Laporan Proyek Akhir Perancangan Aplikasi Daily Planner Berbasis Python Menggunakan Konsep Pemrograman Berorientasi Objek

Mata Kuliah:
Pemrograman Berorientasi Objek

Dosen Pengampu:
Rosita, S.Pd., M.Pd.

Oleh:
Firta Aulika Aji Kusuma
24091397029
2024B

PROGRAM STUDI D4 MANAJEMEN INFORMATIKA
FAKULTAS VOKASI
UNIVERSITAS NEGERI SURABAYA
2025

DAFTAR ISI

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Manajemen waktu merupakan salah satu keterampilan penting yang harus dimiliki oleh setiap individu, terutama mahasiswa. Dengan banyaknya aktivitas yang harus dilakukan setiap hari seperti kuliah, mengerjakan tugas, organisasi, dan kegiatan pribadi lainnya, sering kali terjadi tumpang tindih jadwal atau bahkan lupa akan kegiatan tertentu. Hal ini dapat menyebabkan penurunan produktivitas dan meningkatkan tingkat stres.

Berbagai aplikasi manajemen waktu telah tersedia, namun kebanyakan menggunakan tampilan list atau timeline linear yang kurang memberikan visualisasi menyeluruh tentang bagaimana waktu dalam sehari terbagi. Visualisasi yang kurang intuitif ini membuat pengguna kesulitan melihat pola penggunaan waktu mereka dan merencanakan jadwal dengan lebih efektif.

Aplikasi Daily Planner dikembangkan sebagai solusi untuk masalah tersebut dengan menawarkan pendekatan visualisasi yang berbeda. Aplikasi ini menggunakan jam analog 24 jam untuk menampilkan jadwal harian, memberikan perspektif visual yang unik dan memudahkan pengguna untuk melihat distribusi waktu mereka sepanjang hari. Setiap kegiatan ditampilkan sebagai segmen berwarna pada jam analog, sehingga pengguna dapat dengan cepat memahami bagaimana waktu mereka teralokasi.

Dari sisi teknis, proyek ini merupakan implementasi dari konsep-konsep Object-Oriented Programming (OOP) yang telah dipelajari dalam mata kuliah. Aplikasi ini mendemonstrasikan penerapan tiga pilar utama OOP yaitu Encapsulation, Inheritance, dan Polymorphism dalam konteks aplikasi nyata yang fungsional. Penggunaan Python sebagai bahasa pemrograman dan Tkinter sebagai framework GUI memungkinkan pembuatan aplikasi desktop yang ringan namun powerful.

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, rumusan masalah dalam proyek ini adalah:

1. Bagaimana merancang aplikasi manajemen jadwal dengan visualisasi jam analog 24 jam yang intuitif?
2. Bagaimana mengimplementasikan prinsip Encapsulation dalam struktur data jadwal?
3. Bagaimana menerapkan prinsip Inheritance untuk membuat hirarki class yang modular?
4. Bagaimana mendemonstrasikan Polymorphism dalam perhitungan dan visualisasi jarum jam?
5. Bagaimana mencegah konflik jadwal (overlap) secara otomatis?
6. Bagaimana menyimpan data jadwal secara persisten menggunakan file JSON?

1.3 Tujuan Proyek

Tujuan dari pengembangan aplikasi Daily Planner ini adalah:

1. Tujuan Umum:
    - Mengembangkan aplikasi desktop untuk manajemen jadwal harian dengan interface yang user-friendly
2. Tujuan Khusus:
    - Mengimplementasikan konsep Encapsulation dengan menggunakan protected attributes dan getter methods
    - Menerapkan Inheritance melalui pembuatan abstract base class dan concrete implementation class
    - Mendemonstrasikan Polymorphism melalui method overriding pada class-class jarum jam

    - Membuat sistem deteksi overlap jadwal yang reliable
    - Mengimplementasikan persistent storage menggunakan JSON
    - Membuat visualisasi jam analog 24 jam yang menarik dan informatif

1.4 Manfaat Proyek

Manfaat yang dapat diperoleh dari proyek ini adalah:

1. Manfaat Praktis:
    - Membantu pengguna mengatur jadwal harian dengan lebih terorganisir
    - Memberikan visualisasi yang unik dan mudah dipahami tentang distribusi waktu
    - Mencegah terjadinya konflik jadwal (double booking)
    - Menyediakan tool manajemen waktu yang dapat digunakan secara offline
2. Manfaat Akademis:
    - Memahami implementasi nyata dari konsep OOP dalam aplikasi praktis
    - Belajar membuat aplikasi GUI menggunakan Tkinter
    - Memahami penggunaan abstract base class dan inheritance hierarchy
    - Belajar implementasi data persistence dengan JSON
    - Memahami algoritma deteksi overlap interval waktu
3. Manfaat Pengembangan Skill:
    - Meningkatkan kemampuan problem solving
    - Melatih kemampuan design thinking dalam UI/UX
    - Mengembangkan skill debugging dan testing
    - Melatih kemampuan dokumentasi teknis

1.5 Batasan Proyek

Untuk memfokuskan pengembangan, proyek ini memiliki batasan-batasan sebagai berikut:

1. Aplikasi hanya mendukung planning untuk satu hari (hari ini)
2. Tidak ada fitur multi-user atau sinkronisasi cloud
3. Tidak ada sistem reminder/notifikasi aktif
4. Tidak ada fitur edit jadwal (harus delete dan create ulang)
5. Format waktu menggunakan 24 jam
6. Data disimpan lokal dalam file JSON
7. Aplikasi berjalan di desktop (Windows, macOS, Linux)
8. Tidak ada fitur export ke format lain (PDF, image)
9. Tidak ada kategorisasi atau tagging untuk aktivitas
10. Interface menggunakan bahasa Inggris

BAB 2 – KONSEP OOP YANG DITERAPKAN

2.1 Encapsulation (10%)

Encapsulation diterapkan dengan cara menyembunyikan detail data dan hanya menyediakan akses melalui metode getter/setter.

Contoh implementasi dalam kode:

```python
class BaseScheduleItem(ABC):
    def __init__(self, start_h, start_m, end_h, end_m, desc, color):
        self._start_h = start_h  # protected attribute
        self._start_m = start_m
        self._end_h = end_h
        self._end_m = end_m
        self._desc = desc
        self.color = color

    # Getter methods untuk akses yang terkontrol
    def get_start_h(self): return self._start_h
    def get_desc(self): return self._desc
```

Penjelasan: Atribut seperti _start_h dan _desc tidak dapat diakses langsung dari luar kelas, melainkan melalui metode getter seperti get_start_h() dan get_desc().

2.2 Inheritance (10%)

Inheritance digunakan untuk mewarisi sifat dari kelas induk ke kelas anak, sehingga menghindari duplikasi kode.

Struktur pewarisan yang diterapkan:

```python
# Kelas induk abstrak
class BaseScheduleItem(ABC):
    # ... atribut dan metode umum

# Kelas anak yang mewarisi
class Activity(BaseScheduleItem):
    def to_dict(self):  # method tambahan khusus Activity
        return { ... }

    # Override method abstrak dari parent
    def get_display_text(self):
        return f"{self._start_h:02d}:{self._start_m:02d} - ..."
```

Penjelasan: Activity mewarisi semua atribut dan metode dari BaseScheduleItem, lalu menambahkan fungsionalitas khusus seperti to_dict().

2.3 Polymorphism (10%)

Polymorphism diterapkan melalui method overriding dan penggunaan kelas abstrak dengan method yang diimplementasikan berbeda di setiap anak kelas.

Contoh pada visualisasi jarum jam:

```python
class BaseClockHand(ABC):
    @abstractmethod
    def calculate_angle(self, now):
        pass

class HourHand(BaseClockHand):
    def calculate_angle(self, now):
        return (now.hour * 15) + (now.minute * 0.25)  # logika khusus jam

class MinuteHand(BaseClockHand):
    def calculate_angle(self, now):
        return now.minute * 6 + (now.second * 0.1)  # logika khusus menit
```

Penjelasan: Setiap kelas jarum (HourHand, MinuteHand, SecondHand) memiliki implementasi calculate_angle() yang berbeda, namun dipanggil dengan cara yang sama melalui loop:

```
for hand in self.hands:
    hand.draw(now)  # polymorphic call
```

