**LAPORAN RESMI**

**MODUL V**

**POLIMORFISME**

**PEMOGRAMAN BERBASIS OBJEK**

**Disetujui : Mei 2024**

**Asisten**

**MUHAMMAD ROSYID MAULANA 20.04.411.00019**

**NAMA : ABAS PERMADANI**

**N.R.P : 230441100107**

**DOSEN : ACHMAD ZAIN NUR S.Kom M.T**

**ASISTEN : MUHAMMAD ROSYID MAULANA**

**TGL PRAKTIKUM : 18 MEI 2024**

**LABORATORIUM BISNIS INTELIJEN SISTEM**

**PRODI SISTEM INFORMASI**

**JURUSAN TEKNIK INFORMATIKA**

**FAKULTAS TEKNIK**

**UNIVERSITAS TRUNOJOYO MADURA**

# **BAB I PENDAHULUAN**

## Latar Belakang

Polimorfisme adalah salah satu konsep kunci dalam paradigma pemrograman berorientasi objek yang memungkinkan objek untuk menunjukkan perilaku yang berbeda tergantung pada konteks pemanggilannya. Istilah "polimorfisme" berasal dari bahasa Yunani yang berarti "banyak bentuk". Dalam konteks pemrograman, polimorfisme mengacu pada kemampuan suatu fungsi atau metode untuk melakukan tugas yang berbeda tergantung pada tipe data atau objek yang diterimanya. Konsep ini memungkinkan pengembang untuk menulis kode yang lebih fleksibel dan mudah diubah, dengan kemampuan untuk menggunakan objek dengan cara yang sama tanpa memperhatikan jenis objek yang sebenarnya.

Polimorfisme berasal dari konsep (inheritance) dalam OOP. Dalam pewarisan, kelas dapat mewarisi atribut dan metode dari kelas lain, yang memungkinkan untuk mengorganisasi kode secara hierarkis dan memanfaatkan kembali kode yang ada. Dengan kata lain, polimorfisme memungkinkan kelas turunan untuk memiliki metode dengan nama yang sama tetapi perilaku yang berbeda.

Di sisi lain, Polimorfisme memiliki banyak manfaat dalam pengembangan perangkat lunak. Memungkinkan untuk menulis kode yang lebih generik dan mudah dipelihara dengan memperlakukan objek secara umum tanpa perlu memperhatikan detail implementasi yang spesifik. Polimorfisme juga memfasilitasi konsep abstraksi, di mana pengembang dapat bekerja dengan objek secara umum tanpa perlu mengetahui detail implementasi yang spesifik. Dengan demikian, polimorfisme adalah salah satu aspek yang sangat penting dalam paradigma pemrograman berorientasi objek dan telah menjadi fondasi untuk pembangunan berbagai aplikasi perangkat lunak yang kompleks dan scalable.

## Tujuan

* Mahasiswa mampu memahami konsep Polimorfisme dalam Pemrograman Berorientasi Objek serta mampu mengimplementasikannya.

# **BAB II DASAR TEORI**

## Pengertian Polimerfisme

Polimorfime adalah suatu konsep dasar java. Polimorfime berasal dari dua kata yaitu poli dan morphs, poli berarti banyak morphs berarti bentuk. Polimorfisme juga dapat diartikan sebagai metode dengan penamaan yang sama namun memiliki respon yang berbeda-beda. Implementasi dari polimorfime sendiri adalah Override, Overloading, dan Polimorfisme Runtime.

## Override

Teknik ini telah dijelaskan pada bagian sebelumnya teknik ini adalah teknik menempa atau menulis ulang method superclass pada subclass. Penggunaanya adalah dengan menulis ulang nama method pada bagian subclass.

## Overloading

Overloading adalah kemampuan sebuah class memiliki beberapa method

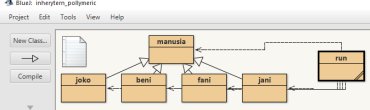
dengan nama yang sama, overloading bisa juga disebut Polimorfisme static. Teknik ini dapat digunakan apabila variablenya berbeda, teknik ini akan dibahas

lebih lanjut nanti.

