IMPLEMENTASI STRUKTUR DATA STACK
PADA FITUR UNDO/REDO DAN NAVIGASI

Disusun Oleh

ABDULLAH AZZAM RABBANI

10240038

Ammar Ichsan Anthony

10240005

Laode Achmed Sayyed Purnomo

10240013

KELAS 10.2A.01

PROGRAM STUDI REKAYASA PERANGKAT LUNAK

UNIVERSITAS BINA SARANA INFORMATIKA

MARGONDA

2024/2025

KATA PENGANTAR

Puji syukur kehadirat Tuhan Yang Maha Esa atas segala rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan makalah ini tepat pada waktunya. Makalah ini berjudul "Implementasi Struktur Data Stack pada Fitur Undo/Redo dan Navigasi" disusun sebagai salah satu pemenuhan tugas mata kuliah.

Makalah ini membahas konsep dasar struktur data stack dan implementasinya dalam bahasa pemrograman Python untuk membangun fungsionalitas yang umum ditemukan dalam aplikasi perangkat lunak, yaitu mekanisme undo/redo untuk aksi pengguna dan sistem navigasi histori halaman.

Penulis menyadari bahwa makalah ini masih jauh dari sempurna. Oleh karena itu, kritik dan saran yang membangun sangat diharapkan demi perbaikan di masa mendatang. Semoga makalah ini dapat memberikan manfaat dan menambah wawasan bagi pembaca.

Jakarta, 06 Juni 2025

Penyusun

DAFTAR ISI

BAB I
PENDAHULUAN

1.1. Latar Belakang

Dalam pengembangan perangkat lunak modern, pengalaman pengguna (user experience) menjadi faktor krusial. Fitur seperti kemampuan untuk membatalkan (undo) dan mengulang (redo) aksi, serta navigasi yang intuitif antar halaman atau state, sangat meningkatkan kenyamanan pengguna. Di balik layar, fitur-fitur ini seringkali memanfaatkan struktur data fundamental, salah satunya adalah stack (tumpukan).

Struktur data stack, dengan prinsip Last-In, First-Out (LIFO), secara alami cocok untuk mengelola histori aksi atau urutan halaman yang dikunjungi. Setiap aksi baru atau kunjungan halaman baru 'ditumpuk' di atas stack. Operasi undo atau kembali (back) mengambil elemen teratas dari stack, sementara redo atau maju (forward) memerlukan mekanisme stack tambahan. Memahami implementasi stack untuk fitur-fitur ini penting bagi pengembang perangkat lunak.

1.2. Rumusan Masalah

Berdasarkan latar belakang tersebut, rumusan masalah dalam makalah ini adalah sebagai berikut:

1. Bagaimana konsep struktur data stack diterapkan untuk mengimplementasikan fitur undo dan redo dalam sebuah aplikasi?
2. Bagaimana struktur data stack digunakan untuk mengelola histori navigasi (back dan forward) antar halaman atau state?
3. Bagaimana implementasi fitur-fitur tersebut menggunakan bahasa pemrograman Python?

1.3. Tujuan

Tujuan dari penulisan makalah ini adalah:

1. Menjelaskan penerapan struktur data stack untuk mekanisme undo/redo.
2. Menjelaskan penggunaan struktur data stack untuk manajemen histori navigasi.

3. Mendemonstrasikan implementasi fitur undo/redo dan navigasi menggunakan stack dalam bahasa Python melalui contoh program interaktif.

1.4. Manfaat

Makalah ini diharapkan dapat memberikan manfaat sebagai berikut:

1. Memberikan pemahaman praktis tentang aplikasi struktur data stack dalam pengembangan perangkat lunak.

2. Menjadi referensi bagi mahasiswa atau pengembang yang ingin mempelajari implementasi fitur undo/redo dan navigasi.

3. Menunjukkan contoh konkret penggunaan Python untuk membangun fungsionalitas berbasis stack.

BAB II
TINJAUAN PUSTAKA

2.1. Bahasa Pemrograman Python

Python adalah bahasa pemrograman interpretatif tingkat tinggi yang dikenal karena sintaksisnya yang bersih, mudah dibaca, dan ekspresif. Diciptakan oleh Guido van Rossum dan pertama kali dirilis pada tahun 1991, Python mendukung berbagai paradigma pemrograman, termasuk pemrograman berorientasi objek, imperatif, fungsional, dan prosedural. Python memiliki pustaka standar yang luas dan ekosistem pustaka pihak ketiga yang kaya, menjadikannya pilihan populer untuk berbagai aplikasi, mulai dari pengembangan web, analisis data, kecerdasan buatan, hingga otomatisasi tugas.

