LAPORAN PROJECT GAME “MEMORY CARD” MENGGUNAKAN KONSEP OOP BERBASIS PYTHON TKINTER

Oleh:

Kelompok 1

MUHAMMAD KHAIRUL ANAM	(24091397002)
ZARYAN NUGRAHA ISLAH	(24091397010)
LUTVIANA DWI JANNATI	(24091397013)
RANDY PRADANA BINTANG OKWIANDA	(24091397018)

PROGRAM STUDI MANAJEMEN INFORMATIKA
FAKULTAS VOKASI
UNIVERSITAS NEGERI SURABAYA
2025

DAFTAR ISI

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Pemrograman Berorientasi Objek (OOP) merupakan paradigma pemrograman yang fundamental dalam pengembangan perangkat lunak modern. Konsep-konsep seperti encapsulation, inheritance, dan polymorphism memungkinkan pembuatan sistem yang modular, mudah dipelihara, dan reusable. Untuk menguasai konsep-konsep tersebut, diperlukan penerapan langsung dalam proyek nyata.

Memory Card Game dipilih sebagai media pembelajaran karena:

- Permainan ini memiliki logika yang jelas dan cocok untuk menerapkan struktur kelas yang terorganisir.
- Elemen interaktif seperti kartu yang bisa dibalik, timer, dan sistem hint memerlukan pengelolaan state yang baik—sesuai dengan prinsip OOP.
- Aplikasi ini dapat dikembangkan dengan antarmuka grafis menggunakan Tkinter, yang memungkinkan integrasi antara logika permainan dan tampilan visual.

Game ini dirancang dengan tiga tingkat kesulitan (4x4, 4x6, 6x6) dan fitur tambahan seperti pause, hint, dan timer, sehingga tidak hanya berfungsi sebagai hiburan, tetapi juga sebagai latihan kognitif untuk mengasah memori pemain.

1.2 Rumusan Masalah

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

- Bagaimana merancang struktur kelas yang menerapkan prinsip encapsulation untuk menyembunyikan detail implementasi kartu dan logika permainan?
- Bagaimana menerapkan inheritance untuk membuat hierarki kelas konfigurasi level (Easy, Medium, Hard) dan kelas tombol kartu dengan perilaku yang berbeda?
- Bagaimana mengimplementasikan polymorphism sehingga tombol kartu dapat menampilkan perilaku visual yang berbeda saat terbuka, terpasang, atau dalam mode hint?
- Bagaimana membuat antarmuka pengguna yang responsif dan intuitif dengan navigasi antar halaman (Main Menu, Level Selection, Game Board) menggunakan Tkinter?
- Bagaimana mengintegrasikan fitur tambahan seperti timer, pause, dan hint tanpa mengganggu alur logika permainan utama?

1.3 Tujuan

Tujuan dari pengembangan aplikasi Memory Card Game ini adalah:

- Membuat aplikasi game berbasis GUI dengan Python dan Tkinter yang menerapkan prinsip OOP secara sistematis.
- Menerapkan encapsulation dengan menggunakan atribut privat/protected dan metode getter/setter implisit dalam kelas GameConfig, CardButton, dan MemoryGame.

Menerapkan inheritance melalui hierarki:

- EasyConfig, MediumConfig, HardConfig yang mewarisi GameConfig.
- MatchableCardButton yang mewarisi CardButton.
- Menerapkan polymorphism melalui:
- Overriding metode apply_match_style() di kelas MatchableCardButton.
- Penggunaan metode yang sama (show_value(), reset_visual()) dengan perilaku berbeda sesuai konteks.
- Mengimplementasikan fitur tambahan seperti pause, hint terbatas, timer real-time, dan feedback visual untuk meningkatkan pengalaman pengguna.
- Menghasilkan aplikasi yang dapat digunakan sebagai alat edukasi untuk memahami penerapan OOP dalam proyek nyata.

1.4 Manfaat

Manfaat dari pengembangan aplikasi ini adalah:

Bagi Pengembang:

- Pengalaman langsung dalam merancang arsitektur perangkat lunak berbasis OOP.
- Pemahaman mendalam tentang integrasi logika permainan dengan antarmuka grafis.
- Kemampuan dalam mengelola state aplikasi yang kompleks (pause, hint, matching logic).

Bagi Pengguna:

- Hiburan yang melatih daya ingat dan konsentrasi melalui tiga level kesulitan.
- Antarmuka yang intuitif dengan navigasi yang mudah dipahami.
- Fitur hint dan pause yang mendukung pengalaman bermain yang fleksibel.

Bagi Pendidikan:

- Contoh nyata implementasi OOP yang dapat digunakan sebagai bahan ajar atau referensi.
- Dokumentasi lengkap yang mencakup diagram kelas dan penjelasan kode untuk studi lebih lanjut.

Bagi Pengembangan Lanjutan:

- Kode yang modular memungkinkan pengembangan fitur tambahan seperti sistem skor, leaderboard, atau tema visual yang berbeda.
- Struktur yang terorganisir memudahkan pemeliharaan dan pengujian.

BAB II LANDASAN TEORI

2.1 Konsep Pemrograman Berorientasi Objek (OOP)

Pemrograman Berorientasi Objek (OOP) adalah paradigma pemrograman yang mengorganisasi kode ke dalam objek yang terdiri dari atribut (data) dan metode (perilaku). OOP memungkinkan pengembangan perangkat lunak yang lebih terstruktur, modular, dan mudah dipelihara. Tiga prinsip utama OOP yang diterapkan dalam aplikasi ini adalah:

1. **Encapsulation (Enkapsulasi)** Enkapsulasi adalah prinsip menyembunyikan detail implementasi suatu objek dan hanya menyediakan antarmuka yang aman untuk berinteraksi. Dalam Python, enkapsulasi diterapkan melalui:
    - Atribut privat: menggunakan konvensi _nama_atribut atau __nama_atribut.
    - Metode publik: menyediakan akses terkontrol ke data internal.

    Contoh dalam proyek: Kelas GameConfig menyembunyikan detail pembangkitan nilai kartu melalui metode generate_card_values(). Kelas MemoryGame mengelola state permainan (seperti opened_cards, matched_indices) secara internal tanpa diekspos langsung ke pengguna.

2. **Inheritance (Pewarisan)** Pewarisan memungkinkan kelas baru (child class) mewarisi atribut dan metode dari kelas lain (parent class). Hal ini mendukung *code reuse* dan hierarki logis.

    Contoh dalam proyek:

    - EasyConfig, MediumConfig, HardConfig mewarisi dari GameConfig.
    - MatchableCardButton mewarisi dari CardButton.

3. **Polymorphism (Polimorfisme)** Polimorfisme memungkinkan objek dari kelas yang berbeda untuk merespons metode yang sama dengan cara yang berbeda. Terdapat dua jenis polimorfisme:

    - Method Overriding: kelas anak mengubah implementasi metode dari kelas induk.
    - Method Overloading (tidak didukung langsung di Python, dapat disimulasikan dengan argumen opsional).

    Contoh dalam proyek: Metode apply_match_style() di MatchableCardButton meng-override metode dengan nama yang sama di CardButton untuk menambahkan styling khusus.

2.2 Python dan Tkinter

1. **Python** Python adalah bahasa pemrograman tingkat tinggi yang mendukung multi-paradigma, termasuk OOP. Kelebihan Python:
    - Sintaksis yang mudah dipahami.
    - Dukungan library yang luas.
    - Cocok untuk pengembangan aplikasi desktop berbasis GUI.
2. **Tkinter** Tkinter adalah library standar Python untuk membuat antarmuka grafis (GUI). Komponen utama:
    - Widget: Button, Label, Frame, dll.
    - Geometry Management: pack(), grid(), place().
    - Event Handling: sistem callback untuk menangani interaksi pengguna.

Penerapan dalam proyek: Seluruh antarmuka game dibangun dengan Tkinter, termasuk:

- Frame navigasi (MainMenu, LevelSelection, MemoryGame).
- Tombol kartu yang merupakan turunan dari ttk.Button.
- Pengaturan layout dengan grid() untuk papan kartu.

2.3 Struktur Data dan Algoritma Pendukung

1. **List dan List 2D** Digunakan untuk menyimpan:
   - Nilai kartu (current_cards).
   - Referensi tombol kartu (buttons sebagai list 2D).
   - Kartu yang sedang terbuka (opened_cards).
2. **Random Shuffle** Modul random digunakan untuk mengacak urutan kartu:

   random.shuffle(cards)

   Memastikan permainan selalu memiliki konfigurasi kartu yang berbeda setiap sesi.

3. **Time Management** Modul time digunakan untuk:
   - Menghitung durasi permainan (time.time()).
   - Mengatur jeda tampilan hint dan pengecekan pasangan kartu.

2.4 Arsitektur Aplikasi Berbasis Frame (Multi-Page)

Aplikasi ini menerapkan pola *multi-frame navigation*, di mana setiap halaman (Main Menu, Level Selection, Game) direpresentasikan sebagai kelas tk.Frame yang berbeda. Kelas utama AplikasiUtama berfungsi sebagai *controller* yang mengelola perpindahan antar frame dengan metode show_frame() dan start_game().

2.5 Game Mechanics dan Cognitive Benefits

Memory Card Game termasuk kategori *game pelatihan kognitif* yang dapat:

- Meningkatkan memori jangka pendek.
- Melatih konsentrasi dan perhatian visual.
- Memberikan tantangan progresif melalui level kesulitan.

Fitur seperti *hint terbatas* dan *timer* menambah elemen strategi dan manajemen waktu dalam permainan.

