PRAKTIKUM UAS
TIKET KERETA

Mata Kuliah:

PEMROGRAMAN OBRIETASI OBJEK

Oleh:

Gading Kent Sadewa

24091397031

2024B

PROGRAM STUDI D4 MANAJEMEN INFORMATIKA

FAKULTAS VOKASI

UNIVERSITAS NEGERI SURABAYA

2025

DAFTAR ISI

DAFTAR ISI II
BAB I PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Tujuan 1
1.3 Ruang Lingkup Proyek 1
1.4 Metodelogi Penelitian 1
BAB II PERANCANGAN APLIKASI 3
2.1 Diagram Class & Penjelasan 3
2.2 Alur Aplikasi / Flowchart 4
2.3 Perancangan Penyimpanan Data 5
BAB III IMPLEMENTASI KONSEP OOP 6
3.1 Implementasi Encapsulation 6
3.2 Implementasi Inheritance 6
3.3 Implementasi Polymorphisme 7
3.4 Implementasi Class Aplikasi Kereta 8
3.6 Implementasi GUI dan Interaksi Pengguna 8
BAB IV HASIL & PEMBAHASAN 8
4.1 Hasil Implementasi Aplikasi 8
4.2 Pembahasan Fitur Aplikasi 9
4.2.1 Fitur Nama Penumpang 9
4.2.2 Fitur Tanggal Keberangkatan 9
4.2.3 Fitur Jam Keberangkatan 10
4.2.4 Fitur Pilih Kereta 10
4.2.5 Fitur Piih Jumlah Tiket 10
4.2.5 Fitur Kelas Tiket 10
4.2.6 Fitur Pilih Bangku 11
4.2.7 Fitur Pesan Tiket 11
4.2.8 Fitur Reset 11
4.2.9 Fitur Riwayat Pemesanan 11
BAB V PENUTUP 12

5.1 Kesimpulan 12
5.2 Saran 12

BAB I PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi informasi telah membawa perubahan besar dalam berbagai bidang, salah satunya pada sektor transportasi. Proses pemesanan tiket kereta api yang sebelumnya dilakukan secara manual kini telah beralih ke sistem digital berbasis aplikasi. Sistem ini memberikan kemudahan bagi pengguna dalam memilih jadwal keberangkatan, kelas layanan, serta bangku penumpang secara mandiri dan efisien.

Dalam mata kuliah Pemrograman Berorientasi Objek (PBO), mahasiswa dituntut untuk memahami dan menerapkan konsep-konsep Object Oriented Programming (OOP) seperti encapsulation, inheritance, dan polymorphism ke dalam sebuah aplikasi nyata. Oleh karena itu, pada tugas Ujian Akhir Semester ini dikembangkan sebuah Sistem Pemesanan Tiket Kereta Api berbasis GUI menggunakan bahasa pemrograman Python dan library Tkinter.

Aplikasi ini dirancang untuk mensimulasikan proses pemesanan tiket kereta api secara sederhana namun fungsional, sehingga dapat membantu mahasiswa memahami penerapan konsep OOP dalam pengembangan perangkat lunak berbasis antarmuka grafis.

1.2 Tujuan

1. Menerapkan konsep Pemrograman Berorientasi Objek dalam sebuah aplikasi nyata.
2. Membuat aplikasi pemesanan tiket kereta api berbasis Graphical User Interface (GUI).
3. Mengimplementasikan fitur validasi input untuk mencegah kesalahan pengguna.
4. Mengimplementasikan fitur pemilihan bangku penumpang secara visual.
5. Melatih kemampuan mahasiswa dalam merancang, mengimplementasikan, dan mendokumentasikan aplikasi berbasis OOP.

1.3 Ruang Lingkup Proyek

1. Input data penumpang (nama dan tanggal keberangkatan)
2. Pemilihan jam keberangkatan dan kereta
3. Pemilihan kelas tiket (Ekonomi, Bisnis, dan Eksekutif)
4. Pemilihan bangku penumpang menggunakan GUI
5. Perhitungan total harga tiket secara otomatis
6. Penampilan struk pemesanan dan riwayat transaksi
7. Fitur reset untuk mengembalikan aplikasi ke kondisi awal

1.4 Metodelogi Penelitian

Metodologi penelitian yang digunakan dalam pengembangan aplikasi Sistem Pemesanan Tiket Kereta Api ini adalah metode pengembangan perangkat lunak sekuensial (sequential development) yang disesuaikan

dengan kebutuhan akademik pada mata kuliah Pemrograman Berorientasi Objek. Metode ini dipilih karena memiliki tahapan yang jelas dan sistematis sehingga memudahkan dalam proses perancangan, implementasi, serta dokumentasi aplikasi. Adapun tahapan metodologi penelitian yang dilakukan adalah sebagai berikut:

1. Analisis Kebutuhan
   Pada tahap ini dilakukan identifikasi kebutuhan sistem berdasarkan ketentuan tugas Ujian Akhir Semester. Analisis mencakup penentuan fitur-fitur utama aplikasi, seperti input data penumpang, pemilihan jadwal dan kereta, pemilihan kelas tiket, pemilihan bangku, perhitungan harga tiket, serta penampilan hasil dan riwayat pemesanan. Selain itu, ditentukan pula konsep Object-Oriented Programming (OOP) yang akan diterapkan, yaitu encapsulation, inheritance, dan polymorphism.

