Nama : Arditya Adjie Rosandi

NIM : 20230801274

Pemrogaman Berorientasi Objek

1. Polymorfisme Dinamis

=> Abstract

```java
package Pertemuan10.PolymorfismeDinamis.Abstract;

public class Pegawai {
    private String NIP;
    private String nama;

    // Constructor untuk inisialisasi nama dan NIP
    public Pegawai(String nama, String NIP) {
        this.nama = nama;
        this.NIP = NIP;
    }

    // Getter untuk nama
    public String getNama() {
        return nama;
    }

    // Getter untuk NIP
    public String getNIP() {
        return NIP;
    }

    // Method untuk mengirim email
    public void kirimEmail(String to, String subjek, String isi) {
        System.out.println(getNama() + " Kirim email ke : " + to);
        System.out.println("Dengan Subjek : " + subjek);
        System.out.println("Dengan Isi : " + isi);
    }
}
```

```java
package Pertemuan10.PolymorfismeDinamis.Abstract;

public class Staff extends Pegawai {
    private String bagian;
    public Staff(String nama, String NIP, String bagian){
        super(nama,NIP);
        setBagian(bagian);
    }

    public void setBagian(String namabagian){
        bagian=namabagian;
    }

    public String getBagian(){
        return bagian;
    }
}
```

```java
package Pertemuan10.PolymorfismeDinamis.Abstract;

public class StaffTester {
    Run | Debug
    public static void main(String[] args) {
        Staff s = new Staff(nama:"Januar", NIP:"1234", bagian:"Keuangan");
        s.kirimEmail(to:"a@test.com", subjek:"info test", isi:"isi email");
        System.out.println("NIP : " + s.getNIP() + "\n" + "Nama : " +
        s.getNama() + "\n" + "Bagian : " + s.getBagian());
    }
}
```

Struktur Program

Program ini terdiri dari **tiga kelas**:

1. **Kelas Pegawai** (Superclass)
2. **Kelas Staff** (Subclass, turunan dari Pegawai)
3. **Kelas StaffTester** (Kelas utama untuk menjalankan program)

1. Kelas Pegawai

Kelas Pegawai berperan sebagai **kelas induk** yang menyimpan data umum dari pegawai.

Atribut

- private String NIP → Menyimpan Nomor Induk Pegawai.
- private String nama → Menyimpan nama pegawai.

Constructor

public Pegawai(String nama, String NIP)

- **Tujuan**: Menginisialisasi atribut nama dan NIP saat objek Pegawai dibuat.

Metode

1. **public String getNama()**
    - Mengembalikan nilai nama.
2. **public String getNIP()**
    - Mengembalikan nilai NIP.
3. **public void kirimEmail(String to, String subjek, String isi)**
    - Menampilkan simulasi pengiriman email ke layar konsol dengan format:
    - [Nama Pegawai] Kirim email ke : [tujuan email]
    - Dengan Subjek : [subjek email]
    - Dengan Isi : [isi email]

2. Kelas Staff

Kelas Staff adalah **subkelas** dari kelas Pegawai. Kelas ini menambahkan atribut **bagian** yang merepresentasikan bagian/departemen tempat staff bekerja.

Atribut

- private String bagian → Menyimpan informasi bagian/departemen tempat bekerja.

Constructor

public Staff(String nama, String NIP, String bagian)

- **Tujuan**: Menginisialisasi atribut nama, NIP, dan bagian.
- **Implementasi**:
    - Memanggil constructor superclass (super(nama, NIP)) untuk mengatur nama dan NIP.
    - Mengatur atribut bagian melalui metode setBagian().

Metode

1. **public void setBagian(String namabagian)**
    - **Tujuan**: Mengatur nilai atribut bagian.
2. **public String getBagian()**
    - **Tujuan**: Mengembalikan nilai atribut bagian.

3. Kelas StaffTester

Kelas StaffTester digunakan untuk **menguji** fungsionalitas kelas Staff. Program ini membuat objek Staff dan memanggil metode yang dimiliki oleh kelas Pegawai dan Staff.