BAB III ANALISIS DAN PERANCANGAN

3.1 Analisis Kebutuhan

Aplikasi Memory Card Game dirancang dengan kebutuhan fungsional sebagai berikut:

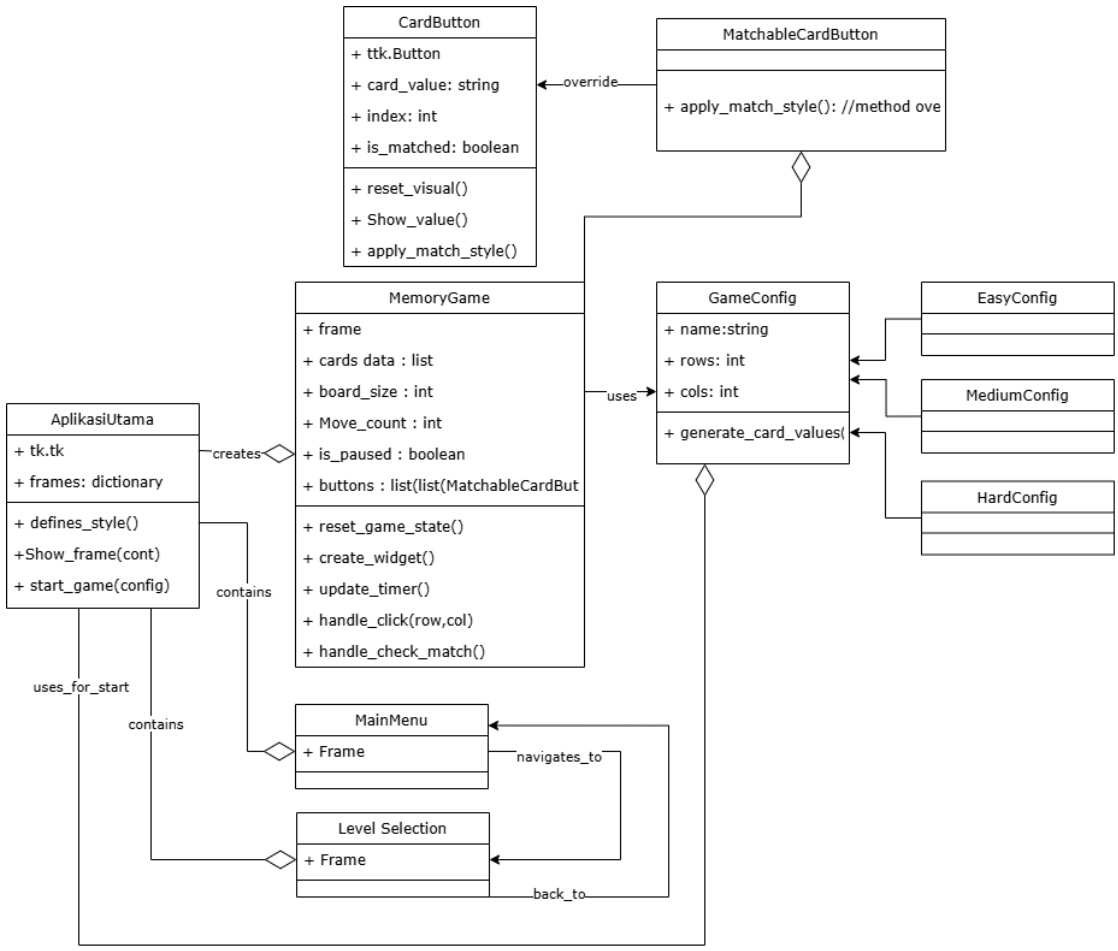
A. Navigasi Antar Halaman:
   - Main Menu → Level Selection → Game Board → Kembali ke Main Menu/Level Selection.

B. Manajemen Level:
   - Tiga level kesulitan dengan ukuran grid berbeda (4×4, 4×6, 6×6).
   - Pembangkitan kartu acak yang berbeda tiap sesi.

C. Logika Game:
   - Pemain membalik dua kartu per langkah.
   - Kartu cocok → tetap terbuka.
   - Kartu tidak cocok → tertutup kembali setelah 1 detik.
   - Game berakhir saat semua pasangan ditemukan.

D. Fitur Tambahan:
   - Timer untuk mencatat durasi permainan.
   - Tombol pause/lanjut.
   - Fitur hint (max 2 kali per game).
   - Reset game ke pemilihan level.

3.2 Class Diagram



PENJELASAN CLASS DIAGRAM MEMORY CARD GAME

1. Kelas AplikasiUtama (Kelas Utama Aplikasi)

Kelas ini merupakan program utama yang mengatur seluruh aplikasi, mewarisi dari Tk.Tk.

Peran: Jendela utama dan pengendali navigasi aplikasi

Atribut: frames (Dictionary): Menyimpan semua halaman aplikasi (MainMenu, LevelSelection, MemoryGame)

Metode Utama:

- define_styles(): Mengatur tampilan visual untuk semua komponen
- show_frame(cont): Menampilkan halaman tertentu yang disimpan dalam frames
- start_game(config): Memulai permainan baru dengan konfigurasi level yang dipilih

Penerapan Komposisi: AplikasiUtama memiliki hubungan komposisi dengan MainMenu, LevelSelection, dan MemoryGame

2. Kelas MemoryGame (Kelas Utama Permainan)

Kelas ini merepresentasikan halaman permainan utama, mewarisi dari Frame.

Peran: Frame yang berisi logika permainan dan antarmuka game

Atribut:

- cards_data (List): Daftar nilai kartu yang sudah diacak
- board_size (Int): Ukuran papan permainan (berasal dari GameConfig)
- move_count (Int): Jumlah langkah yang telah dilakukan pemain
- is_paused (Boolean): Status apakah permainan sedang dijeda
- buttons (List[List[MatchableCardButton]]): Grid 2D tombol kartu

Metode Utama:

- reset_game_state(): Mengatur ulang semua variabel status untuk permainan baru
- create_widgets(): Membuat dan menata semua elemen UI game
- update_timer(): Memperbarui waktu permainan setiap detik
- handle_click(row, col): Menangani logika saat pemain mengklik sebuah kartu
- handle_check_match(): Mengecek apakah dua kartu yang terbuka cocok

Penerapan Asosiasi: MemoryGame menggunakan GameConfig untuk konfigurasi

Penerapan Komposisi: MemoryGame memiliki banyak MatchableCardButton

3. Kelas CardButton (Kelas Dasar Kartu)

Kelas ini merepresentasikan tombol kartu di antarmuka pengguna, mewarisi dari ttk.Button.

Peran: Representasi tombol kartu di UI

Atribut:

- card_value (String): Nilai kartu yang tersembunyi (misal 'A', 'B')
- index (Int): Posisi unik kartu dalam grid
- is_matched (Boolean): Status apakah kartu sudah menemukan pasangannya

Metode Utama:

- reset_visual(): Menyembunyikan nilai kartu, mengembalikan ke state tertutup
- show_value(): Menampilkan nilai kartu saat dibalik
- apply_match_style(): Mengubah tampilan kartu menjadi "terpasang" (matched)

Penerapan Encapsulation: Atribut seperti card_value dan is_matched dikelola internal oleh objek kartu

4. Kelas MatchableCardButton (Kelas Turunan Kartu)

Kelas ini merepresentasikan kartu dengan tampilan khusus saat cocok, mewarisi dari CardButton.

Peran: Kartu dengan perilaku visual khusus saat menemukan pasangan

Metode Utama:

apply_match_style(): Method Overriding - Menimpa metode dari parent class untuk menambahkan styling kustom

Penerapan Inheritance: Mewarisi semua atribut dan metode dari CardButton

Penerapan Polymorphism: Metode apply_match_style() memiliki implementasi berbeda dari parent class

5. Kelas GameConfig (Kelas Induk Konfigurasi)

Kelas ini merupakan kelas dasar untuk menyimpan konfigurasi ukuran papan.

Peran: Template untuk konfigurasi level permainan

Atribut:

- name (String): Nama level (contoh: "Level 1 (4x4)")
- rows (Int): Jumlah baris grid
- cols (Int): Jumlah kolom grid

Metode Utama:

generate_card_values(): Menghasilkan list nilai kartu yang sudah dikocok

Penerapan Inheritance: Diiwarisi oleh EasyConfig, MediumConfig, dan HardConfig

6. Kelas EasyConfig (Konfigurasi Level Mudah)
Kelas ini merepresentasikan konfigurasi untuk level mudah.
Peran: Konfigurasi untuk Level 1 (4x4)
Penerapan Inheritance: Mewarisi semua atribut dan metode dari GameConfig
Spesifikasi: Grid 4×4 dengan 16 kartu

7. Kelas MediumConfig (Konfigurasi Level Sedang)
Kelas ini merepresentasikan konfigurasi untuk level sedang.
Peran: Konfigurasi untuk Level 2 (4x6)
Penerapan Inheritance: Mewarisi semua atribut dan metode dari GameConfig
Spesifikasi: Grid 4×6 dengan 24 kartu

8. Kelas HardConfig (Konfigurasi Level Sulit)
Kelas ini merepresentasikan konfigurasi untuk level sulit.
Peran: Konfigurasi untuk Level 3 (6x6)
Penerapan Inheritance: Mewarisi semua atribut dan metode dari GameConfig
Spesifikasi: Grid 6×6 dengan 36 kartu

9. Kelas MainMenu (Kelas Menu Utama)
Kelas ini merepresentasikan halaman menu utama aplikasi.
Peran: Frame yang menampilkan tombol Play dan Quit
Penerapan Komposisi: Dimiliki oleh AplikasiUtama
Penerapan Asosiasi: Berasosiasi dengan LevelSelection untuk navigasi

10. Kelas LevelSelection (Kelas Pemilihan Level)
Kelas ini merepresentasikan halaman pemilihan tingkat kesulitan.
Peran: Frame yang menampilkan pilihan level (Level 1, 2, 3)

Penerapan Komposisi: Dimiliki oleh AplikasiUtama
Penerapan Asosiasi: Berasosiasi dengan MainMenu untuk navigasi kembali
Penerapan Komposisi: Menggunakan GameConfig untuk memulai permainan

