Bab 3

Konsep Class dan Object

Sub-CPMK yang Dicakup dalam Bab Ini:

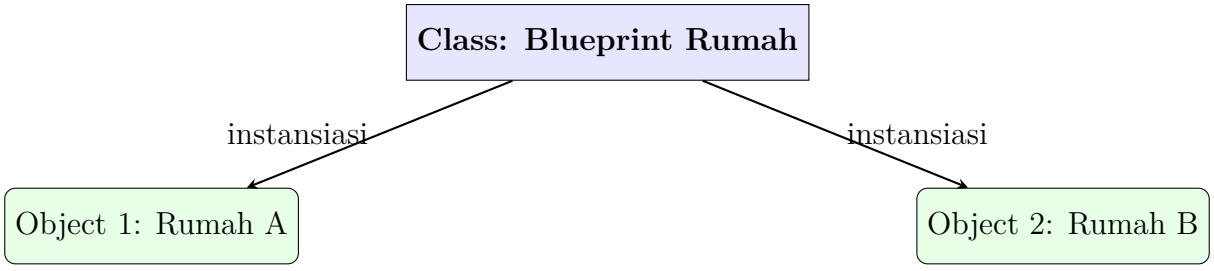
- Sub-CPMK 1.2: Mengidentifikasi class, object, dan attribute dalam studi kasus nyata, serta membuat class sederhana dengan attributes dan methods
- Sub-CPMK 1.2: Memahami perbedaan antara class dan object

3.1 Definisi Class dan Object

3.1.1 Apa itu Class?

Konsep Penting

Class adalah blueprint atau template yang mendefinisikan karakteristik dan perilaku dari suatu entitas [1]. Class mendefinisikan attributes (data) dan methods (behavior) yang akan dimiliki oleh objek.
Analogi: Class seperti cetak biru rumah, sedangkan object adalah rumah yang dibangun berdasarkan cetak biru tersebut.



Gambar 3.1: Konsep Class dan Object

Dalam Java, class didefinisikan dengan keyword `class`:

```java
public class NamaClass {
    // Attributes (instance variables)
    tipeData namaAttribute;

    // Constructor
    public NamaClass() {
        // Inisialisasi
    }

    // Methods
    public void namaMethod() {
        // Implementasi
    }
}
```

Kode Program 3.1: Struktur Dasar Class

3.1.2 Apa itu Object?

Konsep Penting

Object adalah instance (perwujudan) dari class. Object memiliki state (nilai attributes) dan behavior (methods) yang didefinisikan oleh class-nya.

Membuat object dalam Java menggunakan keyword **new**:

```java
NamaClass namaObject = new NamaClass();
```

Kode Program 3.2: Membuat Object

3.1.3 Hubungan Class dan Object

Aspek	Class	Object
Definisi	Template/Blueprint	Instance dari class
Keberadaan	Logical entity	Physical entity
Memory	Tidak menggunakan memory	Menggunakan memory
Jumlah	Satu class	Banyak object dari satu class
Keyword	`class`	`new`

Tabel 3.1: Hubungan Class dan Object

3.2 Attributes (Instance Variables)

3.2.1 Definisi Attributes

Attributes adalah variabel yang didefinisikan di dalam class untuk menyimpan state-/data dari object.

```
public class Mahasiswa {
    // Attributes
    String nim;
    String nama;
    String jurusan;
    int semester;
    double ipk;
} \cite{ref2, ref7}
```

Kode Program 3.3: Contoh Attributes

3.2.2 Jenis Attributes

1. **Instance Variables**: Unik untuk setiap object
2. **Static Variables**: Dibagi oleh semua object (akan dibahas nanti)

3.3 Methods (Behavior)

3.3.1 Definisi Methods

Methods adalah fungsi yang didefinisikan di dalam class untuk menentukan behavior dari object.

```
public class Mahasiswa {
    String nim;
    String nama;
    double ipk;

    // Method untuk menampilkan info
    public void tampilkanInfo() {
        System.out.println("NIM: " + nim);
        System.out.println("Nama: " + nama);
        System.out.println("IPK: " + ipk);
    }

    // Method untuk mengecek kelulusan
    public boolean lulus() {
        return ipk >= 2.0;
    }

    // Method untuk mendapatkan predikat
```

```
19      public String getPredikat() {
20          if (ipk >= 3.5) return "Cum Laude";
21          else if (ipk >= 3.0) return "Sangat Memuaskan";
22          else if (ipk >= 2.5) return "Memuaskan";
23          else return "Cukup";
24      }
25  }
```

Kode Program 3.4: Contoh Methods

3.3.2 Jenis Methods

1. **Accessor Methods (Getter)**: Mengambil nilai attribute
2. **Mutator Methods (Setter)**: Mengubah nilai attribute
3. **Utility Methods**: Melakukan operasi tertentu

