

LAPORAN PRAKTIKUM STRUKTUR DATA
IMPLEMENTASI STACK PADA PEMROGRAMAN JAVA



Oleh :

DERIEL CHAERAHMAN

NIM 2411533007

DOSEN PENGAMPU : DR. WAHYUDI, S.T, M.T

ASISTEN PRAKTIKUM : RAHMAT DWIRIZKI OLDERS

FAKULTAS TEKNOLOGI INFORMASI

DEPARTEMEN INFORMATIKA

UNIVERSITAS ANDALAS

2025

A. Pendahuluan

Praktikum ini dilakukan untuk membuat program yang dapat menyimpan/mengelola data dengan implementasi struktur data pada java yaitu Stack. Mengimplementasikan dengan program java yang sudah dipelajari sebelumnya seperti if statement, looping, scanner class dan lainnya dalam pemrograman Java.

1. Stack (Tumpukan)

Merupakan struktur data linear yang mengikuti aturan tertentu, yaitu LIFO (Last In First Out) atau FILO (First In Last Out), analoginya berupa tumpukan buku dimana buku terbaru diletakan paling atas pada tumpukan. Jenisnya yaitu stack ukuran tetap dan dinamis. Dapat diimplementasikan menggunakan array ataupun LinkedList. Dapat digunakan untuk mengimplementasikan mekanisme 'pembatalan' untuk kembali ke kondisi sebelumnya/untuk membuat algoritma depth-first search dalam graf untuk backtracking. Kelebihannya yaitu implementasi linkedList dari sebuah stack dapat tumbuh dan menyusut sesuai kebutuhan saat runtime dan banyak digunakan pada JVM. Kekurangannya yaitu membutuhkan memori ekstra dan tidak mendukung pengaksesan secara acak.

Method :

- Push() : Menambahkan elemen baru pada tumpukan
- Pop() : Menghapus/ mengembalikan elemen teratas dari tumpukan
- Peek() : Mengembalikan elemen teratas pada stack, seperti get().
- isEmpty() : Memeriksa apakah tumpukan kosong
- Size() : Menemukan jumlah elemen dalam tumpukan

B. Tujuan

Tujuan dari dilakukannya praktikum ini adalah :

1. Memahami dan mengaplikasikan Stack dalam program java untuk menyimpan data.
2. Implementasi method Stack pada program java.
3. Dapat mengaplikasikan statement (if, while), class (Scanner), dll. yang dipelajari sebelumnya ke dalam program.
4. Membuat program struktur data dengan metode OOP (Object Oriented Programming).

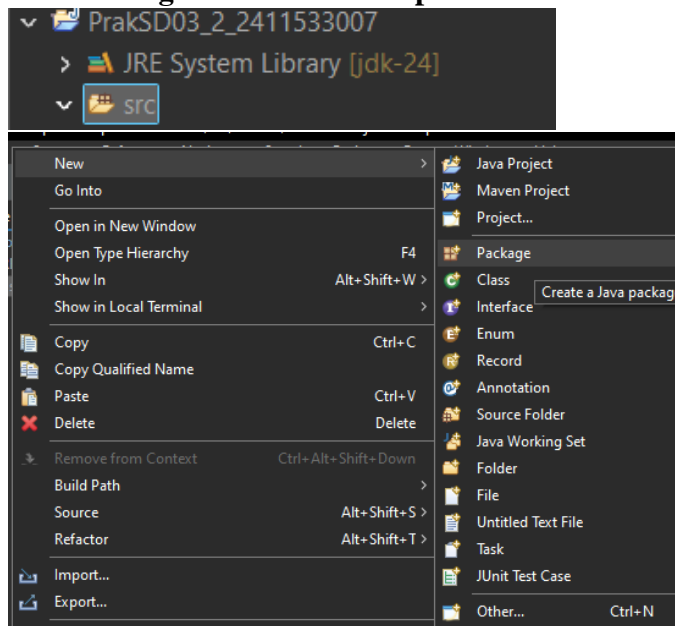
C. Langkah kerja praktikum

a. Alat dan Bahan

1. Perangkat computer atau laptop
2. Jaringan internet
3. IDE (Integrated Development Environment) direkomendasikan Eclipse IDE
4. Java JDK (Java Development Kit)

b. Package pekan 3

1. Buat new package. **Buka java project** yang telah dibuat sebelumnya, lalu **klik kanan** pada folder 'src', setelahnya akan muncul list option, pilih 'new', lalu 'Package' dan beri nama 'pekan3'.