HUBUNGAN ANTAR KELAS
1. Pewarisan (Inheritance)

- CardButton <|-- MatchableCardButton : override
- GameConfig <|-- EasyConfig
- GameConfig <|-- MediumConfig
- GameConfig <|-- HardConfig

Artinya: Kelas di kanan mewarisi semua atribut dan metode dari kelas di kiri

2. Komposisi (Composition)

- AplikasiUtama "1" --o "1" MemoryGame : creates
- MemoryGame "1" --o "*" MatchableCardButton
- AplikasiUtama "1" --o "1" MainMenu : contains
- AplikasiUtama "1" --o "1" LevelSelection : contains
- AplikasiUtama "1" --o "1" GameConfig : uses_for_start

Artinya: Kelas di kiri memiliki kelas di kanan sebagai bagian integral

3. Asosiasi (Association)

- MemoryGame --> GameConfig : uses
- MainMenu --> LevelSelection : navigates_to
- LevelSelection --> MainMenu : back_to

Artinya: Kelas di kiri menggunakan kelas di kanan untuk berkolaborasi

PENERAPAN PRINSIP OOP

1. Encapsulation (Enkapsulasi)
Setiap kelas menyembunyikan detail implementasi internal
Contoh: card_value di CardButton hanya diakses melalui show_value() dan reset_visual()

2. Inheritance (Pewarisan)
MatchableCardButton mewarisi CardButton
EasyConfig, MediumConfig, HardConfig mewarisi GameConfig
Memungkinkan code reuse dan hierarki yang terorganisir

3. Polymorphism (Polimorfisme)
Method overriding pada apply_match_style() di MatchableCardButton
Interface yang sama (generate_card_values()) dengan implementasi berbeda di setiap level

4. Composition (Komposisi)
MemoryGame terdiri dari banyak MatchableCardButton
AplikasiUtama terdiri dari MainMenu, LevelSelection, dan MemoryGame
Hubungan bagian-seluruh yang kuat

BAB IV IMPLEMENTASI DAN PENGGUNAAN SISTEM

4.1 Implementasi Code

1. KELAS GameConfig, EasyConfig, MediumConfig, HardConfig

GameConfig (Kelas Induk)

```
# --- A. KELAS KONFIGURASI LEVEL ---
class GameConfig:
    """Kelas Induk (Polymorphic) untuk menyimpan konfigurasi ukuran papan."""
    def __init__(self, name, rows, cols):
        self.name = name # Nama level (e.g., "Level 1 (4x4)")
        self.rows = rows # Jumlah baris grid
        self.cols = cols # Jumlah kolom grid
        self.total_cards = rows * cols # Total kartu di papan

    def generate_card_values(self):
        """Menghasilkan list nilai kartu (misal ['A', 'A', 'B', 'B', ...])
yang sudah dikocok."""
        num_pairs = self.total_cards // 2

        if self.total_cards % 2 != 0:
            raise ValueError(f"Level {self.name} memiliki jumlah kartu ganjil
({self.total_cards}). Permainan pasangan harus genap.")

        if num_pairs > 26:
            raise ValueError("Grid terlalu besar, melebihi 26 pasangan.")

        values = [chr(65 + i) for i in range(num_pairs)]
        cards = values * 2
        random.shuffle(cards)
        return cards
```

Fungsi: Menyimpan konfigurasi ukuran papan (baris dan kolom)

generate_card_values(): Membuat pasangan huruf (A, B, C...) sesuai jumlah kartu, lalu mengacaknya

Pewarisan: EasyConfig, MediumConfig, HardConfig mewarisi dari GameConfig

EasyConfig, MediumConfig, HardConfig

```
class EasyConfig(GameConfig):
    """Konfigurasi untuk Level 1 (4x4)."""
    def __init__(self):
        super().__init__("Level 1 (4x4)", 4, 4)

class MediumConfig(GameConfig):
    """Konfigurasi untuk Level 2 (4x6)."""
    def __init__(self):
        super().__init__("Level 2 (4x6)", 4, 6)
```

```
class HardConfig(GameConfig):
    """Konfigurasi untuk Level 3 (6x6)."""
    def __init__(self):
        super().__init__("Level 3 (6x6)", 6, 6)
```

Inheritance: Mewarisi semua dari GameConfig

Spesifikasi: Masing-masing menentukan ukuran grid berbeda untuk 3 level kesulitan

2. KELAS CardButton dan MatchableCardButton

CardButton (Kelas Dasar Kartu)

```
# --- B. KELAS TOMBOL KARTU ---
class CardButton(ttk.Button):
    """Kelas dasar untuk tombol kartu."""
    def __init__(self, master, card_value, index, **kwargs):
        super().__init__(master, **kwargs)
        self.card_value = card_value # Nilai kartu yang tersembunyi (misal
'A')
        self.index = index # Indeks posisi kartu di list data
        self.is_matched = False # Status kecocokan kartu

    def reset_visual(self):
        """Menyembunyikan nilai kartu dan mengembalikan ke state normal
(tertutup)."""
        self.config(text=" ", state=tk.NORMAL, style='Card.TButton')

    def show_value(self):
        """Menampilkan nilai kartu saat dibalik."""
        self.config(text=self.card_value, style='Opened.TButton')

    def apply_match_style(self):
        """Menerapkan style dasar saat kartu cocok (Metode Induk)."""
        self.is_matched = True
        self.config(style='Matched.TButton', state=tk.DISABLED)
```

Peran: Tombol yang mewakili satu kartu di UI

Atribut: card_value (nilai kartu), index (posisi), is_matched (status cocok)

Metode: show_value() (buka kartu), reset_visual() (tutup kartu), apply_match_style() (tampilan saat cocok)

MatchableCardButton (Kartu dengan Overriding)

```
class MatchableCardButton(CardButton):
    """Kelas Kartu yang menerapkan styling unik saat cocok (Method
Overriding)."""
    def apply_match_style(self):
        """Menimpa metode induk untuk menambahkan style foreground kustom."""
        super().apply_match_style()
        self.config(style='CustomMatchText.TButton')
```

Inheritance: Mewarisi semua dari CardButton

Polymorphism: Meng-override apply_match_style() untuk tambah style khusus

Method Overriding: Memanggil parent method dulu, lalu tambah style kustom

3. KELAS MemoryGame (Logika Utama Game)

Inisialisasi dan State Management

```
# --- C. KELAS GAME LOGIC DAN TAMPILAN (FRAME) ---
class MemoryGame(tk.Frame):
    """Frame utama yang berisi logika permainan dan tata letak grid."""
    def __init__(self, master, app, initial_config):
        super().__init__(master, bg='white')
        self.app = app # Referensi ke root window (AplikasiUtama)

        self.style = ttk.Style() # Style TTK untuk widget

        self.config = initial_config # Konfigurasi level yang dipilih
        self.timer_id = None # ID untuk mengontrol fungsi after() timer

        self.reset_game_state()

        self.create_widgets()
        self.start_game()

    def reset_game_state(self):
        """Mempersiapkan ulang semua variabel status untuk permainan baru."""
        self.current_cards = self.config.generate_card_values() # List nilai
kartu yang dikocok
        self.buttons = [] # List 2D objek tombol kartu
        self.opened_cards = [] # List 1-2 kartu yang sedang terbuka saat ini
        self.matched_indices = [] # Indeks kartu yang sudah cocok
        self.move_count = 0 # Jumlah langkah (pasangan balik)
        self.can_click = True # Bendera kontrol klik kartu
        self.is_paused = False # Status jeda permainan
        self.hint_count = 2 # Jumlah hint yang tersedia
        self.hint_in_progress = False # Status hint sedang berjalan
```

Fungsi: Frame utama yang berisi seluruh logika permainan

State: Mengelola state game (kartu terbuka, jumlah langkah, status pause, hint)

Pembuatan UI dan Kartu

```
    def create_widgets(self):
        """Membuat dan menata semua elemen UI game (grid kartu, label info,
kontrol)."""
        for widget in self.winfo_children():
            widget.destroy()

        board_frame = tk.Frame(self, bg='white')
        board_frame.pack(pady=10)

        self.buttons = []
        for row in range(self.config.rows):
            button_row = []
            for col in range(self.config.cols):
                card_index = row * self.config.cols + col
                card_value = self.current_cards[card_index]

                button = MatchableCardButton(
                    board_frame,
                    card_value=card_value,
                    index=card_index,
                    style='Card.TButton',
                    command=lambda r=row, c=col: self.handle_click(r, c)
                )

                button.grid(row=row, column=col, padx=5, pady=5)
                button_row.append(button)
            self.buttons.append(button_row)
```

Grid Kartu: Membuat grid MatchableCardButton sesuai konfigurasi level
Event Handler: Setiap kartu memiliki command yang memanggil handle_click()

Logika Click dan Pencocokan Kartu

```
    def handle_click(self, row, col):
        """Menangani logika saat pemain mengklik sebuah kartu."""
        if not self.can_click or self.is_paused:
            return

        button = self.buttons[row][col]

        if button in self.opened_cards or button.is_matched:
            return

        button.show_value()
        button.config(state=tk.DISABLED)
        self.opened_cards.append(button)

        if len(self.opened_cards) == 2:
```

```
            self.can_click = False
            self.lock_all_cards()
            self.move_count += 1
            self.level_label.config(text=f"{self.config.name} | Langkah:
{self.move_count}")

            self.app.after(1000, self.handle_check_match)

    def handle_check_match(self):
        """Mengecek apakah dua kartu yang terbuka cocok atau tidak."""
        if len(self.opened_cards) < 2:
            self.unlock_all_cards()
            self.can_click = True
            return

        card1 = self.opened_cards[0]
        card2 = self.opened_cards[1]

        if card1.card_value == card2.card_value:
            self.status_label.config(text="🎉 Cocok!", fg='green')

            card1.apply_match_style() # Delegasi styling ke objek kartu
            card2.apply_match_style()

            self.matched_indices.append(card1.index)
            self.matched_indices.append(card2.index)

        else:
            self.status_label.config(text="❌ Tidak Cocok. Coba lagi.",
fg='red')

            for button in self.opened_cards:
                button.reset_visual() # Menyembunyikan kartu

        self.opened_cards = []
        self.can_click = True
```

