Buku Ajar & Modul
Praktikum Perkuliahan

SISTEM INFORMASI | ITATS INSTITUT TEKNOLOGI ADHI TAMA SURABAYA

PEMROGRAMAN BERORIENTASI OBJEK

"Panduan Komprehensif Teori, Praktikum Java SE, Pemodelan Kelas, 4 Pilar OOP, hingga Pembuatan Apiikasi GUI Berbasis Rencana Pembelajaran Semester (RPS)"

Budanis Dwi Meilani, S.T., M.Kom.
Ressa Uttungga, S.Kom, M.Kom.

Edisi Pertama, November 2025

PROGRAM STUDI S1 SISTEM INFORMASI
FAKULTAS TEKNIK ELEKTRO DAN TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI ADHI TAMA SURABAYA

BUKU AJAR & MODUL PRAKTIKUM PERKULIAHAN

PEMROGRAMAN BERORIENTASI OBJEK (KODE MK: 25132105)

Panduan Komprehensif Teori, Praktikum Java SE, Pemodelan Kelas, 4 Pilar OOP, hingga Pembuatan Aplikasi GUI Berbasis Rencana Pembelajaran Semester (RPS)

Pemrograman Berorientasi Objek (25132105) - S1 Sistem Informasi ITATS

Dosen Pengembang RPS & Pengampu:

Budanis Dwi Meilani & Resa Uttungga

Bobot: 3 SKS (Semester 2)
Prasyarat: Algoritma & Pemrograman Terstruktur dan Praktikum Algoritma & Pemrograman Terstruktur
Program Studi S1 Sistem Informasi
Fakultas Teknik Elektro dan Teknologi Informasi
Institut Teknologi Adhi Tama Surabaya (ITATS)
Edisi Pertama, Tahun Akademik 2026/2027

KATA PENGANTAR

Puji dan syukur kami panjatkan ke hadirat Tuhan Yang Maha Esa atas karunia dan rahmat-Nya sehingga buku ajar dan modul praktikum perkuliahan "Pemrograman Berorientasi Objek (Kode MK: 25132105)" ini dapat diselesaikan dengan baik.

Pemrograman Berorientasi Objek (Object-Oriented Programming / OOP) merupakan paradigma rekayasa perangkat lunak paling fundamental dan luas digunakan dalam pengembangan sistem informasi modern skala enterprise. Berbeda dengan pemrograman prosedural yang berpusat pada fungsi linier, paradigma OOP memandang dunia perangkat lunak sebagai kumpulan entitas mandiri (Objek) yang memiliki keadaan (State/Attributes) dan perilaku (Behavior/Methods) yang saling berinteraksi.

Buku modul ini disusun secara sistematis mengacu pada Rencana Pembelajaran Semester (RPS) resmi Kurikulum 2025 Program Studi S1 Sistem Informasi ITATS, mencakup 6 Bahan Kajian Utama:

1. Konsep Dasar Pemrograman Berorientasi Objek (Paradigma OOP, Java Virtual Machine, Struktur Kode).
2. Kelas dan Objek (Class Blueprint, Instansiasi di Heap, Konstruktor, Keyword 'this', Overloading).
3. Konsep Enkapsulasi dan Abstraksi (Information Hiding, Access Modifiers, Getter & Setter, Desain Abstraksi).
4. Pewarisan (Inheritance) dan Polimorfisme (Superclass/Subclass, Keyword 'extends'/'super', Dynamic Method Overriding).

5. Kelas Abstrak dan Antarmuka (Abstract Class vs Interface, Multiple Implementation, Loose Coupling).
6. Pembuatan Program Berbasis Antarmuka Grafis (Java GUI Swing: JFrame, JPanel, Layouts, Event-Driven Programming).

Modul ini juga dilengkapi dengan panduan penugasan resmi RPS termasuk Program Awal Java, Program Java Konsep PBO, Program Inheritance, Program Abstrak & Interface, serta persiapan proyek akhir GUI untuk evaluasi UTS dan UAS.

Penulis menyampaikan apresiasi dan terima kasih setinggi-tingginya kepada pimpinan Institut Teknologi Adhi Tama Surabaya (ITATS), pimpinan Fakultas Teknik Elektro dan Teknologi Informasi, serta rekan-rekan dosen di Program Studi Sistem Informasi atas kolaborasi dan dukungannya. Semoga buku modul ini dapat mengantarkan mahasiswa menjadi software engineer yang tangguh, terampil, dan berwawasan arsitektur perangkat lunak yang kokoh.

Kritik dan saran senantiasa kami harapkan demi penyempurnaan edisi berikutnya.

Surabaya, November 2025

Tim Dosen Pengampu,

Budanis Dwi Meilani, S.T., M. Kom.

& Isa Albanna, S. Kom., M. Kom.

PETUNJUK PENGGUNAAN & PETA KOMPETENSI RPS

Buku modul ini disusun berbasis Outcome-Based Education (OBE) dengan metode Student-Centered Learning, Diskusi, Praktikum Laboratorium Komputer, dan Project-Based Learning.

Bagi Dosen Pengampu:

1. Mengarahkan proses belajar mengajar sesuai pemetaan 16 Minggu pada RPS resmi ITATS.
2. Memfasilitasi praktikum pemrograman Java menggunakan kakas resmi (JDK 17/21 LTS, IDE Apache NetBeans / IntelliJ IDEA / Eclipse / VS Code).
3. Melakukan evaluasi unjuk kerja berbasis portofolio kode program yang bersih, modular, dan mematuhi prinsip OOP.

Bagi Mahasiswa:

1. Pelajari Capaian Pembelajaran Lulusan (CPL-7) dan Capaian Pembelajaran Mata Kuliah (CPMK-71) di awal perkuliahan.
2. Praktikkan setiap contoh kode Java di komputer laboratorium secara langsung (hands-on coding), jangan hanya membaca teori.
3. Kerjakan seluruh Lembar Kerja Praktikum (Worksheets) yang merepresentasikan tugas resmi RPS:
    - Tugas 1: Program Awal Java (Minggu 1-2 | Bobot 10%).
    - Tugas 2: Program Java Konsep PBO (Minggu 3-4 | Bobot 15%).
    - Evaluasi Tengah Semester (UTS): Enkapsulasi & Abstraksi (Minggu 5-8 | Bobot 25%).
    - Tugas 3: Program Inheritance Java (Minggu 9-10 | Bobot 15%).
    - Tugas 4: Program Abstrak & Interface Java (Minggu 11-12 | Bobot 10%).

- Tugas 5 & UAS: Pembuatan Program GUI Java Swing (Minggu 13-16 | Bobot 25%).

4. Bangun portofolio kode program di repositori GitHub sebagai bukti kompetensi profesional rekayasa perangkat lunak.

Capaian Pembelajaran Lulusan (CPL) & CPMK

Kode	Deskripsi Capaian Pembelajaran (RPS)
CPL-7	Mampu memahami dan menggunakan berbagai metodologi pengembangan sistem beserta alat permodelan serta menganalisis kebutuhan pengguna dalam membangun sistem informasi yang berkualitas untuk mencapai tujuan organisasi dan/atau enterprise.
CPMK-71	Mampu memahami berbagai metodologi pengembangan sistem.

Pemetaan Sub-CPMK & Bobot Penilaian Tugas RPS

Sub-CPMK & Bobot	Kemampuan Akhir Tiap Tahapan Belajar
SubCPMK 1 (Minggu 1-2 | 10%)	Mahasiswa mampu menjelaskan konsep dasar OOP (class, object) dan mengoperasikan lingkungan pemrograman Java [C2, A2, P3].
SubCPMK 2 (Minggu 3-4 | 20%)	Mahasiswa mampu mendefinisikan kelas, mendeklarasikan atribut/metode, dan menginstansiasi objek serta konstruktor [C2, A2, P3].
SubCPMK 3 (Minggu 5-7 | 20%)	Mahasiswa mampu menerapkan konsep encapsulation dan abstraction dalam desain program Java terstruktur [C3, A2, P3].
SubCPMK 4 (Minggu 9-10 | 20%)	Mahasiswa mampu menerapkan inheritance dan polymorphism untuk menyusun program yang modular dan fleksibel [C3, A2, P3].
SubCPMK 5 (Minggu 11-12 | 10%)	Mahasiswa mampu menggunakan prinsip desain OOP menggunakan kelas abstract dan Interface [C3, A2, P2].

Sub-CPMK & Bobot	Kemampuan Akhir Tiap Tahapan Belajar
SubCPMK 6 (Minggu 13-15 | 20%)	Mahasiswa mampu membuat program aplikasi antarmuka grafis (GUI) interaktif menggunakan Java [C3, A2, P3].

Roadmap Perkuliahan 16 Pertemuan & Korelasi Tugas RPS

Minggu	Bahan Kajian & Modul	Bentuk Evaluasi / Tugas
Minggu 1-2	Konsep Dasar OOP (Bahan Kajian 1 / SubCPMK 1)	Kuliah, Praktik CLI/IDE, Tugas 1: Program Awal Java (10%)
Minggu 3-4	Kelas dan Objek (Bahan Kajian 2 / SubCPMK 2)	Praktik Konstruktor & Instansiasi, Tugas 2: Program PBO (15%)
Minggu 5-7	Enkapsulasi & Abstraksi (Bahan Kajian 3 / SubCPMK 3)	Workshop Getter-Setter, Access Modifiers, Case Studies
Minggu 8	Evaluasi Tengah Semester (UTS)	Ujian Praktik Pemrograman Berorientasi Objek (Bobot 25%)
Minggu 9-10	Inheritance & Polymorphism (Bahan Kajian 4 / SubCPMK 4)	Praktik Superclass/Subclass, Tugas 3: Inheritance Java (15%)
Minggu 11-12	Kelas Abstrak & Interface (Bahan Kajian 5 / SubCPMK 5)	Praktik Kontrak Desain, Tugas 4: Abstract Interface (10%)
Minggu 13-15	Pembuatan Program GUI (Bahan Kajian 6 / SubCPMK 6)	Praktik Java Swing, Layouts, ActionListener, Event Handling
Minggu 16	Evaluasi Akhir Semester (UAS)	Presentasi & Demonstrasi Proyek Aplikasi GUI Java (Bobot 25%)

DAFTAR ISI

BAB 1: KONSEP DASAR PEMROGRAMAN BERORIENTASI OBJEK (OOP)

Capaian Pembelajaran

1. Membedakan paradigma pemrograman terstruktur (prosedural) dengan paradigma pemrograman berorientasi objek (OOP).
2. Menjelaskan 4 pilar fundamental OOP: Enkapsulasi (Encapsulation), Abstraksi (Abstraction), Pewarisan (Inheritance), dan Polimorfisme (Polymorphism).
3. Memahami arsitektur ekosistem Java: Java Development Kit (JDK), Java Runtime Environment (JRE), dan Java Virtual Machine (JVM).
4. Menganalisis proses kompilasi kode sumber Java (.java) menjadi Bytecode platform-independent (.class) hingga dieksekusi oleh JVM.
5. Menyusun dan mengeksekusi program sederhana Java menggunakan metode 'main' sesuai konvensi penamaan standar industri.
6. Menyelesaikan penugasan resmi RPS: 'Tugas 1: Program Awal Java' (Bobot 10%).