c. Program latihanStack

1. **Buat** terlebih dahulu new **class** dengan klik kanan pada package pekan2 dan beri nama 'latihanStack', centang method public static void.
2. **Import** Stack class dari paket **Java.util.Stack** terlebih dahulu untuk menggunakan fungsi/method dari classnya.

```
1 package pekan3;  
2 import java.util.Stack;  
3  
4 public class latihanstack {
```

3. **Deklarasi object Stack** bertipe **Integer** dengan nama 's'.

```
6 public static void main(String[] args) {  
7     Stack<Integer> s = new Stack<Integer>();
```

4. Gunakan **method push()** dari kelas Stack **untuk menambahkan data** ke dalam Stack.

```
8     s.push(43);  
9     s.push(-3);  
10    s.push(17);
```

5. **Tampilkan data** dari Stack menggunakan sysout. Menggunakan **method pop()** yang akan menghapus data paling atas (Data terakhir yang input), method **peek()** untuk mengembalikan nilai dari data paling atas tumpukan saat ini.

```

11     System.out.println("Nilai stack = " + s);
12     System.out.println("Nilai pop = " + s.pop());
13     System.out.println("Nilai stack setelah pop = " + s);
14     System.out.println("Nilai peek = " + s.peek());
15     System.out.println("Nilai stack setelah peek = " + s);
16 }
17
18 }

```

6. Output program. Menampilkan data yang tersimpan pada Stack, yang sebelumnya diinputkan menggunakan method push(). Data awal yang diinputkan akan berada pada tumpukan terbawa (43), dan data paling terakhir diinputkan akan berada di paling atas tumpukan (17), jadi ketika method pop() akan menghapus data paling atas yaitu 17, lalu method peek yang akan mengembalikan data paling atas tumpukan yaitu -3. Jadi data yang tersimpan pada stack hanya tersisa 43 dan -3.

```

Console x Eclipse IDE for Java and DSL Devel...
<terminated> latihanstack [Java Application] C:\Program Files\Ja
Nilai stack = [43, -3, 17]
Nilai pop = 17
Nilai stack setelah pop = [43, -3]
Nilai peek = -3
Nilai stack setelah peek = [43, -3]

```

d. Program contohStack

1. **Buat new class** dengan klik kanan pada package pekan2 dan beri nama 'contohStack', centang method public static void.
2. **Import** Stack dari package **Java.util.** untuk menggunakan fungsi/method dari classnya.

```

1 package pekan3;
2 import java.util.Stack;
3
4 public class contohStack {

```

3. Deklarasi object ArrayStack dengan nama "test".

```

6 public static void main(String[] args) {
7     ArrayStack test = new ArrayStack();

```

4. Inisialisasi data pada array biasa bertipe integer dengan nama "a".

```

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

```

5. Buat for loop, dengan nilai awal 0 sampai kurang dari panjang(length) array "a", dan iterasikan dengan 'tambah satu'. Pada blok kodenya akan menampilkan data(sysout) dari array "a" berdasarkan nilai iterasi perulangan dan sesuai index dari array. Lalu nilai dari array "a" (sesuai indexnya) akan di **push()** ke object dari arrayStack "test".

```

9     for (int i = 0; i < a.length; i++) {
10         System.out.println("Nilai A " + i + " = " + a[i]);
11         test.push(a[i]);
12     }

```

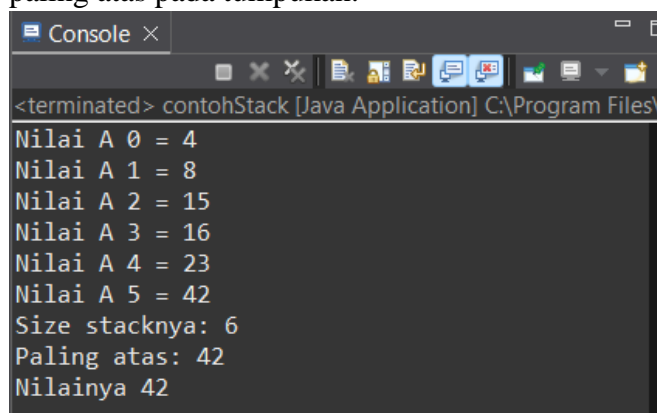
6. Menampilkan data dari arrayStack "test" menggunakan sysout.

```

13     System.out.println("Size stacknya: " + test.size());
14     System.out.println("Paling atas: " + test.top());
15     System.out.println("Nilainya " + test.pop());
16 }

