LAPORAN PRAKTIKUM (1) STRUKTUR DATA



Oleh:

Nama : Arkan Ubaidillah Warman

NIM : 2411537001

Dosen Pengampu : Dr. Wahyudi MT .

JARINGAN KOMPUTER

(membuat arraylist1,dan daftar perpustakaan)

DEPARTEMEN INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI

UNIVERITAS ANDALAS

PADANG

1. PENDAHULUAN

Struktur data merupakan elemen penting dalam pengembangan perangkat lunak karena memungkinkan penyimpanan dan pengelolaan data secara efisien. Salah satu struktur data yang sering digunakan adalah *stack*. Stack adalah struktur data linier yang menerapkan prinsip **LIFO** (*Last In, First Out*), yaitu elemen terakhir yang dimasukkan akan menjadi elemen pertama yang dikeluarkan. Stack digunakan dalam berbagai aplikasi seperti algoritma rekursif, penelusuran ekspresi matematika, manajemen memori, dan sistem operasi.

1. TUJUAN PRAKTIKUM

 Mempelajari konsep dasar struktur data stack.

 Mengimplementasikan operasi dasar pada stack seperti **push**, **pop**, dan **peek**.

 Memahami penerapan stack dalam menyelesaikan masalah komputasi.

 Menguji kinerja stack melalui simulasi atau program sederhana.

1. LANGKAH-LANGKAH
2. latihanStack

Sebuah gambar berisi teks, cuplikan layar, software, tampilan

Konten yang dihasilkan AI mungkin salah.

* Stack<Integer> s = new Stack<Integer>();  
  Membuat objek stack dengan tipe data Integer.
* s.push(42); s.push(-3); s.push(17);  
  Tiga elemen ditambahkan ke dalam stack secara berurutan: pertama 42, lalu -3, lalu 17. Karena stack bersifat **LIFO**, elemen terakhir (17) berada di atas.
* s.pop();  
  Menghapus elemen teratas dari stack (17) dan mengembalikannya. Setelah ini, stack berisi [42, -3].
* s.peek();  
  Mengambil elemen teratas (sekarang -3) **tanpa menghapusnya**.

1. contohStack

Sebuah gambar berisi teks, cuplikan layar, software, tampilan

Konten yang dihasilkan AI mungkin salah.

* Membuat array a dengan angka {4, 8, 15, 16, 23, 42}.
* Mengisi elemen-elemen tersebut ke dalam stack.
* Menampilkan:

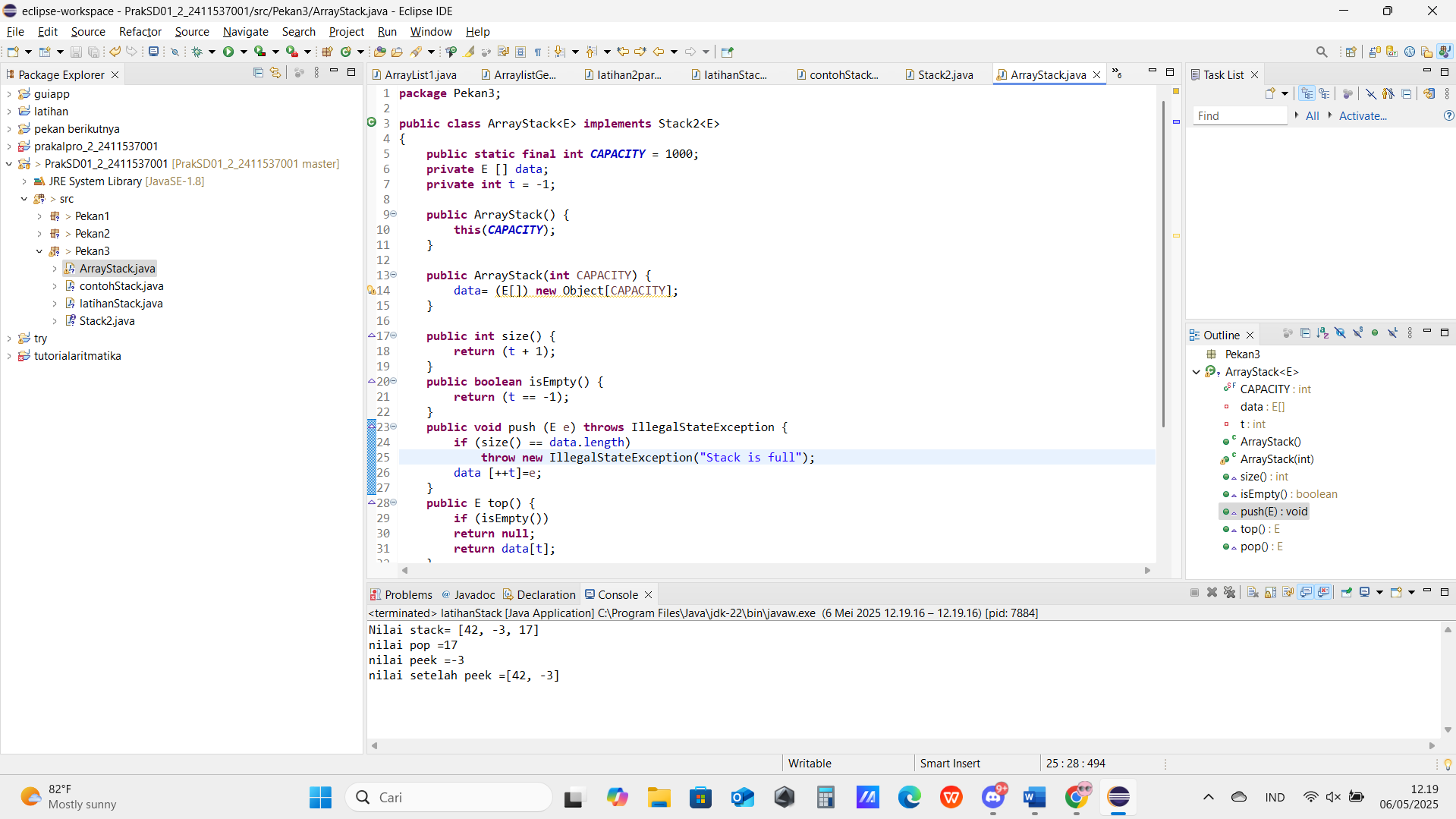
1. Semua elemen saat dimasukkan.
2. Ukuran stack setelah semua data masuk.
3. Elemen paling atas (peek()).
4. Elemen paling atas yang dihapus (pop()).
5. Stack2

Sebuah gambar berisi teks, cuplikan layar, software, tampilan

Konten yang dihasilkan AI mungkin salah.

