LAPORAN UTS OBJECT BASED PROGRAMMING

Disusun oleh:

Darren Evan N. (825240062)

Leticia Michelle P. (825240140)

Dosen Pembimbing:

Lely Hiryanto, S.Kom., M.Sc., Ph.D.

Fakultas Teknologi Informasi

Universitas Tarumanagara

2025/2026

KATA PENGANTAR

Puji syukur ke hadirat Tuhan Yang Maha Esa atas rahmat dan karunia-Nya sehingga kami, tim penyusun, dapat menyelesaikan "Laporan Ujian Tengah Semester Object Based Programming" ini dengan baik dan tepat waktu. Laporan ini disusun sebagai pemenuhan tugas akademis dan dokumentasi atas proyek perancangan dan implementasi

Sistem Informasi Ekspedisi Barang menggunakan bahasa pemrograman Java. Dalam laporan ini, dibahas secara rinci proses pengembangan sistem, mulai dari analisis kebutuhan, perancangan sistem menggunaka *Class Diagram*, hingga implementasi fitur-fitur utama. Tujuan utama dari pengembangan program ini adalah untuk membangun sebuah sistem yang mampu melakukan kalkulasi biaya pengiriman untuk tiga jenis layanan yang berbeda, yaitu Reguler, Kilat, dan Kargo. Sistem ini dirancang untuk meminimalisir potensi kesalahan perhitungan manual sekaligus menyediakan platform terpusat untuk standardisasi harga.

Penyusunan laporan ini tidak akan berjalan lancar tanpa bimbingan dan dukungan dari berbagai pihak. Oleh karena itu, kami ingin mengucapkan terima kasih yang sebesar-besarnya kepada dosen pembimbing kami, Ibu Lely Hiryanto, S.Kom., M.Sc., Ph.D., yang telah memberikan arahan, masukan, dan ilmu yang sangat berharga selama proses pengerjaan proyek dan penyusunan laporan ini. Ucapan terima kasih juga kami sampaikan kepada seluruh pihak yang telah memberikan dukungan moril dan materiel yang tidak dapat kami sebutkan satu per satu.

Kami menyadari bahwa laporan ini masih memiliki banyak kekurangan. Oleh karena itu, kami sangat mengharapkan kritik dan saran yang membangun dari para pembaca demi penyempurnaan di masa mendatang. Semoga laporan ini dapat memberikan manfaat dan menjadi fondasi untuk pengembangan sistem manajemen ekspedisi yang lebih besar dan komprehensif.

DAFTAR ISI

BAB I PENDAHULUAN

1.1 Latar Belakang

Sektor logistik di Indonesia tengah mengalami fase pertumbuhan yang belum pernah terjadi sebelumnya, sebuah transformasi yang secara fundamental didorong oleh adopsi *e-commerce* secara masif oleh masyarakat. Analisis pasar menunjukkan bahwa ukuran pasar logistik *e-commerce* di Indonesia diproyeksikan akan mencapai USD 7.93 miliar pada tahun 2030, dengan laju pertumbuhan tahunan majemuk (*Compound Annual Growth Rate*, CAGR) sebesar 8.52% selama periode 2025 hingga 2030. Pertumbuhan pesat ini ditopang oleh dua pilar utama: peningkatan penetrasi internet yang semakin merata dan kekuatan konsumsi dari populasi kelas menengah yang signifikan. Pada tahun 2019, tercatat ada 57,3 juta warga kelas menengah di Indonesia, yang berkontribusi terhadap 43,3% dari total konsumsi nasional. Kehadiran platform [1].

Lonjakan permintaan ini tidak seragam; konsumen modern memiliki ekspektasi yang beragam, menciptakan sebuah spektrum kebutuhan yang harus dipenuhi oleh penyedia jasa ekspedisi. Preferensi pelanggan sering kali merupakan pertukaran antara kecepatan pengiriman dan biaya yang harus dikeluarkan. Untuk menjawab dinamika pasar ini, perusahaan ekspedisi modern menawarkan berbagai jenis layanan yang memungkinkan pelanggan memilih opsi yang paling sesuai dengan kebutuhan dan anggaran mereka. Opsi-opsi ini umumnya mencakup [2]:

1. Kiriman Reguler: Pilihan paling ekonomis dengan estimasi waktu pengiriman standar, biasanya antara 3-4 hari, cocok untuk pengiriman yang tidak mendesak.
2. Kiriman Kilat: Layanan premium yang memprioritaskan kecepatan, dengan estimasi waktu pengiriman 1-2 hari, ideal untuk pelanggan yang membutuhkan barangnya tiba dengan cepat.
3. Kiriman Kargo: Layanan khusus untuk barang-barang berukuran besar atau berat, umumnya dengan berat minimal 10 kg, yang menggunakan moda transportasi darat atau laut untuk efisiensi biaya.

Setiap jenis layanan ini memiliki struktur biaya yang berbeda, yang ditentukan oleh berbagai faktor seperti kecepatan pengiriman, berat dan dimensi paket, serta jarak tempuh. Oleh karena itu, sebuah sistem informasi ekspedisi yang canggih harus mampu mengelola kompleksitas dari berbagai pilihan layanan ini, memberikan transparansi harga, dan memungkinkan pelanggan untuk membuat pilihan yang tepat secara mandiri [3].

1.2 Batasan Sistem

Sistem yang dikembangkan dalam proyek ini memiliki batasan fungsionalitas yang terfokus pada kalkulasi biaya pengiriman standar. Proses perhitungan yang diimplementasikan hanya mencakup variabel-variabel dasar, yaitu berat paket (dalam kilogram), volume paket, dan jarak pengiriman ke kota tujuan dengan input manual. Lebih lanjut, batasan sistem ini melampaui aspek kalkulasi dan mencakup ketiadaan berbagai fitur operasional yang vital dalam ekosistem logistik. Sistem belum dilengkapi dengan modul pelacakan paket (*real-time tracking*), fungsi untuk menghasilkan resi dan label pengiriman secara otomatis, serta fitur untuk manajemen permintaan penjemputan barang oleh kurir. Selain itu, sistem ini masih bersifat mandiri dan belum memiliki kemampuan untuk terintegrasi dengan API dari berbagai jasa kurir, yang membuatnya tidak dapat menyediakan notifikasi status pengiriman secara proaktif kepada pengguna atau menawarkan perbandingan layanan.

1.3 Tujuan dan Kegunaan

Tujuan utama dari pengembangan program ini adalah untuk membangun sebuah sistem kalkulasi biaya pengiriman berbasis web menggunakan Java. Sistem ini ditargetkan untuk dapat mengimplementasikan tiga skema perhitungan yang berbeda sesuai jenis layanan, yaitu Reguler, Kilat, dan Kargo. Fungsionalitas utamanya adalah menyediakan antarmuka bagi pengguna untuk memasukkan data pengirim, penerima, serta berat barang, yang kemudian diproses untuk menghasilkan output berupa total biaya pengiriman yang akurat dan final.

Dengan tercapainya tujuan tersebut, program ini diharapkan dapat memberikan kegunaan praktis dalam meningkatkan efisiensi dan akurasi proses penentuan ongkos kirim. Sistem ini berfungsi untuk meminimalisir potensi kesalahan perhitungan manual sekaligus menyediakan platform terpusat untuk standardisasi harga. Pada akhirnya, aplikasi ini dapat menjadi fondasi dasar yang solid untuk pengembangan sistem manajemen ekspedisi yang lebih besar dan komprehensif di masa mendatang.

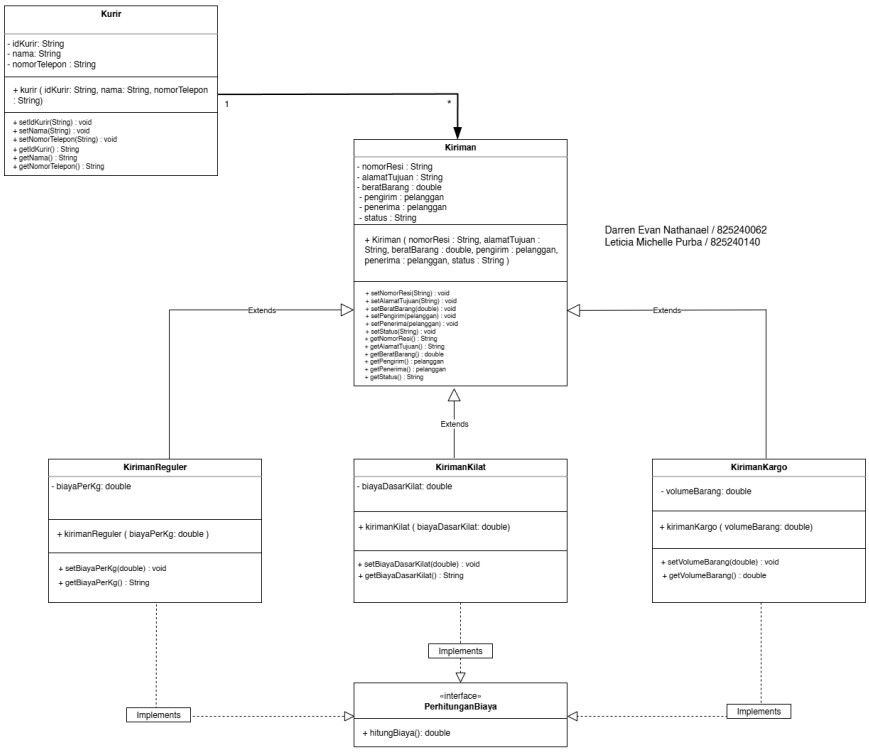
BAB II RANCANGAN SISTEM

2.1 Sistem yang Dirancang

Rancangan sistem pengiriman barang ini digambarkan menggunakan beberapa diagram UML (*Unified Modeling Language*) untuk memvisualisasikan struktur, fungsionalitas, dan interaksi antar komponennya. Pembahasan secara garis besar dimulai dengan *class diagram* yang menjadi fondasi dari struktur sistem.

