Laporan UAS Pemograman Berorientasi Objek
Aplikasi FunMath

Oleh:

I Made Agus Mahayasa (24091397165)

Program Studi D4 Manajemen Informatika

Fakultas Vokasi

Universitas Negeri Surabaya

2025

KATA PENGANTAR

Puji syukur saya panjatkan ke hadirat Tuhan Yang Maha Esa, karena berkat rahmat dan karunia-Nya laporan yang berjudul "Laporan UAS Pemograman Berorientasi Objek Aplikasi FunMath" ini dapat diselesaikan tepat waktu dengan baik. Laporan ini disusun sebagai bagian dari pemenuhan tugas ujian akhir semester (UAS) mata kuliah Pemograman berorientasi Objek (PBO) yang diampu oleh Bapak M Adamu Islam Mashuri, S.Tr.T., M.Tr.Kom.

Laporan ini berisikan penjelasan lengkap dari aplikasi yang saya buat yaitu FunMath yang merupakan sebuah aplikasi pembelajaran matematika sederhana berbasis Python dengan antarmuka grafis menggunakan Tkinter serta dukungan audio dari Pygame, serta menerapkan konsep Object Oriented Programming (OOP) di dalamnya. Melalui pengembangan aplikasi ini, saya berupaya memahami dan mengimplementasikan konsep OOP secara nyata dalam sebuah studi kasus aplikasi nyata pengoprasian matematika sederhana.

Saya menyadari penuh bahwa laporan ini masih jauh dari kata sempurna, memiliki banyak keterbatasan dan kekurangan, baik dari segi penulisan maupun pengembangan aplikasi. Oleh karena itu, saya sangat mengharapkan kritik dan saran yang membangun demi perbaikan di masa mendatang. Akhir kata, saya berharap laporan ini dapat memberikan manfaat bagi siapapun yang membacanya, baik sebagai referensi pembelajaran maupun sebagai gambaran penerapan konsep OOP dalam pengembangan aplikasi sederhana.

DAFTAR ISI

BAB I

PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi informasi telah membawa perubahan signifikan dalam berbagai bidang, termasuk bidang pendidikan. Pemanfaatan teknologi dalam proses pembelajaran dinilai mampu meningkatkan minat belajar serta membantu peserta didik memahami materi secara lebih interaktif dan menyenangkan. Salah satu mata pelajaran yang kurang dimitati karena dianggap sulit oleh sebagian pelajar adalah matematika, terutama dalam memahami operasi dasar seperti penjumlahan, pengurangan, perkalian, dan pembagian. Oleh karena itu, diperlukan sebuah media pembelajaran alternatif yang dapat membantu pengguna berlatih matematika secara mandiri dengan cara yang lebih menarik.

Untuk menjawab tantangan tersebut, FunMath saya rancang agar pembelajaran matematika yang sulit dan sering dianggap membosankan itu berubah menjadi pembelajaran yang menarik dan menyenangkan. Aplikasi pembelajaran matematika ini dikembangkan menggunakan bahasa pemrograman Python dengan antarmuka grafis berbasis Tkinter serta dukungan audio menggunakan Pygame. Yang dimana aplikasi ini dirancang agar pengguna dapat memilih tingkat kesulitan dan jenis operasi matematika sesuai dengan kemauan dan kebutuhan pengguna, serta memperoleh umpan balik langsung atas jawaban yang diberikan. Dengan adanya aplikasi ini, diharapkan proses pembelajaran matematika dapat menjadi lebih interaktif, sekaligus menjadi sarana pembelajaran yang efektif dalam memahami penerapan konsep OOP pada pengembangan aplikasi.

1.2 Rumusan Masalah

1. Bagaimana cara menerapkan konsep OOP dalam aplikasi pembelajaran matematika?
2. Bagaimana cara merancang aplikasi matematika dengan antarmuka grafis yang menarik dan interaktif?

1.3 Tujuan

1. Menerapkan konsep OOP dalam pengembangan aplikasi.
2. Membuat aplikasi edukasi matematika berbasis GUI.
3. Memberikan pengalaman belajar yang interaktif dan menarik kepada pengguna.

BAB II

LANDASAN TEORI

2.1 Object-Oriented Programming (OOP)

Object-Oriented Programming (OOP) merupakan paradigma pemrograman yang berfokus pada penggunaan objek sebagai representasi dari data dan perilaku dalam suatu sistem. OOP bertujuan untuk membuat program lebih terstruktur, mudah dipahami, serta mudah dikembangkan dan dipelihara.

Konsep utama OOP meliputi:

- **Encapsulation**: Penyembunyian data menggunakan atribut private.
- **Inheritance**: Pewarisan sifat dari kelas induk ke kelas turunan.
- **Polymorphism**: Metode dengan nama yang sama tetapi perilakunya berbeda.

2.2 Python

Python merupakan bahasa pemrograman tingkat tinggi yang mendukung paradigma Object-Oriented Programming dan dikenal dengan sintaks yang sederhana serta mudah dipahami. Python banyak digunakan dalam pengembangan aplikasi desktop, web, maupun data science. Dukungan pustaka yang luas menjadikan Python sangat cocok digunakan sebagai media pembelajaran sekaligus pengembangan aplikasi edukatif.

2.3 Tkinter dan Pygame

Kombinasi Tkinter dan Pygame memungkinkan pengembangan aplikasi interaktif dengan tampilan grafis yang sederhana serta dukungan suara, sehingga meningkatkan pengalaman pengguna dalam menggunakan aplikasi.

- **Tkinter** digunakan untuk membangun antarmuka grafis (GUI).
- **Pygame** digunakan untuk pengelolaan audio seperti musik latar dan efek suara.

BAB III

PERANCANGAN SISTEM

3.1 Gambaran Umum Aplikasi

Aplikasi FunMath merupakan aplikasi edukasi berbasis desktop yang dirancang untuk membantu pengguna berlatih operasi matematika dasar secara interaktif. Aplikasi ini menyediakan berbagai jenis operasi dasar matematika, yaitu penjumlahan, pengurangan, perkalian, dan pembagian, yang dapat dipilih sesuai kebutuhan pengguna. Selain itu, aplikasi juga menyediakan pilihan tingkat kesulitan, mulai dari mudah, sedang, hingga sulit, sehingga pengguna dapat menyesuaikan tingkat tantangan dengan kemampuan masing-masing.