2. Perancangan Sistem
   Tahap perancangan dilakukan untuk menggambarkan struktur dan alur sistem sebelum diimplementasikan ke dalam kode program. Pada tahap ini disusun diagram class untuk menunjukkan hubungan antar kelas, serta alur aplikasi (flowchart) untuk menggambarkan proses interaksi pengguna dengan sistem. Perancangan antarmuka pengguna (GUI) juga dilakukan agar aplikasi mudah digunakan dan memiliki alur yang jelas.

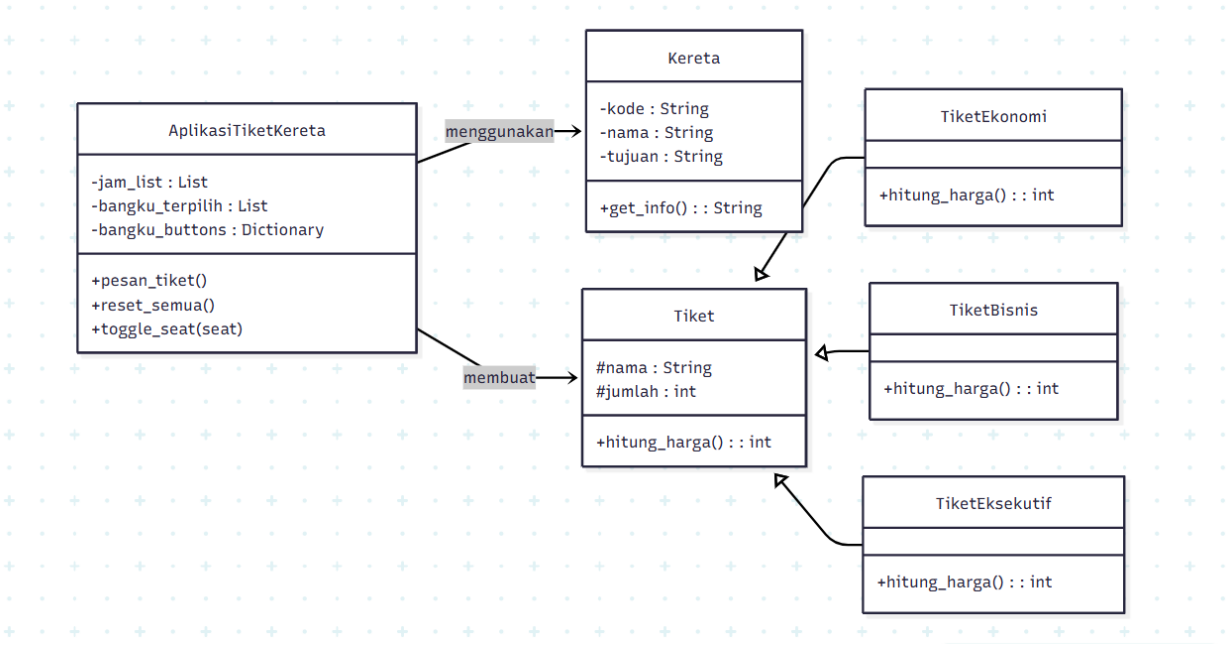
3. Implementasi
   Pada tahap ini dilakukan proses pengkodean aplikasi menggunakan bahasa pemrograman Python dengan menerapkan konsep Object-Oriented Programming. Antarmuka grafis dikembangkan menggunakan library Tkinter. Setiap fitur diimplementasikan sesuai dengan rancangan yang telah dibuat, termasuk validasi input, pemrosesan pemesanan tiket, serta pengelolaan data pemesanan dan riwayat transaksi.

4. Pengujian
   Tahap pengujian dilakukan untuk memastikan seluruh fitur aplikasi berjalan dengan baik dan sesuai dengan kebutuhan. Pengujian dilakukan dengan cara mencoba setiap fungsi, seperti input data penumpang, pemilihan kelas tiket, pemilihan bangku, pemesanan tiket, dan reset pemesanan. Pengujian ini bertujuan untuk memastikan tidak terdapat kesalahan logika (bug) serta memastikan validasi berjalan dengan benar.

5. Dokumentasi
   Tahap akhir adalah penyusunan dokumentasi aplikasi yang mencakup penjelasan perancangan sistem, penerapan konsep OOP, alur aplikasi, serta pembahasan fitur-fitur yang telah diimplementasikan. Dokumentasi ini bertujuan sebagai laporan akademik sekaligus sebagai referensi untuk pengembangan aplikasi di masa depan.