Gambar 2.1 menampilkan *class diagram* untuk sistem pengiriman barang yang dirancang. Sistem ini secara keseluruhan memiliki lima *class*, yaitu Kurir, Kiriman, KirimanReguler, KirimanKilat, dan KirimanKargo. Dari kelima *class* tersebut, Kiriman merupakan sebuah *abstract class*. Hal ini dikarenakan Kiriman merepresentasikan konsep umum dari sebuah pengiriman dan tidak dapat diinstansiasi secara langsung menjadi objek. *Class* ini berfungsi sebagai kerangka dasar yang mewarisi atribut dan metode umum (seperti nomorResi, alamatTujuan, setStatus(), dll.) kepada *subclass* yang lebih spesifik dan konkret, yaitu KirimanReguler, KirimanKilat, dan KirimanKargo.

Selanjutnya, rancangan sistem ini juga menyertakan satu buah *interface*, yaitu PerhitunganBiaya. *Interface* ini mendefinisikan "kontrak" metode hitungBiaya() yang wajib diimplementasikan oleh setiap *class* pengiriman konkret untuk memastikan semuanya memiliki fungsionalitas perhitungan biaya sesuai dengan jenisnya masing-masing.

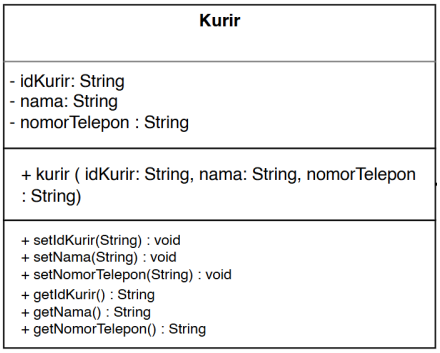


Gambar 2.1 Class Diagram untuk sistem ekspedisi barang

2.2 Class Diagram

Class diagram pada Gambar 2.1 mendeskripsikan struktur statis dari sistem pengiriman barang, Komponen-komponen tersebut adalah sebagai berikut:

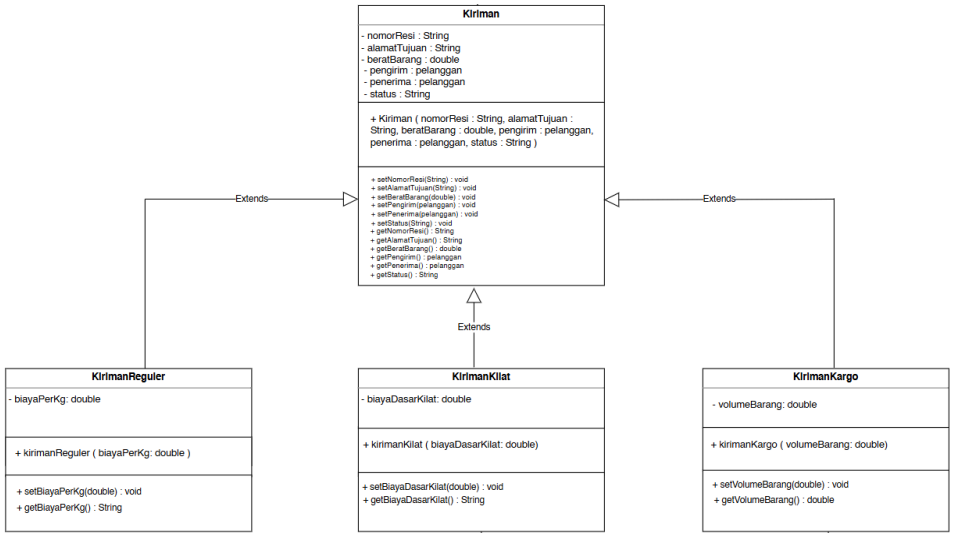
1. *class* Kurir, yang merepresentasikan entitas pengantar barang dengan atribut esensial seperti idKurir untuk identifikasi unik, nama, dan nomorTelepon.



Gambar 2.2 Class Diagram Kurir

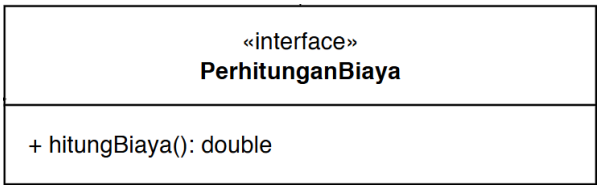
2. *abstract class* Kiriman, yang berfungsi sebagai kerangka dasar untuk semua jenis pengiriman dan tidak dapat diinstansiasi secara langsung. *Class* ini memuat properti umum yang akan diwariskan, seperti nomorResi, alamatTujuan, beratBarang, dan status pengiriman. Dari *class* induk ini, diturunkan tiga *class* konkret yang lebih spesifik:
    a) KirimanReguler
    b) KirimanKilat
    c) KirimanKargo

    Masing-masing merepresentasikan layanan yang berbeda dan memiliki atribut uniknya sendiri untuk kalkulasi biaya, yaitu biayaPerKg untuk layanan reguler, biayaDasarKilat untuk layanan kilat, dan volumeBarang untuk layanan kargo.



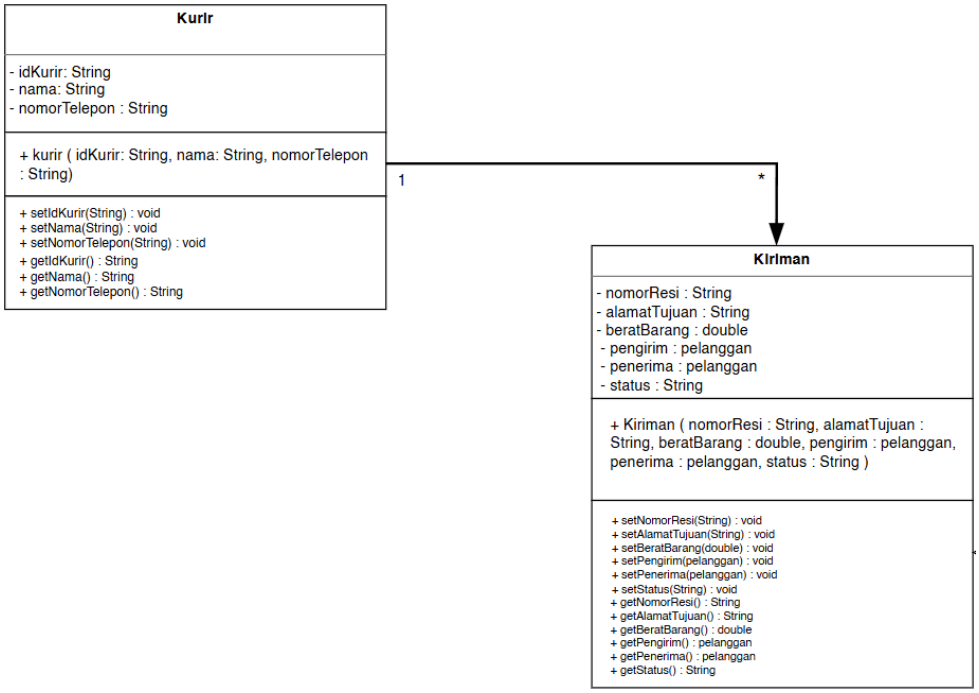
Gambar 2.3 Abstract Class Kiriman dan *inheritance-nya*

3. *interface* PerhitunganBiaya, yang bertindak sebagai "kontrak" dengan mendeklarasikan metode hitungBiaya(), di mana implementasi konkretnya wajib disediakan oleh setiap jenis kiriman untuk menghitung total biaya sesuai aturannya masing-masing.