1.1 Evolusi Paradigma: Prosedural vs. Pemrograman Berorientasi Objek

Pada era awal rekayasa perangkat lunak, paradigma Pemrograman Prosedural (Procedural Programming seperti C, Pascal, BASIC) mendominasi pengembangan aplikasi. Paradigma prosedural berfokus pada rangkaian instruksi linier dan fungsi-fungsi (prosedur) yang memanipulasi data global yang terpisah.

Ketika skala aplikasi berkembang menjadi sistem enterprise dengan ratusan ribu baris kode, pendekatan prosedural mengalami masalah 'Spaghetti Code', di mana perubahan pada satu struktur data

global dapat merusak fungsi-fungsi lain di seluruh aplikasi, serta minimnya kemampuan penggunaan ulang kode (reusability).

Pemrograman Berorientasi Objek (Object-Oriented Programming / OOP) hadir sebagai solusi revolusioner. Filosofi utama OOP adalah memodelkan perangkat lunak menyerupai cara berpikir manusia dalam memandang dunia nyata: sebagai sekumpulan Objek mandiri yang memiliki karakteristik (State/Attributes) dan kemampuan tindakan (Behavior/Methods) yang saling berinteraksi melalui pengiriman pesan (Message Passing).

Matriks Perbandingan Prosedural vs OOP

Parameter	Pemrograman Prosedural	Pemrograman Berorientasi Objek (OOP)
Fokus Utama	Algoritma & Urutan Prosedur (Functions)	Objek yang menggabungkan Data & Method
Struktur Data	Terpisah dari fungsi pengolahnya	Terbungkus rapat di dalam Class (Encapsulated)
Visibilitas Data	Data global dapat diakses bebas (Rentan)	Data terproteksi melalui Access Modifiers
Reusability	Terbatas pada pemanggilan fungsi	Sangat tinggi melalui Inheritance & Interface
Skalabilitas	Sulit dikelola untuk sistem besar	Sangat modular dan mudah dirawat (Maintainable)

1.2 Empat Pilar Fundamental OOP (Core Pillars)

Keunggulan dan ketangguhan paradigma OOP ditopang oleh empat pilar utama:

1. Enkapsulasi (Encapsulation):

Mekanisme membungkus data (atribut) dan kode yang memanipulasinya (metode) menjadi satu kesatuan unit kelas, serta menyembunyikan detail implementasi internal dari akses langsung pihak luar (Information Hiding).

2. Abstraksi (Abstraction):

Prinsip menyederhanakan kompleksitas sistem dengan hanya menampilkan fitur-fitur esensial kepada pengguna dan menyembunyikan rincian komputasi teknis di balik layar.

3. Pewarisan (Inheritance):

Kemampuan sebuah kelas baru (Subclass/Derived Class) untuk mewarisi atribut dan metode dari kelas yang sudah ada (Superclass/Base Class), sehingga mencegah duplikasi kode.

4. Polimorfisme (Polymorphism):

Prinsip 'satu antarmuka, banyak bentuk' yang memungkinkan suatu metode memiliki perilaku berbeda tergantung pada objek konkret yang mengeksekusinya pada saat runtime.

1.3 Arsitektur Ekosistem Java: JDK, JRE, dan JVM

Bahasa pemrograman Java dirancang oleh James Gosling (Sun Microsystems/Oracle) dengan semboyan legendaris: 'Write Once, Run Anywhere' (WORA). Keunggulan portabilitas lintas sistem operasi ini dicapai melalui arsitektur tiga lapis:

- Java Virtual Machine (JVM): Mesin virtual abstrak yang bertugas mengeksekusi instruksi Bytecode ke dalam kode mesin lokal (Native Machine Code) pada sistem operasi target (Windows, Linux, macOS).

- Java Runtime Environment (JRE): Paket lingkungan runtime yang memuat JVM beserta pustaka kelas standar Java (Java Core Libraries) yang dibutuhkan untuk menjalankan program.
- Java Development Kit (JDK): Perangkat lengkap untuk pengembang perangkat lunak, mencakup compiler (javac), debugger, archiver (jar), dan JRE.

1.4 Struktur Dasar Program Java & Konvensi Penamaan

Setiap kode program Java harus berada di dalam definisi sebuah `class`. Titik masuk (entry point) eksekusi program selalu dimulai dari metode utama: `public static void main(String[] args)`.

Konvensi Penamaan Standar Java (Code Conventions):

1. Nama Kelas (PascalCase): Diawali huruf kapital (contoh: `Mahasiswa`, `RekeningBank`, `SistemInformasiAkademik`).
2. Nama Metode & Variabel (camelCase): Diawali huruf kecil, kata berikutnya kapital (contoh: `hitungNilaiAkhir()`, `totalBiaya`, `tampilkanBiodata()`).
3. Nama Konstanta (ALL_CAPS): Huruf kapital semua dipisahkan garis bawah (contoh: `MAX_CAPACITY`, `NILAI_PHI`).

Contoh Program Dasar Java

```
public class ProgramAwal {
    public static void main(String[] args) {
        String prodi = "S1 Sistem Informasi ITATS";
        System.out.println("Selamat Datang di Pemrograman
Berorientasi Objek!");
        System.out.println("Program Studi: " + prodi);
  }
}
```

Aktivitas Praktikum & Penugasan: LEMBAR KERJA 1 (TUGAS 1 RPS): Program Awal Java & Setup Lingkungan

Kerjakan penugasan individu 'Tugas 1: Program Awal Java' (Bobot 10%) dengan menginstal JDK, mengonfigurasi IDE (NetBeans/IntelliJ/VS Code), dan menulis program awal Java.

Bagian A: Identifikasi Lingkungan Pengembangan (Development Environment)

Komponen / Item Evaluasi	Deskripsi / Isian Kode Praktikum
1. Versi Java Development Kit (JDK) Terinstal:	Java SE Version ..... (LTS 17 / 21)
2. Integrated Development Environment (IDE) yang Digunakan:	Apache NetBeans / IntelliJ IDEA / VS Code
3. Perintah Kompilasi Program via Terminal/CLI:	javac NamaProgram.java
4. Perintah Eksekusi Bytecode Program via Terminal/CLI:	java NamaProgram
5. Fungsi Keyword 'public', 'static', 'void', 'main':	public (Akses Global) | static (Tanpa Instansiasi) | void (Tanpa Return)

Bagian B: Spesifikasi Program Awal Java (Tugas 1)

Komponen / Item Evaluasi	Deskripsi / Isian Kode Praktikum
Nama File Sumber Java:	DataMahasiswa.java
Output yang Ditampilkan:	Biodata Mahasiswa (NIM, Nama, Prodi, Mata Kuliah, Semester)

Komponen / Item Evaluasi	Deskripsi / Isian Kode Praktikum
Implementasi Variabel & Tipe Data:	String nim, nama; int semester; double ipkTarget;
Status Kompilasi & Eksekusi:	Kompilasi Sukses Tanpa Warning / Error (Exit Code 0)
Tangkapan Layar Hasil Eksekusi Terminal:	Terlampir pada berkas laporan tugas

Lembar Refleksi & Pengumpulan Portofolio:

1. Jelaskan mengapa konsep 'Bytecode' dan 'JVM' membuat bahasa Java memiliki sifat 'Write Once, Run Anywhere' (WORA):

2. Lampirkan tangkapan layar (screenshot) kode program dan hasil eksekusi terminal pada lembar pengumpulan tugas Anda!

Rangkuman Eksekutif

- Pemrograman Berorientasi Objek (OOP) memodelkan perangkat lunak sebagai kesatuan objek yang memiliki atribut (data) dan metode (perilaku).
- Empat pilar utama OOP adalah Enkapsulasi, Abstraksi, Pewarisan (Inheritance), dan Polimorfisme.
- Arsitektur Java terdiri atas JDK (alat pengembang), JRE (pustaka runtime), dan JVM (mesin virtual eksekusi bytecode).
- Kompiler `javac` mengubah kode `.java` menjadi bytecode `.class` yang portabel di seluruh sistem operasi.
- Struktur program Java diawali deklarasi kelas dengan titik masuk eksekusi pada metode `public static void main(String[] args)`.

Soal Evaluasi Pemahaman BAB 1

1. Bandingkan kelebihan dan kelemahan antara paradigma pemrograman prosedural dengan pemrograman berorientasi objek dalam konteks pengembangan sistem informasi enterprise!
2. Jelaskan secara mendalam makna dan fungsi dari empat pilar utama OOP (Encapsulation, Abstraction, Inheritance, Polymorphism)!
3. Uraikan alur kerja kompilasi dan eksekusi program Java dari penulisan file sumber `.java` hingga dijalankan oleh Java Virtual Machine (JVM)!
4. Jelaskan makna setiap kata kunci pada deklarasi metode: `public static void main(String[] args)`!

BAB 2: KELAS, OBJEK, KONSTRUKTOR, DAN METODE

Capaian Pembelajaran Bab (Bahan Kajian RPS)

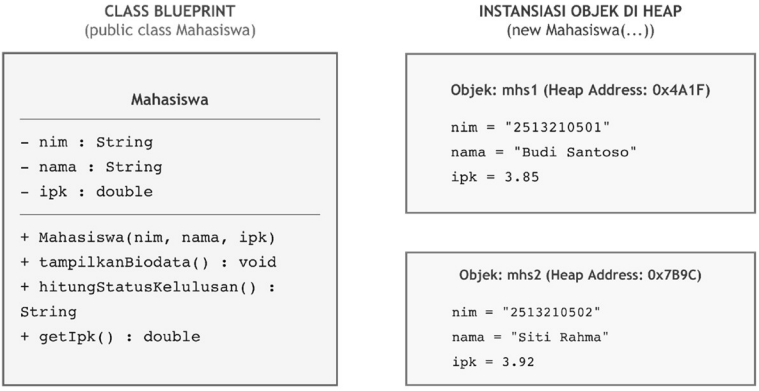
1. Mendefinisikan konsep Kelas (Class) sebagai cetak biru (blueprint) dan Objek (Object) sebagai instansiasi nyata di memori komputer.
2. Merancang struktur kelas lengkap yang memuat atribut (Fields/State) dan metode (Methods/Behavior).
3. Menganalisis mekanisme alokasi memori Stack (variabel referensi) dan memori Heap (objek instan) melalui operator 'new'.
4. Menerapkan konsep Konstruktor (Default, Berparameter, dan Overloading Konstruktor).
5. Menggunakan kata kunci 'this' untuk mengeliminasi ambiguitas penamaan parameter dan memanggil konstruktor internal (this()).
6. Menerapkan teknik Overloading Metode (Method Overloading) sebagai bentuk polimorfisme saat kompilasi.
7. Menyelesaikan penugasan resmi RPS: 'Tugas 2: Program Java Konsep PBO' (Bobot 15%).