Metode main

public static void main(String[] args)

- **Tujuan**: Metode utama yang menjalankan program.

Langkah-langkah Implementasi

1. **Membuat Objek Staff**

   Staff s = new Staff("Januar", "1234", "Keuangan");

- **Penjelasan**:
    - Membuat objek Staff dengan parameter:
        - nama: "Januar"
        - NIP: "1234"
        - bagian: "Keuangan"

- Constructor Staff memanggil constructor dari superclass Pegawai untuk mengatur atribut nama dan NIP.

2. **Memanggil Metode kirimEmail()**

s.kirimEmail("a@test.com", "info test", "isi email");

- **Penjelasan**:
  - Memanggil metode kirimEmail() yang diwarisi dari kelas Pegawai.
  - Menampilkan simulasi pengiriman email ke layar konsol.
  - **Output**:
  - Januar Kirim email ke : a@test.com
  - Dengan Subjek : info test
  - Dengan Isi : isi email

3. **Menampilkan Informasi Staff**

```
System.out.println("NIP : " + s.getNIP() + "\n" +
    "Nama : " + s.getNama() + "\n" +
    "Bagian : " + s.getBagian());
```

- **Penjelasan**:
  - Memanggil metode getNIP(), getNama(), dan getBagian() untuk mengambil nilai atribut dari objek Staff.
  - Menampilkan informasi ke layar konsol.
  - **Output**:
  - NIP : 1234
  - Nama : Januar
  - Bagian : Keuangan

Kesimpulan Program

1. Kelas Pegawai sebagai superclass menyimpan atribut dan metode umum yang dimiliki oleh semua pegawai.
2. Kelas Staff memperluas fungsionalitas Pegawai dengan menambahkan atribut khusus **bagian**.
3. Kelas StaffTester digunakan untuk membuat objek Staff, menguji metode pewarisan, dan menampilkan hasilnya.

Output Lengkap Program:

Januar Kirim email ke : a@test.com

Dengan Subjek : info test

Dengan Isi : isi email

NIP : 1234

Nama : Januar

Bagian : Keuangan

=> Final

```java
package Pertemuan10.PolymorfismeDinamis.Final;

public final class MyMath {
    // Atribut
    public final double pi = 3.1416;

    // Method
    public final double luasLingkaran(double r) {
        return pi * r * r;
    }

    public final double kelilingLingkaran(double r) {
        return 2 * pi * r;
    }

    public final double sin(double derajat){
        return Math.sin(Math.toRadians(derajat));
    }

    public final double cos(double derajat){
        return Math.cos(Math.toRadians(derajat));
    }

    public final double tan(double derajat){
        return Math.tan(Math.toRadians(derajat));
    }

    public final double pangkat(double a, double b){
        return Math.pow(a, b);
    }
}
```

```java
package Pertemuan10.PolymorfismeDinamis.Final;

public class Main
{
    Run | Debug
    public static void main( String[] args )
    {
        // Instance object
        MyMath mm = new MyMath();
        System.out.println("Luas Lingkaran = " + mm.luasLingkaran(r:5));
        System.out.println("Keliling Lingkaran = " + mm.kelilingLingkaran(r:5));
        System.out.println("Sin 30 derajat = " + mm.sin(derajat:30));
        System.out.println("Cos 30 derajat = " + mm.cos(derajat:30));
        System.out.println("Tan 30 derajat = " + mm.tan(derajat:30));
        System.out.println("2^8 = " + mm.pangkat(a:2, b:8));
    }
}
```

Struktur Program

Program ini terdiri dari **dua kelas**:

1. **Kelas MyMath**
2. **Kelas Main**

1. Kelas MyMath

Kelas **MyMath** adalah kelas yang bersifat **final**, yang artinya kelas ini **tidak dapat diturunkan** (diwarisi) oleh kelas lain.