3.4 Constructor

3.4.1 Apa itu Constructor?

Konsep Penting

Constructor adalah method khusus yang dipanggil saat object dibuat. Constructor digunakan untuk menginisialisasi attributes object.
Ciri-ciri constructor:

- Nama sama dengan nama class
- Tidak memiliki return type (bahkan bukan `void`)
- Dipanggil otomatis saat object dibuat dengan **`new`**

```
1   public class Mahasiswa {
2       String nim;
3       String nama;
4       double ipk;
5
6       // Constructor tanpa parameter (default constructor)
7       public Mahasiswa() {
8           nim = "0000";
9           nama = "Unknown";
10          ipk = 0.0;
11      }
12
13      // Constructor dengan parameter
14      public Mahasiswa(String nim, String nama) {
15          this.nim = nim;
```

```
16          this.nama = nama;
17          this.ipk = 0.0;
18      }
19
20      // Constructor dengan semua parameter
21      public Mahasiswa(String nim, String nama, double ipk) {
22          this.nim = nim;
23          this.nama = nama;
24          this.ipk = ipk;
25      }
26  }
```

Kode Program 3.5: Contoh Constructor

3.4.2 Constructor Overloading

Class dapat memiliki multiple constructors dengan parameter yang berbeda. Ini disebut ***constructor overloading***.

3.5 Keyword `this`

3.5.1 Penggunaan `this`

Keyword **this** merujuk pada object saat ini. Digunakan untuk:

1. Membedakan instance variable dengan parameter yang namanya sama
2. Memanggil constructor lain dalam class yang sama
3. Mengembalikan instance saat ini

```
1   public class Mahasiswa {
2       private String nama;
3       private double ipk;
4
5       // this untuk membedakan parameter dan instance variable
6       public Mahasiswa(String nama, double ipk) {
7           this.nama = nama;  // this.nama = instance variable
8           this.ipk = ipk;    // nama = parameter
9       }
10
11      // this untuk memanggil constructor lain
12      public Mahasiswa(String nama) {
13          this(nama, 0.0);  // Memanggil constructor di atas
14      }
15
16      // this untuk mengembalikan object saat ini
17      public Mahasiswa setNama(String nama) {
```

```
18          this.nama = nama;
19          return this;  // Method chaining
20      }
21  }
```

Kode Program 3.6: Penggunaan this

3.6 Object Creation dan Instantiation

3.6.1 Membuat Object

Proses membuat object disebut ***instantiation***. Dalam Java, gunakan keyword **new**:

```
1  public class Main {
2      public static void main(String[] args) {
3          // Membuat object dengan default constructor
4          Mahasiswa mhs1 = new Mahasiswa();
5
6          // Membuat object dengan parameterized constructor
7          Mahasiswa mhs2 = new Mahasiswa("123456", "Budi Santoso"
               );
8
9          // Mengakses attributes (jika public)
10         mhs2.ipk = 3.75;
11
12         // Memanggil methods
13         mhs2.tampilkanInfo();
14         System.out.println("Predikat: " + mhs2.getPredikat());
15
16         // Cek kelulusan
17         if (mhs2.lulus()) {
18             System.out.println(mhs2.nama + " dinyatakan LULUS")
                   ;
19         }
20     }
21 }
```

Kode Program 3.7: Membuat dan Menggunakan Object

3.6.2 Memory Allocation

Ketika object dibuat:

1. Memory dialokasikan di **heap**
2. Reference variable disimpan di **stack**
3. Constructor dipanggil untuk inisialisasi

Catatan

Jika tidak ada constructor yang didefinisikan, Java otomatis menyediakan **default constructor** tanpa parameter yang menginisialisasi attributes dengan nilai default (0, null, false).

3.7 Static vs Instance Members

3.7.1 Instance Members

Instance members (attributes dan methods) adalah milik object. Setiap object memiliki copy sendiri.

3.7.2 Static Members

Static members adalah milik class, bukan object. Dibagi oleh semua object.

```
1  public class Mahasiswa {
2      // Instance variable (setiap object punya copy sendiri)
3      private String nama;
4      private double ipk;
5
6      // Static variable (dibagi semua object)
7      private static int jumlahMahasiswa = 0;
8      private static String namaUniversitas = "Universitas Lorem
           Ipsum";
9
10     public Mahasiswa(String nama, double ipk) {
11         this.nama = nama;
12         this.ipk = ipk;
13         jumlahMahasiswa++;  // Increment setiap object dibuat
14     }
15
16     // Instance method
17     public void tampilkanInfo() {
18         System.out.println("Nama: " + nama);
19         System.out.println("IPK: " + ipk);
20     }
21
22     // Static method
23     public static int getJumlahMahasiswa() {
24         return jumlahMahasiswa;
25     }
26
27     public static String getNamaUniversitas() {
28         return namaUniversitas;
29     }
30 }
```

```
31
32 // Penggunaan
33 public class Main {
34     public static void main(String[] args) {
35         // Akses static member tanpa object
36         System.out.println("Universitas: " + Mahasiswa.
               getNamaUniversitas());
37
38         Mahasiswa mhs1 = new Mahasiswa("Budi", 3.5);
39         Mahasiswa mhs2 = new Mahasiswa("Ani", 3.8);
40
41         // Akses static member melalui class name
42         System.out.println("Total mahasiswa: " + Mahasiswa.
               getJumlahMahasiswa());
43         // Output: Total mahasiswa: 2
44     }
45 }
```