2.1 Konsep Kelas (Blueprint) dan Objek (Instance)

Dalam Pemrograman Berorientasi Objek, Kelas (Class) adalah tipe data abstrak yang berfungsi sebagai cetak biru (blueprint), pola, atau template yang mendefinisikan variabel-variabel (atribut) dan fungsi-fungsi (metode) yang akan dimiliki oleh objek.

Objek (Object / Instance) adalah entitas konkret yang dibentuk di memori komputer berdasarkan cetak biru kelas tersebut. Satu kelas tunggal dapat digunakan untuk menghasilkan ratusan objek yang memiliki tipe struktur yang sama, namun menyimpan nilai data (State) yang saling independen satu sama lain.

Analogi Klasik: Kelas adalah rancangan arsitektur denah rumah di atas kertas, sedangkan Objek adalah rumah-rumah fisik nyata yang telah dibangun di komplek perumahan.



Gambar 2.1: Anatomi Class Blueprint dan Instansiasi Objek di Memori Heap

Dua Elemen Kunci Kelas

1. Atribut (State / Properties): Variabel yang merepresentasikan kondisi atau data yang disimpan objek (misal: nim, nama, ipk, saldo).
2. Metode (Behavior / Functions): Fungsi yang merepresentasikan tindakan atau aksi yang dapat dilakukan oleh objek (misal: hitungTagihan(), ubahStatus(), cetakKHS()).

2.2 Instansiasi Objek & Manajemen Memori (Stack vs. Heap)

Di bahasa Java, objek tidak diciptakan secara statis, melainkan dialokasikan secara dinamis pada saat runtime menggunakan operator `new`.

Manajemen Memori Java:

- Stack Memory: Menyimpan variabel lokal dan variabel referensi objek yang berisi alamat penunjuk (memory reference pointer).

- Heap Memory: Area memori tempat seluruh data objek fisik nyata beserta nilai atribut-atributnya dialokasikan.

Sintaks Instansiasi: [`Mahasiswa mhs1 = new Mahasiswa();`]

Penjelasan: [`mhs1` adalah variabel referensi yang disimpan di Stack, sedangkan `new Mahasiswa()` mengalokasikan ruang memori di Heap dan mengembalikan alamat referensinya ke `mhs1`.]

2.3 Konstruktor (Constructors) & Keyword 'this'

Konstruktor adalah metode khusus di dalam kelas yang otomatis dieksekusi tepat pada saat objek baru diinstansiasi dengan operator `new`. Konstruktor memiliki dua aturan baku: namanya harus SAMA PERSIS dengan nama kelas dan TIDAK MEMILIKI nilai balik (return type), bahkan tidak boleh menggunakan kata kunci `void`.

Jenis-Jenis Konstruktor:

1. Default Constructor: Konstruktor tanpa parameter yang otomatis disediakan kompiler Java jika pengembang tidak mendefinisikannya secara manual.
2. Parameterized Constructor: Konstruktor yang menerima argumen data untuk menginisialisasi nilai awal atribut objek secara langsung saat pembuatan.
3. Constructor Overloading: Mendefinisikan lebih dari satu konstruktor dalam satu kelas dengan jumlah atau tipe parameter yang berbeda.

Peran Kata Kunci `this`: Merujuk pada objek saat ini yang sedang aktif. Digunakan untuk mengatasi 'variable shadowing' (ketika nama

parameter metode sama dengan nama atribut kelas, misal: `this.nama = nama;`).

Contoh Konstruktor & Keyword 'this'

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


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


    public void tampilkanInfo() {
        System.out.println(this.nim + " - " + this.nama + " [IPK: "
+ this.ipk + "]");
  }
}
```

2.4 Overloading Metode (Method Overloading)

Method Overloading adalah kemampuan sebuah kelas untuk memiliki beberapa metode dengan NAMA YANG SAMA, asalkan memiliki DAFTAR PARAMETER (Method Signature) YANG BERBEDA (berbeda jumlah parameter, tipe data parameter, atau urutan tipe data).

Method Overloading merupakan bentuk Polimorfisme Waktu Kompilasi (Compile-Time / Static Polymorphism), di mana kompiler

Java secara otomatis menentukan metode mana yang akan dipanggil berdasarkan argumen yang dikirimkan saat pemanggilan fungsi.

Aturan Validitas Method Overloading

Mengubah tipe nilai balik (return type) saja TANPA mengubah parameter BUKAN merupakan overloading yang valid dan akan menimbulkan pesan kesalahan kompilasi (Compile Error)!

Aktivitas Praktikum & Penugasan: LEMBAR KERJA 2 (TUGAS 2 RPS): Program Java Konsep PBO

Kerjakan penugasan terstruktur 'Tugas 2: Program Java Konsep PBO' (Bobot 15%) dengan merancang kelas entitas sistem informasi lengkap dengan atribut, multiple constructor, keyword this, dan method overloading.

Bagian A: Perancangan Struktur Kelas Entitas

Komponen / Item Evaluasi	Deskripsi / Isian Kode Praktikum
1. Nama Kelas yang Dirancang:	Kelas Barang / Produk (Sistem Kasir Ritel)
2. Daftar Atribut (Fields) & Tipe Data:	kodeBarang (String), namaBarang (String), harga (double), stok (int)
3. Ragam Konstruktor yang Dibuat:	Konstruktor Default & Konstruktor Berparameter 4 Argumen
4. Implementasi Keyword 'this':	this.kodeBarang = kodeBarang; this.harga = harga;
5. Metode Overloading yang Dibuat:	hitungDiskon(double persen) dan hitungDiskon(double persen, double potonganMember)

Bagian B: Instansiasi & Pengujian Objek di Kelas Main

Komponen / Item Evaluasi	Deskripsi / Isian Kode Praktikum
Instansiasi Objek 1 (Nama & Nilai Awal):	Barang b1 = new Barang("B001", "Laptop ASUS Zenbook", 15000000, 10);
Instansiasi Objek 2 (Nama & Nilai Awal):	Barang b2 = new Barang("B002", "Mouse Wireless Logitech", 250000, 50);

Komponen / Item Evaluasi	Deskripsi / Isian Kode Praktikum
Pengujian Pemanggilan Metode:	b1.tampilkanDetail(); double diskon = b1.hitungDiskon(10);
Hasil Eksekusi Program di Console:	Menampilkan data barang dan nominal diskon secara presisi
Pengecekan Garbage Collection:	Objek referensi null dibersihkan otomatis oleh JVM Garbage Collector

Lembar Refleksi & Pengumpulan Portofolio:

1. Jelaskan perbedaan proses yang terjadi di memori Stack dan Heap saat baris kode `Mahasiswa mhs = new Mahasiswa();` dieksekusi oleh JVM:

.....

2. Mengapa penggunaan method overloading dapat meningkatkan keterbacaan dan kenyamanan pemanggilan fungsi dalam pemrograman API?

Rangkuman Eksekutif BAB 2

- Kelas (Class) adalah cetak biru abstrak, sedangkan Objek (Object) adalah instansiasi nyata di memori Heap.
- Variabel referensi disimpan di Stack dan menunjuk ke alamat alokasi objek fisik di memori Heap.
- Konstruktor bertugas menginisialisasi state objek dan memiliki nama yang identik dengan nama kelas tanpa tipe return.
- Kata kunci `this` merepresentasikan objek saat ini untuk mengatasi shadowing variabel dan menghubungkan konstruktor.
- Method Overloading memungkinkan beberapa metode bernama sama dengan parameter berbeda untuk polimorfisme statis.

Soal Evaluasi Pemahaman BAB 2

1. Uraikan perbedaan konseptual antara Class dan Object serta jelaskan bagaimana hubungan keduanya dalam arsitektur perangkat lunak!
2. Jelaskan mekanisme kerja alokasi memori Stack dan Heap pada saat operator `new` digunakan untuk menginstansiasi objek baru di Java!
3. Apa fungsi utama dari Konstruktor? Jelaskan apa yang terjadi jika seorang programmer tidak menuliskan konstruktor sama sekali pada sebuah kelas Java!
4. Jelaskan aturan-aturan yang harus dipenuhi agar suatu metode dapat dikatakan mengalami Method Overloading yang valid!

BAB 3: ENKAPSULASI, ABSTRAKSI, DAN MANAJEMEN HAK AKSES

Capaian Pembelajaran

1. Memahami prinsip dasar Enkapsulasi (Encapsulation) dan Sembunyi Informasi (Information Hiding).
2. Menganalisis tingkatan hak akses (Access Modifiers: private, default, protected, public) pada level atribut, metode, dan kelas.
3. Merancang metode Asesor (Getter) dan Mutator (Setter) yang dilengkapi dengan aturan validasi integritas data bisnis.
4. Menerapkan prinsip Abstraksi dalam memisahkan antarmuka publik dari kompleksitas komputasi internal kelas.
5. Menyusun kelas entitas yang aman (Robust & Secure Entity Class) untuk sistem informasi perbankan dan transaksi e-commerce.
1. 6. Menyelesaikan persiapan materi dan proyek portofolio Ujian Tengah Semester (UTS) (Bobot 25% kumulatif).

3.1 Konsep Enkapsulasi dan Sembunyi Informasi (Information Hiding)

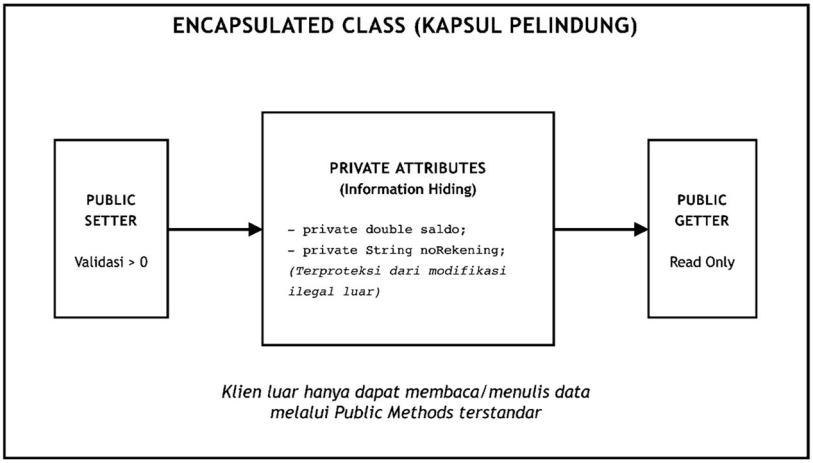
Enkapsulasi adalah proses pembungkusan data (atribut) dan perilaku (metode) yang beroperasi pada data tersebut ke dalam satu unit tunggal yang terproteksi (Class). Tujuan fundamental enkapsulasi adalah menyembunyikan detail representasi internal suatu objek dari modifikasi sembarangan oleh pihak luar konsep ini dikenal sebagai Information Hiding.

- Mengapa Data Tidak Boleh Dibiarkan Public?

Jika suatu atribut (seperti `public double saldo`) dideklarasikan public, kelas atau kode luar mana pun dapat mengubah nilainya secara ilegal tanpa melalui pemeriksaan (misal: mengisi nilai saldo

dengan angka negatif `-5000000` atau mengubah status tanpa otorisasi).

Dengan enkapsulasi, atribut dibuat `private` dan modifikasi nilai hanya dapat dilakukan melalui metode resmi (Setter) yang memiliki logika validasi ketat.



Gambar 3.1: Konsep Enkapsulasi, Information Hiding, dan Antarmuka Getter-Setter

Manfaat Utama Enkapsulasi

1. Kontrol Validasi Penuh: Menolak data tidak valid sebelum tersimpan ke memori.
2. Fleksibilitas Modifikasi Internal: Pengembang bebas merombak algoritma internal tanpa merusak kode klien pemanggil.
3. Read-Only / Write-Only Attributes: Objek dapat dibuat hanya bisa dibaca (hanya menyediakan Getter) atau hanya bisa ditulis.

3.2 Tingkatan Hak Akses (Access Modifiers) di Java

Java menyediakan empat tingkatan penentu hak akses (Access Modifiers) untuk mengontrol visibilitas elemen kelas:

1. `private`: Elemen hanya dapat diakses oleh kode di dalam KELAS YANG SAMA. Digunakan sebagai standar baku untuk seluruh atribut (Fields).

2. `default` (tanpa kata kunci / package-private): Elemen dapat diakses oleh seluruh kelas yang berada di dalam PACKAGE YANG SAMA.

3. `protected`: Elemen dapat diakses oleh kelas di dalam package yang sama DAN oleh kelas turunan (Subclasses) meskipun berada di package yang berbeda.

4. `public`: Elemen dapat diakses secara bebas oleh SELURUH KELAS di mana pun di dalam proyek aplikasi. Digunakan untuk antarmuka metode publik dan konstruktor.

Matriks Spektrum Visibilitas Access Modifiers

Modifier	Class Sama	Package Sama	Subclass Luar Package	Dunia Luar (Global)
private	YA	TIDAK	TIDAK	TIDAK
default (no modifier)	YA	YA	TIDAK	TIDAK
protected	YA	YA	YA	TIDAK
public	YA	YA	YA	YA

3.3 Perancangan Getter, Setter, dan Validasi Integritas

Pola standar desain kelas terenkapsulasi (JavaBeans Convention):

- Getter Method (Asesor): Metode publik yang mengembalikan nilai atribut privat (penamaan: `getNamaAtribut()` atau `isNamaAtribut()` untuk tipe boolean).
- Setter Method (Mutator): Metode publik yang menerima parameter untuk memperbarui nilai atribut privat dengan validasi (penamaan: `setNamaAtribut(tipe baru)`).

Contoh Enkapsulasi Kelas RekeningBank

