LAPORAN PRAKTIKUM 3

STRUKTUR DATA

Oleh:
Ibrahim Mousa Dhani
2411532010

Dosen Pengampu : Dr. Wahyudi S.T, M.T.

Mata Kuliah : Struktur Data

FAKULTAS TEKNOLOGI INFORMASI

DEPARTEMEN INFORMATIKA

UNIVERSITAS ANDALAS

PADANG

A. Pendahuluan

Struktur data stack merupakan salah satu struktur data linear yang menggunakan prinsip **LIFO** (Last In First Out), di mana element yang terakhir dimasukkan akan menjadi element pertama yang keluar. Stack banyak digunakan dalam berbagai aplikasi seperti algoritma rekursif, pengecekan ekspresi matematika, proses undo/redo, serta komputasi ekspresi postfix. Pada bahasa pemrograman Java, struktur data stack dapat diimplementasikan menggunakan kelas stack dari pustaka Java Collection Framework maupun secara manual dengan array atau antarmuka khusus. Praktikum ini bertujuan untuk memahami konsep dasar stack serta implementasinya dalam berbagai bentuk kode program Java.

Berikut beberapa operasi dasar pada Stack di Java :

1. Push() berfungsi untuk menambahkan element ke stack
2. Pop() berfungsi untuk menghapus dan mengambil element teratas
3. Peek() berfungsi untuk melihat element paling atas tanpa menghapusnya dari dalam stack
4. isEmpty berfungsi untuk mengecek apakah stack kosong
5. Size() berfungsi untuk menghitung jumlah element yang ada di dalam stack

B. Tujuan Praktikum

1. Memahami dan mengetahui konsep dasar Stack pada Java
2. Memahami dan mengetahui operasi dasar Stack pada Java
3. Memhami implementasi Stack dalam berbagai bentuk kode pada program Java
4. Melatih kemampuan logika dan algoritma dalam menyelesaikan permasalahan yang melibatkan stack

C. Langkah Langkah

a. latihanStack

1. Pertama buat class baru terlebih dahulu pada package pekan 3, dan beri nama latihanStack
2. Setelah itu Import kelas Stack, dari Java Collection Framework dengan cara mengetikkan ‘import.java.util.Stack;’

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

3. Lalu buat Stack dengan nama ‘s’ dan bertipe data Integer

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

4. Kemudian tambahkan element ke dalam stack yaitu 42, -3, 17 dengan mekanisme Last In, First Out (LIFO).

```
Stack<Integer> s = new Stack<>();
s.push(42);
s.push(-3);
s.push(17);
```

5. Setelah semua element dimasukkan ke dalam stack, tampilkan isi stack saat ini dengan cara seperti pada gambar berikut.

```
System.out.println("nilai stack= "+ s);
```

6. Jika ingin menghapus/ mengeluarkan element didalam stack dan ingin menampilkan element tersebut, gunakan pop()

```
System.out.println("nilai pop = "+ s.pop());
```

7. Untuk melihat isi Stack setelah di pop

```
System.out.println("nilai stack setelah pop= "+ s);
```

8. Lalu untuk melihat element/angka paling atas pada Stack, tapi tidak menghapusnya gunakan peek()

```
System.out.println("nilai peek= "+ s.peek());
```

9. Untuk melihat isi Stack setelah di peek

```
System.out.println("nilai stack setelah peek = "+ s);
```

10. Setelah semua benar dan tidak ada eror, maka jalankan program, lalu akan menghasilkan output sebagai berikut

```
nilai stack= [42, -3, 17]
nilai pop = 17
nilai stack setelah pop= [42, -3]
nilai peek= -3
nilai stack setelah peek = [42, -3]
```

11. Berikut kode program versi lengkapnya

```
1  package pekan3;
2
3  import java.util.Stack;
4
5  public class latihanStack {
6
7      public static void main(String[] args) {
8          // TODO Auto-generated method
9          Stack<Integer> s = new Stack<>();
10         s.push(42);
11         s.push(-3);
12         s.push(17);
13         System.out.println("nilai stack= "+ s);
14         System.out.println("nilai pop = "+ s.pop());
15         System.out.println("nilai stack setelah pop= "+ s);
16         System.out.println("nilai peek= "+ s.peek());
17         System.out.println("nilai stack setelah peek = "+ s);
18     }
19 }
```

b. contohStack

1. Buat class baru pada package 3, dan beri nama contohStack
2. Setelah itu Import kelas Stack atau import semua library dari Java Collection Framework dengan cara seperti berikut

```
package pekan3;

import java.util.*;

public class contohStack {
```

