PRAKTIKUM UAS
SIMULASI ATM

Mata Kuliah:

PEMROGRAMAN OBRIETASI OBJEK

Oleh:

Afta Wildana Zakcy

24091397038

2024B

PROGRAM STUDI D4 MANAJEMEN INFORMATIKA

FAKULTAS VOKASI

UNIVERSITAS NEGERI SURABAYA

2025

DAFTAR ISI

BAB I PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi informasi saat ini semakin pesat dan telah mempengaruhi berbagai aspek kehidupan, termasuk di bidang perbankan. Salah satu layanan perbankan yang banyak digunakan oleh masyarakat adalah Anjungan Tunai Mandiri (ATM). ATM memungkinkan nasabah untuk melakukan berbagai transaksi seperti pengecekan saldo, penyetoran, penarikan, dan transfer dana secara mandiri tanpa harus datang ke teller bank.

Dalam dunia pendidikan, khususnya pada mata kuliah Pemrograman Berorientasi Objek (Object-Oriented Programming / OOP)**,** mahasiswa dituntut untuk tidak hanya memahami konsep secara teori, tetapi juga mampu menerapkannya dalam sebuah aplikasi nyata. Penerapan konsep OOP seperti encapsulation, inheritance, dan polymorphism sangat penting untuk menghasilkan aplikasi yang terstruktur, mudah dipelihara, dan mudah dikembangkan.

Oleh karena itu, pada proyek ini dikembangkan sebuah aplikasi simulasi ATM berbasis GUI menggunakan bahasa pemrograman Python dan library Tkinter. Aplikasi ini dipilih karena memiliki alur yang jelas, fitur yang umum ditemui di dunia nyata, serta sangat cocok untuk mengimplementasikan konsep-konsep OOP secara nyata dan terukur.

1.2 Tujuan

- Tujuan Umum

  Tujuan umum dari proyek ini adalah untuk merancang dan membangun sebuah aplikasi simulasi ATM berbasis GUI menggunakan bahasa pemrograman Python dengan menerapkan konsep Object-Oriented Programming (OOP**)** secara sistematis, sehingga mahasiswa mampu memahami dan mengimplementasikan prinsip OOP dalam pengembangan aplikasi nyata.

- Tujuan Spesifik

  Adapun tujuan spesifik dari pembuatan aplikasi simulasi ATM ini adalah sebagai berikut:

1. Menerapkan konsep encapsulation dalam pengelolaan data akun, saldo, dan riwayat transaksi.

2. Mengimplementasikan struktur program berbasis class dan object untuk merepresentasikan sistem ATM.
3. Mengembangkan antarmuka grafis (GUI) menggunakan Tkinter agar aplikasi mudah digunakan oleh pengguna.
4. Mengimplementasikan fitur utama ATM, yaitu:
    - Pembuatan akun (registrasi),
    - Login pengguna,
    - Cek saldo,
    - Setor tunai,
    - Tarik tunai,
    - Transfer antar rekening,
    - Riwayat transaksi.
5. Mengelola penyimpanan data secara lokal menggunakan file JSON sebagai media penyimpanan data akun dan transaksi.
6. Melatih kemampuan analisis, perancangan, dan implementasi aplikasi berbasis OOP sesuai dengan kebutuhan akademik.
7. Menyusun dokumentasi proyek yang mencakup penjelasan konsep OOP, diagram class, alur aplikasi, serta bukti implementasi dalam bentuk screenshot.

1.3 Ruang Lingkup Proyek

Agar pembahasan dan implementasi proyek tetap terfokus, maka ruang lingkup dari aplikasi simulasi ATM ini dibatasi sebagai berikut:

1. Aplikasi hanya merupakan simulasi, bukan sistem perbankan sesungguhnya.
2. Data rekening, saldo, dan riwayat transaksi disimpan secara lokal menggunakan file JSON, tanpa menggunakan database server.
3. Aplikasi dikembangkan menggunakan:
    - Bahasa pemrograman Python,
    - Library Tkinter untuk antarmuka grafis.
4. Fitur yang disediakan dalam aplikasi meliputi:
    - Pembuatan akun (register),
    - Login pengguna,

- Cek saldo,
- Setor dan tarik tunai,
- Transfer antar rekening,
- Menampilkan riwayat transaksi.

5. Aplikasi tidak membahas aspek keamanan tingkat lanjut seperti enkripsi data, jaringan, atau integrasi sistem perbankan nyata.