Atribut

1. **public final double pi**
    - **Tujuan**: Menyimpan nilai konstanta π (pi).
    - **Tipe**: double
    - **Aksesibilitas**: public (dapat diakses di mana saja).
    - **Final**: Nilainya **tidak dapat diubah** setelah diinisialisasi.

Metode

Semua metode dalam kelas MyMath diberi modifier **final**. Artinya, metode ini **tidak dapat dioverride** (ditimpa) di kelas turunannya (jika memungkinkan).

1. **public final double luasLingkaran(double r)**
    - **Tujuan**: Menghitung luas lingkaran.
    - **Parameter**:
        - double r: Jari-jari lingkaran.
    - **Rumus**: π×r×r\pi \times r \times r
    - **Return**: Nilai luas lingkaran bertipe double.

2. **public final double kelilingLingkaran(double r)**
    - **Tujuan**: Menghitung keliling lingkaran.
    - **Parameter**:
        - double r: Jari-jari lingkaran.
    - **Rumus**: 2×π×r2 \times \pi \times r
    - **Return**: Nilai keliling lingkaran bertipe double.
3. **public final double sin(double derajat)**
    - **Tujuan**: Menghitung nilai sinus dari sudut tertentu.
    - **Parameter**:
        - double derajat: Sudut dalam satuan derajat.
    - **Implementasi**:
        - Konversi derajat ke radian menggunakan Math.toRadians(derajat).
        - Hitung sinus menggunakan Math.sin().
    - **Return**: Nilai sinus bertipe double.
4. **public final double cos(double derajat)**
    - **Tujuan**: Menghitung nilai cosinus dari sudut tertentu.
    - **Parameter**:
        - double derajat: Sudut dalam satuan derajat.
    - **Return**: Nilai cosinus bertipe double.
5. **public final double tan(double derajat)**
    - **Tujuan**: Menghitung nilai tangen dari sudut tertentu.
    - **Parameter**:
        - double derajat: Sudut dalam satuan derajat.
    - **Return**: Nilai tangen bertipe double.
6. **public final double pangkat(double a, double b)**
    - **Tujuan**: Menghitung nilai pangkat.
    - **Parameter**:
        - double a: Bilangan dasar (base).
        - double b: Pangkat (eksponen).
    - **Implementasi**: Menggunakan metode Math.pow(a, b).
    - **Return**: Hasil perpangkatan bertipe double.

2. Kelas Main

Kelas Main berisi metode main untuk menguji fungsionalitas kelas **MyMath**.

Metode main

public static void main(String[] args)

- **Tujuan**: Metode utama yang digunakan untuk menjalankan program.
- **Implementasi**:
    1. Membuat objek MyMath bernama mm.
    2. Memanggil metode-metode dalam kelas MyMath untuk melakukan perhitungan.
    3. Menampilkan hasil perhitungan ke layar konsol.