3. Lalu buat Stack dengan nama ‘test’ dan bertipe data Integer

```
Stack<Integer> test = new Stack<>();
```

4. Buat array berisi angka, Ini adalah data awal yang akan dimasukkan ke stack.

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

5. Gunakan for untuk memasukkan semua angka ke stack, for akan mengulang dari 0 sampai panjang array, push() digunakan untuk memasukkan angka ke dalam stack, dan System.out.println() untuk menampilkan isi array satu per satu.

```
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]);
```

6. Untuk menampilkan berapa banyak angka yang ada di stack gunakan size()

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

7. Lalu untuk melihat element/angka paling atas pada Stack, tapi tidak menghapusnya gunakan peek()

```
System.out.println("palinng atas: "+ test.peek());
```

8. Jika ingin menghapus/ mengeluarkan element didalam stack dan ingin menampilkan element tersebut, gunakan pop()

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

9. Setelah semua benar dan tidak ada eror, maka jalankan program, lalu akan menghasilkan output sebagai berikut

```
nilai A0= 4
nilai A1= 8
nilai A2= 15
nilai A3= 16
nilai A4= 23
nilai A5= 42
size stacknya: 6
palinng atas: 42
nilainya 42
```

10. Berikut kode program versi lengkapnya

```
1  package pekan3;
2
3  import java.util.*;
4
5  public class contohStack {
6
7      public static void main(String[] args) {
8          // TODO Auto-generated method stub
9          Stack<Integer> test = new Stack<>();
10         Integer[] a = {4, 8, 15, 16, 23, 42};
11         for(int i = 0; i< a.length; i++)  {
12             System.out.println("nilai A"+i+"= "+ a[i]);
13             test.push(a[i]);
14 }
15         System.out.println("size stacknya: "+ test.size());
16         System.out.println("palinng atas: "+ test.peek());
17         System.out.println("nilainya "+ test.pop());
18     }
19
20 }
```

c. Stack2

Kali ini kita akan membuat struktur data Stack sendiri (bukan memakai Stack bawaan Java) yaitu ArrayStack. Hal pertama yang harus kita lakukan yaitu membuat interface yang berfungsi untuk mendefinisikan aturan/pedoman stack. Disini kita akan membuat interface dengan nama Stack2 :

1. Pertama buat interface pada package pekan 3 dan beri nama Stack2 seperti pada gambar berikut

```
1  package pekan3;
2
3  public interface Stack2<E>{
```

interface artinya ini bukan **class biasa**, tapi hanya berisi daftar **metode** (tanpa isi). <E> artinya interface ini generik, bisa digunakan untuk berbagai jenis data, seperti Integer, String, dll.

2. Tambahkan method Stack yaitu :
    - int size() = menghitung isi stack
    - boolean isEmpty() = mengecek apakah stack kosong, mengembalikan true jika stack kosong, dan false jika tidak.
    - push(E e) = Menambah element e ke bagian atas stack.
    - E top() = Mengembalikan element teratas di stack, tapi tidak menghapusnya. Sama cara kerjanya dengan peek()
    - E pop() = Menghapus/ mengeluarkan element teratas dari stack.

```
1  package pekan3;
2
3  public interface Stack2<E>{
4      int size();
5      boolean isEmpty();
6      void push (E e);
7      E top();
8      E pop();
9  }
```

d. ArrayStack

Selanjutnya kita akan membuat class ArrayStack yang mengikuti interface Stack2

1. Pertama buat class baru dan beri nama sesuai struktur data stack sendiri yang akan dibuat, yaitu ArrayStack dan jangan lupa gunakan Interface Stack2 agar class ini mengikuti aturan dari interface Stack2 (yang sebelumnya dibuat).

```
package pekan3;

public class ArrayStack<E>  implements Stack2<E>
```

2. Selanjutnya buat Variabel untuk menyimpan data, CAPACITY: ukuran maksimum stack adalah 1000 (default), data[] : array tempat menyimpan data, dan t : penunjuk element paling atas. -1 berarti kosong

```
{
public static final int CAPACITY = 1000;
private E[] data;
private int t = -1;
```

3. Lalu buat constructor tanpa parameter dan buat constructor kedua dengan parameter kapasitas

```
public ArrayStack() {
    this(CAPACITY);
}
```

Gambar diatas adalah constructor default. Artinya, kalau kamu membuat stack tanpa menyebut ukuran, constructor ini akan dipanggil, di dalamnya ada this(CAPACITY); yang artinya memanggil constructor kedua dengan ukuran default (biasanya 1000).