1.4 Metodelogi Penelitian

Metodologi pengembangan aplikasi yang digunakan dalam proyek ini adalah metode pengembangan sederhana berbasis tahapan (sequential development) yang disesuaikan dengan kebutuhan akademik. Tahapan pengembangan aplikasi meliputi:

1. Analisis Kebutuhan
   Pada tahap ini dilakukan identifikasi kebutuhan aplikasi berdasarkan ketentuan tugas, meliputi fitur utama ATM dan penerapan konsep OOP.
2. Perancangan Sistem
   Tahap perancangan meliputi pembuatan:
   - Diagram class untuk menggambarkan hubungan antar kelas,
   - Alur aplikasi untuk menggambarkan interaksi pengguna dengan sistem.
3. Implementasi
   Pada tahap ini dilakukan pengkodean aplikasi menggunakan Python dengan menerapkan prinsip OOP dan antarmuka grafis menggunakan Tkinter.
4. Pengujian
   Aplikasi diuji dengan menjalankan setiap fitur untuk memastikan semua fungsi berjalan sesuai dengan yang direncanakan dan tidak terjadi kesalahan (bug).
5. Dokumentasi
   Tahap akhir adalah penyusunan dokumentasi yang mencakup penjelasan konsep OOP, diagram class, penjelasan fitur, serta screenshot hasil implementasi aplikasi.

BAB II PERANCANGAN APLIKASI

2.1 Diagram Class & Penjelasan

Penjelasan:

Class diagram pada aplikasi ATM Simulator menggambarkan struktur sistem secara keseluruhan serta hubungan antar kelas yang saling berinteraksi. Diagram ini dirancang untuk merepresentasikan penerapan konsep Object-Oriented Programming (OOP) secara jelas, meliputi

encapsulation, inheritance, dan polymorphism. Pada sistem ini terdapat empat kelas utama, yaitu `Account`, `SavingAccount`, `Bank`, dan `ATMApp`, yang masing-masing memiliki peran dan tanggung jawab berbeda dalam menjalankan fungsi aplikasi.

Kelas `Account` merupakan kelas dasar yang merepresentasikan akun pengguna dalam sistem ATM. Kelas ini menyimpan data penting seperti nomor rekening, PIN, saldo, dan riwayat transaksi. Seluruh atribut pada kelas ini bersifat terlindungi untuk menjaga keamanan data (encapsulation), sehingga akses terhadap saldo dan riwayat transaksi hanya dapat dilakukan melalui method tertentu seperti `get_balance()` dan `get_history()`. Selain itu, kelas `Account` menyediakan method `deposit()` untuk menambah saldo, `withdraw()` untuk melakukan penarikan, serta `add_history()` untuk mencatat setiap aktivitas transaksi yang dilakukan oleh pengguna.

Kelas `SavingAccount` merupakan turunan dari kelas `Account` yang menerapkan konsep inheritance. Kelas ini melakukan override pada method `withdraw()` untuk menambahkan aturan khusus berupa biaya administrasi setiap kali pengguna melakukan penarikan tunai. Dengan adanya overriding ini, sistem mampu menampilkan perilaku yang berbeda meskipun method yang digunakan memiliki nama yang sama, sehingga menunjukkan penerapan konsep polymorphism. Kelas `SavingAccount` digunakan ketika pengguna berhasil melakukan autentikasi, sehingga setiap transaksi penarikan akan otomatis mengikuti aturan yang telah ditentukan.

Kelas `Bank` berperan sebagai pengelola utama data akun dalam sistem. Kelas ini bertanggung jawab dalam menyimpan, memperbarui, dan memvalidasi data akun yang tersimpan dalam file JSON. Selain itu, kelas `Bank` menyediakan fungsi autentikasi melalui method `authenticate()`, pembuatan akun baru melalui `add_account()`, serta proses transfer antar rekening melalui method `transfer()`. Hubungan antara kelas `Bank` dan `Account` bersifat agregasi, di mana satu objek `Bank` dapat mengelola banyak objek `Account` tanpa bergantung langsung pada siklus hidupnya.

Kelas `ATMApp` merupakan kelas yang menangani antarmuka pengguna (Graphical User Interface) berbasis Tkinter. Kelas ini bertugas mengatur alur interaksi antara pengguna dan sistem, mulai dari proses login, pendaftaran akun, pengecekan saldo, penyetoran, penarikan, transfer, hingga penampilan riwayat transaksi. Kelas `ATMApp` memiliki relasi asosiasi dengan kelas `Bank` dan `Account`, karena kelas ini menggunakan layanan yang disediakan oleh kedua kelas tersebut

untuk menjalankan fungsi aplikasi. Dengan adanya kelas ini, seluruh logika bisnis dapat terhubung dengan antarmuka pengguna secara terstruktur dan mudah dipahami.

