PEMROGRAMAN BERORIENTASI OBJEK

STUDI KASUS GUI : APLIKASI TIMER SEDERHANA

Disusun Oleh:

Maulia Hafifatun Solihah

5230411231

PROGRAM STUDI INFORMATIKA

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS TEKNOLOGI YOGYAKARTA

2024/2025

DAFTAR ISI

BAB I

PENDAHULUAN

Penggunaan teknologi dalam pengelolaan waktu semakin meningkat untuk mendukung efisiensi dan produktivitas masyarakat modern. Aplikasi timer menjadi salah satu solusi populer untuk mengatur aktivitas dan mengingatkan jadwal. Dalam pengembangannya, Python sering digunakan karena kesederhanaannya serta dukungan library seperti Tkinter, yang merupakan library bawaan Python. Tkinter mempermudah pengembangan aplikasi GUI yang ringan, fungsional, dan tidak memerlukan instalasi tambahan, sehingga cocok untuk aplikasi sederhana dengan antarmuka yang menarik dan intuitif.

Studi kasus pembuatan timer dengan tkinter memberikan peluang untuk mempelajari konsep seperti manipulasi waktu, fungsi asinkron, dan pengelolaan event GUI. Selain menghasilkan aplikasi yang fungsional dan mudah digunakan, proyek ini juga relevan untuk pemula maupun pengembang tingkat menengah yang ingin mendalami pengembangan aplikasi desktop berbasis Python. Aplikasi ini dapat dikembangkan lebih lanjut untuk kebutuhan yang lebih kompleks dan menjadi contoh integrasi antara fitur interaktif dan algoritma logis.

BAB II

PEMBAHASAN

A. Alur Kerja Timer

1. Menambahkan Timer
   Pengguna menekan tombol "+ Tambah Timer", memasukkan durasi dalam detik melalui dialog, dan timer ditampilkan dengan label waktu serta tombol kontrol (Start, Pause, Reset, Clear).

2. Timer ditampilkan:

   Setelah timer ditambahkan, antarmuka memperlihatkan label durasi timer: Menampilkan durasi dalam format menit:detik (misalnya 01:30 untuk 90 detik).

- Tombol kontrol:
  - Start: Untuk memulai timer dan menghitung mundur.
  - Pause: Untuk menjeda timer saat sedang berjalan.
  - Reset: Untuk mengatur ulang timer ke durasi awal.
  - Clear: Untuk menghapus timer dari daftar jika timer sudah tidak dibutuhkan lagi.

  Semua timer yang ditambahkan akan muncul dalam area scrollable sehingga pengguna dapat menambahkan banyak timer sekaligus tanpa membatasi tampilan aplikasi.

3. Pengguna menjalankan timer:

- Ketika tombol "Start" pada timer ditekan, fungsi start_timer() akan dijalankan. Fungsi ini akan memulai hitungan mundur.

- Timer akan dihitung mundur dalam satuan detik, dan label durasi akan terus diperbarui setiap detiknya.
- Threading digunakan agar timer berjalan di *background* tanpa membekukan antarmuka pengguna. Setiap detik, sisa waktu diperbarui dan ditampilkan pada label timer.
- Jika durasi mencapai 0 detik, timer selesai dan aplikasi akan memutar bunyi notifikasi menggunakan winsound.Beep(), dan menampilkan pesan menggunakan messagebox.showinfo() bahwa timer telah selesai.

4. Pemberitahuan saat selesai:

- Setelah timer selesai (ketika waktu mencapai 0), aplikasi akan memutar suara notifikasi menggunakan winsound.Beep() untuk memberi tahu pengguna bahwa timer sudah habis, dan akan menampilkan pesan menggunakan messagebox.showinfo() yang memberitahukan pengguna bahwa waktu sudah habis.
- Setelah pemberitahuan, tombol Start akan kembali aktif untuk memungkinkan pengguna memulai ulang timer, tombol Pause dinonaktifkan, dan tombol Reset diaktifkan untuk mengatur ulang timer.

5. Pengguna dapat:

- Menjeda atau Melanjutkan Timer:

  Jika timer sedang berjalan, pengguna bisa menekan tombol Pause untuk menghentikan sementara timer. Jika timer dijeda, tombol Pause akan berubah menjadi "Continue" untuk melanjutkan hitungan mundur. Dan jika Continue ditekan, timer akan dilanjutkan dari waktu yang dijeda.

- Mengatur Ulang Waktu ke Durasi Awal:

  Pengguna bisa menekan tombol Reset untuk mengembalikan timer ke durasi awal yang telah dimasukkan. Timer akan kembali ke nilai awal dan dimulai dari sana jika tombol Start ditekan lagi.

- Menghapus Timer dari Daftar:

  Jika timer sudah selesai atau tidak diperlukan lagi, pengguna bisa menekan tombol Clear untuk menghapus timer dari antarmuka dan dari daftar timer yang aktif.

6. Reset temua timer:

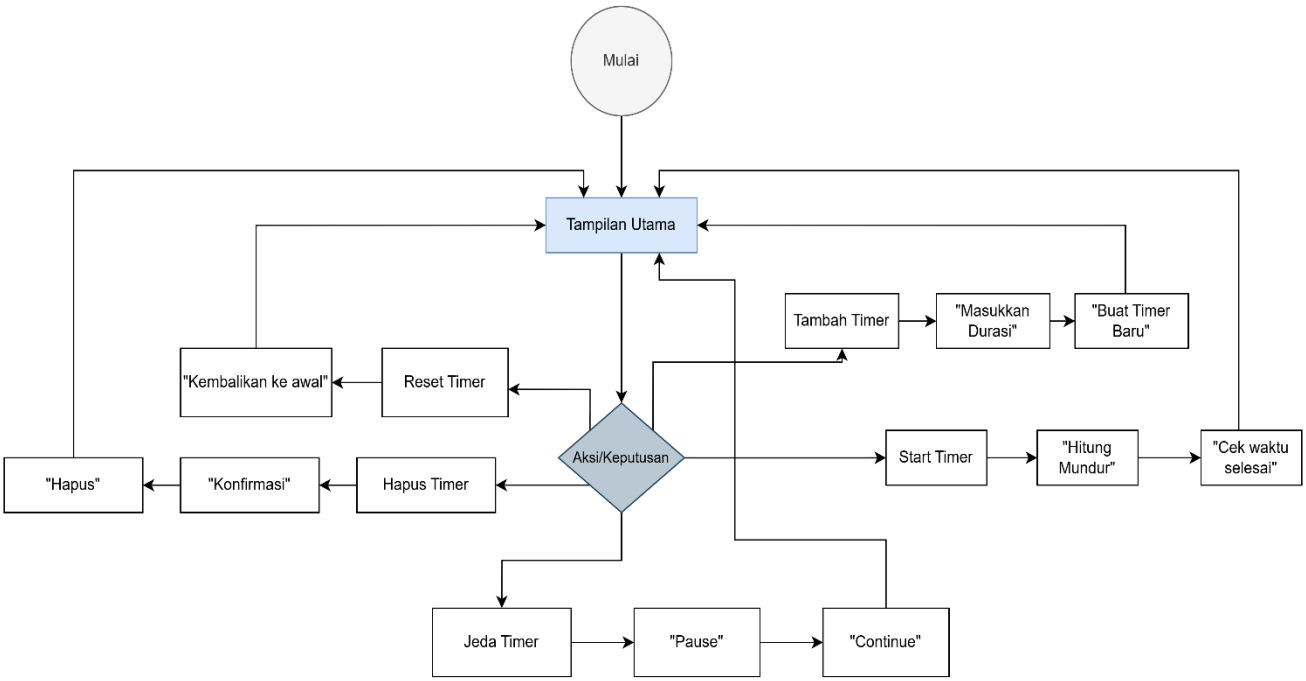
Jika pengguna menekan tombol "Reset Semua", aplikasi akan memanggil fungsi reset_all_timers(). Fungsi ini akan:

- Mengatur ulang semua timer yang ada dalam daftar self.timers ke durasi awal mereka.
- Membuat semua tombol Start aktif kembali, tombol Pause dinonaktifkan, dan tombol Reset diaktifkan untuk setiap timer.