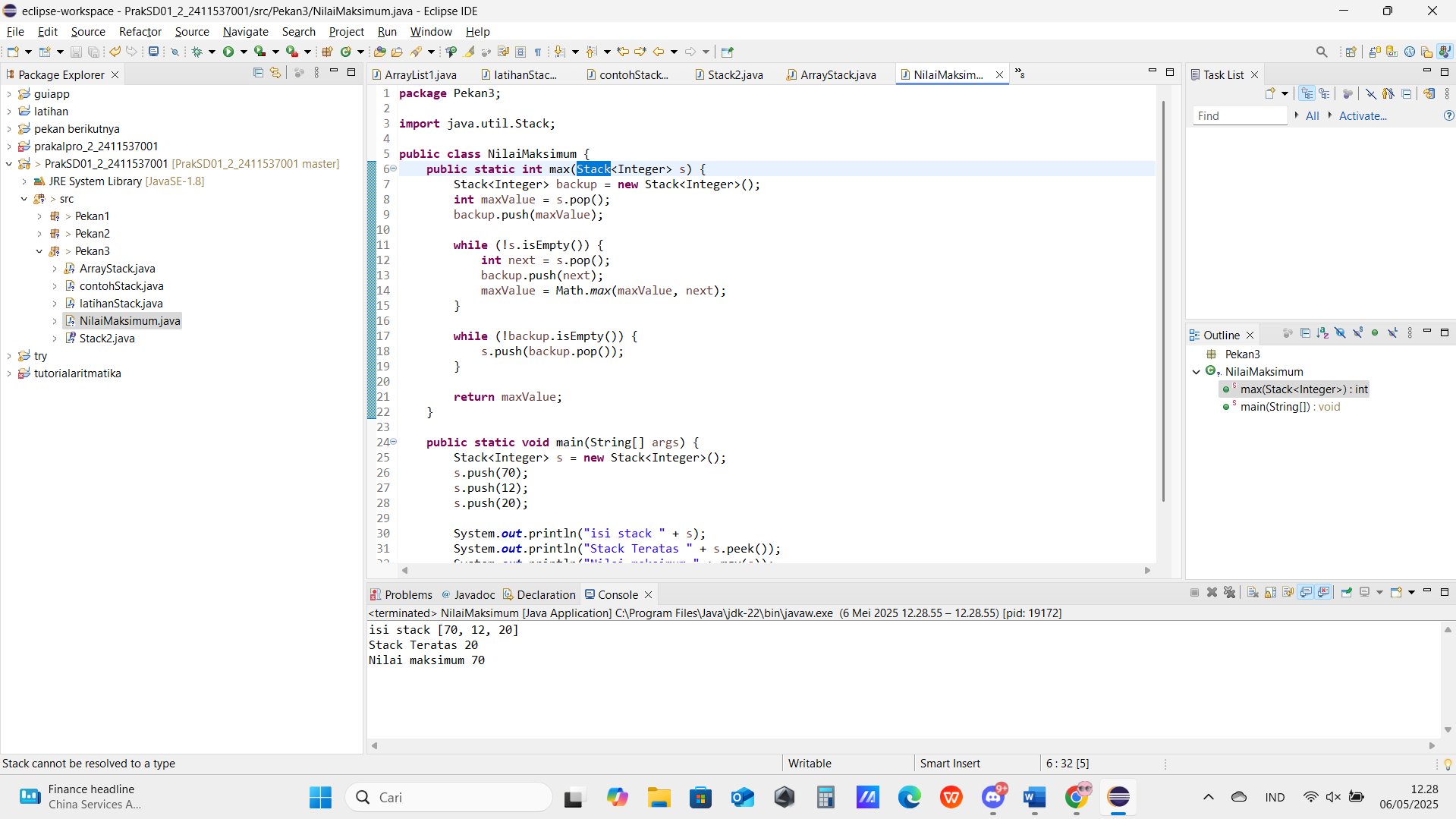
| **Method** | **Fungsi** |
| --- | --- |
| size() | Mengembalikan jumlah elemen dalam stack |
| isEmpty() | Mengembalikan true jika stack kosong |
| push(E e) | Menambahkan elemen e ke atas stack |
| top() | Melihat elemen paling atas stack (tanpa menghapus) |
| pop() | Menghapus dan mengembalikan elemen paling atas stack |

1. ArrayStack<E>



| **Fungsi** | **Tujuan** |
| --- | --- |
| ArrayStack() | Konstruktor default, memakai kapasitas 1000 |
| ArrayStack(int capacity) | Konstruktor dengan kapasitas custom |
| size() | Mengembalikan jumlah elemen dalam stack (t + 1) |
| isEmpty() | Mengecek apakah stack kosong (t == -1) |
| push(E e) | Menambah elemen ke atas stack |
| top() | Mengembalikan elemen paling atas tanpa menghapusnya |
| pop() | Menghapus dan mengembalikan elemen paling atas |

1. nilaiMaksimum



**Fungsi max(Stack<Integer> s)**

Digunakan untuk mencari **nilai maksimum** dari sebuah stack, **tanpa merusak isi stack**.

Langkah-langkahnya:

1. Ambil (pop) elemen pertama sebagai nilai awal maxValue.
2. Simpan semua elemen ke stack backup sambil membandingkan dengan maxValue.
3. Setelah itu, pindahkan semua isi backup kembali ke stack s agar stack kembali seperti semula.
4. Kembalikan maxValue.