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.

2.2 Alur Aplikasi / Flowchart

Alur aplikasi ATM Simulator menggambarkan tahapan interaksi antara pengguna (user) dengan sistem sejak aplikasi dijalankan hingga pengguna keluar dari aplikasi. Alur ini dirancang agar mudah dipahami dan mencerminkan proses kerja sistem secara logis.

1. Proses Memulai Aplikasi

Aplikasi dijalankan melalui file utama Python. Sistem akan menampilkan halaman login sebagai tampilan awal aplikasi. Pada tahap ini, sistem belum memuat data akun ke dalam sesi pengguna, tetapi hanya menyiapkan antarmuka login.

2. Proses Login dan Registrasi

Pengguna memasukkan nomor rekening dan PIN pada form login. Sistem kemudian memanggil method authenticate() pada class Bank untuk mencocokkan data yang dimasukkan dengan data akun yang tersimpan di file JSON.

- Jika data valid, sistem membuat objek akun bertipe SavingAccount dan menyimpannya sebagai akun aktif.
- Jika data tidak valid, sistem menampilkan pesan kesalahan.
- Jika pengguna belum memiliki akun, pengguna dapat memilih menu registrasi. Sistem akan menyimpan data akun baru melalui method add_account().

3. Menampilkan Menu Utama

Setelah login berhasil, sistem menampilkan menu utama yang berisi beberapa fitur, yaitu:

- Cek saldo
- Setor tunai
- Tarik tunai
- Transfer
- Riwayat transaksi

- Logout

Menu ini menjadi pusat navigasi utama bagi pengguna selama menggunakan aplikasi.

4. Proses Cek Saldo

Ketika pengguna memilih menu cek saldo, sistem memanggil method get_balance() dari objek akun aktif. Saldo kemudian ditampilkan dalam bentuk pesan dialog tanpa mengubah data akun.

5. Proses Setor Tunai

Pada menu setor tunai, pengguna diminta memasukkan jumlah uang yang akan disetor. Sistem akan memanggil method deposit() pada objek akun untuk menambahkan saldo. Setelah itu, sistem memperbarui data akun ke dalam file JSON menggunakan method update_account().

6. Proses Tarik Tunai

Pada menu tarik tunai, pengguna memasukkan jumlah uang yang ingin ditarik. Sistem memanggil method withdraw() dari objek akun aktif. Karena akun menggunakan class SavingAccount, sistem menerapkan biaya administrasi secara otomatis melalui method overriding. Jika saldo mencukupi, transaksi berhasil dan data diperbarui; jika tidak, sistem menampilkan pesan kesalahan.

7. Proses Transfer Antar Rekening

Pada menu transfer, pengguna memasukkan nomor rekening tujuan dan jumlah transfer. Sistem memvalidasi rekening tujuan dan saldo pengirim. Jika valid, sistem mengurangi saldo pengirim dan menambahkan saldo ke rekening tujuan serta mencatat riwayat transaksi pada kedua akun.

8. Proses Melihat Riwayat Transaksi

Ketika pengguna memilih menu riwayat transaksi, sistem mengambil daftar riwayat melalui method get_history() dan menampilkannya dalam bentuk dialog. Riwayat ini berisi seluruh aktivitas transaksi yang pernah dilakukan pengguna.

9. Proses Logout

Pengguna dapat keluar dari akun dengan memilih menu logout. Sistem akan menghapus sesi akun aktif dan kembali menampilkan halaman login, sehingga aplikasi siap digunakan oleh pengguna lain.

2.3 Perancangan Penyimpanan Data

Perancangan penyimpanan data pada aplikasi Simulasi ATM ini menggunakan media penyimpanan berupa file dengan format JSON (JavaScript Object Notation). Pemilihan format JSON didasarkan pada kemudahan dalam membaca, menulis, dan mengelola data menggunakan

bahasa pemrograman Python. Format ini juga cocok digunakan untuk aplikasi simulasi berskala kecil hingga menengah yang tidak memerlukan sistem basis data kompleks.

Data yang disimpan dalam file JSON meliputi informasi akun pengguna, seperti nomor rekening, PIN, saldo, dan riwayat transaksi. Setiap akun disimpan sebagai satu objek yang memiliki atribut-atribut tersebut, sehingga memudahkan sistem dalam mengakses dan memperbarui data saat terjadi transaksi. Riwayat transaksi disimpan dalam bentuk daftar yang berisi catatan aktivitas, seperti setor tunai, tarik tunai, dan transfer, lengkap dengan keterangan jumlah transaksi.

