Bab 14

Evaluasi, Refleksi, dan Integrasi Kompetensi

Bab ini berisi asesmen komprehensif untuk mengukur pencapaian seluruh CPMK yang telah dipelajari sepanjang mata kuliah Pemrograman Berorientasi Objek.

14.1 Asesmen Akhir Komprehensif

14.1.1 Petunjuk Umum

Asesmen ini dirancang untuk mengukur pencapaian CPMK-1, CPMK-2, CPMK-3, dan CPMK-4 secara menyeluruh [5]. Kerjakan dengan jujur dan mandiri.

Alokasi Waktu:

- Bagian A (Pilihan Ganda): 30 menit
- Bagian B (Essay): 45 menit
- Bagian C (Analisis Kode): 45 menit
- Bagian D (Coding Challenge): 90 menit

14.1.2 Bagian A: Pilihan Ganda (CPMK-1)

Petunjuk: Pilih satu jawaban yang paling tepat.

1. Manakah yang BUKAN merupakan pilar OOP?
   (a) Abstraksi
   (b) Enkapsulasi
   (c) Kompilasi
   (d) Polimorfisme

2. Dalam Java, keyword untuk mewarisi class adalah:
   (a) inherits
   (b) extends
   (c) implements
   (d) super
3. Method overriding terjadi ketika:
   (a) Dua method dalam class yang sama memiliki nama sama
   (b) Subclass mendefinisikan ulang method dari superclass
   (c) Method memiliki parameter yang berbeda
   (d) Method dipanggil berkali-kali
4. Interface dalam Java:
   (a) Dapat memiliki instance variables
   (b) Semua methods harus abstract (sebelum Java 8)
   (c) Dapat di-instantiate
   (d) Hanya boleh memiliki satu method
5. Prinsip SOLID yang menyatakan "class harus terbuka untuk extension tapi tertutup untuk modification" adalah:
   (a) Single Responsibility Principle
   (b) Open/Closed Principle
   (c) Liskov Substitution Principle
   (d) Dependency Inversion Principle
6. Design pattern yang memastikan hanya ada satu instance dari class adalah:
   (a) Factory Pattern
   (b) Singleton Pattern
   (c) Observer Pattern
   (d) Strategy Pattern
7. Dalam Java Collections, struktur data yang tidak mengizinkan duplikat adalah:
   (a) List
   (b) Set
   (c) Map
   (d) Queue
8. Exception yang harus di-handle dengan try-catch disebut:
   (a) RuntimeException

(b) Checked Exception

(c) Unchecked Exception

(d) Error

9. Dalam UML Class Diagram, relasi "has-a" digambarkan dengan:

   (a) Generalization

   (b) Association/Aggregation/Composition

   (c) Dependency

   (d) Realization

10. Kelas yang digunakan untuk menulis objek ke file secara serialization adalah:

    (a) FileWriter

    (b) ObjectOutputStream

    (c) BufferedReader

    (d) Scanner

11. Annotation untuk menandai method sebagai test case dalam JUnit adalah:

    (a) @Test

    (b) @TestCase

    (c) @Unit

    (d) @Assert

12. Refactoring yang memecah method panjang menjadi beberapa method kecil disebut:

    (a) Extract Method

    (b) Inline Method

    (c) Replace Conditionals with Polymorphism

    (d) Move Method

14.1.3 Bagian B: Essay (CPMK-1, CPMK-2)

Petunjuk: Jawab dengan jelas dan lengkap.

1. Jelaskan perbedaan antara abstract class dan interface dalam Java! Kapan sebaiknya menggunakan abstract class dan kapan menggunakan interface? Berikan contoh kasus untuk masing-masing.

2. Jelaskan konsep polymorphism dalam OOP! Berikan contoh kode Java yang mendemonstrasikan compile-time polymorphism dan runtime polymorphism.

3. Jelaskan 5 prinsip SOLID! Pilih salah satu prinsip dan berikan contoh kode yang melanggar prinsip tersebut, kemudian perbaiki kode tersebut.

4. Gambarkan Class Diagram untuk sistem perpustakaan sederhana yang memiliki minimal 5 class dengan relasi yang tepat (association, aggregation, composition, inheritance).