7. Hapus semua timer:

Jika pengguna menekan tombol "Hapus Semua", aplikasi akan menghapus semua timer yang ada.

B. Activity Diagram



C. Code dan Penjelasan

1. Import Library

```
import tkinter as tk
from tkinter import messagebox, simpledialog
import time
import threading
import winsound
```

Penjelasan:

- tkinter: Library GUI yang digunakan untuk membuat antarmuka pengguna.
- messagebox & simpledialog: Modul dari Tkinter untuk menampilkan pesan dan dialog input sederhana.
- time: Digunakan untuk fungsi delay saat timer berjalan.
- threading: Untuk menjalankan timer di *thread* terpisah, sehingga aplikasi tidak membeku selama proses berlangsung.
- winsound: Untuk memberikan notifikasi berupa bunyi saat timer selesai.

2. Kelas Timer

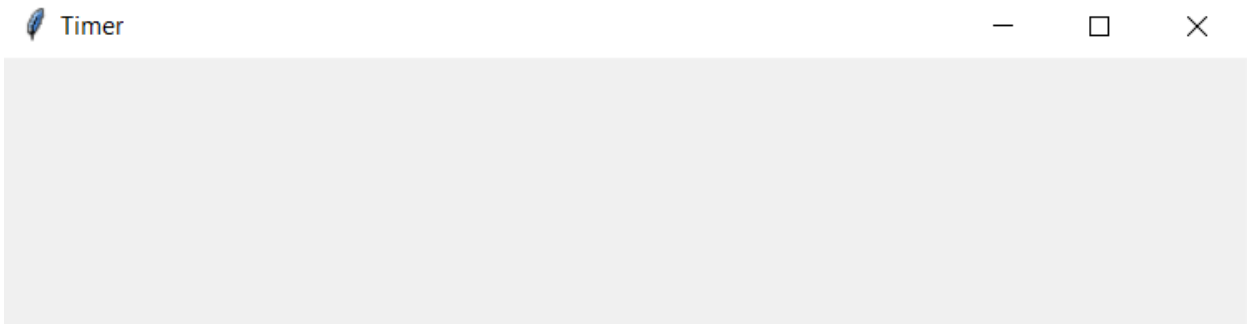
Kelas ini adalah inti dari aplikasi, yang mengatur logika dan antarmuka aplikasi timer.

```
class Timer:
    def __init__(self, root):
        self.root = root
        self.root.title("Timer")
        self.root.geometry("600x400")
        self.root.configure(bg='#f0f0f0')
```

Penjelasan:

__init__ (Constructor): Digunakan untuk tampilan utama aplikasi timer, seperti merancang ruangan dengan tombol – tombol dan tempat untuk timer.

Output:



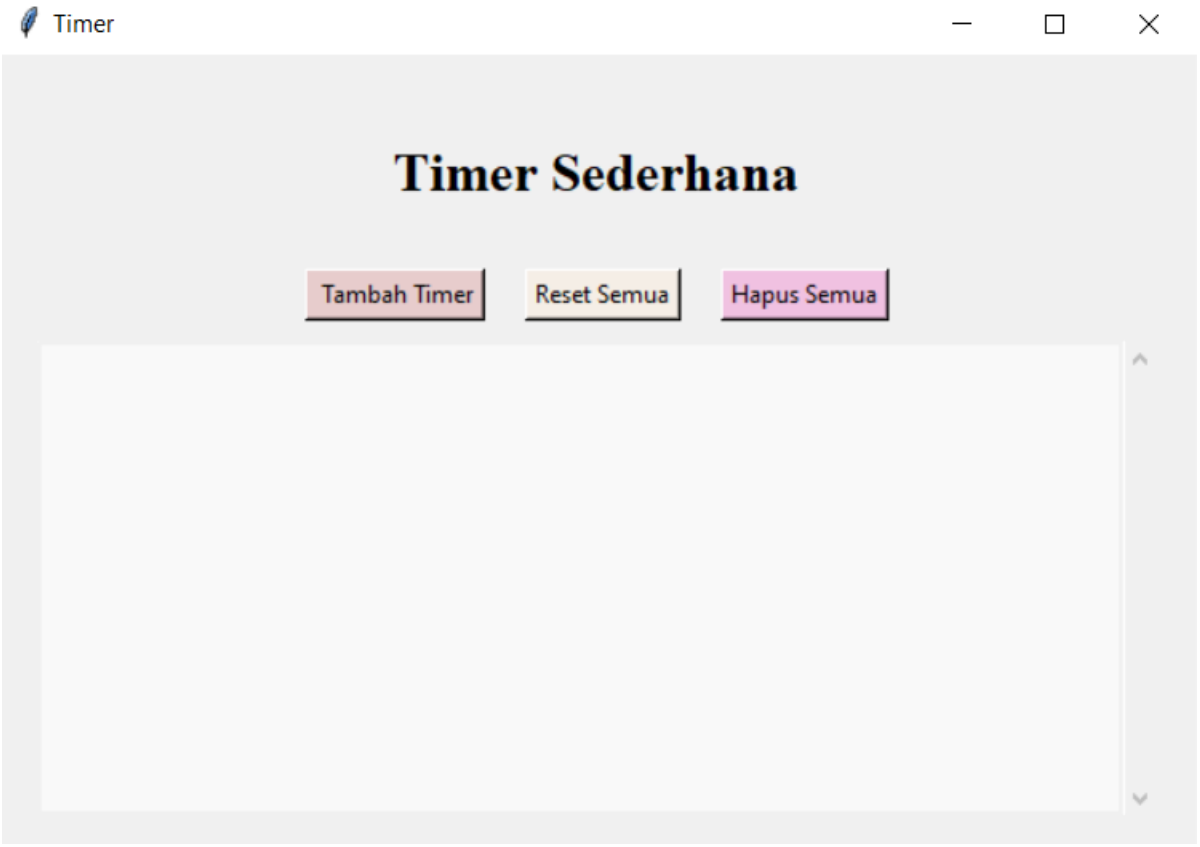
Komponen – komponen penting:

```
# 1. Frame utama
self.main_frame = tk.Frame(root, bg='#f0f0f0')
self.main_frame.pack(padx=20, pady=20, fill=tk.BOTH, expand=True)

# 2. Judul
self.title_label = tk.Label(self.main_frame, text="Timer Sederhana", font=("Times New Roman", 20, 'bold'))
self.title_label.pack(pady=20)

# 3. Frame untuk tombol
self.button_frame = tk.Frame(self.main_frame, bg='#f0f0f0')
self.button_frame.pack(pady=10)

# 4. Tombol tambah timer
self.add_timer_btn = tk.Button(self.button_frame, text=" Tambah Timer", command=self.add_new_timer, bg='#E7CCCC', fg='black')
self.add_timer_btn.pack(side=tk.LEFT, padx=10)

# 5. Tombol reset semua timer
self.reset_all_btn = tk.Button(self.button_frame, text="Reset Semua", command=self.reset_all_timers, bg='#F5EEE6', fg='black')
self.reset_all_btn.pack(side=tk.LEFT, padx=10)

# 6. Tombol hapus semua timer
self.clear_btn = tk.Button(self.button_frame, text="Hapus Semua", command=self.clear_semua, bg='#F0C1E1', fg='black')
self.clear_btn.pack(side=tk.LEFT, padx=10)

# 7. Scrollable frame untuk timer
self.canvas = tk.Canvas(self.main_frame, bg='#F9F9F9')
self.scrollbar = tk.Scrollbar(self.main_frame, orient="vertical", command=self.canvas.yview)
self.scrollable_frame = tk.Frame(self.canvas, bg='#F9F9F9')
self.scrollable_frame.bind("<Configure>", lambda e: self.canvas.configure(scrollregion=self.canvas.bbox("all")))
self.canvas.create_window((0, 0), window=self.scrollable_frame, anchor="nw")
self.canvas.configure(yscrollcommand=self.scrollbar.set)
self.canvas.pack(side="left", fill="both", expand=True)
self.scrollbar.pack(side="right", fill="y")

# 8. List untuk menyimpan timer aktif
self.timers = []
```

Penjelasan:

1) Frame Utama (self.main_frame): Seperti kotak besar yang menampung semua elemen di aplikasi. Warna yang digunakan untuk latar belakang adalah abu – abu muda, jarak dipinggir (padding 20 pixel), serta dapat mengembang penuh di jendela.
2) Label Judul (self.title_label): Membuat tulisan "Timer Sederhana" di bagian atas dengan font Times New Roman, ukuran 20 dan tebal.
3) Frame Tombol (self.button_frame): Kotak kecil untuk menampung tombol-tombol
4) Tombol Tambah Timer (self.add_timer_btn) dengan warna merah muda, disusun mendatar (dari kiri)
5) Tombol Reset Semua (self.reset_all_btn) dengan warna krem, disusun mendatar (dari kiri)
6) Tombol Hapus Semua (self.clear_btn) degan warna ungu muda, disusun mendatar (dari kiri)
7) Scrollable Frame (self.canvas dan self.scrollbar): Ini merupakan area untuk menampung timer-timer yang telah ditambahkan. Kalau timer banyak, bisa di-scroll. Canvas merupakan "kanvas" tempat frame bisa digulir, sedangkan Scrollbar adalah garis penggulung di sisi kanan
8) List untuk menyimpan timer aktif (self.timers = [])

Output:



Fungsi – fungsi:

1) Fungsi untuk menambah timer

```
def add_new_timer(self):
    # Memilih durasi timer
    duration = simpledialog.askinteger("Tambah Timer", "Masukkan durasi timer (dalam detik):", minvalue=1, maxvalue=3600)

    if duration:
        timer_frame = tk.Frame(self.scrollable_frame, bg='#ffffff', bd=1, relief=tk.RAISED)
        timer_frame.pack(fill=tk.X, pady=5, padx=5)

        # Membuat label durasi timer
        duration_label = tk.Label(timer_frame, text=self.format_time(duration), font=("Times", 15), bg='#ffffff')
        duration_label.pack(side=tk.LEFT, pady=10)

        # Tombol kontrol
        control_frame = tk.Frame(timer_frame, bg='#ffffff')
        control_frame.pack(side=tk.RIGHT, pady=10)

        start_btn = tk.Button(control_frame, text="Start", command=lambda: self.start_timer(duration_label, duration, start_btn, pause_btn, reset_btn), bg='#E6A4B4', fg='black')
        start_btn.pack(side=tk.LEFT, padx=2)

        pause_btn = tk.Button(control_frame, text="Pause", command=lambda: self.pause_timer(duration_label, start_btn, pause_btn), bg='#FFCCD2',fg='black',state=tk.DISABLED)
        pause_btn.pack(side=tk.LEFT, padx=2 )

        reset_btn = tk.Button(control_frame, text="Reset", command=lambda: self.reset_timer(duration_label, duration, start_btn, pause_btn, reset_btn),bg='#EBD8C3',fg='black')
        reset_btn.pack(side=tk.LEFT, padx=2)

        delete_btn = tk.Button(control_frame, text="Clear", command=lambda: self.delete_timer(timer_frame, duration_label), bg='#C1A3A3', fg='black')
        delete_btn.pack(side=tk.LEFT, padx=2)

        # Simpan timer
        timer_info = {
            'frame': timer_frame,
            'label': duration_label,
            'start_btn': start_btn,
            'pause_btn': pause_btn,
            'reset_btn': reset_btn,
            'duration': duration,
            'remaining': duration,
            'running': False,
            'paused': False
        }
        self.timers.append(timer_info)
```

Penjelasan:

```
duration = simpledialog.askinteger("Tambah Timer", "Masukkan durasi timer (dalam detik):", minvalue=1, maxvalue=3600)
```

Fungsi ini menggunakan simpledialog.askinteger untuk meminta pengguna memasukkan durasi timer dalam detik. Pengguna hanya dapat memasukkan angka antara 1 dan 3600 detik (1 jam). Jika pengguna membatalkan input atau tidak memasukkan durasi, nilai duration akan menjadi None dan timer tidak akan ditambahkan.

```
timer_frame = tk.Frame(self.scrollable_frame, bg='#ffffff', bd=1, relief=tk.RAISED)
timer_frame.pack(fill=tk.X, pady=5, padx=5)
```

Setelah durasi timer berhasil dimasukkan, sebuah frame (timer_frame) dibuat untuk menampung elemen-elemen GUI terkait timer tersebut. Frame ini memiliki border dan relief untuk memberikan efek visual yang lebih baik. Kemudian frame ini akan diposisikan di dalam scrollable_frame, yang memungkinkan tampilan vertikal yang bisa digulir.

```
duration_label = tk.Label(timer_frame, text=self.format_time(duration), font=("Times", 15), bg='#ffffff')
duration_label.pack(side=tk.LEFT, pady=10)
```

Sebuah label dibuat untuk menampilkan durasi timer yang dimasukkan oleh pengguna. Durasi yang ditampilkan diformat menggunakan fungsi format_time untuk mengubah detik menjadi format waktu "menit:detik" (misalnya, 90 detik menjadi 01:30).

```
control_frame = tk.Frame(timer_frame, bg='#ffffff')
control_frame.pack(side=tk.RIGHT, pady=10)
```

control_frame dibuat untuk menampung tombol-tombol kontrol yang akan digunakan untuk memulai, menjeda, mereset, dan menghapus timer.

- Tombol Start: Tombol ini digunakan untuk memulai timer. Ketika tombol ditekan, fungsi start_timer akan dipanggil untuk memulai hitungan mundur.
- Tombol Pause: Tombol ini digunakan untuk menjeda timer yang sedang berjalan. Pada awalnya tombol ini dalam keadaan dinonaktifkan (state=tk.DISABLED), dan baru akan aktif setelah tombol "Start" ditekan.
- Tombol Reset: Tombol ini digunakan untuk mereset timer ke durasi awal dan memulai kembali.
- Tombol Clear: Tombol ini digunakan untuk menghapus timer dari tampilan dan menghapusnya dari daftar timer yang aktif.

```
timer_info = {
                'frame': timer_frame,
                'label': duration_label,
                'start_btn': start_btn,
                'pause_btn': pause_btn,
                'reset_btn': reset_btn,
                'duration': duration,
                'remaining': duration,
                'running': False,
                'paused': False
            }
            self.timers.append(timer_info)
```

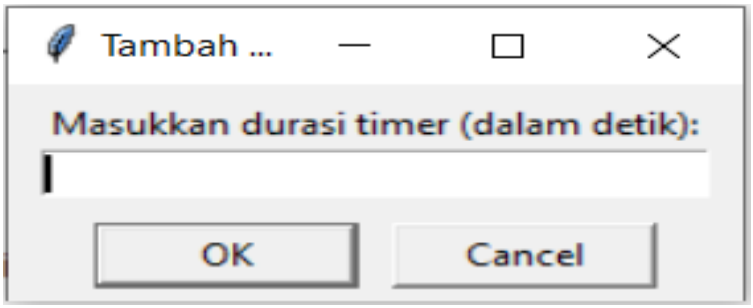
Semua informasi tentang timer yang baru dibuat disimpan dalam dictionary timer_info, yang berisi:

- Referensi ke frame, label, dan tombol kontrol timer.
- Durasi awal (duration) dan sisa waktu yang tersisa (remaining).
- Status apakah timer sedang berjalan (running) atau dijeda (paused).

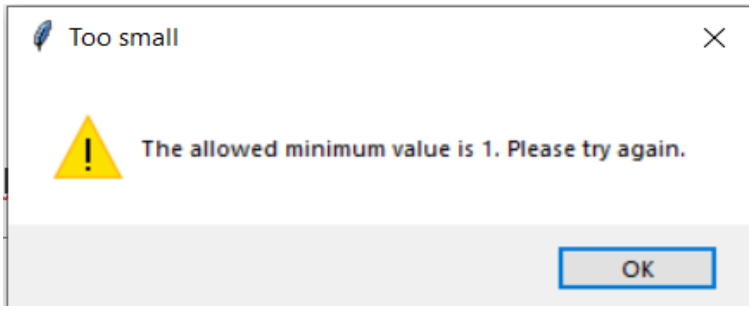
Kemudian, timer_info ditambahkan ke dalam list self.timers yang menyimpan semua timer yang aktif dalam aplikasi.