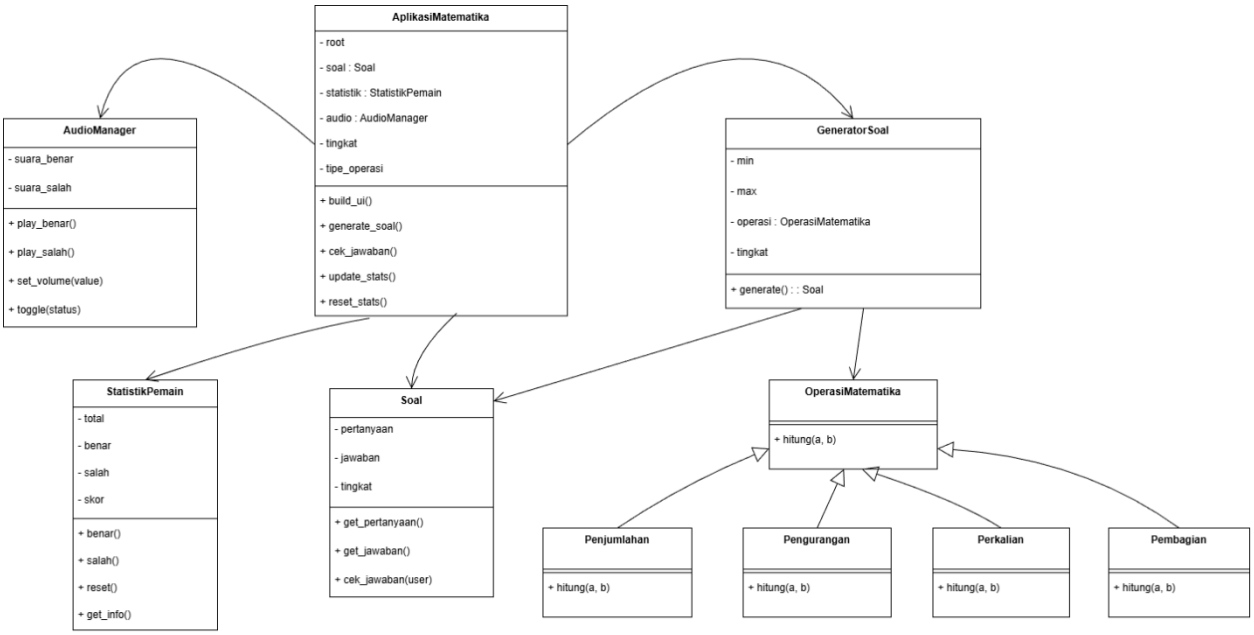
Aplikasi ini dikembangkan menggunakan bahasa pemrograman Python dengan pendekatan Object-Oriented Programming (OOP). Antarmuka grafis dibangun menggunakan pustaka Tkinter, sementara fitur audio seperti musik latar dan efek suara memanfaatkan pustaka Pygame. Penggunaan OOP memungkinkan pemisahan tanggung jawab antar kelas, seperti pengelolaan soal, statistik pemain, dan audio, sehingga struktur program menjadi lebih terorganisir dan mudah dikembangkan.

Secara umum, alur penggunaan aplikasi dimulai dari pemilihan jenis operasi dan tingkat kesulitan, kemudian sistem akan menghasilkan soal secara acak. Pengguna memasukkan jawaban melalui kolom input, dan aplikasi akan memberikan umpan balik secara langsung berupa teks dan suara. Statistik permainan seperti skor, jumlah jawaban benar, salah, serta tingkat akurasi ditampilkan secara real-time, sehingga pengguna dapat memantau perkembangan hasil belajarnya.

3.2 Diagram Class

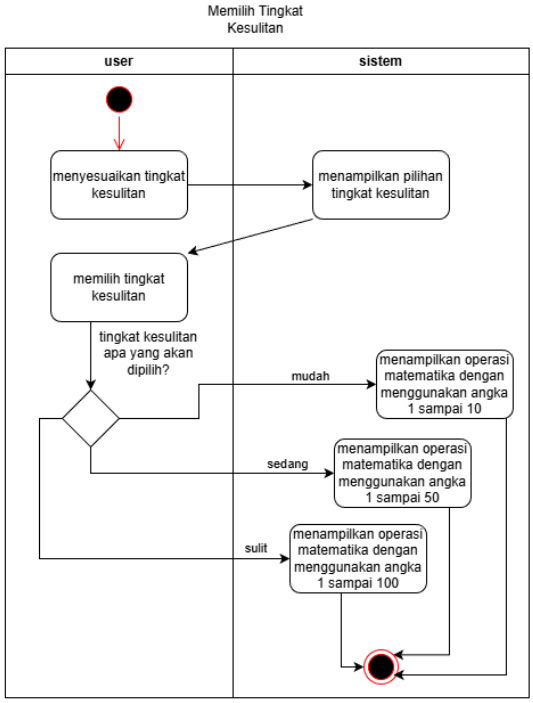
Aplikasi ini terdiri dari beberapa kelas utama, antara lain:

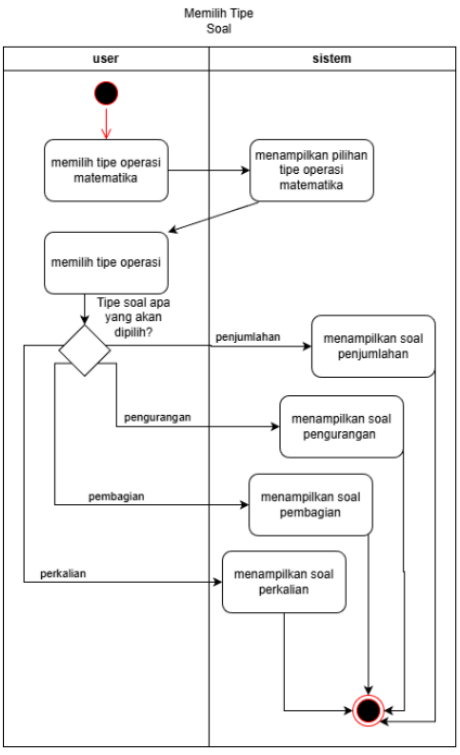
- AplikasiMatematika
- Soal dan generator soal
- StatistikPemain
- AudioManager
- OperasiMatematika dan turunannya