BAB II PERANCANGAN APLIKASI

2.1 Diagram Class & Penjelasan



Penjelasan:

Class diagram pada aplikasi Sistem Pemesanan Tiket Kereta Api menggambarkan struktur sistem secara keseluruhan serta hubungan antar kelas yang saling berinteraksi dalam menjalankan fungsi aplikasi. Diagram ini dirancang untuk merepresentasikan penerapan konsep Object-Oriented Programming (OOP) secara jelas, yang meliputi encapsulation, inheritance, dan polymorphism. Pada sistem ini terdapat beberapa kelas utama, yaitu Kereta, Tiket, TiketEkonomi, TiketBisnis, TiketEksekutif, dan AplikasiTiketKereta, yang masing-masing memiliki peran dan tanggung jawab berbeda.

Kelas Kereta berfungsi untuk merepresentasikan data kereta api yang tersedia dalam sistem. Kelas ini menyimpan informasi penting seperti kode kereta, nama kereta, dan tujuan perjalanan. Seluruh atribut pada kelas Kereta bersifat private sebagai bentuk penerapan encapsulation, sehingga data hanya dapat diakses melalui method tertentu. Method get_info() digunakan untuk menampilkan informasi kereta dalam bentuk teks yang nantinya digunakan pada antarmuka aplikasi saat pengguna memilih kereta.

Kelas Tiket merupakan kelas dasar (parent class) yang merepresentasikan konsep umum tiket pemesanan. Kelas ini menyimpan data nama penumpang dan jumlah tiket yang dipesan. Atribut pada kelas Tiket bersifat protected agar dapat diakses oleh kelas turunannya. Selain itu, kelas Tiket memiliki method hitung_harga() yang berfungsi sebagai method umum untuk perhitungan harga tiket. Method ini nantinya akan dioverride oleh kelas turunan sesuai dengan jenis kelas tiket yang dipilih.

Kelas TiketEkonomi, TiketBisnis, dan TiketEksekutif merupakan kelas turunan dari kelas Tiket yang menerapkan konsep inheritance. Ketiga kelas ini melakukan override terhadap method hitung_harga() untuk menentukan harga tiket berdasarkan kelas layanan yang berbeda. Perbedaan implementasi method yang sama pada masing-masing kelas menunjukkan penerapan konsep polymorphism, di mana sistem dapat memanggil method hitung_harga() yang sama namun menghasilkan nilai yang berbeda sesuai dengan jenis objek tiket yang digunakan.

Kelas AplikasiTiketKereta merupakan kelas utama yang berperan sebagai pengelola sistem dan antarmuka pengguna (Graphical User Interface). Kelas ini menangani seluruh alur aplikasi, mulai dari input data penumpang, pemilihan kereta dan jadwal keberangkatan, pemilihan kelas tiket, hingga pemilihan bangku penumpang secara visual. Selain itu, kelas ini juga bertanggung jawab dalam melakukan validasi data, memproses pemesanan tiket, menampilkan struk pemesanan, serta menyimpan riwayat transaksi.

Hubungan antara kelas AplikasiTiketKereta dengan kelas Kereta bersifat asosiasi, di mana AplikasiTiketKereta menggunakan data dari kelas Kereta untuk menampilkan pilihan kereta kepada pengguna. Sementara itu, hubungan antara AplikasiTiketKereta dan kelas Tiket bersifat dependensi, karena AplikasiTiketKereta membuat objek Tiket sesuai dengan kelas tiket yang dipilih oleh pengguna (Ekonomi, Bisnis, atau Eksekutif).

Secara keseluruhan, class diagram yang dirancang telah mencerminkan struktur aplikasi ATM Simulator yang modular dan terorganisir. Setiap kelas memiliki tanggung jawab yang jelas sehingga memudahkan proses pengembangan, pemeliharaan, serta pengembangan fitur di masa depan. Diagram ini juga menunjukkan bahwa aplikasi telah menerapkan prinsip-prinsip dasar OOP secara konsisten dan sesuai dengan kebutuhan sistem. Secara keseluruhan, class diagram yang dirancang telah mencerminkan struktur aplikasi Sistem Pemesanan Tiket Kereta Api yang terorganisir dan modular. Setiap kelas memiliki tanggung jawab yang jelas sehingga memudahkan proses pengembangan, pemeliharaan, dan pengembangan fitur di masa depan. Diagram ini juga menunjukkan bahwa aplikasi telah menerapkan prinsip-prinsip dasar Pemrograman Berorientasi Objek secara konsisten dan sesuai dengan kebutuhan sistem.

2.2 Alur Aplikasi / Flowchart

Alur aplikasi Sistem Pemesanan Tiket Kereta Api dimulai ketika pengguna menjalankan aplikasi dan sistem menampilkan halaman utama pemesanan tiket. Pada tahap awal ini, pengguna diminta untuk mengisi data penumpang, seperti nama penumpang dan jumlah tiket yang akan dipesan. Sistem akan melakukan validasi untuk memastikan seluruh data wajib telah diisi sebelum pengguna dapat melanjutkan ke tahap berikutnya.