Output Lengkap Program

Luas Lingkaran = 78.54

Keliling Lingkaran = 31.416

Sin 30 derajat = 0.5

Cos 30 derajat = 0.8660254037844386

Tan 30 derajat = 0.5773502691896257

2^8 = 256.0

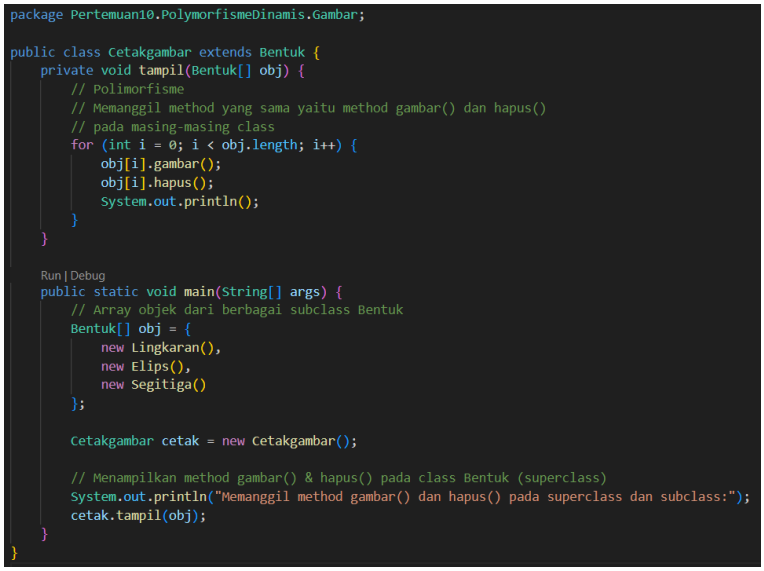
=> Gambar

```java
package Pertemuan10.PolymorfismeDinamis.Gambar;

public class Bentuk {
    protected void gambar() {
        System.out.println("superclass -> Menggambar");
    }

    protected void hapus() {
        System.out.println("superclass -> Menghapus Gambar");
    }
}
```

```java
package Pertemuan10.PolymorfismeDinamis.Gambar;

public class Elips extends Bentuk {
    protected void gambar() {
        System.out.println("subclass -> Menggambar Elips");
    }

    protected void hapus() {
        System.out.println("subclass -> Menghapus Gambar Elips");
    }
}
```

```java
package Pertemuan10.PolymorfismeDinamis.Gambar;

public class Lingkaran extends Bentuk {
    protected void gambar() {
        System.out.println("subclass -> Menggambar Lingkaran");
    }

    protected void hapus() {
        System.out.println("subclass -> Menghapus Gambar Lingkaran");
    }
}
```

```java
package Pertemuan10.PolymorfismeDinamis.Gambar;

public class Segitiga extends Bentuk {
    protected void gambar() {
        System.out.println("subclass -> Menggambar Segitiga");
    }

    protected void hapus() {
        System.out.println("subclass -> Menghapus Gambar Segitiga");
    }
}
```



```java
package Pertemuan10.PolymorfismeDinamis.Gambar;

public class Cetakgambar extends Bentuk {
    private void tampil(Bentuk[] obj) {
        // Polimorfisme
        // Memanggil method yang sama yaitu method gambar() dan hapus()
        // pada masing-masing class
        for (int i = 0; i < obj.length; i++) {
            obj[i].gambar();
            obj[i].hapus();
            System.out.println();
        }
    }

    Run | Debug
    public static void main(String[] args) {
        // Array objek dari berbagai subclass Bentuk
        Bentuk[] obj = {
            new Lingkaran(),
            new Elips(),
            new Segitiga()
        };

        Cetakgambar cetak = new Cetakgambar();

        // Menampilkan method gambar() & hapus() pada class Bentuk (superclass)
        System.out.println("Memanggil method gambar() dan hapus() pada superclass dan subclass:");
        cetak.tampil(obj);
    }
}
```

1. Kelas Bentuk (Superclass)

Kelas ini mendefinisikan dua metode:

- **gambar()**: Menampilkan pesan default yang menyatakan "superclass -> Menggambar".
- **hapus()**: Menampilkan pesan default yang menyatakan "superclass -> Menghapus Gambar".

Metode-metode ini nantinya akan di-*override* oleh kelas-kelas yang mewarisi kelas Bentuk.

2. Kelas Lingkaran, Elips, dan Segitiga (Subclass)

Ketiga kelas ini adalah **subclass** dari kelas Bentuk, dan mereka masing-masing meng-override metode gambar() dan hapus() untuk memberikan implementasi yang lebih spesifik sesuai dengan jenis objek mereka.

- **Lingkaran** mengimplementasikan metode gambar() untuk menampilkan pesan "Menggambar Lingkaran" dan hapus() untuk menampilkan pesan "Menghapus Gambar Lingkaran".
- **Elips** mengimplementasikan metode gambar() untuk menampilkan pesan "Menggambar Elips" dan hapus() untuk menampilkan pesan "Menghapus Gambar Elips".
- **Segitiga** mengimplementasikan metode gambar() untuk menampilkan pesan "Menggambar Segitiga" dan hapus() untuk menampilkan pesan "Menghapus Gambar Segitiga".