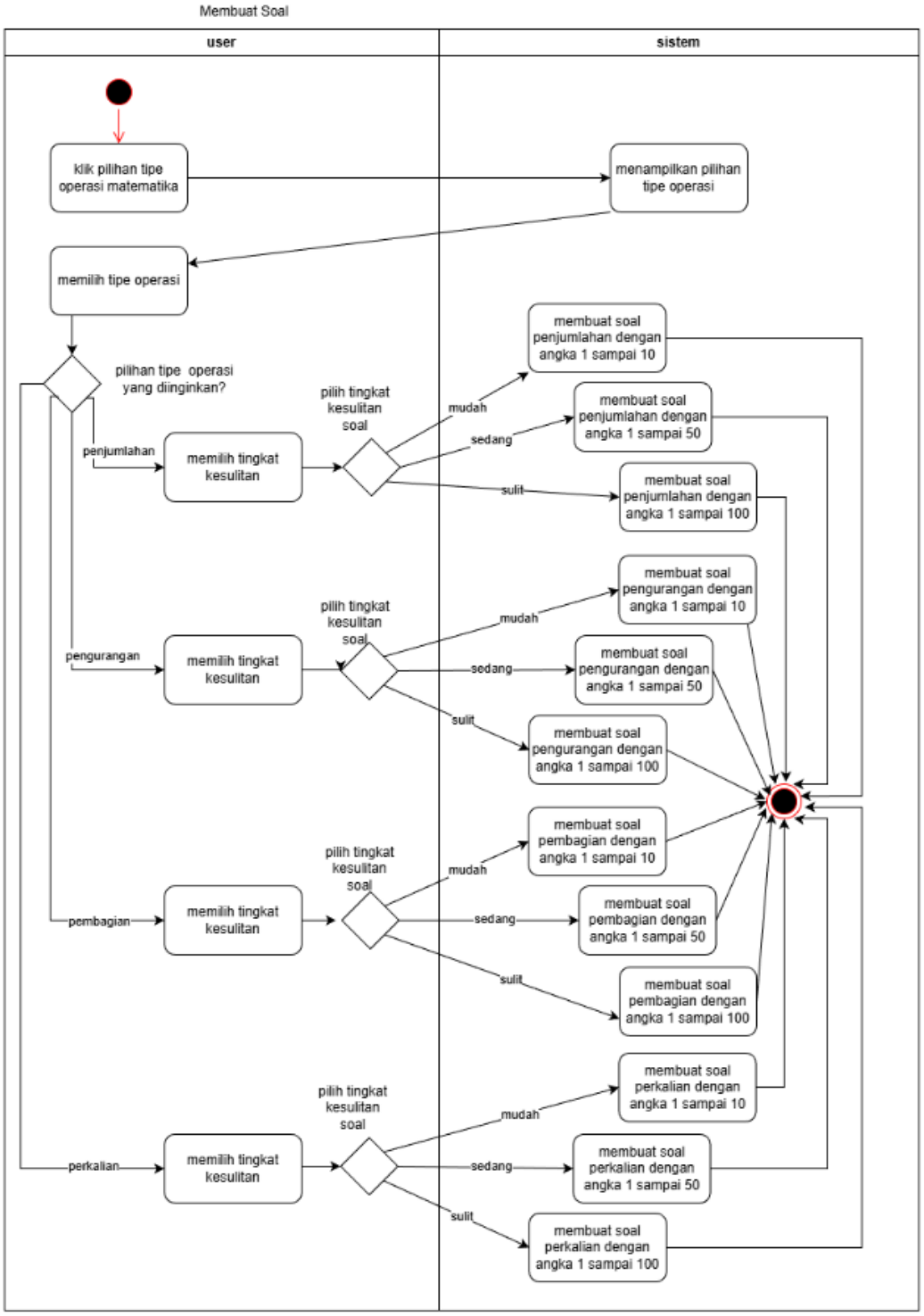
3.3 Alur Aplikasi

1. Aplikasi dijalankan
2. Pengguna memilih tingkat dan jenis operasi

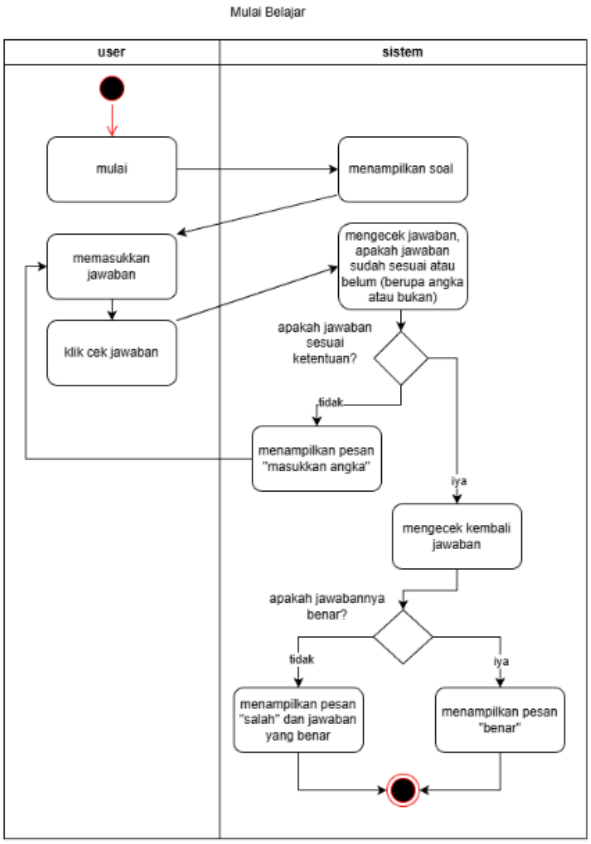




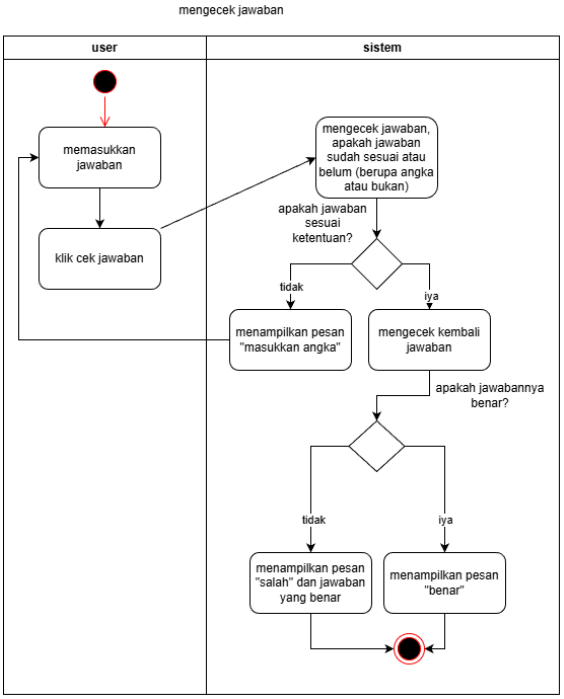
3. Sistem menghasilkan soal



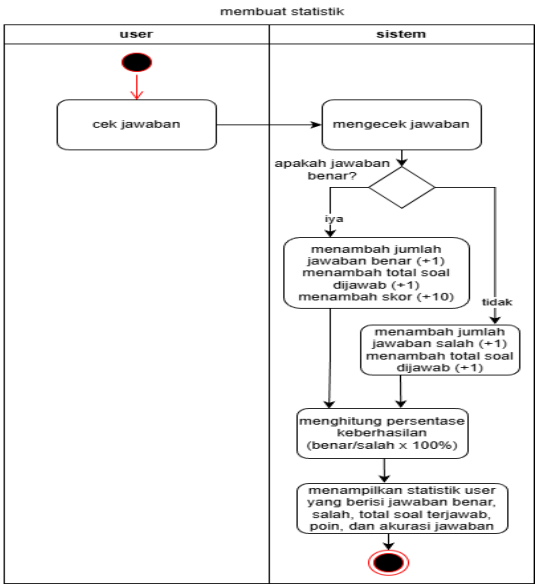
4. Pengguna memasukkan jawaban



5. Sistem memeriksa jawaban

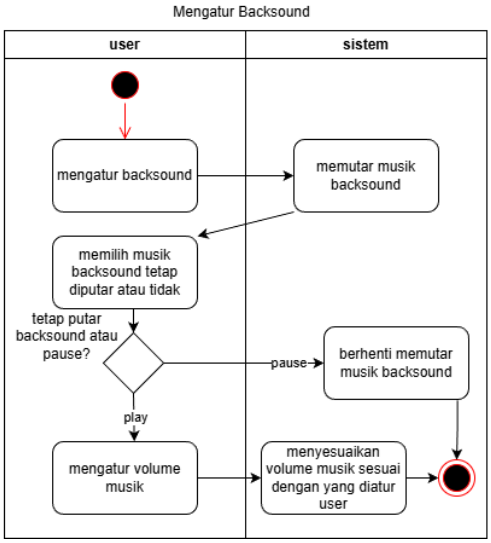


6. Statistik diperbarui

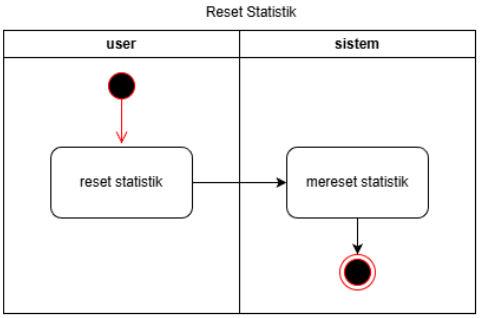


Opsional

1. Mengatur music latar



2. Mereset statistic



BAB IV

IMPLEMENTASI DAN PEMBAHASAN

4.1 Struktur Program

Struktur program pada aplikasi FunMath disusun secara modular dengan memisahkan fungsi utama ke dalam beberapa bagian agar mudah dipahami dan dikembangkan. Secara umum, struktur program terdiri dari bagian import library, pembuatan jendela utama aplikasi (GUI main window), penambahan widget antarmuka, serta main loop sebagai pengendali jalannya aplikasi.

Import Library

```
1  import tkinter as tk
2  import pygame
3  import random
4
```

Pada bagian awal program dilakukan proses import library yang dibutuhkan. Library tkinter digunakan untuk membangun antarmuka grafis (GUI), pygame digunakan untuk mengelola audio seperti musik latar dan efek suara, sedangkan random digunakan untuk menghasilkan soal matematika secara acak. Import library ini merupakan fondasi utama aplikasi karena seluruh komponen GUI, audio, dan logika pengacakan soal bergantung pada pustaka-pustaka tersebut.

Create the GUI Application Main Window

Pembuatan jendela utama aplikasi dilakukan dengan memanfaatkan objek Tk() dari library Tkinter. Jendela utama ini berfungsi sebagai wadah seluruh komponen antarmuka aplikasi. Proses inisialisasi GUI dilakukan di bagian main program dan di dalam konstruktor kelas AplikasiMatematika.

```
299      root = tk.Tk()
300      AplikasiMatematika(root)
```

Di dalam konstruktor kelas AplikasiMatematika, dilakukan pengaturan judul jendela, ukuran layar, serta warna latar belakang aplikasi.

```python
class AplikasiMatematika:
    """Kelas utama GUI permainan matematika"""

    def __init__(self, root):
        self.root = root
        self.root.title("✨ FUNMATH ✨")
        self.root.geometry("750x550")
        self.root.minsize(720, 500)
        self.root.configure(bg="#E3F2FD")

        # Property aplikasi
        self.soal = None
        self.boleh_lanjut = False
        self.music_playing = True

        # Model
        self.statistik = StatistikPemain()
        self.audio = AudioManager()

        self.tingkat = tk.StringVar(value="Mudah")
        self.tipe_operasi = tk.StringVar(value="Penjumlahan")

        self.build_ui()
        self.update_stats()
```

