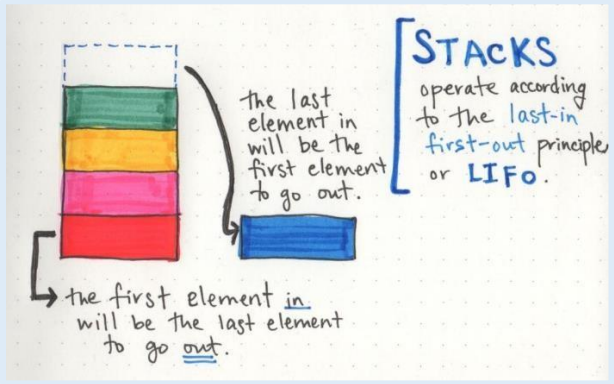
[7] Stack

✧ Stack adalah struktur data linear yang menggunakan konsep LIFO (Last In First Out).



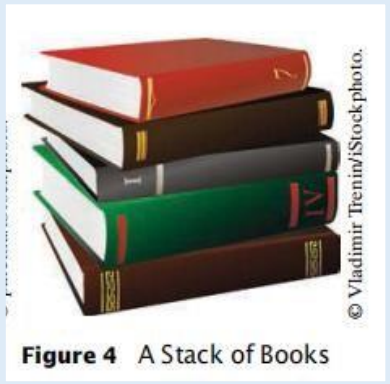
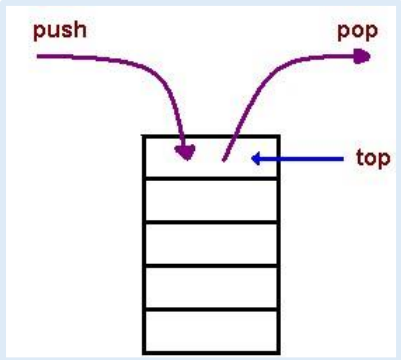
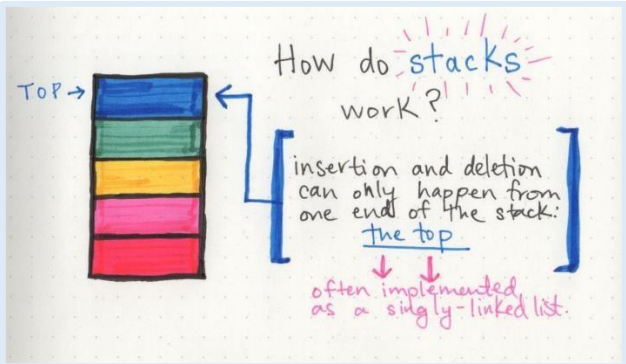


Figure 4 A Stack of Books

✧ Kita hanya bisa **menambah dan menghapus elemen** dari satu sisi saja, yaitu **top** (bagian atas stack).





Operasi yang sering digunakan dalam implementasi stack:

- Push: operasi untuk menambahkan elemen baru ke bagian paling atas (top) dari stack
- Pop: operasi untuk menghapus elemen dari bagian paling atas (top) stack
- Peek: operasi untuk melihat elemen di bagian atas stack tanpa menghapusnya
- isEmpty: operasi untuk memeriksa apakah stack kosong

Implementasi stack di kehidupan sehari-hari

- Saat menggunakan text editor seperti Word, yaitu ketika melakukan undo/redo terdapat operasi push dan pop “state” dari suatu stack
- Saat menggunakan browser seperti Google Chrome, contoh ketika membuka postingan Twitter/X lalu klik tombol “back”, terdapat operasi push (membuka postingan Twitter/X) dan pop (kembali ke beranda)
- Evaluasi ekspresi matematika dan pemrosesan *syntax tree* dalam pengenalan kalimat atau frasa

Contoh program (menggunakan Java Collection)

