class Shape:

- a. Buat instance variable → color: String, filled: Boolean
- b. Buat constructor
 - No-arg constructor → color = “green”, filled = true
 - Constructor dengan parameter yang menerima color dan filled
- c. Buat getter dan setter → getColor(), setColor(String color), dan isFilled()
- d. Override toString() → return "A Shape with color of xxx and filled/Not filled"

Instruksi Class Circle:

- a. Buat instance variable → radius: double
- b. Buat constructor
 - No-arg constructor → radius = 1.0
 - Constructor dengan radius
 - Constructor dengan radius, color, dan filled (memakai super(color, filled) untuk memanggil superclass)
- c. Buat getter dan setter untuk radius
- d. Buat method
 - getArea() → $\pi * \text{radius} * \text{radius}$
 - getPerimeter() → $2 * \pi * \text{radius}$
- e. Override toString() → return " A Circle with radius=xxx, which is a subclass of yyy" (di mana yyy adalah super.toString()).

Instruksi Class Rectangle:

- a. Buat instance variable → width: double, length: double
- b. Buat constructor
 - No-arg constructor → width = 1.0, length = 1.0
 - Constructor dengan width dan length
 - Constructor dengan width, length, color, dan filled (memakai super(color, filled) untuk memanggil superclass)

- c. Buat getter dan setter untuk width dan length
- d. Buat method
 - `getArea()` → `width * length`
 - `getPerimeter()` → `2 * (width + length)`
- e. Override `toString()` → return "A Rectangle with width=xxx and length=zzz, which is a subclass of yyy" (di mana yyy adalah `super.toString()`).

Instruksi Class Square:

- a. Buat constructor
 - No-arg constructor → otomatis panggil Rectangle no-arg.
 - Constructor dengan side
 - Constructor dengan side, color, dan filled
- b. Override setter
 - `setSide(double side)` → set width dan length sama-sama side
 - `setWidth(double side)` → set width dan length sama-sama side
 - `setLength(double side)` → set width dan length sama-sama side
- c. Buat getter tambahan `getSide()` → bisa return width (atau length, sama saja).
- d. Override `toString()` → return "A Square with side=xxx, which is a subclass of yyy" (di mana yyy adalah `super.toString()`).

Implementasi Program

a. Shape.java

```
1 public class Shape{
2     private String color;
3     private Boolean filled;
4
5     public Shape(){
6         color="Green";
7         filled=true;
8     }
9
10    public Shape(String color, Boolean filled){
11        this.color=color;
12        this.filled=filled;
13    }
14
15    public String getColor(){
16        return color;
17    }
18    public void setColor(String color){
19        this.color = color;
20    }
21
22    public Boolean isFilled(){
23        return filled;
24    }
25
26    public void setFilled(boolean filled){
27        this.filled = filled;
28    }
29
30    public String toString(){
31        return "A shape with color of "+color+" and "+(filled?"filled":"not filled");
32    }
33 }
```

b. Circle2.java

```
1 public class Circle2 extends Shape{
2     private double radius;
3
4     public Circle2(){
5         radius=1.0;
6     }
7
8     public Circle2(double radius, String color, boolean filled){
9         super(color, filled);
10        this.radius = radius;
11    }
12
13    public Circle2(double radius) {
14        this.radius = radius;
15    }
16
17    public double getRadius(){
18        return radius;
19    }
20
21    public void setRadius(double radius){
22        this.radius = radius;
23    }
24
25    public double getArea(){
26        return radius*radius*Math.PI;
27    }
28
29    public double getPerimeter(){
30        return 2*radius*Math.PI;
31    }
32
33    public String toString() {
34        return "A Circle with radius=" + radius + ", which is a subclass of " + super.toString();
35    }
36 }
```

c. Rectangle.java

```
1 public class Rectangle extends Shape{
2     private double width;
3     private double length;
4
5     public Rectangle(){
6         width=1.0;
7         length=1.0;
8     }
9
10    public Rectangle(double width, double length){
11        this.width = width;
12        this.length = length;
13    }
14
15    public Rectangle(double width, double length, String color, boolean filled){
16        super(color,filled);
17        this.width = width;
18        this.length = length;
19    }
20
21    public double getWidth(){
22        return width;
23    }
24
25    public void setWidth(double width){
26        this.width = width;
27    }
28
29    public double getLength(){
30        return length;
31    }
32
33    public void setLength(double length){
34        this.length = length;
35    }
36
37    public double getArea(){
38        return width*length;
39    }
40
41    public double getPerimeter(){
42        return 2*(width+length);
43    }
44
45    public String toString() {
46        return "A Rectangle with width=" + width + " and length = " + length + ", which is a subclass of " + super.toString();
47    }
48 }
```

d. Square.java

```
1 public class Square extends Rectangle{
2     public Square(){
3         super(1.0,1.0);
4     }
5
6     public Square(double side) {
7         super(side, side);
8     }
9
10    public Square(double side, String color, boolean filled){
11        super(side,side,color,filled);
12    }
13
14    public double getSide(){
15        return getWidth();
16    }
17
18    public void setSide(double side){
19        super.setWidth(side);
20        super.setLength(side);
21    }
22
23    public void setWidth(double side){
24        setSide(side);
25    }
26
27    public void setLength(double side){
28        setSide(side);
29    }
30
31    public String toString() {
32        return "A Square with side=" + getSide() + ", which is a subclass of " + super.toString();
33    }
34
35 }
```