Add Widget (Penambahan Komponen Antarmuka)

Widget merupakan elemen antarmuka yang memungkinkan pengguna berinteraksi dengan aplikasi. Pada aplikasi ini, widget ditambahkan melalui metode build_ui(). Beberapa widget utama yang digunakan antara lain Label, Button, Entry, Frame, OptionMenu, dan Scale.

```python

    # ==================== USER INTERFACE ====================
    def build_ui(self):
        """Membangun antarmuka aplikasi"""

        header = tk.Label(
            self.root, text="✨ Belajar Matematika Ceria ✨",
            font=("Comic Sans MS", 24, "bold"), bg="#1976D2", fg="white", pady=10
        )
        header.pack(fill="x")

        main = tk.Frame(self.root, bg="#E3F2FD")
        main.pack(expand=True, fill="both")

        # LEFT PANEL
        left = tk.Frame(main, bg="#BBDEFB", width=130)
        left.pack(side="left", fill="y")

        tk.Label(left, text="🎵 Volume", font=("Arial", 10, "bold"), bg="#BBDEFB").pack(pady=10)

        volume_slider = tk.Scale(
            left, from_=100, to=0, orient="vertical",
            command=lambda v: AudioManager.set_volume(int(v) / 100),
            length=200, bg="#BBDEFB", highlightthickness=0
        )
        volume_slider.set(50)
        volume_slider.pack(pady=5)

        self.btn_music = tk.Button(
            left, text="Pause Musik", bg="#1565C0", fg="white", width=12,
            command=self.toggle_music
        )
        self.btn_music.pack(pady=10)

        # CENTER PANEL
        center = tk.Frame(main, bg="#E3F2FD")
        center.pack(expand=True, fill="both")

        top_controls = tk.Frame(center, bg="#E3F2FD")
        top_controls.pack(pady=10)

        tk.OptionMenu(top_controls, self.tipe_operasi,
                      "Penjumlahan", "Pengurangan", "Perkalian", "Pembagian").pack(side="left", padx=8)

        tk.OptionMenu(top_controls, self.tingkat,
                      "Mudah", "Sedang", "Sulit").pack(side="left", padx=8)

        self.label_soal = tk.Label(center, text="Klik Mulai untuk mulai",
                                   font=("Arial", 26, "bold"), bg="#E3F2FD")
        self.label_soal.pack(pady=20)

        self.entry = tk.Entry(center, font=("Arial", 22), width=6, justify="center")
        self.entry.pack()
        self.entry.bind("<Return>", lambda e: self.cek_jawaban())

        self.btn_next = tk.Button(center, text="Mulai", bg="#43A047", fg="white",
                                  font=("Arial", 14, "bold"), command=self.generate_soal)
        self.btn_next.pack(pady=10)

        tk.Button(center, text="Cek Jawaban", bg="#0288D1", fg="white",
                  font=("Arial", 13, "bold"), command=self.cek_jawaban).pack(pady=5)

        self.label_feedback = tk.Label(center, font=("Arial", 22, "bold"), bg="#E3F2FD")
        self.label_feedback.pack(pady=10)

        self.label_stats = tk.Label(center, font=("Arial", 14, "bold"), bg="#E3F2FD")
        self.label_stats.pack(pady=10)

        tk.Button(center, text="Reset Statistik", bg="#FB8C00", fg="white",
                  font=("Arial", 13, "bold"), command=self.reset_stats).pack(pady=10)

```

Selain itu, aplikasi juga menggunakan widget Entry sebagai media input jawaban serta OptionMenu untuk memilih jenis operasi dan tingkat kesulitan.

Main Loop Aplikasi

Main loop merupakan bagian terpenting yang menjaga aplikasi tetap berjalan dan responsif terhadap interaksi pengguna. Pada aplikasi ini, main loop dijalankan menggunakan metode mainloop() dari Tkinter.

```
301     root.mainloop()
```

Main loop berfungsi untuk menangani event seperti klik tombol, input keyboard, dan pembaruan tampilan GUI. Selama main loop berjalan, aplikasi akan terus aktif hingga pengguna menutup jendela aplikasi.

4.2 Implementasi Konsep OOP

Pada pengembangan aplikasi FunMath pendekatan Object-Oriented Programming (OOP) diterapkan secara konsisten untuk meningkatkan keteraturan struktur program, kemudahan pemeliharaan, serta pengembangan fitur di masa depan. Konsep OOP yang diimplementasikan meliputi Encapsulation, Inheritance, dan Polymorphism, yang dijelaskan sebagai berikut.

Encapsulation

Encapsulation merupakan konsep OOP yang bertujuan untuk membungkus data (atribut) dan metode dalam satu kesatuan kelas serta membatasi akses langsung terhadap data tersebut dari luar kelas. Tujuan utama encapsulation adalah menjaga keamanan data, mencegah perubahan nilai yang tidak terkontrol, serta memastikan bahwa perubahan data hanya dilakukan melalui mekanisme yang telah ditentukan.

Pada aplikasi ini, encapsulation diterapkan dengan menggunakan atribut privat yang diawali dengan tanda __. Atribut privat tersebut tidak dapat diakses langsung dari luar kelas, melainkan melalui metode publik (getter atau method logika). Contoh penerapan encapsulation dapat dilihat pada kelas Soal.

```python
class Soal:
    """Menyimpan data soal dan jawaban (encapsulation)"""

    def __init__(self, pertanyaan, jawaban, tingkat):
        self.__pertanyaan = pertanyaan
        self.__jawaban = jawaban
        self.__tingkat = tingkat

    def get_pertanyaan(self):
        return self.__pertanyaan

    def get_jawaban(self):
        return self.__jawaban

    def cek_jawaban(self, user):
        return int(user) == self.__jawaban
```

Pada kode di atas, atribut __pertanyaan, __jawaban, dan __tingkat disembunyikan dari akses langsung. Aplikasi hanya dapat menampilkan pertanyaan atau jawaban melalui metode get_pertanyaan() dan get_jawaban(). Dengan cara ini, data soal tidak dapat diubah secara sembarangan oleh bagian program lain.

