# MODUL PRAKTIKUM

**TI03209 – PEMROGRAMAN BERORIENTASI OBJEK**

Disusun oleh

Eko Mailansa, S.Kom., M.T.

**D3 – TEKNIK INFORMATIKA**

**POLITEKNIK HASNUR**

**BARTIO KUALA**

**TAHUN 2023**

# MODUL V
# *Polimorphism*

## V.1. Tujuan Praktikum

Praktikum yang dilakukan pada modul 5 bertujuan untuk:

1. Mengerti dan memahami konsep dari *Polimorfisme* dalam pemrograman Java
2. Mengerti definisi *Polimorfisme* serta penerapannya dalam pemrograman Java
3. Mengerti dan memahami keuntungan dan kekurangan mengimplementasikan pola *Polimorfisme* dalam pemrograman Java

## V.2. Indikator Pencapaian

Setelah melakukan praktikum maka, di harapkan praktikan mampu:

1. Membuat program sederhana yang mengimplementasikan pola *Polimorfisme*.
2. Membuat konstruktor dalam program sederhana yang di tulis dengan mengimplementasikan kata kunci **super**.

## V.3. Materi

Kata polimorfisme artinya mempunyai banyak bentuk yang berasal dari kata "***poly***" artinya banyak dan "***morphs***" artinya bentuk, jadi artinya banyak bentuk. Dengan kata sederhana, polimorfisme dapat di definisikan sebagai kemampuan suatu pesan untuk ditampilkan dalam lebih dari satu bentuk dan ini terjadi ketika kita memiliki banyak kelas yang dihubungkan satu sama lain melalui pewarisan.

### Ilustrasi Polimorfisme dalam Kehidupan Nyata

Seseorang pada saat yang sama dapat memiliki karakteristik yang berbeda-beda. Ibarat laki-laki sekaligus adalah ayah, suami dan karyawan. Jadi orang yang sama mempunyai perilaku yang berbeda dalam situasi yang berbeda. Ini disebut polimorfisme.

Polimorfisme dianggap sebagai salah satu fitur penting Pemrograman Berorientasi Objek. Polimorfisme memungkinkan kita melakukan suatu tindakan dengan cara yang berbeda. Dengan kata lain, polimorfisme memungkinkan dalam mendefinisikan satu *interface* dan memiliki banyak implementasi.

**Jenis polimorfisme**

Di Java polimorfisme dibagi menjadi dua jenis yaitu:

- Polimorfisme waktu kompilasi
- Polimorfisme Waktu Proses

**Polimorfisme Waktu Kompilasi**

Ia juga dikenal sebagai polimorfisme statis. Jenis polimorfisme ini dicapai dengan kelebihan fungsi atau kelebihan operator.

**Polimorfisme Waktu Proses**

Hal ini memungkinkan objek dari kelas turunan untuk berperilaku seolah-olah itu adalah objek dari kelas dasar atau kelas induk. Kelas turunan dapat mengambil alih fungsi virtual dari kelas induk untuk menyediakan implementasinya sendiri dan pemanggilan fungsi akan diselesaikan pada saat *runtime* bergantung pada tipe objek sebenarnya.

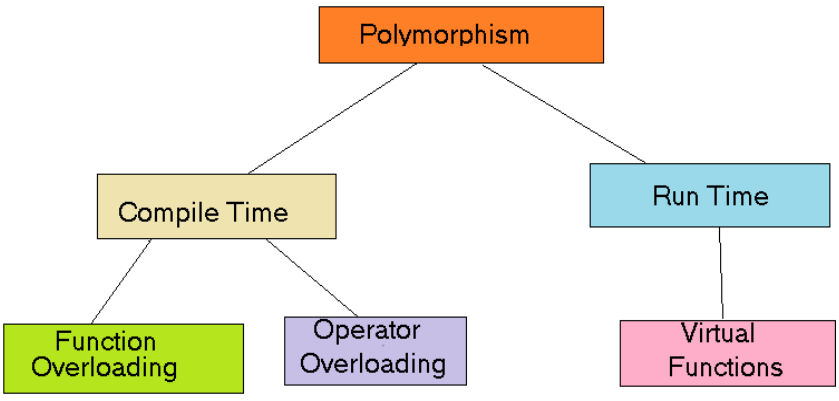


Diagram polimorfisme

Di Java, Polimorfisme adalah sebuah konsep yang memungkinkan objek dari kelas yang berbeda diperlakukan sebagai objek dari kelas yang sama. Ini memungkinkan objek berperilaku berbeda berdasarkan tipe kelas spesifiknya.