Dalam konteks makalah ini, Python dipilih karena kemudahan penggunaannya dalam mengimplementasikan struktur data dan algoritma. Struktur data bawaan seperti list dapat dengan mudah diadaptasi untuk berfungsi sebagai stack, meskipun

implementasi kelas stack kustom seringkali lebih disukai untuk kejelasan dan enkapsulasi.

2.2. Struktur Data Stack

Stack (tumpukan) adalah struktur data linear abstrak yang mengikuti prinsip Last-In, First-Out (LIFO). Artinya, elemen terakhir yang dimasukkan ke dalam stack akan menjadi elemen pertama yang dikeluarkan. Analogi yang sering digunakan adalah tumpukan piring; piring baru diletakkan di atas, dan piring yang diambil adalah yang paling atas.

Operasi dasar pada stack meliputi:

1. Push: Menambahkan elemen baru ke posisi paling atas (top) stack.
2. Pop: Menghapus dan mengembalikan elemen yang berada di posisi paling atas stack. Operasi ini gagal jika stack kosong.
3. Peek (atau Top): Mengembalikan elemen di posisi paling atas tanpa menghapusnya. Operasi ini gagal jika stack kosong.
4. isEmpty: Memeriksa apakah stack tidak berisi elemen.
5. Size (atau Length): Mengembalikan jumlah elemen dalam stack.

Stack memiliki banyak aplikasi dalam ilmu komputer, termasuk evaluasi ekspresi matematika, penelusuran graf (Depth First Search), manajemen memori (call stack untuk fungsi), dan seperti yang dibahas dalam makalah ini, implementasi fitur undo/redo serta histori navigasi.

BAB III
METODE TUGAS

3.1. Alat dan Bahan

Alat dan bahan yang digunakan dalam implementasi program demo ini adalah:

1. Perangkat Keras: Komputer atau laptop dengan sistem operasi yang mendukung Python.
2. Perangkat Lunak:
    a. Interpreter Python (versi 3.x direkomendasikan).
    b. Teks editor atau Integrated Development Environment (IDE) untuk menulis kode (misalnya, VS Code, PyCharm, Notepad++).
    c. File skrip Python `stack_app.py` yang berisi definisi kelas `Stack`, `UndoRedoManager`, dan `NavigationHistory`.
    d. File skrip Python `interactive_demo_full.py` yang berisi logika demo interaktif.

3.2. Langkah Pembuatan

Langkah-langkah pembuatan program demo interaktif adalah sebagai berikut:

1. Definisi Kelas Stack: Membuat kelas `Stack` dasar yang mengimplementasikan operasi inti LIFO (push, pop, peek, isEmpty, size, clear). Kelas ini menjadi fondasi bagi manajer undo/redo dan histori navigasi.
2. Implementasi UndoRedoManager: Membuat kelas `UndoRedoManager` yang menggunakan dua instance dari kelas `Stack`: satu untuk menyimpan histori aksi yang bisa di-undo (`undo_stack`) dan satu lagi untuk menyimpan histori aksi yang bisa di-redo (`redo_stack`). Mengimplementasikan metode `perform_action`, `undo`, dan `redo`.
3. Implementasi Navigation History: Membuat kelas `NavigationHistory` yang juga menggunakan dua instance `Stack`: satu untuk histori halaman sebelumnya (`back_stack`) dan satu untuk histori halaman selanjutnya

(`forward_stack`). Mengimplementasikan metode `visit`, `go_back`, dan `go_forward`.

4. Pembuatan Demo Interaktif: Membuat skrip `interactive_demo_full.py` yang mengimpor kelas-kelas di atas. Skrip ini menyediakan menu berbasis teks bagi pengguna untuk berinteraksi dengan fitur undo/redo dan navigasi, serta menampilkan visualisasi state stack saat ini.
5. Pengujian: Menjalankan skrip demo interaktif dan menguji semua fungsionalitas untuk memastikan logika stack bekerja sesuai harapan.