```
public class RekeningBank {
    private String noRekening;
    private double saldo;

    public RekeningBank(String noRekening, double saldoAwal) {
        this.noRekening = noRekening;
        setSaldo(saldoAwal); // Validasi lewat setter
  }

    public double getSaldo() {
        return this.saldo;
  }

    public void setSaldo(double saldoBaru) {
        if (saldoBaru >= 0) {
            this.saldo = saldoBaru;
        } else {
            System.out.println("Error: Saldo tidak boleh negatif!");
    }
  }
}
```

3.4 Prinsip Abstraksi dalam Desain Sistem

Abstraksi berfokus pada pertanyaan 'Apa yang dilakukan oleh objek?' (What it does), bukan 'Bagaimana objek melakukannya secara teknis di dalam?' (How it does it).

Ketika seorang pengguna menekan tombol 'Tarik Tunai' pada mesin ATM, pengguna hanya peduli bahwa uang keluar dari mesin. Pengguna tidak perlu mengetahui kompleksitas query database, verifikasi protokol enkripsi hardware, atau mekanisme motor penghitung lembaran uang di dalam brankas.

Dalam pemrograman berorientasi objek, abstraksi diterapkan dengan mengekspos metode-metode publik yang intuitif dan menyembunyikan algoritma komputasi yang rumit di dalam metode internal privat.

Aktivitas Praktikum & Penugasan: LEMBAR KERJA 3 (PERSIAPAN UTS): Desain Kelas Terenkapsulasi & Validasi

Rancang kelas entitas bisnis yang menerapkan prinsip Enkapsulasi penuh, Sembunyi Informasi, dan Aturan Validasi Mutator untuk persiapan Ujian Tengah Semester (UTS - Bobot 25%).

Bagian A: Spesifikasi Atribut Privat & Aturan Validasi Bisnis

Komponen / Item Evaluasi	Deskripsi / Isian Kode Praktikum
1. Nama Kelas Entitas:	Kelas AkunPengguna (Sistem E-Commerce)
2. Atribut 1 (Tipe, Modifier, Validasi):	private String username; (Panjang minimal 6 karakter alfanumerik)
3. Atribut 2 (Tipe, Modifier, Validasi):	private String email; (Harus memuat karakter '@' dan '.')
4. Atribut 3 (Tipe, Modifier, Validasi):	private double saldoEwallet; (Tidak boleh bernilai < 0)
5. Status Akses Modifiers Atribut:	100% Private (Tidak ada atribut public/default)

Bagian B: Implementasi Logika Validasi Setter & Pengujian

Komponen / Item Evaluasi	Deskripsi / Isian Kode Praktikum
Logika Validasi setSaldoEwallet(double saldo):	if (saldo >= 0) { this.saldoEwallet = saldo; } else { throw IllegalArgumentException; }
Pengujian Kasus Positif (Input Valid):	setSaldoEwallet(500000) -> Nilai saldo sukses terupdate
Pengujian Kasus Negatif (Input Ilegal):	setSaldoEwallet(-200000) -> Sistem menolak dan memunculkan notifikasi error

Komponen / Item Evaluasi	Deskripsi / Isian Kode Praktikum
Penerapan Read-Only Attribute:	idPengguna hanya memiliki Getter, nilai diinisialisasi sekali di Konstruktor
Kesiapan Portofolio UTS:	Kode sumber dan diagram kelas UML siap diuji pada evaluasi UTS

Lembar Refleksi & Pengumpulan Portofolio:

1. Mengapa membiarkan atribut kelas berstatus 'public' dianggap sebagai pelanggaran fatal terhadap prinsip arsitektur rekayasa perangkat lunak OOP?

2. Jelaskan bagaimana penerapan enkapsulasi mempermudah pengujian unit (Unit Testing) dan pemeliharaan jangka panjang aplikasi!

Rangkuman Eksekutif BAB 3

- Enkapsulasi membungkus data dan metode ke dalam satu kelas serta menyembunyikan representasi data internal (Information Hiding).
- Access Modifiers di Java mengatur hak akses: `private` (kelas sama), `default` (package sama), `protected` (turunan), dan `public` (global).
- Atribut wajib berstatus `private` dan diakses melalui metode `Getter` (baca) dan `Setter` (tulis dengan validasi).
- Abstraksi menyederhanakan kompleksitas dengan hanya menampilkan antarmuka penting dan menyembunyikan detail teknis komputasi.
- Penguasaan enkapsulasi dan abstraksi menjadi penentu utama dalam evaluasi Ujian Tengah Semester (UTS).

Soal Evaluasi Pemahaman BAB 3

1. Uraikan definisi Enkapsulasi dan Information Hiding serta jelaskan risiko keamanan yang terjadi jika atribut kelas dideklarasikan sebagai `public`!

2. Bandingkan secara komprehensif spektrum visibilitas keempat Access Modifiers di Java (`private`, `default`, `protected`, `public`)!
3. Buatlah rancangan kode kelas Java `Karyawan` dengan atribut privat `gajiPokok` dan metode setter yang memastikan gaji tidak boleh kurang dari UMK (Rp 4.500.000)!
4. Jelaskan bagaimana konsep Abstraksi dan Enkapsulasi saling bekerja sama dalam menyusun arsitektur sistem informasi yang aman dan modular!

BAB 4: PEWARISAN (INHERITANCE) DAN POLIMORFISME (POLYMORPHISM)

Capaian Pembelajaran Bab (Bahan Kajian RPS)

1. Memahami prinsip Pewarisan (Inheritance) berbasis hubungan 'IS-A' antara Superclass dan Subclass.
2. Mengimplementasikan kata kunci 'extends' untuk mewariskan atribut dan metode tanpa duplikasi kode.
3. Menggunakan kata kunci 'super' untuk memanggil konstruktor induk dan mengeksekusi metode superclass.
4. Menerapkan Penimpaan Metode (Method Overriding) dengan anotasi '@Override' untuk menghasilkan perilaku polimorfik.
5. Menganalisis mekanisme Dynamic Method Dispatch (Polimorfisme Waktu Runtime) di mana referensi superclass menampung objek turunan.
6. Menerapkan mekanisme Object Casting dan pengecekan tipe menggunakan operator 'instanceof'.
7. Menyelesaikan penugasan resmi RPS: 'Tugas 3: Program Inheritance Java' (Bobot 15%).

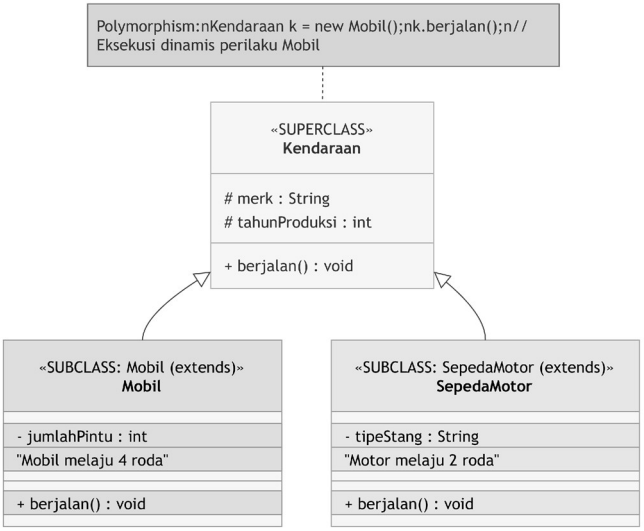
4.1 Konsep Pewarisan (Inheritance) & Hubungan 'IS-A'

Pewarisan (Inheritance) adalah mekanisme dalam OOP di mana suatu kelas baru (Subclass / Kelas Anak) diturunkan dari kelas yang sudah ada (Superclass / Kelas Induk). Subclass secara otomatis mewarisi seluruh atribut dan metode non-private milik superclass, serta dapat menambahkan atribut dan metode baru yang lebih spesifik.

Pewarisan memodelkan hubungan 'IS-A' (Adalah Suatu): Mobil IS-A Kendaraan, Dosen IS-A Pegawai, RekeningGiro IS-A RekeningBank.

Manfaat Utama Pewarisan:

- Penggunaan Ulang Kode (Code Reusability): Kode yang sudah teruji di superclass tidak perlu ditulis ulang di setiap subclass.
- Kemudahan Pemeliharaan: Perbaikan logika pada superclass secara otomatis terdistribusi ke seluruh subclass turunannya.
- Struktur Taksonomi Alami: Membentuk hierarki kelas yang logis dan rapi.



Gambar 4.1: Hierarki Pewarisan (Inheritance) dan Polimorfisme Dinamis (Method Overriding)

Sifat Pewarisan di Java (Single Inheritance)

Java menganut Single Inheritance untuk kelas: suatu subclass HANYA DAPAT mewarisi tepat SATU superclass secara langsung (`class Mobil extends Kendaraan`). Java tidak mendukung Multiple Inheritance antar-kelas untuk menghindari ambiguitas 'Diamond Problem'.

4.2 Penggunaan Kata Kunci 'super' & Rantai Konstruktor

Kata kunci `super` merujuk secara langsung pada superclass terdekat dari objek saat ini. Memiliki dua kegunaan utama:

1. Memanggil Konstruktor Superclass (`super(argumen)`): Harus menjadi pernyataan PERTAMA (First Statement) di dalam konstruktor subclass untuk memastikan inisialisasi state superclass selesai sebelum subclass dijalankan.
2. Mengakses Metode atau Atribut Superclass yang Tertimpa (`super.namaMetode()`): Digunakan ketika subclass ingin memanfaatkan kembali logika dasar milik superclass sebelum menambahkan logika tambahan.

4.3 Penimpaan Metode (Method Overriding) vs Overloading

Method Overriding terjadi ketika subclass mendefinisikan ulang metode yang SUDAH DIDEKLARASIKAN di superclass dengan NAMA, DAFTAR PARAMETER, dan TIPE RETURN YANG SAMA PERSIS.

Anotasi `@Override` wajib disematkan di atas deklarasi metode untuk menginstruksikan kompiler Java agar memvalidasi kesesuaian signature metode terhadap superclass.

Perbedaan Kunci: Overloading terjadi di dalam kelas yang sama dengan parameter berbeda (Polimorfisme Statis), sedangkan Overriding terjadi antar-kelas turunan dengan parameter yang identik (Polimorfisme Dinamis).

Contoh Implementasi Method Overriding