Proses penyimpanan dan pembacaan data dilakukan secara terpisah melalui class khusus yang bertugas menangani operasi file. Dengan memisahkan mekanisme penyimpanan data dari logika utama aplikasi, sistem menjadi lebih modular dan menerapkan prinsip Object-Oriented Programming dengan baik. Data akan dimuat saat aplikasi dijalankan dan disimpan kembali setiap kali terjadi perubahan saldo atau transaksi, sehingga konsistensi data tetap terjaga.

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 Account:
    def __init__(self, acc_number, pin, balance=0, history=None):
        self._acc_number = acc_number
        self._pin = pin
        self._balance = balance
        self._history = history if history else []

    def deposit(self, amount):
        self._balance += amount
        self._history.append(f"Setor Rp {amount}")

    def withdraw(self, amount):
        if amount > self._balance:
            return False
        self._balance -= amount
        self._history.append(f"Tarik Rp {amount}")
        return True

    def add_history(self, text):
        self._history.append(text)

    def get_balance(self):
        return self._balance

    def get_history(self):
        return self._history
```

Pada kode di atas, atribut __saldo dan __pin bersifat private, sehingga tidak dapat diakses langsung dari luar class. Akses terhadap data tersebut dilakukan melalui method yang telah disediakan.

3.2 Implementasi Inheritance

Inheritance digunakan untuk menurunkan sifat dan perilaku dari sebuah class ke class lain. Pada aplikasi ini, class SavingAccount merupakan turunan dari class Account.

```
class SavingAccount(Account):
```

Dengan inheritance, class SavingAccount secara otomatis memiliki seluruh atribut dan method dari class Account, sehingga tidak perlu menuliskan ulang kode yang sama. Hal ini membuat kode lebih efisien dan mudah dikembangkan.

3.3 Implementasi Polymorphisme

Polymorphism memungkinkan method dengan nama yang sama memiliki perilaku yang berbeda tergantung pada objek yang memanggilnya. Pada aplikasi ATM Simulator, polymorphism diimplementasikan melalui method overriding pada method withdraw() di kelas SavingAccount.

```
class SavingAccount(Account):
    def withdraw(self, amount):
        admin_fee = 2000
        total = amount + admin_fee
        if total > self._balance:
            return False
        self._balance -= total
        self._history.append(
            f"Tarik Rp {amount} (Admin Rp {admin_fee})"
        )
        return True
```

Method withdraw() pada class SavingAccount merupakan bentuk method overriding, yang menunjukkan konsep polymorphism karena method yang sama memiliki perilaku berbeda pada class turunan.. Namun, pada kelas SavingAccount, method withdraw() dioverride untuk menambahkan biaya administrasi. Meskipun method yang dipanggil memiliki nama yang sama, sistem menjalankan logika yang berbeda sesuai dengan jenis objek yang digunakan. Hal ini menunjukkan penerapan polymorphism secara nyata dalam aplikasi.

Polymorphism juga tampak saat objek akun dibuat oleh sistem. Method authenticate() mengembalikan objek bertipe SavingAccount, tetapi diperlakukan sebagai objek Account.

```
def authenticate(self, acc_number, pin):
    data = self.accounts.get(acc_number)
    if data and data["pin"] == pin:
        return SavingAccount(
            acc_number,
            pin,
            data["balance"],
            data.get("history", [])
        )
    return None
```

Objek SavingAccount digunakan melalui referensi Account, tetapi tetap menjalankan method withdraw() versi SavingAccount.

3.4 Implementasi Kelas Bank sebagai Pengelola Data

Kelas Bank berfungsi sebagai pengelola data akun dan pusat logika bisnis sistem. Kelas ini bertanggung jawab dalam proses penyimpanan dan pembacaan data akun dari file JSON, autentikasi pengguna, pembuatan akun baru, pembaruan data akun, serta proses transfer antar rekening.

Melalui kelas Bank, seluruh data akun disimpan secara terpusat sehingga memudahkan pengelolaan dan pemeliharaan sistem. Kelas ini juga berperan sebagai penghubung antara data akun dan antarmuka pengguna.

3.6 Implementasi GUI dan Interaksi Pengguna

Antarmuka pengguna pada aplikasi ATM Simulator dikembangkan menggunakan pustaka Tkinter. Implementasi GUI dilakukan pada kelas ATMApp, yang bertugas menampilkan form login, menu utama, serta berbagai fitur transaksi seperti setor, tarik, transfer, cek saldo, dan riwayat transaksi.