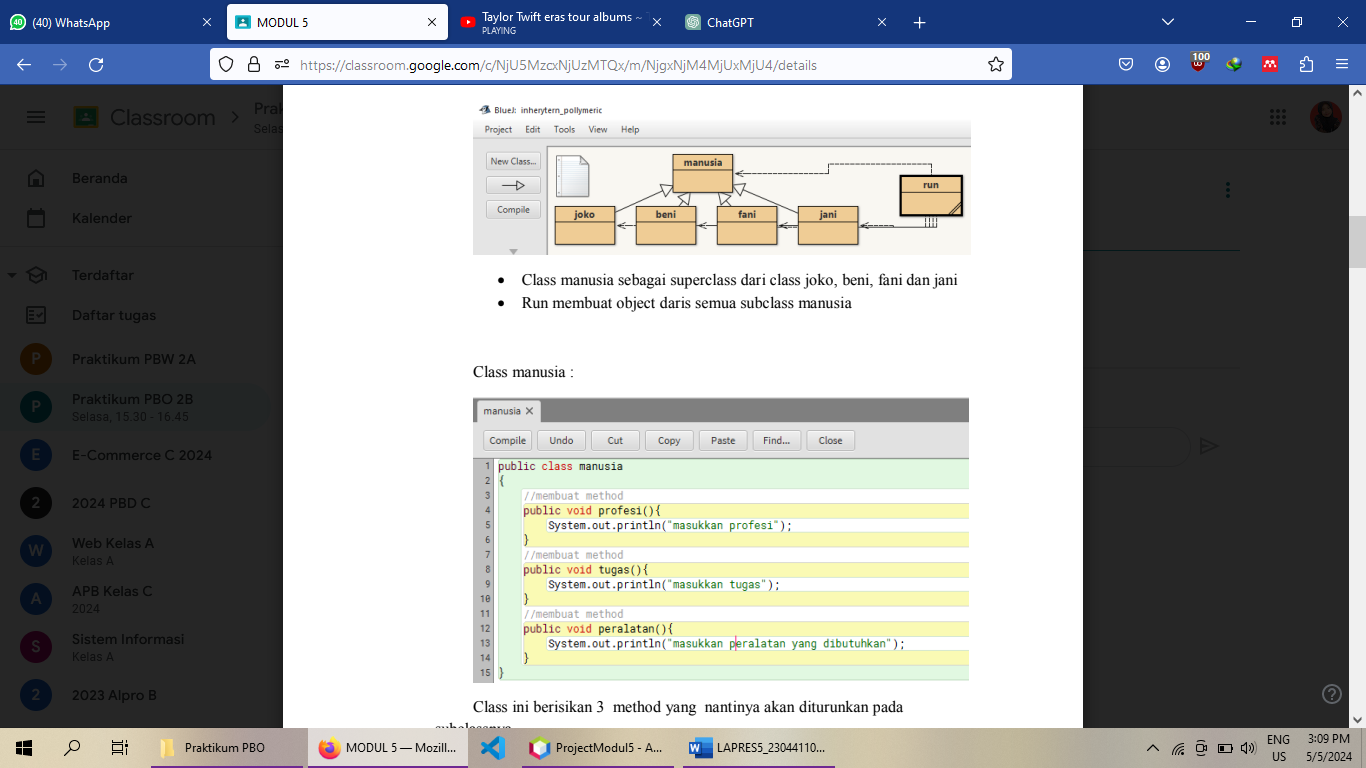
## Polimorfisme Runtime

Polimorfisme Runtime seperti namanya polimorfisme ini terjadi ketika program dijalankan, dalam implementasinya menggunakan metode inheritance dan override. Dalam membuat object biasa “manusia a = new manusia” manusia yang pertama berfungsi sebagai variable dan yang berikutnya adalah object, namun sekarang object dibuat dengan variable dan object yang berberda “makhluk\_hidup a = new manusia” makhluk hidup disini adalah superclass sedangkan manusia adalah subclassnya.