```
public class Pegawai {
    protected String nama;
```

```
    protected double gajiPokok;

    public Pegawai(String nama, double gaji) {
        this.nama = nama;
        this.gajiPokok = gaji;
  }

    public double hitungTotalGaji() {
        return this.gajiPokok;
  }
}

public class Manajer extends Pegawai {
    private double tunjanganJabatan;

    public Manajer(String nama, double gaji, double tunjangan) {
        super(nama, gaji); // Panggil konstruktor Pegawai
        this.tunjanganJabatan = tunjangan;
  }

  @Override
    public double hitungTotalGaji() {
        return super.hitungTotalGaji() + this.tunjanganJabatan;
  }
}
```

4.4 Polimorfisme Dinamis & Dynamic Method Dispatch

Polimorfisme (Banyak Bentuk) adalah kemampuan variabel referensi bertipe superclass untuk menampung berbagai objek konkret dari subclass-subclass turunannya:

`Pegawai p1 = new Manajer("Budi", 10000000, 5000000);`

`Pegawai p2 = new Staff("Siti", 5000000, 500000);`

Ketika `p1.hitungTotalGaji()` dipanggil, JVM saat runtime secara cerdas mengeksekusi metode `hitungTotalGaji()` milik kelas `Manajer`, bukan milik `Pegawai`. Mekanisme pemilihan metode berdasarkan tipe objek nyata saat runtime ini disebut sebagai Dynamic Method Dispatch.

Operator `instanceof` digunakan untuk memeriksa tipe objek nyata sebelum melakukan Type Casting (misal: `if (p1 instanceof Manajer) { Manajer m = (Manajer) p1; }`).

[Kata Kunci 'final'

- `final class`: Kelas tidak dapat diturunkan/diwarisi lagi oleh kelas mana pun (contoh: kelas `java.lang.String`).
- `final method`: Metode tidak dapat di-override oleh subclass.

Aktivitas Praktikum & Penugasan: LEMBAR KERJA 4 (TUGAS 3 RPS): Program Inheritance & Polimorfisme Java

Kerjakan penugasan kelompok 'Tugas 3: Program Inheritance' (Bobot 15%) dengan merancang hierarki pewarisan (Superclass & Subclass), pemanggilan super(), method overriding, dan dynamic polymorphism.

Bagian A: Perancangan Hierarki Pewarisan Kelas (UML Hierarchy)

Komponen / Item Evaluasi	Deskripsi / Isian Kode Praktikum
1. Superclass Induk (Atribut & Metode):	Kendaraan (platNomor, merk, warna | infoKendaraan(), hitungPajak())
2. Subclass 1 (Atribut & Overriding):	Mobil extends Kendaraan (kapasitasPenumpang | override hitungPajak())
3. Subclass 2 (Atribut & Overriding):	Truk extends Kendaraan (kapasitasMuatanTon | override hitungPajak())
4. Implementasi Konstruktor 'super()':	super(platNomor, merk, warna); diinisialisasi pada baris pertama konstruktor
5. Penggunaan Anotasi @Override:	Disematkan pada seluruh implementasi hitungPajak() di subclass

Bagian B: Pengujian Polimorfisme Array & Dynamic Dispatch

Komponen / Item Evaluasi	**Deskripsi / Isian Kode Praktikum**
Deklarasi Array Polimorfik:	Kendaraan[] daftar = new Kendaraan[3];

Komponen / Item Evaluasi	Deskripsi / Isian Kode Praktikum
Pengisian Objek Campuran:	daftar[0] = new Mobil(...); daftar[1] = new Truk(...); daftar[2] = new Mobil(...);
Perulangan Eksekusi Polimorfik:	for (Kendaraan k : daftar) { System.out.println(k.hitungPajak()); }
Penerapan Operator instanceof:	if (k instanceof Truk) { Truk t = (Truk) k; t.tampilkanKapasitasMuatan(); }
Status Kelulusan Uji Coba Program:	Output perhitungan pajak berhasil tampil spesifik sesuai tipe objek nyata

Lembar Refleksi & Pengumpulan Portofolio:

1. Jelaskan bagaimana penerapan polimorfisme mempermudah penambahan jenis subclass baru di masa depan tanpa perlu mengubah kode program utama:

.....

3. Mengapa bahasa Java melarang pewarisan jamak (Multiple Inheritance) antar kelas dan bagaimana solusi Java dalam mengatasi kebutuhan tersebut?

Rangkuman Eksekutif BAB 4

- Pewarisan (Inheritance) menghubungkan Superclass dan Subclass melalui kata kunci `extends` untuk memaksimalkan reusability.
- Kata kunci `super` digunakan untuk memanggil konstruktor superclass dan mengakses metode induk yang ditimpa.
- Method Overriding mendefinisikan ulang metode superclass pada subclass dengan signature yang identik (`@Override`).
- Polimorfisme dinamis (Dynamic Method Dispatch) mengeksekusi metode berdasarkan tipe objek riil di memori Heap saat runtime.
- Operator `instanceof` menjamin keamanan tipe data (Type Safety) sebelum proses downcasting objek dilakukan.

Soal Evaluasi Pemahaman BAB 4

1. Uraikan konsep Pewarisan (Inheritance) dan jelaskan hubungan 'IS-A' beserta contoh hierarki kelas dalam sistem akademik kampus!
2. Bandingkan perbedaan mendasar antara Method Overloading dengan Method Overriding dalam aspek waktu resolusi (Compile-Time vs Runtime), letak kelas, dan syarat signature!
3. Apa fungsi dari kata kunci `super` pada baris pertama konstruktor subclass? Apa yang terjadi jika programmer tidak menuliskan pemanggilan `super()` secara eksplisit?
4. Jelaskan bagaimana Dynamic Method Dispatch bekerja saat variabel referensi `Superclass` memanggil metode yang telah di-override oleh berbagai objek `Subclass`!

BAB 5: KELAS ABSTRAK DAN ANTARMUKA (INTERFACE)

Capaian Pembelajaran Bab

1. Memahami filosofi perancangan kelas abstrak (Abstract Class) dan metode abstrak (Abstract Method) menggunakan kata kunci 'abstract'.
2. Mendefinisikan Antarmuka (Interface) sebagai kontrak perilaku (Behavioral Contract) murni yang mengikat kelas implementor.
3. Mengimplementasikan kata kunci 'implements' dan menerapkan pewarisan antarmuka ganda (Multiple Interface Implementation).
4. Menganalisis matriks perbandingan komprehensif antara Abstract Class versus Interface dalam arsitektur perangkat lunak.
5. Menerapkan prinsip Loose Coupling dan Polimorfisme berbasis antarmuka dalam merancang modul pembayaran terintegrasi.
6. Menyelesaikan penugasan resmi RPS: 'Tugas 4: Program Java Menggunakan Kelas Abstrak dan Interface' (Bobot 10%).

5.1 Konsep Kelas Abstrak (Abstract Class) & Metode Abstrak

Dalam hierarki pewarisan yang kompleks, sering kali terdapat kelas induk konseptual yang terlalu umum untuk dapat diinstansiasi menjadi objek nyata secara langsung. Sebagai contoh: kelas `BentukGeometri` atau `Hewan`. Kita tidak pernah menemukan objek fisik yang sekadar berwujud 'Bentuk Geometri umum' tanpa bentuk spesifik (seperti Lingkaran, Persegi, atau Segitiga).

Kelas Abstrak (Abstract Class) adalah kelas yang dideklarasikan dengan kata kunci `abstract`, yang TIDAK DAPAT diinstansiasi secara langsung menggunakan operator `new`. Kelas abstrak hanya berfungsi sebagai superclass dasar yang menyediakan template struktur bagi subclass-subclass turunannya.

Metode Abstrak (Abstract Method): Metode yang hanya memiliki deklarasi tanda tangan (Method Signature) dan diakhiri titik koma `;`, TANPA memiliki blok tubuh implementasi `{ }`. Subclass konkret yang mewarisi kelas abstrak WAJIB meng-override dan mengimplementasikan seluruh metode abstrak tersebut.

Karakteristik Kelas Abstrak

• Boleh memiliki metode konkret (metode yang memiliki kode implementasi `{ }`) sekaligus metode abstrak.

• Boleh memiliki atribut (State), konstanta, konstruktor (untuk dipanggil via `super()`), dan berbagai access modifier (`protected`, `public`, `private`).

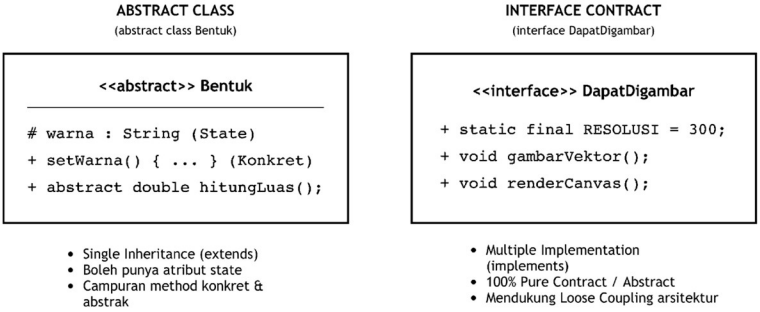
5.2 Konsep Antarmuka (Interface) Sebagai Kontrak Desain

Antarmuka (Interface) adalah cetak biru kelas yang hanya memuat deklarasi konstanta (`public static final`) dan metode-metode abstrak (`public abstract`). Interface bertindak sebagai Kontrak Perjanjian (Contractual Agreement): kelas mana pun yang menyatakan `implements` suatu interface berjanji secara mutlak untuk menyediakan implementasi nyata bagi seluruh metode yang tercantum dalam interface tersebut.

Mengapa Interface Sangat Krusial di Java?

1. Mengatasi Keterbatasan Single Inheritance: Sebuah kelas Java dapat mengimplementasikan BANYAK interface sekaligus (`class PembayaranDigital implements Transaksi, Notifikasi, AuditLog`).

2. Keterkaitan Lemah (Loose Coupling): Memisahkan definisi antarmuka dari implementasi teknisnya, sehingga modul perangkat lunak dapat diganti tanpa merusak modul lain.



Gambar 5.1: Desain Kontrak OOP: Abstract Class vs Interface Architecture

Contoh Implementasi Multiple Interfaces

