**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](#_Toc208657745)

[1. Latar Belakang 3](#_Toc208657746)

[2. Tujuan Praktikum 3](#_Toc208657747)

[3. Manfaat Praktikum 3](#_Toc208657748)

[BAB II HASIL PENGERJAAN 5](#_Toc208657749)

[1. Hasil Pengerjaan Task 1.1 5](#_Toc208657750)

[Hasil Pengujian: 6](#_Toc208657751)

[2. Hasil Pengerjaan Task 1.2 7](#_Toc208657752)

[Hasil Pengujian: 9](#_Toc208657753)

[3. Hasil Pengerjaan Task 1.3 10](#_Toc208657754)

[Hasil Pengujian: 10](#_Toc208657755)

[4. Hasil Pengerjaan Task 2.1 11](#_Toc208657756)

[Instruksi Class Shape: 12](#_Toc208657757)

[Instruksi Class Circle: 12](#_Toc208657758)

[Instruksi Class Rectangle: 12](#_Toc208657759)

[Instruksi Class Square: 13](#_Toc208657760)

[Implementasi Program 14](#_Toc208657761)

[Hasil Pengujian: 16](#_Toc208657762)

[5. Implementasi Inheritance, Super, dan Overriding pada Kasus Karyawan 17](#_Toc208657763)

[BAB III LESSON LEARNED 23](#_Toc208657764)

# **BAB I PENDAHULUAN**

## **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.

## **Tujuan Praktikum**

1. Mempelajari dan memahami konsep *generalization* dan *specialization* dalam OOP.
2. Menjelaskan penerapan *inheritance* pada program berbasis java.
3. Mengimplementasikan metode *overriding* pada *subclass.*
4. Memahami fungsi kata kunci *super* untuk mengakses metode dan konstrukto *superclass.*

## **Manfaat Praktikum**

1. Memperoleh pemahaman praktis tentang penerapan *generalization* dan *specialization*.
2. Mampu menerapkan *inheritance* untuk mengurangi duplikasi kode.
3. Memahami perbedaan antara metode yang diwariskan dan metode yang dioverride.
4. Mampu menggunakan super untuk mendukung implementasi yang lebih fleksibel.

# **BAB II HASIL PENGERJAAN**

## **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:**

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

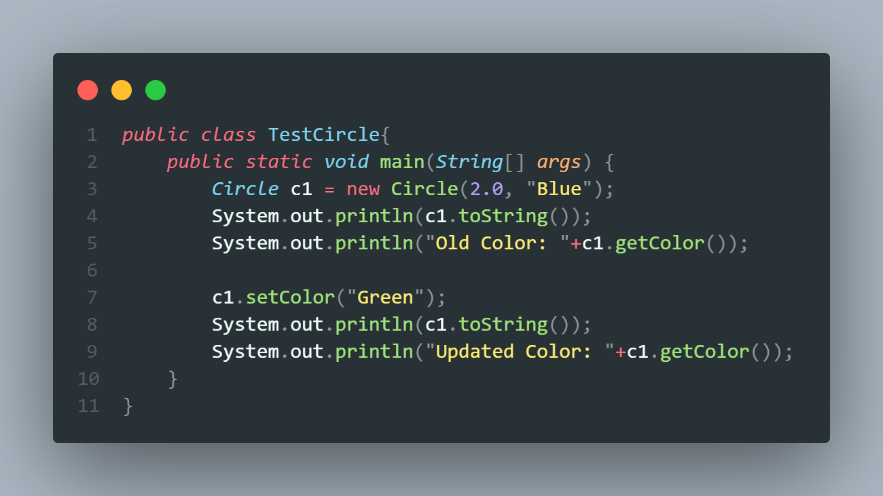
**Implementasi Program Sebelum Mengerjakan Instruksi dari Task:**



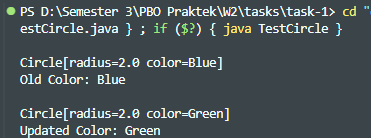
|  |  |
| --- | --- |
| **Implementasi Instruksi Task 1.1** | **Penjelasan** |
| Tambahkan variabel color dengan tipe data String | Variabel color ditambahkan sebagai atribut baru dari kelas Circle |
| Buat constructor baru | Constructor Circle(double r, String c) memungkinkan objek Circle dibuat langsung dengan radius dan warna tertentu |
| Getter dan Setter untuk variable color | Getter dan Setter untuk color berfungsi mengambil dan mengubah nilai warna setelah objek dibuat. |

