

LAPORAN PRAKTIKUM STRUKTUR DATA
PENERAPAN STACK DAN IMPLEMENTASINYA



OLEH :

JOVANTRI IMMANUEL GULO

NIM 2411532014

MATA KULIAH STRUKTUR DATA

DOSEN PENGAMPU :

Dr. Ir. Wahyudi, S.T, M.T

FAKULTAS TEKNOLOGI INFORMASI

DEPARTEMEN INFORMATIKA

UNIVERSITAS ANDALAS

PADANG, 6 MEI 2025

A. Pendahuluan

Struktur data merupakan cara untuk menyimpan dan mengatur data agar dapat digunakan secara efisien. Salah satu struktur data linear yang penting dan sering digunakan dalam pemrograman adalah Stack. Stack adalah struktur data yang mengikuti prinsip LIFO (Last In, First Out), artinya elemen terakhir yang dimasukkan akan menjadi elemen pertama yang dikeluarkan. Analoginya mirip seperti tumpukan piring: piring terakhir yang ditumpuk akan menjadi piring pertama yang diambil.

Operasi dasar yang dapat dilakukan pada stack antara lain:

- `push(e)`: Menambahkan elemen `e` ke atas stack.
- `pop()`: Menghapus dan mengembalikan elemen teratas dari stack.
- `top()` atau `peek()`: Mengakses elemen teratas tanpa menghapusnya.
- `isEmpty()`: Memeriksa apakah stack kosong.
- `size()`: Mengembalikan jumlah elemen dalam stack.

Dalam implementasinya, stack dapat dibangun menggunakan array atau struktur data dinamis seperti linked list. Bahasa pemrograman Java menyediakan class Stack bawaan, namun kita juga bisa membuat implementasi sendiri seperti pada ArrayStack untuk memahami lebih dalam prinsip kerja stack.

B. Tujuan

Tujuan dari dilakukannya praktikum ini adalah

1. Mampu menerapkan stack dalam beberapa program.
2. Mampu mengimplementasikan penggunaan stack pada interface ataupun konstruktor.

C. Langkah – langkah Pengerjaan

Berikut adalah langkah-langkah dalam pengerjaan praktikum kali ini:

1. Pertama, buat class dengan nama **latihanStack**

```
> latihanStack.java
```

2. Lalu, import **stack** terlebih dahulu dari java util

```
3 import java.util.Stack;
```

3. Kemudian buat kelas main untuk latihanStack

```
5 public class latihanStack {  
6     public static void main(String[] args) {
```

4. Deklarasikan stack dengan tipe data integer dengan nama **s**.

```
7         Stack<Integer> s = new Stack<Integer>();
```

5. Lakukan push pada stack dengan menggunakan **s.push**

```
8         s.push(42);  
9         s.push(-3);  
10        s.push(17);
```

6. Kemudian, kita print nilai stack pada awal, nilai stack pada saat dipop, nilai stack yang teratas, dan nilai stack setelah di peek.

```
11        System.out.println("Nilai stack = "+s);  
12        System.out.println("Nilai pop = "+s.pop());  
13        System.out.println("Nilai peek = "+s.peek());  
14        System.out.println("Nilai stack setelah peek = "+s);
```

7. Nantinya, 42 akan menjadi yang terbawah, -3 di tengah, dan 17 di atas. Saat dilakukan pop, nilai yang keluar adalah nilai **17** karena paling atas, lalu saat dilakukan peek, maka yang ter-peek adalah nilai -3, dikarenakan -3 adalah nilai teratas setelah 17 di-pop.
8. Hasilnya adalah sebagai berikut

9. Kemudian, kita membuat class yang baru lagi dengan nama **contohStack**

> contohStack.java

10. Lalu buat class dan class mainnya

```
3 public class contohStack {  
4     public static void main(String[] args) {
```

11. Implementasikan **ArrayStack** sebagai metode yang akan kita gunakan nantinya

```
    ArrayStack test = new ArrayStack();
```

12. Lalu, kita implementasikan array dan memuat beberapa data.

```
    Integer[] a = {4, 8, 15, 16, 23, 42};
```

13. Setelah itu, kita akan menginisialisasi masing-masing nilai dari stack dengan menginisiasinya dari A0 hingga sejumlah dengan jumlah stack yang ada dan juga mengambil isi dari array per indexnya dan dipush ke dalam stack **test**.

```
7         for(int i = 0; i < a.length; i++) {  
8             System.out.println("Nilai A"+i+" = "+ a[i]);  
9             test.push(a[i]);  
10        }
```

14. Kemudian, kita deklarasi berapa sih besar stacknya.

```
    System.out.println("Size stacknya: "+test.size());
```

15. Dan juga kita deklarasi apa elemen yang paling atas dari stack tersebut.

```
    System.out.println("Paling atas: "+test.top());
```

16. Lalu juga, terakhir, kita deklarasi nilai yang kita pop.

```
    System.out.println("Nilainya: "+test.pop());
```

17. Nantinya, program ini akan mengambil isi dari array **a** dan akan memasukkannya ke dalam stack, sehingga stack **test** dapat diadaptasikan dan diisikan dari sebuah array.