2.4 Abstraction

Abstraction diterapkan melalui penggunaan kelas abstrak dan interface yang menyembunyikan kompleksitas implementasi dan hanya menampilkan fungsionalitas penting kepada pengguna.
Contoh implementasi abstraction:

```
# Abstract Base Class untuk item jadwal
class BaseScheduleItem(ABC):
    @abstractmethod
    def get_display_text(self):
        """Abstraksi: hanya mendefinisikan signature method"""
        pass

    # Method konkret yang bisa langsung digunakan
    def get_start_total_minutes(self):
        return self._start_h * 60 + self._start_m

# Abstract Base Class untuk jarum jam
class BaseClockHand(ABC):
    @abstractmethod
    def calculate_angle(self, now):
        """Abstraksi: setiap jarum punya cara hitung sudut berbeda"""
        pass

    # Method konkret yang sama untuk semua jarum
    def draw(self, now):
        angle_degrees = self.calculate_angle(now)  # polymorphic call
        # ... implementasi menggambar garis
```

Penjelasan:

1. Abstraction Level 1: BaseScheduleItem dan BaseClockHand sebagai kontrak abstrak
    - Hanya mendefinisikan "apa" yang harus dilakukan, bukan "bagaimana" melakukannya
    - Pengguna hanya perlu tahu bahwa ada method get_display_text() dan calculate_angle()
2. Abstraction Level 2: Detail trigonometri dan konversi waktu tersembunyi

```
# Kompleksitas tersembunyi di balik abstraksi
angle_rad = math.radians(angle_degrees)
end_x = self.cx + self.length * math.sin(angle_rad)
end_y = self.cy - self.length * math.cos(angle_rad)
```

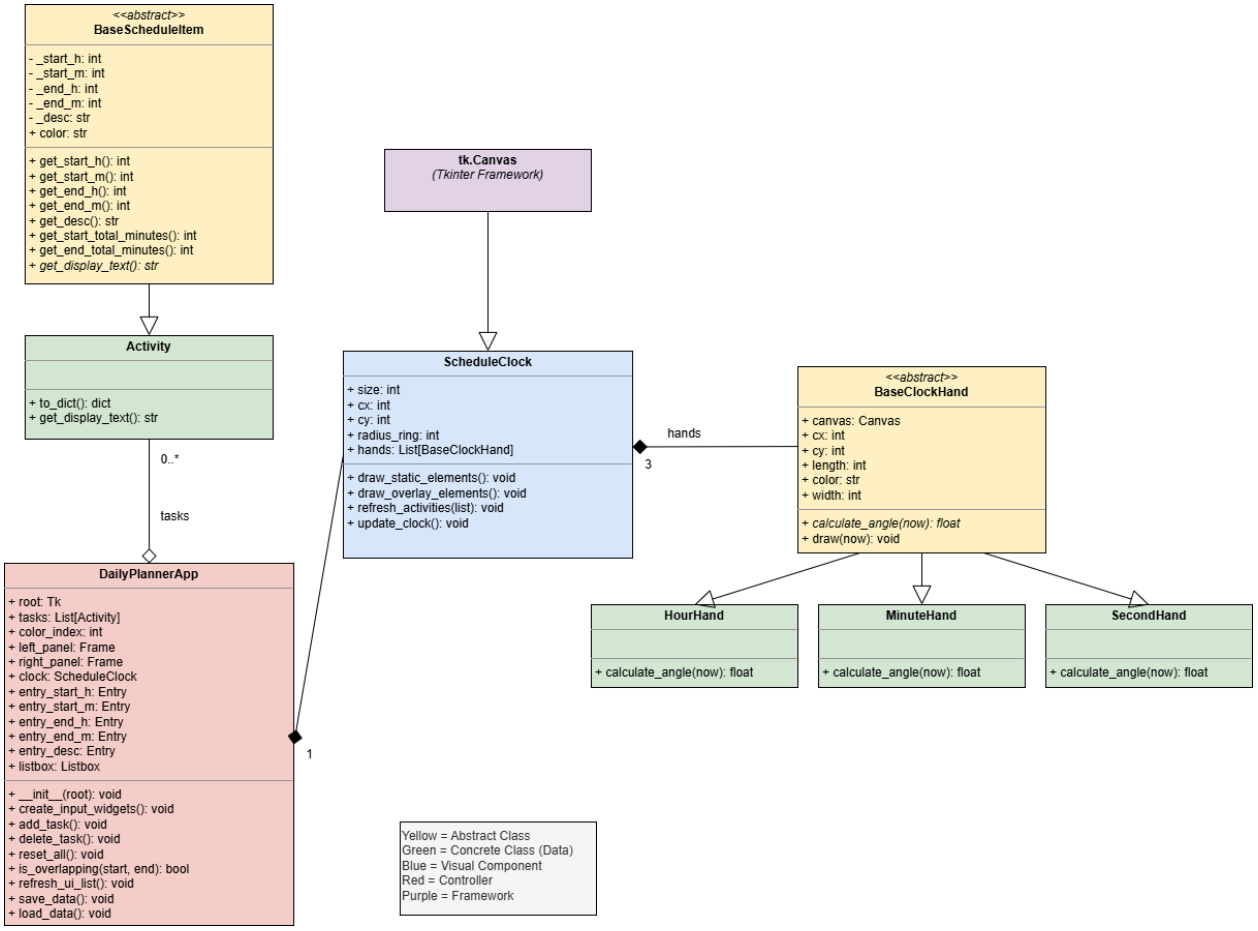
3. Abstraction Level 3: GUI layer menyembunyikan kompleksitas OOP
    - User hanya melihat form input dan visualisasi yang sederhana
    - Tidak perlu memahami inheritance hierarchy atau polymorphism

