LAPORAN PRAKTIKUM

MATA KULIAH ALGORITMA DAN STRUKTUR DATA

PERTEMUAN 9 : STACK

KAYLA RACHMAUDINA SATITI PUTRI

2341760103

D-IV SISTEM INFORMASI BISNIS

JURUSAN TEKNOLOGI INFORMASI POLITEKNIK

NEGERI MALANG

2024

JOBSHEET VIII
STACK

7.1 Tujuan Praktikum

Setelah melakukan materi praktikum ini, mahasiswa mampu:

1. Mengenal struktur data Stack
2. Membuat dan mendeklarasikan struktur data Stack
3. Menerapkan algoritma Stack dengan menggunakan array

7.2. Praktikum 1

Waktu percobaan: 30 menit

Pada percobaan ini, kita akan membuat program yang mengimplementasikan struktur data Stack dan operasi-operasi dasar pada struktur data Stack menggunakan array.

7.2.1 Langkah-langkah Percobaan

1. Buat folder dengan nama Praktikum07. Buat file Stack.java.
2. Tulis kode untuk membuat atribut dan konstruktor pada class Stack

```java
public class Stack15 {
    int data[];
    int size;
    int top;

    public Stack15(int size) {
        this.size = size;
        data = new int[size];
        top = -1;
    }
```

3. Lalu tambahkan method isFull() dan isEmpty() pada class Stack

```java
    public boolean isFull() {
        if (top == size - 1) {
            return true;
        } else {
            return false;
        }
    }

    public boolean isEmpty() {
        if (top == -1) {
            return true;
        } else {
            return false;
        }
    }
```

4. Tambahkan method push(int data) dan pop()

```
28      public void push(int dt) {
29          if (!isFull()) {
30              top++;
31              data[top] = dt;
32          } else {
33              System.out.println("Stack penuh");
34          }
35      }
36
37      public void pop() {
38          if (!isEmpty()) {
39              int x = data[top];
40              top--;
41              System.out.println("Data yang dikeluarkan dari Stack : " + x);
42          } else {
43              System.out.println("Stack masih kosong");
44          }
45      }
```

5. Tambahkan method peek()

```
47      public void peek() {
48          System.out.println("Elemen teratas Stack : " + data[top]);
49      }
```

6. Tambahkan method print() dan clear()

```
51      public void print() {
52          System.out.println("Isi stack : ");
53          for (int i = top; i >= 0; i--) {
54              System.out.println(data[i] + " ");
55          }
56          System.out.println("");
57      }
58
59      public void clear() {
60          if (!isEmpty()) {
61              for (int i = top; i >= 0; i--) {
62                  top--;
63              }
64              System.out.println("Stack sudah dikosongkan");
65          } else {
66              System.out.println("Stack masih kosong");
67          }
68      }
69  }
```

7. Buat file **StackDemo.java** untuk mengimplementasikan class StackDemo yang berisi fungsi main untuk membuat objek Stack dan mengoperasikan method-method pada class Stack.

```
1   public class StackDemo15 {
2       public static void main(String[] args) {
3           Stack15 stack = new Stack15(10);
4           stack.push(8);
5           stack.push(12);
6           stack.push(18);
7           stack.print();
8           stack.pop();
9           stack.peek();
10          stack.pop();
11          stack.push(-5);
12          stack.print();
13      }
14  }
```

8. Compile dan run class StackDemo.

7.2.2 Verifikasi Hasil Percobaan

```
Isi stack :
18
12
8

Data yang dikeluarkan dari Stack : 18
Elemen teratas Stack : 12
Data yang dikeluarkan dari Stack : 12
Isi stack :
-5
8
```

7.2.3 Pertanyaan

1. Pada method pop(), mengapa diperlukan pemanggilan method isEmpty()? Apa yang terjadi jika tidak ada pemanggilan isEmpty()?

   Digunakan untuk memeriksa apakah stack kosong sebelum melakukan operasi pop. Jika tidak ada pemanggilan method isEmpty(), maka program akan langsung akan melakukan operasi pengurangan indeks top tanpa mengambil data terlebih dahulu. Hal ini dapat menyebabkan error pengindeksan (index out of bounds) jika method pop() dipanggil saat stack masih kosong.