## Contoh Program

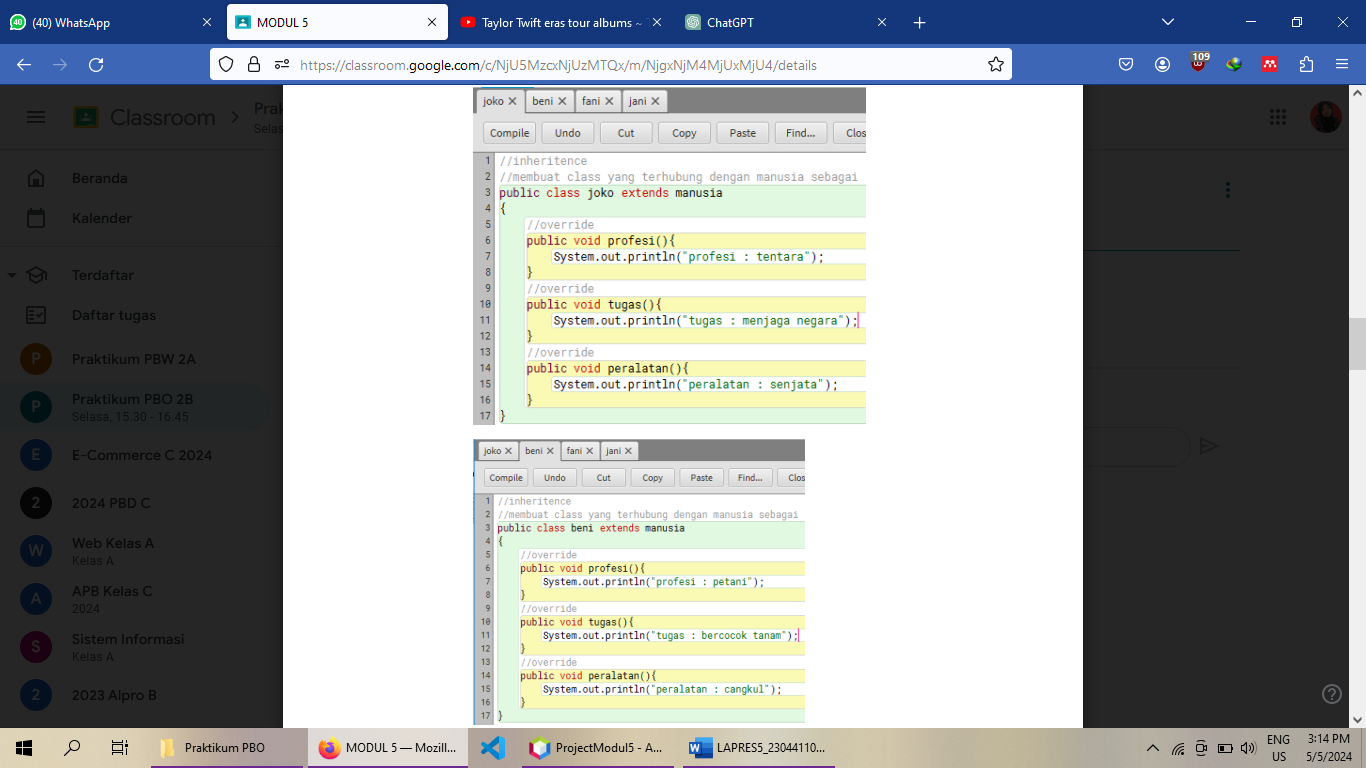
****

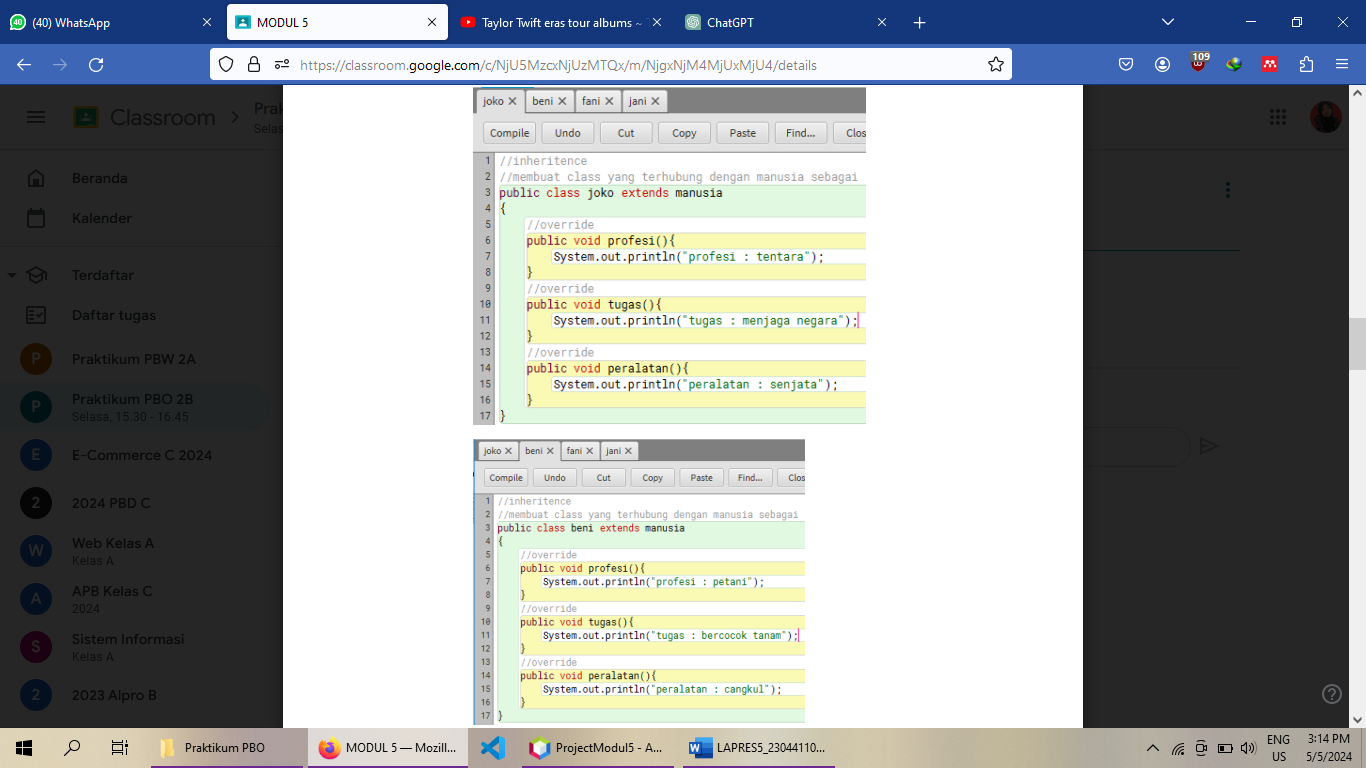
* Class manusia sebagai superclass dari class joko, beni, fani dan jani
* Run membuat object daris semua subclass manusia