Manfaat Abstraction dalam aplikasi ini:

1. Reduced Complexity: Developer hanya perlu implement method abstract
2. Loose Coupling: Komponen bisa diganti tanpa mempengaruhi sistem
3. Code Reusability: Logika umum didefinisikan sekali di abstract class
4. Maintainability: Perubahan implementasi tidak mengubah interface

BAB 3 – DESAIN SISTEM: CLASS DIAGRAM

Berikut adalah class diagram yang menggambarkan hubungan antar kelas dalam aplikasi:



3.1 Warna dan Simbol

Warna	Jenis Class	Contoh dalam Aplikasi
Kuning	Abstract Base Class	BaseScheduleItem, BaseClockHand
Hijau	Concrete Data Class	Activity, HourHand, MinuteHand, SecondHand
Biru	Visual Component	ScheduleClock
Merah	Controller/Main App	DailyPlannerApp
Ungu	Framework Class	tk.Canvas (dari Tkinter)

Simbol Relasi:

- Panah Putih Segitiga: Inheritance (pewarisan)
- Diamond Hitam: Composition (kepemilikan kuat)
- Diamond Putih: Aggregation (kepemilikan lemah)

3.2 Hubungan Antar Kelas

a. Inheritance (Pewarisan):
   - Activity ← BaseScheduleItem
   - HourHand, MinuteHand, SecondHand ← BaseClockHand
   - ScheduleClock ← tk.Canvas

b. Composition (Kepemilikan Kuat):
   - DailyPlannerApp → ScheduleClock (1 objek)
   - ScheduleClock → BaseClockHand (3 objek jarum)

c. Aggregation (Kepemilikan Lemah):
   - DailyPlannerApp → Activity (0 atau lebih jadwal)

3.3 Implementasi OOP dalam Diagram

✓ ENCAPSULATION:
- Atribut protected (_start_h, _desc) di BaseScheduleItem
- Akses hanya melalui getter methods (get_start_h(), get_desc())

✓ INHERITANCE:
- Activity mewarisi semua properti dari BaseScheduleItem
- Tiga jenis jarum jam mewarisi dari BaseClockHand

✓ POLYMORPHISM:
- Method calculate_angle() di-override berbeda di setiap jarum
- Method get_display_text() di-override di Activity

✓ ABSTRACTION:
- BaseScheduleItem dan BaseClockHand sebagai abstract base classes
- Hanya mengekspos interface yang diperlukan

BAB 4 – PENJELASAN FITUR APLIKASI

4.1 Fitur Utama

1. Input Jadwal: Pengguna dapat menambahkan aktivitas dengan menentukan jam mulai, jam selesai, dan deskripsi.
2. Visualisasi Pie Chart: Aktivitas ditampilkan sebagai irisan warna-warni pada lingkaran jam 24 jam.
3. Jam Analog Real-Time: Menampilkan waktu sekarang dengan jarum jam, menit, dan detik yang bergerak.
4. Validasi Otomatis: Sistem mencegah jadwal yang bertabrakan (overlapping).
5. Penyimpanan Data: Jadwal disimpan ke file firta_plan.json dan dimuat ulang saat aplikasi dibuka.