Hasil Pengujian:

TestShape.java

```
1  public class TestShape{
2      public static void main(String[] args) {
3
4          // Kumpulan objek
5          Shape s1 = new Shape("Blue",true);
6          Circle2 c1 = new Circle2(5.0, "Red", false);
7          Rectangle r1 = new Rectangle(4.0,6.0,"Yellow",true);
8          Square sq1 = new Square(3.0, "Purple", false);
9
10         System.out.println("=== Info Shape ===");
11         System.out.println(s1.toString());
12
13         System.out.println("\n=== Info Circle ===");
14         System.out.println(c1.toString());
15         System.out.println("Radius    : " + c1.getRadius());
16         System.out.println("Area      : " + c1.getArea());
17         System.out.println("Perimeter: " + c1.getPerimeter());
18
19         System.out.println("\n=== Info Rectangle ===");
20         System.out.println(r1.toString());
21         System.out.println("Width    : " + r1.getWidth());
22         System.out.println("Length   : " + r1.getLength());
23         System.out.println("Area     : " + r1.getArea());
24         System.out.println("Perimeter: " + r1.getPerimeter());
25
26         System.out.println("\n=== Info Square ===");
27         System.out.println(sq1.toString());
28         System.out.println("Side     : " + sq1.getSide());
29         System.out.println("Area     : " + sq1.getArea());
30         System.out.println("Perimeter: " + sq1.getPerimeter());
31     }
32 }
```

Output Terminal

```
PS D:\Semester 3\PBO Praktek\W2> cd "d:\Semester 3\PBO Praktek\W2\tasks\task-2\" ; if ($?) { javac TestShape.java } ; if ($?) { java TestShape }
=== Info Shape ===
A shape with color of Blue and filled

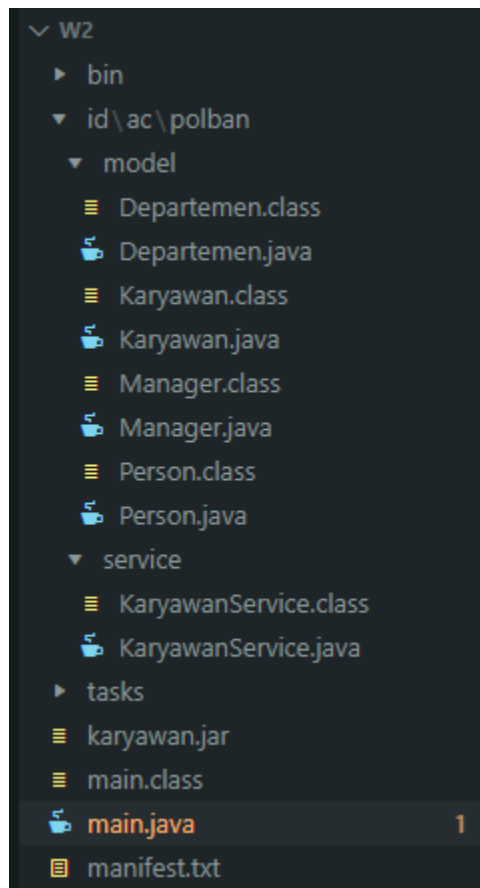
=== Info Circle ===
A Circle with radius=5.0, which is a subclass of A shape with color of Red and not filled
Area      : 78.53981633974483
Perimeter: 31.41592653589793

=== Info Rectangle ===
A Rectangle with width=4.0 and length = 6.0, which is a subclass of A shape with color of Yellow and filled
Width     : 4.0
Length    : 6.0
Area      : 24.0
Perimeter: 20.0

=== Info Square ===
A Square with side=3.0, which is a subclass of A Rectangle with width=3.0 and length = 3.0, which is a subclass of A shape with color of Purple and not filled
Side      : 3.0
Area      : 9.0
Perimeter: 12.0
PS D:\Semester 3\PBO Praktek\W2\tasks\task-2>
```