2. Jelaskan perbedaan antara method peek() dengan method pop() pada class Stack.

   Method peek() hanya melihat elemen teratas pada stack tanpa mengubahnya, sedangkan method pop() mengambil (mengeluarkan) elemen teratas dan mengurangi indeks top sehingga stack menjadi lebih rendah.

7.3. Praktikum 2

Waktu percobaan : 45 menit

Pada percobaan ini, kita akan membuat program yang mengilustrasikan tumpukan pakaian yang disimpan ke dalam stack. Karena sebuah pakaian mempunyai beberapa informasi, maka implementasi Stack dilakukan dengan menggunakan array of object untuk mewakili setiap elemennya.

7.3.1. Langkah-langkah Percobaan

1. Perhatikan Diagram Class Pakaian berikut ini:

Pakaian
jenis: String warna: String merk: String ukuran: String harga: double
Pakaian(jenis: String, warna: String, merk: String, ukuran: String, harga: double)

Berdasarkan diagram class tersebut, akan dibuat program class Pakaian dalam Java.

2. Buat class baru dengan nama **Pakaian**.
3. Tambahkan atribut-atribut Pakaian seperti pada Class Diagram Pakaian, kemudian tambahkan pula konstruktornya seperti gambar berikut ini.

```java
public class Pakaian15 {
    String jenis, warna, merk, ukuran;
    double harga;

    Pakaian15(String jenis, String warna, String merk, String ukuran, double harga) {
        this.jenis = jenis;
        this.warna = warna;
        this.merk = merk;
        this.ukuran = ukuran;
        this.harga = harga;
    }
}
```

4. Setelah membuat class Pakaian, selanjutnya perlu dibuat class **Stack** yang berisi atribut dan method sesuai diagram Class Stack berikut ini:

Stack
size: int top: int data[]: Pakaian

Stack(size: int) IsEmpty(): boolean IsFull(): boolean push(): void
pop(): void peek(): void print(): void clear(): void

Keterangan**: Tipe data pada variabell **data** menyesuaikan dengan data yang akan akan disimpan di dalam Stack. Pada praktikum ini, data yang akan disimpan merupakan array of object dari Pakaian, sehingga tipe data yang digunakan adalah **Pakaian

5. Buat class baru dengan nama Stack. Kemudian tambahkan atribut dan konstruktor seperti gambar berikut ini.

```
public class Stack15_Pakaian {
    int size;
    int top;
    Pakaian15 data[];

    public Stack15_Pakaian(int size){
        this.size = size;
        data = new Pakaian15[size];
        top = -1;
    }
```

6. Buat method **IsEmpty** bertipe boolean yang digunakan untuk mengecek apakah stack kosong.

```
public class Stack15_Pakaian {
    int size;
    int top;
    Pakaian15 data[];

    public Stack15_Pakaian(int size){
        this.size = size;
        data = new Pakaian15[size];
        top = -1;
    }
```

7. Buat method **IsFull** bertipe boolean yang digunakan untuk mengecek apakah stack sudah terisi penuh.

```
    public boolean isFull() {
        if (top == size - 1) {
            return true;
        } else {
            return false;
        }
    }
```

8. Buat method **push** bertipe void untuk menambahkan isi elemen stack dengan parameter **pkn** yang berupa object **Pakaian**

```
    public void push(Pakaian15 pkn) {
        if (!isFull()) {
            top++;
            data[top] = pkn;
        } else {
            System.out.println("Isi stack penuh!");
        }
    }
```

9. Buat method **Pop** bertipe void untuk mengeluarkan isi elemen stack. Karena satu elemen stack terdiri dari beberapa informasi (jenis, warna, merk, ukuran, dan harga), maka ketika mencetak data juga perlu ditampilkan semua informasi tersebut

```
37      public void pop() {
38          if (!isEmpty()) {
39              Pakaian15 x = data[top];
40              top--;
41              System.out.println("Data yang keluar : " + x.jenis + " " + x.warna + " " + x.merk + " " + x.ukuran + " " + x.harga);
42          } else {
43              System.out.println("Stack masih kosong");
44          }
45      }
```