Setelah data penumpang diisi, pengguna memilih informasi perjalanan yang meliputi kereta, tujuan, tanggal keberangkatan, jam keberangkatan, serta kelas tiket yang tersedia. Sistem menampilkan pilihan-pilihan tersebut melalui antarmuka grafis berbasis GUI. Apabila terdapat data yang belum dipilih atau tidak valid, sistem akan menampilkan pesan peringatan dan mengarahkan pengguna untuk melengkapi kembali data pemesanan.

Pada tahap selanjutnya, pengguna melakukan pemilihan bangku penumpang sesuai dengan jumlah tiket yang dipesan. Sistem hanya mengizinkan pengguna memilih bangku sebanyak jumlah tiket yang telah ditentukan. Setiap bangku yang dipilih akan ditandai secara visual untuk memudahkan pengguna dalam melihat status bangku yang tersedia dan yang telah terpilih.

Setelah seluruh data pemesanan dan bangku penumpang telah dipilih dengan benar, pengguna dapat menekan tombol pesan tiket. Sistem kemudian memproses pemesanan dengan menghitung total harga tiket berdasarkan kelas tiket dan jumlah penumpang. Jika proses pemesanan berhasil, sistem akan menampilkan struk pemesanan yang berisi detail transaksi, seperti data penumpang, informasi kereta, jam keberangkatan, bangku yang dipilih, serta total harga yang harus dibayar.

Setiap pemesanan yang berhasil akan secara otomatis disimpan ke dalam riwayat transaksi. Riwayat ini dapat dilihat kembali oleh pengguna selama aplikasi masih berjalan. Selain itu, sistem juga menyediakan fitur reset yang memungkinkan pengguna untuk menghapus seluruh data input dan riwayat pemesanan, sehingga aplikasi kembali ke kondisi awal dan siap digunakan untuk pemesanan berikutnya.

Alur aplikasi diakhiri ketika pengguna menutup aplikasi. Dengan alur yang terstruktur dan sistematis ini, aplikasi Sistem Pemesanan Tiket Kereta Api dapat digunakan secara mudah dan intuitif, serta mencerminkan proses pemesanan tiket kereta secara sederhana namun fungsional.

2.3 Perancangan Penyimpanan Data

Perancangan penyimpanan data pada aplikasi Sistem Pemesanan Tiket Kereta Api dilakukan dengan memanfaatkan media penyimpanan sementara berupa struktur data di dalam memori selama aplikasi berjalan. Pendekatan ini dipilih karena aplikasi bersifat simulasi dan tidak memerlukan penyimpanan data permanen menggunakan basis data atau file eksternal. Dengan metode ini, pengelolaan data menjadi lebih sederhana dan sesuai dengan kebutuhan aplikasi berskala kecil hingga menengah.

Data yang dikelola dalam aplikasi meliputi informasi pemesanan tiket, seperti nama penumpang, kereta yang dipilih, tujuan perjalanan, tanggal dan jam keberangkatan, kelas tiket, jumlah tiket, serta bangku penumpang yang dipilih. Selain itu, sistem juga menyimpan data riwayat pemesanan dalam bentuk daftar (list) yang berisi catatan setiap transaksi pemesanan yang berhasil dilakukan selama aplikasi berjalan.

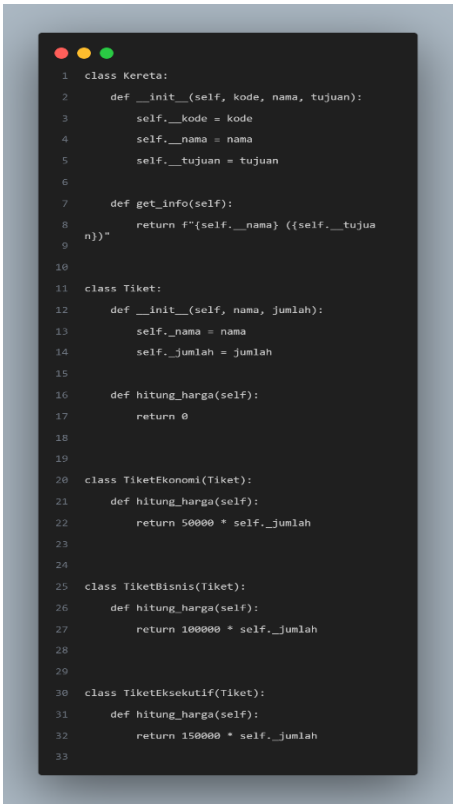
Penyimpanan data pemesanan dan riwayat transaksi dikelola langsung oleh class AplikasiTiketKereta menggunakan struktur data Python seperti list dan dictionary. Setiap kali pengguna berhasil melakukan pemesanan tiket, data pemesanan akan ditambahkan ke dalam daftar riwayat transaksi. Pendekatan ini memudahkan sistem dalam menampilkan kembali riwayat pemesanan tanpa perlu melakukan proses pembacaan data dari media penyimpanan eksternal.

