

LAPORAN PRATIKUM

“PEKAN 3”

Disusun Untuk Memenuhi Tugas Mata Kuliah Struktur Data

DOSEN PENGAMPU:

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



DISUSUN OLEH:

Karimah Irsyadiyah (2411533018)

UNIVERSITAS ANDALAS

T.A 2024/2025

Daftar Pustaka

BAB I PENDAHULUAN.....	3
1.1 Latar Belakang.....	3
1.2 Tujuan	3
1.3 Alat dan Bahan.....	3
BAB II PEMBAHASAN	4
2.1 Langkah-langkah Praktikum dan Pembahasan Program	4
BAB III PENUTUP	12
3.1 Kesimpulan	12

BAB I

PENDAHULUAN

1.1 Latar Belakang

Struktur data merupakan fondasi penting dalam pemrograman, salah satunya adalah stack. Stack adalah struktur data linear yang menggunakan prinsip LIFO (Last In First Out), di mana elemen terakhir yang dimasukkan akan menjadi elemen pertama yang dikeluarkan. Dalam praktikum ini, kita mempelajari implementasi stack menggunakan Java, baik menggunakan class bawaan Java (`java.util.Stack`) maupun membuat implementasi sendiri menggunakan array.

1.2 Tujuan

- Memahami konsep dasar dan prinsip kerja stack.
- Mengimplementasikan stack menggunakan array (`ArrayStack`).
- Menggunakan stack dalam berbagai studi kasus seperti menghitung ekspresi postfix dan mencari nilai maksimum.

1.3 Alat dan Bahan

- Perangkat Keras: Laptop/PC
- Perangkat komputer dengan IDE (contoh: IntelliJ IDEA / Eclipse).
- Java Development Kit (JDK).
- Library Java `java.util.Stack`.
- Source code: `ArrayStack.java`, `Stack2.java`, `contohStack.java`, `contohStack2.java`, `latihanStack.java`, `NilaiMaksimum.java`, `StackPostfix.java`.

BAB II

PEMBAHASAN

2.1 Langkah-langkah Praktikum dan Pembahasan Program

A. Class 1: Stack2.java (Interface Stack)

Pada praktikum ini, kita mempelajari enam buah program yang memanfaatkan ArrayList sebagai struktur data utama. Setiap file memiliki tujuan spesifik untuk menunjukkan cara kerja dan implementasi fitur ArrayList dalam konteks yang berbeda, dari yang paling sederhana hingga penerapan dengan objek buatan sendiri.

Isi interface:

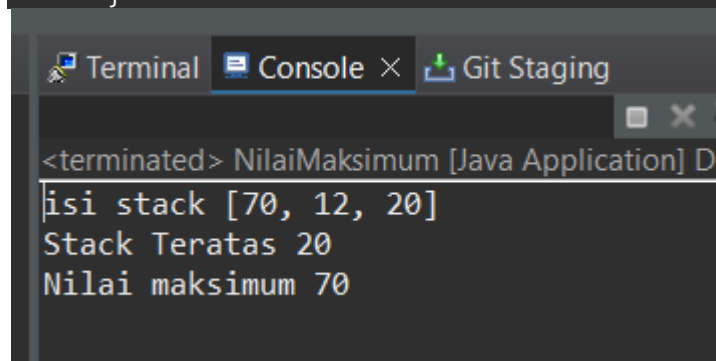
- `size()`: Mengembalikan jumlah elemen dalam stack.
- `isEmpty()`: Mengembalikan true jika stack kosong.
- `push(E e)`: Menambahkan elemen ke stack.
- `top()`: Melihat elemen teratas tanpa menghapusnya.
- `pop()`: Menghapus dan mengembalikan elemen teratas dari stack.

```
package pekan3;
import java.util.*;

public interface Stack2<E> {

    int size();
    boolean isEmpty();
    void push(E e);
    E top();
    E pop();

}
```



```
<terminated> NilaiMaksimum [Java Application] D
isi stack [70, 12, 20]
Stack Teratas 20
Nilai maksimum 70
```

B. Class 2: ArrayStack.java

Selanjutnya dibuat class `ArrayStack` yang mengimplementasikan interface `Stack2`. Langkah-langkah implementasi:

- **Deklarasi dan inisialisasi array:** Stack disimpan dalam array data dengan kapasitas maksimum 1000.
- **Variabel t:** Menyimpan indeks dari elemen teratas stack, awalnya diset ke -1.
- **Method push(E e):** Menambahkan elemen ke array di indeks t+1. Jika array sudah penuh, dilemparkan `IllegalStateException`.
- **Method top():** Mengembalikan elemen pada indeks t tanpa menghapusnya.
- **Method pop():** Mengembalikan elemen teratas, lalu menghapusnya dari array dan menurunkan t.
- **Method isEmpty() dan size():** Untuk mengecek apakah stack kosong dan menghitung jumlah elemen stack.

```
package pekan3;

public class ArrayStack <E> implements Stack2<E>
{
    public static final int CAPACITY = 1000;
    private E[] data;
    private int t = -1;

    public ArrayStack() {
        this (CAPACITY);
    }

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

    public int size() {
        return (t + 1);
    }

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

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

    public E top() {
        if (isEmpty())
            return null;
        return data[t];
    }
}
```

```

    public E pop() {
        if (isEmpty())
            return null;
        E answer = data [t];
        data [t] = null;
        t--;
        return answer;
    }
}

```

C. Class 3: contohStack.java

Class ini merupakan penggunaan stack menggunakan library bawaan Java (**java.util.Stack**).

Langkah-langkah:

1. Membuat stack bertipe Integer dengan `Stack<Integer> test = new Stack<>();`
2. Inisialisasi array `Integer[] a = {4, 8, 15, 16, 23, 42};`
3. Melakukan perulangan untuk:
 - Menampilkan isi array dengan format nilai `A[i] =`
 - Memasukkan nilai ke dalam stack dengan `push()`.
4. Menampilkan informasi stack:
 - Ukuran stack: `test.size()`
 - Nilai teratas (tidak dihapus): `test.peek()`
 - Nilai yang di-pop (dihapus): `test.pop()`

```

package pekan3;
import java.util.*;

class contohStack {

    public static void main(String[] args) {
        Stack<Integer> test = new Stack<Integer>();
        Integer[] a = {4, 8, 15, 16, 23, 42};
        for (int i = 0; i < a.length; i++) {
            System.out.println("nilai A"+i+"= " + a [i]);
            test.push (a[i]);
        }
        System.out.println("size stacknya: "+test.size());
        System.out.println("paling atas: "+test.peek());
        System.out.println("nilainya " + test.pop());
    }
}

```

```
}  
}
```

D. Class 4: contohStack2.java

Class ini merupakan penggunaan stack dengan class buatan sendiri (ArrayStack).

Langkah-langkah:

1. Membuat objek stack: `ArrayStack test = new ArrayStack();`
2. Inisialisasi array `Integer[] a = {4, 8, 15, 16, 23, 42};`
3. Melakukan perulangan untuk:
 - Menampilkan isi array.
 - Memasukkan nilai ke dalam stack menggunakan method `push()`.
4. Menampilkan hasil:
 - Ukuran stack dengan `test.size()`.
 - Nilai paling atas dengan `test.top()`.
 - Nilai yang di-pop dengan `test.pop()`.

Class ini mirip dengan `contohStack.java`, namun implementasi stack-nya bukan dari Java library melainkan buatan sendiri.

```
package pekan3;  
  
public class contohStack2 {  
    public static void main(String[] args) {  
        ArrayStack test = new ArrayStack();  
        Integer [] a = {4, 8, 15, 16, 23, 42};  
        for (int i = 0; i < a.length; i++) {  
            System.out.println("Nilai A "+i+ " = "+a[i]);  
            test.push(a[i]);  
        }  
        System.out.println("Size stacknya: " + test.size());  
        System.out.println("Paling atas: " + test.top());  
        System.out.println("Nilainya: " + test.pop());  
    }  
}
```

E. Class 5: latihanStack.java

Program ini memberikan latihan penggunaan stack dengan cara yang sederhana.

Langkah-langkah:

1. Membuat stack `Stack<Integer> s = new Stack<>();`
2. Memasukkan tiga angka: 42, -3, 17 dengan `push()`.
3. Menampilkan isi stack awal: `System.out.println(s);`
4. Melakukan operasi `pop()` lalu menampilkan hasil stack setelah `pop`.
5. Melihat elemen teratas dengan `peek()` lalu menampilkan stack setelah `peek`.

Tujuannya agar mahasiswa memahami cara kerja `push`, `pop`, dan `peek`.

```
package pekan3;
import java.util.*;

public class latihanStack {

    public static void main(String[] args) {
        Stack<Integer> s = new Stack<Integer>();
        s.push(42);
        s.push(-3);
        s.push(17);
        System.out.println("nilai stack="+s);
        System.out.println("nilai pop = "+s.pop());
        System.out.println("nilai stack setelah pop="+s);
        System.out.println("nilai peek = "+s.peek());
        System.out.println("nilai stack setelah peek="+s);
    }
}
```

F. Class 6: NilaiMaksimum.java

Program ini bertujuan untuk mencari nilai maksimum dari elemen yang ada dalam stack tanpa menghilangkan data aslinya.