```python
class ATMApp:
    def __init__(self, root):
        self.root = root
        self.root.title("ATM Simulator")
        self.bank = Bank()
        self.current_account = None
        self.login_screen()

    def clear(self):
        for w in self.root.winfo_children():
            w.destroy()
```

Setiap tombol pada GUI terhubung langsung dengan method tertentu sehingga memungkinkan interaksi dua arah antara pengguna dan sistem. Dengan desain ini, pengguna dapat menggunakan aplikasi secara intuitif tanpa perlu memahami proses internal sistem.

3.7 Implementasi Algoritma Sistem ATM

Algoritma utama dalam aplikasi ATM Simulator mencakup proses login, validasi data, pengolahan transaksi, serta pencatatan riwayat transaksi. Alur kerja sistem dimulai dari autentikasi pengguna, kemudian pengguna dapat memilih fitur yang diinginkan dari menu utama.

```python
def proses():
    try:
        ok, msg = self.bank.transfer(
            self.current_account,
            ent_rek.get(),
            int(ent_amt.get())
        )
        if ok:
            messagebox.showinfo("Sukses", msg)
            self.main_menu()
        else:
            messagebox.showerror("Error", msg)
    except:
        messagebox.showerror("Error", "Input tidak valid")
```

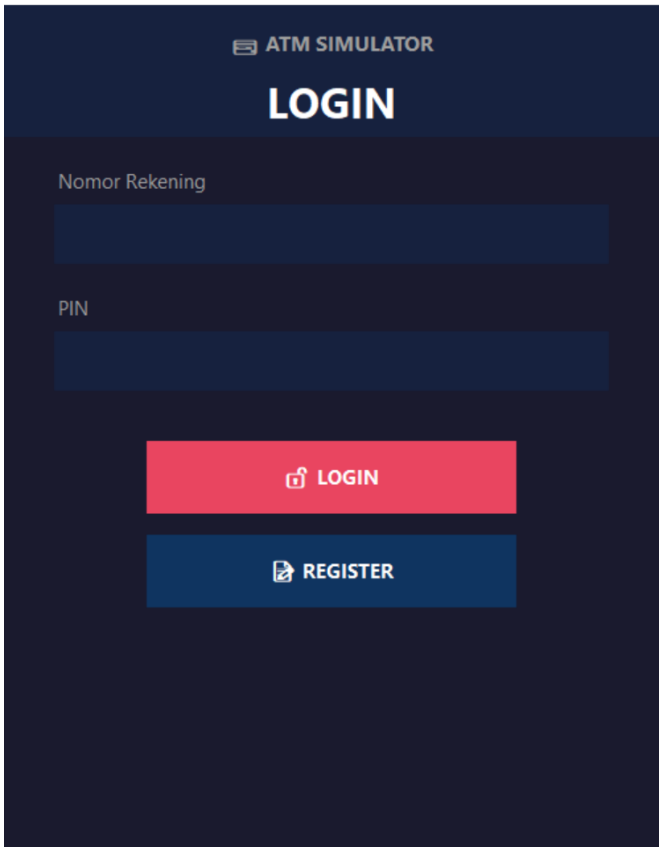
Setiap transaksi yang dilakukan akan memengaruhi saldo akun dan dicatat ke dalam riwayat transaksi. Untuk transaksi transfer, sistem melakukan pengecekan keberadaan rekening tujuan serta kecukupan saldo sebelum proses dilanjutkan. Algoritma ini memastikan setiap transaksi berjalan secara aman dan terkontrol.

BAB IV HASIL & PEMBAHASAN

4.1 Hasil Implementasi Aplikasi

Pada bab ini dibahas hasil implementasi dari aplikasi ATM Simulator Berbasis Object-Oriented Programming (OOP) yang telah dikembangkan menggunakan bahasa pemrograman Python dengan pustaka Tkinter untuk antarmuka grafis. Aplikasi berhasil dijalankan sesuai dengan perencanaan pada bab sebelumnya dan mampu menjalankan seluruh fitur utama seperti login, registrasi akun, setor tunai, tarik tunai, transfer antar rekening, pengecekan saldo, serta penampilan riwayat transaksi.