18. Sebelum itu, untuk mendeklarasikan **ArrayStack**, kita perlu buat interface class dengan nama **Stack2** dan class baru dengan nama **ArrayStack**.

19. Kita fokus ke **Stack2** terlebih dahulu, ini merupakan sebuah interface, bukan class.

> Stack2.java

20. Lalu, kita import packagenya pada isi file Stack2.

```
1 package pekan3;
```

21. Deklarasikan interface Stack2.


```
3 public interface Stack2<E> {
```

22. Isi dari interface Stack2 tersebut adalah sebagai berikut.

```
4    int size();
5    boolean isEmpty();
6    void push (E e);
7    E top();
8    E pop();
```

Memuat integer size (ukuran dari stack), Boolean isEmpty (mengecek apakah stack kosong), melakukan push untuk tipe data, melihat posisi elemen teratas, dan mengintip elemen posisi teratas.

23. Lalu, kita masuk ke file **ArrayStack** dengan membuat class baru.

>  **ArrayStack.java**

24. Kita deklarasikan packagenya dan juga public class yang digunakan, yaitu public class ArrayStack dengan implementasi dari Stack2.

```
1    package pekan3;
2
3    public class ArrayStack<E> implements Stack2<E> {
```

25. Kita deklarasikan beberapa variable dengan karakteristiknya masing-masing, seperti identifier Capacity dengan keyword final dan tipe data int, lalu array data dengan keyword private, dan integer t dengan keyword private dan bernilai "-1".

```
5    public static final int CAPACITY = 1000;
6
7    private E[] data;
8
9    private int t = -1;
```

26. Kita deklarasikan konstruktor, yaitu membuat ArrayStack dengan kapasitas final sebanyak 1000.

```
11    public ArrayStack() {
12        this(CAPACITY);
13    }
```

27. Dan deklarasi konstruktor untuk kapasitas custom dari ArrayStack, sehingga pengguna bebas dalam menentukan berapa kapasitas dari ArrayStack yang mereka inginkan.

```
15    public ArrayStack(int capacity) {
16        data = (E[]) new Object[capacity];
17    }
```

28. Mengembalikan jumlah elemen yang terdapat di dalam stack

```
19    public int size() {
20        return (t + 1);
21    }
```

29. Memeriksa apakah stack kosong atau tidak, apakah tetap sama dengan nilai awal yaitu "-1"

```
23    public boolean isEmpty() {
24        return (t == -1);
25    }
```

30. Menambahkan elemen ke atas stack dengan memeriksa apakah stack sudah penuh atau belum, apabila belum akan menambahkan element ke data[t+1] dengan menaikkan t(++t), apabila sudah maka throw IllegalStateException.

```

27 public void push(E e) throws IllegalStateException {
28     if(size() == data.length) throw new IllegalStateException("Stack is full");
29     data[++t] = e;
30 }

```

31. Mengembalikan elemen paling atas tanpa menghapusnya, sama seperti fungsi seek pada stack, lalu juga apabila kosong, akan di-return **null**.

```

32 public E top() {
33     if(isEmpty())
34         return null;
35     return data[t];
36 }

```

32. Menghapus sekaligus mengembalikan elemen paling atas, ambil nilai di data[t], set data[t] jadi null untuk menghindari kebocoran data, menurunkan nilai t, lalu mengembalikan nilai yang diambil.

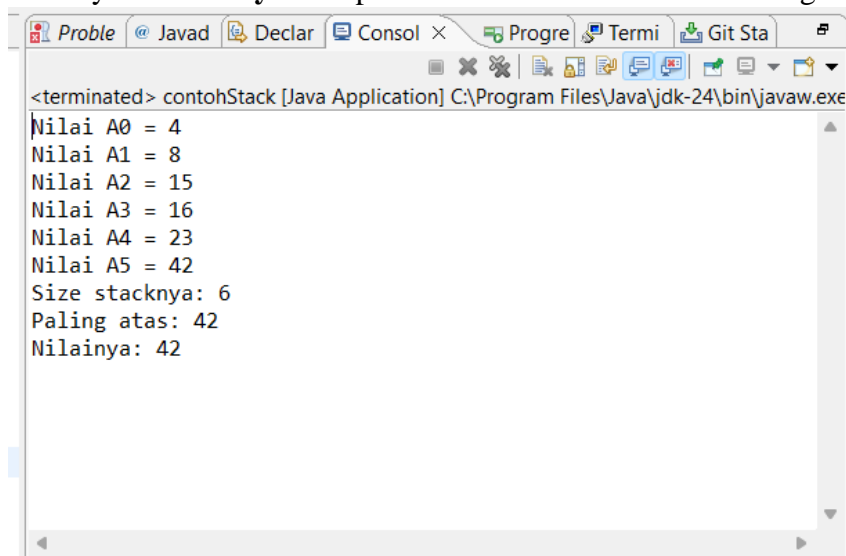
```

38 public E pop() {
39     if(isEmpty())
40         return null;
41     E answer = data[t];
42     data[t] = null;
43     t--;
44     return answer;
45 } }