Output:

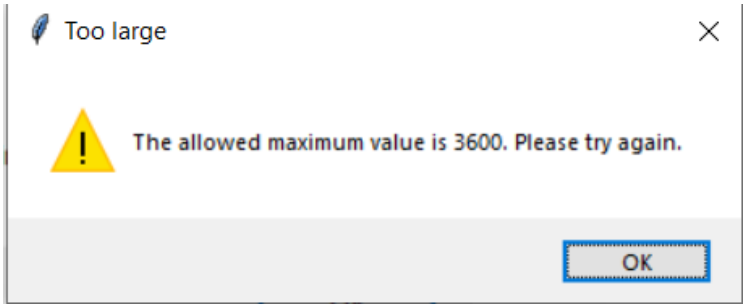
Dialog untuk memasukkan durasi timer (dalam detik).



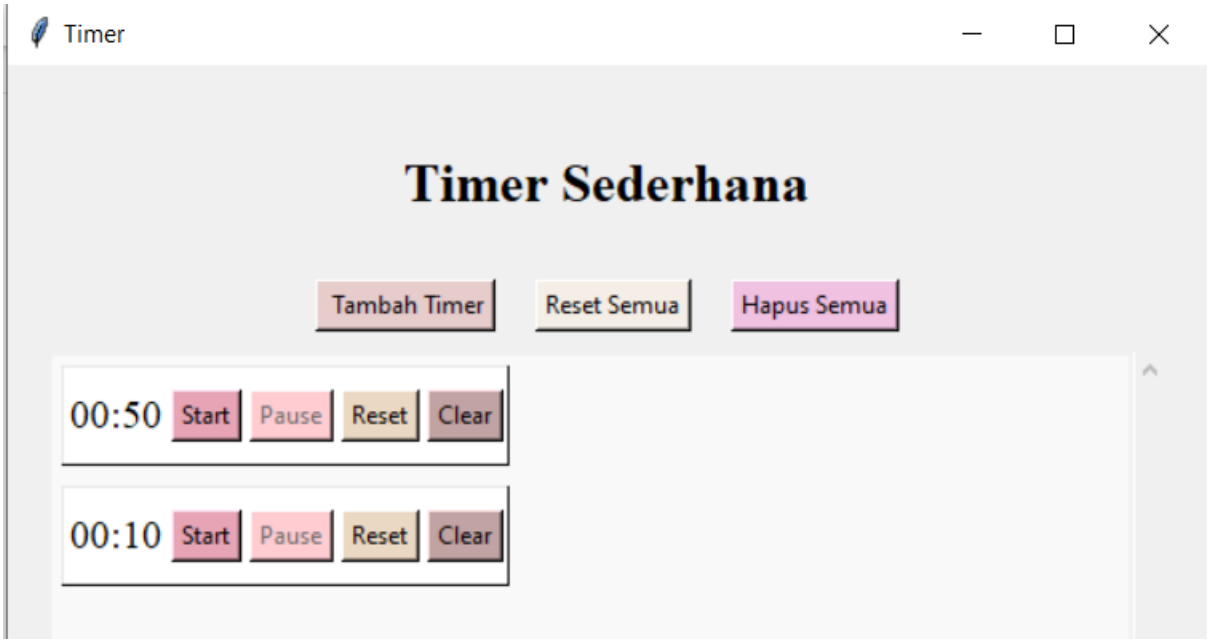
Jika kurang dari minimal detik yang ditentukan (kurang dari 1).



Jika lebih dari maksimal detik yang ditentukan (lebih dari 3600).



Timer yang disimpan.



2) Fungsi untuk mengubah detik menjadi format waktu "menit:detik"

```
def format_time(self, seconds):
        # Konversi detik ke format menit:detik
        minutes, secs = divmod(seconds, 60)
        return f"{minutes:02d}:{secs:02d}"
```

Penjelasan:

```
minutes, secs = divmod(seconds, 60)
```

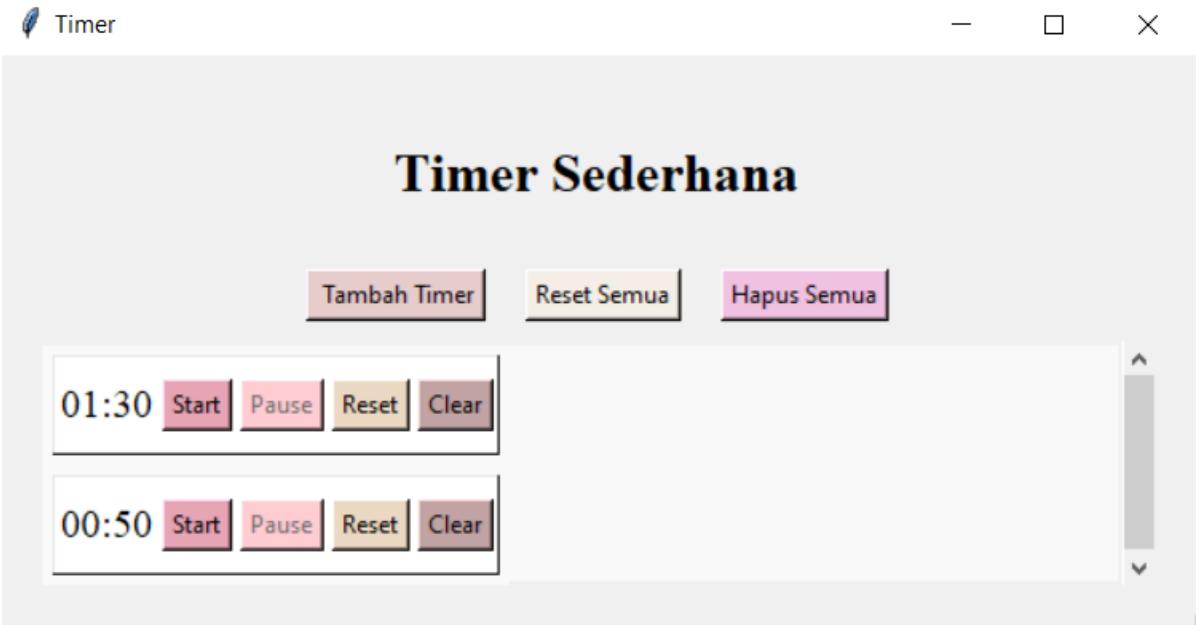
- Fungsi divmod(a, b) membagi a dengan b dan mengembalikan hasilnya dalam bentuk tuple yang berisi hasil pembagian (quotient) dan sisa hasil bagi (remainder).
- Dalam hal ini, kita membagi jumlah detik (seconds) dengan 60 (karena 1 menit = 60 detik). Minutes berarti hasil pembagian (jumlah menit), dan secs adalah sisa hasil bagi (jumlah detik yang tersisa setelah dikurangi menit). Contoh: Jika seconds = 150, maka divmod(150, 60) menghasilkan (2, 30), yang berarti 2 menit dan 30 detik.

```
return f"{minutes:02d}:{secs:02d}"
```

- Fungsi ini menggunakan f-string untuk menghasilkan output dalam format yang diinginkan.
- {minutes:02d} dan {secs:02d} adalah cara untuk memformat variabel minutes dan secs agar selalu ditampilkan dalam dua digit (menambahkan angka 0 di depan jika kurang dari 10). Misalnya, 5 menit akan ditampilkan sebagai "05", dan 3 detik akan ditampilkan sebagai "03".
- Dengan demikian, f-string ini memastikan bahwa hasil akhir selalu dalam format MM:SS (menit:detik).