5. Implementasi Inheritance, Super, dan Overriding pada Kasus Karyawan

a. Struktur Folder



a. Implementasi Program

Person.java – superclass

```

1  package id.ac.polban.model;
2
3  public abstract class Person {
4      private static int counter = 1;
5      private int id;
6      private String name;
7
8      // Konstruktor default (counter id otomatis)
9      public Person() {
10         this.id = counter++;
11     }
12
13     // Konstruktor dengan nama
14     public Person(String name) {
15         this();
16         this.name = name;
17     }
18
19     // Getter / Setter
20     public int getId() { return id; }
21     public String getName() { return name; }
22     public void setName(String name) { this.name = name; }
23
24     // Display dasar (dapat di-override oleh subclass)
25     public void display() {
26         System.out.println("ID: " + id);
27         System.out.println("Name: " + name);
28     }
29
30     // Input data dasar (dapat di-override)
31     public void inputData(java.util.Scanner scanner) {
32         System.out.print("Nama: ");
33         this.name = scanner.nextLine();
34     }
35 }

```

Kelas Person merupakan sebuah kelas yang berfungsi sebagai superclass dan menjadi dasar bagi pembentukan kelas turunan. Kelas ini memiliki atribut id yang diatur secara otomatis menggunakan variabel statis counter. Selain itu, terdapat atribut name yang merepresentasikan nama dari objek. Konstruktor pada kelas ini terdiri atas dua bentuk, yaitu konstruktor default yang hanya mengatur nilai id, serta konstruktor dengan parameter name yang memanfaatkan konstruktor default untuk tetap menjaga konsistensi pemberian nilai id. Kelas ini juga menyediakan metode display() yang menampilkan informasi dasar berupa id dan name, serta metode inputData() untuk melakukan pengisian data nama melalui input pengguna. Dengan rancangan tersebut, kelas Person digunakan untuk diwariskan pada kelas lain agar atribut dan metode dasar dapat digunakan kembali maupun disesuaikan sesuai kebutuhan spesifik dari kelas turunan.

Karyawan.java – subclass

```
1 package id.ac.polban.model;
2
3 public class Karyawan extends Person {
4     private Departemen departemen;
5     private int gaji;
6
7     // Konstruktor default memanggil super()
8     public Karyawan() {
9         super();
10    }
11
12    // Konstruktor dengan parameter
13    public Karyawan(String name, Departemen departemen, int gaji) {
14        super(name);
15        this.departemen = departemen;
16        this.gaji = gaji;
17    }
18
19    // Getter / Setter
20    public Departemen getDepartemen() { return departemen; }
21    public int getGaji() { return gaji; }
22    public void setDepartemen(Departemen departemen) { this.departemen = departemen; }
23    public void setGaji(int gaji) { this.gaji = gaji; }
24
25    // Override display
26    @Override
27    public void display() {
28        super.display();
29        if (departemen != null) {
30            System.out.println("Departemen: " + departemen.getName());
31            System.out.println("Deskripsi: " + departemen.getDesc());
32        }
33        System.out.println("Gaji: " + gaji);
34        System.out.println("-----");
35    }
36
37    // Override inputData
38    @Override
39    public void inputData(java.util.Scanner scanner) {
40        super.inputData(scanner);
41        System.out.print("Gaji: ");
42        this.gaji = scanner.nextInt();
43        scanner.nextLine();
44    }
45 }
```

Kelas Karyawan merupakan turunan dari kelas Person yang digunakan untuk merepresentasikan seorang karyawan dalam sistem. Pada kelas ini ditambahkan atribut baru, yaitu departemen yang menunjukkan unit kerja karyawan, serta gaji yang menyimpan jumlah gaji pokok. Konstruktor pada kelas Karyawan terdiri dari konstruktor default yang memanggil konstruktor superclass (Person) tanpa parameter, dan konstruktor dengan parameter name, departemen, serta gaji yang memanfaatkan konstruktor superclass dengan parameter name untuk menginisialisasi data dasar, kemudian menambahkan nilai departemen dan gaji. Kelas ini menyediakan metode getter dan setter untuk mengakses maupun mengubah nilai departemen serta gaji. Selain itu, metode display() ditimpa dari superclass agar tidak hanya menampilkan informasi dasar berupa id dan name, tetapi juga informasi departemen (jika tersedia), deskripsi departemen, serta gaji karyawan. Demikian pula, metode inputData() ditimpa untuk menambahkan proses input gaji setelah pengisian data dasar dari Person. Dengan rancangan tersebut, kelas Karyawan memperluas fungsionalitas dari Person dengan menambahkan data dan

perilaku yang lebih spesifik sesuai kebutuhan representasi seorang karyawan.