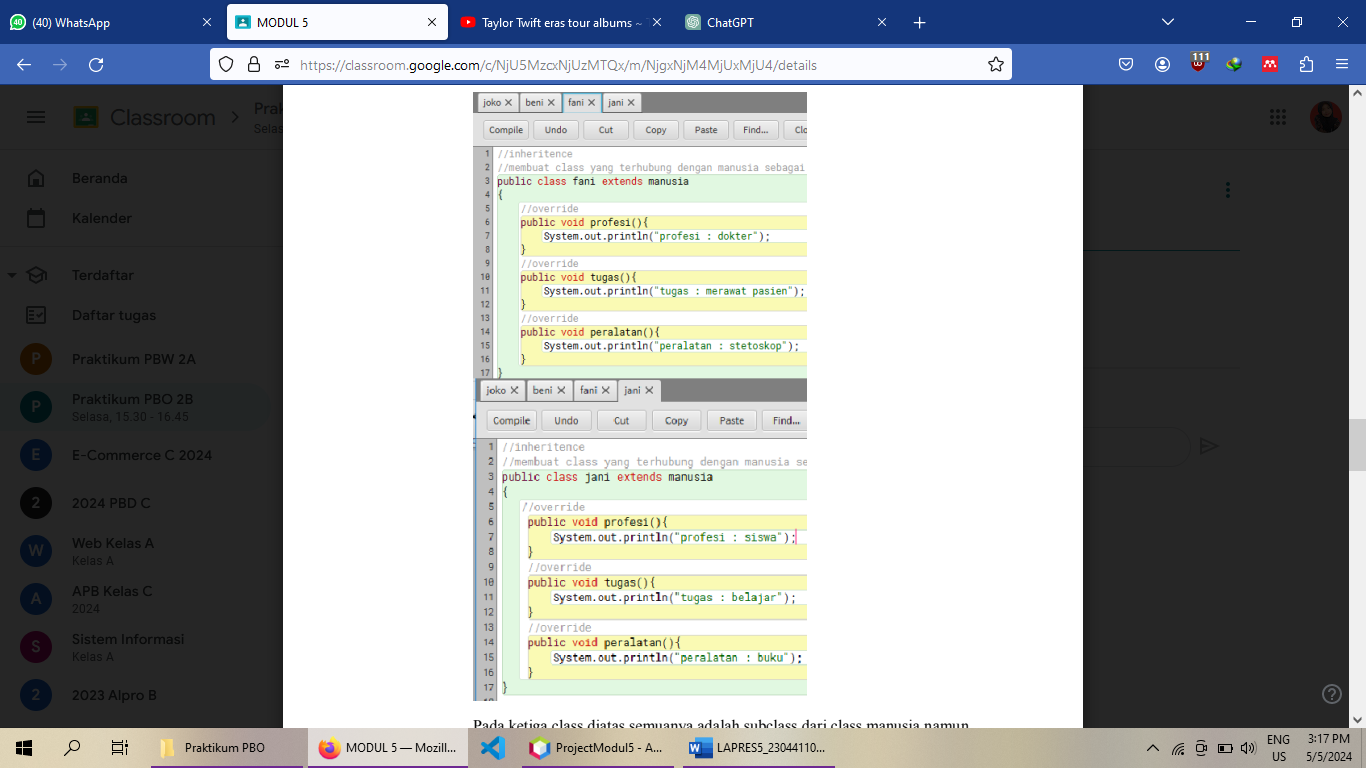
Class manusia :

