

LAPORAN PRAKTIKUM STRUKTUR DATA
“Stack”



OLEH :
EZZA ADDINI
2311532001

DOSEN PENGAMPU: Dr. WAHYUDI, M. T.

DEPARTEMEN INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS ANDALAS
PADANG
2024

A. PENDAHULUAN

Dalam struktur data, "stack" adalah kumpulan elemen yang disusun secara linear dengan dua operasi utama, yaitu push (menambahkan elemen ke stack) dan pop (menghapus elemen teratas dari stack). Stack biasanya disusun menurut prinsip LIFO (Last In, First Out), yang berarti elemen terakhir yang dimasukkan ke dalam stack akan menjadi elemen pertama yang dihapus.

Dalam bahasa pemrograman Java, kelas Stack biasanya digunakan untuk mengimplementasikan stack, tetapi sejak Java 6, disarankan untuk menggunakan kelas Deque dari paket yang sama untuk mengimplementasikan stack karena Stack telah dianggap sebagai kelas yang sudah usang. Untuk membuat Deque berfungsi sebagai stack, metode push() dapat digunakan untuk menambahkan elemen dan metode pop() dapat menghapus elemen teratas.

B. TUJUAN

Tujuan dari praktikum ini adalah:

1. Membuat program latihan Stack.
2. Membuat program ArrayStack.
3. Membuat program untuk menentukan nilai maksimum pada sebuah Stack.
4. Membuat program postfix pada Stack.

C. LANGKAH KERJA

a. Program Latihan Stack

1. Buat package dan class baru, namakan sesuai dengan yang diperintahkan atau yg diinginkan. Jangan lupa untuk menambahkan "import java.util.*;" pada baris setelah package.

```
LatihanStack.java ×  
1 package Pekan2;  
2 import java.util.*;  
3 public class LatihanStack {
```

2. Deklarasikan tipe data stack yang akan digunakan.

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

3. Inisialisasikan elemen-elemen yang akan dimasukkan ke dalam stack.

```
s.push(42);  
s.push(-3);  
s.push(17);
```

4. Buatlah sintaks untuk menampilkan output program.

```
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);
```

5. Jalankan program.

```
Console ×  
<terminated> LatihanStack [Java Application] C:\Users  
Nilai stack: [42, -3, 17]  
Nilai pop: 17  
Nilai stack setelah pop: [42, -3]  
Nilai peek: -3  
Nilai stack setelah peek: [42, -3]
```

b. Membuat Program ArrayStack

1. Buat class baru dan namakan sesuai perintah atau yang diinginkan, lalu buat "import java.util.*;" pada baris setelah package.

```
LatihanStack.java ContohStack.java ×  
1 package Pekan2;  
2 import java.util.*;  
3 public class ContohStack {
```

2. Buat kelas ArrayStack seperti berikut ini.

```
public static void main(String[] args) {  
    ArrayStack test = new ArrayStack();  
    // ...
```

3. Deklarasikan nilai yang akan diinputkan pada ArrayStack.

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

4. Buatlah sintaks for loop-nya.

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

5. Buat sintaks untuk menampilkan output program.

```
System.out.println("Size stack-nya: " + test.size());
System.out.println("Paling atas: " + test.top());
System.out.println("Nilainya: " + test.pop());
```

6. Klik kanan pada package Pekan2, klik New, buat Interface baru dengan nama yang diinginkan.

```
LatihanStack.java ContohStack.java S
1 package Pekan2;
2
3 public interface Stack2<E> {
```

7. Deklarasikan variabel yang akan digunakan.

```
4 int size();
5 boolean isEmpty();
6 void push(E e);
7 E top();
8 E pop();
9 }
```

8. Kemudian, buat class baru lagi dengan nama yang diinginkan.
9. Namakan public class-nya dengan nama sebagai berikut.

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

10. Buatkan method seperti berikut.

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

11. Buat sintaks untuk membuat stack dengan kapasitas default.

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

12. Buat sintaks untuk membuat stack dengan kapasitas yang diberikan.

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

13. Buat sintaks untuk mengembalikan jumlah elemen dalam stack.

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

14. Buat sintaks untuk memeriksa apakah stack kosong atau tidak.

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

15. Buat sintaks untuk menambahkan elemen baru ke dalam stack.

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

16. Buat sintaks untuk mengembalikan elemen teratas dari stack tanpa menghapusnya.

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

17. Buat sintaks untuk menghapus dan mengembalikan elemen teratas dari stack.

```
public E pop() {
    if (isEmpty())
        return null;
    E answer = data[t];
    data[t] = null; //dereference to help garbage collection
    t--;
    return answer;
}
```