Encapsulation juga diterapkan pada kelas StatistikPemain, di mana data statistik seperti skor, jumlah jawaban benar, dan salah hanya dapat dimodifikasi melalui metode tertentu.

```python
class StatistikPemain:
    """Menyimpan statistik pemain (encapsulation)"""

    def __init__(self):
        self.__total = 0
        self.__benar = 0
        self.__salah = 0
        self.__skor = 0

    def benar(self):
        self.__benar += 1
        self.__total += 1
        self.__skor += 10

    def salah(self):
        self.__salah += 1
        self.__total += 1

    def reset(self):
        self.__benar = self.__salah = self.__total = self.__skor = 0

    def get_info(self):
        akurasi = (self.__benar / self.__total) * 100 if self.__total else 0
        return self.__skor, self.__benar, self.__salah, akurasi
```

Dengan pendekatan ini, setiap perubahan data statistik selalu mengikuti aturan yang telah ditetapkan, sehingga konsistensi data tetap terjaga.

Inheritance

Inheritance atau pewarisan merupakan konsep OOP yang memungkinkan suatu kelas turunan (subclass) mewarisi atribut dan metode dari kelas induk (superclass). Konsep ini digunakan untuk mengurangi duplikasi kode dan mempermudah pengembangan sistem. Dalam aplikasi ini, inheritance diterapkan pada fitur operasi matematika. Yang dimana kelas OperasiMatematika berperan sebagai kelas induk atau super class yang mendefinisikan metode umum hitung().

```python
# ==================== Inheritance & polymorphism ====================
class OperasiMatematika: #super class
    """Kelas dasar operasi matematika (polymorphism)"""

    def hitung(self, a, b):
        raise NotImplementedError


class Penjumlahan(OperasiMatematika): #sub class
    def hitung(self, a, b):
        return f"{a} + {b} = ?", a + b


class Pengurangan(OperasiMatematika): #sub class
    def hitung(self, a, b):
        return f"{max(a, b)} - {min(a, b)} = ?", abs(a - b)


class Perkalian(OperasiMatematika): #sub class
    def hitung(self, a, b):
        return f"{a} × {b} = ?", a * b


class Pembagian(OperasiMatematika): #sub class
    def hitung(self, a, b):
        return f"{a * b} ÷ {b} = ?", a
```

Kelas ini tidak diimplementasikan secara langsung, melainkan diwarisi oleh kelas-kelas turunan atau sub classnya seperti Penjumlahan, Pengurangan, Perkalian, dan Pembagian. Setiap kelas turunan mengimplementasikan metode hitung() sesuai dengan logika operasinya masing-masing. Dengan inheritance, struktur program menjadi lebih rapi dan mudah diperluas. Jika di masa depan ingin menambahkan jenis operasi baru, pengembang cukup membuat kelas turunan baru tanpa perlu mengubah kode inti aplikasi.

Polymorphism

Polymorphism adalah konsep OOP yang memungkinkan metode dengan nama yang sama memiliki perilaku yang berbeda tergantung pada objek yang memanggilnya. Polymorphism membuat program menjadi lebih fleksibel dan mendukung pengembangan berbasis abstraksi.

Pada aplikasi ini, polymorphism diterapkan melalui metode hitung() yang dimiliki oleh semua kelas turunan dari OperasiMatematika. Meskipun metode yang dipanggil memiliki nama yang sama, hasil dan proses perhitungannya berbeda sesuai dengan jenis operasi. Dan jenis Polymorphism yang Digunakan adalah Method Overriding

Method overriding terjadi ketika kelas turunan menuliskan ulang (meng-override) metode yang sudah didefinisikan di kelas induk, dengan nama metode yang sama tetapi isi (implementasi) yang berbeda.

```python
# ==================== Inheritance & polymorphism ====================
class OperasiMatematika: #super class
    """Kelas dasar operasi matematika (polymorphism)"""

    def hitung(self, a, b):
        raise NotImplementedError


class Penjumlahan(OperasiMatematika): #sub class
    def hitung(self, a, b):
        return f"{a} + {b} = ?", a + b


class Pengurangan(OperasiMatematika): #sub class
    def hitung(self, a, b):
        return f"{max(a, b)} - {min(a, b)} = ?", abs(a - b)


class Perkalian(OperasiMatematika): #sub class
    def hitung(self, a, b):
        return f"{a} × {b} = ?", a * b


class Pembagian(OperasiMatematika): #sub class
    def hitung(self, a, b):
        return f"{a * b} ÷ {b} = ?", a
```

Bisa dilihat pada potongan kode tersebut, super class atau kelas induk memiliki method hitung() yang diwarisi ke semua sub class namun isi atau prilaku method pada setiap sub class berbeda. Penerapan polymorphism ini membuat kode lebih modular, mudah dibaca, dan sesuai dengan prinsip open-closed, yaitu sistem terbuka untuk penambahan fitur baru tetapi tertutup terhadap perubahan kode yang sudah ada.

4.3 Interaksi Pengguna

Interaksi pengguna pada aplikasi FunMath direalisasikan melalui komponen antarmuka grafis (GUI) yang dibangun menggunakan pustaka Tkinter. Setiap interaksi pengguna, seperti memilih menu, menekan tombol, atau memasukkan jawaban, akan diproses oleh metode tertentu pada kelas AplikasiMatematika. Subbab ini menjelaskan bagaimana interaksi tersebut diimplementasikan dari sisi pemrograman.

Pemilihan Jenis Operasi dan Tingkat Kesulitan

Pengguna berinteraksi dengan aplikasi pertama kali melalui pemilihan jenis operasi matematika dan tingkat kesulitan menggunakan widget OptionMenu. Nilai pilihan pengguna disimpan dalam variabel bertipe StringVar agar dapat digunakan oleh sistem saat menghasilkan soal.

```python
self.tingkat = tk.StringVar(value="Mudah")
self.tipe_operasi = tk.StringVar(value="Penjumlahan")
```

```
        tk.OptionMenu(top_controls, self.tipe_operasi,
                      "Penjumlahan", "Pengurangan", "Perkalian", "Pembagian").pack(side="left", padx=8)

        tk.OptionMenu(top_controls, self.tingkat,
                      "Mudah", "Sedang", "Sulit").pack(side="left", padx=8)
```

Melalui mekanisme ini, sistem dapat menyesuaikan jenis soal yang dihasilkan berdasarkan preferensi pengguna.