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

Gambar diatas adalah constructor kedua, digunakan jika ingin menentukan sendiri ukuran stack.

4. Buat method size(), sesuai dengan interface Stack2 yang telah dibuat tadi

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

t adalah indeks element teratas, jadi, ukuran stack adalah t + 1.

5. Buat method isEmpty, sesuai dengan interface Stack2 yang telah dibuat tadi

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

Stack dianggap kosong jika tidak ada element, yaitu saat t = -1.

6. Buat method push(E e), berfungsi untuk Menambahkan element ke atas stack. Sebelum menambahkan, dicek apakah stack sudah penuh. Lalu ++t menaikkan indeks t lalu memasukkan e ke posisi tersebut.

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

7. Buat method top(), Mengembalikan element paling atas tanpa menghapusnya. Jika stack kosong, kembalikan null. (Cara kerjanya sama dengan peek().)

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

8. Buat method pop()

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

    t--;
    return answer;
}
```

Berfungsi untuk menghapus element paling atas dan mengembalikannya. Setelah dihapus, nilai t diturunkan.

e. contohStack2

Buat program/class utama untuk mencoba menjalankan stack buatan sendiri yaitu ArrayStack caranya sama seperti membuat class baru pada umumnya.

1. Buat class baru pada package pekan 3, dan disini saya beri nama contohStack2

```
package pekan3;

import java.util.*;

public class contohStack2 {
```

2. Lalu buat Object Stack dengan nama ‘test’ dan bertipe data String, class ini akan memanggil dan menggunakan class ArrayStack yang sebelumnya dibuat

```
ArrayStack<Integer> test = new ArrayStack<>();
```

3. Selanjutnya lakukan sama seperti class contohStack sebelumnya yaitu buat array berisi angka, Ini adalah data awal yang akan dimasukkan ke stack, dan Gunakan for untuk memasukkan semua angka ke stack, for akan mengulang dari 0 sampai panjang array, push() digunakan untuk memasukkan angka ke dalam stack, dan System.out.println() untuk menampilkan isi array satu per satu.

```
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]);
```

4. Untuk menampilkan jumlah element yang ada di stack, gunakan size()

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

5. Berbeda dengan class contohStack yang sebelumnya yang mengguakan peek() untuk menampilkan element paling atas pada tanpa menghapusnya, pada class contohStack2 ini menggunakan top().

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

6. Dan yang terakhir untuk menghapus/ mengeluarkan element didalam stack dan ingin menampilkan element tersebut, gunakan pop()

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

7. Setelah semua benar dan tidak ada eror, maka jalankan program, lalu akan menghasilkan output sebagai berikut

```
nilai A0= 4
nilai A1= 8
nilai A2= 15
nilai A3= 16
nilai A4= 23
nilai A5= 42
size stacknya: 6
palinng atas: 42
nilainya 42
```

Outputnya sama dengan class contohStack, tetapi pada class ini menggunakan Stack sendiri (bukan memakai Stack bawaan Java)

8. Berikut kode program versi lengkapnya

```java
package pekan3;

import java.util.*;

public class contohStack2 {

    public static void main(String[] args) {

        ArrayStack<Integer> 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("palinng atas: "+ test.top());
        System.out.println("nilainya "+ test.pop());
    }

}
```

f. NilaiMaksimum

1. Buat class baru pada package , dan beri nama NilaiMaksimum
2. Setelah itu Import kelas Stack dari Java Collection Framework

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

public class NilaiMaksimum {
```

3. Selanjutnya kita akan membuat method max, yaitu bertugas untuk mencari nilai maksimum dari isi stack, sambil menjaga isi stack tetap sama setelah proses pencarian. Pop satu element pertama dari stack s → misalnya 20 anggap itu sementara sebagai nilai maksimum. Dan simpan element itu ke stack backup agar bisa dikembalikan nanti.

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

4. Iterasi Sisa Stack, ulangi hingga stack s kosong. Bandingkan setiap element dengan maxValue untuk menemukan nilai maksimum sebenarnya. Setiap element juga disimpan di stack backup

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

5. Ambil satu per satu dari backup dan kembalikan ke s. Karena Stack bersifat LIFO, data yang sudah di-pop bisa dikembalikan dalam urutan semula. Dan kembailikan nilai maksimum.