```

33. Dengan telah dibuatnya kelas interface Stack2, class ArrayStack, maka ArrayStack dapat digunakan di file manapun.

34. Hasilnya dari **ArrayStack** pada file **contohStack** adalah sebagai berikut.



```

<terminated> contohStack [Java Application] C:\Program Files\Java\jdk-24\bin\javaw.exe
Nilai A0 = 4
Nilai A1 = 8
Nilai A2 = 15
Nilai A3 = 16
Nilai A4 = 23
Nilai A5 = 42
Size stacknya: 6
Paling atas: 42
Nilainya: 42

```

35. Kita buat project baru, dengan class baru yaitu **NilaiMaksimum**.

```
> NilaiMaksimum.java
```

36. Kita import packagenya terlebih dahulu.

```
1 package pekan3;
```

37. Kita import juga library yang akan kita gunakan, yaitu Stack.

```
3 import java.util.Stack;
```

38. Kita deklarasikan kelas NilaiMaksimum dan method max untuk mencari nilai maksimum dari Stack<Integer> s.

```

5 public class NilaiMaksimum {
6
7     public static int max(Stack<Integer> s) {

```

39. Mendeklarasikan stack dengan tipe data integer dan identifiernya adalah **backup**.

```

8         Stack<Integer> backup = new Stack<Integer>();

```

40. Stack **backup** membantu dalam menyimpan hasil pop dari stack s untuk dapat nantinya dikembalikan lagi (tidak hilang).

```

9             int maxValue = s.pop();
10            backup.push(maxValue);

```

41. Selama s stack belum kosong (is not empty), maka lakukan lagi perbandingan antara value yang dipop (teratas) dan juga dengan value teratas yang tadinya telah disimpan di stack **backup**, lalu mengupdate nilai maxValue dengan nilai max hasil perbandingan dua variable.

```

11            while(!s.isEmpty()) {
12                int next = s.pop();
13                backup.push(next);
14                maxValue = Math.max(maxValue, next);
15            }

```

42. Mengembalikan semua elemen dari *hasil pop s di dalam stack backup*, lalu dikembalikan kembali pada stack s dengan **s.push(backup.pop());**

```

16            while (!backup.isEmpty()) {
17                s.push(backup.pop());
18            }

```

43. Mengembalikan nilai maximum yang telah ditemukan.

```

19            return maxValue;
20        }

```

44. Kita masuk ke method main

```

21    public static void main(String[] args) {

```

45. Deklarasikan stack s yang nantinya akan kita gunakan

```

22        Stack<Integer> s = new Stack<Integer>();

```

46. Lakukan push pada stack s dan simpan integer di dalamnya dengan menggunakan **s.push**

```

23            s.push(70);
24            s.push(12);
25            s.push(20);

```

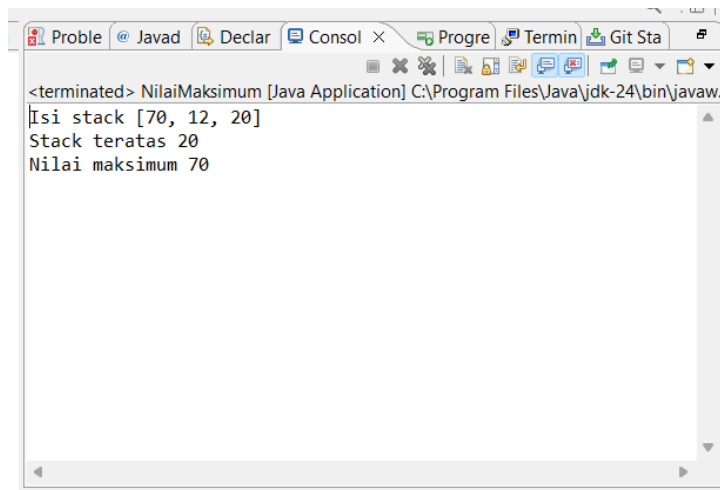
47. Melihat isi stack dengan s, melihat isi stack teratas dengan **s.peek()**, dan menampilkan nilai maksimum dengan **max(s)**

```

26            System.out.println("Isi stack "+s);
27            System.out.println("Stack teratas "+s.peek());
28            System.out.println("Nilai maksimum "+max(s));
29        }
30    }