Output:

Diberikan 90 detik dan 50 detik:



3) Fungsi untuk memulai timer

```
def start_timer(self, label, duration, start_btn, pause_btn, reset_btn):
    timer_info = next(t for t in self.timers if t['label'] == label)

    if not timer_info['running']:
        timer_info['running'] = True
        timer_info['paused'] = False
        start_btn.config(state=tk.DISABLED)
        pause_btn.config(state=tk.NORMAL)
        reset_btn.config(state=tk.DISABLED)

        def countdown():
            while timer_info['remaining'] > 0 and timer_info['running']:
                if not timer_info['paused']:
                    timer_info['remaining'] -= 1
                    label.config(text=self.format_time(timer_info['remaining']))

                    if timer_info['remaining'] == 0:
                        # Notifikasi saat timer selesai
                        winsound.Beep(1000, 500)
                        messagebox.showinfo("Timer Selesai", "Waktu anda habis!")

                        # Reset tombol
                        start_btn.config(state=tk.NORMAL)
                        pause_btn.config(state=tk.DISABLED)
                        reset_btn.config(state=tk.NORMAL)
                        timer_info['running'] = False
                        break

                time.sleep(1)

        threading.Thread(target=countdown, daemon=True).start()
```

Penjelasan:

```
timer_info = next(t for t in self.timers if t['label'] == label)
```

Fungsi ini mencari objek timer yang sesuai dengan label yang diberikan. self.timers merupakan daftar yang menyimpan semua timer yang sedang aktif, dan setiap timer diwakili oleh dictionary timer_info yang berisi informasi tentang timer (misalnya, remaining waktu, running status, dll.).

```
if not timer_info['running']:
            timer_info['running'] = True
            timer_info['paused'] = False
            start_btn.config(state=tk.DISABLED)
            pause_btn.config(state=tk.NORMAL)
            reset_btn.config(state=tk.DISABLED)
```

Jika timer belum berjalan (yaitu running adalah False), maka status timer diubah menjadi running = True, dan status paused diatur menjadi False (karena timer dimulai, bukan dijeda).

Tombol-tombol kontrol diperbarui:

- Tombol Start menjadi dinonaktifkan (state=tk.DISABLED), karena timer sudah mulai.
- Tombol Pause diaktifkan (state=tk.NORMAL), sehingga pengguna dapat menjeda timer yang sedang berjalan.

- Tombol Reset dinonaktifkan (state=tk.DISABLED) sampai timer dijeda atau selesai.

```
def countdown():
    while timer_info['remaining'] > 0 and timer_info['running']:
        if not timer_info['paused']:
            timer_info['remaining'] -= 1
            label.config(text=self.format_time(timer_info['remaining']))

            if timer_info['remaining'] == 0:
                # Notifikasi saat timer selesai
                winsound.Beep(1000, 500)
                messagebox.showinfo("Timer Selesai", "Waktu anda habis!")

                # Reset tombol
                start_btn.config(state=tk.NORMAL)
                pause_btn.config(state=tk.DISABLED)
                reset_btn.config(state=tk.NORMAL)
                timer_info['running'] = False
                break

        time.sleep(1)
```

Mekanisme Hitung Mundur:

- Fungsi countdown berjalan dalam *thread* terpisah (agar tidak mengganggu antarmuka pengguna), yang memastikan aplikasi tetap responsif meskipun timer sedang berjalan.
- Dalam loop while, selama remaining lebih besar dari 0 dan running adalah True, timer akan terus menghitung mundur.
- Cek Status Pause: Timer hanya akan dikurangi jika status paused adalah False (yaitu timer tidak dijeda).
- Setiap detik, waktu yang tersisa (remaining) dikurangi 1, dan label waktu pada GUI diperbarui dengan format waktu yang baru menggunakan self.format_time.

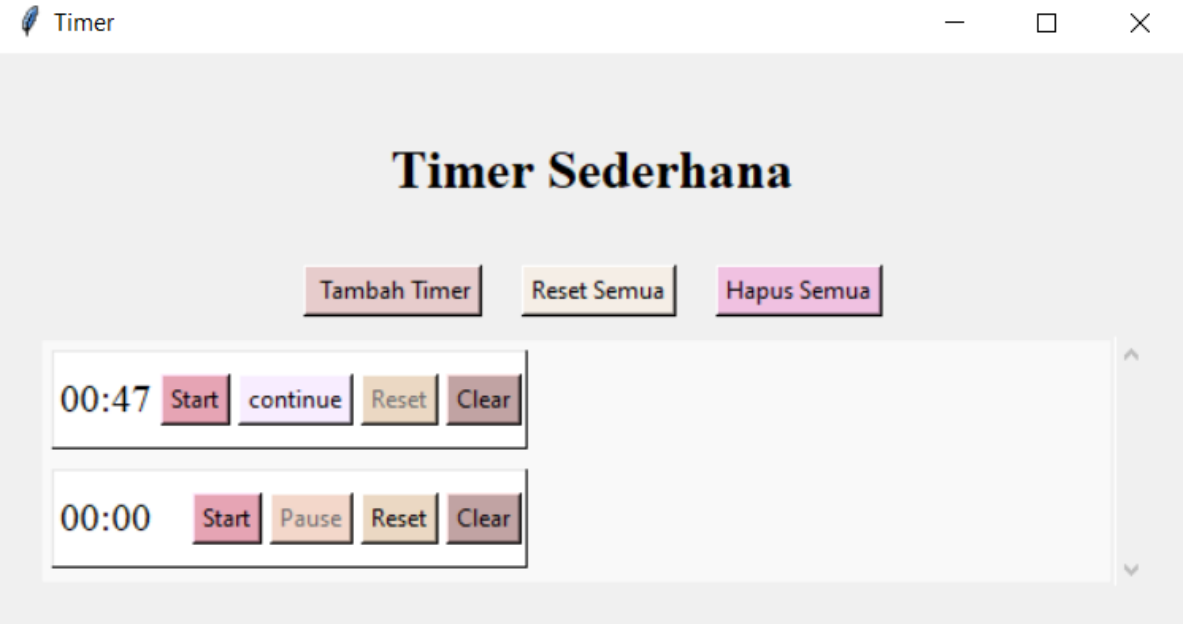
Notifikasi Saat Timer Selesai:

- Saat timer mencapai 0, aplikasi memberi notifikasi dengan menggunakan winsound.Beep untuk membuat bunyi beep, dan menampilkan pesan menggunakan messagebox.showinfo untuk memberi tahu pengguna bahwa waktu telah habis.
- Tombol-tombol kontrol kemudian di-reset:
  - Tombol Start kembali diaktifkan, karena timer sudah selesai dan dapat dimulai ulang.
  - Tombol Pause dinonaktifkan, karena tidak ada lagi yang perlu dijeda.
  - Tombol Reset kembali diaktifkan, memungkinkan pengguna untuk mereset timer jika perlu.

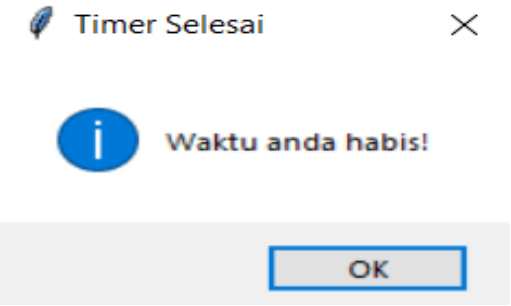
```
threading.Thread(target=countdown, daemon=True).start()
```

- Fungsi countdown dijalankan dalam *thread* terpisah, yang memungkinkan aplikasi tetap responsif (tidak "membeku") saat timer berjalan.
- Penggunaan daemon=True berarti thread ini akan berhenti secara otomatis ketika aplikasi ditutup, tanpa mempengaruhi eksekusi aplikasi lainnya.

Output:



Notifikasi ketika waktu habis:



4) Fungsi untuk menjeda timer

```
def pause_timer(self, label, start_btn, pause_btn):
        timer_info = next(t for t in self.timers if t['label'] == label)

        if timer_info['running']:
            if not timer_info['paused']:
                timer_info['paused'] = True
                pause_btn.config(text="continue", bg='#F8EDFF')
                start_btn.config(state=tk.NORMAL)
            else:
                timer_info['paused'] = False
                pause_btn.config(text="Pause", bg='#F3D7CA')
                start_btn.config(state=tk.DISABLED)
```

Penjelasan:

```
timer_info = next(t for t in self.timers if t['label'] == label)
```

- Fungsi ini mencari objek timer yang sesuai dengan label yang diberikan. self.timers adalah daftar yang menyimpan semua timer yang aktif dalam aplikasi.
- Setiap timer disimpan dalam dictionary yang berisi informasi tentang timer, termasuk durasi, waktu yang tersisa (remaining), serta status apakah timer sedang berjalan (running) atau dijeda (paused).

```
if timer_info['running']:
```

- Sebelum menjeda atau melanjutkan timer, fungsi ini memeriksa apakah timer tersebut sedang berjalan (running == True).
- Jika timer tidak berjalan (misalnya jika timer sudah selesai atau belum dimulai), maka fungsi tidak akan melakukan apa-apa dan langsung selesai.