Interaksi Tombol Mulai dan Soal Selanjutnya

Interaksi selanjutnya dilakukan melalui tombol **Mulai** atau **Soal Selanjutnya**. Tombol ini terhubung dengan metode generate_soal(), yang bertugas membuat dan menampilkan soal baru.

```
self.btn_next = tk.Button(center, text="Mulai", bg="#43A047", fg="white",
                          font=("Arial", 14, "bold"), command=self.generate_soal)
self.btn_next.pack(pady=10)
```

Ketika tombol ditekan, aplikasi akan memanggil generator soal, lalu menampilkan pertanyaan pada label soal. Tombol yang sama juga digunakan untuk melanjutkan ke soal berikutnya setelah jawaban diperiksa.

Input Jawaban Pengguna

Pengguna memasukkan jawaban melalui widget Entry. Untuk meningkatkan kenyamanan, aplikasi mendukung dua cara pengiriman jawaban, yaitu dengan menekan tombol Cek Jawaban atau tombol Enter pada keyboard.

```
self.entry = tk.Entry(center, font=("Arial", 22), width=6, justify="center")
self.entry.pack()
self.entry.bind("<Return>", lambda e: self.cek_jawaban())
```

Pendekatan ini membuat interaksi lebih fleksibel dan responsif terhadap kebiasaan pengguna.

Pemeriksaan Jawaban dan Umpan Balik

Setelah pengguna mengirimkan jawaban, metode cek_jawaban() akan dijalankan. Metode ini memeriksa kesesuaian jawaban pengguna dengan jawaban yang benar, kemudian memberikan umpan balik berupa teks dan suara.

```
if self.soal.cek_jawaban(self.entry.get()):
    self.statistik.benar()
    self.audio.play_benar()
    self.label_feedback.config(text="✓ Benar!", fg="green")
else:
    self.statistik.salah()
    self.audio.play_salah()
    self.label_feedback.config(text=f"✗ Salah! Jawaban: {self.soal.get_jawaban()}", fg="red")
```

Jika jawaban benar maka sistem akan menampilkan pesan jawaban benar dengan warna hijau dan juga suara latar jawaban benar, sedangkan jika salah pesan yang ditampilkan jawaban salah dilengkapi dengan jawaban yang benar dengan warna merah dan dilengkapi suara music salah. Dengan umpan balik langsung ini bertujuan dapat meningkatkan keterlibatan pengguna dan membantu proses pembelajaran menjadi lebih menarik dan menyenangkan.

Tampilan Statistik Permainan

Setiap interaksi pengguna akan memengaruhi statistik permainan. Informasi seperti skor, jumlah jawaban benar, salah, dan tingkat akurasi ditampilkan secara real-time menggunakan widget Label.

```
self.label_stats.config(
    text=f"Skor: {skor} | Benar: {benar} | Salah: {salah} | Akurasi: {akurasi:.1f}%"
)
```

Pengguna juga dapat menekan tombol Reset Statistik untuk mengulang pencatatan statistik dari awal.

```
tk.Button(center, text="Reset Statistik", bg="#FB8C00", fg="white",
          font=("Arial", 13, "bold"), command=self.reset_stats).pack(pady=10)
```

Kontrol Audio sebagai Interaksi Tambahan

Selain interaksi utama, aplikasi menyediakan fitur pengaturan volume dan tombol pause/play musik latar sebagai bentuk interaksi tambahan.

```
self.btn_music = tk.Button(
    left, text="Pause Musik", bg="#1565C0", fg="white", width=12,
    command=self.toggle_music
)
self.btn_music.pack(pady=10)
```

Fitur ini memungkinkan pengguna menyesuaikan pengalaman penggunaan aplikasi sesuai preferensi masing-masing.

BAB V

TANTANGAN DAN SOLUSI

Dalam proses pengembangan aplikasi FunMath, terdapat beberapa tantangan yang dihadapi oleh pengembang, baik dari sisi teknis pemrograman maupun dari sisi perancangan aplikasi. Tantangan-tantangan ini menjadi bagian penting dalam proses pembelajaran karena mendorong pengembang untuk memahami konsep pemrograman secara lebih mendalam serta mencari solusi yang tepat dan efektif.

5.1 Tantangan dalam Penerapan Konsep OOP

Salah satu tantangan utama dalam pengembangan aplikasi ini adalah menerapkan konsep Object-Oriented Programming (OOP) secara konsisten dan sesuai dengan kaidah yang benar. Pada tahap awal, pengembang mengalami kesulitan dalam menentukan pembagian tanggung jawab antar kelas, khususnya dalam memisahkan logika aplikasi, pengelolaan data, dan antarmuka pengguna. Untuk mengatasi hal tersebut, solusi yang diterapkan adalah dengan melakukan perancangan ulang struktur kelas, sehingga setiap kelas memiliki satu tanggung jawab utama, seperti kelas Soal untuk data soal, StatistikPemain untuk statistik, dan AudioManager untuk pengelolaan audio.

5.2 Tantangan Integrasi Tkinter dan Pygame

Tantangan berikutnya adalah mengintegrasikan pustaka Tkinter dan Pygame dalam satu aplikasi. Tkinter digunakan untuk membangun antarmuka grafis, sedangkan Pygame digunakan untuk pengolahan audio. Pada awalnya, terjadi kendala terkait inisialisasi audio dan pengaturan volume yang tidak berjalan sesuai harapan. Solusi yang dilakukan adalah memusatkan seluruh pengelolaan audio ke dalam satu kelas khusus, yaitu AudioManager, sehingga inisialisasi pygame.mixer hanya dilakukan sekali dan pengaturan audio menjadi lebih terkontrol.