```

- Output program (Sebelumnya terlebih dahulu perlu membuat program interface stack2 dan program class arrayStack). Program menampilkan data dari array “a” menggunakan for loop blok kode dengan kondisi bernilai true dan mencetaknya dengan string “Nilai A 0 = (nilai sesuai index pada array)”. Melakukan method **push()** yang akan mengirim **data** dari for loop ke object dari **arrayStack**. Menampilkan isi dari arrayStack menggunakan sysout dan method nya yaitu, method **size()** untuk melihat banyaknya data yang tersimpan, method **top()** untuk melihat data paling atas pada tumpukan, dan method **pop()** untuk mengeluarkan data paling atas pada tumpukan.



```

<terminated> contohStack [Java Application] C:\Program Files\
Nilai A 0 = 4
Nilai A 1 = 8
Nilai A 2 = 15
Nilai A 3 = 16
Nilai A 4 = 23
Nilai A 5 = 42
Size stacknya: 6
Paling atas: 42
Nilainya 42

```

e. Program interface stack2

- Program interface stack2. Merupakan class generic “<E>”, jadi bisa menggunakan berbagai tipe data (String, Integer, dll).

```

1 package pekan3;
2
3 public interface stack2<E> {

```

- Isi dari interface stack2 sebagai berikut. Deklarasikan integer size (ukuran dari stack), boolean isEmpty(mengecek apakah stack kosong), method push data, melihat posisi element teratas (top) dan menghapus elemen teratas(pop).

```

4     int size();           // Mengembalikan jumlah elemen dalam stack
5     boolean isEmpty();    // Mengecek apakah stack kosong
6     void push(E e);       // Menambahkan elemen ke puncak stack
7     E top();              // Mengambil elemen teratas (tanpa menghapus)
8     E pop();              // Menghapus dan mengembalikan elemen teratas
9 }

```

- Penjelasan : Semua class yang mengimplementasikan interface ini wajib menyediakan kode untuk semua method-nya.

f. Program ArrayStack

- Deklarasi public class menggunakan implementasi dari interface stack2

```

1 package pekan3;
2
3 public class ArrayStack<E> implements stack2<E> {

```

2. Deklarasikan atribut dan konstruktor, **CAPACITY** merupakan kapasitas maksimum stack yaitu 1000 (elemen), **data** berfungsi sebagai array yang menyimpan elemen stack dan **int t** berfungsi sebagai Index dari elemen paling atas (top). -1 artinya stack kosong.

```
4    public static final int CAPACITY = 1000;
5        //default array capacity
6    private E[] data;    //generic array used for storage
7    private int t = -1;
```

3. Konstruktor default, membuat arrayStack dengan kapasitas default 1000 elemen.

```
9    public ArrayStack() {
10        this(CAPACITY);
11    }    //constructs stack with default capacity
```

4. Deklarasikan konstruktor untuk kapasitas khusus dari arrayStack, jadi user bebas menentukan kapasitas arrayStack yang diinginkan.

```
13    public ArrayStack(int capacity) {
14        //constructs stack with given capacity
15        data = (E[]) new Object[capacity];
16    }
```

5. Implementasi method dari interface, ukuran stack/ size(), karena index dimulai dari 0 (t = -1 pada deklarasi atribut awal), maka untuk menghitung jumlah pasti dari elemen tersimpan saat pengembalian(getter) maka : t + 1.

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

6. Memeriksa apakah stack kosong / isEmpty() bertipe boolean, "t == -1" yang artinya apakah isi dari stack tetap sama dengan nilai awal yaitu "t = -1".

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

7. Method yang menambahkan elemen ke atas Stack, dengan memeriksa apakah stack sudah penuh = atau belum, jika belum akan menambahkan elemen ke data [t++], dan apabila sudah penuh maka throw IllegalStateException.

```
26    public void push(E e) throws IllegalStateException {
27        if (size() == data.length)
28            throw new
29                IllegalStateException ("Stack is full");
30        data[++t] = e;    //increment before storing new item
31    }
```

8. Mengembalikan elemen paling atas tumpukan, jika kosong akan di return null.

```
33    public E top() {
34        if (isEmpty())
35            return null;
36        return data[t];
37    }
```

