Bab 2

Landasan Teori dan Konsep Dasar OOP

Sub-CPMK yang Dicakup dalam Bab Ini:

- Sub-CPMK 1.1: Menjelaskan perbedaan antara pemrograman prosedural dan berorientasi objek serta mengidentifikasi karakteristik utama paradigma OOP

2.1 Sejarah dan Evolusi Paradigma Pemrograman

Pemrograman komputer telah mengalami evolusi paradigma sejak komputer pertama kali diciptakan [1]. Paradigma pemrograman adalah cara fundamental dalam mengorganisasi dan menulis program komputer.

2.1.1 Timeline Paradigma Pemrograman

1. **1950-an**: Machine Language dan Assembly
2. **1960-an**: Pemrograman Prosedural (FORTRAN, COBOL)
3. **1970-an**: Pemrograman Terstruktur (C, Pascal)
4. **1980-an**: Pemrograman Berorientasi Objek (Smalltalk, C++)
5. **1990-an**: Popularisasi OOP (Java, Python)
6. **2000-an**: Multi-paradigm Languages

2.2 Paradigma Prosedural vs Berorientasi Objek

2.2.1 Pemrograman Prosedural

Pemrograman prosedural adalah paradigma yang berfokus pada **prosedur** atau **fungsi** yang beroperasi pada data.

Karakteristik:

- Program terdiri dari serangkaian instruksi/prosedur
- Data dan fungsi terpisah
- Alur program linear (top-down)
- Fokus pada "bagaimana" melakukan sesuatu

Contoh

Contoh Kode Prosedural (Pseudo-code):

```
// Data terpisah
struct Mahasiswa {
    string nim;
    string nama;
    double ipk;
}

// Fungsi terpisah
function hitungPredikat(double ipk) {
    if (ipk >= 3.5) return "Cum Laude";
    else if (ipk >= 3.0) return "Sangat Memuaskan";
    else return "Memuaskan";
}

// Penggunaan
Mahasiswa mhs;
mhs.nim = "1234";
mhs.nama = "Budi";
mhs.ipk = 3.7;
string predikat = hitungPredikat(mhs.ipk);
```

Perhatikan bahwa data (`Mahasiswa`) dan fungsi (`hitungPredikat`) terpisah.

2.2.2 Pemrograman Berorientasi Objek

Pemrograman Berorientasi Objek (OOP) adalah paradigma yang berfokus pada **objek** yang mengenkapsulasi data dan perilaku.

Karakteristik:

- Program terdiri dari objek-objek yang berinteraksi
- Data dan fungsi digabung dalam satu unit (class)
- Fokus pada "apa" yang dimodelkan
- Lebih dekat dengan dunia nyata

Contoh

Contoh Kode OOP (Java):

```java
public class Mahasiswa {
    // Data (attributes)
    private String nim;
    private String nama;
    private double ipk;

    // Constructor
    public Mahasiswa(String nim, String nama) {
        this.nim = nim;
        this.nama = nama;
        this.ipk = 0.0;
    }

    // Behavior (methods)
    public void setIPK(double ipk) {
        this.ipk = ipk;
    }

    public String getPredikat() {
        if (ipk >= 3.5) return "Cum Laude";
        else if (ipk >= 3.0) return "Sangat Memuaskan";
        else return "Memuaskan";
    }
}

// Penggunaan
Mahasiswa mhs = new Mahasiswa("1234", "Budi");
mhs.setIPK(3.7);
String predikat = mhs.getPredikat();
```

Kode Program 2.1: Class Mahasiswa dengan OOP

Data dan fungsi menyatu dalam class **`Mahasiswa`**.

2.2.3 Perbandingan Prosedural vs OOP

2.3 Konsep Dasar OOP

OOP dibangun di atas empat pilar utama:

Aspek	**Prosedural**	**OOP**
Fokus	Fungsi/Prosedur	Objek
Struktur Data	Data terpisah dari fungsi	Data dan fungsi dalam class
Akses Data	Global atau parameter	Encapsulation (private/public)
Code Reuse	Function library	Inheritance
Maintainability	Sulit untuk program besar	Lebih mudah dengan modularitas
Real-world Modeling	Abstrak	Lebih natural
Contoh Bahasa	C, Pascal, FORTRAN	Java, C++, Python [1, 2]

Tabel 2.1: Perbandingan Prosedural vs OOP

2.3.1 1. Abstraksi (Abstraction)

Konsep Penting

Abstraksi adalah proses menyembunyikan detail implementasi dan hanya menampilkan fungsionalitas kepada pengguna.
Contoh: Ketika menggunakan mobil, Anda hanya perlu tahu cara menggunakan setir, pedal gas, dan rem. Anda tidak perlu tahu detail mesin internal.