Dengan memisahkan pengelolaan data dari tampilan antarmuka pengguna, aplikasi tetap menerapkan prinsip Object-Oriented Programming secara konsisten. Meskipun data tidak disimpan secara permanen setelah aplikasi ditutup, perancangan penyimpanan data ini sudah mencukupi untuk mendukung fungsi utama aplikasi dan memudahkan pengembangan lebih lanjut, seperti penambahan fitur penyimpanan berbasis file atau database di masa depan.

BAB III IMPLEMENTASI KONSEP OOP

3.1 Implementasi Encapsulation

Konsep encapsulation diterapkan untuk melindungi data penting pada aplikasi Simulasi ATM. Pada sistem ini, encapsulation diimplementasikan pada class Account, di mana atribut saldo dan PIN tidak diakses secara langsung dari luar class. Perubahan terhadap saldo hanya dapat dilakukan melalui method tertentu seperti setor dan tarik tunai. contoh implementasi encapsulation pada class Account:



```python
class Kereta:
    def __init__(self, kode, nama, tujuan):
        self.__kode = kode
        self.__nama = nama
        self.__tujuan = tujuan

    def get_info(self):
        return f"{self.__nama} ({self.__tujuan})"


class Tiket:
    def __init__(self, nama, jumlah):
        self._nama = nama
        self._jumlah = jumlah

    def hitung_harga(self):
        return 0


class TiketEkonomi(Tiket):
    def hitung_harga(self):
        return 50000 * self._jumlah


class TiketBisnis(Tiket):
    def hitung_harga(self):
        return 100000 * self._jumlah


class TiketEksekutif(Tiket):
    def hitung_harga(self):
        return 150000 * self._jumlah

```

Encapsulation pada kode di atas diterapkan dengan cara membatasi akses langsung terhadap atribut class. Pada class Kereta, atribut __kode, __nama, dan __tujuan dideklarasikan sebagai private menggunakan double underscore (__), sehingga tidak dapat diakses langsung dari luar class dan hanya dapat digunakan melalui method get_info(). Selain itu, pada class Tiket, atribut _nama dan _jumlah dideklarasikan sebagai protected, yang berarti hanya dapat diakses oleh class tersebut dan class turunannya (TiketEkonomi, TiketBisnis, dan TiketEksekutif). Dengan penerapan ini, data internal tetap terlindungi dan hanya dapat dimanipulasi melalui mekanisme yang telah ditentukan, sesuai dengan prinsip encapsulation dalam Object-Oriented Programming.

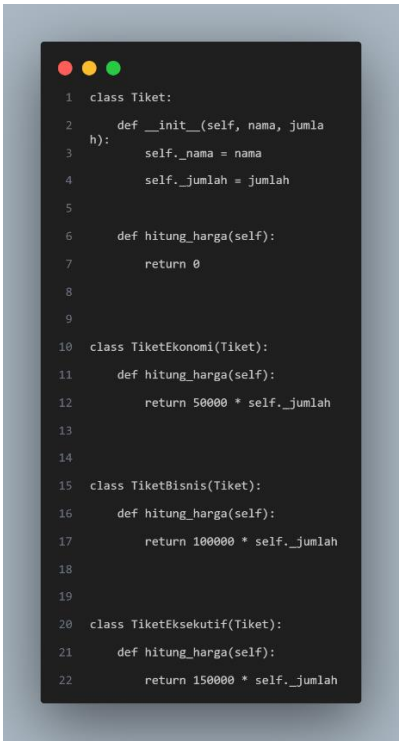
3.2 Implementasi Inheritance

```
class Kereta:
class Tiket:
class TiketEkonomi(Tiket):
class TiketBisnis(Tiket):
class TiketEksekutif(Tiket):
```

Ketiga class turunan tersebut mewarisi atribut _nama dan _jumlah serta method dari class Tiket, kemudian mengoverride method hitung_harga() sesuai dengan aturan harga masing-masing kelas tiket. Dengan adanya inheritance, struktur kode menjadi lebih efisien karena menghindari duplikasi kode dan memudahkan pengembangan sistem.

3.3 Implementasi Polymorphisme

Polymorphism memungkinkan sebuah method dengan nama yang sama memiliki perilaku yang berbeda tergantung pada objek yang memanggilnya. Pada aplikasi Sistem Pemesanan Tiket Kereta Api, polymorphism diimplementasikan melalui method overriding pada method hitung_harga() di class turunan dari class Tiket.



```
class Tiket:
    def __init__(self, nama, jumla
h):
        self._nama = nama
        self._jumlah = jumlah

    def hitung_harga(self):
        return 0


class TiketEkonomi(Tiket):
    def hitung_harga(self):
        return 50000 * self._jumlah


class TiketBisnis(Tiket):
    def hitung_harga(self):
        return 100000 * self._jumlah


class TiketEksekutif(Tiket):
    def hitung_harga(self):
        return 150000 * self._jumlah
```