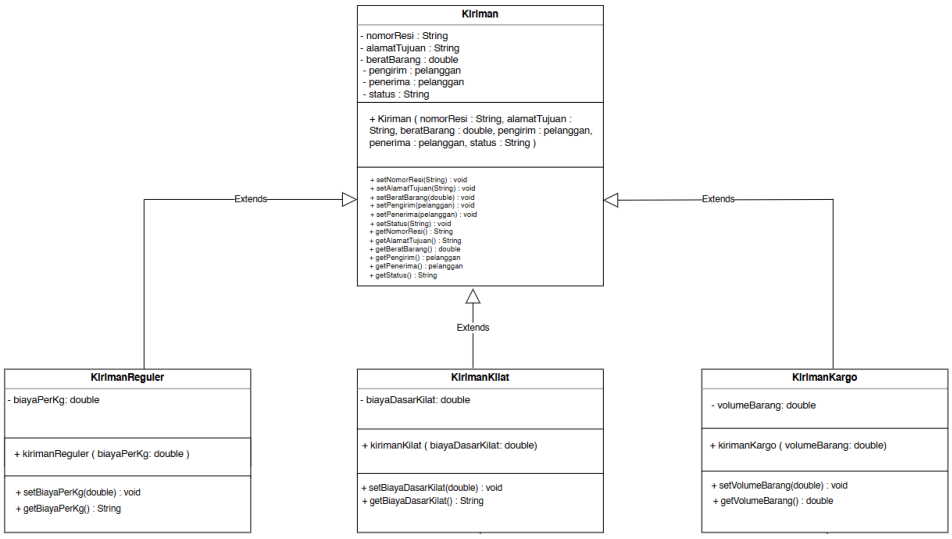
Gambar 2.4 Class *Interface* PerhitunganBiaya

Hubungan antar komponen pertama adalah Asosiasi (*Association*), yang terjadi antara *class* Kurir dan Kiriman. Relasi ini memiliki multiplisitas *one-to-many* (1..*), yang berarti satu objek Kurir dapat terhubung dengan satu atau lebih objek Kiriman. Dalam konteks sistem, ini menggambarkan bahwa seorang kurir dapat bertanggung jawab untuk mengirimkan banyak paket, namun setiap paket atau kiriman hanya ditangani oleh satu kurir pada satu waktu.



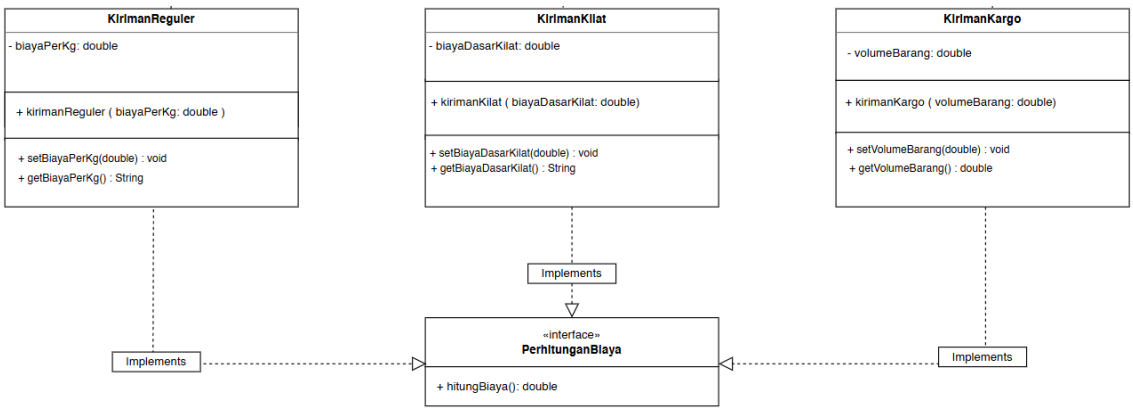
Gambar 2.5 *Association* Class Kurir dengan Class Kiriman

Selanjutnya adalah hubungan Pewarisan (*Inheritance*). Relasi ini terlihat dari *abstract class* Kiriman sebagai induk (*superclass*) ke tiga *class* (KirimanReguler, KirimanKilat, KirimanKargo) sebagai anak (*subclass*). Melalui pewarisan, setiap *subclass* secara otomatis memiliki semua atribut dan metode yang didefinisikan pada *class* Kiriman.



Gambar 2.6 *Inheritance* Class Kiriman dengan Class KirimanReguler, KirimanKilat , dan KirimanKargo

Terakhir, terdapat hubungan Realisasi (*Implementation*) antara *interface* PerhitunganBiaya dengan ketiga *class* kiriman konkret. Relasi ini berfungsi sebagai sebuah "kontrak" yang mewajibkan KirimanReguler, KirimanKilat, dan KirimanKargo untuk menyediakan implementasi atau logika spesifik untuk metode hitungBiaya() yang dideklarasikan dalam *interface*. Hal ini memastikan bahwa setiap jenis layanan pengiriman memiliki cara perhitungan biayanya sendiri, sekaligus memungkinkan sistem untuk menerapkan konsep polimorfisme, di mana objek dari jenis kiriman yang berbeda dapat diproses secara seragam untuk menghitung biayanya.



Gambar 2.6 *Inheritance* Class Kiriman dengan Class KirimanReguler, KirimanKilat , dan KirimanKargo

BAB III PEMBUATAN SISTEM

3.1 Implementasi Sistem yang Dirancang

3.1.1 Bahasa Pemograman dan Library

1. Bahasa Pemograman
   Program sistem informasi ekspedisi ini dibangun menggunakan bahasa pemrograman Java. Java memiliki dukungan kuat terhadap paradigma *Object-Based Programming (OBP)*, yang sangat sesuai untuk memodelkan entitas dunia nyata seperti Kiriman, Pelanggan, dan Kurir ke dalam bentuk objek dalam program. Konsep seperti *inheritance*, *polymorphism*, dan *encapsulation*.
2. Library
   Untuk menunjang fungsionalitas program, beberapa library standar dari Java Development Kit (JDK) diimpor dan digunakan. Berikut adalah daftar beserta penjelasannya:

   a) java.util.ArrayList & java.util.List: Digunakan untuk menyimpan daftar objek Kiriman dan Kurir. ArrayList dipilih karena merupakan struktur data dinamis yang ukurannya dapat bertambah atau berkurang secara otomatis saat data kiriman ditambahkan (daftarKiriman.add()) atau dihapus (daftarKiriman.remove()).
   b) java.util.Scanner: Library ini berfungsi sebagai jembatan interaksi antara pengguna dan aplikasi. Scanner digunakan untuk membaca input dari pengguna melalui konsol, seperti memilih menu, memasukkan data pengirim, penerima, dan detail kiriman lainnya.
   c) java.util.InputMismatchException: Digunakan sebagai bagian dari mekanisme *error handling*. Exception ini ditangkap menggunakan blok try-catch untuk menangani kasus ketika pengguna memasukkan tipe data yang salah (misalnya, memasukkan teks saat diminta angka), sehingga program tidak berhenti secara paksa dan dapat memberikan pesan kesalahan yang informatif.
   d) java.io.*: Merupakan paket yang berisi kelas-kelas untuk operasi *Input/Output* (I/O). Dalam program ini, kelas-kelas seperti BufferedWriter, FileWriter, BufferedReader, dan FileReader digunakan untuk mengimplementasikan fitur histori. Fungsinya adalah untuk menyimpan data transaksi kiriman ke dalam sebuah file eksternal (histori.txt) dan membacanya kembali, sehingga data tidak hilang saat program ditutup.
   e) kiriman.*, kurir.Kurir, pelanggan.Pelanggan, perhitunganbiaya.PerhitunganBiaya: Ini bukanlah library eksternal, melainkan paket (*package*) yang dibuat sendiri untuk mengorganisasi kelas-kelas program. Penggunaan paket ini menunjukkan penerapan struktur kode yang baik, di mana kelas-kelas dengan fungsionalitas terkait dikelompokkan bersama untuk meningkatkan keterbacaan dan kemudahan pengelolaan kode.

3.1.2 Tahapan Pembuatan Program

Pembuatan program dilakukan secara bertahap dengan pendekatan modular, dimulai dari fondasi data hingga fungsionalitas utama.

1. Perancangan Kelas dan Struktur Data (Model) Tahap pertama adalah merancang dan membuat kelas-kelas inti yang merepresentasikan objek-objek dalam sistem. Ini termasuk pembuatan kelas Pelanggan, Kurir, dan kelas Kiriman sebagai kelas induk (*superclass*). Selanjutnya, dibuat kelas-kelas turunan seperti KirimanReguler, KirimanKilat, dan KirimanKargo yang mewarisi sifat dari Kiriman namun memiliki atribut dan metode perhitungan biaya yang spesifik. Pada tahap ini, *interface* PerhitunganBiaya juga dibuat untuk memastikan setiap jenis kiriman memiliki metode hitungBiaya().