10. Buat method **peek** bertipe void untuk memeriksa elemen stack pada posisi paling atas.

```
47      public void peek() {
48          System.out.println("Elemen teratas   : " + data[top].jenis + " " + data[top].warna + " " + data[top].merk + " " + data[top].ukuran + " " + data[top].harga);
49      }
```

11. Buat method **print** bertipe void untuk menampilkan seluruh elemen pada stack.

```
51      public void print() {
52          System.out.println("\nIsi stack : ");
53          for (int i = top; i >= 0; i--) {
54              System.out.println(data[i].jenis + " " + data[i].warna + " " + data[i].merk + " " + data[i].ukuran + " " + data[i].harga + " ");
55          }
56          System.out.println("");
57      }
```

12. Buat method **clear** bertipe void untuk menghapus seluruh isi stack.

```
59      public void clear() {
60          if (!isEmpty()) {
61              for (int i = top; i >= 0; i--) {
62                  top--;
63              }
64              System.out.println("Stack sudah dikosongkan");
65          } else {
66              System.out.println("Stack masih kosong");
67          }
68      }
69  }
```

13. Selanjutnya, buat class baru dengan nama **StackMain**. Buat fungsi main, kemudian lakukan instansiasi objek dari class **Stack** dengan nama **stk** dan nilai parameternya adalah 5.

```
1  import java.util.Scanner;
2
3  public class Stack15Main {
4      public static void main(String[] args) {
5          Scanner sc = new Scanner(System.in);
6          Stack15_Pakaian stk = new Stack15_Pakaian(5);
```

14. Deklarasikan Scanner dengan nama **sc**
15. Tambahkan kode untuk menerima input data Pakaian, kemudian semua informasi tersebut dimasukkan ke dalam stack

```
        char pilih;
        do {
            System.out.println();
            System.out.print("Jenis\t: ");
            String jenis = sc.nextLine();
            System.out.print("Warna\t: ");
            String warna = sc.nextLine();
            System.out.print("Merk\t: ");
            String merk = sc.nextLine();
            System.out.print("Ukuran\t: ");
            String ukuran = sc.nextLine();
            System.out.print("Harga\t: ");
            double harga = sc.nextDouble();

            Pakaian15 p = new Pakaian15(jenis, warna, merk, ukuran, harga);
            System.out.print("Apakah anda akan menambahkan data baru ke stack (y/n)? ");
            pilih = sc.next().charAt(0);
            // mengabaikan karakter new line
            sc.nextLine();
            stk.push(p);
        } while (pilih == 'y');
```

***Catatan**: sintaks sc.nextLine() sebelum sintaks st.push(p) digunakan untuk mengabaikan karakter new line*

16. Lakukan pemanggilan method print, method pop, dan method peek dengan urutan sebagai berikut.

```
        stk.print();
        stk.pop();
        stk.peek();
        stk.print();
```

17. Compile dan jalankan class **StackMain**, kemudian amati hasilnya.

7.3.2. Verifikasi Hasil Percobaan

```
Jenis   : kaos
Warna   : hitam
Merk    : nevada
Ukuran  : m
Harga   : 85000
Apakah anda akan menambahkan data baru ke stack (y/n)? y

Jenis   : kemeja
Warna   : putih
Merk    : styves
Ukuran  : xl
Harga   : 127000
Apakah anda akan menambahkan data baru ke stack (y/n)? y

Jenis   : celana
Warna   : biru
Merk    : levis
Ukuran  : l
Harga   : 189500
Apakah anda akan menambahkan data baru ke stack (y/n)? n

Isi stack :
celana biru levis l 189500.0
kemeja putih styves xl 127000.0
kaos hitam nevada m 85000.0

Data yang keluar : celana biru levis l 189500.0
Elemen teratas   : kemeja putih styves xl 127000.0

Isi stack :
kemeja putih styves xl 127000.0
kaos hitam nevada m 85000.0
```

7.3.3. Pertanyaan

1. Berapa banyak data pakaian yang dapat ditampung di dalam stack? Tunjukkan potongan kode program untuk mendukung jawaban Anda tersebut!