3.3.1. Listing Program

Berikut adalah listing kode sumber lengkap untuk file `interactive_demo_full.py`. Kode untuk `stack_app.py`

```
def visualize_stack(stack_name, stack_items):
    """Helper function to visualize a stack vertically."""
    print(f"--- {stack_name} ---")
    if not stack_items:
        print("[ Kosong ]")
    else:
        print("  (Top)")
        for item in reversed(stack_items):
            print(f"  [ {item} ]")
        print("  (Bottom)")
    print("-" * (len(stack_name) + 6)) # Separator line

def handle_advanced_stack_operations(nav_history: NavigationHistory,
undo_redo_manager: UndoRedoManager):
    """Handles advanced stack operations like size, peek, clear."""
    print("\n--- Operasi Stack Lanjutan ---")
    print("Pilih Stack:")
    print("  1: Back Stack (Navigasi)")
    print("  2: Forward Stack (Navigasi)")
    print("  3: Redo Stack (Undo/Redo)")
    print("  0: Kembali ke Menu Utama")

    stack_choice = input("Pilih stack (1-3) atau 0 untuk kembali: ")

    target_stack: Stack | None = None
    stack_display_name = ""

    if stack_choice == '1':
        target_stack = nav_history.back_stack
        stack_display_name = "Back Stack"
    elif stack_choice == '2':
        target_stack = nav_history.forward_stack
        stack_display_name = "Forward Stack"
    elif stack_choice == '3':
```

```
        target_stack = undo_redo_manager.redo_stack
        stack_display_name = "Redo Stack"
    elif stack_choice == '0':
        return
    else:
        print("Pilihan stack tidak valid.")
        return

    print(f"\nOperasi untuk {stack_display_name}:")
    print("  1: Tampilkan Ukuran (Size)")
    print("  2: Lihat Elemen Teratas (Peek)")
    print("  3: Kosongkan Stack (Clear)")
    print("  0: Kembali")

    op_choice = input("Pilih operasi (1-3) atau 0 untuk kembali: ")

    if op_choice == '1':
        print(f"Ukuran {stack_display_name}: {target_stack.size()}")
    elif op_choice == '2':
        top_element = target_stack.peek()
        if top_element is not None:
            print(f"Elemen teratas {stack_display_name}: [
{top_element} ]")
        else:
            print(f"{stack_display_name} kosong, tidak ada elemen
teratas.")
    elif op_choice == '3':
        target_stack.clear()
        print(f"{stack_display_name} telah dikosongkan.")
    elif op_choice == '0':
        return
    else:
        print("Pilihan operasi tidak valid.")

def run_interactive_demo():
    """Menjalankan simulasi interaktif untuk Undo/Redo dan Navigasi."""

    undo_redo_manager = UndoRedoManager()
    navigation_history = NavigationHistory()

    print("Selamat Datang di Demo Interaktif Stack!")
    print("Fitur Lengkap: Undo/Redo, Navigasi, Visualisasi, Size, Peek,
Clear.")

    while True:
        print("\n--- Menu Utama ---")
        print("Pilih aksi:")
        print("  [Undo/Redo]")
        print("    1: Lakukan Aksi")
        print("    2: Undo Aksi Terakhir")
        print("    3: Redo Aksi Terakhir")
        print("  [Navigasi]")
        print("    4: Kunjungi Halaman Baru")
        print("    5: Kembali (Back)")
        print("    6: Maju (Forward)")
        print("  [Tampilan & Info]")
        print("    7: Tampilkan Status & Visualisasi Stack")
```

```
        print("    8: Operasi Stack Lanjutan (Size, Peek, Clear)")
        print("  [Lainnya]")
        print("    0: Keluar")

        choice = input("Masukkan pilihan Anda: ")

        if choice == '1':
            action_description = input("Masukkan deskripsi aksi (misal:
'menulis paragraf'): ")
            if action_description:
                undo_redo_manager.perform_action(action_description)
            else:
                print("Deskripsi aksi tidak boleh kosong.")
        elif choice == '2':
            undo_redo_manager.undo()
        elif choice == '3':
            undo_redo_manager.redo()
        elif choice == '4':
            page_name = input("Masukkan nama halaman (misal: 'Halaman
Login'): ")
            if page_name:
                navigation_history.visit(page_name)
            else:
                print("Nama halaman tidak boleh kosong.")
        elif choice == '5':
            navigation_history.go_back()
        elif choice == '6':
            navigation_history.go_forward()
        elif choice == '7':
            print("\n--- Status & Visualisasi Undo/Redo ---")
            visualize_stack("Undo Stack",
undo_redo_manager.undo_stack._items)
            visualize_stack("Redo Stack",
undo_redo_manager.redo_stack._items)
            print(f"\nState Saat Ini (Undo/Redo):
{undo_redo_manager.current_state}")

            print("\n--- Status & Visualisasi Navigasi ---")
            visualize_stack("Back Stack",
navigation_history.back_stack._items)
            print(f"\nHalaman Saat Ini: [
{navigation_history.current_page} ]")
            visualize_stack("Forward Stack",
navigation_history.forward_stack._items)

        elif choice == '8':
            handle_advanced_stack_operations(navigation_history,
undo_redo_manager)

        elif choice == '0':
            print("Keluar dari demo. Sampai jumpa!")
            break
        else:
            print("Pilihan tidak valid. Silakan coba lagi.")

if __name__ == "__main__":
    # Pastikan stack_app.py ada di direktori yang sama atau PYTHONPATH
```

```
    try:
        run_interactive_demo()
    except ImportError:
        print("Error: Pastikan file 'stack_app.py' berada di direktori
yang sama")
        print("atau dapat diakses melalui PYTHONPATH Anda.")
    except Exception as e:
        print(f"Terjadi error tak terduga: {e}")
```