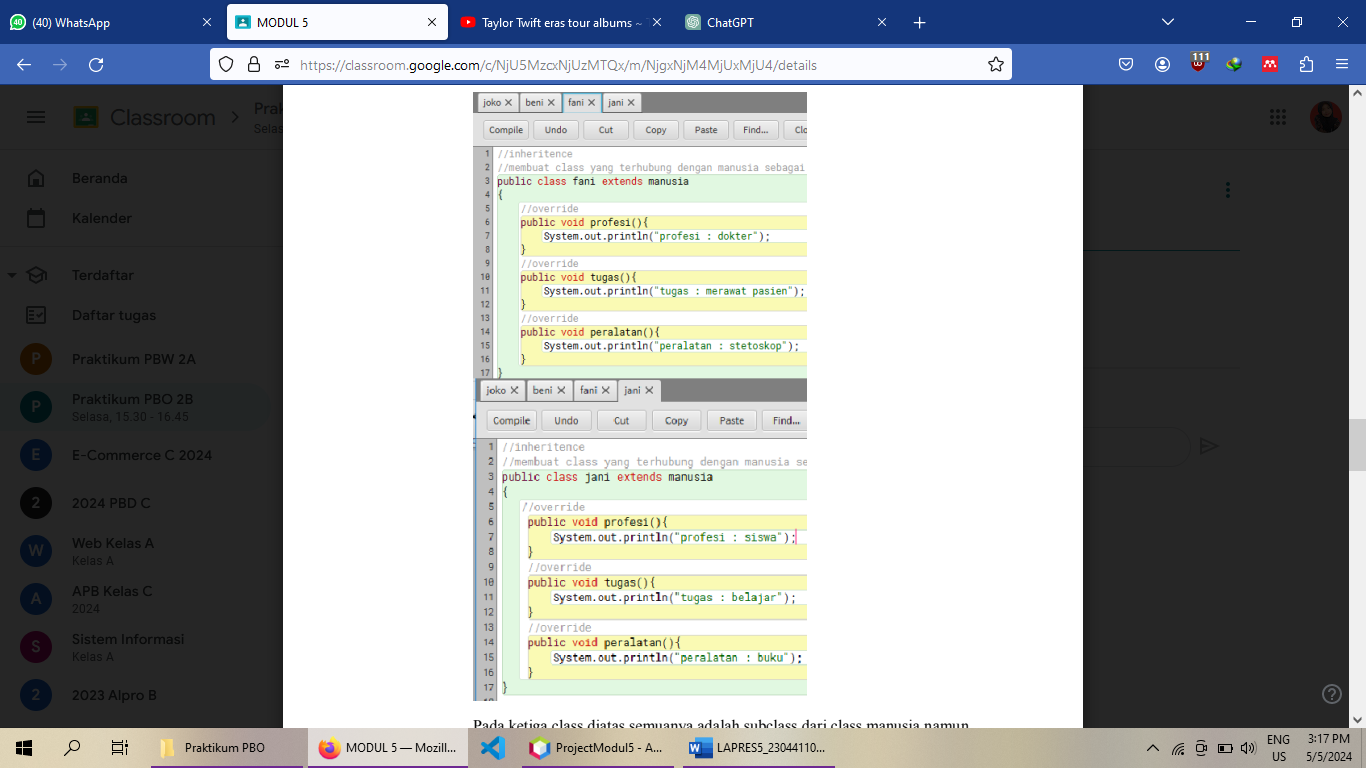
Class ini berisikan 3 method yang nantinya akan diturunkan pada subclassnya.

Class joko, beni, fani, dan jani :