**Keuntungan Polimorfisme di Java**

- ✓ Meningkatkan penggunaan kembali kode dengan mengizinkan objek dari kelas yang berbeda diperlakukan sebagai objek dari kelas yang sama.
- ✓ Meningkatkan keterbacaan dan pemeliharaan kode dengan mengurangi jumlah kode yang perlu ditulis dan dipelihara.
- ✓ Mendukung pengikatan dinamis, memungkinkan metode yang benar dipanggil saat runtime, berdasarkan kelas objek sebenarnya.
- ✓ Memungkinkan objek diperlakukan sebagai tipe tunggal, membuatnya lebih mudah untuk menulis kode generik yang dapat menangani objek dengan tipe berbeda.

**Kekurangan Polimorfisme di Java**

- ✓ Dapat mempersulit pemahaman perilaku suatu objek, terutama jika kodenya rumit.
- ✓ Hal ini dapat menyebabkan masalah kinerja, karena perilaku polimorfik mungkin memerlukan komputasi tambahan saat runtime.

**Mengapa dan Kapan Menggunakan "Inheritance" dan "Polimorfisme"?**

Berguna untuk penggunaan kembali kode yang di tulis: menggunakan kembali atribut dan metode kelas yang ada saat membuat kelas baru.

## V.4. Alat dan Bahan

Peralatan dan bahan-bahan yang digunakan antara lain:

- ✓ Satu set Komputer
- ✓ Aplikasi IDE Netbeans 8.2
- ✓ Modul 4 – Mata kuliah Pemrograman Berorientasi Objek

## V.5. Praktikum

a. Class Mahasiswa

```
public class Mahasiswa {

    private final int nim;
    private final String nama;
    private String prodi;
    private final String mk;

    //Construstor with parameters
    public Mahasiswa(int nim, String nama, String prodi, String mk) {
        System.out.println("Nilai MataKuliah Mahasiswa");
        this.nim = nim;
        this.nama = nama;
        this.prodi = prodi;
        this.mk = mk;
    }

    public void mailCheck() {
        System.out.println("Mailing a check to " + this.nim + " " + this.prodi);
    }

    @Override
    public String toString() {
        return nim + " " + nama + " " + prodi + " " + mk;
    }

    //Getter and Setter Methods
    public int getNim() {
        return nim;
    }

    public String getNama() {
        return nama;
    }

    public String getProdi() {
        return prodi;
    }

    public void setProdi(String newProdi) {
        this.prodi = newProdi;
    }

    public String getMk() {
        return mk;
    }
}
```

b. Class Nilai

```
public final class Nilai extends Mahasiswa {

    private int nilai;

    public Nilai(int nim, String nama, String prodi, String mk, int nilai) {
        super(nim, nama, prodi, mk);
        setNilai(nilai);
    }

    @Override
    public void mailCheck() {
        System.out.println("MailCheck Kelas Nilai");
        System.out.print("Mailing check to " + getNim() + " Nilai MK PBO " + nilai);
        System.out.println("\n");
    }

    public int getNilai() {
        return nilai;
    }

    public void setNilai(int nilai) {
        this.nilai = nilai;
    }
}
```

c. Class NilaiDemo

```
public class NilaiDemo {

    public static void main(String[] args) {
        Nilai n = new Nilai(20202020, "Aninda", "Teknik Informatika", "ADPL", 62);
        Mahasiswa mhs = new Nilai(20202022, "Putri", "Teknik Informatika", "ADPL", 90);
        System.out.println("\nMemanggil melalui reference Nilai --");
        n.mailCheck();
        System.out.println("\nMemanggil melalui reference Mahasiswa --");
        mhs.mailCheck();
    }
}
```

## V.6. Referensi

### V.6.1. Utama


- ✓ Pecinovský, Rudolf, CSc. OOP – Learn Object Oriented Thinking and Programming. ISBN 978-80-904661-8-0 (paperback) & ISBN 978-80-904661-9-7 (PDF). Published in the Czech Republic by Tomáš Bruckner, Řepín – Živonín, Academic Series. 2013.
- ✓ Wiener, Richard (University of Colorado, Colorado Springs) & J. Pinson, Lewis (University of Colorado, Colorado Springs). Fundamentals of OOP and Data Structures in Java. ISBN 0 521 66220 6 hardback and eISBN 0-511-00168-1 virtual (netLibrary Edition). PUBLISHED BY CAMBRIDGE UNIVERSITY PRESS. 2000.