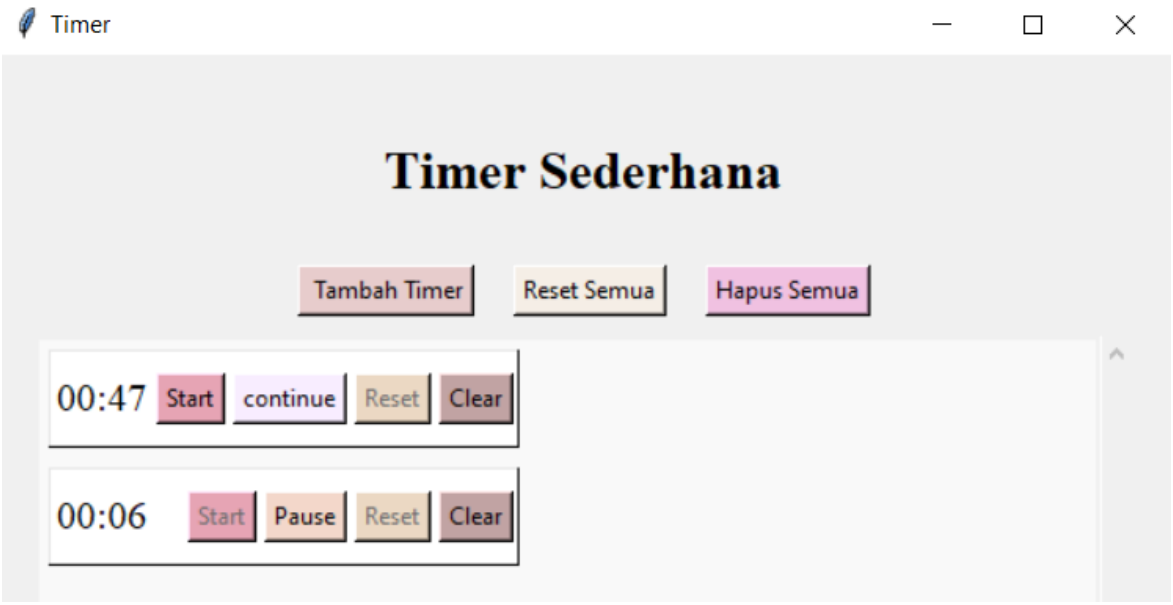
```
if not timer_info['paused']:
    timer_info['paused'] = True
    pause_btn.config(text="continue", bg='#F8EDFF')
    start_btn.config(state=tk.NORMAL)
```

- Jika timer sedang berjalan dan belum dijeda (paused == False), maka status paused diubah menjadi True untuk menunjukkan bahwa timer sedang dijeda.
- Tombol Pause diubah menjadi Continue untuk memberi tahu pengguna bahwa mereka dapat melanjutkan timer. Warna tombol juga diubah menjadi lebih terang dengan bg='#F8EDFF untuk memberikan petunjuk visual.
- Tombol Start diaktifkan kembali (state=tk.NORMAL) agar pengguna dapat melanjutkan timer jika diinginkan.

```
else:
    timer_info['paused'] = False
    pause_btn.config(text="Pause", bg='#F3D7CA')
    start_btn.config(state=tk.DISABLED)
```

- Jika timer sudah dijeda (paused == True), maka timer akan dilanjutkan dengan mengubah status paused menjadi False.
- Tombol Pause dikembalikan ke teks semula, yaitu "Pause", dan warna tombol diubah ke latar belakang default dengan bg='#F3D7CA'.
- Tombol Start dinonaktifkan lagi (state=tk.DISABLED) untuk mencegah pengguna menekan tombol Start saat timer sudah berjalan (karena timer sudah dimulai dan hanya bisa dijeda atau dilanjutkan).

Output:



5) Fungsi untuk mereset

```
def reset_timer(self, label, duration, start_btn, pause_btn, reset_btn):
        timer_info = next(t for t in self.timers if t['label'] == label)

        timer_info['running'] = False
        timer_info['paused'] = False
        timer_info['remaining'] = duration

        label.config(text=self.format_time(duration))
        start_btn.config(state=tk.NORMAL)
        pause_btn.config(state=tk.DISABLED)
        reset_btn.config(state=tk.NORMAL)
```

Penjelasan:

```
 timer_info = next(t for t in self.timers if t['label'] == label)
```

Fungsi ini mencari objek timer yang sesuai dengan label yang diberikan. Dalam daftar self.timers, setiap timer disimpan dalam bentuk dictionary yang berisi informasi tentang timer tersebut, termasuk waktu yang tersisa (remaining), status timer (running, paused), dan durasi awalnya.

```
timer_info['running'] = False
timer_info['paused'] = False
timer_info['remaining'] = duration
```

- running = False: Menandakan bahwa timer tidak sedang berjalan (dihentikan), sehingga tidak ada hitungan mundur yang berlangsung.
- paused = False: Status timer diubah menjadi tidak dijeda (karena timer akan dimulai dari awal lagi).

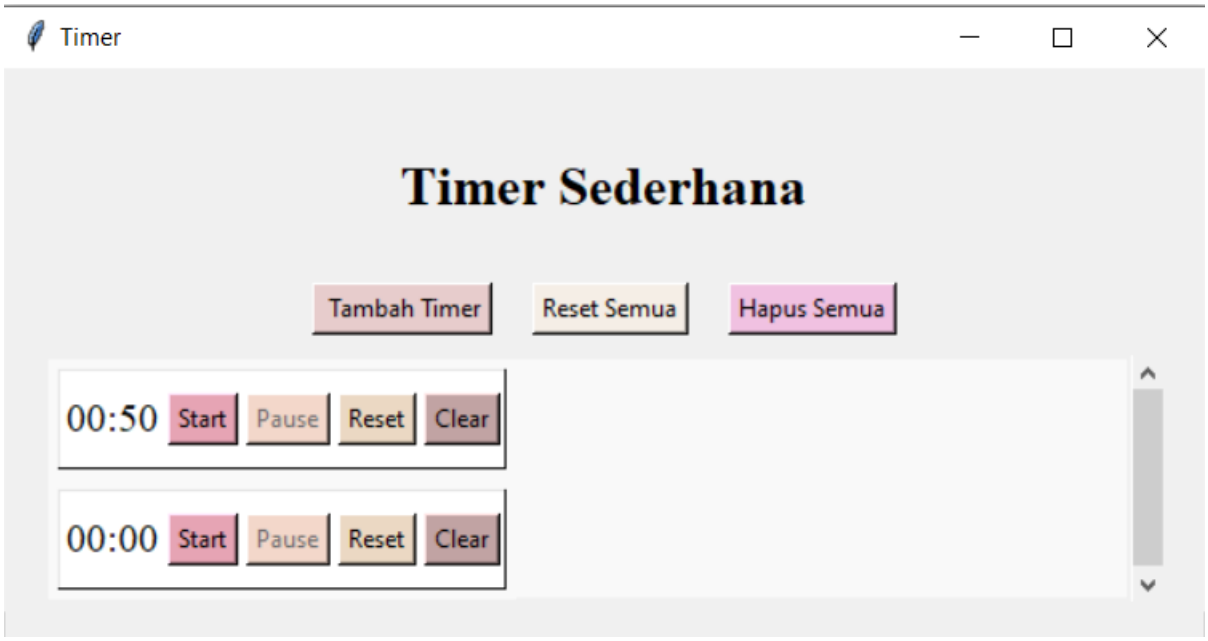
- remaining = duration: Mengembalikan waktu yang tersisa ke durasi awal yang ditetapkan untuk timer. Jadi, timer akan dimulai kembali dengan waktu yang sama seperti saat pertama kali dibuat.

```
label.config(text=self.format_time(duration))
```

- Tampilan label yang menunjukkan waktu pada GUI diperbarui dengan waktu yang tersisa, yang dikembalikan ke nilai duration (waktu awal yang ditetapkan).
- Fungsi self.format_time(duration) digunakan untuk mengubah durasi dalam detik menjadi format yang lebih mudah dibaca, seperti "mm:ss" (menit:detik).