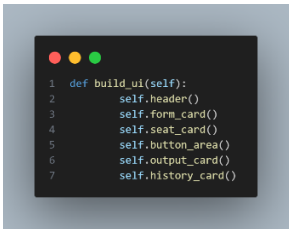
3.4 Implementasi Class Aplikasi Kereta

```
class AplikasiTiketKereta:
    def __init__(self, root):
        self.root = root
        self.root.title("Sistem Pemesanan Tiket Kereta")
        self.root.geometry("800x850")
        self.root.configure(bg="#EEF2F7")

        self.jam_list = ["06:00", "08:00", "10:00", "12:00",
                         "14:00", "16:00", "18:00", "20:00"]

        self.bangku_terpilih = []
        self.bangku_buttons = {}

        self.kereta_list = [
            Kereta("K01", "Argo Bromo", "Surabaya - Jakarta"),
            Kereta("K02", "Mutiara Timur", "Surabaya - Banyuwangi"),
            Kereta("K03", "Sancaka", "Surabaya - Yogyakarta")
        ]

        self.build_ui()

```

Class AplikasiTiketKereta berfungsi sebagai kelas utama yang mengelola seluruh alur aplikasi dan antarmuka pengguna berbasis GUI. Pada method __init__(), class ini melakukan inisialisasi jendela aplikasi, seperti pengaturan judul, ukuran tampilan, dan warna latar belakang. Selain itu, class ini juga menyiapkan data penting yang digunakan dalam proses pemesanan tiket, seperti daftar jam keberangkatan, daftar bangku yang dipilih, serta referensi tombol bangku. Objek kereta dibuat dalam bentuk list yang berisi beberapa instance dari class Kereta, sehingga memudahkan pengguna dalam memilih kereta yang tersedia. Setelah seluruh data awal disiapkan, method build_ui() dipanggil untuk membangun komponen antarmuka pengguna yang akan digunakan selama aplikasi berjalan.

3.6 Implementasi GUI dan Interaksi Pengguna

Method build_ui() berfungsi sebagai pengatur utama pembuatan antarmuka aplikasi. Method ini memanggil beberapa fungsi lain secara berurutan, yaitu header() untuk bagian judul, form_card() untuk form input pemesanan, seat_card() untuk pemilihan bangku, button_area() untuk tombol aksi, output_card() untuk menampilkan hasil pemesanan, dan history_card() untuk menampilkan riwayat pemesanan. Dengan pemisahan ini, struktur GUI menjadi lebih rapi dan mudah dikelola.



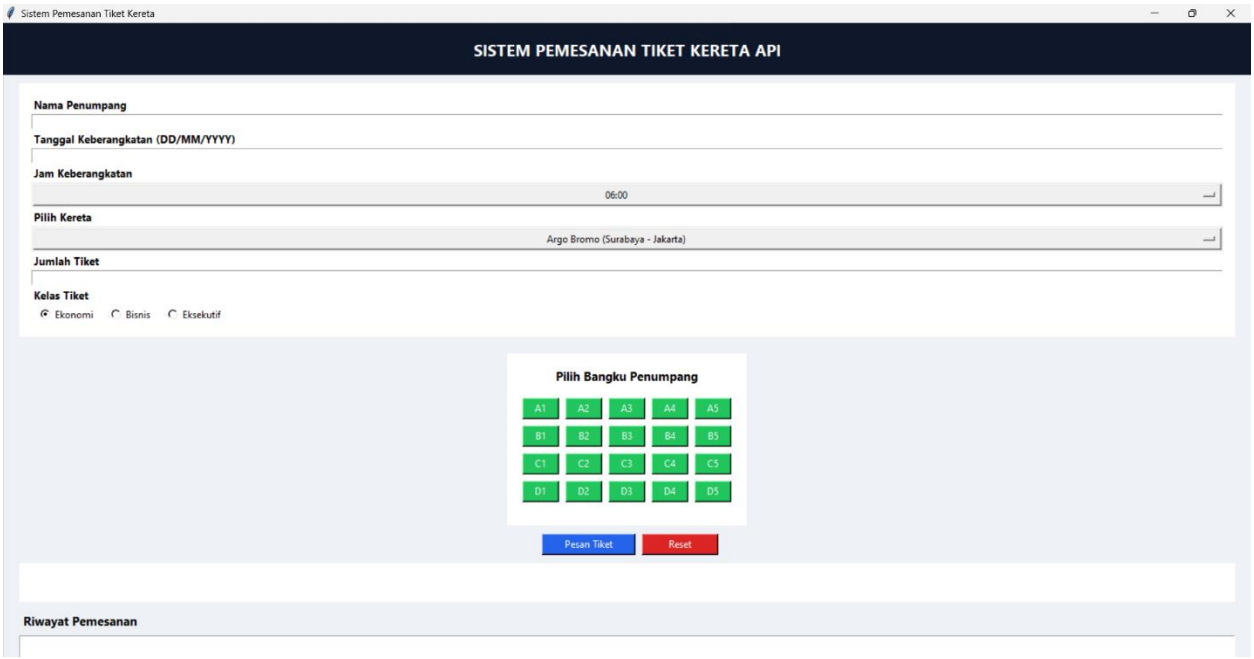
```
def build_ui(self):
        self.header()
        self.form_card()
        self.seat_card()
        self.button_area()
        self.output_card()
        self.history_card()
```

BAB IV HASIL & PEMBAHASAN

4.1 Hasil Implementasi Aplikasi

Pada bab ini dibahas hasil implementasi dari aplikasi Sistem Pemesanan Tiket Kereta Api berbasis Object-Oriented Programming (OOP) yang dikembangkan menggunakan bahasa pemrograman Python dengan pustaka Tkinter sebagai antarmuka grafis. Aplikasi berhasil dijalankan sesuai dengan perencanaan dan mampu menjalankan fitur utama, seperti pengisian data penumpang, pemilihan jadwal dan kereta, pemilihan kelas tiket, pemilihan bangku penumpang, serta penampilan hasil pemesanan dan riwayat transaksi.