4.2 Cara Penggunaan

1. Jalankan file main.py
2. Masukkan:
    - Jam mulai (HH : MM)
    - Jam selesai (HH : MM)
    - Nama aktivitas
3. Klik "Add to Schedule"
4. Aktivitas akan muncul di list dan sebagai irisan di jam visual.
5. Gunakan "Delete Selected" atau "Reset All" untuk mengelola daftar.

BAB 5 – IMPLEMENTASI ALUR APLIKASI

5.1 Inisialisasi dan Startup Program

Proses dimulai ketika pengguna menjalankan file main.py. Aplikasi pertama-tama melakukan inisialisasi dengan membuat instance dari kelas DailyPlannerApp. Pada tahap ini, beberapa komponen penting dibuat:

1. Pembuatan Main Window: Objek Tk() dari Tkinter diinisialisasi sebagai jendela utama aplikasi.
2. Setup Layout: Jendela dibagi menjadi dua panel utama:
    - Panel Kiri: Berisi semua widget input (text entry, buttons, listbox) dengan latar belakang warna THEME["bg_panel"]
    - Panel Kanan: Tempat visualisasi jam analog dengan ukuran 600x600 pixel
3. Load Data Existing: Aplikasi memeriksa keberadaan file firta_plan.json. Jika file ada, data jadwal dimuat dan dikonversi kembali menjadi objek Activity.
4. Setup Visual Clock: Instance ScheduleClock dibuat yang secara otomatis mulai menggambar jam analog real-time.

5.2 Proses Input Jadwal Baru

Ketika pengguna memasukkan data jadwal dan menekan tombol "Add to Schedule", aplikasi menjalankan serangkaian validasi dan proses:

Tahap 1: Pengambilan dan Parsing Input

- Data dari empat field input (start_h, start_m, end_h, end_m) diambil sebagai string
- String dikonversi ke integer dengan nilai default "0" jika field kosong
- Deskripsi aktivitas diambil dari field teks

Tahap 2: Validasi Format dan Logika

```
# Validasi range waktu
if not (0 <= sh <= 23) or not (0 <= eh <= 23): raise ValueError
if not (0 <= sm <= 59) or not (0 <= em <= 59): raise ValueError

# Validasi durasi tidak nol
start_total = sh * 60 + sm
end_total = eh * 60 + em
if start_total == end_total: tampilkan_error()

# Validasi deskripsi tidak kosong
if not desc: tampilkan_error()
```

Tahap 3: Deteksi Tabrakan Jadwal

Aplikasi menggunakan algoritma interval checking yang canggih:

- Semua waktu dikonversi ke total menit sejak 00:00
- Untuk setiap jadwal yang sudah ada, sistem memeriksa apakah interval baru bertabrakan
- Tabrakan terjadi jika: (start_baru < end_lama) DAN (end_baru > start_lama)
- Jika ditemukan tabrakan, pengguna mendapatkan pesan error dan proses dihentikan

Tahap 4: Pembuatan Objek dan Update UI

Jika semua validasi berhasil:

1. Pemilihan Warna Otomatis: Warna dipilih dari palet THEME["pie_colors"] secara berurutan
2. Pembuatan Objek Activity: Instance baru dari kelas Activity dibuat dengan semua parameter
3. Penambahan ke List: Objek ditambahkan ke list self.tasks dan diurutkan berdasarkan waktu mulai
4. Update Visualisasi:
    - Listbox diperbarui dengan memanggil refresh_ui_list()
    - Pie chart di jam analog diperbarui dengan clock.refresh_activities()
5. Penyimpanan Otomatis: Data langsung disimpan ke file JSON

5.3 Mekanisme Visualisasi Real-Time

Komponen ScheduleClock bekerja secara independen dengan mekanisme update setiap detik:

Loop Animasi Kontinu:

```
def update_clock(self):
    now = datetime.now()  # Ambil waktu sistem saat ini

    # Hapus gambar jarum lama
    self.delete("hands")

    # Gambar ketiga jarum dengan posisi baru
    for hand in self.hands:  # hands berisi [HourHand, MinuteHand, SecondHand]
        hand.draw(now)  # Polymorphic call

    # Jadwalkan update lagi dalam 1000ms
    self.after(1000, self.update_clock)
```

Proses Perhitungan Sudut:

Setiap jenis jarum memiliki logika perhitungan sudut yang berbeda:

- HourHand: (jam * 15) + (menit * 0.25) → 1 jam = 15 derajat
- MinuteHand: (menit * 6) + (detik * 0.1) → 1 menit = 6 derajat
- SecondHand: detik * 6 → 1 detik = 6 derajat

Koordinat dan Trigonometri:

Setelah sudut dihitung, sistem menggunakan fungsi trigonometri untuk menentukan titik akhir jarum:

```
angle_rad = math.radians(angle_degrees)
end_x = center_x + length * math.sin(angle_rad)
end_y = center_y - length * math.cos(angle_rad)
```

5.4 Sistem Penyimpanan dan Pemulihan Data

Aplikasi menggunakan file JSON untuk persistensi data dengan mekanisme two-way serialization:

Proses Serialization (Save):

```
def save_data(self):
    # Konversi list objek Activity ke list dictionary
    data = [task.to_dict() for task in self.tasks]

    # Tulis ke file JSON
    with open(DATA_FILE, 'w') as f:
        json.dump(data, f, indent=2)
```

Proses Deserialization (Load):

```
def load_data(self):
    if os.path.exists(DATA_FILE):
        with open(DATA_FILE, 'r') as f:
            data = json.load(f)

        # Rekonstruksi objek Activity dari dictionary
        for item in data:
            new_activity = Activity(
                item['start_h'], item['start_m'],
                item['end_h'], item['end_m'],
                item['desc'], item['color']
            )
            self.tasks.append(new_activity)
```

5.5 Manajemen Jadwal (Delete & Reset)

Operasi Delete:

1. Pengguna memilih item dari Listbox
2. Sistem mengambil index seleksi
3. Objek dihapus dari list self.tasks
4. UI dan visualisasi diperbarui
5. Data otomatis disimpan ke JSON

Operasi Reset All:

1. Pengguna menekan tombol "Reset All"
2. Dialog konfirmasi muncul untuk prevent accidental deletion
3. Jika dikonfirmasi: self.tasks.clear() menghapus semua data
4. UI dan visualisasi diperbarui
5. File JSON ditimpa dengan array kosong

5.6 Arsitektur Three-Tier yang Terimplementasi

Aplikasi mengikuti pola arsitektur three-tier yang jelas:

1. Presentation Layer (GUI):

- DailyPlannerApp: Controller utama yang mengatur alur aplikasi
- ScheduleClock: Komponen visualisasi custom
- Widget Tkinter (Entry, Button, Listbox): Interface pengguna

2. Business Logic Layer:

- Validasi input dan error handling
- Algoritma deteksi tabrakan jadwal
- Logika sorting dan manajemen list
- Konversi waktu dan perhitungan matematis

3. Data Layer:

- BaseScheduleItem & Activity: Model data jadwal
- File firta_plan.json: Storage persistent
- Serialization/Deserialization logic

5.7 Error Handling dan User Feedback

Aplikasi memiliki sistem error handling yang komprehensif:

Jenis-jenis Error yang Ditangani:

1. Format Error: Input bukan angka atau di luar range 0-23/0-59
2. Logical Error: Waktu mulai = waktu selesai
3. Conflict Error: Jadwal bertabrakan dengan yang sudah ada
4. Empty Field: Deskripsi aktivitas kosong
5. File Error: Gagal baca/tulis file JSON

Mekanisme Feedback:

- Messagebox: Untuk error yang membutuhkan user action
- Visual Highlight: Field input dengan nilai invalid
- Automatic Cleanup: Reset form setelah success operation
- Confirmation Dialog: Untuk operasi destruktif (reset all)