3. Kelas Cetakgambar (Main Program)

- Di kelas Cetakgambar, terdapat metode **tampil()** yang menerima array objek bertipe Bentuk. Dalam metode ini, program melakukan iterasi untuk memanggil metode gambar() dan hapus() pada setiap objek dalam array.
    - Meskipun referensi yang digunakan adalah Bentuk[], yang dipanggil adalah implementasi spesifik dari metode gambar() dan hapus() sesuai dengan jenis objek yang ada (misalnya, Lingkaran, Elips, atau Segitiga). Ini adalah contoh dari

Polimorfisme Dinamis, di mana metode yang sama (gambar() dan hapus()) akan berperilaku berbeda tergantung pada jenis objek yang memanggilnya.

- **Array Objek Bentuk[]**:
    - Dalam program, dibuat array objek yang berisi objek-objek dari kelas-kelas Lingkaran, Elips, dan Segitiga. Ketiga objek tersebut semuanya memiliki tipe referensi Bentuk[], tetapi masing-masing objek ini akan menjalankan implementasi metode yang sesuai dari kelas masing-masing.
- **Pemanggilan Metode**:
    - Metode **gambar()** dan **hapus()** dipanggil pada setiap objek dalam array, dan meskipun objek-objek tersebut bertipe Bentuk, mereka menjalankan metode yang di-*override* sesuai dengan jenis objek spesifiknya. Ini menunjukkan **Polimorfisme Dinamis** dalam aksi.

4. Output Program

Saat program dijalankan, output yang dihasilkan adalah:

Memanggil method gambar() dan hapus() pada superclass dan subclass:

subclass -> Menggambar Lingkaran

subclass -> Menghapus Gambar Lingkaran

subclass -> Menggambar Elips

subclass -> Menghapus Gambar Elips

subclass -> Menggambar Segitiga

subclass -> Menghapus Gambar Segitiga

=> Interface

```
package Pertemuan10.PolymorfismeDinamis.Interface;

public interface Bidang2D {
    double getKeliling();
    double getLuas();
}
```

```java
package Pertemuan10.PolymorfismeDinamis.Interface;

public class BujurSangkar implements Bidang2D {
    public double sisi;

    public double getKeliling() {
        return 4*sisi;
    }

    public double getLuas() {
        return sisi*sisi;
    }
}
```

```java
package Pertemuan10.PolymorfismeDinamis.Interface;

public class Lingkaran implements Bidang2D {
    public double radius;

    public double getKeliling() {
        return 2*Math.PI*radius;
    }

    public double getLuas() {
        return Math.PI*radius*radius;
    }
}
```

```java
package Pertemuan10.PolymorfismeDinamis.Interface;

public class PersegiPanjang implements Bidang2D {
    public double panjang;
    public double lebar;

    public double getKeliling() {
        return 2*(panjang + lebar);
    }

    public double getLuas() {
        return panjang*lebar;
    }
}
```

1. Interface Bidang2D

Interface Bidang2D mendeklarasikan dua metode:

- **getKeliling()**: Untuk menghitung keliling bidang 2D.

- **getLuas()**: Untuk menghitung luas bidang 2D.

Kedua metode ini tidak memiliki implementasi di dalam interface. Setiap kelas yang mengimplementasikan interface ini harus menyediakan implementasi spesifik untuk metode-metode tersebut.

2. Kelas BujurSangkar, Lingkaran, dan PersegiPanjang

Ketiga kelas ini mengimplementasikan interface Bidang2D, yang berarti mereka harus mengimplementasikan kedua metode yang dideklarasikan di dalam interface, yaitu getKeliling() dan getLuas().