Logika: Maksimal 2 kartu terbuka, cek kecocokan setelah 1 detik

Polymorphism: apply_match_style() dipanggil - jika MatchableCardButton, gunakan versi override

State Update: Update matched_indices jika cocok, reset kartu jika tidak cocok

4. KELAS AplikasiUtama (Controller Utama)

Manajemen Frame dan Navigasi

```
class AplikasiUtama(tk.Tk):
    """Kelas root window utama yang mengatur style dan navigasi antar
frame."""
    def __init__(self):
        super().__init__()
        self.title("🎮 Memory Card Game (OOP Tkinter)")
        self.configure(bg='white')

        # Mendaftarkan style TTK agar tersedia di semua frame sejak awal
        self.define_styles()

        # Frame Container: Menampung semua frame (MainMenu, LevelSelection,
MemoryGame)
        self.container = tk.Frame(self, bg='white')
        self.container.pack(side="top", fill="both", expand=True)

        self.frames = {} # Dictionary untuk menyimpan referensi frame

        # Inisialisasi MainMenu dan LevelSelection
        for F in (MainMenu, LevelSelection):
            frame = F(self.container, self)
            self.frames[F] = frame
            frame.grid(row=0, column=0, sticky="nsew") # Menggunakan grid
untuk layering

        self.show_frame(MainMenu) # Tampilkan Main Menu saat pertama kali
dibuka
    def show_frame(self, cont):
        """Menampilkan frame tertentu (halaman) di atas container."""
        frame = self.frames[cont]
        frame.tkraise()
        self.config(menu=tk.Menu(self))

    def start_game(self, config):
        """Membuat instance MemoryGame baru dan memindahkannya ke tampilan
game."""
        # Hapus frame game lama jika ada
        if MemoryGame in self.frames:
            self.frames[MemoryGame].destroy()
            del self.frames[MemoryGame]

        # Buat frame game baru dengan konfigurasi level yang dipilih
        game_frame = MemoryGame(self.container, self, initial_config=config)
        self.frames[MemoryGame] = game_frame
        game_frame.grid(row=0, column=0, sticky="nsew")
        game_frame.tkraise()
```

Fungsi: Window utama yang mengatur navigasi antar halaman
Frame Management: Menyimpan semua frame dalam dictionary frames
Navigasi: show_frame() untuk ganti halaman, start_game() untuk mulai permainan baru

5. KELAS MainMenu dan LevelSelection

MainMenu (Halaman Utama)

```
class MainMenu(tk.Frame):
    """Frame yang menampilkan tombol Play dan Quit."""
    def __init__(self, master, app):
        super().__init__(master, bg='white')
        self.app = app # Referensi ke AplikasiUtama

        tk.Label(self, text="🎮 MEMORY CARD GAME 🃏", font=('Arial', 24,
'bold'), bg='white').pack(pady=40)

        ttk.Button(
            self,
            text="▶ PLAY",
            command=lambda: self.app.show_frame(LevelSelection),
            style='Menu.TButton'
        ).pack(pady=15, ipadx=20)

        ttk.Button(
            self,
            text="❌ QUIT",
            command=self.app.quit,
            style='Reset.TButton'
        ).pack(pady=15, ipadx=20)
```

Peran: Halaman awal dengan tombol PLAY dan QUIT
Navigasi: PLAY → LevelSelection

LevelSelection (Pilih Level)

```
# --- F. KELAS PEMILIHAN LEVEL ---
class LevelSelection(tk.Frame):
    """Frame yang menampilkan pilihan level (Level 1, 2, 3)."""
    def __init__(self, master, app):
        super().__init__(master, bg='white')
        self.app = app

        tk.Label(self, text="PILIH LEVEL KESULITAN", font=('Arial', 20,
'bold'), bg='white').pack(pady=30)

        levels = [EasyConfig(), MediumConfig(), HardConfig()]

        for level in levels:
```

```
            ttk.Button(
                self,
                text=level.name,
                command=lambda l=level: self.app.start_game(l),
                style='Menu.TButton'
            ).pack(pady=10, ipadx=10, fill='x', padx=50)

        ttk.Button(
            self,
            text="◀ BACK",
            command=lambda: self.app.show_frame(MainMenu),
            style='Reset.TButton'
        ).pack(pady=20, ipadx=20)
```

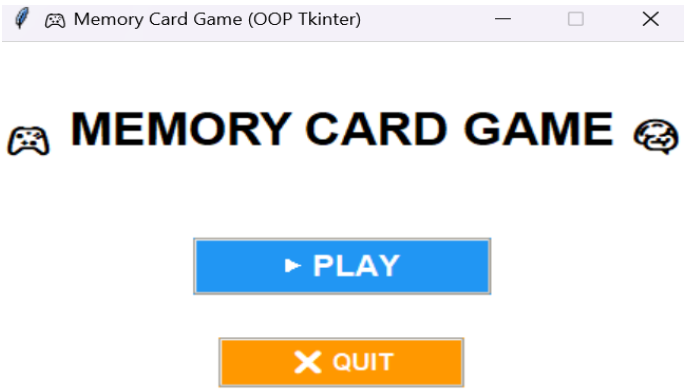
Peran: Halaman untuk memilih tingkat kesulitan

Fungsi: Membuat instance GameConfig sesuai pilihan, mulai game dengan start_game()

4.2 Fitur Game Memory Card

Berikut adalah fitur-fitur yang dimiliki oleh program Memory Card Game, diantaranya:

TAMPILAN AWAL



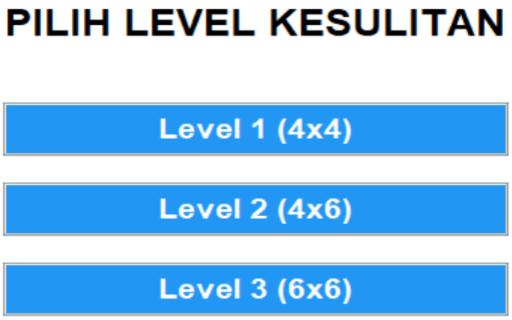
1. **Fitur play**

Pada Fitur Pertama ini berfungsi untuk memulai memainkan game memory card

2. **Fitur Quit**

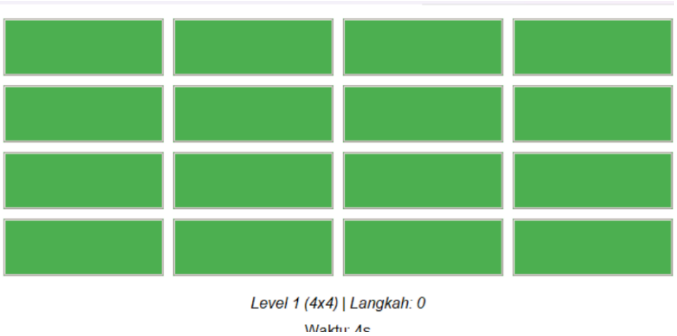
Pada Fitur Ini berfungsi untuk mengakhiri dan log out dari game tersebut

3. **Fitur Level**



Diatas adalah tampilan fitur level dari 1-3 dengan jumlah card yang berbeda beda tinggal menyesuaikan user mau memilih level berapa

4. **Fitur Level 1(4x4)**

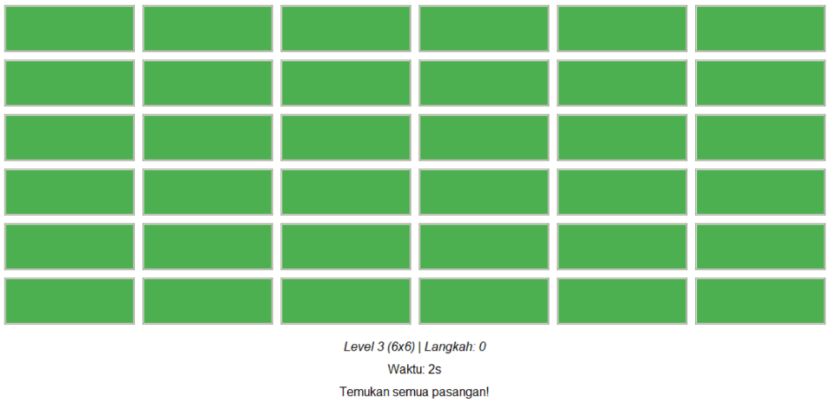


Diatas Tampilan fitur ketika user memilih level 1 yang sesuai ketentuan dimana jumlah card yang uda sesuai yaitu panjang berjumlah 4 card dan lebar berjumlah 4 card, disitu dibawah terdapat keterangan langkah brp saja ketika membuka kartu dan seberapa lama waktu yang user dapatkan ketika berhasil mencocokkan semua kartu

5. **Fitur Level 2 (4x6)**

Sama Seperti level sebelumnya bedanya pada fitur ini hanya dijumlah card yang dimana panjang berjumlah 4 card dan lebar berjumlah 6 card sehingga membuat user merasakan tingkatan tantangan pada level ini