3.3.2. Hasil Running

Berikut adalah contoh output (hasil running) dari eksekusi program `interactive_demo_full.py`, menunjukkan beberapa interaksi pengguna:

```
Selamat Datang di Demo Interaktif Stack!
Fitur Lengkap: Undo/Redo, Navigasi, Visualisasi, Size, Peek, Clear.

--- Menu Utama ---
Pilih aksi:
  [Undo/Redo]
    1: Lakukan Aksi
    2: Undo Aksi Terakhir
    3: Redo Aksi Terakhir
  [Navigasi]
    4: Kunjungi Halaman Baru
    5: Kembali (Back)
    6: Maju (Forward)
  [Tampilan & Info]
    7: Tampilkan Status & Visualisasi Stack
    8: Operasi Stack Lanjutan (Size, Peek, Clear)
  [Lainnya]
    0: Keluar
Masukkan pilihan Anda: 4
Masukkan nama halaman (misal: 'Halaman Login'): Halaman Utama
Mengunjungi: Halaman Utama

--- Menu Utama ---
... (menu repeats)
Masukkan pilihan Anda: 4
Masukkan nama halaman (misal: 'Halaman Login'): Halaman Produk
Mengunjungi: Halaman Produk

--- Menu Utama ---
... (menu repeats)
Masukkan pilihan Anda: 7

--- Status & Visualisasi Undo/Redo ---
--- Undo Stack ---
[ Kosong ]
---------------
--- Redo Stack ---
[ Kosong ]
---------------

State Saat Ini (Undo/Redo): State Awal

--- Status & Visualisasi Navigasi ---
```

```
--- Back Stack ---
  (Top)
  [ Halaman Utama ]
  (Bottom)
----------------

Halaman Saat Ini: [ Halaman Produk ]
--- Forward Stack ---
[ Kosong ]
-----------------

--- Menu Utama ---
... (menu repeats)
Masukkan pilihan Anda: 5
Kembali ke: Halaman Utama

--- Menu Utama ---
... (menu repeats)
Masukkan pilihan Anda: 7

--- Status & Visualisasi Undo/Redo ---
... (unchanged)

--- Status & Visualisasi Navigasi ---
--- Back Stack ---
[ Kosong ]
----------------

Halaman Saat Ini: [ Halaman Utama ]
--- Forward Stack ---
  (Top)
  [ Halaman Produk ]
  (Bottom)
------------------

--- Menu Utama ---
... (menu repeats)
Masukkan pilihan Anda: 0
Keluar dari demo. Sampai jumpa!
```

BAB IV
PEMBAHASAN

4.1. Analisa dan Pembahasan

Program demo interaktif yang dibuat berhasil menunjukkan implementasi struktur data stack untuk dua fitur penting: undo/redo dan histori navigasi. Pembahasan berikut akan menganalisis cara kerja masing-masing fitur berdasarkan implementasi stack.

Implementasi Undo/Redo:

Kelas `UndoRedoManager` menggunakan dua stack: `undo_stack` dan `redo_stack`. Ketika metode `perform_action(deskripsi)` dipanggil:

1. Deskripsi aksi didorong (push) ke `undo_stack`.
2. `redo_stack` dikosongkan, karena aksi baru membatalkan kemungkinan redo sebelumnya.