- **BujurSangkar**:
    - **getKeliling()**: Menghitung keliling bujur sangkar dengan rumus 4 * sisi.
    - **getLuas()**: Menghitung luas bujur sangkar dengan rumus sisi * sisi.
- **Lingkaran**:
    - **getKeliling()**: Menghitung keliling lingkaran dengan rumus 2 * Math.PI * radius.
    - **getLuas()**: Menghitung luas lingkaran dengan rumus Math.PI * radius * radius.
- **PersegiPanjang**:
    - **getKeliling()**: Menghitung keliling persegi panjang dengan rumus 2 * (panjang + lebar).
    - **getLuas()**: Menghitung luas persegi panjang dengan rumus panjang * lebar.

3. Polimorfisme Dinamis

Meskipun ketiga kelas ini memiliki tipe yang berbeda (BujurSangkar, Lingkaran, PersegiPanjang), mereka semua mengimplementasikan interface yang sama, yaitu Bidang2D. Ini memungkinkan kita untuk menggunakan **Polimorfisme Dinamis**, di mana objek-objek dari kelas-kelas ini bisa diperlakukan sebagai objek bertipe Bidang2D.

Hal ini memungkinkan kita untuk memanggil metode getKeliling() dan getLuas() pada objek bertipe Bidang2D, tanpa peduli objek spesifik yang digunakan. Yang dieksekusi adalah implementasi metode sesuai dengan jenis objek yang ada (misalnya, Lingkaran, BujurSangkar, atau PersegiPanjang).

```java
package Pertemuan10.PolymorfismeDinamis.Static;

public final class MyMath {
    // Atribut
    public static final double pi = 3.14;

    // Method
    public static double kelilingLingkaran(double r) {
        return 2 * pi * r;
    }

    public static double luasLingkaran(double r) {
        return pi * r * r;
    }

    public static double sin(double derajat){
        return Math.sin(Math.toRadians(derajat));
    }

    public static double cos(double derajat){
        return Math.cos(Math.toRadians(derajat));
    }

    public static double tan(double derajat){
        return Math.tan(Math.toRadians(derajat));
    }

    public static double pangkat(double bilangan, int pangkat){
        return Math.pow(bilangan, pangkat);
    }

    public static double getPi() {
        return pi;
    }
}
```

```java
package Pertemuan10.PolymorfismeDinamis.Static;

public class Main
{
    Run | Debug
    public static void main( String[] args )
    {
        // Tidak perlu instansiasi objek
        System.out.println("Besar PI adalah " + MyMath.pi);
        System.out.println("Keliling lingkaran dengan jari-jari 5 adalah " + MyMath.kelilingLingkaran(r:5));
        System.out.println("Luas lingkaran dengan jari-jari 5 adalah " + MyMath.luasLingkaran(r:5));
        System.out.println("Sinus 30 derajat adalah " + MyMath.sin(derajat:30));
        System.out.println("Cosinus 30 derajat adalah " + MyMath.cos(derajat:30));
        System.out.println("Tangens 30 derajat adalah " + MyMath.tan(derajat:30));
        System.out.println("5 pangkat 3 adalah " + MyMath.pangkat(bilangan:5, pangkat:3));
    }
}
```

1. Kelas MyMath

Kelas MyMath berisi beberapa metode statis dan satu atribut statis yang digunakan untuk melakukan perhitungan matematis seperti menghitung keliling dan luas lingkaran, menghitung nilai trigonometri, dan menghitung pangkat.

- **Atribut Statis pi**:
    - Atribut ini menyimpan nilai konstanta π (pi), yang digunakan dalam perhitungan keliling dan luas lingkaran.
    - Atribut ini adalah static, yang berarti nilainya dapat diakses langsung menggunakan nama kelas (MyMath.pi), tanpa perlu membuat objek dari kelas MyMath.