6. **Fitur Level 3 (6x6)**



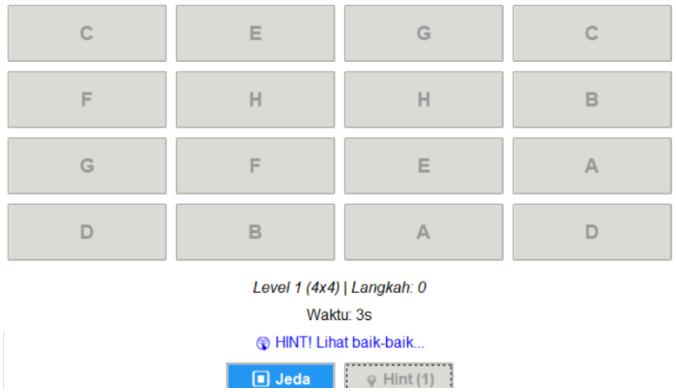
Sama juga seperti level sebelum"nya beda disini card berjumlah lebih banyak yaitu panjang 6 card dan lebar 6 card pada fitur tingkatan ini membuat user merasakan hard game yang sulit saat mencocokkan kartu yang begitu banyak

7. **Fitur Jeda dan Hint (bantuan)**

- Yang pertama fitur jeda dimana fitur ini berfungsi untuk menjeda permainan ketika user ada kesibukan lain atau ingin berhenti sejenak dr game tanpa harus memulai dari awal disini waktu akan berhenti dan ketika akan melanjutkan game tinggal pencet fitur lanjut setelah game dijeda
- Kedua yaitu fitur Hint yang dimana ketika user kesulitan/bingung dalam mencocokkan maka bisa menggunakan fitur tersebut sehingga lebih gampang dan cepat dalam menemukan card nya

Tampilan fitur lanjut

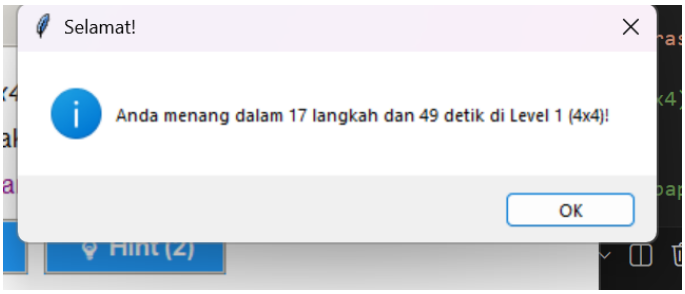
Tampilan Fitur Hint



8. **Fitur Reset Game**

Ini Adalah Fitur reset yang dimana ketika user mengklik ikon tersebut maka seluruh permainan akan direset dari waktu juga card yang sudah dicocokkan dan akan kembali ke laman pilih level

9. **Fitur Ketika Game Berhasil**



Fitur Terakhir ini berfungsi menampilkan Hasil dari game yang telah dimainkan yang dimana user dapat mengetahui hasilnya dan ketika akan kembali bisa mengklik fitur OK

4.3 Cara penggunaan Aplikasi

Cara Penggunaan Aplikasi Memory Card Game

Bagian ini menjelaskan langkah-langkah penggunaan aplikasi **Memory Card Game** yang dibangun menggunakan Python dan Tkinter. Aplikasi ini dirancang agar mudah digunakan oleh pengguna dari berbagai kalangan, termasuk pemula.

1. Membuka Aplikasi

Pengguna menjalankan aplikasi dengan membuka file program menggunakan Python.
Setelah dijalankan, aplikasi akan menampilkan **Halaman Menu Utama**.

2. Menu Utama

Saat aplikasi pertama kali dijalankan, pengguna akan diarahkan ke **halaman Menu Utama**. Halaman ini berfungsi sebagai titik awal interaksi pengguna dengan aplikasi sebelum memasuki permainan.

Pada menu utama terdapat dua tombol:

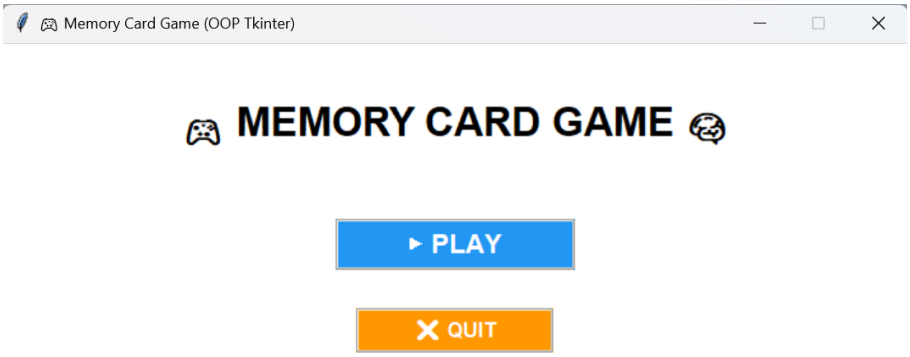
1. **▶ PLAY**

Tombol ini digunakan untuk memulai permainan. Ketika tombol PLAY ditekan, pengguna akan diarahkan ke halaman pemilihan level, di mana pengguna dapat menentukan tingkat kesulitan permainan yang akan dimainkan

2. ✕ **QUIT**

   Tombol ini digunakan untuk menutup aplikasi dan mengakhiri sesi permainan. Setelah tombol QUIT ditekan, aplikasi akan tertutup secara otomatis.

Untuk memulai permainan, pengguna cukup memilih tombol **PLAY**, sehingga aplikasi akan melanjutkan ke tahap selanjutnya tanpa memerlukan pengaturan tambahan.



3. **Memilih Level Permainan**

   Pada halaman **Level Selection**, pengguna diberikan beberapa pilihan tingkat kesulitan permainan yang dapat disesuaikan dengan kemampuan pengguna. Setiap level memiliki ukuran papan kartu yang berbeda, sehingga memengaruhi tingkat kesulitan permainan.

   Di halaman **Level Selection**, Adapun pilihan level yang tersedia adalah sebagai berikut:

- **Level 1 (4x4)** → mudah

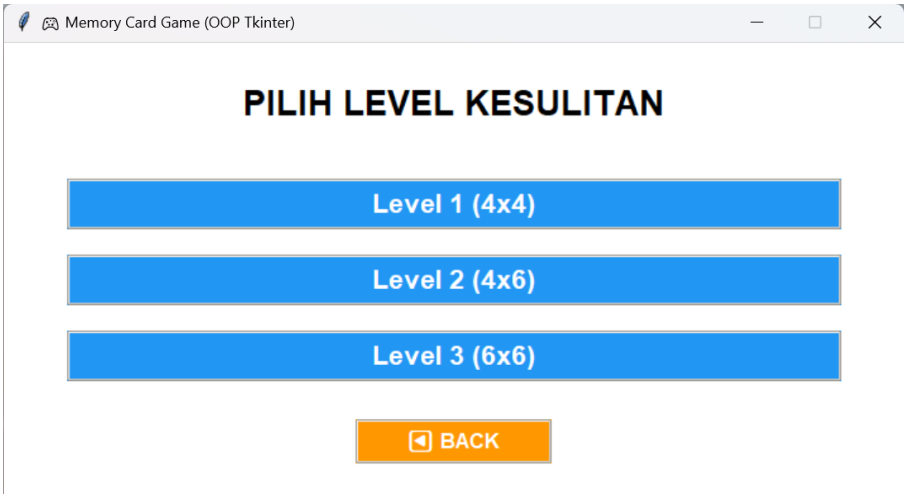
  Merupakan level dengan tingkat kesulitan paling rendah. Jumlah kartu yang lebih sedikit membuat pemain lebih mudah mengingat posisi kartu, sehingga level ini cocok untuk pemula.

- **Level 2 (4x6)** → menengah

  Merupakan level dengan tingkat kesulitan menengah. Jumlah kartu lebih banyak dibandingkan level sebelumnya, sehingga pemain membutuhkan konsentrasi dan daya ingat yang lebih baik.

- **Level 3 (6x6)** → sulit

  Merupakan level dengan tingkat kesulitan tinggi. Ukuran papan yang besar menyebabkan jumlah kartu meningkat secara signifikan, sehingga pemain dituntut untuk memiliki strategi dan daya ingat yang lebih kuat.



Setelah pengguna memilih salah satu level, sistem akan secara otomatis mengatur papan permainan sesuai dengan level yang dipilih dan menampilkan halaman permainan tanpa perlu melakukan pengaturan tambahan.

4. **Tampilan Permainan**

Setelah level dipilih, aplikasi akan menampilkan **halaman permainan utama**. Pada halaman ini, pengguna dapat melihat seluruh elemen antarmuka yang digunakan selama permainan berlangsung. Tampilan permainan dirancang agar mudah dipahami dan memberikan informasi yang jelas kepada pengguna. Dalam halaman permainan, pengguna akan melihat:

- **Grid kartu** (ukuran sesuai level)

  Grid kartu merupakan area utama permainan yang berisi kartu-kartu tertutup. Jumlah dan susunan kartu menyesuaikan dengan level kesulitan yang dipilih oleh pengguna. Setiap kartu dapat diklik untuk menampilkan isi kartu dan mencari pasangan yang sama.

- **Informasi langkah (move count)**

  Informasi langkah menunjukkan jumlah percobaan yang telah dilakukan oleh pemain dalam membuka pasangan kartu. Setiap dua kartu yang dibuka dihitung sebagai satu langkah. Informasi ini membantu pengguna mengetahui seberapa efisien permainan yang dilakukan.

- **Timer waktu**

  Timer berfungsi untuk menampilkan durasi waktu permainan sejak permainan dimulai. Waktu akan terus berjalan selama permainan berlangsung dan akan berhenti ketika permainan selesai atau dijeda. Informasi ini digunakan untuk mengukur kecepatan pemain dalam menyelesaikan permainan.