   5, dari kode program berikut ini size nya 5.

```
6           Stack15_Pakaian stk = new Stack15_Pakaian(5);
```

2. Perhatikan class **StackMain**, pada saat memanggil fungsi push, parameter yang dikirimkan adalah **p**. Data apa yang tersimpan pada variabel **p** tersebut?

```
stk.push(p);
```

   Data pakaian seperti jenis, warna, merk, ukuran, harga

3. Apakah fungsi penggunaan **do-while** yang terdapat pada class **StackMain**?

Agar data pakaian bisa dimasukkan secara berulang-ulang ke dalam stack hingga mencapai kapasitas maksimumnya.

4. Modifikasi kode program pada class **StackMain** sehingga pengguna dapat memilih operasioperasi pada stack (push, pop, peek, atau print) melalui pilihan menu program dengan memanfaatkan kondisi IF-ELSE atau SWITCH-CASE!

```
8           int pilihan;
9
10          do {
11              System.out.println("\nPilih Operasi:");
12              System.out.println("1. Push");
13              System.out.println("2. Pop");
14              System.out.println("3. Peek");
15              System.out.println("4. Print");
16              System.out.println("5. Exit");
17              System.out.print("Masukkan pilihan: ");
18              pilihan = sc.nextInt();
19              sc.nextLine();
20
21              switch (pilihan) {
22                  case 1:
23                      System.out.println();
24                      System.out.print("Jenis\t: ");
25                      String jenis = sc.nextLine();
26                      System.out.print("Warna\t: ");
27                      String warna = sc.nextLine();
28                      System.out.print("Merk\t: ");
29                      String merk = sc.nextLine();
30                      System.out.print("Ukuran\t: ");
31                      String ukuran = sc.nextLine();
32                      System.out.print("Harga\t: ");
33                      double harga = sc.nextDouble();
34
35                      Pakaian15 p = new Pakaian15(jenis, warna, merk, ukuran, harga);
36                      stk.push(p);
37                      break;
38                  case 2:
39                      stk.pop();
40                      break;
41                  case 3:
42                      stk.peek();
43                      break;
44                  case 4:
45                      stk.print();
46                      break;
47                  case 5:
48                      System.out.println("Program selesai.");
49                      break;
50                  default:
51                      System.out.println("Pilihan tidak valid.");
52              }
53          } while (pilihan != 5);
54
55          sc.close();
56      }
57  }
```

7.4. Praktikum 3

Waktu percobaan : 30 menit

Pada percobaan ini, kita akan membuat program untuk melakukan konversi notasi infix menjadi notasi postfix.

7.4.1. Langkah-langkah Percobaan

1. Perhatikan Diagram Class berikut ini:

Postfix
n: int top: int stack: char[]
Postfix(total: int) push(c: char): void pop(): void IsOperand(c: char): boolean IsOperator(c: char): boolean derajat(c: char): int konversi(Q: String): string

Berdasarkan diagram class tersebut, akan dibuat program class Postfix dalam Java.

2. Buat class baru dengan nama **Postfix**. Tambahkan atribut **n**, **top**, dan **stack** sesuai diagram class Postfix tersebut.
3. Tambahkan pula konstruktor berparameter seperti gambar berikut ini.

```
public class Postfix15 {
    int n, top;
    char[] stack;

    public Postfix15(int total) {
        n = total;
        top = -1;
        stack = new char[n];
        push('(');
    }
```

4. Buat method **push** dan **pop** bertipe void.

```
    public void push(char c) {
        top++;
        stack[top] = c;
    }

    public char pop() {
        char item = stack[top];
        top--;
        return item;
    }
```

5. Buat method **IsOperand** dengan tipe boolean yang digunakan untuk mengecek apakah elemen data berupa operand.

```
23      public boolean isOperand(char c) {
24          if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == ' ' || c == '.') {
25              return true;
26          } else {
27              return false;
28          }
29      }
```

6. Buat method **IsOperator** dengan tipe boolean yang digunakan untuk mengecek apakah elemen data berupa operator.

```
31      public boolean isOperator(char c) {
32          if (c == '^' || c == '%' || c == '/' || c == '*' || c == '-' || c == '+') {
33              return true;
34          } else {
35              return false;
36          }
37      }
```