Secara umum, aplikasi telah memenuhi tujuan utama proyek, yaitu menerapkan konsep OOP secara nyata dalam sebuah aplikasi berbasis GUI. Setiap fitur dapat diakses melalui menu utama setelah pengguna berhasil melakukan proses login.



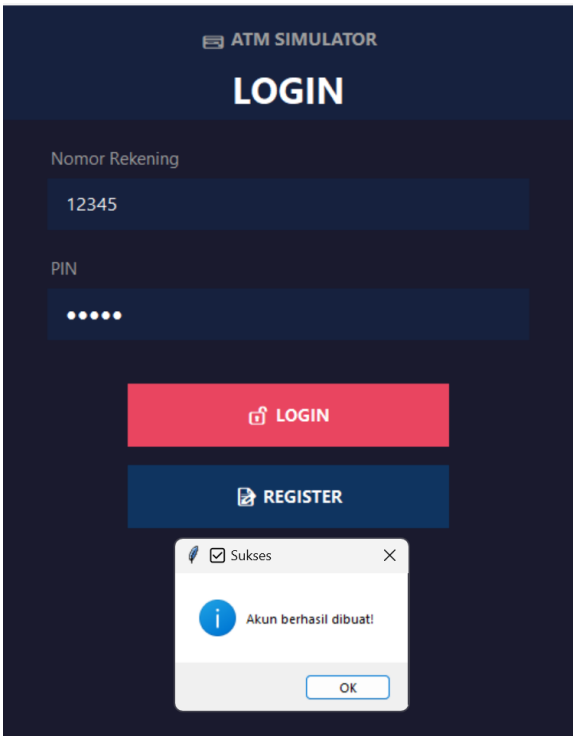
Gambar 4.1 Tampilan Halaman Login Aplikasi ATM Simulator

4.2 Pembahasan Fitur Aplikasi

4.2.1 Fitur Login dan Registrasi

Fitur login memungkinkan pengguna masuk ke sistem dengan memasukkan nomor rekening dan PIN. Proses autentikasi dilakukan oleh sistem dengan mencocokkan data yang dimasukkan pengguna dengan data yang tersimpan pada file JSON. Apabila data sesuai, pengguna akan diarahkan ke menu utama aplikasi.

Selain login, aplikasi juga menyediakan fitur registrasi untuk membuat akun baru. Data akun yang berhasil dibuat akan langsung disimpan ke dalam media penyimpanan lokal sehingga dapat digunakan kembali pada saat aplikasi dijalankan ulang.



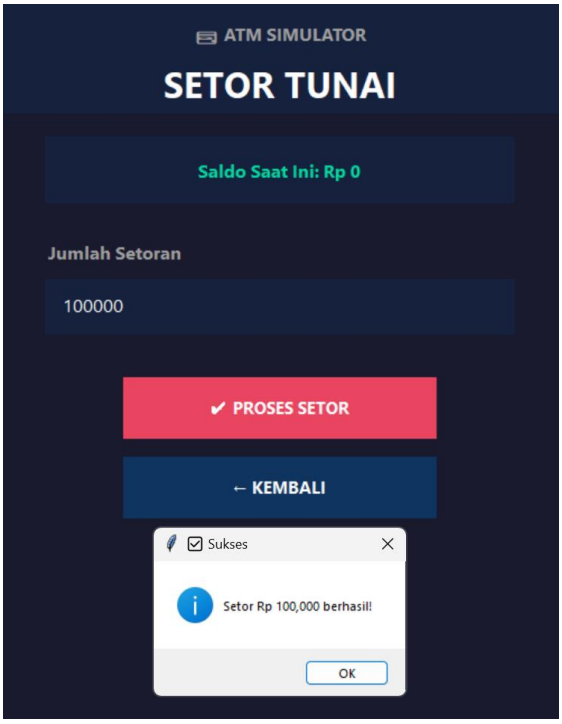
Gambar 4.2 Proses Registrasi Akun Baru

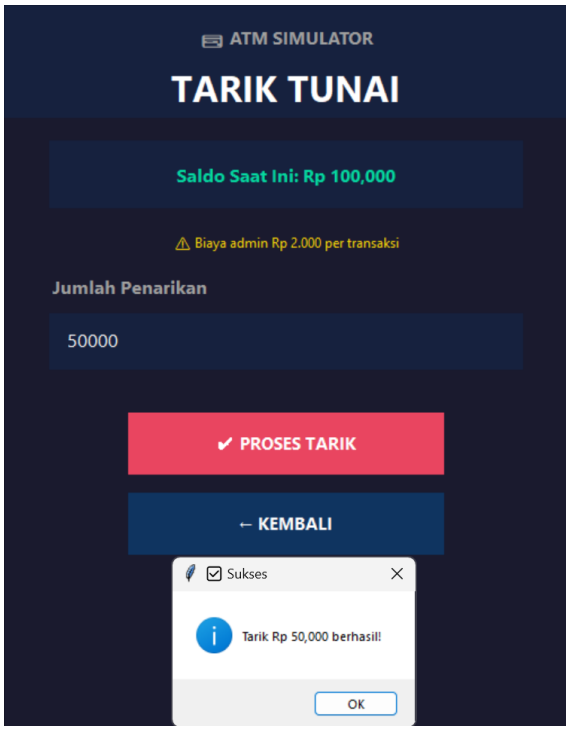
4.2.2 Fitur Setor dan Tarik Tunai

Fitur setor tunai memungkinkan pengguna menambahkan saldo dengan memasukkan jumlah uang yang diinginkan. Sistem akan memvalidasi input dan menambahkan saldo ke akun yang sedang aktif. Setiap transaksi setor akan dicatat ke dalam riwayat transaksi.

Fitur tarik tunai memungkinkan pengguna menarik saldo dari rekening. Pada implementasi polymorphism, proses tarik tunai pada kelas SavingAccount dikenakan biaya

administrasi, yang membedakan perilakunya dari kelas induk Account. Hal ini menunjukkan keberhasilan penerapan method overriding.



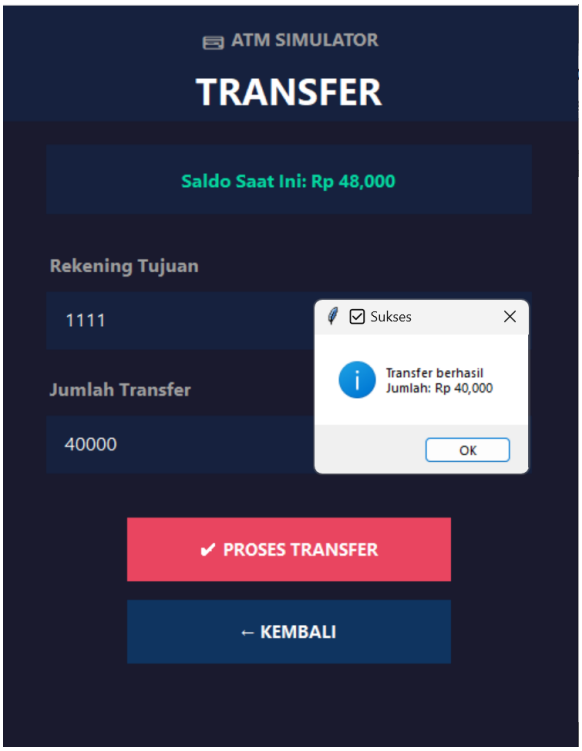


Gambar 4.3 Tampilan Menu Setor Tunai *Gambar 4.4 Tampilan Menu Tarik Tunai*

4.2.3 Fitur Transfer Antar Rekening

Fitur transfer memungkinkan pengguna mengirim saldo ke rekening lain yang terdaftar dalam sistem. Pengguna diminta memasukkan nomor rekening tujuan dan jumlah transfer. Sistem akan melakukan validasi terhadap saldo dan keberadaan rekening tujuan sebelum transaksi diproses.