- **Metode Statis**:
    - Semua metode dalam kelas MyMath didefinisikan sebagai metode statis (menggunakan kata kunci static).
    - Metode ini dapat diakses langsung menggunakan nama kelas tanpa membuat objek dari kelas MyMath. Beberapa metode yang ada antara lain:
        - **kelilingLingkaran(double r)**: Menghitung keliling lingkaran berdasarkan jari-jari r dengan rumus 2 * pi * r.
        - **luasLingkaran(double r)**: Menghitung luas lingkaran berdasarkan jari-jari r dengan rumus pi * r * r.
        - **sin(double derajat)**: Menghitung sinus dari sudut dalam derajat.
        - **cos(double derajat)**: Menghitung kosinus dari sudut dalam derajat.
        - **tan(double derajat)**: Menghitung tangen dari sudut dalam derajat.
        - **pangkat(double bilangan, int pangkat)**: Menghitung nilai bilangan yang dipangkatkan dengan pangkat yang diberikan.
        - **getPi()**: Mengembalikan nilai konstanta π (pi).

2. Kelas Main

Kelas Main adalah kelas yang menjalankan program. Di sini, semua metode statis dari kelas MyMath digunakan secara langsung melalui nama kelas tanpa perlu membuat objek dari MyMath.

- **Pemanggilan Metode Statis**:
    - Dalam Main, Anda dapat melihat bahwa semua metode dipanggil langsung menggunakan nama kelas MyMath, misalnya MyMath.kelilingLingkaran(5) untuk menghitung keliling lingkaran dengan jari-jari 5, atau MyMath.sin(30) untuk menghitung nilai sinus dari sudut 30 derajat.
    - Tidak ada kebutuhan untuk membuat objek MyMath karena metode-metode tersebut adalah statis.

3. Keuntungan Metode Statis

- **Tidak Perlu Membuat Objek**: Metode statis dapat diakses langsung melalui nama kelas tanpa perlu membuat instance (objek) dari kelas tersebut. Ini menghemat memori dan membuat kode lebih ringkas jika tidak membutuhkan status objek.
- **Penggunaan Bersama (Shared Usage)**: Atribut dan metode statis dimiliki bersama oleh semua objek dari kelas tersebut. Dalam hal ini, nilai pi digunakan di seluruh kelas tanpa perlu membuat objek MyMath.
- **Cocok untuk Fungsi yang Tidak Bergantung pada Status Objek**: Metode-metode dalam kelas MyMath adalah fungsi yang tidak bergantung pada status internal objek. Mereka hanya membutuhkan parameter yang diberikan untuk melakukan perhitungan.

4. Contoh Output Program

Saat program dijalankan, output yang dihasilkan adalah sebagai berikut:

Besar PI adalah 3.14

Keliling lingkaran dengan jari-jari 5 adalah 31.400000000000002

Luas lingkaran dengan jari-jari 5 adalah 78.5

Sinus 30 derajat adalah 0.49999999999999994

Cosinus 30 derajat adalah 0.8660254037844387

Tangens 30 derajat adalah 0.5773502691896257

5 pangkat 3 adalah 125.0

Output ini mencerminkan perhitungan-perhitungan yang dilakukan oleh metode-metode statis dalam kelas MyMath.

2. Polymorfisme Statis

```java
package Pertemuan10.PolymorfismeStatis;

public class Lingkaran {
    //Method menghitung luas dengan jari-jari
    float luas(float r){
        return (float) (Math.PI * r * r);
    }

    //Method menghitung luas dengan diameter
    double luas (double d){
        return (double) (1/4 *Math.PI *d);
    }
}
```

```java
package Pertemuan10.PolymorfismeStatis;

public class Overloading {
    public void Tampil() {
        System.out.println("I love Java");
    }

    public void Tampil(int i) {
        System.out.println("Method dengan 1 parameter = " + i);
    }

    public void Tampil(int i, int j) {
        System.out.println("Method dengan 2 parameter = " + i + " & " + j);
    }

    public void Tampil(String str) {
        System.out.println(str);
    }

    Run | Pertemuan10.PolymorfismeStatis.Overloading
    publ
        Overloading objek = new Overloading();
        objek.Tampil();
        objek.Tampil(i:8);
        objek.Tampil(i:6, j:7);
        objek.Tampil(str:"Hello world");
    }
}
```

```java
package Pertemuan10.PolymorfismeStatis;

class Polymorph {
    public int tambah(int x, int y) {
        return x+y;
    }
    public String tambah(String x, String y) {
        return x+ " " +y;
    }
}

public class PolymorphTester {
    Run | Debug
    public static void main(String[] args) {
        Polymorph p=new Polymorph();
        System.out.println("2 + 3 = " +p.tambah(x:2, y:3));
        System.out.println("\"2\" + \"3\" = " +p.tambah(x:"2", y:"3"));
    }
}
```