2. Pembuatan Menu Utama dan Navigasi Setelah model data terbentuk, fokus beralih ke pembuatan kelas Main yang berfungsi sebagai titik masuk program. Di sini, struktur menu utama dibuat menggunakan perulangan while dan logika percabangan switch-case untuk navigasi antar menu. Fungsi-fungsi dasar seperti tampilkanMenu() dibuat untuk menampilkan opsi kepada pengguna.

3. Implementasi Fitur Inti (CRUD) Fungsionalitas utama yaitu *Create, Read, Update, Delete* (CRUD) diimplementasikan secara berurutan:
    a) Create: Fungsi buatKiriman() diimplementasikan untuk menangani proses input data dari pengguna, mulai dari data pengirim, penerima, detail barang, hingga pemilihan kurir. Objek baru kemudian dibuat dan disimpan ke dalam ArrayList daftarKiriman.
    b) Read: Fungsi tampilkanDaftarKiriman() dibuat untuk menampilkan semua data yang tersimpan di ArrayList dalam format tabel yang rapi dan mudah dibaca.
    c) Update: Fungsi updateKiriman() dikembangkan untuk memungkinkan pengguna mengubah status sebuah kiriman berdasarkan nomor urut yang dipilih dari daftar.
    d) Delete: Fungsi hapusKiriman() dibuat untuk menghapus data kiriman dari ArrayList setelah konfirmasi dari pengguna.

4. Implementasi Fitur Penyimpanan Histori tahap selanjutnya adalah memastikan data dapat disimpan secara persisten. Fungsi simpanKeHistori(), simpanSemuaDataKeHistori(), dan bacaHistori() dibuat menggunakan library java.io.*. Fungsi ini menangani penulisan data ke file histori.txt setiap kali ada kiriman baru, pembaruan status, atau penghapusan data, serta membaca kembali data dari file tersebut saat pengguna ingin melihat histori.

5. Penyempurnaan dan Penanganan Kesalahan Tahap terakhir adalah melakukan penyempurnaan pada antarmuka pengguna (misalnya, menambahkan pesan "Tekan Enter untuk kembali") dan mengimplementasikan penanganan kesalahan (*error handling*) menggunakan try-catch untuk mengantisipasi input yang tidak valid dari pengguna, sehingga program menjadi lebih kuat (*robust*) dan ramah pengguna (*user-friendly*).

3.2 Implementasi Abstract Class Member

Alur utama program dikendalikan oleh sebuah perulangan *while* yang menampilkan menu utama. Navigasi antar fitur diatur oleh blok *switch-case* yang mengeksekusi metode yang sesuai berdasarkan input numerik dari pengguna. Untuk menjaga kestabilan aplikasi, diimplementasikan juga blok *try-catch* yang secara spesifik menangani *InputMismatchException* jika pengguna memasukkan data yang tidak valid. Selanjutnya, proses pembuatan data baru diatur dalam metode buatKiriman(). Metode ini bertugas mengumpulkan semua input pengguna untuk kemudian membuat instansiasi objek dari kelas turunan yang spesifik, seperti KirimanReguler, KirimanKilat, atau KirimanKargo. Konsep Polimorfisme diterapkan secara efektif dalam metode ini, di mana objek yang baru dibuat di-casting ke interface PerhitunganBiaya untuk memanggil metode hitungBiaya() yang implementasinya unik untuk setiap jenis kiriman.

Untuk pengelolaan data, fungsionalitas pembaruan dan penghapusan diimplementasikan dalam metode updateKiriman() dan hapusKiriman(). Keduanya bekerja dengan pola yang sama: menampilkan daftar data yang ada, meminta pilihan pengguna berdasarkan nomor urut, lalu memodifikasi *ArrayList* daftarKiriman. Operasi ini dilakukan dengan memanggil metode seperti setStatus() untuk memperbarui status atau remove() untuk menghapus sebuah elemen dari daftar. Terakhir, untuk memastikan data tidak hilang saat program ditutup, fitur persistensi data diimplementasikan melalui metode seperti simpanSemuaDataKeHistori(). Metode ini menyimpan kondisi terkini dari ArrayList ke dalam file histori.txt dengan menggunakan kelas BufferedWriter dan FileWriter. Proses ini mengiterasi setiap objek dalam daftar dan menuliskan detailnya, sehingga semua penambahan, perubahan, atau penghapusan data dapat tersimpan secara permanen.

```java
// Darren Evan Nathanael 825240062
import java.io.*;
import java.util.ArrayList;
import java.util.InputMismatchException;
import java.util.List;
import java.util.Scanner;
import kiriman.*;
import kurir.Kurir;
import pelanggan.Pelanggan;
import perhitunganbiaya.PerhitunganBiaya;

public class Main {
    private static List<Kiriman> daftarKiriman = new ArrayList<>();
    private static List<Kurir> daftarKurir = new ArrayList<>();

    Run main | Debug main | Run | Debug
    public static void main(String[] args) {
        daftarKurir.add(new Kurir(idKurir:"K01", nama:"Budi Santoso", nomorTelepon:"0812345678"));
        daftarKurir.add(new Kurir(idKurir:"K02", nama:"Citra Lestari", nomorTelepon:"0823456789"));
        daftarKurir.add(new Kurir(idKurir:"K03", nama:"Ahmad Fauzi", nomorTelepon:"0834567890"));

        Scanner scanner = new Scanner(System.in);
        int pilihan = 0;

        // MENAMPILKAN MENU UTAMA
        while (pilihan != 5) {
            tampilkanMenu();
            try {
                System.out.print(s:"Masukkan pilihan Anda: ");
                pilihan = scanner.nextInt();
                scanner.nextLine();

                switch (pilihan) {
                    case 1:
                        pilihJenisKiriman(scanner);
                        break;
                    case 2:
                        lihatDaftarKirimanDanTunggu(scanner);
                        break;
                    case 3:
                        updateKiriman(scanner);
                        break;
                    case 4:
                        hapusKiriman(scanner);
                        break;
                    case 5:
                        System.out.println(x:"Terima kasih telah menggunakan program ini!");
                        break;
                    default:
                        System.out.println(x:"Pilihan tidak valid. Silakan coba lagi.");
                }
```

Lampiran 3.1 *Source code* Main.java

3.3 Implementasi Class

3.3.1 *Abstract* Class Kiriman

Kelas Kiriman.java merupakan sebuah abstract class yang berfungsi sebagai kerangka dasar atau *blueprint* untuk semua jenis pengiriman dalam sistem. Kelas ini mendefinisikan seluruh atribut esensial yang dimiliki setiap kiriman seperti nomorResi, pengirim, penerima, dan status dengan hak akses private untuk menerapkan prinsip enkapsulasi. Untuk mengakses dan memodifikasi data privat tersebut, disediakan metode publik setter dan getter. Sebuah constructor juga disiapkan untuk memastikan setiap objek kiriman dibuat dengan data awal yang lengkap. Pada dasarnya, kelas ini menjadi fondasi yang akan diwariskan (inheritance) oleh kelas-kelas yang lebih spesifik (seperti KirimanReguler dan KirimanKargo), sehingga mereka dapat berbagi struktur data yang sama sambil menambahkan logika fungsional yang unik. Source code lengkap terdapat di halaman Lampiran.

```java
//Leticia Michelle Purba / 825240140

package kiriman;

import pelanggan.Pelanggan;
import kurir.Kurir;

public abstract class Kiriman {
    private String nomorResi;
    private String alamatTujuan;
    private String status;
    private double beratBarang;
    private Pelanggan pengirim;
    private Pelanggan penerima;
    private Kurir kurir;

    public Kiriman(String nomorResi, String alamatTujuan, double beratBarang, Pelanggan pengirim, Pelanggan penerima, String status, Kurir kurir) {
        this.nomorResi = nomorResi;
        this.alamatTujuan = alamatTujuan;
        this.beratBarang = beratBarang;
        this.pengirim = pengirim;
        this.penerima = penerima;
        this.status = status;
        this.kurir = kurir;
    }

    // SETTER AND GETTER
    public String getNomorResi() {
        return nomorResi;
    }

    public void setNomorResi(String nomorResi) {
        this.nomorResi = nomorResi;
    }
```

Lampiran 3.2 *Source Code Abstract* Class Kiriman

3.3.2 *Inheritance* Class Kiriman ke Class KirimanReguler, KirimanKilat, dan KirimanKargo