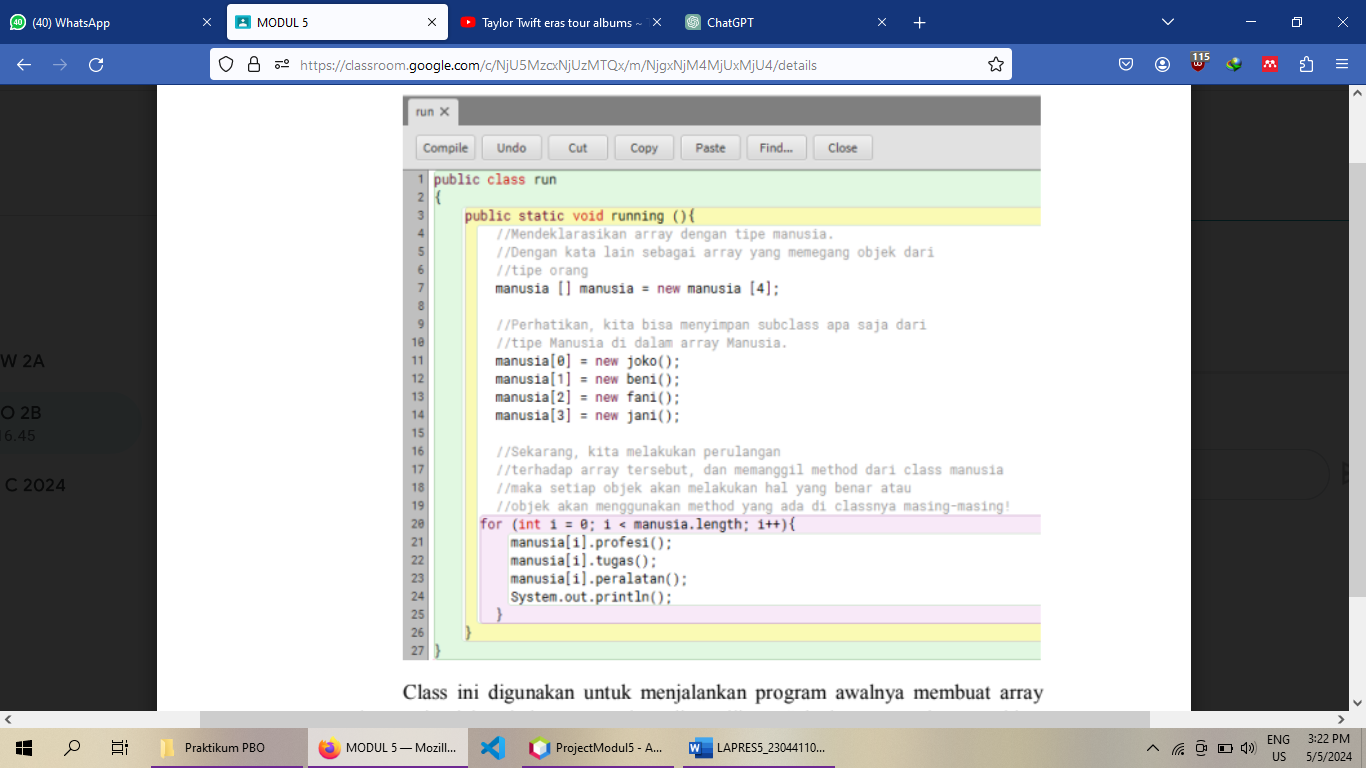


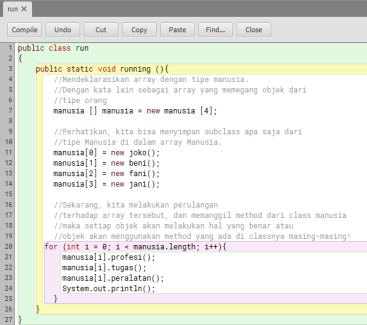




Pada keempat class diatas semuanya adalah subclass dari class manusia namun semuanya dilakukan proses inheritence dan disesuaikan dengan kebutuhannya masing-masing.

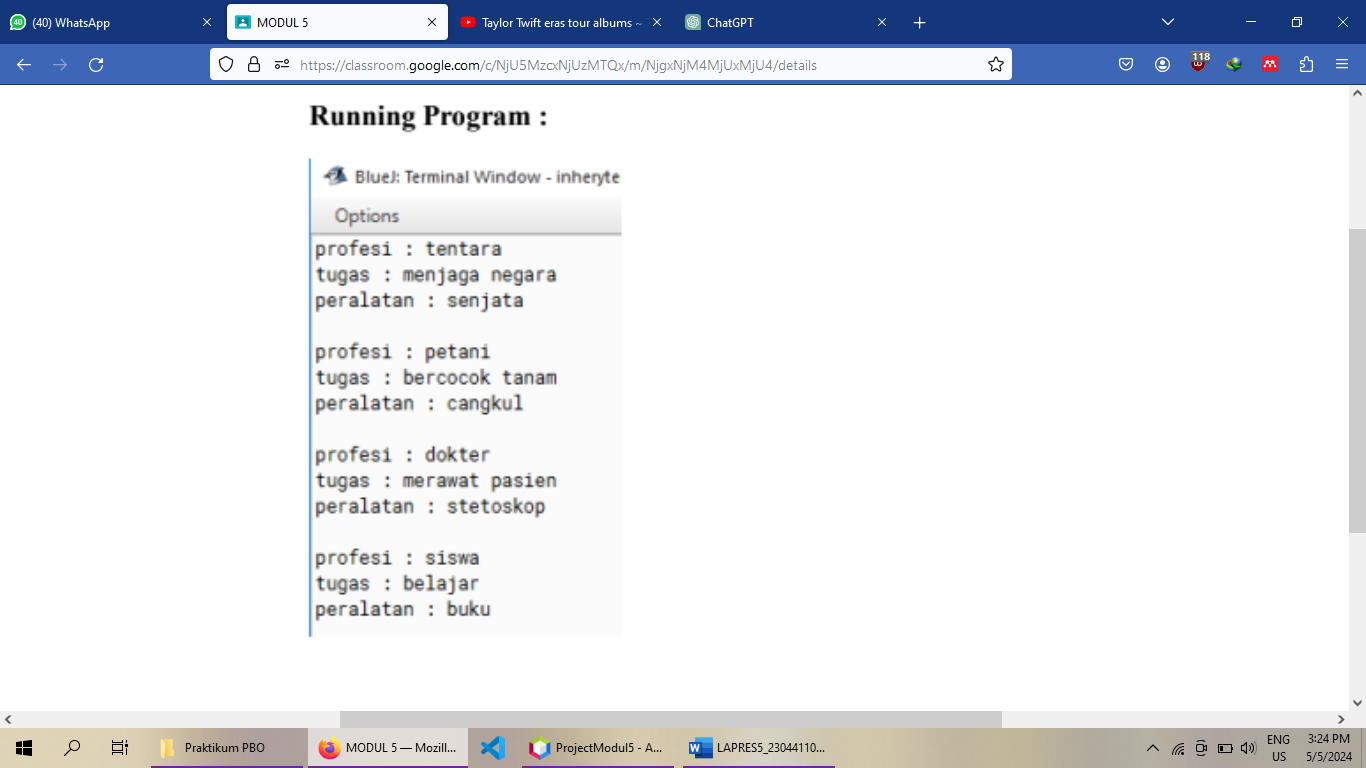
Class run :





Class ini digunakan untuk menjalankan program awalnya membuat array dengan jumlah subclass yang akan ditampilkan. Selanjutnya pembuatan object pada array yang menggunakan metode Polimorfisme Runtime. Kemudian melakukan perulangan sebanyak jumlah array dari manusia, didalam array itu kita panggil semua method yang ada pada superclass manusia.