```

48. Berikut adalah hasil dari programnya.



```
<terminated> NilaiMaksimum [Java Application] C:\Program Files\Java\jdk-24\bin\javaw.  
[Isi stack [70, 12, 20]  
Stack teratas 20  
Nilai maksimum 70
```

49. Kita beralih ke program baru dengan nama **StackPostfix**

>  StackPostfix.java

50. Kita deklarasikan package yang kita gunakan

```
1 package pekan3;
```

51. Kita deklarasikan library yang akan kita gunakan

```
3 import java.util.Scanner;  
4 import java.util.Stack;
```

52. Kita deklarasikan public class StackPostfix dan juga method public static int postfixEvaluate

```
6 public class StackPostfix {  
7  
8     public static int postfixEvaluate(String expression) {
```

53. Kita deklarasikan stack dengan nama s dan scanner dengan nama **input**

```
9         Stack<Integer> s = new Stack<Integer>();  
10        Scanner input = new Scanner(expression);
```

54. Kita buat kondisi **while**

```
    while(input.hasNext()) {
```

55. Ketika input punya angka, maka kita akan push ke dalam stack s tadi

```
        if(input.hasNextInt()) {  
            s.push(input.nextInt());
```

56. Apabila berupa operator, maka kita akan mengambil 2 angka dari stack s dan melakukan operasi dan mengembalikannya ke dalam stack dengan **push**.

```
14        } else {  
15            String operator = input.next();  
16            int operand2 = s.pop();  
17            int operand1 = s.pop();  
18            if(operator.equals("+")) {  
19                s.push(operand1 + operand2);
```

57. Berikut adalah operator untuk +, -, *, /.


```

18         if(operator.equals("+")) {
19             s.push(operand1 + operand2);
20         } else if (operator.equals("-")) {
21             s.push(operand1 - operand2);
22         } else if (operator.equals("*")) {
23             s.push(operand1 * operand2);
24         } else {
25             s.push(operand1 / operand2);
26         }

```

58. Hasil akhir akan menjadi satu-satunya elemen yang tersisa di stack.

```

29         return s.pop();
30     }

```

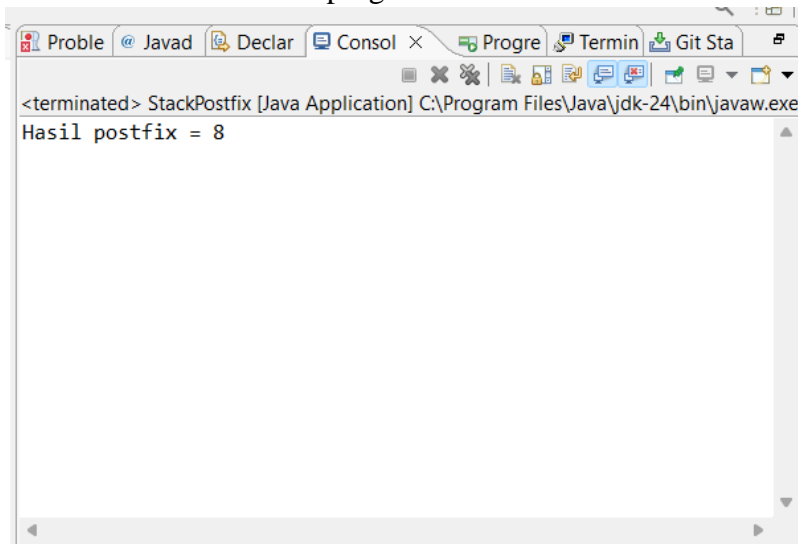
59. Lalu, yang terakhir, kita buat kelas main dan kita jalankan fungsi dari method **postfixEvaluate**.

```

32     public static void main(String[] args) {
33         System.out.println("Hasil postfix = "+postfixEvaluate("5 2 5 * + 7 -"));
34     }

```

60. Berikut adalah hasil dari program tersebut.



D. Kesimpulan

Dari praktikum yang telah dilakukan, maka dapat diambil kesimpulan bahwa implementasi stack dan penggunaannya dalam pemrograman sangatlah marak dijumpai dikarenakan fungsi daripada stack ini sangat berdampak pada struktur data. Pengelolaan struktur data dengan menggunakan stack, akan memudahkan pengembang dalam mengembangkan programnya, dikarenakan stack dapat digunakan untuk melakukan pengelolaan struktur data seperti menyimpan, mengeluarkan, dan metode-metode yang membantu lainnya.