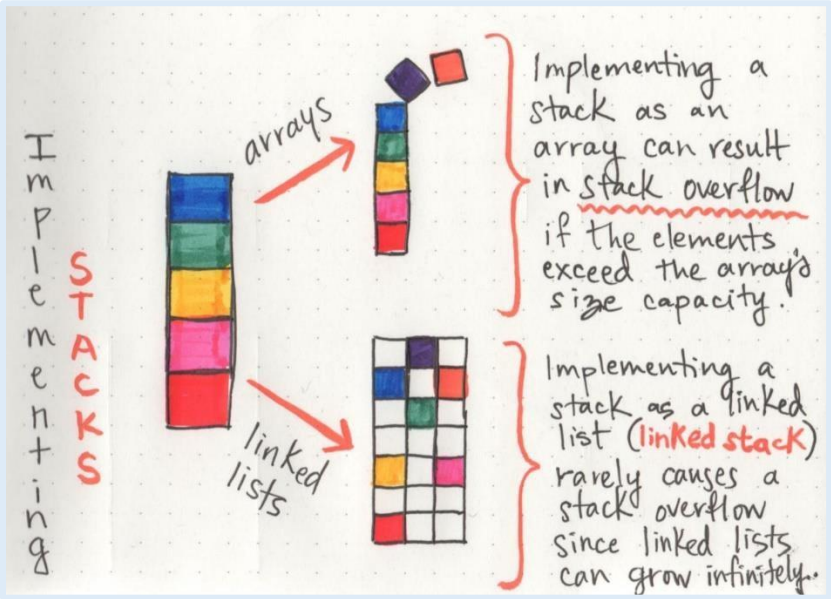
```
import java.util.Stack;

public class StackExample {
    public static void main(String[] args) {
        // Create a new stack
        Stack<Integer> stack = new Stack<>();

        // Push elements onto the stack
        stack.push(1);
        stack.push(2);
        stack.push(3);
        stack.push(4);

        // Print stack
        System.out.println(stack);

        // Pop element from the stack
        stack.pop();
        stack.pop();

        // Print stack after pop operation
        System.out.println(stack);

        // Peek elements of the stack
        System.out.println("The last element is " + stack.peek());

        // Search element

        // Pop elements from the stack
        while(!stack.isEmpty()) {
            stack.pop();
        }

        // Check elements of stack
        System.out.println("Is empty? " + stack.empty());

        // Output
        // [1, 2, 3, 4]
        //  [1, 2]
        // The last element is 2
        // Is empty? true
    }
}
```

Contoh program (menggunakan Array)

```
public class Stack {
    private String[] stack;
    private int max;
    private int top;

    public Stack(int size) {
        max = size;
        stack = new String[max];
        top = -1;
    }

    public void push(String value) {
        stack[++top] = value;
    }

    public String pop() {
        return stack[top--];
    }

    public String peek() {
        return stack[top];
    }

    public boolean isEmpty() {
        return top == -1;
    }

    public boolean isFull() {
        return top == max - 1;
    }

    public void tampilStack() {
        for (int i = 0; i <= top; i++) {
            System.out.println(stack[i]);
        }
    }

    public static void main(String[] args) {
        Stack s = new Stack(5);

        System.out.println("Berikut ini adalah contoh program stack menggunakan
array dengan jumlah indeks 5:");
```

```
        System.out.println("-- push() Fisika, Kimia, Matematika");
        s.push("Fisika");
        s.push("Kimia");
        s.push("Matematika");
        s.tampilStack();
        System.out.println();

        System.out.println("-- peek()");
        System.out.println(s.peek() + "\n"); // return Matematika

        System.out.println("--- push() B. Indonesia, Biologi");
        s.push("B. Indonesia");
        s.push("Biologi");
        s.tampilStack();
        System.out.println();

        System.out.println("-- isEmpty()");
        System.out.println(s.isEmpty() + "\n"); // false

        System.out.println("-- isFull()");
        System.out.println(s.isFull() + "\n"); // true

        System.out.println("-- pop()");
        System.out.println(s.pop() + "\n"); // remove Biologi

        System.out.println("-- stack sekarang ini berisi:");
        s.tampilStack();
        System.out.println();

        System.out.println("-- isFull()");
        System.out.println(s.isFull() + "\n"); // false
    }
}
```

Implementasi Stack menggunakan Array vs Linked List



- ✧ Array merupakan struktur data yang statis, sehingga sebelum array dibuat kita harus menentukan sizenya (program akan mengalokasikan memori dan ruang sejumlah yang kita pesan).
- ✧ Permasalahannya adalah stack tidak memiliki batasan sebanyak apa jumlah elemennya, sehingga stack dapat berukuran makin besar. Saat elemen pada stack memenuhi batasan, maka tidak akan ada ruang lagi sehingga menyebabkan **stack overflow.**
- ✧ LinkedList merupakan struktur data dinamis, kita dapat menambah ukurannya sehingga memori dapat disimpan dimana saja, karena LinkedList memiliki node yang menunjuk alamat memori. Stack overflow pada LinkedList jarang terjadi, biasanya terjadi karena memori dari komputer yang habis.