1. Class KirimanReguler
   Kelas KirimanReguler mewarisi (extends) semua sifat dasar dari kelas Kiriman, seperti nomorResi, pengirim, penerima, dan status. Dengan kata lain, ia adalah "anak" dari Kiriman. Keunikannya adalah penambahan atribut spesifik yaitu biayaPerKg dan implementasi metode hitungBiaya() yang logikanya adalah mengalikan berat barang dengan biaya per kilogram, sesuai dengan karakteristik layanan reguler.

```java
//Leticia Michelle Purba / 825240140

package kiriman;

import pelanggan.Pelanggan;
import perhitunganbiaya.PerhitunganBiaya;
import kurir.Kurir;

public class KirimanReguler extends Kiriman implements PerhitunganBiaya {
    private double biayaPerKg;

    public KirimanReguler(String nomorResi, String alamatTujuan, double beratBarang, Pelanggan pengirim, Pelanggan penerima, String status, double biayaPerKg, Kurir kurir) {
        super(nomorResi, alamatTujuan, beratBarang, pengirim, penerima, status, kurir);
        this.biayaPerKg = biayaPerKg;
    }

    // SETTER AND GETTER
    public double getBiayaPerKg() {
        return biayaPerKg;
    }

    public void setBiayaPerKg(double biayaPerKg) {
        this.biayaPerKg = biayaPerKg;
    }

    @Override
    public double hitungBiaya() {
        return getBeratBarang() * this.biayaPerKg;
    }
}
```

Gambar 3.3 *Source Code* Class KirimanReguler

2. Class KirimanKilat

Sama seperti KirimanReguler, kelas KirimanKilat juga merupakan turunan dari Kiriman dan mewarisi semua atribut umumnya. Yang membedakannya adalah ia memiliki atribut biayaDasarKilat yang khas untuk layanan kilat. Implementasi metode hitungBiaya() di kelas ini juga spesifik, yaitu menjumlahkan biaya dasar dengan hasil perkalian berat barang, mencerminkan skema tarif pengiriman cepat.

```java
//Leticia Michelle Purba / 825240140

package kiriman;

import pelanggan.Pelanggan;
import perhitunganbiaya.PerhitunganBiaya;
import kurir.Kurir;

public class KirimanKilat extends Kiriman implements PerhitunganBiaya {
    private double biayaDasarKilat;

    public KirimanKilat(String nomorResi, String alamatTujuan, double beratBarang, Pelanggan pengirim, Pelanggan penerima, String status, double biayaDasarKilat, Kurir kurir) {
        super(nomorResi, alamatTujuan, beratBarang, pengirim, penerima, status, kurir);
        this.biayaDasarKilat = biayaDasarKilat;
    }

    // SETTER AND GETTER
    public double getBiayaDasarKilat() {
        return biayaDasarKilat;
    }

    public void setBiayaDasarKilat(double biayaDasarKilat) {
        this.biayaDasarKilat = biayaDasarKilat;
    }

    @Override
    public double hitungBiaya() {
        return this.biayaDasarKilat + (getBeratBarang() * 5000);
    }
}
```

Gambar 3.4 *Source Code* Class KirimanKilat

3. Class KirimanKargo

Kelas KirimanKargo mewarisi (extends) Kiriman untuk mendapatkan semua properti standar pengiriman. Kelas ini menambahkan atribut uniknya sendiri yaitu volumeBarang karena pengiriman kargo seringkali mempertimbangkan dimensi barang. Oleh karena itu, implementasi metode hitungBiaya() di sini adalah yang paling kompleks, karena perhitungannya melibatkan kombinasi antara berat barang dan volume barang untuk menentukan total biaya.

```java
//Leticia Michelle Purba / 825240140

package kiriman;

import pelanggan.Pelanggan;
import perhitunganbiaya.PerhitunganBiaya;
import kurir.Kurir;

public class KirimanKargo extends Kiriman implements PerhitunganBiaya {

    private double volumeBarang;

    public KirimanKargo(String nomorResi, String alamatTujuan, double beratBarang,
                        Pelanggan pengirim, Pelanggan penerima, String status,
                        double volumeBarang, Kurir kurir) {

        super(nomorResi, alamatTujuan, beratBarang, pengirim, penerima, status, kurir);

        this.volumeBarang = volumeBarang;
    }

    // SETTER AND GETTER
    public double getVolumeBarang() {
        return volumeBarang;
    }

    public void setVolumeBarang(double volumeBarang) {
        this.volumeBarang = volumeBarang;
    }

    @Override
    public double hitungBiaya() {
        return (getBeratBarang() * 3000) + (this.volumeBarang * 50000);
    }
}
```

Gambar 3.5 *Source Code* Class KirimanKargo

3.3.3 *Interface* Class PerhitunganBiaya

PerhitunganBiaya.java adalah sebuah interface, yang bisa diibaratkan sebagai sebuah kontrak atau aturan main. Ia tidak berisi logika, melainkan hanya mendeklarasikan bahwa setiap kelas yang setuju untuk menggunakannya (implements) wajib menyediakan implementasi konkret dari metode hitungBiaya() yang mengembalikan nilai double. Tujuannya adalah untuk menyeragamkan cara perhitungan biaya di semua jenis kiriman (KirimanReguler, KirimanKilat, KirimanKargo), sehingga memastikan semuanya memiliki fungsi kalkulasi biaya meskipun dengan rumus yang berbeda-beda.

```java
//Leticia Michelle Purba / 825240140

package perhitunganbiaya;
    public interface PerhitunganBiaya {
    double hitungBiaya();
}
```

Gambar 3.6 *Source Code* Class PerhitunganBiaya

3.3.4 *Association* Class Kurir

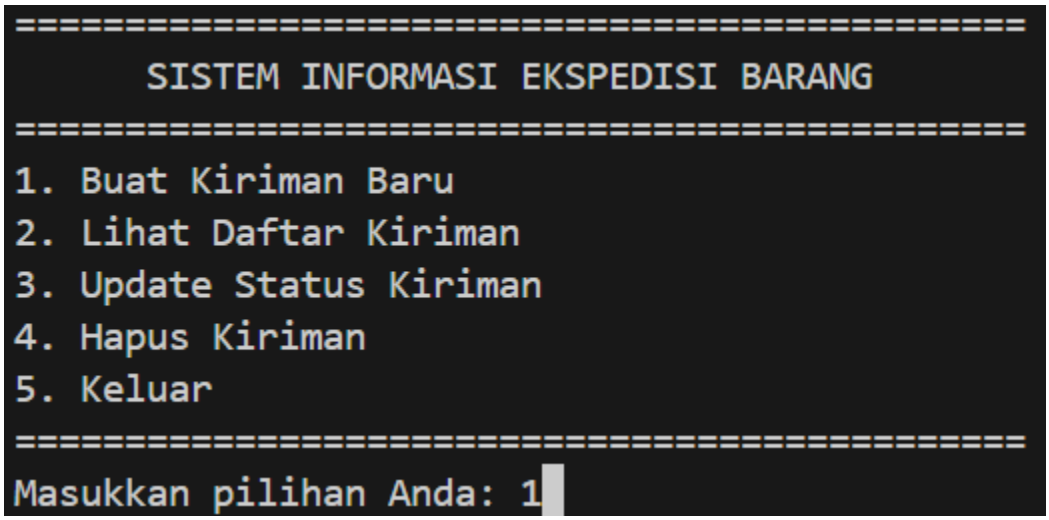
Hubungan antara kelas Kurir dan Kiriman diimplementasikan menggunakan konsep Asosiasi (Association). Secara teknis, hubungan ini didefinisikan di dalam kelas Kiriman melalui deklarasi atribut private Kurir kurir;. Deklarasi tersebut menetapkan bahwa setiap objek Kiriman wajib memiliki satu referensi ke objek Kurir, yang mencerminkan bahwa satu kiriman ditangani oleh satu kurir. Oleh karena itu, kelas Kurir berfungsi sebagai entitas mandiri yang dapat digunakan kembali dan diasosiasikan dengan berbagai objek Kiriman yang berbeda. Source code lengkap terdapat di halaman Lampiran.

```java
// Darren Evan Nathanael 825240062

package kurir;

public class Kurir {
    private String idKurir;
    private String nama;
    private String nomorTelepon;

    public Kurir(String idKurir, String nama, String nomorTelepon) {
        this.idKurir = idKurir;
        this.nama = nama;
        this.nomorTelepon = nomorTelepon;
    }

    // --- GETTER & SETTER ---
    public String getIdKurir() {
        return idKurir;
    }

    public void setIdKurir(String idKurir) {
        this.idKurir = idKurir;
    }
```

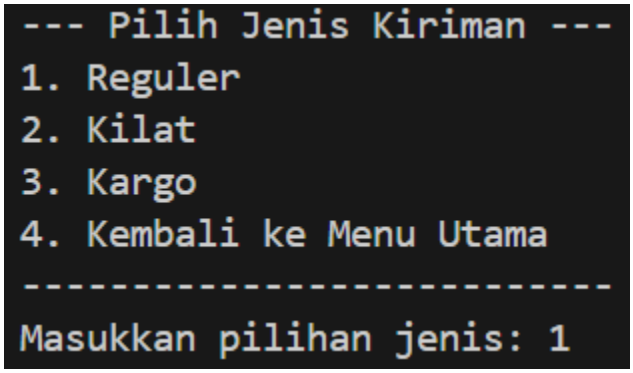
Lampiran 3.7 *Source Code Association* Class Kurir