BAB 6 – PENGUJIAN DAN HASIL APLIKASI

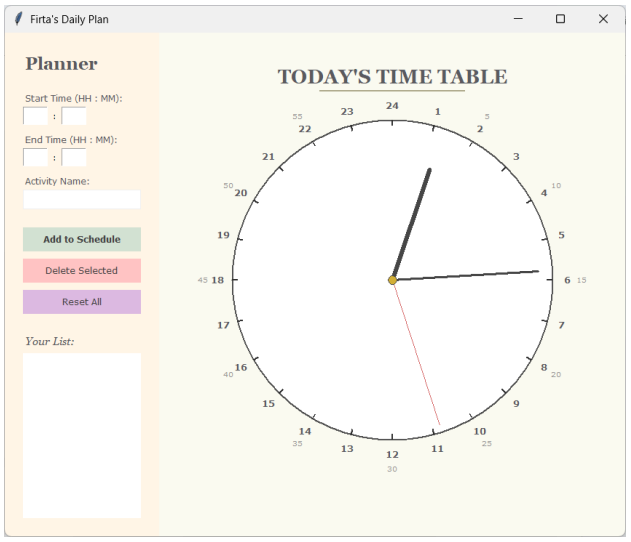
6.1 Metode Pengujian

Aplikasi diuji dengan berbagai skenario untuk memastikan semua fitur berfungsi dengan baik:

1. Pengujian Validasi Input:
   - Input waktu yang tidak valid (jam > 23, menit > 59)
   - Input waktu mulai sama dengan waktu selesai
   - Input deskripsi kosong
2. Pengujian Deteksi Tabrakan:
   - Jadwal yang bertabrakan sepenuhnya
   - Jadwal yang bertabrakan sebagian
   - Jadwal yang berurutan tanpa tabrakan
   - Jadwal yang melewati tengah malam
3. Pengujian Penyimpanan Data:
   - Simpan data dan restart aplikasi
   - Edit data dan cek persistensi
   - Hapus semua data dan restart

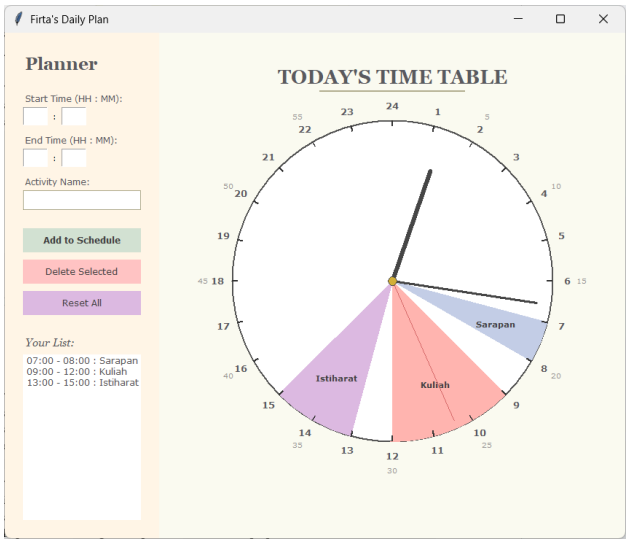
6.2 Hasil Pengujian dan Screenshot

6.2.1 Tampilan Awal Aplikasi



Deskripsi: Tampilan default aplikasi dengan jam analog 24 jam, panel input kosong di kiri, dan list jadwal kosong. Jarum jam bergerak real-time.

6.2.2 Input Jadwal Valid

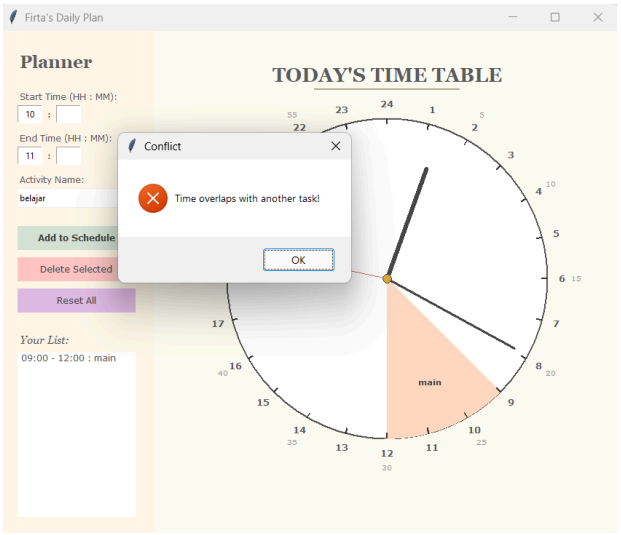


Contoh jadwal yang berhasil ditambahkan:

- 07:00 - 08:00 : Sarapan (warna biru)
- 09:00 - 12:00 : Kuliah (warna merah)
- 13:00 - 15:00 : Istirahat (warna ungu)

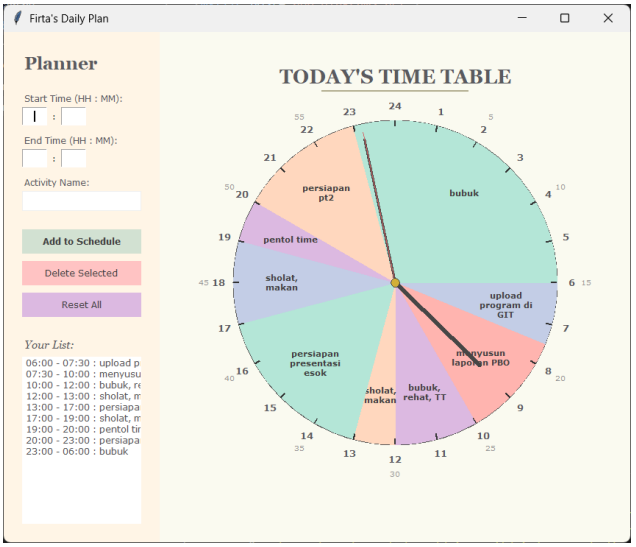
  Setiap jadwal muncul sebagai irisan warna di pie chart dan di listbox.

6.2.3 Deteksi Tabrakan Jadwal



Saat mencoba menambah jadwal 10:00-11:00 ketika sudah ada jadwal 09:00-12:00, sistem menampilkan popup error: "Time overlaps with another task!"

6.2.4 Visualisasi Pie Chart Multi-warna



Pie chart menunjukkan irisan dengan warna berbeda untuk setiap aktivitas. Teks deskripsi muncul di tengah setiap irisan. Irisan terlihat proporsional sesuai durasi waktu.

BAB 7 – KESIMPULAN

Aplikasi Daily Planner telah berhasil mengimplementasikan prinsip-prinsip Object-Oriented Programming (OOP) secara komprehensif dan praktis. Melalui pengembangan aplikasi pengelola jadwal harian dengan visualisasi interaktif ini, tiga pilar utama OOP—encapsulation, inheritance, dan polymorphism—telah diterapkan dengan baik. Encapsulation tercermin dalam penggunaan atribut protected dan metode getter yang mengamankan akses data, sementara inheritance diwujudkan melalui hierarki kelas yang jelas antara BaseScheduleItem dan Activity serta BaseClockHand dengan turunannya. Polimorfisme ditunjukkan melalui kemampuan method overriding yang memungkinkan setiap jenis jarum jam memiliki implementasi perhitungan sudut yang berbeda namun dipanggil dengan cara yang sama.

Aplikasi ini tidak hanya berfungsi sebagai alat pengelola jadwal yang efektif dengan fitur visualisasi pie chart yang intuitif dan jam analog real-time, tetapi juga berperan sebagai media pembelajaran OOP yang visual dan interaktif. Dengan antarmuka pengguna yang user-friendly, validasi input otomatis yang mencegah kesalahan, serta sistem penyimpanan data persisten menggunakan file JSON, aplikasi ini memberikan pengalaman pengguna yang menyeluruh. Struktur kode yang modular dan terorganisir dengan baik memudahkan pengembangan lebih lanjut dan pemeliharaan kode di masa depan.

Keberhasilan implementasi konsep OOP dalam proyek ini menunjukkan bahwa paradigma pemrograman berorientasi objek tidak hanya relevan secara teori tetapi juga sangat aplikatif dalam pengembangan perangkat lunak nyata. Aplikasi Daily Planner siap digunakan untuk kebutuhan manajemen waktu sehari-hari sekaligus berfungsi sebagai contoh konkret implementasi OOP yang dapat menjadi referensi pembelajaran bagi pengembang pemula maupun mahasiswa yang mempelajari konsep pemrograman berorientasi objek.