Berdasarkan hasil implementasi yang ditunjukkan pada Gambar 4.1, aplikasi menampilkan form pemesanan tiket yang lengkap dan mudah digunakan. Sistem melakukan validasi data sebelum pemesanan diproses dan menampilkan struk pemesanan setelah transaksi berhasil dilakukan. Selain itu, riwayat pemesanan ditampilkan untuk membantu pengguna melihat kembali data transaksi yang telah dilakukan. Secara keseluruhan, aplikasi telah berjalan dengan baik dan sesuai dengan tujuan pengembangan sistem.



4.2 Pembahasan Fitur Aplikasi

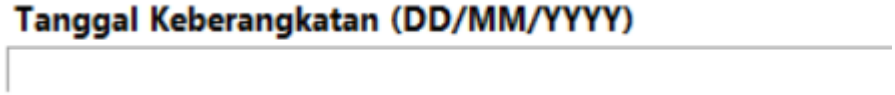
4.2.1 Fitur Nama Penumpang

Fitur Nama Penumpang digunakan untuk mencatat identitas penumpang yang melakukan pemesanan tiket kereta api. Pengguna diwajibkan mengisi nama penumpang pada kolom input yang tersedia sebelum melakukan pemesanan tiket. Sistem akan melakukan validasi untuk memastikan bahwa kolom nama tidak kosong. Apabila nama penumpang belum diisi, sistem akan menampilkan pesan peringatan dan proses pemesanan tidak dapat dilanjutkan. Data nama penumpang yang telah diinput akan ditampilkan pada struk pemesanan serta disimpan dalam riwayat transaksi, sehingga memudahkan identifikasi pemesanan yang telah dilakukan.

4.2.2 Fitur Tanggal Keberangkatan

Fitur Tanggal Keberangkatan digunakan untuk menentukan tanggal perjalanan kereta api yang akan dipesan oleh pengguna. Pengguna wajib mengisi tanggal keberangkatan pada kolom input yang telah disediakan sebelum melakukan pemesanan tiket. Sistem melakukan validasi untuk memastikan bahwa kolom tanggal tidak kosong. Apabila tanggal keberangkatan belum diisi, sistem akan menampilkan pesan peringatan dan proses pemesanan tiket tidak dapat dilanjutkan. Informasi

tanggal keberangkatan yang telah diinput akan ditampilkan pada struk pemesanan dan disimpan sebagai bagian dari data transaksi pemesanan.



4.2.3 Fitur Jam Keberangkatan

Fitur Fitur Jam Keberangkatan digunakan untuk memilih waktu keberangkatan kereta sesuai jadwal yang telah disediakan oleh sistem. Pengguna dapat memilih jam keberangkatan melalui komponen dropdown yang berisi daftar jam tertentu. Pemilihan jam ini bersifat wajib dan akan otomatis tersimpan sebagai bagian dari data pemesanan tiket. Jam keberangkatan yang dipilih akan ditampilkan pada struk pemesanan sebagai informasi perjalanan.

4.2.4 Fitur Pilih Kereta

Fitur Pilih Kereta memungkinkan pengguna memilih jenis kereta yang akan digunakan sesuai dengan rute perjalanan yang tersedia. Sistem menampilkan daftar kereta dalam bentuk dropdown yang berisi nama kereta dan tujuan perjalanan. Data kereta diambil dari objek kelas Kereta, sehingga penerapan konsep Object-Oriented Programming dapat terlihat. Kereta yang dipilih akan dicantumkan pada struk pemesanan tiket.

4.2.5 Fitur Piih Jumlah Tiket

Selama Fitur Jumlah Tiket digunakan untuk menentukan banyaknya tiket yang akan dipesan oleh pengguna. Input jumlah tiket harus berupa angka dan bernilai lebih dari nol. Sistem melakukan validasi untuk memastikan bahwa jumlah tiket telah diisi dengan benar. Jumlah tiket ini juga digunakan sebagai acuan dalam proses pemilihan bangku dan perhitungan total harga tiket.

4.2.5 Fitur Kelas Tiket

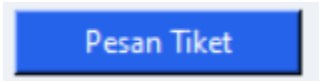
Fitur Kelas Tiket memungkinkan pengguna memilih kelas perjalanan, yaitu Ekonomi, Bisnis, atau Eksekutif. Setiap kelas memiliki harga yang berbeda dan diimplementasikan menggunakan konsep inheritance dan polymorphism melalui kelas TiketEkonomi, TiketBisnis, dan TiketEksekutif. Kelas yang dipilih akan menentukan perhitungan harga tiket secara otomatis.