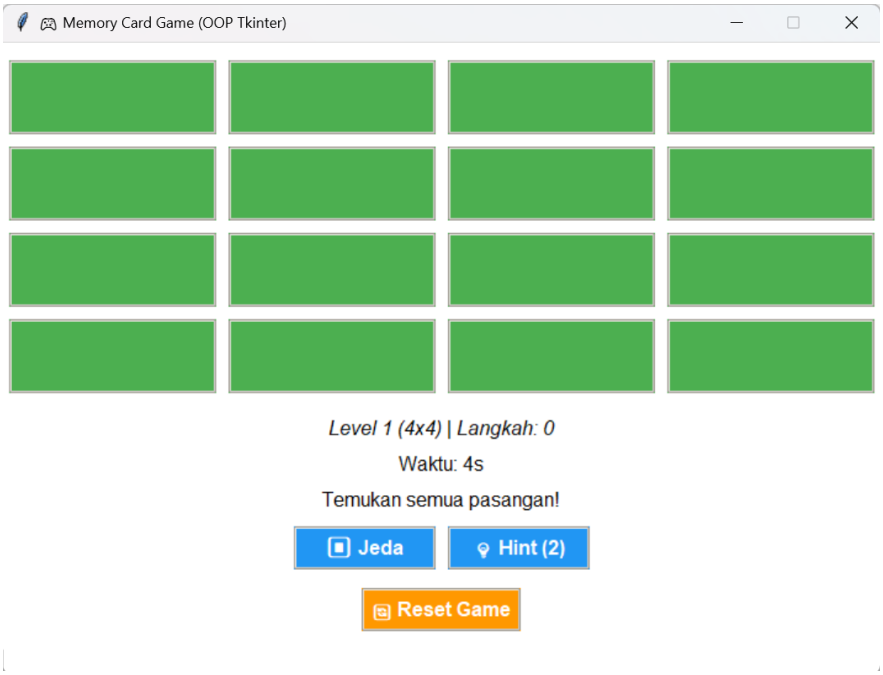
- **Status permainan** (informasi cocok/tidak cocok)

  Status permainan menampilkan pesan informasi kepada pengguna, seperti pemberitahuan ketika kartu cocok, tidak cocok, permainan dijeda, atau ketika seluruh pasangan kartu telah ditemukan. Status ini membantu pengguna memahami kondisi permainan secara real-time.

- **Tombol kontrol:**

  Pada bagian bawah tampilan permainan terdapat beberapa tombol kontrol yang digunakan untuk mengatur jalannya permainan, yaitu:

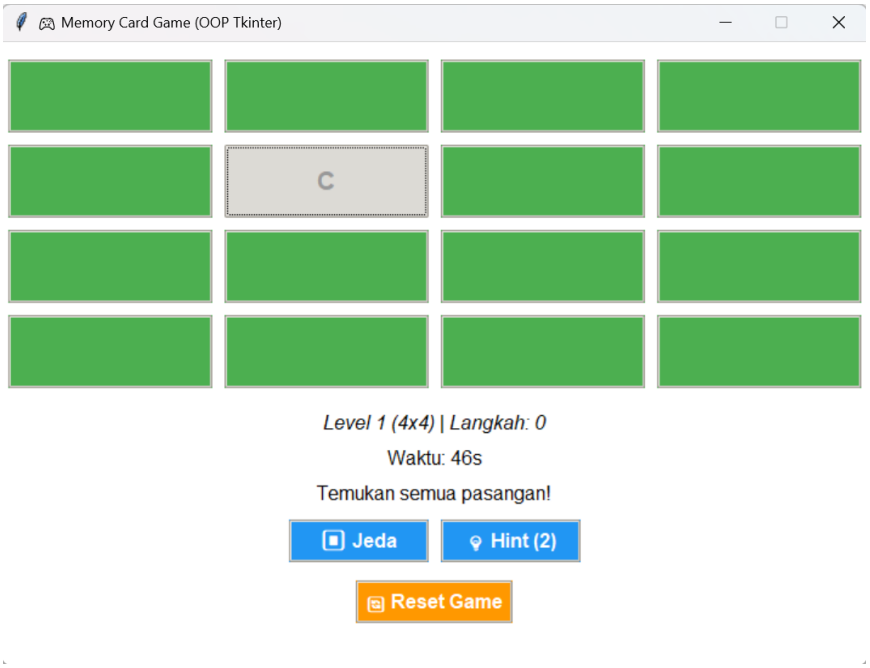
  - **⏸ Jeda / ▶ Lanjut**
  - **Hint**
  - **Reset Game**



5. Cara Bermain

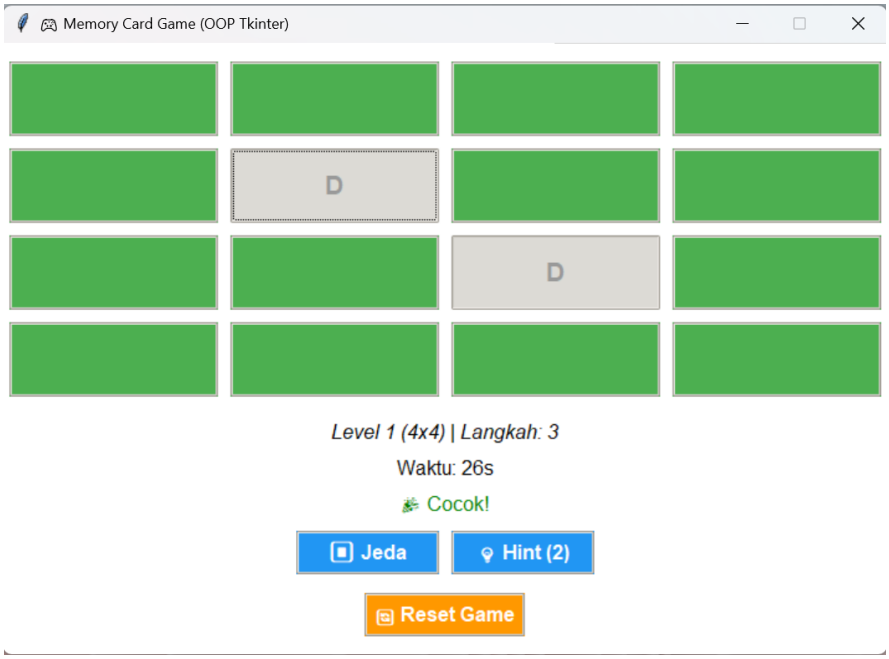
Cara bermain Memory Card Game dilakukan dengan mencocokkan pasangan kartu yang memiliki nilai yang sama. Adapun langkah-langkah permainan adalah sebagai berikut:

1) Pemain mengklik salah satu kartu pada papan permainan untuk membuka dan menampilkan nilai kartu tersebut.

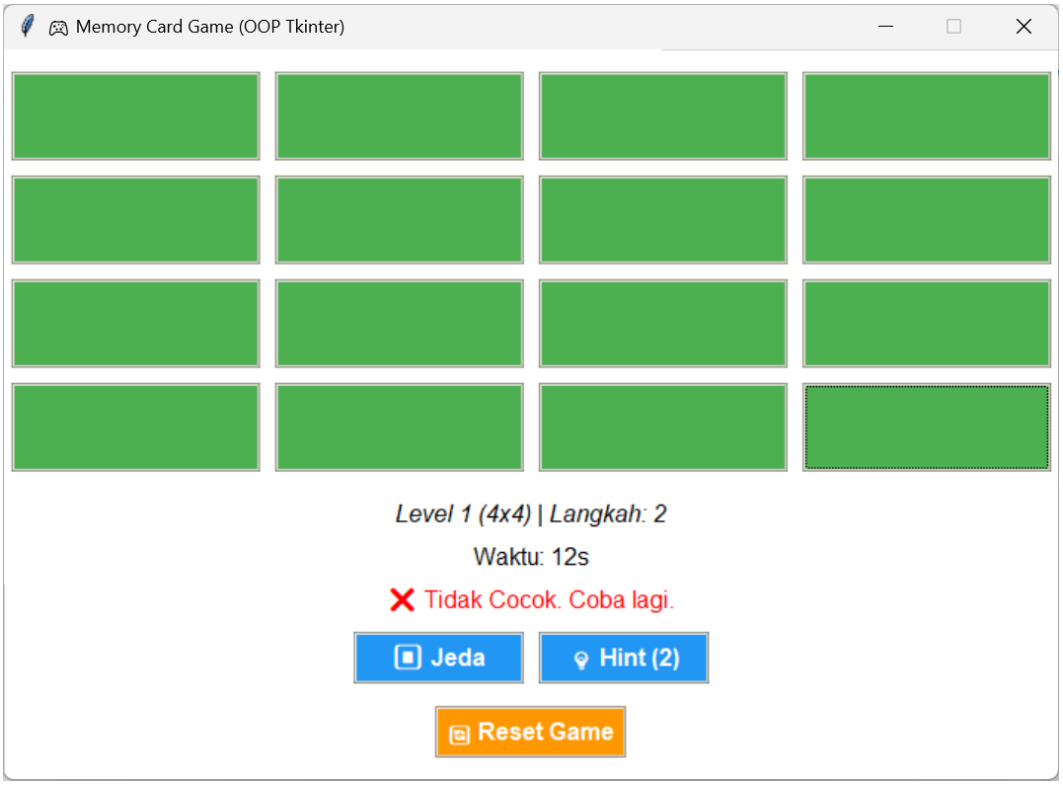


2) Pemain kemudian mengklik kartu lain yang berbeda untuk mencari pasangan dengan nilai yang sama.

3) Apabila kedua kartu yang dibuka memiliki nilai yang sama, maka kartu tersebut dinyatakan **cocok**, ditandai dengan perubahan pada tampilan warna kartu, dan kartu akan terkunci sehingga tidak dapat dibuka kembali.



4) Apabila kedua kartu yang dibuka memiliki nilai yang berbeda, maka kartu akan tertutup kembali setelah beberapa saat.