```
public interface DapatDibayar {
    void prosesPembayaran(double jumlah);
}

public interface DapatDicetak {
    void cetakStruk();
}

public class PembayaranQris implements DapatDibayar, DapatDicetak {
  @Override
    public void prosesPembayaran(double jumlah) {
        System.out.println("Memproses QRIS sebesar Rp " + jumlah);
  }

  @Override
    public void cetakStruk() {
        System.out.println("Mencetak struk transaksi digital
QRIS...");
  }
}
```

5.3 Matriks Perbandingan: Abstract Class vs. Interface

Panduan Pemilihan Kapan Menggunakan Abstract Class vs Interface:

- Gunakan Abstract Class: Jika kelas-kelas turunan memiliki hubungan kekerabatan yang sangat erat (Strong 'IS-A' Relationship), membutuhkan pembagian atribut (State), konstruktor bersama, atau metode konkret dasar yang sama.
- Gunakan Interface: Jika ingin mendefinisikan kemampuan atau perilaku bersama (Behavioral Capability / 'CAN-DO') untuk kelas-kelas yang tidak saling berhubungan secara taksonomi (misal: kelas `Mobil` dan kelas `Dokumen` sama-sama dapat mengimplementasikan interface `DapatDieksporPDF`).

[PERHATIAN / BEST PRACTICE] Matriks Perbandingan Abstract Class vs Interface

Fitur / Karakteristik	Abstract Class	Interface
Keyword Deklarasi	`abstract class`	`interface`
Keyword Implementasi	`extends` (Single)	`implements` (Multiple)
Tipe Metode	Campuran (Abstrak & Konkret)	100% Abstrak (Kecuali default/static Java 8+)
Atribut / Fields	Boleh instance variables biasa	Wajib `public static final` (Konstanta)
Konstruktor	Boleh memiliki konstruktor	TIDAK BISA memiliki konstruktor
Fokus Perancangan	Identitas & Struktur Dasar (What it IS)	Perilaku & Kontrak (What it CAN DO)

5.4 Prinsip Loose Coupling & Polimorfisme Antarmuka

Dalam rekayasa sistem enterprise, arsitek perangkat lunak selalu menerapkan prinsip: 'Program to an interface, not an implementation'.

Dengan menggunakan tipe referensi Interface (`DapatDibayar metode = new PembayaranQris();` atau `DapatDibayar metode = new TransferBank();`), program utama dapat berganti penyedia pembayaran secara instan tanpa perlu mengubah satu baris pun logika di antarmuka kasir.

Aktivitas Praktikum & Penugasan: LEMBAR KERJA 5 (TUGAS 4 RPS): Program Abstrak & Interface Java

Kerjakan penugasan kelompok 'Tugas 4: Program Abstrak & Interface' (Bobot 10%) dengan membangun modul transaksi terintegrasi yang memadukan Abstract Class dan Multiple Interfaces.

Bagian A: Perancangan Kelas Abstrak & Antarmuka Kontrak

Komponen / Item Evaluasi	Deskripsi / Isian Kode Praktikum
1. Nama Kelas Abstrak (Induk):	abstract class KomponenSistem (id, nama, status | abstract void inisialisasi())
2. Nama Interface 1 & Metode:	interface DapatDiaudit (void catatLogAktivitas(String aksi))
3. Nama Interface 2 & Metode:	interface DapatDireset (void resetKeDefault())
4. Kelas Konkret Implementor:	class ModulAutentikasi extends KomponenSistem implements DapatDiaudit, DapatDireset
5. Implementasi Seluruh Metode Abstrak:	100% metode ter-override dengan logika sistem yang valid

Bagian B: Pengujian Polimorfisme Berbasis Interface

Komponen / Item Evaluasi	Deskripsi / Isian Kode Praktikum
Deklarasi Referensi Interface:	DapatDiaudit auditor = new ModulAutentikasi("AUTH-01", "Login SSO");
Pemanggilan Metode Kontrak:	auditor.catatLogAktivitas("User admin berhasil login pada 12:45 WIB");

Komponen / Item Evaluasi	Deskripsi / Isian Kode Praktikum
Uji Multiple Implementation:	DapatDireset resetter = (DapatDireset) auditor; resetter.resetKeDefault();
Verifikasi Loose Coupling:	Modul audit dapat menangani objek modul transaksi lain yang ber-interface sama
Status Kelulusan Program:	Program lolos uji coba kompilasi dan eksekusi tanpa kesalahan runtime

Lembar Refleksi & Pengumpulan Portofolio:

1. Mengapa prinsip 'Program to an Interface, not an Implementation' menjadi pedoman utama dalam merancang sistem informasi modern berskala enterprise?

2. Jelaskan perbedaan mendasar penggunaan kata kunci 'extends' dengan 'implements' pada kelas Java!

Rangkuman Eksekutif BAB 5

- Kelas Abstrak (`abstract class`) adalah kelas template induk yang tidak dapat diinstansiasi langsung dan dapat memuat metode abstrak.
- Antarmuka (`interface`) adalah kontrak murni yang mewajibkan kelas implementor (`implements`) menyediakan logika seluruh metodenya.
- Java mendukung pewarisan antarmuka ganda (Multiple Interface Implementation) untuk memperkaya fungsionalitas kelas.
- Gunakan Abstract Class untuk hubungan kekeluargaan erat ('IS-A') dan Interface untuk kontrak kemampuan bersama ('CAN-DO').
- Prinsip Loose Coupling berbasis interface mempermudah pengembangan sistem yang modular dan tahan terhadap perubahan.

Soal Evaluasi Pemahaman BAB 5

1. Uraikan definisi dan karakteristik Kelas Abstrak (Abstract Class) serta jelaskan mengapa sebuah metode abstrak tidak boleh memiliki blok kode tubuh `{ }`!
2. Jelaskan konsep Antarmuka (Interface) sebagai kontrak formal dalam rekayasa perangkat lunak dan bagaimana interface memecahkan keterbatasan Single Inheritance di Java!
3. Susun tabel komparasi detail antara Abstract Class vs Interface ditinjau dari aspek keyword, konstruktor, tipe atribut, dan tujuan desainnya!
4. Jelaskan makna dari prinsip arsitektur perangkat lunak: 'Program to an interface, not an implementation' beserta contoh penerapannya di industri e-commerce!

BAB 6: PEMROGRAMAN ANTARMUKA GRAFIS (GUI) MENGGUNAKAN JAVA

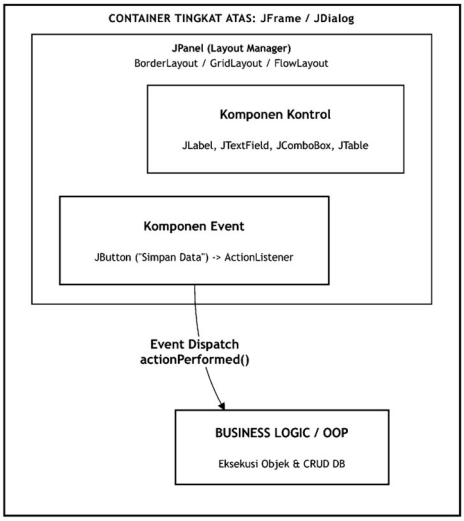
Capaian Pembelajaran Bab (Bahan Kajian RPS)

1. Memahami arsitektur pemrograman berbasis antarmuka grafis (Graphical User Interface / GUI) dan pustaka Java Swing.
2. Menguasai hierarki komponen GUI: Top-Level Container (JFrame), Intermediate Container (JPanel), dan Atomic Controls (JButton, JTextField, JTable).
3. Menerapkan berbagai manajer tata letak visual (Layout Managers: BorderLayout, FlowLayout, GridLayout, BoxLayout).
4. Mengimplementasikan model pemrograman berbasis peristiwa (Event-Driven Programming) menggunakan ActionListener dan Lambda Expressions.
5. Mengintegrasikan prinsip-prinsip OOP (Class, Encapsulation, Inheritance, Polymorphism) ke dalam aplikasi GUI berbasis pola MVC.
6. Menyelesaikan penugasan resmi RPS: 'Pembuatan Program GUI Menggunakan Java' (Bobot 20%) dan portofolio proyek akhir evaluasi UAS (Bobot 25% kumulatif).

6.1 Konsep Dasar Antarmuka Grafis & Pustaka Java Swing

Program berbasis konsol (Command Line Interface / CLI) berinteraksi dengan pengguna melalui input teks hitam-putih yang kaku. Sebagian besar aplikasi sistem informasi modern di dunia industri menuntut antarmuka yang intuitif, menarik, dan interaktif berupa Graphical User Interface (GUI).

Java menyediakan pustaka grafis standar `javax.swing` (Java Swing) yang dibangun di atas fondasi Abstract Window Toolkit (AWT). Keunggulan Java Swing adalah bersifat 'Lightweight' (ditulis murni dalam bahasa Java tanpa bergantung langsung pada komponen antarmuka sistem operasi lokal) dan mendukung konsep 'Pluggable Look and Feel' yang konsisten di berbagai platform.



Gambar 6.1: Arsitektur Komponen Java GUI Swing dan Model Pemrograman Berbasis Peristiwa

Hierarki Tiga Lapis Komponen GUI Swing

1. Top-Level Containers: Jendela utama aplikasi (`JFrame`, `JDialog`).
2. Intermediate Containers: Panel pengelompok wadah komponen (`JPanel`, `JScrollPane`, `JTabbedPane`).
3. Atomic Components: Kontrol interaktif individual (`JLabel`, `JTextField`, `JButton`, `JComboBox`, `JTable`, `JCheckBox`).

6.2 Manajemen Tata Letak Visual (Layout Managers)

Di lingkungan Java GUI, penempatan posisi dan ukuran tombol tidak disarankan menggunakan koordinat piksel absolut (Absolute Positioning), karena tata letak akan berantakan ketika ukuran jendela diperbesar atau dijalankan pada resolusi layar berbeda. Pengembang wajib menggunakan Manajer Tata Letak (Layout Managers):

- `BorderLayout`: Membagi panel menjadi 5 area kardinal: `NORTH`, `SOUTH`, `EAST`, `WEST`, dan `CENTER` (area tengah akan otomatis membesar memenuhi ruang tersisa). Menjadi layout bawaan `JFrame`.
- `FlowLayout`: Menyusun komponen secara berurutan mengalir dari kiri ke kanan. Jika baris penuh, otomatis pindah ke baris di bawahnya. Menjadi layout bawaan `JPanel`.
- `GridLayout`: Menyusun komponen ke dalam bentuk matriks kisi-kisi (Grid) dengan jumlah baris dan kolom berukuran seragam.
- `BoxLayout`: Menyusun komponen dalam satu baris horizontal (`X_AXIS`) atau satu kolom vertikal (`Y_AXIS`).

6.3 Pemrograman Berbasis Peristiwa (Event-Driven Programming)

Aplikasi GUI tidak berjalan secara linier dari atas ke bawah, melainkan menggunakan paradigma Pemrograman Berbasis Peristiwa (Event-Driven Programming). Program berada dalam status bersiaga (Idle State) menunggu aksi/kejadian yang dipicu oleh pengguna (seperti klik mouse pada tombol, penekanan tombol keyboard, atau pemilihan item dropdown).

Tiga Komponen Arsitektur Event Handling:

1. Event Source (Sumber Peristiwa): Komponen GUI tempat aksi terjadi (misal: objek `JButton btnSimpan`).
2. Event Object (Objek Peristiwa): Data yang dibangkitkan saat aksi terjadi (misal: objek `ActionEvent`).

3. Event Listener (Pendengar Peristiwa): Antarmuka yang bertugas menangkap peristiwa dan mengeksekusi metode penangan (misal: `ActionListener` dengan metode `actionPerformed()`).

Contoh Event Handling dengan Lambda Expression