Jika transaksi berhasil, saldo pengirim akan berkurang dan saldo penerima akan bertambah, serta kedua transaksi akan dicatat ke dalam riwayat masing-masing akun.

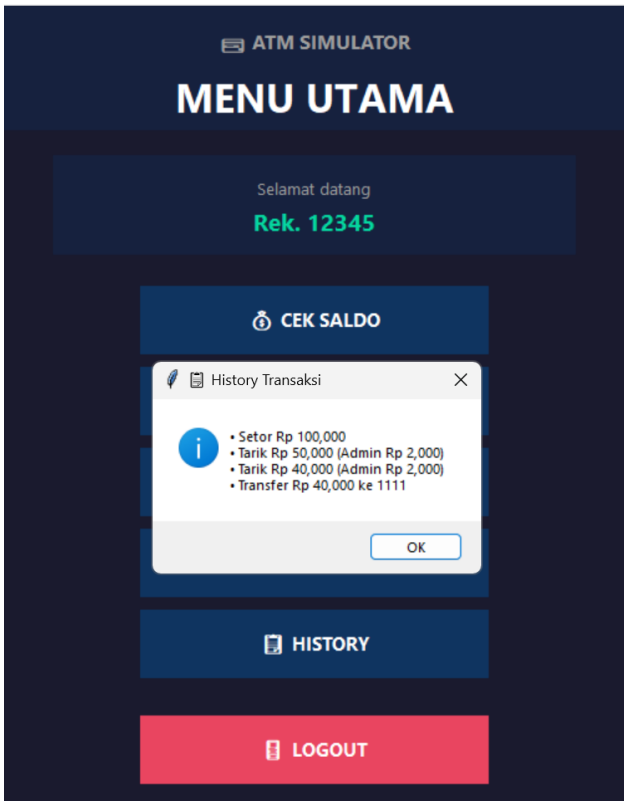


Gambar 4.5 Proses Transfer Antar Rekening

4.2.4 Fitur Cek Saldo dan Riwayat Transaksi

Fitur cek saldo digunakan untuk menampilkan jumlah saldo terkini dari akun yang sedang login. Informasi saldo ditampilkan melalui jendela dialog sehingga mudah dibaca oleh pengguna.

Fitur riwayat transaksi menampilkan daftar seluruh aktivitas keuangan yang pernah dilakukan oleh pengguna, seperti setor, tarik, dan transfer. Fitur ini membantu pengguna dalam memantau aktivitas rekening.



Gambar 4.6 Tampilan Riwayat Transaksi

4.3 Tantangan dan Kendala Pengembangan

Selama proses pengembangan aplikasi, terdapat beberapa tantangan yang dihadapi. Tantangan utama adalah memastikan data akun tetap tersimpan meskipun aplikasi ditutup. Hal ini diatasi dengan menggunakan file JSON sebagai media penyimpanan data lokal.

Selain itu, pengelolaan alur GUI juga menjadi tantangan tersendiri, terutama dalam mengatur perpindahan antar menu tanpa menumpuk tampilan sebelumnya. Masalah ini diselesaikan dengan menerapkan metode penghapusan widget sebelum menampilkan tampilan baru.

Tantangan lainnya adalah penerapan konsep OOP agar benar-benar terlihat nyata dan tidak hanya bersifat teori. Dengan memisahkan tanggung jawab kelas dan menerapkan inheritance serta polymorphism, tantangan ini dapat diatasi dengan baik.

BAB V PENUTUP

5.1 Kesimpulan

Berdasarkan hasil perancangan, implementasi, dan pengujian aplikasi ATM Simulator berbasis Object-Oriented Programming (OOP), dapat disimpulkan bahwa aplikasi ini berhasil dikembangkan sesuai dengan tujuan yang telah ditetapkan pada awal proyek. Aplikasi mampu menjalankan seluruh fitur utama, seperti login dan registrasi akun, setor tunai, tarik tunai, transfer antar rekening, pengecekan saldo, serta penampilan riwayat transaksi dengan baik dan stabil.