BAB IV PENGUJIAN PROGRAM APLIKASI

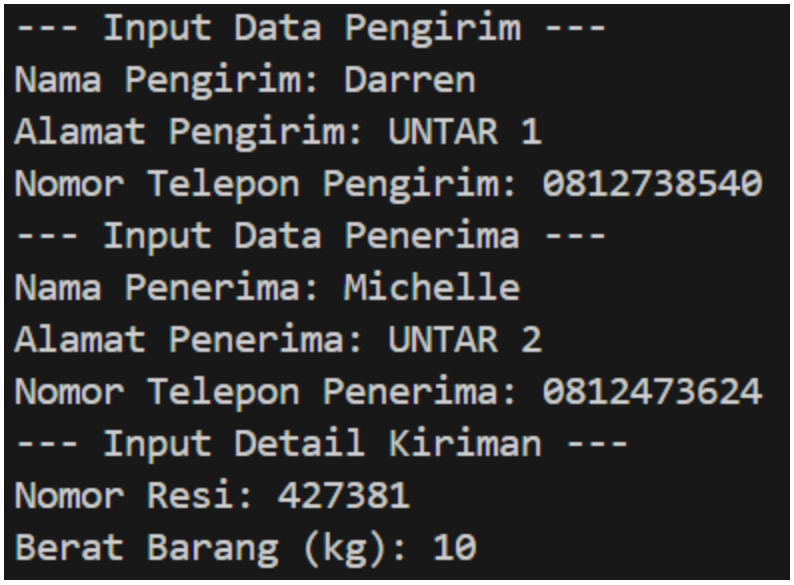
4.1 Skenario Pengujian Membuat kiriman baru



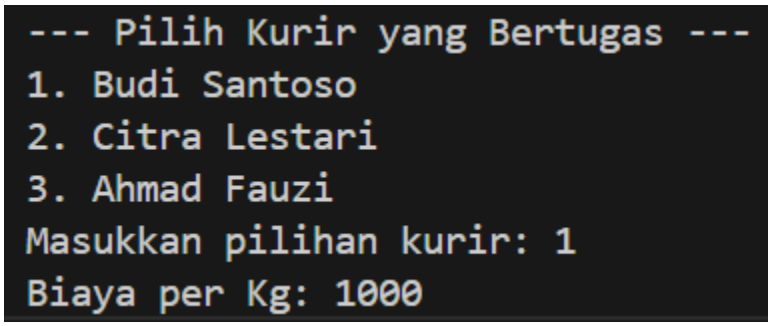
Ketik 1 pada menu utama untuk membuat kiriman baru.



Memilih jenis kiriman.



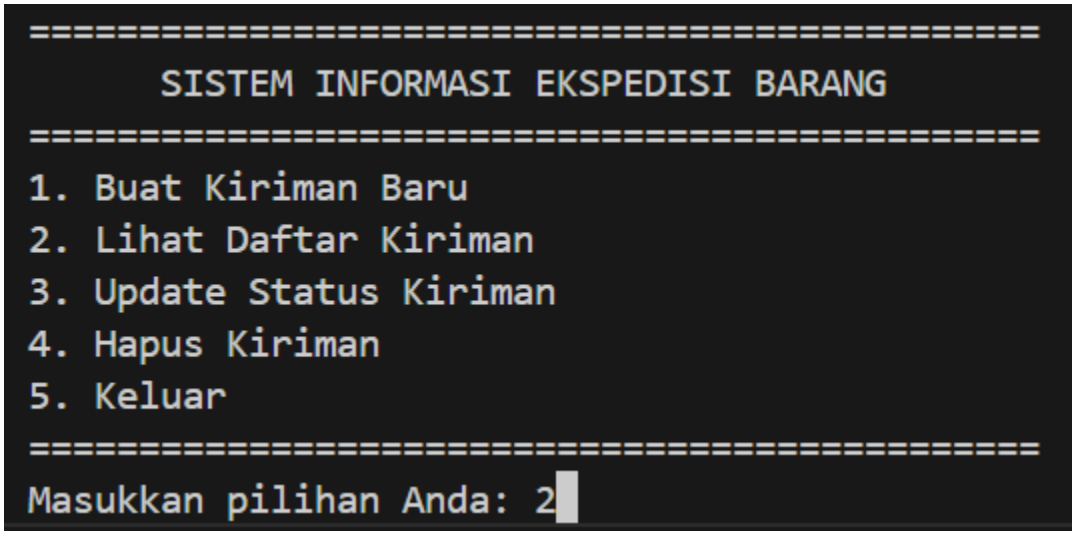
Menginput data pengirim, penerima dan detail dari kiriman.



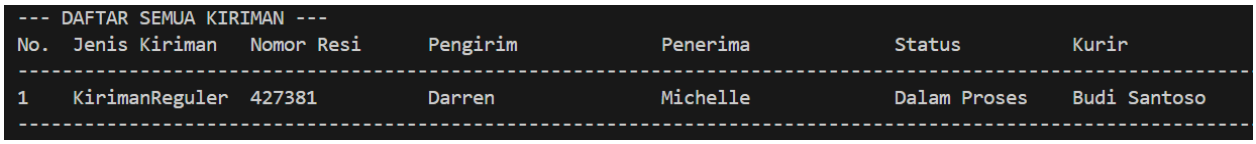
Memilih kurir yang bertugas dan memasukan biaya per kg.

Kiriman telah dibuat dan akan ditampilkan ringkasan pengiriman serta disimpan pada file “histori.txt”.

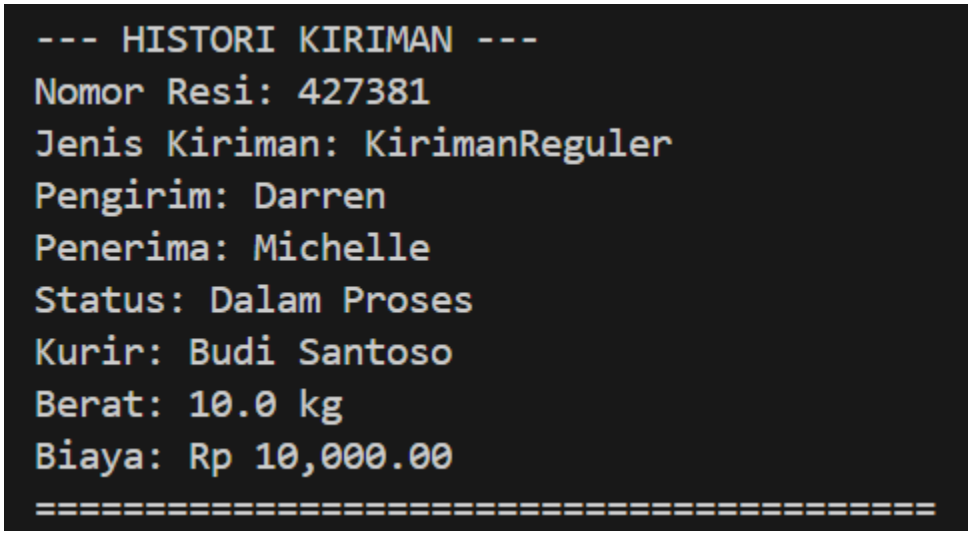
4.2 Pengujian Skenario Melihat Daftar Kiriman



Ketik 2 pada menu utama untuk melihat daftar kiriman.

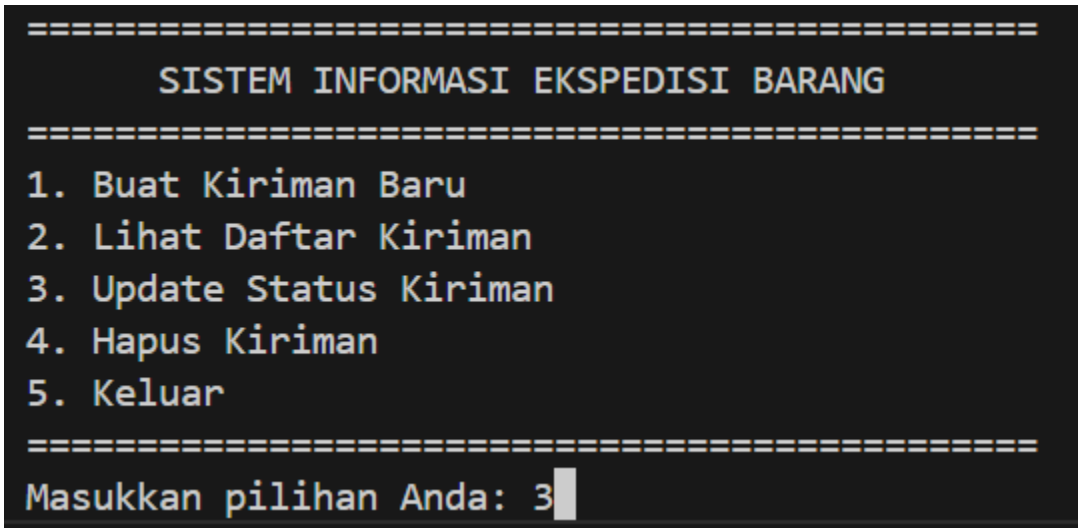


Program akan menampilkan daftar semua kiriman yang sedang berjalan.