Ketika metode `undo()` dipanggil:

1. Memeriksa apakah `undo_stack` tidak kosong.
2. Jika tidak kosong, aksi teratas di-pop dari `undo_stack`.
3. Aksi yang di-pop tersebut kemudian di-push ke `redo_stack`, memungkinkan aksi tersebut untuk di-redo nanti.

Ketika metode `redo()` dipanggil:

1. Memeriksa apakah `redo_stack` tidak kosong.
2. Jika tidak kosong, aksi teratas di-pop dari `redo_stack`.
3. Aksi yang di-pop tersebut di-push kembali ke `undo_stack`, memungkinkan aksi tersebut di-undo lagi.

Pendekatan ini secara efektif mengelola histori aksi dan memungkinkan pembatalan serta pengulangan berkat sifat LIFO dari stack.

Implementasi Navigasi:

Kelas `NavigationHistory` juga menggunakan dua stack: `back_stack` dan `forward_stack`, serta variabel `current_page`.

Ketika metode `visit(halaman_baru)` dipanggil:

1. Jika `current_page` tidak kosong (bukan kunjungan pertama), halaman saat ini di-push ke `back_stack`.
2. `halaman_baru` ditetapkan sebagai `current_page`.
3. `forward_stack` dikosongkan, karena kunjungan baru membatalkan histori 'maju'.

Ketika metode `go_back()` dipanggil:

1. Memeriksa apakah `back_stack` tidak kosong.
2. Jika tidak kosong, `current_page` saat ini di-push ke `forward_stack` (memungkinkan 'maju' kembali ke halaman ini).
3. Halaman teratas di-pop dari `back_stack` dan ditetapkan sebagai `current_page` baru.

Ketika metode `go_forward()` dipanggil:

1. Memeriksa apakah `forward_stack` tidak kosong.
2. Jika tidak kosong, `current_page` saat ini di-push ke `back_stack` (memungkinkan 'kembali' ke halaman ini).
3. Halaman teratas di-pop dari `forward_stack` dan ditetapkan sebagai `current_page` baru.

Logika ini meniru cara kerja tombol back/forward pada browser web, memanfaatkan dua stack untuk menyimpan histori masa lalu dan masa depan relatif terhadap halaman saat ini.

Fitur tambahan seperti `size`, `peek`, dan `clear` pada stack memberikan kemampuan introspeksi dan kontrol lebih lanjut, yang didemonstrasikan dalam menu 'Operasi Stack Lanjutan'. Visualisasi stack membantu memahami state internal dari `UndoRedoManager` dan `NavigationHistory` secara real-time.

BAB V
PENUTUP

5.1. Kesimpulan

Berdasarkan pembahasan yang telah dilakukan, dapat ditarik kesimpulan bahwa struktur data *stack* dengan prinsip LIFO (*Last-In, First-Out*) terbukti sangat efektif untuk mengimplementasikan berbagai fungsionalitas interaktif dalam aplikasi. Secara spesifik, penggunaan dua *stack*, masing-masing untuk mekanisme *undo* dan *redo*, memungkinkan pengelolaan histori aksi pengguna secara efisien, sehingga tindakan yang telah dibatalkan dapat dengan mudah diulang.

Demikian pula, fitur navigasi *back/forward* dapat diimplementasikan secara optimal dengan memanfaatkan dua *stack* terpisah, yang berfungsi untuk mencatat histori halaman yang telah dikunjungi pengguna dan halaman yang dapat diakses kembali setelah melakukan operasi kembali (navigasi '*back*'). Lebih lanjut, bahasa pemrograman Python menunjukkan keunggulan dalam memfasilitasi implementasi struktur data *stack*, baik melalui penggunaan tipe data *list* bawaan maupun melalui pembuatan kelas kustom. Kemudahan implementasi ini menjadikan Python pilihan yang sangat sesuai untuk pengembangan prototipe maupun aplikasi final yang memerlukan fungsionalitas berbasis *stack*, sebagaimana telah didemonstrasikan dalam kajian ini.

DAFTAR PUSTAKA

[1] Dokumentasi Resmi Python 3. Tersedia: https://docs.python.org/3/

[2] Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *\Introduction to Algorithms (3rd ed.). MIT Press.