Kode Program 3.8: Static vs Instance

Catatan

Best Practice:

- Akses static members melalui class name, bukan object
- Static methods tidak bisa mengakses instance variables secara langsung
- Gunakan static untuk utility methods atau constants

3.8 Contoh Lengkap: Class Buku

```
1  public class Buku {
2      // Attributes
3      private String isbn;
4      private String judul;
5      private String penulis;
6      private int tahunTerbit;
7      private double harga;
8      private boolean tersedia;
9
10     // Static variable
11     private static int jumlahBuku = 0;
12
13     // Constructor
14     public Buku(String isbn, String judul, String penulis,
15                 int tahunTerbit, double harga) {
16         this.isbn = isbn;
17         this.judul = judul;
18         this.penulis = penulis;
```

```
19          this.tahunTerbit = tahunTerbit;
20          this.harga = harga;
21          this.tersedia = true;
22          jumlahBuku++;
23      }
24
25      // Getter methods
26      public String getISBN() {
27          return isbn;
28      }
29
30      public String getJudul() {
31          return judul;
32      }
33
34      public String getPenulis() {
35          return penulis;
36      }
37
38      public int getTahunTerbit() {
39          return tahunTerbit;
40      }
41
42      public double getHarga() {
43          return harga;
44      }
45
46      public boolean isTersedia() {
47          return tersedia;
48      }
49
50      // Setter methods
51      public void setHarga(double harga) {
52          if (harga > 0) {
53              this.harga = harga;
54          }
55      }
56
57      public void setTersedia(boolean tersedia) {
58          this.tersedia = tersedia;
59      }
60
61      // Utility methods
62      public void pinjam() {
63          if (tersedia) {
64              tersedia = false;
65              System.out.println("Buku '" + judul + "' berhasil
                    dipinjam");
66          } else {
67              System.out.println("Buku tidak tersedia");
68          }
```

```
69      }
70
71      public void kembalikan() {
72          tersedia = true;
73          System.out.println("Buku '" + judul + "' berhasil
               dikembalikan");
74      }
75
76      public void tampilkanInfo() {
77          System.out.println("=== Informasi Buku ===");
78          System.out.println("ISBN: " + isbn);
79          System.out.println("Judul: " + judul);
80          System.out.println("Penulis: " + penulis);
81          System.out.println("Tahun: " + tahunTerbit);
82          System.out.println("Harga: Rp " + harga);
83          System.out.println("Status: " + (tersedia ? "Tersedia"
               : "Dipinjam"));
84      }
85
86      // Static method
87      public static int getJumlahBuku() {
88          return jumlahBuku;
89      }
90  }
```

Kode Program 3.9: Class Buku - Contoh Lengkap

```
1   public class TestBuku {
2       public static void main(String[] args) {
3           // Membuat object buku
4           Buku buku1 = new Buku("978-0134685991", "Effective Java
                ",
5                                 "Joshua Bloch", 2018, 450000);
6
7           Buku buku2 = new Buku("978-0596009205", "Head First
                Java",
8                                 "Kathy Sierra", 2005, 350000);
9
10          // Menampilkan informasi
11          buku1.tampilkanInfo();
12          System.out.println();
13
14          // Meminjam buku
15          buku1.pinjam();
16          buku1.pinjam();  // Coba pinjam lagi
17
18          // Mengembalikan buku
19          buku1.kembalikan();
20
21          // Mengubah harga
22          buku2.setHarga(300000);
23
```

```
24          // Menampilkan jumlah buku
25          System.out.println("\nTotal buku: " + Buku.
               getJumlahBuku());
26      }
27  }
```

Kode Program 3.10: Penggunaan Class Buku

Aktivitas Pembelajaran

1. **Identifikasi Class**: Dari studi kasus sistem perpustakaan, identifikasi minimal 5 class yang diperlukan beserta attributes dan methods-nya.

2. **Implementasi Class**: Buat class `Mobil` dengan attributes: merk, model, tahun, warna, harga. Tambahkan constructor dan methods yang sesuai.