Penerapan konsep OOP pada aplikasi ini telah dilakukan secara nyata dan terstruktur. Konsep encapsulation diterapkan dengan membatasi akses langsung terhadap atribut penting seperti saldo dan PIN melalui penggunaan atribut protected dan method khusus. Konsep inheritance diterapkan melalui pewarisan kelas SavingAccount dari kelas Account, sehingga kelas turunan dapat menggunakan kembali atribut dan method dari kelas induk. Selanjutnya, konsep polymorphism diterapkan melalui method overriding pada fungsi withdraw(), yang memungkinkan perilaku penarikan saldo berbeda sesuai dengan jenis akun yang digunakan.

Selain itu, penggunaan Tkinter sebagai antarmuka grafis mempermudah interaksi pengguna dengan sistem. Pemisahan tanggung jawab antar kelas, seperti kelas Account, Bank, dan ATMApp, membuat struktur kode menjadi lebih modular, mudah dipahami, serta mudah dikembangkan di masa depan. Dengan demikian, aplikasi ini telah memenuhi kriteria penilaian baik dari sisi fungsionalitas maupun penerapan konsep OOP.

5.2 Saran

Meskipun aplikasi ATM Simulator ini telah berjalan dengan baik, masih terdapat beberapa peluang pengembangan yang dapat dilakukan untuk meningkatkan kualitas dan keamanan aplikasi. Salah satu saran pengembangan adalah penggunaan sistem basis data seperti MySQL atau PostgreSQL sebagai pengganti penyimpanan file JSON agar data lebih aman dan skalabel.

Selain itu, fitur keamanan dapat ditingkatkan dengan menambahkan enkripsi PIN, pembatasan jumlah percobaan login, serta pencatatan waktu transaksi secara lebih detail. Dari sisi antarmuka, tampilan GUI dapat diperbaiki dengan desain yang lebih modern dan responsif agar pengalaman pengguna menjadi lebih baik.

Pengembangan selanjutnya juga dapat mencakup penambahan jenis akun lain dengan aturan transaksi yang berbeda, sehingga penerapan inheritance dan polymorphism menjadi lebih

kompleks dan bervariasi. Dengan adanya pengembangan tersebut, aplikasi ini diharapkan dapat menjadi media pembelajaran OOP yang lebih komprehensif dan aplikatif.