Output:

Setelah salah satu timer direset:



6) Fungsi untuk mereset semua timer

```
def reset_all_timers(self):
        for timer in self.timers:
            self.reset_timer(
                timer['label'],
                timer['duration'],
                timer['start_btn'],
                timer['pause_btn'],
                timer['reset_btn']
            )
```

Penjelasan:

```
for timer in self.timers:
```

Fungsi ini melakukan perulangan (loop) terhadap setiap objek timer yang ada dalam daftar self.timers. Setiap elemen dalam daftar self.timers adalah sebuah dictionary yang berisi

informasi tentang timer, seperti label (untuk tampilan waktu), duration (durasi awal timer), serta tombol kontrol (seperti start_btn, pause_btn, dan reset_btn).

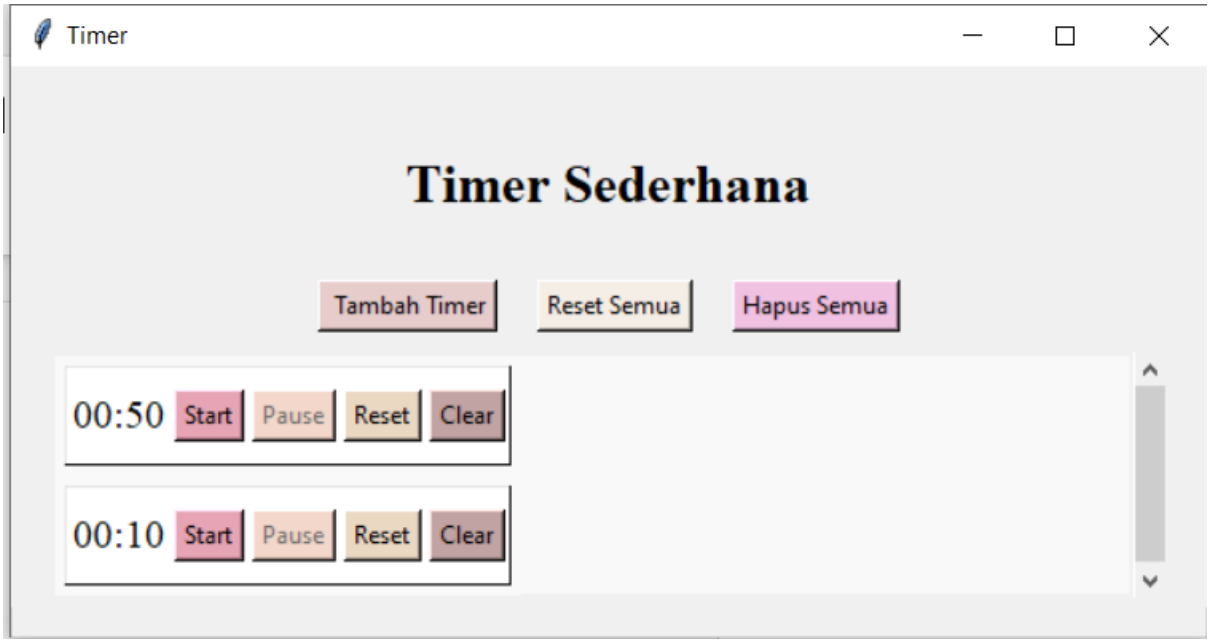
```
self.reset_timer(
                timer['label'],
                timer['duration'],
                timer['start_btn'],
                timer['pause_btn'],
                timer['reset_btn']
            )
```

Untuk setiap timer, fungsi reset_all_timers memanggil fungsi reset_timer dengan parameter yang sesuai. Parameter yang dipassing ke reset_timer adalah:

- timer['label']: Label yang digunakan untuk menampilkan waktu yang tersisa pada timer.
- timer['duration']: Durasi awal timer, yang digunakan untuk mereset waktu yang tersisa ke nilai semula.
- timer['start_btn']: Tombol untuk memulai timer.
- timer['pause_btn']: Tombol untuk menjeda atau melanjutkan timer.
- timer['reset_btn']: Tombol untuk mereset timer.

Output:

Setelah mereset semua:



7) Fungsi untuk menghapus timer

```
 def delete_timer(self, frame, label):
        timer_info = next((t for t in self.timers if t['label'] == label),
None)

        if timer_info:
            timer_info['running'] = False
            frame.destroy()
            self.timers.remove(timer_info)
```

Penjelasan:

```
timer_info = next((t for t in self.timers if t['label'] == label), None)
```

- Fungsi ini mencari timer dalam daftar self.timers yang memiliki label yang cocok dengan label yang diberikan.
- self.timers adalah daftar yang menyimpan informasi tentang semua timer yang aktif, dan setiap timer disimpan sebagai dictionary.
- Fungsi next() digunakan untuk mengambil elemen pertama yang cocok dengan kondisi t['label'] == label.
- Jika tidak ada timer yang memiliki label yang sesuai, maka None akan dikembalikan

```
 if timer_info:
            timer_info['running'] = False
```

Jika timer dengan label yang sesuai ditemukan (yaitu timer_info bukan None), maka status running pada timer_info diubah menjadi False. Ini menghentikan timer tersebut, karena mengubah status running menjadi False memastikan bahwa timer tersebut tidak lagi berjalan.

```
frame.destroy()
```

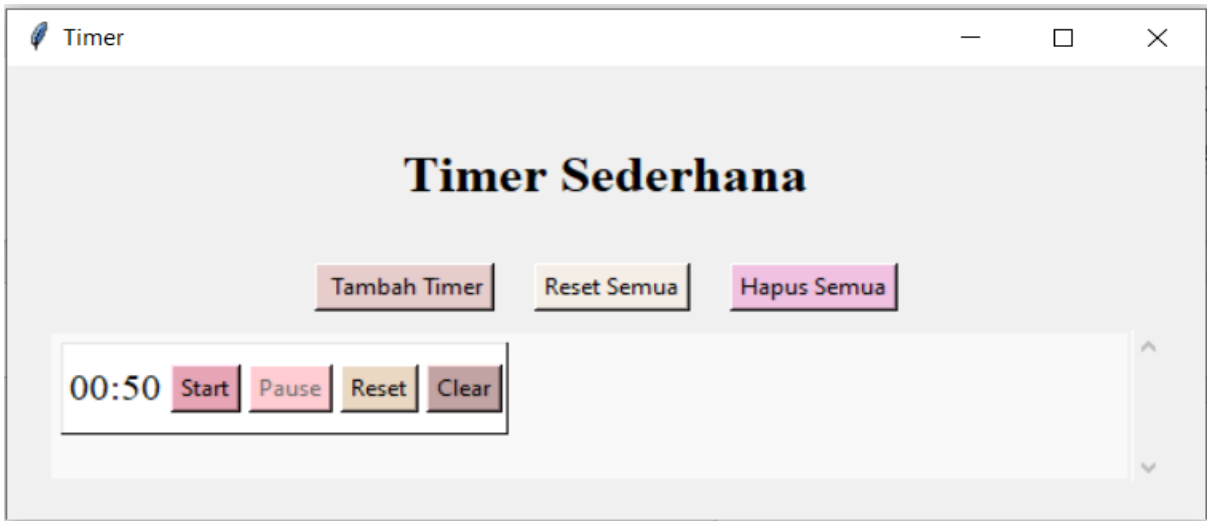
- Setelah timer dihentikan, langkah ini menghapus frame yang menyimpan elemen-elemen GUI terkait dengan timer tersebut (misalnya, label, tombol Start/Pause/Reset).
- Fungsi frame.destroy() menghapus seluruh elemen frame dari antarmuka pengguna, sehingga timer tidak akan terlihat lagi di layar.

```
self.timers.remove(timer_info)
```

Setelah frame dihapus, informasi timer yang bersangkutan dihapus dari daftar self.timers dengan menggunakan remove(). Hal ini memastikan bahwa timer tersebut tidak lagi disimpan dalam daftar dan tidak akan diproses lebih lanjut dalam aplikasi.