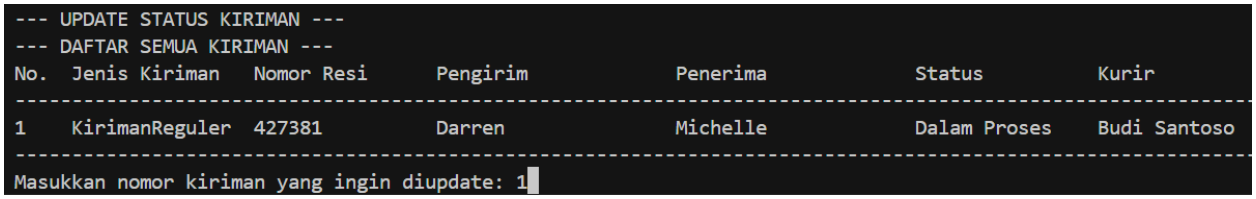


Program juga akan menampilkan data histori pengiriman yang telah tersimpan di “histori.txt”.

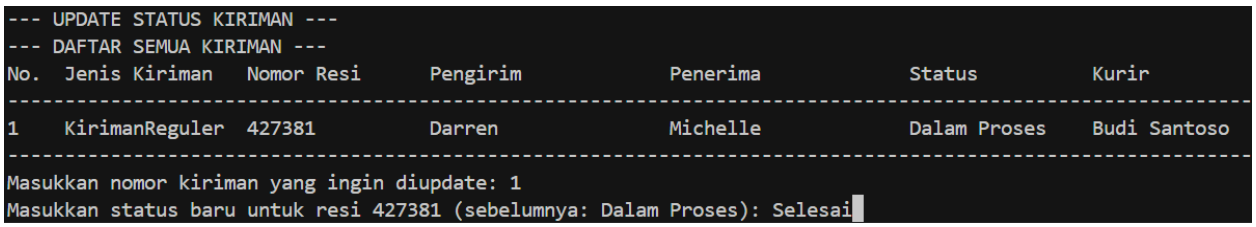
4.3 Pengujian Skenario Update Status Kiriman



Ketik 3 pada menu utama.



Program akan menampilkan daftar semua kiriman yang berurut menggunakan nomor sehingga kita bisa tinggal ketik nomor kiriman yang kita mau untuk meng-update status kiriman.



Masukan status terbaru yang ingin kita update misalnya : “Selesai” atau “Diretur”.

```
--- DAFTAR SEMUA KIRIMAN ---
No.  Jenis Kiriman   Nomor Resi      Pengirim            Penerima            Status          Kurir
-------------------------------------------------------------------------------------------------------
1    KirimanReguler  427381          Darren              Michelle            Selesai         Budi Santoso
-------------------------------------------------------------------------------------------------------

--- HISTORI KIRIMAN ---
Nomor Resi: 427381
Jenis Kiriman: KirimanReguler
Pengirim: Darren
Penerima: Michelle
Status: Selesai
Kurir: Budi Santoso
Berat: 10.0 kg
Biaya: Rp 10,000.00
========================================
Tekan Enter untuk kembali ke menu utama...
```

Kiriman yang sudah kita update statusnya akan berubah ketika kita kembali ke menu utama lalu memilih menu “Lihat Daftar Kiriman”. Status di dalam file histori.txt juga akan berubah.

4.4 Pengujian Skenario Menghapus Kiriman

```
==================================================
      SISTEM INFORMASI EKSPEDISI BARANG
==================================================
1. Buat Kiriman Baru
2. Lihat Daftar Kiriman
3. Update Status Kiriman
4. Hapus Kiriman
5. Keluar
==================================================
Masukkan pilihan Anda: 4
```

Ketik 4 pada menu utama.

```
--- HAPUS KIRIMAN ---
--- DAFTAR SEMUA KIRIMAN ---
No.  Jenis Kiriman   Nomor Resi      Pengirim            Penerima            Status          Kurir
-------------------------------------------------------------------------------------------------------
1    KirimanReguler  427381          Darren              Michelle            Selesai         Budi Santoso
-------------------------------------------------------------------------------------------------------
Masukkan nomor kiriman yang ingin dihapus: 1
```

Masukan nomor kiriman yang ingin dihapus.

```
--- HAPUS KIRIMAN ---
--- DAFTAR SEMUA KIRIMAN ---
No.  Jenis Kiriman   Nomor Resi      Pengirim            Penerima            Status          Kurir
-------------------------------------------------------------------------------------------------------
1    KirimanReguler  427381         Darren              Michelle            Selesai         Budi Santoso
-------------------------------------------------------------------------------------------------------
Masukkan nomor kiriman yang ingin dihapus: 1
Anda yakin ingin menghapus kiriman dengan resi 427381? (y/n): y
```

Konfirmasi untuk menghapus kiriman dengan mengetik y.

```
==============================================
      SISTEM INFORMASI EKSPEDISI BARANG
==============================================
1. Buat Kiriman Baru
2. Lihat Daftar Kiriman
3. Update Status Kiriman
4. Hapus Kiriman
5. Keluar
==============================================
Masukkan pilihan Anda: 2
--- DAFTAR SEMUA KIRIMAN ---
Belum ada data pengiriman yang dibuat.

--- HISTORI KIRIMAN ---
Tekan Enter untuk kembali ke menu utama...
```

Kiriman akan terhapus ketika kita kembali ke menu utama lalu memilih menu Lihat Daftar Kiriman. Data di histori.txt juga akan terhapus.

BAB V PENUTUP

5.1 Kesimpulan

Berdasarkan hasil implementasi dan pengujian sistem yang telah diuraikan pada bab-bab sebelumnya, dapat ditarik beberapa kesimpulan sebagai berikut:

1. Program aplikasi Sistem Informasi Ekspedisi Barang telah berhasil dibangun sesuai dengan tujuan utama, yaitu menyediakan fungsionalitas untuk menghitung biaya pengiriman berbasis web menggunakan Java.
2. Sistem telah berhasil mengimplementasikan tiga skema perhitungan biaya yang berbeda untuk jenis layanan Kiriman Reguler, Kiriman Kilat, dan Kiriman Kargo, sesuai dengan rancangan yang telah dibuat.
3. Konsep-konsep fundamental dari *Object-Based Programming* telah berhasil diterapkan, meliputi penggunaan *abstract class* Kiriman sebagai kerangka dasar , mekanisme *inheritance* pada kelas turunan (KirimanReguler, KirimanKilat, KirimanKargo) , *interface* PerhitunganBiaya untuk standarisasi metode kalkulasi , serta *association* antara kelas Kurir dan Kiriman.
4. Fungsionalitas dasar untuk pengelolaan data, seperti membuat, melihat, memperbarui status, dan menghapus data kiriman (CRUD), telah berjalan sesuai skenario pengujian. Sistem juga mampu menyimpan data secara persisten ke dalam file eksternal histori.txt.