18. Jalankan program ArrayStack pada class ArrayStack.

```
Console x
<terminated> ContohStack [Ja
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 stack-nya: 6
Paling atas: 42
Nilainya: 42
```

c. Program Nilai Maksimum pada Stack

1. Buat class baru dan namakan sesuai perintah atau yang diinginkan, lalu buat “import java.util.*;” pada baris setelah package.

```
package Pekan2;
import java.util.*;
public class NilaiMaksimum {
```

2. Deklarasikan tipe data stack yang akan digunakan.

```
public static int max(Stack<Integer> s) {
    Stack<Integer> backup = new Stack<Integer>();
```

3. Buat sintaks untuk mengambil nilai maksimum dari stack s dan memindahkannya ke stack backup.

```
int maxValue = s.pop();
backup.push(maxValue);
```

4. Buat sintaks untuk mengambil semua elemen dari stack s, memindahkannya ke stack backup, serta mencari nilai maksimum dari semua elemen di dalam stack s.

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

5. Buat sintaks untuk memindahkan semua elemen dari stack backup kembali ke stack s, serta mengembalikan nilai maksimum yang telah dihitung sebelumnya.

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

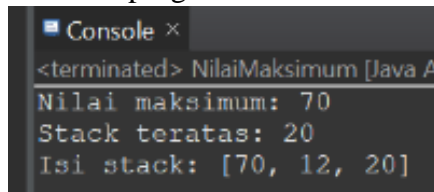
6. Buat sintaks untuk menginputkan data ke dalam stack.

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

7. Buat sintaks untuk menampilkan output program.

```
System.out.println("Nilai maksimum: " + max(s));
System.out.println("Stack teratas: " + s.peek());
System.out.println("Isi stack: " + s);
```

8. Jalankan program.



```
Console x
<terminated> NilaiMaksimum [Java A
Nilai maksimum: 70
Stack teratas: 20
Isi stack: [70, 12, 20]
```

d. Program PostFix Stack

1. Buatlah class baru dan namakan sesuai perintah atau yang diinginkan, lalu buat "import java.util.*;" pada baris setelah package.

```
1 package Pekan2;
2 import java.util.*;
3 public class StackPostFix {
4
```

2. Deklarasikan nama stack yang akan dibuat.

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

3. Buat sintaks untuk membaca masukan dari suatu sumber, memeriksa apakah masukan berupa operand (bilangan bulat) atau operator, dan kemudian ditambahkan ke dalam stack.

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

4. Deklarasikan variabel yang akan digunakan.

```
String operator = input.next();
int operand2 = s.pop();
int operand1 = s.pop();
```

5. Buat sintaks if seperti berikut.

```

        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);
        }
    }
}

```

6. Buat sintaks untuk mengembalikan nilai dari elemen teratas stack `s` setelah menghapusnya dari stack.

```

return s.pop();

```

7. Buat sintaks untuk menampilkan output program.

```

}
public static void main(String[] args) {
    System.out.println("Hasil postfix = " + postfixEvaluate("5 2 4 * + 7 -"));
}

```

8. Jalankan program.

```

Console x
<terminated> StackPostFix [Java 7]
Hasil postfix = 6

```

D. KESIMPULAN

Dari praktikum yang telah dilakukan, maka dapat diambil kesimpulan bahwa "stack" adalah kumpulan elemen yang disusun secara linear dengan dua operasi utama, yaitu push (menambahkan elemen ke stack) dan pop (menghapus elemen teratas dari stack). Stack biasanya disusun menurut prinsip LIFO (Last In, First Out), yang berarti elemen terakhir yang dimasukkan ke dalam stack akan menjadi elemen pertama yang dihapus.

Pada praktikum kali ini, kita membuat program stack sederhana dengan operasi push, pop, dan peek. Kemudian, dibuat juga program gabungan ArrayStack dengan cara membuat class Interface serta class lain yang akan mengimplementasikan program ArrayStack. Lalu kita juga membuat program untuk menentukan nilai maksimum dari sebuah stack, serta yang terakhir program untuk membuat postfix pada stack.