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

    return maxValue;
}
```

6. Lalu lanjut pada method utama(main), buat Object Stack dengan nama ‘s’ dan bertipe Integer
7. Tambahkan element ke dalam Stack yang telah dibuat yaitu : 70, 12, 20

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

8. Setelah semua element ditambahkan untuk melihat seluruh isi Stack bisa dengan cara seperti gambar berikut

```
System.out.println("isi stack " + s);
```

9. Lalu untuk melihat element/angka paling atas pada Stack, tapi tidak menghapusnya gunakan peek()

```
System.out.println("Stack Teratas " + s.peek());
```

10. Untuk melihat nilai maksimum dapat menggunakan max(s)

```
System.out.println("Nilai maksimum " + max(s));
```

11. Setelah semua benar dan tidak ada eror, maka jalankan program, lalu akan menghasilkan output sebagai berikut

```
isi stack [70, 12, 20]
Stack Teratas 20
Nilai maksimum 70
```

12. Berikut kode program versi lengkapnya

```java
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. StackPostfix

1. Seperti biasa pertama buat class baru dan beri nama StackPostfix
2. Import kelas Stack dan Scanner dari Java Collection Framework

```java
package pekan3;

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

public class StackPostfix {
```

3. Lalu buat method postfixEvaluate(), method ini menerima string postfix (misal: "5 2 5 + + 7 -") dan akan mengembalikan hasilnya sebagai int.

```java
public static int postfixEvaluate(String expression) {
    Stack<Integer> s = new Stack<Integer>();
    Scanner input = new Scanner(expression);
```

String<Integer> untuk menyimpan operand. Scanner input = new Scanner(expression) digunakan agar kita bisa membaca token satu per satu dari string tersebut

4. Lakukan iterasi sampai tidak ada lagi token dalam ekspresi postfix. Jika token berikutnya adalah angka (operand), langsung dimasukkan ke dalam stack. Jika token bukan angka, berarti itu operator (+, -, *, atau /).

```
while (input.hasNext()) {
    if (input.hasNextInt()) {
        // an operand (integer)
        s.push(input.nextInt());
    } else {
        // an operator
```

5. Ambil operand kedua dari stack karena stack bersifat LIFO, ini adalah nilai terakhir yang masuk. Dan Ambil operand pertama dari stack (nilai sebelumnya), bandingkan jenis operator, lalu lakukan operasi sesuai simbol. Hasil dari operasi tadi dimasukkan kembali ke stack, ambil hasil akhir dari stack (seharusnya hanya tersisa 1 nilai), dan itu adalah hasil evaluasi postfix.

```
            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();
```

6. Lalu lanjut pada method utama(main), memanggil fungsi postfixEvaluate() yang telah dibuat tadi dengan ekspresi "5 2 5 + + 7 -"(sesuai keinginan) dan menampilkan hasilnya ke layar.

```
public static void main(String[] args) {
    System.out.println("hasil postfix = " + postfixEvaluate("5 2 5 + + 7 -"));
}
```

Proses yang terjadi: Ekspresi "5 2 5 + + 7 -" dikirim ke fungsi postfixEvaluate. Fungsi akan memprosesnya dan mengembalikan hasil akhir. System.out.println akan mencetak hasilnya ke layar:

7. Output yang akan dihasilkan yaitu :

```
hasil postfix = 5
```

8. Berikut kode program versi lengkapnya

```java
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 -"));
    }

}
```

D. Kesimpulan

Dalam praktikum ini, mahasiswa telah mempelajari konsep dan implementasi struktur data Stack di Java. Stack merupakan struktur data yang mengikuti prinsip LIFO (Last In First Out), di mana element terakhir yang dimasukkan akan menjadi element pertama yang dikeluarkan. Melalui implementasi menggunakan class bawaan java.util.Stack serta pembuatan stack custom berbasis array (ArrayStack), mahasiswa memahami bagaimana operasi dasar seperti push, pop, peek/top, size, dan isEmpty dijalankan. Selain itu, mahasiswa juga mencoba penerapan stack dalam kasus nyata seperti evaluasi ekspresi postfix dan pencarian nilai maksimum dalam stack.

Dengan praktik langsung ini, mahasiswa tidak hanya memahami teori struktur data stack, tetapi juga mampu mengimplementasikannya secara mandiri dalam kode Java