Output:

Setelah menghapus salah satu timer:



8) Fungsi untuk menghapus semua timer yang ada

```
def clear_semua(self):
        confirm = messagebox.askquestion("Konfirmasi", "Yakin ingin menghapus
semua timer?")

        if confirm == "yes":
            for timer in self.timers:
                timer['frame'].destroy()
                self.timers.clear()

        messagebox.showinfo("Informasi", "Semua timer berhasil dihapus!")
```

Penjelasan:

```
confirm = messagebox.askquestion("Konfirmasi", "Yakin ingin menghapus semua
timer?")
```

- Fungsi messagebox.askquestion() digunakan untuk menampilkan kotak dialog dengan dua pilihan: "Yes" atau "No".
- "Yes" berarti pengguna ingin melanjutkan penghapusan semua timer, sementara "No" berarti pengguna membatalkan tindakan tersebut.
- Nilai yang dikembalikan oleh askquestion adalah string "yes" atau "no", dan ini disimpan dalam variabel confirm.

```
if confirm == "yes":
```

Jika pengguna memilih "Yes" (yaitu confirm == "yes"), maka langkah berikutnya akan dijalankan untuk menghapus semua timer.

```
for timer in self.timers:
    timer['frame'].destroy()
    self.timers.clear()
```

- for timer in self.timers: Ini akan melakukan loop untuk setiap timer yang ada dalam daftar self.timers.

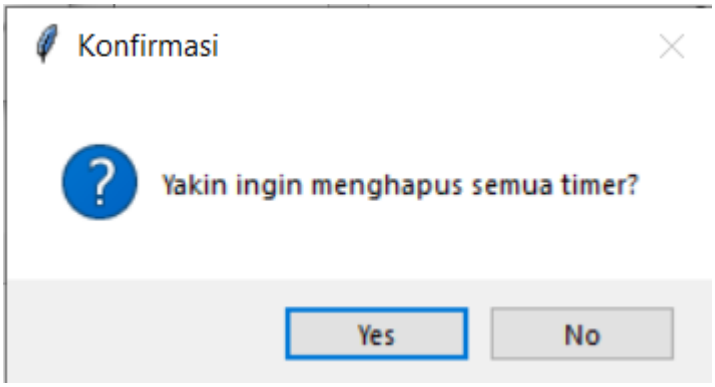
- timer['frame'].destroy(): Setiap timer diwakili oleh sebuah dictionary yang menyimpan berbagai informasi tentang timer tersebut, termasuk frame yang mewakili elemen GUI (misalnya, label dan tombol terkait). Fungsi destroy() dipanggil untuk menghapus frame dari antarmuka pengguna.
- self.timers.clear(): Setelah menghapus semua frame, fungsi ini membersihkan daftar self.timers, menghapus semua data terkait timer yang ada dalam aplikasi.

```
messagebox.showinfo("Informasi", "Semua timer berhasil dihapus!")
```

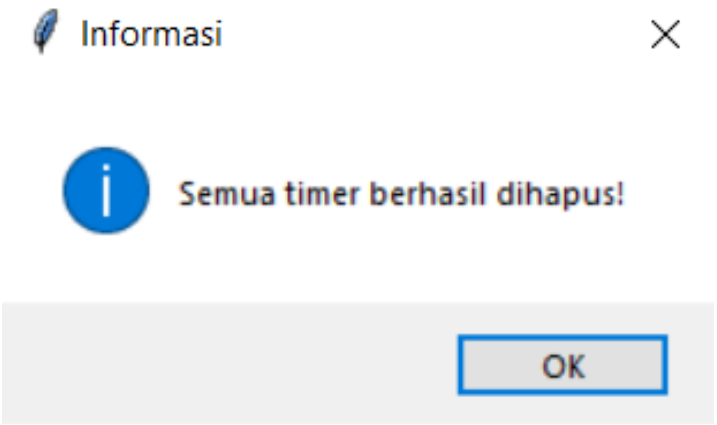
Setelah semua timer dihapus, sebuah kotak dialog informasi ditampilkan untuk memberitahukan pengguna bahwa semua timer telah berhasil dihapus. Kotak dialog ini menggunakan messagebox.showinfo() untuk menampilkan pesan berjudul "Informasi" dan pesan "Semua timer berhasil dihapus!".

Output:

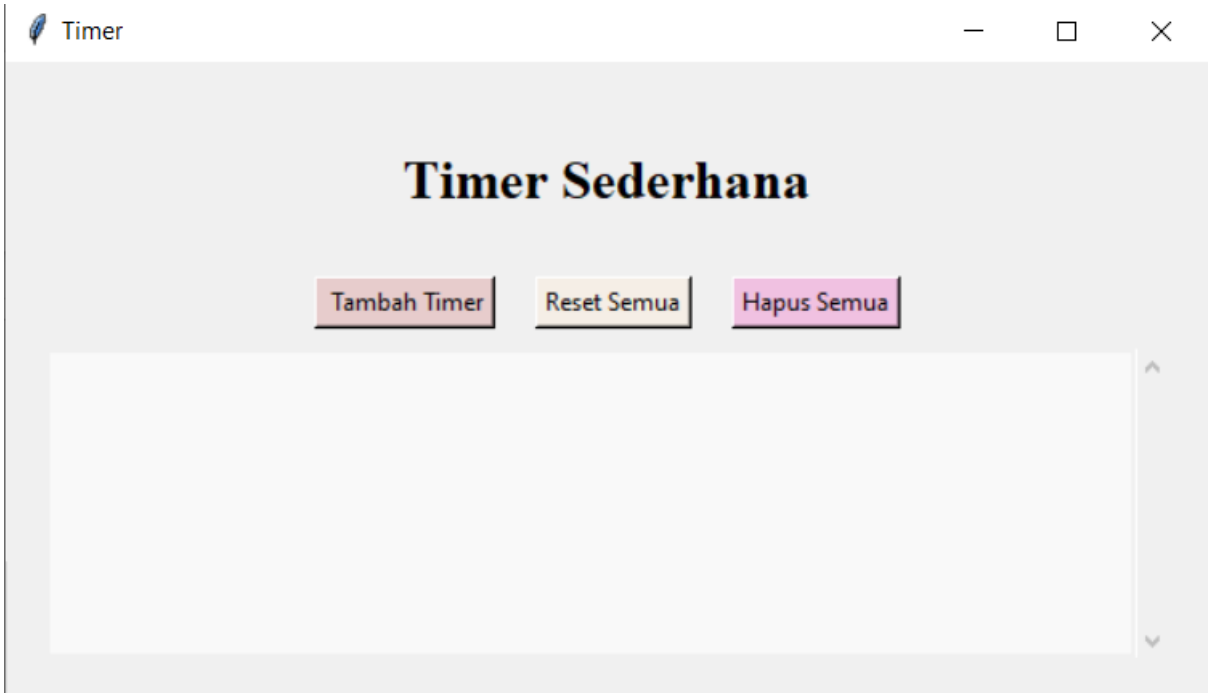
Dialog konfirmasi menghapus semua timer:



Jika memilih “Yes”



Setelah di hapus semua:



3. Fungsi Utama (__main__)

```
if __name__ == "__main__":
    root = tk.Tk()
    App = Timer(root)
    root.mainloop()
```

Penjelasan:

Digunakan untuk membuat jendela aplikasi utama (root) menggunakan Tkinter, membuat instance dari kelas Timer untuk menginisialisasi aplikasi, dan memanggil mainloop agar aplikasi tetap berjalan hingga ditutup oleh pengguna.

BAB III

PENUTUP

A. Kesimpulan

Studi kasus pembuatan aplikasi timer menggunakan Tkinter menunjukkan bahwa Python dengan library bawaan seperti Tkinter adalah pilihan yang efektif untuk membangun aplikasi GUI sederhana namun fungsional. Dengan memanfaatkan fitur bawaan Tkinter, aplikasi ini mampu menawarkan fungsi seperti penambahan timer, kontrol waktu (*start, pause, reset*), hingga penghapusan timer secara interaktif. Proses pengembangan ini tidak hanya melibatkan pemahaman teknis seperti threading dan pengelolaan event, tetapi juga desain antarmuka yang ramah pengguna.