9. Menghapus sekaligus mengembalikan elemen paling atas tumpukan saat ini, mengambil nilai data [t], lalu mengembalikan nilai yang diambil.

```

39 public E pop() {
40     if (isEmpty())
41         return null;
42     E answer = data[t];
43     data[t] = null;
44     t--;
45     return answer;
46 }
47 }

```

10. Penjelasan program : Merupakan implementas manual Stack menggunakan array generik "<E>". Bertujuan menyimpan data menggunakan konsep LIFO/ tumpukan, menyediakan operasi dasar/method dari class Stack dan dapat digunakan untuk berbagai tipe data melalui parameter generik <E>. Cocok untuk aplikasi berupa undo/redo, penelusuran rekursif(DFS), navigasi halaman browser, dll.

g. Program NilaiMaksimum

1. **Buat new class** dengan klik kanan pada package pekan2 dan beri nama 'NilaiMaksimum', centang method public static void.
2. **Import** Stack dari package **Java.util.** untuk menggunakan fungsi/method dari classnya dan deklarasi **class NilaiMaksimum.**

```

1 package pekan3;
2 import java.util.Stack;
3
4 public class NilaiMaksimum {

```

3. Deklarasikan **method** static, berfungsi untuk mencari nilai maksimum integer tanpa menghapus data dari stack asli secara permanen. Mengembalikan nilai bertipe integer.

```

6 public static int max(Stack<Integer> s) {

```

4. Deklarasi stack dengan tipe integer dengan nama "backup". Menyiapkan 'backup' stack sementara, untuk menyimpan elemen saat proses pengambilan elemen dari "s".

```

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

```

5. Stack backup membantu dalam menyimpan hasil pop dari stack "s" yang nantinya dapat dikembalikan lagi (tidak dihapus permanen).

```

8         int maxVal = s.pop();
9         backup.push(maxVal);

```

6. Menggunakan while loop, untuk memeriksa selama "s" stack tidak kosong, lakukan perbandingan antara value yang di pop() dan dengan value teratas yang sebelumnya sudah di push ke "backup", lalu mengupdate nilai maxVal menjadi nilai max hasil perbandingan dua variabel.

```

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

```

7. Menggunakan while loop, untuk mengembalikan semua elemen dari hasil pop() stack "s" yang tersimpan pada stack **backup** (sebelumnya di push ke sini), lalu dikembalikan kembali pada stack "s".

```
17     while (!backup.isEmpty()) {
18         s.push(backup.pop());
19     }
```

8. Mengembalikan nilai maksimum yang telah ditemukan dari backup.

```
20
21     return maxValue;
22 }
```

9. Masuk ke method main, deklarasikan Stack bertipe integer dengan nama "s".

```
24 public static void main(String[] args) {
25     Stack<Integer> s = new Stack<>();
```

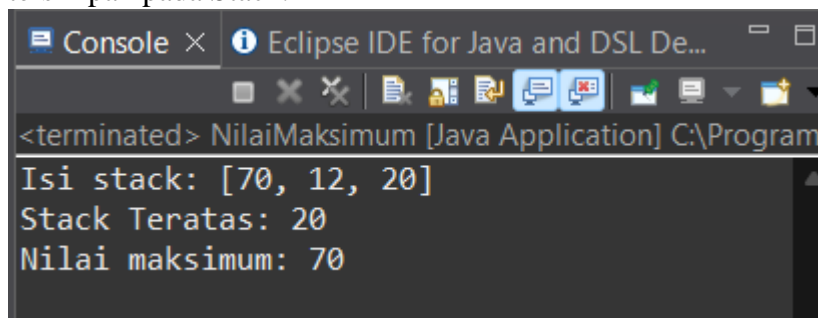
10. Lakukan method push(), untuk memasukkan data ke Stack "s".

```
26     s.push(70);
27     s.push(12);
28     s.push(20);
```

11. Menampilkan data tersimpan pada Stack "s", menggunakan method peek() untuk melihat elemen teratas pada Stack, dan method max(s) untuk menampilkan nilai maksimum dari elemen tersimpan pada Stack "s".

```
29
30     System.out.println("Isi stack: " + s);
31     System.out.println("Stack Teratas: " + s.peek());
32     System.out.println("Nilai maksimum: " + max(s));
33 }
34 }
```