## Running program

****

Dapat menampilkan semua anggota subclass dari class manusia dengan kaidah polimorfisme.

# **BAB IV IMPLEMENTASI**

## Soal

Buatlah sebuah program Java yang dapat mengelola data diri karyawan dalam sebuah perusahaan. Program ini harus menggunakan konsep "abstract class" dan "abstract method" untuk mendefinisikan atribut dan method umum yang dimiliki oleh semua jenis karyawan. Gunakan juga konsep polymorphism untuk memproses data karyawan dari berbagai jenis (misalnya, Karyawan Tetap dan Karyawan Kontrak) secara dinamis berdasarkan input dari pengguna. Simpan data karyawan dalam array.

* Spesifikasi

1. Abstract Class Karyawan :

* Atribut: nama, usia, dan posisi.
* Method abstrak: tampilkanData().

1. Class KaryawanTetap dan KaryawanKontrak yang Mengextends Abstract Class Karyawan :

* Implementasi method tampilkanData() untuk menampilkan data diri karyawan dengan format yang sesuai.
* Class KaryawanTetap memiliki tambahan atribut gajiBulanan.
* Class KaryawanKontrak memiliki tambahan atribut upahPerJam dan jumlahJamKerja.

1. Main Class :

* Meminta pengguna untuk memasukkan data diri karyawan (nama, usia, posisi, jenis karyawan, dan data tambahan sesuai jenis karyawan).
* Menyimpan data karyawan dalam sebuah array.
* Menampilkan semua data karyawan yang telah dimasukkan.

## Source Code

**Package Karyawan**

package com.mycompany.modul45;

abstract class Karyawan {

String nama;

int usia;

String posisi;

public Karyawan(String nama, int usia, String posisi){

this.nama = nama;

this.usia = usia;

this.posisi = posisi;

}

abstract void tampilkanData();

}

**Package KaryawanKontrak**

package com.mycompany.modul45;

public class KaryawanKontrak extends Karyawan {

double upahPerJam;

int jumlahJamKerja;

public KaryawanKontrak(String nama, int usia, String posisi, double upahPerJam, int jumlahJamKerja) {

super(nama, usia, posisi);

this.upahPerJam = upahPerJam;

this.jumlahJamKerja = jumlahJamKerja;

}

@Override

void tampilkanData() {

System.out.println("===== Karyawan Kontrak =====");

System.out.println("Nama : " + nama);

System.out.println("Usia : " + usia);

System.out.println("Posisi : " + posisi);

System.out.println("Upah Per Jam : " + upahPerJam);

System.out.println("Jumlah Jam Kerja : " + jumlahJamKerja);

}

}

**Package KaryawanTetap**

package com.mycompany.modul45;

public class KaryawanTetap extends Karyawan {

double gajibulanan;

public KaryawanTetap(String nama, int usia, String posisi, double gajibulanan){

super(nama, usia, posisi);

this.gajibulanan = gajibulanan;

}

void tampilkanData(){

System.out.println("===== Karyawan Tetap =====");

System.out.println("Nama : " + nama);

System.out.println("Usia : " + usia);

System.out.println("Posisi : " + posisi);

System.out.println("Gaji : " + gajibulanan);

}

}

**Package modul45**

package com.mycompany.modul45;

import java.util.ArrayList;

import java.util.Scanner;

public class Modul45 {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

ArrayList<Karyawan> karyawanList = new ArrayList<>();

while (true) {

System.out.println("===== Masukkan Data Karyawan =====");

System.out.print("Nama Karyawan : ");

String nama = scanner.nextLine();

System.out.print("Usia Karyawan : ");

int usia = scanner.nextInt();

scanner.nextLine(); // Consume newline

System.out.print("Posisi Karyawan : ");

String posisi = scanner.nextLine();

System.out.print("Jenis Karyawan (1. Tetap, 2. Kontrak): ");

int jenisKaryawan = scanner.nextInt();

if (jenisKaryawan == 1) {

System.out.print("Gaji Bulanan : ");

double gajiBulanan = scanner.nextDouble();

scanner.nextLine(); // Consume newline

Karyawan karyawan = new KaryawanTetap(nama, usia, posisi, gajiBulanan);

karyawanList.add(karyawan);