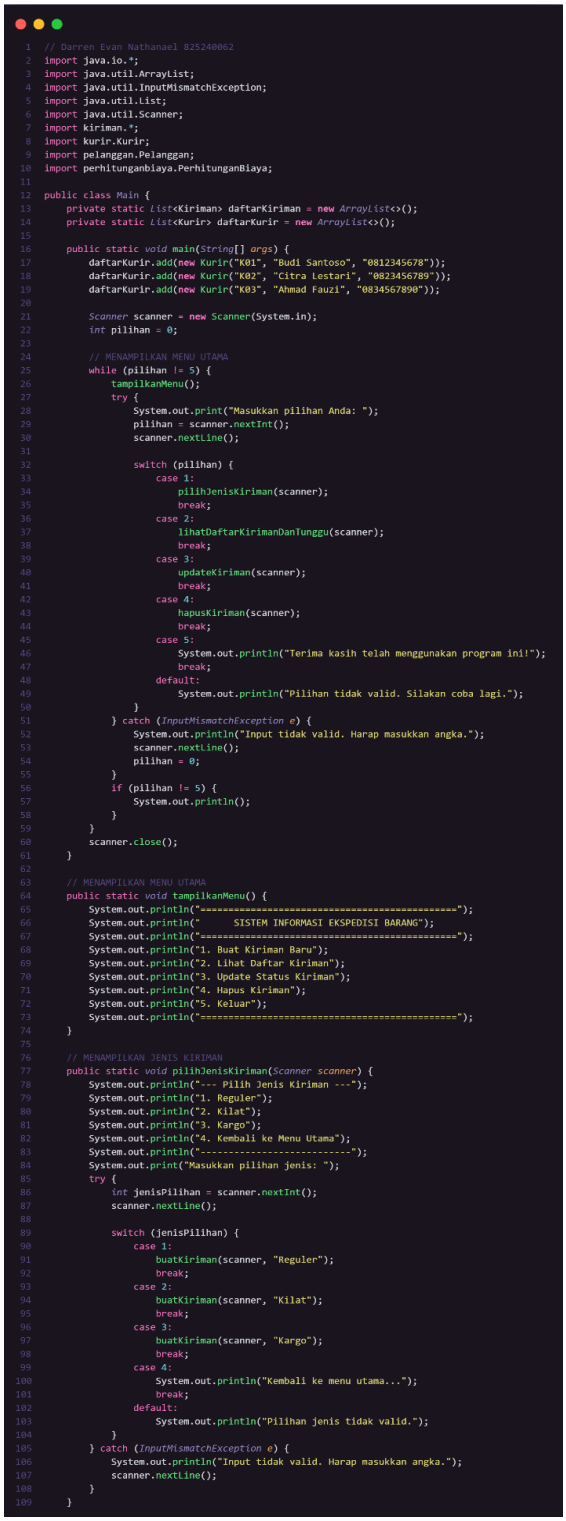
5.2 Saran

Meskipun sistem telah memenuhi tujuan awalnya, terdapat beberapa area yang dapat dikembangkan lebih lanjut untuk menjadikannya sebuah aplikasi yang lebih komprehensif dan fungsional. Berdasarkan batasan sistem yang telah diidentifikasi, berikut adalah beberapa saran untuk pengembangan di masa mendatang:

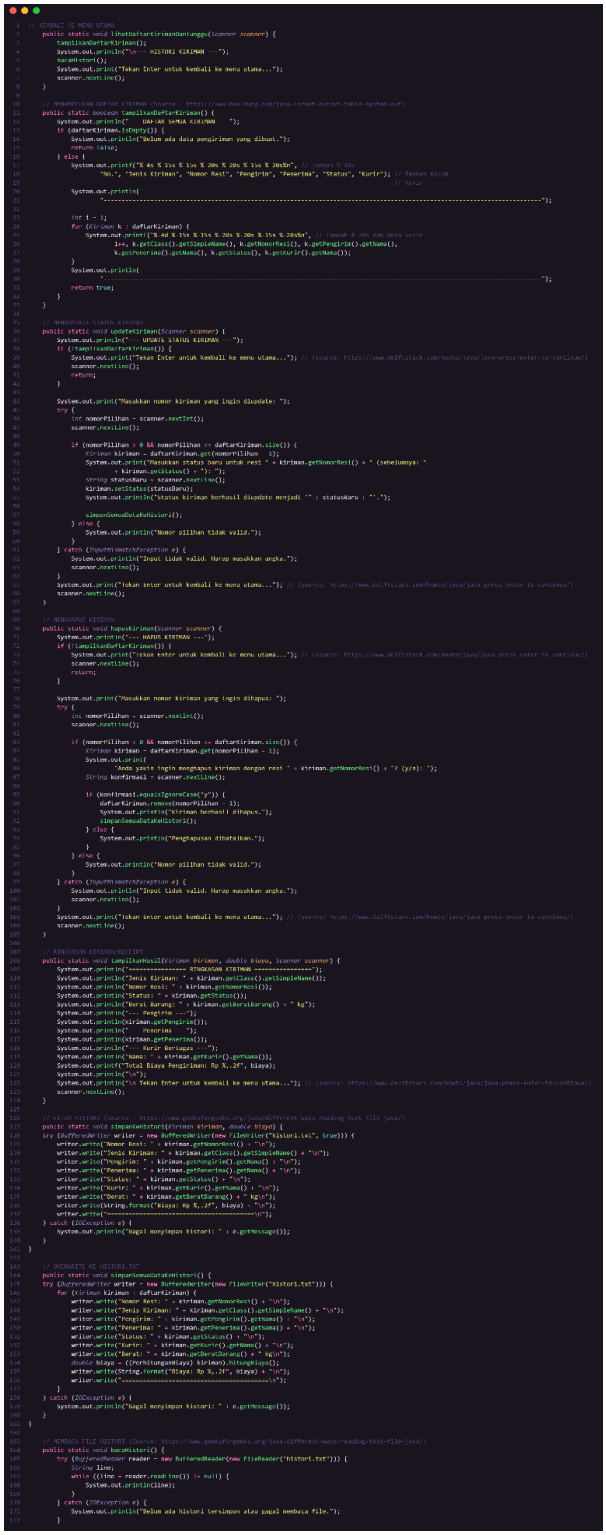
1. Menambahkan modul pelacakan paket (*real-time tracking*) agar pengguna dapat memantau posisi dan status kiriman secara langsung.
2. Mengembangkan fitur untuk menghasilkan resi dan label pengiriman secara otomatis setelah data kiriman baru berhasil dibuat, untuk meningkatkan efisiensi operasional.
3. Mengimplementasikan fitur manajemen permintaan penjemputan barang oleh kurir, sehingga pengguna dapat menjadwalkan penjemputan paket dari lokasi mereka.
4. Melakukan integrasi dengan API dari berbagai jasa kurir eksternal untuk menyediakan perbandingan layanan dan harga secara *real-time*, serta memungkinkan pengiriman notifikasi status secara proaktif kepada pengguna.

LAMPIRAN

Lampiran 3.1



Bagian 1



Bagian 2

Lampiran 3.2

```java
//Leticia Michelle Purba / 825240140

package kiriman;

import pelanggan.Pelanggan;
import kurir.Kurir;

public abstract class Kiriman {
    private String nomorResi;
    private String alamatTujuan;
    private String status;
    private double beratBarang;
    private Pelanggan pengirim;
    private Pelanggan penerima;
    private Kurir kurir;

    public Kiriman(String nomorResi, String alamatTujuan, double beratBarang, Pelanggan pengirim, Pelanggan penerima, String status, Kurir kurir) {
        this.nomorResi = nomorResi;
        this.alamatTujuan = alamatTujuan;
        this.beratBarang = beratBarang;
        this.pengirim = pengirim;
        this.penerima = penerima;
        this.status = status;
        this.kurir = kurir;
    }

    // SETTER AND GETTER
    public String getNomorResi() {
        return nomorResi;
    }

    public void setNomorResi(String nomorResi) {
        this.nomorResi = nomorResi;
    }

    public String getAlamatTujuan() {
        return alamatTujuan;
    }

    public void setAlamatTujuan(String alamatTujuan) {
        this.alamatTujuan = alamatTujuan;
    }

    public String getStatus() {
        return status;
    }

    public void setStatus(String status) {
        this.status = status;
    }

    public double getBeratBarang() {
        return beratBarang;
    }

    public void setBeratBarang(double beratBarang) {
        this.beratBarang = beratBarang;
    }

    public Pelanggan getPengirim() {
        return pengirim;
    }

    public void setPengirim(Pelanggan pengirim) {
        this.pengirim = pengirim;
    }

    public Pelanggan getPenerima() {
        return penerima;
    }

    public void setPenerima(Pelanggan penerima) {
        this.penerima = penerima;
    }

    public Kurir getKurir() {
        return kurir;
    }

    public void setKurir(Kurir kurir) {
        this.kurir = kurir;
    }
}
```

Lampiran 3.7

```java
// Darren Evan Nathanael 825240062

package kurir;

public class Kurir {
    private String idKurir;
    private String nama;
    private String nomorTelepon;

    public Kurir(String idKurir, String nama, String nomorTelepon) {
        this.idKurir = idKurir;
        this.nama = nama;
        this.nomorTelepon = nomorTelepon;
    }

    // --- GETTER & SETTER ---
    public String getIdKurir() {
        return idKurir;
    }

    public void setIdKurir(String idKurir) {
        this.idKurir = idKurir;
    }

    public String getNama() {
        return nama;
    }

    public void setNama(String nama) {
        this.nama = nama;
    }

    public String getNomorTelepon() {
        return nomorTelepon;
    }

    public void setNomorTelepon(String nomorTelepon) {
        this.nomorTelepon = nomorTelepon;
    }

    @Override
    public String toString() {
        return "ID: " + idKurir + ", Nama: " + nama;
    }
}
```

DAFTAR PUSTAKA

[1] Mordor Intelligence. (2025). *Indonesia e-commerce logistics market - size & share analysis, growth trends & forecasts (2024 - 2029)*. Indonesia eCommerce Logistics Market Size & Share Analysis - Industry Research Report - Growth Trends

[2] *Jurnal Intelek Insan Cendikia*, *1*(8). (Oktober 2024). Analisis faktor yang paling mempengaruhi keputusan pelanggan dalam memilih jasa ekspedisi. ANALISIS FAKTOR YANG PALING MEMPENGARUHI KEPUTUSAN PELANGGAN DALAM MEMILIH JASA EKSPEDISI | Jurnal Intelek Insan Cendikia

[3] FedEx. *Bagaimana biaya pengiriman dihitung*. Bagaimana perhitungan biaya pengiriman | FedEx Indonesia