12. Output program dan alur kerjanya : Menyimpan sejumlah bilangan ke dalam struktur data Stack bertipe integer, lalu mencari nilai maksimum (menggunakan fungsi max()) dari stack tanpa merusak listnya, lalu menampilkan isi stack, elemen teratas(peek) dan nilai maksimum yang tersimpan pada Stack.



```
<terminated> NilaiMaksimum [Java Application] C:\Program
Isi stack: [70, 12, 20]
Stack Teratas: 20
Nilai maksimum: 70
```

h. Program StackPostfix

1. **Buat** new **class** dengan klik kanan pada package pekan2 dan beri nama 'StackPostfix', centang method public static void
2. **Import** Stack class dan Scanner class dari paket **Java.util.** terlebih dahulu untuk menggunakan fungsi/method dari classnya.


```

1 package pekan3;
2 import java.util.Scanner;
3 import java.util.Stack;
4
5 public class StackPostfix {

```

3. Deklarasikan method **postfixEvaluate** dengan parameter **String expression** berfungsi sebagai ekspresi matematika dalam bentuk postfix yang dipisahkan dengan spasi. Deklarasikan object **Stack** dengan tipe integer dengan nama "s" dan object **Scanner** identifier "input" yang akan membaca input dari **String expression**, token dalam string akan dibaca **satu per satu**, dipisahkan oleh spasi.

```

7 public static int postfixEvaluate(String expression) {
8     Stack<Integer> s = new Stack<>();
9     Scanner input = new Scanner(expression);

```

4. While loop dengan kondisi menerima input dari user dengan method **hasNext()** yang akan mengembalikan nilai boolean.

```

11 while (input.hasNext()) {

```

5. If statement dengan kondisi jika input terdapat angka, maka akan dipush ke stack "s", dan disimpan sebagai operand.

```

12     if (input.hasNextInt()) {
13         s.push(input.nextInt());

```

6. Else statement ketika if kondisi bernilai false, akan mengeksekusi blok kode berikut, apabila berupa operator, maka akan diambil 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();

```

7. Switch dengan kondisi berdasarkan hasil input user yang tersimpan pada variabel operator sebelumnya, menyesuaikan case operator berdasarkan inputan user.

```

19         switch (operator) {
20             case "+":
21                 s.push(operand1 + operand2);
22                 break;
23             case "-":
24                 s.push(operand1 - operand2);
25                 break;
26             case "*":
27                 s.push(operand1 * operand2);
28                 break;
29             case "/":
30                 s.push(operand1 / operand2);
31                 break;
32         }
33     }
34 }

```

8. Hasil akhir akan menjadi elemen yang tersisa di Stack.

```

35
36     return s.pop();
37 }

```

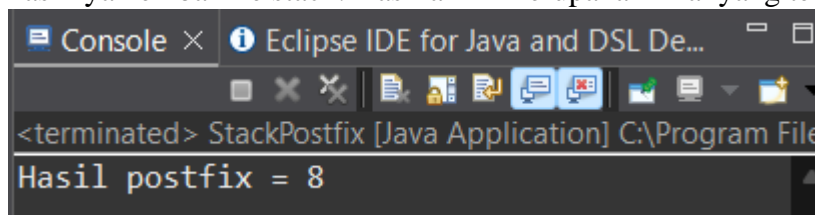
9. Buat main method, jalankan fungsi dari method **StackPostfix** dan tampilkan pada sysout.

```

39 public static void main(String[] args) {
40     System.out.println("Hasil postfix = " + postfixEvaluate("5 2 5 * + 7 -"));
41 }
42 }

```

10. Output program : Program membaca ekspresi postfix(dipisah spasi), lalu menyimpan angka ke stack, lalu menentukan operator dengan mengambil dua operand dari stack, lalu melakukan operasi aritmatika dan menyimpan hasilnya kembali ke stack. Hasil akhir merupakan nilai yang tersisa di stack.



The screenshot shows the Eclipse IDE console window. The title bar reads 'Console x Eclipse IDE for Java and DSL De...'. The console output shows the text '<terminated> StackPostfix [Java Application] C:\Program File' followed by the result 'Hasil postfix = 8'.

D. Kesimpulan

Setelah melakukan praktikum ini dapat memahami dan mengimplementasikan Stack (tumpukan) dan methodnya. Membuat program yang lebih kompleks dan interaktif dengan konsep OOP berupa program ArrayStack. Menerapkan stack untuk menyelesaikan permasalahan nyata, seperti evaluasi ekspresi matematika postfix. Menggunakan while loop dalam method main untuk membuat program lebih interaktif. Stack dapat diimplementasikan untuk mengelola struktur data dengan baik berupa menyimpan data, dan dapat menyimpan dengan baik proses undo/redo.