} else if (jenisKaryawan == 2) {

System.out.print("Upah per Jam : ");

double upahPerJam = scanner.nextDouble();

System.out.print("Jumlah Jam Kerja : ");

int jumlahJamKerja = scanner.nextInt();

scanner.nextLine(); // Consume newline

Karyawan karyawan = new KaryawanKontrak(nama, usia, posisi, upahPerJam, jumlahJamKerja);

karyawanList.add(karyawan);

} else {

System.out.println("Jenis karyawan tidak valid!");

}

System.out.print("Apakah Anda ingin menambahkan karyawan lagi? (y/n): ");

String lagi = scanner.nextLine();

if (!lagi.equalsIgnoreCase("y")) {

break;

}

}

System.out.println("\nData Karyawan :");

for (Karyawan karyawan : karyawanList) {

karyawan.tampilkanData();

System.out.println();

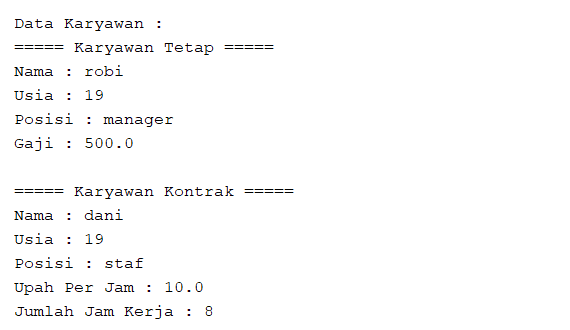
}

scanner.close();

}

}

## Hasil



## Penjelasan

Program di atas merupakan suatu program Java untuk menginputkan data karyawan dari suatu perusahaan. Pada program tersebut terdapat class Karyawan yang merupakan abstract class dan bertugas untuk mewariskan atribut serta method abstrak yang ia punya terhadap subclass-nya atau child class-nya.

Kemudian ada juga class KaryawanTetap dan KaryawanKontrak, kelas-kelas ini merepresentasikan child class. Kelas KaryawanTetap dan KaryawanKontrak mewarisi atribut dari kelas Karyawan dan mengimplementasikan method abstrak tampilkanData(). Selain itu, juga ada class Main yang merupakan tempat eksekusi utama dari program ini. Pada class ini juga terdapat ArrayList untuk menyimpan hasil inputan data karyawan dari user.

Ketika program berjalan, user akan diminta untuk mengisi data karyawan seperti nama, usia, posisi, dan jenis karyawan (tetap atau kontrak). Jika karyawan adalah karyawan tetap, user juga akan diminta mengisi gaji bulanan. Jika karyawan adalah karyawan kontrak, user akan diminta mengisi upah per jam dan jumlah jam kerja. Data karyawan tersebut kemudian disimpan dalam ArrayList. Setelah semua data karyawan dimasukkan, program akan menampilkan semua data karyawan yang telah diinputkan oleh user.

# **BAB V PENUTUP**

## Analisa

Dari hasil praktikum, praktikan dapat menganalisa bahwa Polimorfisme adalah salah satu konsep kunci dalam pemrograman berorientasi objek yang memungkinkan objek untuk menunjukkan perilaku yang berbeda tergantung pada konteks pemanggilannya. Dalam konteks program yang diberikan, polimorfisme termanifestasi melalui penggunaan kelas abstrak "Karyawan" dan kelas-kelas turunannya seperti "KaryawanTetap", "KaryawanKontrak".

Polimorfisme memungkinkan untuk penanganan hierarki kelas dengan cara yang lebih efisien dan fleksibel. Dalam program tersebut, penambahan jenis karyawan baru hanya memerlukan penambahan kelas turunan baru dari kelas abstrak "Karyawan" dengan implementasi metode "tampilkanData()" yang sesuai. Hal ini menunjukkan bagaimana polimorfisme memfasilitasi pemeliharaan dan perluasan kode dengan cara yang modular dan elegan.Top of Form Untuk mengoptimalkan polimorfisme, penting untuk memperhatikan struktur hierarki kelas dan desain abstraksi dengan cermat. Terlalu banyak kelas turunan atau terlalu dalamnya hierarki kelas dapat menyebabkan kompleksitas yang berlebihan.

## Kesimpulan

1. Polimorfime adalah suatu konsep dasar java. Polimorfime berasal dari dua kata yaitu poli dan morphs, poli berarti banyak morphs berarti bentuk.
2. Polimorfisme juga dapat diartikan sebagai metode dengan penamaan yang sama, namun memiliki respon yang berbeda-beda.
3. Implementasi dari polimorfime sendiri adalah Override, Overloading, dan Polimorfisme Runtime.
4. Dalam mengoptimalkan polimorfisme, penting untuk memperhatikan struktur hierarki kelas dan desain abstraksi dengan cermat
5. **Overloading adalah** metode dengan nama yang sama memiliki parameter yang berbeda, memungkinkan pemanggilan yang berbeda dengan perilaku yang berbeda.