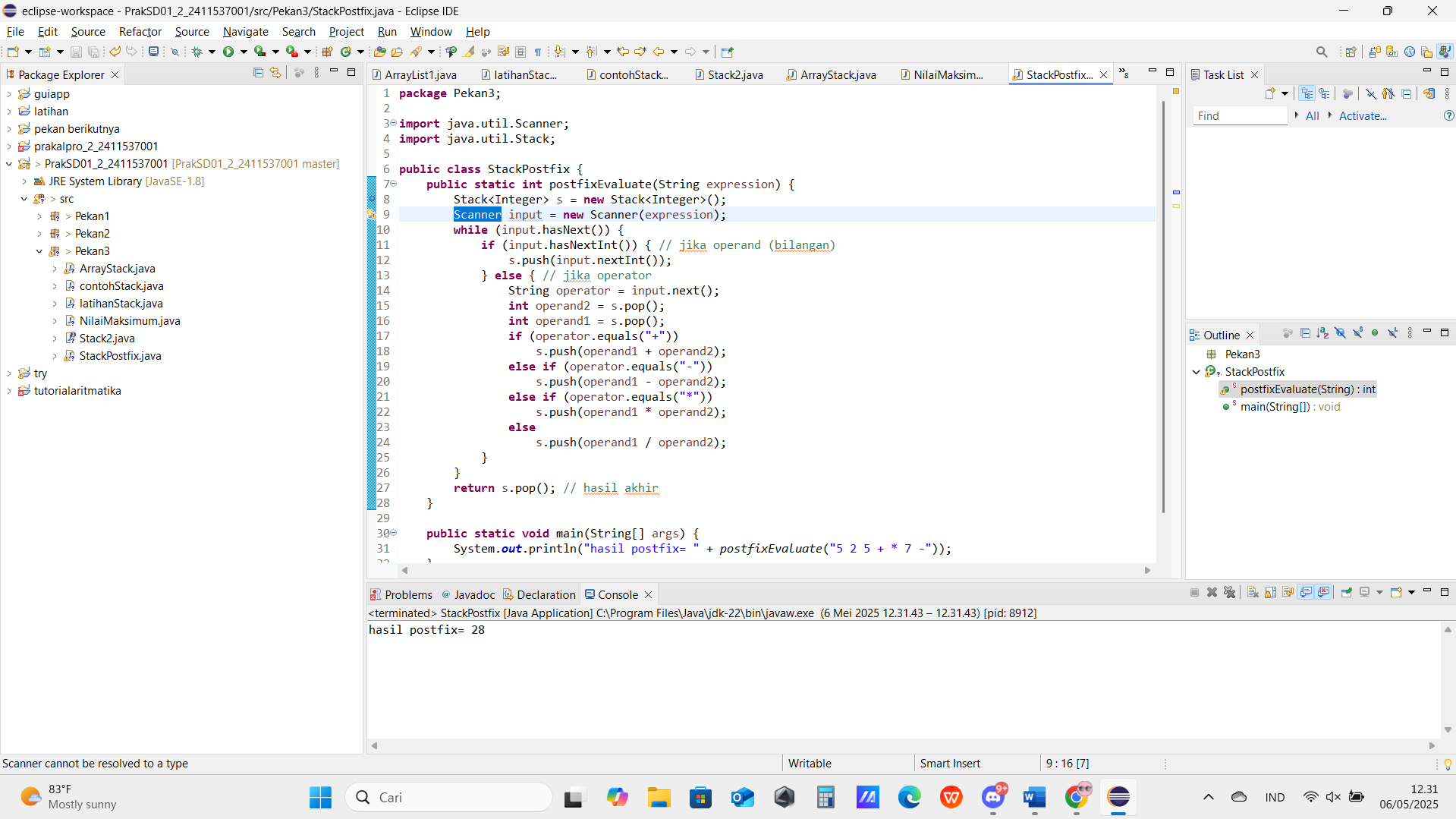
 Stack diisi dengan: 70, 12, dan 20.

 Cetak isi stack.

 Cetak elemen paling atas (peek()).

 Cetak nilai maksimum menggunakan fungsi max().

1. StackPostfix



1. Gunakan Scanner untuk membaca setiap token dari string ekspresi.
2. Jika token berupa angka (operand), masukkan ke stack.
3. Jika token berupa operator (+, -, \*, /):
   * Ambil dua angka terakhir dari stack (operand2, operand1).
   * Hitung: operand1 operator operand2.
   * Masukkan hasilnya kembali ke stack.
4. Setelah semua token diproses, elemen terakhir di stack adalah hasil akhir.
5. KESIMPULAN

Dari percobaan yang dilakukan, dapat disimpulkan bahwa *stack* adalah struktur data yang sederhana namun sangat berguna untuk berbagai proses komputasi yang membutuhkan kontrol urutan eksekusi data. Operasi dasar pada stack seperti **push** dan **pop** dapat diimplementasikan dengan mudah, baik menggunakan array maupun linked list. Pemahaman tentang cara kerja stack sangat penting dalam pengembangan aplikasi yang melibatkan pemrosesan berurutan atau rekursif.