Manager.java – subclass, inheritance, override, super

```
1 package id.ac.polban.model;
2
3 public class Manager extends Karyawan {
4     private int tunjangan;
5
6     public Manager() {
7         super();
8     }
9
10    public Manager(String name, Departemen departemen, int gaji, int tunjangan) {
11        super(name, departemen, gaji);
12        this.tunjangan = tunjangan;
13    }
14
15    public int getTunjangan() { return tunjangan; }
16    public void setTunjangan(int tunjangan) { this.tunjangan = tunjangan; }
17
18    // Override display
19    @Override
20    public void display() {
21        super.display();
22        System.out.println("Tunjangan: " + tunjangan);
23        System.out.println("-----");
24    }
25
26    // Override inputData
27    @Override
28    public void inputData(java.util.Scanner scanner) {
29        super.inputData(scanner); // input nama & gaji dari Karyawan
30        System.out.print("Tunjangan: ");
31        this.tunjangan = scanner.nextInt();
32        scanner.nextLine();
33    }
34
35    // Override getGaji
36    @Override
37    public int getGaji() {
38        return super.getGaji() + tunjangan;
39    }
40 }
```

Kelas Manager merupakan kelas turunan dari Karyawan yang merepresentasikan seorang karyawan dengan jabatan manajerial. Pada kelas ini ditambahkan atribut khusus berupa tunjangan, yang membedakan manajer dari karyawan biasa. Konstruktor yang tersedia terdiri atas konstruktor default yang memanggil konstruktor induk tanpa parameter, serta konstruktor dengan parameter name, departemen, gaji, dan tunjangan yang memanfaatkan konstruktor milik kelas Karyawan untuk menginisialisasi data dasar, kemudian menambahkan nilai tunjangan. Kelas ini juga menyediakan metode getTunjangan dan setTunjangan untuk mengakses maupun mengubah nilai tunjangan. Selanjutnya, terdapat beberapa metode yang menimpa (override) metode dari superclass, antara lain display() yang menambahkan informasi tunjangan setelah

menampilkan data karyawan, inputData() yang selain menerima input dasar dari kelas Karyawan juga meminta input tunjangan, serta getGaji() yang dimodifikasi sehingga nilai gaji yang dikembalikan merupakan hasil penjumlahan antara gaji dasar dan tunjangan. Dengan demikian, kelas Manager tidak hanya mewarisi atribut dan perilaku dari Karyawan, tetapi juga memperluas fungsionalitasnya agar sesuai dengan kebutuhan representasi seorang manajer.

b. Hasil Program

```
Masukan jumlah karyawan: 2

Karyawan ke - 1
Apakah karyawan ini seorang manager? (y/n): y
Nama: Asep
Gaji: 10000000
Tunjangan: 3750000
Pilih Departemen:
1. IT
Masukan nomor departemen: 1

Karyawan ke - 2
Apakah karyawan ini seorang manager? (y/n): n
Nama: Budi
Gaji: 5000000
Pilih Departemen:
1. IT
Masukan nomor departemen: 1
```

Menambah karyawan, di mana salah 1 karyawan merupakan manager

Pilih Karyawan untuk Cetak Slip Gaji:

1. Asep
2. Budi

1

Slip Gaji untuk Asep

Departemen: IT

Gaji: 13750000

=====

Selamat Datang di Aplikasi Manajemen Karyawan!

1. Tambah Departemen
2. Tambah Karyawan
3. Lihat Data
4. Cetak Slip Gaji
5. Keluar

Silakan Pilih Menu: 4

Pilih Karyawan untuk Cetak Slip Gaji:

1. Asep
2. Budi

2

Slip Gaji untuk Budi

Departemen: IT

Gaji: 5000000

=====

Perbedaan Cetak Slip Gaji antara manager dengan karyawan

BAB III

LESSON LEARNED

Lesson learned dari praktikum ini adalah saya semakin memahami penerapan pewarisan (inheritance), pemanggilan konstruktor superclass dengan super, serta penyesuaian perilaku method melalui override. Pada tahap awal, yaitu task 1.1, 1.2, dan 1.3, saya berlatih dasar-dasar pengenalan konsep inheritance dengan membuat kelas dasar dan kelas turunan sederhana, mempraktikkan bagaimana atribut serta method dari superclass dapat digunakan kembali oleh subclass, serta mengenal peran super dalam pemanggilan konstruktor induk. Kemudian pada task 2.1, saya mulai menerapkan override untuk menyesuaikan method yang diwariskan, sehingga subclass dapat memiliki perilaku yang lebih spesifik tanpa menghilangkan fungsi dasar dari superclass. Melalui rangkaian latihan ini hingga implementasi kelas Person, Karyawan, dan Manager, saya menyadari bahwa desain hierarki kelas yang baik membuat kode menjadi lebih efisien, mudah diperluas, serta mengurangi duplikasi. Praktikum ini menegaskan pentingnya pemahaman konsep inheritance, super, dan override sebagai dasar utama dalam membangun sistem yang terstruktur dan berorientasi objek.