Langkah-langkah:

1. Stack diisi dengan tiga nilai: 70, 12, 20.
2. Memanggil method `max(Stack s)`:
 - Mengambil elemen pertama sebagai `maxValue`.

- Melakukan iterasi seluruh isi stack untuk mencari nilai maksimum sambil menyimpannya ke stack cadangan (backup).
- Setelah itu, isi stack utama dikembalikan dari backup agar tidak hilang.

3. Menampilkan:

- Isi stack.
- Elemen teratas.
- Nilai maksimum.

Metode ini menunjukkan bagaimana stack bisa digunakan untuk proses perhitungan dengan tetap menjaga datanya.

```
package pekan3;
import java.util.Stack;

public class NilaiMaksimum {
    public static int max(Stack<Integer> s) {
        Stack<Integer> backup = new Stack<Integer>();
        int maxValue = s.pop();
        backup.push(maxValue);

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

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

        return maxValue;
    }

    public static void main(String[] args) {
        Stack<Integer> s = new Stack<Integer>();
        s.push(70);
        s.push(12);
        s.push(20);

        System.out.println("isi stack " + s);
        System.out.println("Stack Teratas " + s.peek());
        System.out.println("Nilai maksimum " + max(s));
    }
}
```

G. Class 7: StackPostfix.java

Program ini digunakan untuk **menghitung nilai dari ekspresi postfix** (notasi polandia balik) menggunakan stack.

Langkah-langkah:

1. Input ekspresi postfix berupa string, misalnya: "5 2 5 + + 7 -"
2. Gunakan Scanner untuk membaca token satu per satu:
 - Jika angka, masukkan ke stack.
 - Jika operator, ambil dua angka dari stack lalu hitung hasilnya dan masukkan kembali ke stack.
3. Terakhir, hasil evaluasi postfix ditampilkan dengan System.out.println().

Program ini sangat berguna untuk memahami salah satu penggunaan stack dalam evaluasi ekspresi matematika.

```
package pekan3;

import java.util.Stack;
import java.util.Scanner;

public class StackPostfix {

    public static int postfixEvaluate(String expression) {

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

        Scanner input = new Scanner(expression);

        while (input.hasNext()) {

            if (input.hasNextInt()) {

                // an operand (integer)

                s.push(input.nextInt());

            } else {

                // an operator

                String operator = input.next();
```

```

        int operand2 = s.pop();

        int operand1 = s.pop();

        if (operator.equals("+")) {

            s.push(operand1 + operand2);

        } else if (operator.equals("-")) {

            s.push(operand1 - operand2);

        } else if (operator.equals("*")) {

            s.push(operand1 * operand2);

        } else {

            s.push(operand1 / operand2);

        }

    }

}

return s.pop();

}

public static void main(String[] args) {

    System.out.println("hasil postfix = " + postfixEvaluate("5 2 5 + + 7 -
"));

}

}

```

BAB III

PENUTUP

3.1 Kesimpulan

Berdasarkan hasil praktikum yang telah dilakukan, dapat disimpulkan bahwa:

1. Struktur data stack merupakan struktur data linear yang mengikuti prinsip Last In First Out (LIFO), yaitu elemen terakhir yang dimasukkan adalah elemen pertama yang akan diambil.
2. Dalam praktikum ini, stack berhasil diimplementasikan dengan dua pendekatan, yaitu:
 - Menggunakan class Stack dari Java Collections Framework (`java.util.Stack`).
 - Membuat implementasi stack sendiri menggunakan array melalui class `ArrayStack`, yang mengimplementasikan interface `Stack2`.
3. Melalui program `contohStack` dan `contohStack2`, dipahami bahwa stack dapat menyimpan data dari array, lalu dilakukan operasi dasar seperti `push`, `pop`, `peek/top`, dan `size`.
4. Pada class `NilaiMaksimum`, stack juga terbukti dapat digunakan untuk menyelesaikan permasalahan logika seperti pencarian nilai maksimum sambil tetap menjaga isi stack agar tidak berubah.
5. Pada program `StackPostfix`, stack digunakan untuk mengevaluasi ekspresi postfix (notasi Polandia), menunjukkan bahwa stack sangat efektif untuk memproses ekspresi matematika atau parsing simbol.
6. Pembuatan interface `Stack2` dan implementasinya di `ArrayStack` memberikan pemahaman mengenai penggunaan antarmuka dan pewarisan dalam Java, serta bagaimana struktur data dapat dikustomisasi.