```
JButton btnHitung = new JButton("Hitung Nilai Akhir");


// Menggunakan Lambda Expression (Java 8+)
btnHitung.addActionListener(e -> {
    double tugas = Double.parseDouble(txtTugas.getText());
    double uts = Double.parseDouble(txtUts.getText());
    double uas = Double.parseDouble(txtUas.getText());
    double nilaiAkhir = (0.3 * tugas) + (0.35 * uts) + (0.35 * uas);
    lblHasil.setText("Nilai Akhir: " + String.format("%.2f",
nilaiAkhir));
});
```

6.4 Integrasi OOP GUI: Pola Desain Model-View-Controller (MVC)

Untuk menjaga kode program GUI tetap bersih dan terstruktur, pengembang menerapkan pola arsitektur MVC sederhana:

- Model: Kelas entitas data murni yang menerapkan enkapsulasi (misal: `Mahasiswa.java`).
- View: Kelas jendela grafis yang merancang formulir dan tabel (misal: `FormMahasiswaView.java` berbasis `JFrame`).
- Controller: Kelas penghubung logika bisnis yang menangani event tombol dan memanipulasi data tabel (misal: `MahasiswaController.java`).

Thread Safety di Java GUI

Seluruh pembuatan dan pembaruan komponen GUI Swing wajib dieksekusi di dalam Event Dispatch Thread (EDT) menggunakan: `SwingUtilities.invokeLater(() -> { new FormUtama().setVisible(true); });`

Aktivitas Praktikum & Penugasan: LEMBAR KERJA 6 (TUGAS 5 & UAS): Pembuatan Program Aplikasi Java GUI Terpadu

Kerjakan penugasan terpadu 'Tugas 5 & UAS: Pembuatan Program Aplikasi GUI Java' (Bobot 25% kumulatif) dengan merancang aplikasi desktop interaktif berbasis Swing yang mengintegrasikan seluruh pilar OOP.

Bagian A: Spesifikasi Form Antarmuka Grafis (View Layout)

Komponen / Item Evaluasi	Deskripsi / Isian Kode Praktikum
1. Judul Aplikasi GUI Desktop:	Sistem Informasi Kasir & Inventaris Barang (SiKasir ITATS)
2. Komponen Input yang Digunakan:	JTextField (Kode, Nama, Harga), JSpinner (Jumlah), JComboBox (Kategori)
3. Komponen Output & Tabel Data:	JTable (DefaultTableModel untuk menampilkan daftar belanja real-time)
4. Layout Manager yang Diterapkan:	BorderLayout utama, JPanel formulir GridLayout(4,2), JPanel tombol FlowLayout
5. Tombol Aksi (JButton Events):	btnTambah (Hitung & Masuk Tabel), btnHapus (Hapus Baris), btnSelesai (Cetak Struk)

Bagian B: Integrasi Pilar OOP & Pengujian Program

Komponen / Item Evaluasi	Deskripsi / Isian Kode Praktikum
Implementasi Class & Encapsulation:	Kelas Model Barang terenkapsulasi penuh dengan validasi harga > 0

Komponen / Item Evaluasi	Deskripsi / Isian Kode Praktikum
Implementasi Inheritance & Polimorfisme:	Hierarki ProdukFisik dan ProdukDigital dengan perbedaan perhitungan pajak
Penerapan Event Handling:	ActionListener terpasang pada tombol dengan validasi form kosong
Pengujian Error Handling (Try-Catch):	NumberFormatException ditangani dengan pesan JOptionPane dialog yang ramah
Kesiapan Presentasi Proyek Akhir UAS:	Aplikasi berjalan mulus, kode bersih, siap didemonstrasikan pada minggu ke-16

Lembar Refleksi & Pengumpulan Portofolio:

1. Jelaskan bagaimana penerapan prinsip modularitas OOP dan pola MVC mempermudah proses modifikasi antarmuka GUI tanpa mengganggu logika bisnis di baliknya:
2. Susun berkas kode sumber (.java), file eksekusi (.jar), serta dokumentasi petunjuk pengguna (User Manual) kelompok Anda untuk evaluasi penentuan kelulusan UAS!

Rangkuman Eksekutif BAB 6

- Java Swing menyediakan pustaka komponen antarmuka grafis yang bersifat lightweight dan portabel lintas platform.
- Hierarki GUI tersusun atas Top-Level Containers (JFrame), Intermediate Containers (JPanel), dan Atomic Components (JButton, JTextField, JTable).
- Layout Managers (BorderLayout, FlowLayout, GridLayout) menjamin tampilan tetap proporsional pada berbagai resolusi layar.
- Pemrograman berbasis peristiwa (Event-Driven) menggunakan ActionListener untuk merespons aksi pengguna secara asinkron.
- Pola MVC memisahkan komponen antarmuka visual dari kelas entitas model data untuk menghasilkan kode yang bersih dan profesional.

Soal Evaluasi Pemahaman BAB 6

1. Uraikan hierarki tiga lapis komponen antarmuka grafis pada pustaka Java Swing (Top-Level, Intermediate, Atomic) dan berikan masing-masing 2 contoh kelasnya!
2. Jelaskan fungsi dari Layout Managers di Java dan bandingkan mekanisme penataan visual antara `BorderLayout`, `FlowLayout`, dan `GridLayout`!
3. Uraikan arsitektur Event-Driven Programming di Java yang melibatkan Event Source, Event Object, dan Event Listener dalam skenario penekanan tombol simpan!
4. Bagaimana cara mengintegrasikan kelas entitas model bisnis berbasis OOP dengan komponen `JTable` menggunakan `DefaultTableModel` pada aplikasi Java GUI?

GLOSARIUM ISTILAH PEMROGRAMAN BERORIENTASI OBJEK

Abstract Class	Kelas yang dideklarasikan dengan kata kunci 'abstract', tidak dapat diinstansiasi langsung, dan berfungsi sebagai superclass template yang dapat memuat metode abstrak dan konkret.
Abstract Method	Metode yang hanya memuat tanda tangan deklarasi tanpa blok kode implementasi `{ }`, yang wajib di-override oleh subclass konkret turunan.
Abstraction	Prinsip menyembunyikan rincian komputasi teknis internal dan hanya mengekspos fungsi-fungsi esensial kepada pengguna/klien.
Access Modifier	Kata kunci pengendali visibilitas elemen kelas di Java, mencakup `private`, `default` (package-private), `protected`, dan `public`.
ActionListener	Antarmuka pendengar peristiwa pada pustaka Java Swing yang bertugas menangkap aksi penekanan tombol (`JButton`) atau penekanan Enter pada formulir.
Attribute (Field/State)	Variabel data yang dideklarasikan di dalam kelas yang merepresentasikan karakteristik atau status yang disimpan oleh objek.
Bytecode	Kode instruksi tingkat menengah hasil kompilasi program Java (`.class`) yang bersifat independen terhadap platform dan dieksekusi oleh JVM.
Class (Kelas)	Cetak biru (blueprint) atau template tipe data terdefinisi yang memuat atribut dan metode untuk menghasilkan objek.
Constructor (Konstruktor)	Metode khusus di dalam kelas yang otomatis dipanggil saat pembentukan objek baru (`new`) untuk menginisialisasi atribut.
Dynamic Method Dispatch	Mekanisme runtime di mana pemanggilan metode polimorfik diputuskan berdasarkan tipe objek riil di memori Heap, bukan tipe variabel referensinya.

Encapsulation (Enkapsulasi)	Pembungkusan data dan metode ke dalam satu kelas serta pembatasan akses langsung terhadap data melalui metode Getter dan Setter (Information Hiding).
Event-Driven Programming	Paradigma pemrograman di mana alur eksekusi aplikasi dikendalikan oleh peristiwa eksternal (klik mouse, ketukan keyboard, timer).
Getter (Asesor)	Metode publik yang digunakan untuk membaca nilai atribut privat sebuah objek secara aman.
Inheritance (Pewarisan)	Mekanisme penurunan karakteristik atribut dan metode dari Superclass ke Subclass menggunakan kata kunci `extends`.
Instance of (instanceof)	Operator biner Java untuk memeriksa apakah suatu objek merupakan instansi dari suatu kelas tertentu sebelum proses Type Casting dilakukan.
Instantiation (Instansiasi)	Proses pembuatan objek fisik di memori Heap berdasarkan cetak biru kelas menggunakan operator `new`.
Interface (Antarmuka)	Tipe referensi di Java yang merupakan kontrak murni, hanya berisi deklarasi metode abstrak dan konstanta, diimplementasikan dengan kata kunci `implements`.