5. Jelaskan konsep Test-Driven Development (TDD)! Apa keuntungan menggunakan TDD dalam pengembangan software?

6. Jelaskan contoh refactoring sederhana yang dapat Anda lakukan untuk memperbaiki code smell "Long Method".

14.1.4 Bagian C: Analisis Kode (CPMK-3, CPMK-4)

Petunjuk: Analisis kode berikut dan jawab pertanyaan.

```
public class BankAccount {
    public String accountNumber;
    public double balance;
    public String ownerName;

    public void deposit(double amount) {
        balance = balance + amount;
    }

    public void withdraw(double amount) {
        balance = balance - amount;
    }

    public double getBalance() {
        return balance;
    }
}

public class SavingsAccount extends BankAccount {
    public double interestRate;

    public void addInterest() {
        balance = balance + (balance * interestRate);
    }
}
```

Kode Program 14.1: Kode untuk Dianalisis

Pertanyaan:

1. Identifikasi minimal 5 masalah dalam kode di atas terkait dengan prinsip OOP dan best practices.

2. Perbaiki kode tersebut dengan menerapkan encapsulation yang tepat.

3. Tambahkan validasi yang diperlukan pada method `deposit()` dan `withdraw()`.

4. Implementasikan constructor yang sesuai untuk kedua class.

5. Apakah kode ini melanggar prinsip SOLID? Jelaskan!

14.1.5 Bagian D: Coding Challenge (CPMK-2, CPMK-3, 4)

Petunjuk: Implementasikan sistem berikut dengan menerapkan semua konsep OOP yang telah dipelajari.

Studi Kasus: Sistem Manajemen Toko Buku Online

Buat sistem manajemen toko buku online dengan requirements berikut:

Requirements:

1. **Class `Buku`**:
   - Attributes: isbn, judul, penulis, penerbit, tahunTerbit, harga, stok
   - Methods: getInfo(), updateStok(), hitungDiskon()
2. **Class `BukuFisik`** (extends **`Buku`**):
   - Attribute tambahan: berat, dimensi
   - Method: hitungOngkir()
3. **Class `Ebook`** (extends **`Buku`**):
   - Attribute tambahan: ukuranFile, format
   - Method: download()
4. **Class `Pelanggan`**:
   - Attributes: idPelanggan, nama, email, alamat
   - Methods: register(), updateProfile()
5. **Class `Keranjang`**:
   - Attributes: daftarBuku (ArrayList), pelanggan
   - Methods: tambahBuku(), hapusBuku(), hitungTotal(), checkout()
6. **Interface `Pembayaran`**:
   - Methods: prosesPembayaran(), verifikasiPembayaran()
7. **Class `PembayaranKartuKredit`** dan **`PembayaranTransfer`** (implements **`Pembayaran`**)

Kriteria Penilaian:

- Penerapan encapsulation (private attributes, getter/setter)
- Penggunaan inheritance yang tepat
- Implementasi polymorphism
- Penggunaan interface

- Validasi data yang sesuai
- Exception handling
- Code quality (naming, comments, structure)
- Unit testing untuk minimal 3 methods

14.2 Rubrik Penilaian Komprehensif

14.2.1 Rubrik untuk Coding Challenge

Kriteria	**Excellent (4)**	**Good (3)**	**Fair (2)**	**Poor (1)**
Encapsulation	Semua attributes private dengan getter/setter yang tepat	Sebagian besar private	Beberapa public	Semua public
Inheritance	Hierarki class tepat, code reuse optimal	Inheritance digunakan dengan baik	Inheritance kurang optimal	Tidak menggunakan inheritance
Polymorphism	Men-de-mon-stra-si-kan compile-time dan runtime polymorphism	Menggunakan salah satu jenis polymorphism	Polymorphism minimal	Tidak ada polymorphism
Interface	Interface digunakan dengan tepat	Interface digunakan tapi kurang optimal	Interface ada tapi tidak efektif	Tidak menggunakan interface
Exception Handling	Comprehensive error handling	Error handling untuk kasus utama	Minimal error handling	Tidak ada error handling
Code Quality	Clean code, well-documented, follows conventions	Good structure, adequate comments	Basic structure, minimal comments	Poor structure, no comments
Unit Testing	Comprehensive tests, good coverage	Tests untuk fungsi utama	Minimal testing	No testing

Tabel 14.1: Rubrik Penilaian Coding Challenge

Komponen	**Bobot**
Bagian A: Pilihan Ganda	20%
Bagian B: Essay	20%
Bagian C: Analisis Kode	20%
Bagian D: Coding Challenge	40%
Total	**100%**

Tabel 14.2: Matriks Bobot Penilaian Akhir

14.2.2 Bobot Penilaian

14.3 Tinjauan Pencapaian Kompetensi Secara Menyeluruh

14.3.1 Pemetaan Asesmen ke CPMK

CPMK	**Diukur Melalui**
CPMK-1: Memahami konsep dasar OOP	Pilihan Ganda (1-4), Essay (1-2)
CPMK-2: Merancang solusi dengan UML	Essay (4), Coding Challenge (Design)
CPMK-3: Mengimplementasikan OOP dengan SOLID	Analisis Kode, Coding Challenge (Implementation)
CPMK-4: Menganalisis dan mengevaluasi kualitas kode	Analisis Kode, Coding Challenge (Quality)