Dalam OOP, abstraksi dicapai melalui:

- Abstract classes
- Interfaces
- Encapsulation

2.3.2 2. Enkapsulasi (Encapsulation)

Konsep Penting

Enkapsulasi adalah pembungkusan data dan method yang beroperasi pada data tersebut dalam satu unit (class), serta menyembunyikan detail internal dari luar.

Manfaat enkapsulasi:

- **Data Hiding**: Melindungi data dari akses tidak sah
- **Modularity**: Kode lebih terorganisir
- **Flexibility**: Mudah mengubah implementasi internal
- **Maintainability**: Lebih mudah dipelihara

2.3.3 3. Pewarisan (Inheritance)

Konsep Penting

Inheritance adalah mekanisme di mana class baru (subclass) dapat mewarisi properties dan methods dari class yang sudah ada (superclass).

Manfaat inheritance:

- **Code Reusability**: Tidak perlu menulis ulang kode yang sama
- **Hierarchical Classification**: Organisasi class yang terstruktur
- **Extensibility**: Mudah menambah fitur baru

2.3.4 4. Polimorfisme (Polymorphism)

Konsep Penting

Polimorfisme adalah kemampuan objek untuk mengambil banyak bentuk. Dalam OOP, ini berarti satu interface dapat digunakan untuk tipe data yang berbeda.

Jenis polimorfisme:

- **Compile-time Polymorphism**: Method overloading
- **Runtime Polymorphism**: Method overriding

2.4 Keuntungan dan Tantangan OOP

2.4.1 Keuntungan OOP

1. **Modularitas**: Kode terorganisir dalam class-class yang independen
2. **Reusability**: Code reuse melalui inheritance dan composition
3. **Maintainability**: Lebih mudah menemukan dan memperbaiki bug
4. **Scalability**: Mudah menambah fitur baru tanpa mengubah kode existing
5. **Real-world Modeling**: Lebih natural dalam memodelkan dunia nyata
6. **Data Security**: Enkapsulasi melindungi data
7. **Collaboration**: Tim dapat bekerja pada class yang berbeda secara paralel

2.4.2 Tantangan OOP

1. **Learning Curve**: Membutuhkan pemahaman konsep yang lebih dalam
2. **Overhead**: Bisa lebih lambat untuk program sederhana
3. **Complexity**: Bisa menjadi terlalu kompleks jika tidak dirancang dengan baik
4. **Design Effort**: Membutuhkan perencanaan dan desain yang matang

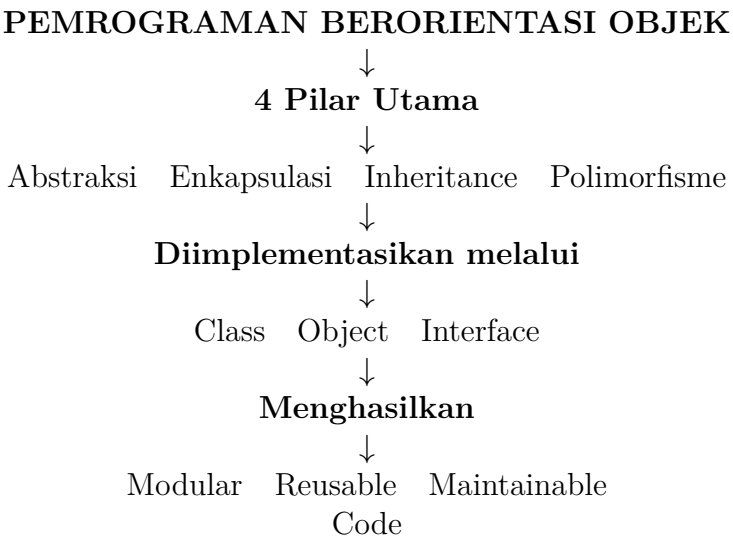
Catatan

OOP bukan solusi untuk semua masalah. Untuk program sederhana atau script kecil, pendekatan prosedural mungkin lebih efisien. Gunakan OOP ketika:

- Program cukup besar dan kompleks
- Membutuhkan code reuse yang tinggi
- Banyak developer bekerja pada proyek yang sama
- Sistem perlu mudah di-maintain dan di-extend

2.5 Peta Konsep OOP

Berikut adalah peta konsep yang menunjukkan hubungan antar konsep OOP:



Aktivitas Pembelajaran

1. **Diskusi Kelompok**: Identifikasi 5 contoh objek di dunia nyata (misalnya: mobil, HP, ATM) dan tentukan attributes dan behaviors yang bisa dimodelkan dalam OOP.