### **Hasil Pengujian:**

**TestCircle.java**



Output di terminal



## **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:**

1. Subclass Cylinder mewarisi method getArea() dari superclass Circle
2. Override method getArea() di dalam subclass Cylinder agar menghitung luas permukaan tabung dengan rumus: Surface area = 2 π x radius x height + 2 x base-area
3. Perbaiki method getVolume() agar tetap menggunakan base area (dari superclass Circle), bukan luas permukaan tabung

**Implementasi Program Sebelum Mengerjakan Instruksi dari Task:**



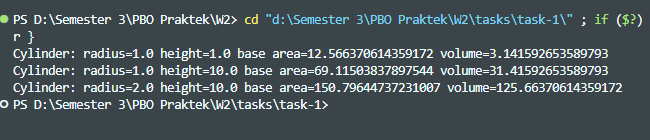
|  |  |
| --- | --- |
| **Implementasi Instruksi Task 1.2** | **Penjelasan** |
| Override method getArea() di dalam subclass Cylinder | Method ini merupakan implementasi overriding dari getArea() yang ada di kelas Circle. Jika di kelas Circle hanya menghitung luas lingkaran, maka di kelas Cylinder method ini diubah agar menghitung luas permukaan tabung, yaitu dengan menjumlahkan dua kali luas alas lingkaran (2 \* baseArea) dan luas selimut tabung (circumference \* height). |
| Perbaiki method getVolume() | Agar tidak salah menggunakan getArea() yang sudah dioverride di subclass, method ini memanggil super.getArea() dari kelas Circle sehingga yang dihitung adalah luas alas lingkaran, bukan luas permukaan tabung. |

### **Hasil Pengujian:**

**TestCylinder.java**



Output di Terminal:



## **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:**

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

**Implementasi Program:**

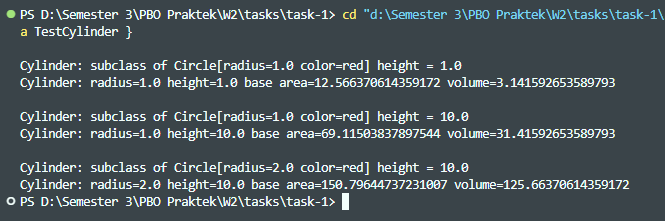
|  |  |
| --- | --- |
| **Implementasi Instruksi Task 1.3** | **Penjelasan** |
|  | Kode toString() di kelas Cylinder berfungsi untuk menampilkan deskripsi objek tabung dalam bentuk string yang lebih lengkap. Di dalamnya digunakan super.toString() untuk memanggil deskripsi dari kelas Circle, yaitu radius dan color, lalu ditambahkan informasi khusus milik Cylinder, yaitu height. |

### **Hasil Pengujian:**

**TestCylinder.java**



Output di Terminal:



## **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:**

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

### **Instruksi Class Circle:**

1. Buat instance variable → radius: double
2. Buat constructor
   * No-arg constructor → radius = 1.0
   * Constructor dengan radius
   * Constructor dengan radius, color, dan filled (memakai super(color, filled) untuk memanggil superclass)
3. Buat getter dan setter untuk radius
4. Buat method
   * getArea() → π \* radius \* radius
   * getPerimeter() → 2 \* π \* radius
5. Override toString() → return " A Circle with radius=xxx, which is a subclass of yyy" (di mana yyy adalah super.toString()).

### **Instruksi Class Rectangle:**

1. Buat instance variable → width: double, length: double
2. 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)
3. Buat getter dan setter untuk width dan length
4. Buat method
   * getArea() → width \* length
   * getPerimeter() → 2 \* (width + length)
5. 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:**