3. **Object Interaction**: Buat class `Dosen` dan `MataKuliah`. Buat program yang mendemonstrasikan interaksi antara object dosen dan mata kuliah.

4. **Static vs Instance**: Buat class `Counter` dengan static variable untuk menghitung jumlah object yang dibuat. Test dengan membuat beberapa object.

5. **Code Review**: Bertukar kode dengan teman, review class yang dibuat teman Anda. Berikan feedback tentang naming, encapsulation, dan design.

Latihan dan Refleksi

1. Jelaskan perbedaan antara class dan object! Berikan analogi dari dunia nyata.

2. Apa fungsi constructor? Apa yang terjadi jika tidak mendefinisikan constructor dalam class?

3. Jelaskan perbedaan antara instance variable dan static variable! Kapan sebaiknya menggunakan static variable?

4. Buat class `RekeningBank` dengan attributes: nomorRekening, namaPemilik, saldo. Tambahkan methods: setor(), tarik(), cekSaldo(). Implementasikan validasi yang sesuai.

5. Buat class `Lingkaran` dengan attribute radius. Tambahkan methods untuk menghitung luas dan keliling. Buat program untuk membuat beberapa object lingkaran dengan radius berbeda.

6. Modifikasi class `Mahasiswa` untuk menambahkan array nilai mata kuliah. Tambahkan method untuk menghitung IPK berdasarkan nilai-nilai tersebut.

7. Buat class `Produk` untuk sistem e-commerce dengan attributes yang sesuai. Implementasikan method untuk menghitung harga setelah diskon.

8. **Refleksi**: Apa tantangan terbesar yang Anda hadapi dalam memahami konsep class dan object? Bagaimana Anda mengatasinya?

Asesmen (Evaluasi Kinerja)

Instrumen Penilaian untuk Sub-CPMK 1.2

A. Pilihan Ganda

1. Keyword yang digunakan untuk membuat object dalam Java adalah:
   (a) create
   (b) new
   (c) object
   (d) instance

2. Manakah pernyataan yang BENAR tentang constructor?
   (a) Constructor harus memiliki return type
   (b) Nama constructor harus berbeda dengan nama class
   (c) Constructor dipanggil otomatis saat object dibuat
   (d) Satu class hanya boleh memiliki satu constructor

3. Static variable dalam class:
   (a) Unik untuk setiap object
   (b) Dibagi oleh semua object
   (c) Tidak bisa diakses dari luar class
   (d) Harus selalu private

4. Keyword `this` digunakan untuk:
   (a) Merujuk pada superclass
   (b) Merujuk pada object saat ini
   (c) Membuat object baru
   (d) Menghapus object

B. Essay

1. Jelaskan perbedaan antara instance method dan static method! Berikan contoh penggunaan masing-masing.

2. Mengapa encapsulation penting? Jelaskan menggunakan class **`RekeningBank`** sebagai contoh.

C. Coding Challenge

1. Buat class **`Karyawan`** dengan attributes: nik, nama, jabatan, gajiPokok. Tambahkan:
   - Constructor dengan parameter
   - Getter dan setter methods
   - Method hitungGajiBersih() yang menghitung gaji setelah potongan pajak 10%
   - Method tampilkanInfo()
2. Buat program main yang membuat minimal 3 object karyawan dan menampilkan informasi mereka.

Rubrik Penilaian: Lihat Lampiran A

Checklist Pencapaian Kompetensi

Centang item berikut setelah Anda yakin telah menguasainya:

☐ Saya dapat menjelaskan perbedaan antara class dan object

☐ Saya dapat membuat class dengan attributes dan methods

☐ Saya dapat membuat constructor dengan dan tanpa parameter

☐ Saya dapat membuat dan menggunakan object dari class

☐ Saya memahami penggunaan keyword `this`

☐ Saya dapat membedakan instance variable dan static variable

☐ Saya dapat mengidentifikasi class dan object dari studi kasus nyata

☐ Saya dapat mengimplementasikan getter dan setter methods

Rangkuman

Bab ini membahas konsep fundamental OOP: Class dan Object. Class adalah blueprint untuk membuat object, sedangkan object adalah instance dari class.

Poin Kunci:

- Class terdiri dari attributes (data) dan methods (behavior)

- Object dibuat menggunakan keyword `new`
- Constructor adalah method khusus untuk inisialisasi object
- Keyword `this` merujuk pada object saat ini
- Instance members milik object, static members milik class
- Encapsulation dicapai dengan access modifiers dan getter/setter

Kata Kunci: ***Class***, ***Object***, ***Attribute***, ***Method***, ***Constructor***, ***this***, ***static***, ***instance***, ***instantiation***

Daftar Pustaka

[1] Joshua Bloch. *Effective Java*. Addison-Wesley Professional, 3rd edition, 2018.