5.3 Tantangan Penanganan Input Pengguna

Penanganan input pengguna juga menjadi tantangan tersendiri, terutama dalam memastikan bahwa input yang diberikan berupa angka dan tidak menimbulkan kesalahan program. Kesalahan input seperti memasukkan huruf atau karakter non-numerik dapat menyebabkan error pada saat proses pemeriksaan jawaban. Untuk mengatasi masalah ini, diterapkan mekanisme penanganan kesalahan (exception handling) menggunakan blok try-except, sehingga aplikasi tetap berjalan dengan baik dan memberikan pesan peringatan yang informatif kepada pengguna.

5.4 Tantangan Desain Interaksi Pengguna

Merancang interaksi pengguna yang sederhana namun tetap interaktif juga menjadi tantangan dalam pengembangan aplikasi ini. Pengembang harus memastikan bahwa pengguna dapat dengan mudah memahami alur penggunaan aplikasi tanpa instruksi yang rumit. Solusi yang diterapkan adalah dengan menggunakan elemen GUI yang umum dan familiar, seperti tombol, menu pilihan, serta pemberian umpan balik langsung berupa teks dan suara. Selain itu, dukungan tombol Enter

sebagai alternatif tombol klik juga meningkatkan kenyamanan pengguna dalam berinteraksi dengan aplikasi.

5.5 Tantangan Pengujian dan Debugging

Tahap pengujian dan debugging merupakan tantangan terakhir yang dihadapi dalam pengembangan aplikasi. Beberapa kesalahan logika, seperti perhitungan statistik yang tidak sesuai atau pergantian soal yang tidak berjalan semestinya, ditemukan pada saat pengujian. Solusi yang dilakukan adalah dengan melakukan pengujian secara bertahap pada setiap fitur serta menambahkan validasi pada kondisi-kondisi tertentu. Dengan pendekatan ini, kesalahan dapat diidentifikasi dan diperbaiki lebih cepat.

Secara keseluruhan, berbagai tantangan yang dihadapi selama proses pengembangan aplikasi ini memberikan pengalaman belajar yang berharga bagi pengembang. Melalui penerapan solusi yang tepat, aplikasi FunMath berhasil dikembangkan menjadi aplikasi pembelajaran yang fungsional, interaktif, dan sesuai dengan prinsip Object-Oriented Programming.

BAB VI

PENUTUP

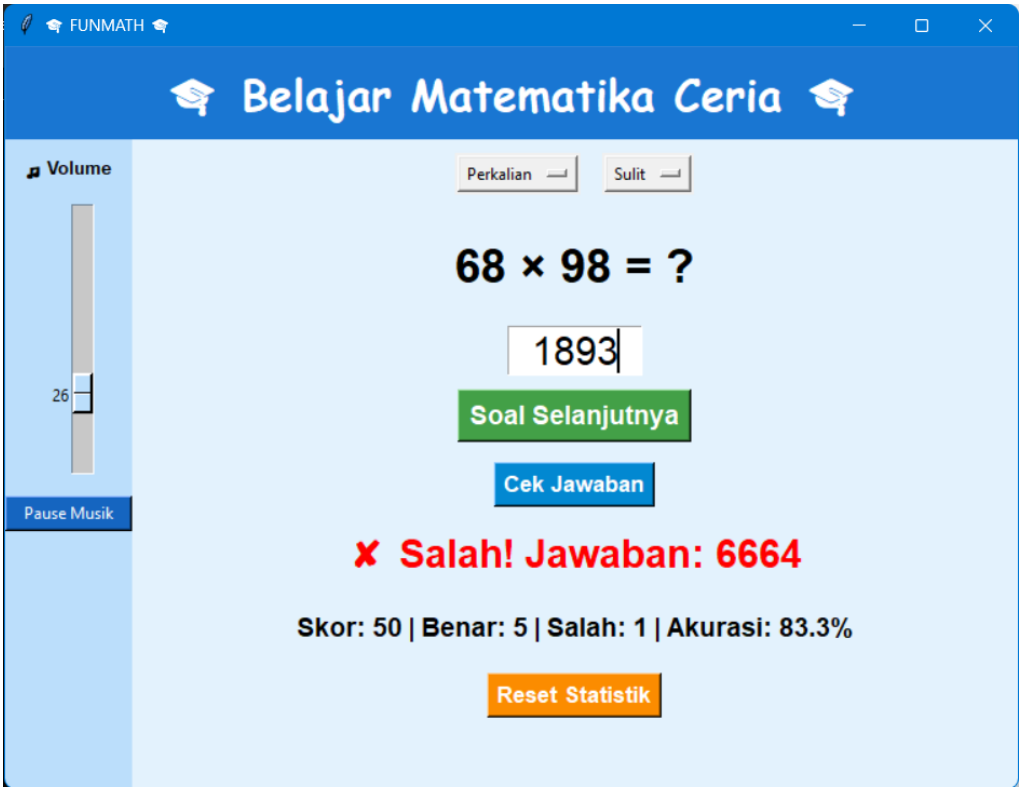
Kesimpulan

Berdasarkan hasil perancangan dan implementasi aplikasi FunMath, dapat disimpulkan bahwa penerapan konsep Object-Oriented Programming (OOP) menggunakan bahasa pemrograman Python dapat dilakukan secara efektif dalam pengembangan aplikasi pembelajaran. Konsep utama OOP seperti encapsulation, inheritance, dan polymorphism telah berhasil diimplementasikan melalui pembagian kelas yang terstruktur dan saling berinteraksi sesuai dengan fungsinya masing-masing.

Aplikasi ini mampu menyediakan media pembelajaran matematika yang interaktif dengan memanfaatkan antarmuka grafis Tkinter serta dukungan audio dari Pygame. Fitur-fitur seperti pemilihan jenis operasi, tingkat kesulitan, umpan balik langsung, serta statistik permainan terbukti dapat meningkatkan keterlibatan pengguna dalam proses belajar. Selain itu, struktur program yang modular memudahkan pengembangan dan pemeliharaan aplikasi di masa mendatang.

LAMPIRAN

Tampilan aplikasi



Link Github: https://github.com/mahayasa78/UAS-PBO-aplikasi-FunMath