4.2.6 Fitur Pilih Bangku

Fitur Pilih Bangku digunakan untuk memilih tempat duduk penumpang sesuai dengan jumlah tiket yang dipesan. Bangku ditampilkan dalam bentuk tombol-tombol yang merepresentasikan posisi kursi. Sistem membatasi jumlah bangku yang dapat dipilih sesuai dengan jumlah tiket dan memberikan peringatan jika melebihi batas. Bangku yang terpilih akan ditampilkan pada struk pemesanan tiket.

4.2.7 Fitur Pesan Tiket

Fitur Pesan Tiket digunakan untuk memproses pemesanan berdasarkan data yang telah diinput pengguna. Sistem akan melakukan validasi input, menghitung total harga sesuai kelas dan jumlah tiket, serta menampilkan hasil pemesanan dalam bentuk struk. Data pemesanan juga disimpan ke dalam riwayat pemesanan.



4.2.8 Fitur Reset

Fitur Reset Pemesanan digunakan untuk mengosongkan seluruh data input dan mengembalikan aplikasi ke kondisi awal. Saat tombol reset ditekan, sistem akan menghapus isian nama penumpang, tanggal keberangkatan, jumlah tiket, pilihan kelas, serta membersihkan bangku yang telah dipilih dan hasil pemesanan yang ditampilkan. Fitur ini memudahkan pengguna untuk melakukan pemesanan baru tanpa harus menutup dan membuka ulang aplikasi.

4.2.9 Fitur Riwayat Pemesanan

Fitur ini menampilkan daftar pemesanan yang telah dilakukan sebelumnya. Riwayat berisi detail transaksi sehingga pengguna dapat melihat kembali hasil pemesanan yang telah dibuat.

BAB V PENUTUP

5.1 Kesimpulan

Berdasarkan hasil perancangan, implementasi, dan pengujian yang telah dilakukan, dapat disimpulkan bahwa aplikasi Sistem Pemesanan Tiket Kereta Api berbasis Object-Oriented Programming (OOP) berhasil dikembangkan sesuai dengan tujuan yang telah ditetapkan. Aplikasi ini mampu mensimulasikan proses pemesanan tiket kereta api secara sederhana namun fungsional, mulai dari pengisian data penumpang, pemilihan jadwal dan kereta, pemilihan kelas tiket, pemilihan bangku, hingga penampilan hasil pemesanan dan riwayat transaksi.

Penerapan konsep OOP pada aplikasi telah dilakukan dengan baik dan terstruktur. Konsep encapsulation diterapkan melalui pembatasan akses terhadap atribut penting pada class Kereta dan Tiket. Konsep inheritance diterapkan melalui pewarisan class Tiket oleh class TiketEkonomi, TiketBisnis, dan TiketEksekutif. Sementara itu, konsep polymorphism diterapkan melalui mekanisme method overriding pada method hitung_harga(), sehingga sistem dapat menghitung harga tiket secara fleksibel sesuai dengan kelas tiket yang dipilih.

Penggunaan library Tkinter sebagai antarmuka grafis memudahkan interaksi pengguna dengan aplikasi. Selain itu, pemisahan fungsi antar class membuat struktur program menjadi lebih modular, mudah dipahami, dan mudah dikembangkan di masa mendatang. Dengan demikian, aplikasi ini tidak hanya memenuhi kebutuhan fungsional, tetapi juga berhasil menjadi media pembelajaran yang efektif dalam memahami dan menerapkan konsep Pemrograman Berorientasi Objek secara nyata.

5.2 Saran

Meskipun aplikasi Sistem Pemesanan Tiket Kereta Api telah berjalan dengan baik dan mampu memenuhi tujuan pembelajaran, masih terdapat beberapa hal yang dapat dikembangkan untuk meningkatkan kualitas aplikasi di masa mendatang. Salah satu saran pengembangan adalah menambahkan penyimpanan data permanen menggunakan basis data atau file eksternal agar data pemesanan dan riwayat transaksi tetap tersimpan meskipun aplikasi ditutup.

Selain itu, validasi input dapat ditingkatkan, khususnya pada format tanggal keberangkatan agar lebih terstruktur dan mengurangi kemungkinan kesalahan pengguna. Dari sisi antarmuka, tampilan GUI dapat dikembangkan dengan desain yang lebih modern dan responsif agar pengalaman pengguna menjadi lebih nyaman dan menarik.

Pengembangan selanjutnya juga dapat mencakup penambahan fitur lanjutan, seperti pemilihan gerbong, informasi ketersediaan bangku secara real-time, serta pencetakan atau penyimpanan struk pemesanan. Dengan adanya pengembangan tersebut, aplikasi ini diharapkan dapat menjadi media pembelajaran OOP yang lebih komprehensif serta memiliki nilai fungsional yang lebih tinggi.