1. Kelas Lingkaran

Kelas Lingkaran memiliki dua metode dengan nama yang sama, yaitu **luas()**, tetapi dengan parameter yang berbeda:

- **luas(float r)**: Menghitung luas lingkaran berdasarkan jari-jari r menggunakan rumus π * r * r. Parameter r bertipe float.
- **luas(double d)**: Menghitung luas lingkaran berdasarkan diameter d menggunakan rumus (1/4) * π * d^2. Parameter d bertipe double.

Konsep yang diterapkan: Ini adalah contoh **Overloading**, di mana dua metode dengan nama yang sama digunakan untuk operasi yang serupa, tetapi dengan parameter yang berbeda (tipe parameter yang berbeda, yaitu float dan double).

2. Kelas Overloading

Kelas ini menunjukkan **Overloading** metode Tampil() yang memiliki empat variasi:

- **Tampil()**: Metode tanpa parameter, hanya mencetak "I love Java".
- **Tampil(int i)**: Metode yang menerima satu parameter bertipe int, mencetak nilai parameter tersebut.
- **Tampil(int i, int j)**: Metode yang menerima dua parameter bertipe int, mencetak keduanya.
- **Tampil(String str)**: Metode yang menerima satu parameter bertipe String, mencetak nilai parameter tersebut.

Konsep yang diterapkan: Ini adalah contoh **Overloading** dalam Java, di mana metode yang memiliki nama yang sama dapat memiliki parameter yang berbeda (baik jumlah maupun tipe parameter), memungkinkan pemanggilan metode yang berbeda bergantung pada parameter yang diberikan.

Penggunaan:

objek.Tampil(); // Memanggil metode tanpa parameter

objek.Tampil(8); // Memanggil metode dengan satu parameter (int)

objek.Tampil(6, 7); // Memanggil metode dengan dua parameter (int)

objek.Tampil("Hello world"); // Memanggil metode dengan satu parameter (String)

3. Kelas Polymorph

Kelas Polymorph memiliki dua metode dengan nama yang sama, yaitu **tambah()**, tetapi dengan tipe parameter yang berbeda:

- **tambah(int x, int y)**: Menambahkan dua bilangan bulat (int).
- **tambah(String x, String y)**: Menggabungkan dua string.

Konsep yang diterapkan: Ini adalah contoh lain dari **Overloading**. Meski nama metode sama, Java membedakan keduanya berdasarkan tipe parameter yang diterima (int vs. String).

Penggunaan:

Polymorph p = new Polymorph();

System.out.println("2 + 3 = " + p.tambah(2, 3)); // Memanggil metode dengan parameter int

System.out.println("\"2\" + \"3\" = " + p.tambah("2", "3")); // Memanggil metode dengan parameter String

Kesimpulan tentang Polimorfisme Statis dan Overloading

- **Polimorfisme Statis**: Dalam hal ini, polimorfisme diterapkan melalui **Overloading**, yaitu memiliki beberapa metode dengan nama yang sama tetapi dengan parameter yang berbeda. Pemilihan metode yang tepat dilakukan pada saat kompilasi berdasarkan tipe dan jumlah parameter.
- **Overloading** memungkinkan satu nama metode digunakan untuk berbagai tindakan yang terkait, tetapi berbeda dalam cara mereka diimplementasikan. Overloading meningkatkan keterbacaan dan fleksibilitas kode, karena Anda tidak perlu menggunakan nama metode yang berbeda untuk operasi yang serupa.