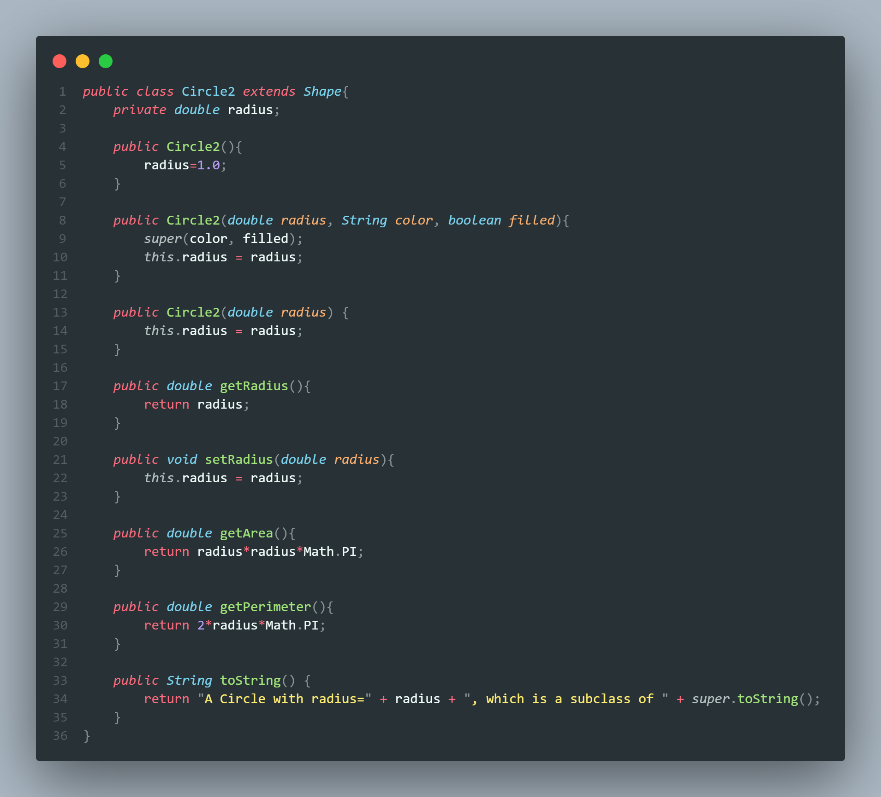
1. Buat constructor
   * No-arg constructor → otomatis panggil Rectangle no-arg.
   * Constructor dengan side
   * Constructor dengan side, color, dan filled
2. Override setter
   * setSide(double side) → set width dan length sama-sama side
   * setWidth(double side) → set width dan length sama-sama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