2. **Analisis Kode**: Diberikan kode prosedural untuk sistem perpustakaan sederhana, identifikasi bagian mana yang bisa diubah menjadi class dalam OOP.

3. **Perbandingan**: Tulis program sederhana (misalnya kalkulator) dalam dua versi: prosedural dan OOP. Bandingkan kelebihan dan kekurangan masing-masing.

4. **Mind Mapping**: Buat mind map yang menghubungkan 4 pilar OOP dengan contoh implementasi konkret dalam Java.

Latihan dan Refleksi

1. Jelaskan perbedaan utama antara pemrograman prosedural dan OOP! Berikan contoh kasus di mana OOP lebih cocok digunakan.

2. Sebutkan dan jelaskan 4 pilar OOP! Berikan contoh sederhana untuk masing-masing pilar.

3. Mengapa enkapsulasi penting dalam OOP? Apa yang terjadi jika semua data dalam class bersifat public?

4. Berikan 3 keuntungan dan 2 tantangan dalam menggunakan OOP.

5. Identifikasi objek-objek dalam sistem e-commerce (misalnya Tokopedia). Sebutkan minimal 5 class yang mungkin ada beserta attributes dan methods-nya.

6. Apakah semua program harus ditulis dengan OOP? Jelaskan kapan sebaiknya menggunakan OOP dan kapan tidak.

7. **Refleksi**: Bagaimana pemahaman Anda tentang OOP sebelum dan sesudah mempelajari bab ini? Konsep mana yang paling sulit dipahami?

Asesmen (Evaluasi Kinerja)

Instrumen Penilaian untuk Sub-CPMK 1.1

A. Pilihan Ganda

1. Manakah yang BUKAN merupakan pilar OOP?

   (a) Abstraksi
   (b) Enkapsulasi
   (c) Kompilasi
   (d) Polimorfisme

2. Dalam OOP, data dan method yang beroperasi pada data tersebut digabung dalam satu unit yang disebut:
   (a) Function
   (b) Module
   (c) Class
   (d) Library
3. Keuntungan utama enkapsulasi adalah:
   (a) Membuat program lebih cepat
   (b) Melindungi data dari akses tidak sah
   (c) Mengurangi ukuran file
   (d) Mempermudah kompilasi

B. Essay

1. Jelaskan dengan kata-kata Anda sendiri apa yang dimaksud dengan "objek" dalam OOP dan berikan 2 contoh objek dari dunia nyata beserta attributes dan behaviors-nya.
2. Bandingkan pendekatan prosedural dan OOP dalam menyelesaikan masalah pengelolaan data mahasiswa. Mana yang lebih cocok dan mengapa?

Rubrik Penilaian: Lihat Lampiran A

Checklist Pencapaian Kompetensi

Centang item berikut setelah Anda yakin telah menguasainya:

☐ Saya dapat menjelaskan perbedaan antara pemrograman prosedural dan OOP

☐ Saya dapat menyebutkan dan menjelaskan 4 pilar OOP

☐ Saya dapat memberikan contoh konkret untuk setiap pilar OOP

☐ Saya dapat mengidentifikasi kapan sebaiknya menggunakan OOP

☐ Saya memahami keuntungan dan tantangan dalam menggunakan OOP

☐ Saya dapat mengidentifikasi objek, attributes, dan methods dari dunia nyata

Rangkuman

Bab ini membahas landasan teori Pemrograman Berorientasi Objek, termasuk evolusi paradigma pemrograman, perbandingan antara pendekatan prosedural dan OOP, serta 4 pilar utama OOP.

Poin Kunci:

- OOP adalah paradigma yang berfokus pada objek yang mengenkapsulasi data dan perilaku
- 4 Pilar OOP: Abstraksi, Enkapsulasi, Inheritance, Polimorfisme
- OOP menawarkan modularitas, reusability, dan maintainability yang lebih baik
- OOP lebih cocok untuk sistem besar dan kompleks
- Pemahaman konsep dasar OOP adalah fondasi untuk mempelajari implementasi OOP dalam Java

Kata Kunci: ***OOP***, ***Class***, ***Object***, ***Abstraksi***, ***Enkapsulasi***, ***Inheritance***, ***Polimorfisme***, ***Modularitas***

Daftar Pustaka

[1] Cay S. Horstmann. *Core Java Volume I–Fundamentals*. Prentice Hall, 11th edition, 2019.

[2] Oracle. The java tutorials. https://docs.oracle.com/javase/tutorial/, 2024.