Tabel 14.3: Pemetaan Komponen Asesmen ke CPMK

14.3.2 Self-Assessment Checklist

Sebelum mengerjakan asesmen akhir, pastikan Anda telah menguasai:

Checklist Pencapaian Kompetensi

Centang item berikut setelah Anda yakin telah menguasainya:

☐ Konsep dasar OOP (class, object, encapsulation, inheritance, polymorphism)

☐ Perbedaan abstract class dan interface

☐ Cara membuat dan menggunakan UML diagrams

☐ Implementasi inheritance dan polymorphism dalam Java

☐ Exception handling yang tepat

☐ Prinsip-prinsip SOLID

☐ Design patterns dasar (Singleton, Factory, Observer)

☐ Java Collections Framework

☐ File I/O dan serialization

☐ Unit testing dengan JUnit

☐ Refactoring dan identifikasi code smell

☐ Best practices dalam penulisan kode Java

14.4 Rekomendasi Tindak Lanjut Pembelajaran

14.4.1 Untuk Mahasiswa yang Sudah Menguasai Semua CPMK

Jika Anda telah menguasai semua materi dengan baik, pertimbangkan untuk:

1. **Proyek Portofolio**: Buat aplikasi Java lengkap yang mendemonstrasikan semua konsep OOP
2. **Eksplorasi Framework**: Pelajari Spring Framework atau Jakarta EE
3. **Design Patterns Lanjut**: Pelajari 23 GoF Design Patterns secara mendalam
4. **Clean Code**: Baca buku "Clean Code" oleh Robert C. Martin
5. **Kontribusi Open Source**: Berkontribusi pada proyek Java open source
6. **Sertifikasi**: Persiapkan Oracle Certified Professional Java Programmer

14.4.2 Untuk Mahasiswa yang Masih Perlu Perbaikan

Jika ada CPMK yang belum dikuasai dengan baik:

1. **Review Materi**: Baca ulang bab terkait dengan lebih teliti
2. **Praktik Lebih Banyak**: Kerjakan lebih banyak latihan coding
3. **Konsultasi**: Diskusikan dengan dosen atau asisten
4. **Study Group**: Bentuk kelompok belajar dengan teman
5. **Online Resources**: Manfaatkan tutorial online (Coursera, Udemy, YouTube)
6. **Coding Practice**: Gunakan platform seperti HackerRank, LeetCode untuk latihan

14.4.3 Sumber Belajar Tambahan

Buku:

- "Effective Java" - Joshua Bloch [1]
- "Head First Design Patterns" - Freeman & Robson [3]
- "Clean Code" - Robert C. Martin [6]
- "Refactoring" - Martin Fowler [2]
- "Design Patterns" - Gamma et al. [4]

Online Courses:

- Java Programming and Software Engineering Fundamentals (Coursera)
- Object Oriented Programming in Java (Udemy)
- Design Patterns in Java (Pluralsight)

Practice Platforms:

- HackerRank (Java track)
- LeetCode (OOP problems)
- Codewars
- Exercism (Java track)

Rangkuman

Selamat! Anda telah menyelesaikan perjalanan pembelajaran Pemrograman Berorientasi Objek. Dari memahami konsep dasar hingga menerapkan design patterns dan best practices, Anda telah membangun fondasi yang kuat dalam pengembangan software modern.

Apa yang Telah Anda Pelajari:

- Paradigma OOP dan 4 pilar utama
- Class, Object, Constructor, dan Method
- Encapsulation dan Information Hiding
- Inheritance dan Polymorphism
- Abstract Class dan Interface
- UML untuk perancangan sistem
- Exception Handling

- Prinsip SOLID
- Design Patterns
- Collections dan Generics
- File I/O dan Serialization
- Unit Testing dan TDD
- Refactoring dan Code Quality

Langkah Selanjutnya: Terus praktik, buat proyek nyata, dan jangan berhenti belajar. OOP adalah fondasi, tetapi masih banyak yang bisa dipelajari dalam dunia pengembangan software!

Daftar Pustaka

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

[2] Martin Fowler. *Refactoring: Improving the Design of Existing Code*. Addison-Wesley Professional, 2nd edition, 2018.

[3] Eric Freeman and Elisabeth Robson. *Head First Design Patterns*. O'Reilly Media, 2nd edition, 2020.

[4] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.

[5] Craig Larman. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design*. Prentice Hall, 3rd edition, 2004.

[6] Robert C. Martin. *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Prentice Hall, 2017.