Java Virtual Machine (JVM)	Mesin virtual abstrak yang menginterpretasikan dan mengeksekusi instruksi Java Bytecode ke dalam kode mesin lokal sistem operasi target.
JFrame	Wadah tingkat atas (Top-Level Container) pada pustaka Java Swing yang merepresentasikan jendela aplikasi desktop utama dengan judul dan tombol kontrol jendela.
JPanel	Wadah perantara (Intermediate Container) pada Java Swing yang digunakan untuk mengelompokkan beberapa komponen visual dengan manajer tata letak tertentu.
Layout Manager	Objek pengatur tata letak visual komponen GUI (BorderLayout, FlowLayout, GridLayout, BoxLayout) agar adaptif terhadap perubahan ukuran jendela.

Loose Coupling	Prinsip desain rekayasa perangkat lunak di mana modul-modul sistem memiliki ketergantungan minimal satu sama lain, biasanya dicapai melalui Interface.
Method (Metode)	Fungsi atau blok kode di dalam kelas yang merepresentasikan perilaku, aksi, atau operasi kalkulasi yang dapat dijalankan oleh objek.
Method Overloading	Mendefinisikan beberapa metode bernama sama di dalam satu kelas dengan daftar parameter (signature) yang berbeda (Polimorfisme Statis).
Method Overriding	Mendefinisikan ulang metode superclass di dalam subclass dengan signature yang identik menggunakan anotasi `@Override` (Polimorfisme Dinamis).
Model-View-Controller (MVC)	Pola arsitektur perangkat lunak yang memisahkan logika model data (Model), antarmuka grafis (View), dan pengendali aksi (Controller).
Object (Objek)	Instansi konkret dari suatu kelas yang memiliki state (nilai data) dan behavior (kemampuan metode) yang menempati ruang di memori Heap.
Polymorphism (Polimorfisme)	Prinsip 'satu antarmuka, banyak bentuk' yang memungkinkan objek dari berbagai subclass merespons pemanggilan metode yang sama dengan cara yang berbeda.
Setter (Mutator)	Metode publik yang digunakan untuk mengubah nilai atribut privat sebuah objek dengan menyertakan aturan validasi integritas.
Superclass (Kelas Induk)	Kelas yang menjadi sumber pewarisan atribut dan metode bagi kelas-kelas turunan di bawahnya.
This Keyword	Kata kunci penunjuk ke objek saat ini yang sedang aktif di memori untuk mengatasi ambiguitas parameter dan merantai konstruktor.

RUBRIK PENILAIAN & MATRIKS KORELASI TUGAS RPS (BOBOT TOTAL 100%)

Evaluasi pembelajaran Mata Kuliah Pemrograman Berorientasi Objek (Kode MK: 25132105) mengacu pada matriks penugasan resmi RPS Kurikulum 2021:

1. Matriks Evaluasi Tugas RPS & Portofolio (Bobot Total 100%)

Komponen Penilaian RPS	Bobot	Indikator Penilaian & Artefak Kode
1. Tugas 1: Program Awal Java (SubCPMK 1 | Minggu 1-2)	10%	Keterampilan mengonfigurasi JDK/IDE, menyusun sintaks dasar Java, memahami struktur main(), dan eksekusi kompilasi Bytecode bebas error.
2. Tugas 2: Program Java Konsep PBO (SubCPMK 2 | Minggu 3-4)	15%	Kemampuan merancang Class, mendefinisikan atribut/metode, menerapkan multi-konstruktor, keyword 'this', instansiasi Objek di Heap, dan Method Overloading.
3. Ujian Tengah Semester (UTS) (SubCPMK 2 & 3 | Minggu 8)	25%	Evaluasi praktik mendalam penerapan Enkapsulasi, Access Modifiers (private, protected, public), perancangan Getter/Setter bervalidasi, dan prinsip Abstraksi.
4. Tugas 3: Program Inheritance Java (SubCPMK 4 | Minggu 9-10)	15%	Penguasaan hierarki pewarisan (Superclass/Subclass, extends), pemanggilan super(), Penimpaan Metode (Method Overriding), dan Dynamic Method Dispatch.

Komponen Penilaian RPS	Bobot	Indikator Penilaian & Artefak Kode
5. Tugas 4: Program Abstrak & Interface (SubCPMK 4 & 5 | Minggu 11-12)	10%	Keterampilan mendesain arsitektur kontrak sistem menggunakan Abstract Class dan Multiple Interfaces (implements) untuk menghasilkan Loose Coupling.
6. Tugas 5 & UAS: Proyek Aplikasi GUI Java (SubCPMK 5 & 6 | Minggu 13-16)	25%	Kualitas integrasi seluruh pilar OOP ke dalam aplikasi antarmuka grafis (Java Swing: JFrame, JPanel, JTable, Layouts, ActionListener Event-Driven, Pola MVC).

2. Skala Konversi Nilai & Standar Kualitas Perangkat Lunak

Rentang Nilai & Huruf	Deskripsi Kualitatif Capaian
Sangat Baik (A: 85 - 100)	Arsitektur kode sangat bersih, modular, mematuhi seluruh kaidah OOP (Encapsulation, Inheritance, Polymorphism, Interface), penanganan error exception lengkap, antarmuka GUI interaktif dan responsif.
Baik (B: 70 - 84)	Kode program berjalan benar, struktur class dan objek tepat, enkapsulasi diterapkan, pewarisan dan antarmuka berfungsi, penanganan event GUI berjalan baik.
Cukup (C: 55 - 69)	Program dapat dikompilasi namun masih terdapat atribut yang tidak terenkapsulasi, atau logika polimorfisme/GUI masih mengalami galat minor.
Kurang (D/E: < 55)	Gagal mengompilasi program, tidak memahami paradigma berorientasi objek, tugas tidak dikumpulkan sesuai spesifikasi RPS.

DAFTAR PUSTAKA UTAMA SESUAI RPS RESMI

Horstmann, C. S., & Cornell, G. Core Java 2: Volume 1 - Fundamentals. Upper Saddle River, NJ: Prentice Hall. [Pustaka Utama RPS]

Riley, D. D. (2002). The Object of JAVA: Introduction to Programming Using Software Engineering Principles. Boston: Addison-Wesley. [Pustaka Utama RPS]

Balagurusamy, E. (1995). Object-Oriented Programming with C++. New Delhi: Tata McGraw-Hill Publishing Company Limited. [Pustaka Utama RPS]

Kadir, A. (2004). Dasar Pemrograman Java 2. Yogyakarta: CV Andi Offset. [Pustaka Utama RPS]

Purnama, R. Tuntunan Pemrograman Java. Bandung: Informatika. [Pustaka Utama RPS]

Deitel, P., & Deitel, H. (2018). Java How to Program: Early Objects (11th ed.). Boston: Pearson Education.

Bloch, J. (2018). Effective Java (3rd ed.). Boston: Addison-Wesley Professional.

Oracle Corporation. (2024). Java SE Documentation & Java Language Specification (JLS). Oracle Java Platform Standard Edition.

PROFIL DOSEN PENGEMBANG RPS & PENGAMPU

Budanis Dwi Meilani, S.T., M.Kom.

Dosen Pengembang Rencana Pembelajaran Semester (RPS), Koordinator Rumpun Mata Kuliah Sistem Informasi, dan Ketua Program Studi S1 Sistem Informasi, Fakultas Teknik Elektro dan Teknologi Informasi, Institut Teknologi Adhi Tama Surabaya (ITATS).

Resa Uttungga, S.Kom., M.Kom.

Dosen Pengampu Mata Kuliah Pemrograman Berorientasi Objek dan Rekayasa Perangkat Lunak pada Program Studi S1 Sistem Informasi, Fakultas Teknik Elektro dan Teknologi Informasi, Institut Teknologi Adhi Tama Surabaya (ITATS).

PANDUAN LENGKAP PEMROGRAMAN BERORIENTASI OBJEK BERBASIS OBE & JAVA PRAKTIKUM

Buku modul praktikum ini menyajikan panduan terstruktur penguasaan paradigma Object-Oriented Programming (OOP) menggunakan Java SE, mulai dari konsep kelas dan alokasi objek di memori, 4 pilar fundamental (enkapsulasi, abstraksi, pewarisan, dan polimorfisme), desain kontrak interface, hingga pembuatan aplikasi desktop interaktif berbasis Java GUI Swing dan pola arsitektur MVC.

- Konsep Dasar OOP & Ekosistem Java (JDK, JRE, JVM)
- Kelas, Objek, Konstruktor & Method Overloading
- Enkapsulasi, Hak Akses & Information Hiding
- Pewarisan (Inheritance) & Polimorfisme Dinamis
- Kelas Abstrak & Antarmuka (Interface Contract)
- Pemrograman GUI Desktop (Java Swing) & Pola MVC

Software Engineer • Java Backend Developer • Desktop App Developer • System Programmer

Diterbitkan oleh Program Studi S1 Sistem Informasi ITATS