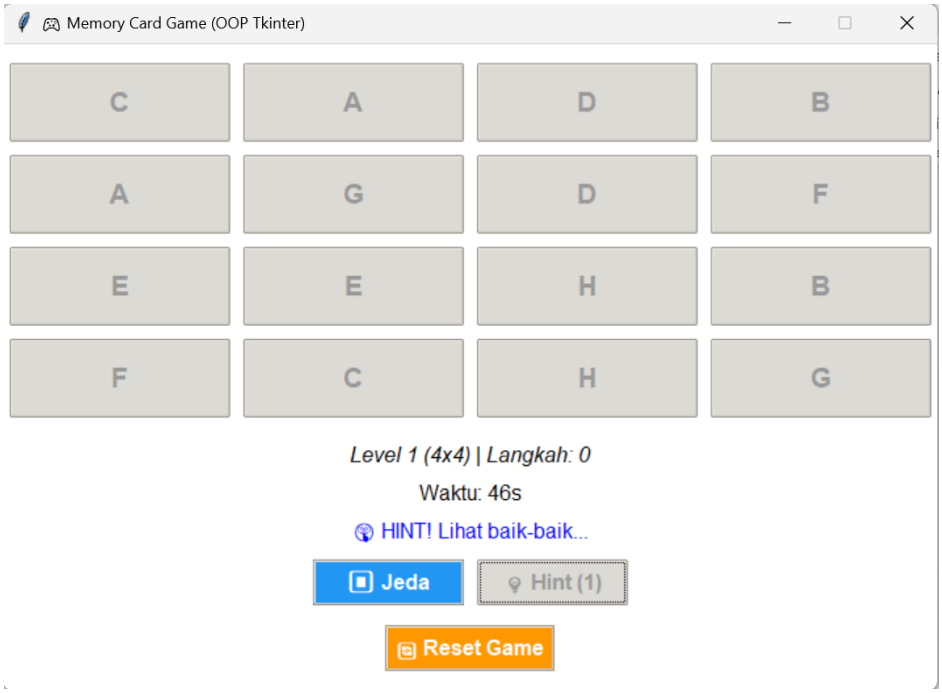
Selama permainan berlangsung, aplikasi akan menampilkan pesan umpan balik kepada pengguna.
Jika dua kartu yang dibuka memiliki nilai yang sama, akan muncul tulisan **"Cocok!"** berwarna hijau sebagai tanda bahwa pasangan kartu berhasil ditemukan. Sebaliknya, jika kartu yang dibuka tidak sama, akan muncul tulisan **"Tidak Cocok. Coba lagi."** berwarna merah sebagai informasi bahwa pemain perlu mencoba kembali. Pesan ini membantu pengguna memahami hasil dari setiap langkah yang dilakukan.

5) Fungsi Tombol Tambahan

a. Tombol Jeda (⏸ / ▶)

- Menghentikan sementara permainan (timer berhenti).
- Menonaktifkan semua kartu.
- Klik kembali untuk melanjutkan.

b. Tombol Hint (💡)



- Menampilkan semua kartu secara terbuka selama 2 detik.
- Hint terbatas (default: 2 kali).
- Setelah selesai, kartu kembali tertutup kecuali kartu yang sudah cocok.

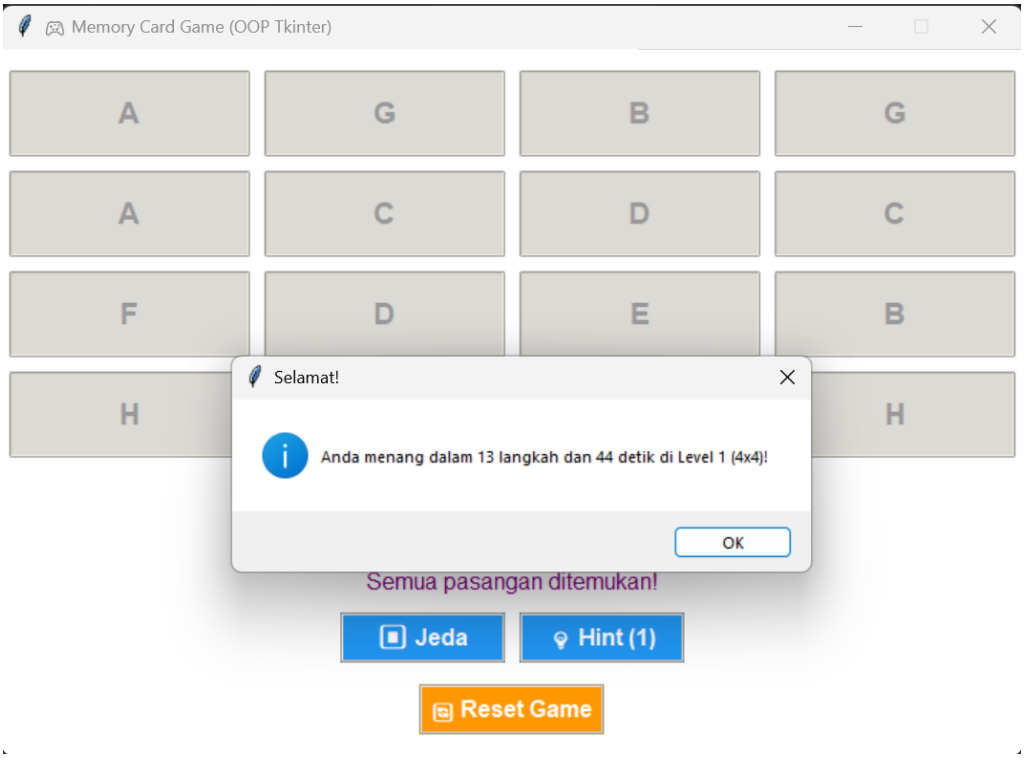
c. Tombol Reset Game (🔄)

- Menghentikan permainan saat ini.
- Kembali ke halaman pemilihan level.

6. Selesai Bermain

Permainan akan dinyatakan **selesai** apabila seluruh pasangan kartu pada papan permainan berhasil ditemukan oleh pengguna.

Aplikasi menampilkan pesan **"Selamat! Anda menang"** yang berisi informasi mengenai **jumlah langkah** yang telah dilakukan serta **waktu** yang dibutuhkan untuk menyelesaikan permainan.berisi waktu dan jumlah langkah.

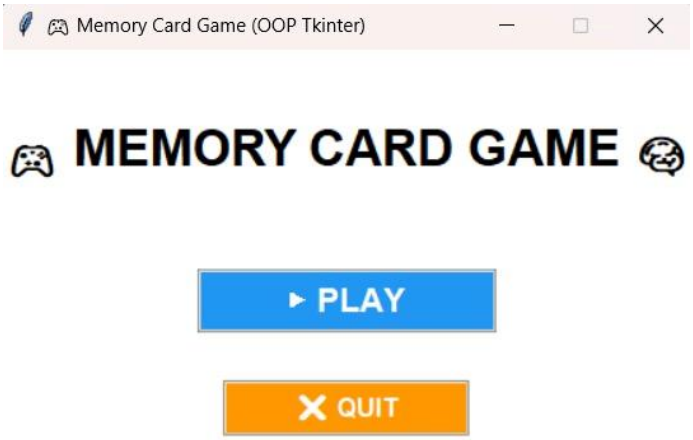


Setelah pengguna menutup pesan tersebut, aplikasi akan secara otomatis mengarahkan pengguna kembali ke **halaman pemilihan level,** sehingga pengguna dapat memilih level lain atau memulai permainan kembali.

7. **Menutup Aplikasi**
   Pengguna dapat mengakhiri penggunaan aplikasi dengan beberapa cara yang telah disediakan, yaitu:
   1. Memilih tombol **QUIT** yang tersedia pada halaman menu utama untuk menutup aplikasi secara langsung.
   2. Menutup jendela aplikasi menggunakan tombol penutup jendela pada sistem operasi.

   Dengan demikian, pengguna dapat keluar dari aplikasi kapan saja sesuai kebutuhan.

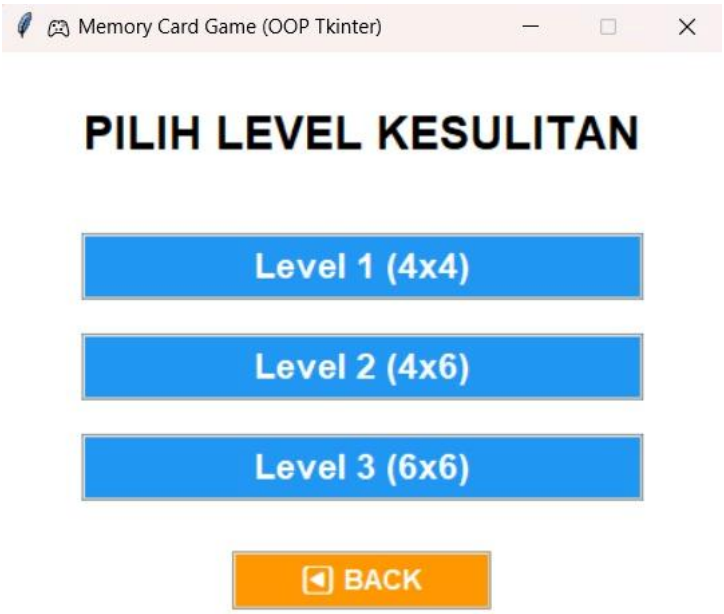
4.4 Fitur Fungsionalitas Utama



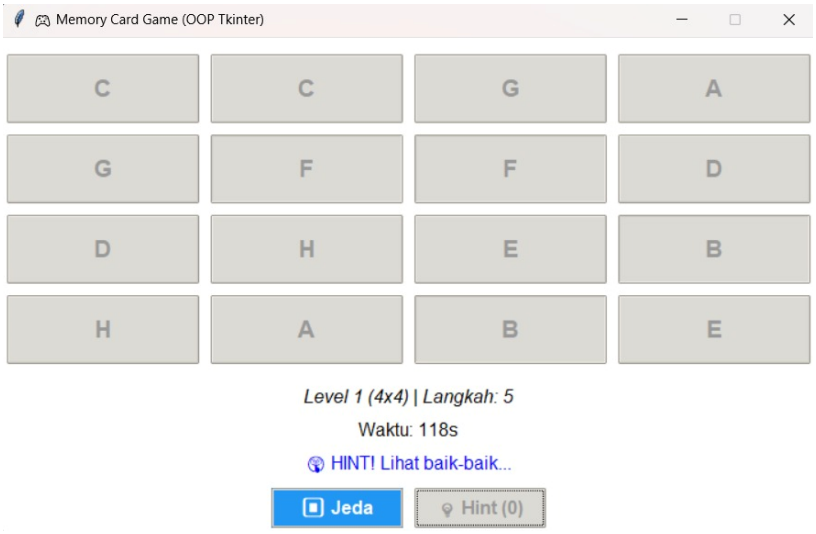
Terdapat fitur **play** dan **quit.** Fungsinya:

Quit : untuk keluar dari window tkinter

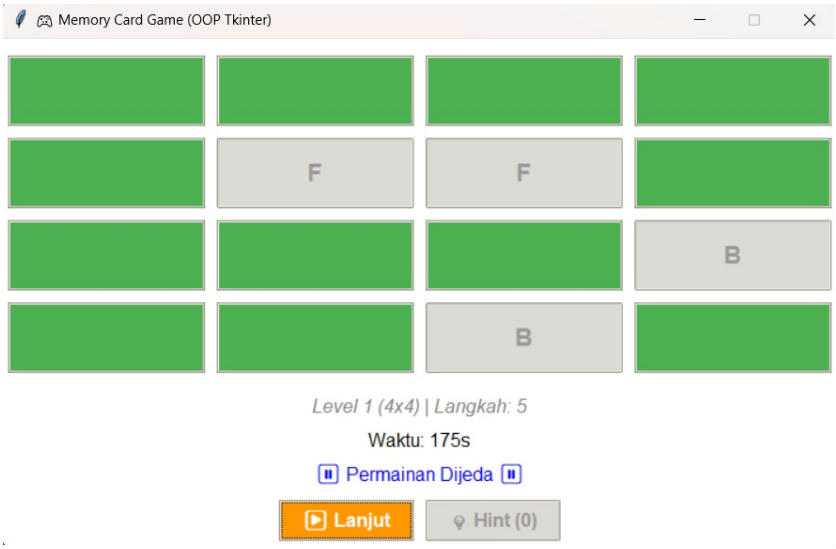
Play: komponen utama untuk memulai permainan (akan diarahkan ke leve select)



Beberapa pilihan level pada memory card game, diantaranya ada level termudah, level 1 dengan layout grid kartu 4x4. Medium 4x6, dan hard 6x6

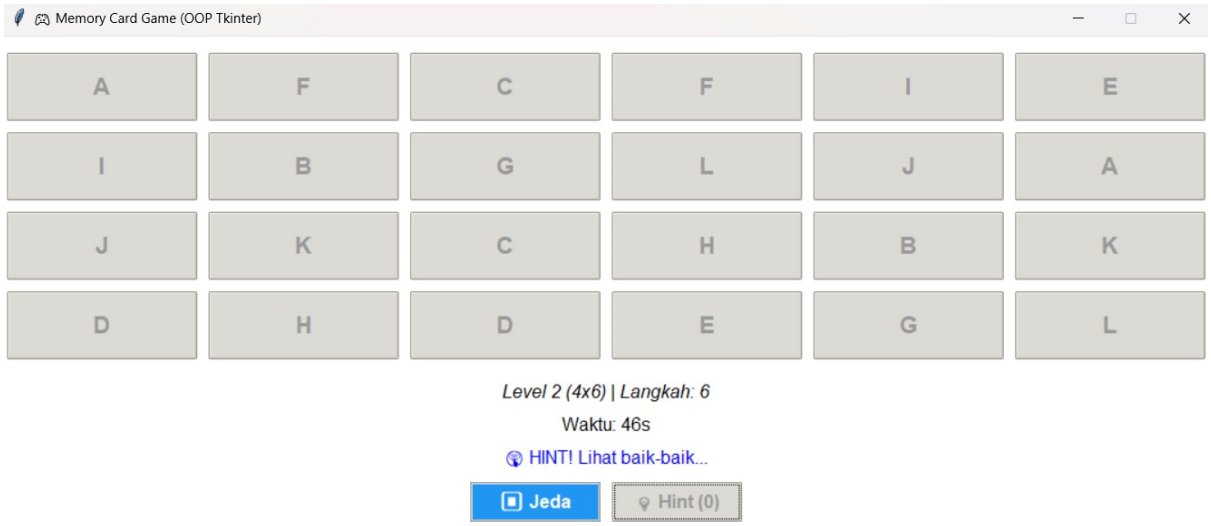


Tampilan game level 1



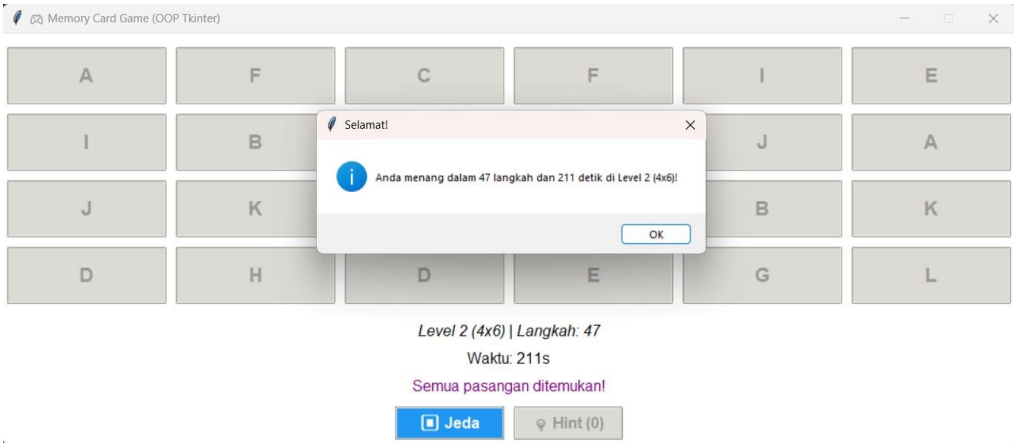
Tampilan ketika membuka beberapa kartu

Tampilan ketika sudah membuka semua kartu

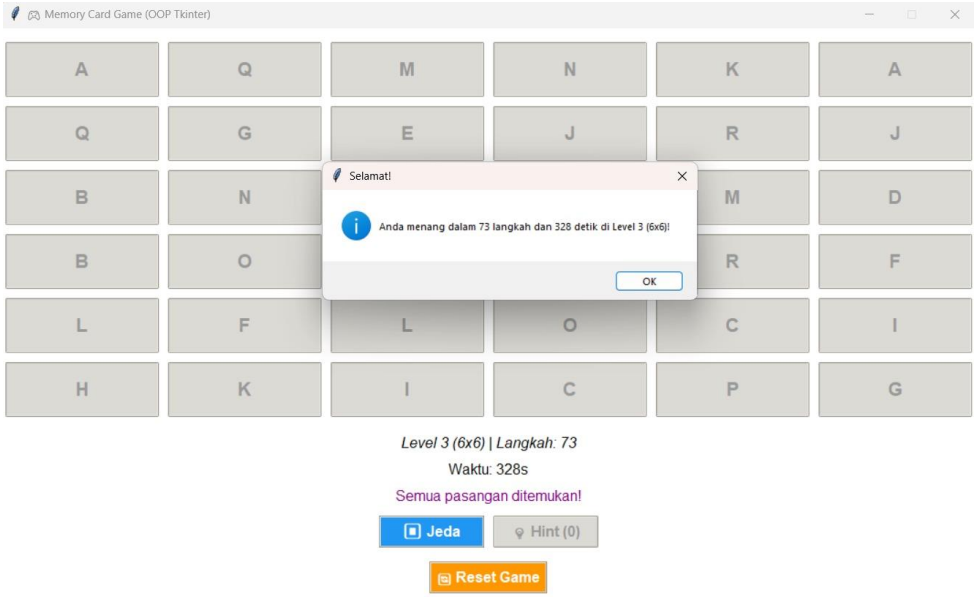


Tampilan ketika menekan button **hint**

Tampilan level 2



Tampilan level complete pada level 2



Tampilan ketika level 3 sudah selesai

Beberapa button utama yg ada di dalam game memory card. User akan menemukan button ini ketika sudah masuk ke dalam game. Tiap button berfungsi:

- Reset game: untuk keluar dari game/level dan mengulangnya kembali
- Jeda : untuk pause timer/menjeda game
- Hint: untuk menampilkan keseluruhan kartu dalam waktu 2 detik, dan memiliki kesempatan 2 kali untuk menggunakannya.

BAB V KESIMPULAN

Proyek game "Memory Card" ini telah berhasil menerapkan konsep Object-Oriented Programming (OOP) dengan menggunakan Python Tkinter, di mana prinsip encapsulation, inheritance, dan polymorphism diimplementasikan secara efektif dalam struktur kelas seperti GameConfig, CardButton, dan MemoryGame, sehingga menghasilkan aplikasi yang modular, mudah dipelihara, serta dilengkapi fitur-fitur interaktif seperti tiga tingkat kesulitan, timer, pause, dan hint yang tidak hanya berfungsi sebagai hiburan tetapi juga sebagai media edukasi untuk memahami penerapan OOP dalam pengembangan perangkat lunak berbasis GUI.

Link Github Repository:

Zaryan19/MemoryCardGame_UASProjectPBO: Project UAS Pemrograman Berorientasi Objek Memory Card Game berbasis Tkinter Python dengan penerapan konsep OOP. Kelompok 1 2024A