### V.6.2. Pendukung


- ✓ Object Oriented Programming in JAVA. Di akses pada 16 Agustus 2023. https:// www. tutorialspoint.com/object-oriented-programming-in-java/index.asp
- ✓ Object Oriented Programming (OOPs) Concept In Java - GreekforGreeks. Di akses pada 16 Agustus 2023. https:// www.google.com/amp/s/www.greekforgreeks.com/ object-oriented-programming-oops-concept-in-java/
- ✓ Object Oriented Thinking and Programming. Di akses pada 16 Agustus 2023. https://www.w3schools.in/java/java_oop.asp


## V.7. Tugas Tambahan

1. Tambahkan *statement* tanpa membuat *method* baru agar nilai yang ditampilkan menampilkan nilai huruf sesuai dengan aturan yang berlaku.

| NO. | Nilai Huruf | Nilai Angka | Bobot | Keterangan |
|---|---|---|---|---|
| 1 | A | 85 – 100 | 4 | LULUS |
| 2 | A- | 80 – 84,99 | 3.7 | LULUS |
| 3 | B+ | 75 – 79,99 | 3.3 | LULUS |
| 4 | B | 70 – 74,99 | 3 | LULUS |
| 5 | B- | 65 – 69,99 | 2.7 | LULUS |
| 6 | C+ | 60 – 64,99 | 2.3 | LULUS |
| 7 | C | 55 – 59,99 | 2 | LULUS |
| 8 | D | 40 – 54,99 | 1 | TIDAK LULUS |
| 9 | E | 0 – 39,99 | 0 | TIDAK LULUS |

Result:

```
Memanggil melalui reference Nilai --
MailCheck Kelas Nilai
Mailing check to 20202020 Nilai MK PBO 62 NH = C+


Memanggil melalui reference Mahasiswa --
MailCheck Kelas Nilai
Mailing check to 20202022 Nilai MK PBO 90, NH = A [LULUS]
```

2. Dari Class Nilai ubah *statement* pada *method* **setNilai** agar nilai yang ditampilkan hanya nilai mahasiswa yang lulus (>= 55), jika nilai mahasiswa kurang dari 55 tidak akan di tampilkan.
3. Dari Class NilaiDemo untuk nilai dari variable nim, nama, prodi, mk, nilai di inputkan oleh user/pengguna (harus memanggil method dengan memanggil melalui reference class Mahasiswa dan class Nilai)

Result:

```
run:
== Reference Nilai ==
NIM    : 20201234
Nama   : Anjal Perkasa
Prodi  : Teknik Informatika
MK     : Pemrograman Berorientasi Objek
Nilai  : 94

== Reference Mahasiswa ==
NIM    : 20202345
Nama   : Aris Pratama
Prodi  : Teknik Informatika
MK     : Analisis Dan Desain Perangkat Lunak
Nilai  : 58

Memanggil melalui reference Nilai --
MailCheck Kelas Nilai
Mailing check to 20201234 Nilai MK PBO 94, NH = A [LULUS]


Memanggil melalui reference Mahasiswa --
MailCheck Kelas Nilai
Mailing check to 20202345 Nilai MK PBO 58 NH = C [LULUS]
```

Input dari user

## V.8. Laporan

- ✓ Laporan "Praktikum" di tulis sesuai dengan contoh format yang telah di berikan.
- ✓ Laporan "Tugas Tambahan" di tulis sesuai dengan contoh format yang telah di berikan dan di lampirkan di setiap laporan "Praktikum" sesuai dengan Modul Praktikum.
- ✓ Isi laporan praktikum ➔ Praktikum dan Tugas Tambahan.
- ✓ *Rename* file laporan dengan format nama yang benar (nama_modul + nim + kelas) dan berikan ekstensi atau tipe file .pdf.
  (contoh: **Modul1_20202020_3B.pdf**)