### **Implementasi Program**

#### **Shape.java**



#### **Circle2.java**



#### **Rectangle.java**



#### **Square.java**

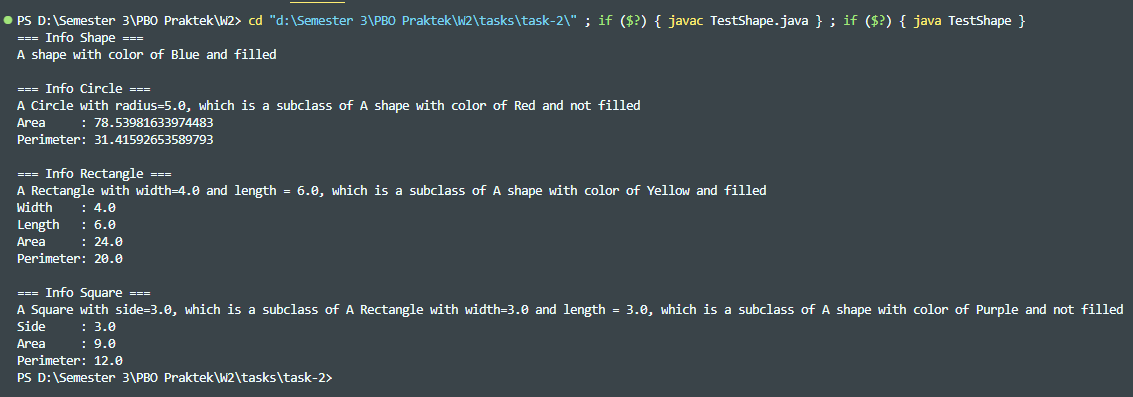


### **Hasil Pengujian:**

**TestShape.java**



**Output Terminal**

****

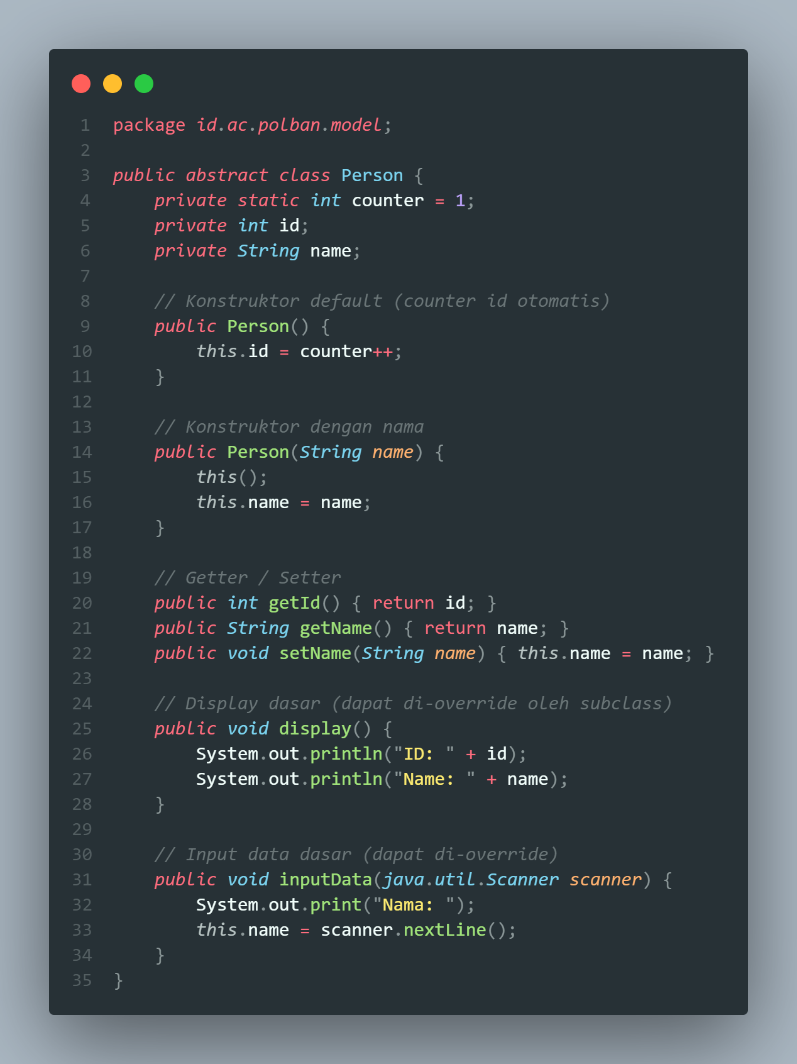
## **Implementasi Inheritance, Super, dan Overriding pada Kasus Karyawan**

1. **Struktur Folder**

****

1. **Implementasi Program**

**Person.java – superclass**



Kelas Person merupakan sebuah kelasyang 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**



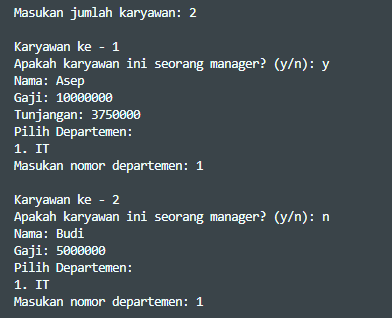
Kelas Karyawan merupakan turunan dari kelas Person yang digunakan untuk merepresentasikan seorang karyawan dalam sistem. Pada kelas ini ditambahkan atribut baru, yaitu departemen yang menunjukkan unit kerja karyawan, serta gaji yang menyimpan jumlah gaji pokok. Konstruktor pada kelas Karyawan terdiri 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**

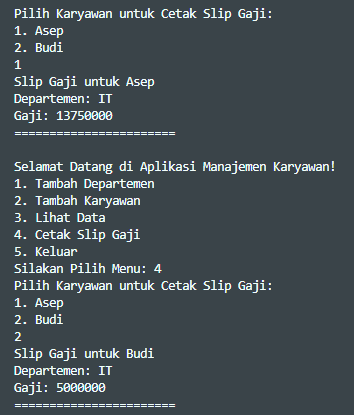


Kelas Manager merupakan kelas turunan dari Karyawan yang merepresentasikan seorang karyawan dengan jabatan manajerial. Pada kelas ini ditambahkan atribut khusus berupa tunjangan, yang membedakan manajer dari karyawan biasa. Konstruktor yang tersedia terdiri atas konstruktor default yang memanggil konstruktor 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.

1. **Hasil Program**

****

Menambah karyawan, di mana salah 1 karyawan merupakan manager



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.