7. Buat method **derajat** yang mempunyai nilai kembalian integer untuk menentukan derajat operator.

```
39      public int derajat(char c) {
40          switch (c) {
41              case '^':
42                  return 3;
43              case '%':
44                  return 2;
45              case '/':
46                  return 2;
47              case '*':
48                  return 2;
49              case '-':
50                  return 1;
51              case '+':
52                  return 1;
53              default:
54                  return 0;
55          }
56      }
```

8. Buat method konversi untuk melakukan konversi notasi infix menjadi notasi postfix dengan cara mengecek satu persatu elemen data pada **String Q** sebagai parameter masukan.

```
58      public String konversi(String Q) {
59          String P = "";
60          char c;
61          for (int i = 0; i < n; i++) {
62              c = Q.charAt(i);
63
64              if (isOperand(c)) {
65                  P = P + c;
66              }
67
68              if (c == '(') {
69                  push(c);
70              }
71
72              if (c == ')') {
73                  while (stack[top] != '(') {
74                      P = P + pop();
75                  }
76                  pop();
77              }
78
79              if (isOperator(c)) {
80                  while (derajat(stack[top]) >= derajat(c)) {
81                      P = P + pop();
82                  }
83                  push(c);
84              }
85          }
86          return P;
87      }
88  }
```

9. Selanjutnya, buat class baru dengan nama **PostfixMain**. Buat class main, kemudian buat variabel P dan Q. Variabel P digunakan untuk menyimpan hasil akhir notasi postfix setelah dikonversi, sedangkan variabel Q digunakan untuk menyimpan masukan dari pengguna berupa ekspresi matematika dengan notasi infix. Deklarasikan variabel Scanner dengan nama sc, kemudian panggil fungsi *built-in* **trim** yang digunakan untuk menghapus adanya spasi di depan atau di belakang teks dari teks persamaan yang dimasukkan oleh pengguna.

```
1  import java.util.Scanner;
2
3  public class PostFixMain15 {
4      public static void main(String[] args) {
5          Scanner sc = new Scanner(System.in);
6          String P, Q;
7          System.out.println("Masukkan ekspresi matematika (infix) : ");
8          Q = sc.nextLine();
9          Q = Q.trim();
10         Q = Q + ")";
```

Penambahan string **“)”** digunakan untuk memastikan semua simbol/karakter yang masih berada di stack setelah semua persamaan terbaca, akan dikeluarkan dan dipindahkan ke postfix.

10. Buat variabel total untuk menghitung banyaknya karaketer pada variabel Q.

```
11
12         int total = Q.length();
```

11. Lakukan instansiasi objek dengan nama **post** dan nilai parameternya adalah total. Kemudian panggil method **konversi** untuk melakukan konversi notasi infix Q menjadi notasi postfix P.

```
14         Postfix15 post = new Postfix15(total);
15         P = post.konversi(Q);
16         System.out.println("Postfix : " + P);
17     }
18 }
```

12. Compile dan jalankan class **PostfixMain** dan amati hasilnya.

7.4.2. Verifikasi Hasil Percobaan

```
Masukkan ekspresi matematika (infix) :
a+b*(c+d-e)/f
Postfix : abcd+e-*f/+
```

7.4.3. Pertanyaan

1. Perhatikan class **Postfix**, jelaskan alur kerja method **derajat**!

   Method derajat menghitung derajat dari suatu operator yang diinputkan, tetapi ia digunakan dalam method konversi untuk mengonversi infix expression menjadi postfix expression.

2. Apa fungsi kode program berikut?

   ```
   c = Q.charAt(i);
   ```

   Untuk mengambil karakter pada posisi i dari string Q, dan menyimpannya dalam variabel c untuk diproses selanjutnya.

3. Jalankan kembali program tersebut, masukkan ekspresi **5*4^(1+2)%3**. Tampilkan hasilnya!

   ```
   Masukkan ekspresi matematika (infix) :
   5*4^(1+2)%3
   Postfix : 5412+^*3%
   ```

4. Pada soal nomor 3, mengapa tanda kurung tidak ditampilkan pada hasil konversi? Jelaskan!

   Karena tanda kurung bukan merupakan derajat operator aritmatika.

7.5. Tugas

1. Perhatikan dan gunakan kembali kode program pada **Praktikum 2**. Tambahkan method **getMax** pada class **Stack** yang digunakan untuk mencari dan menampilkan data pakaian dengan harga tertinggi dari semua data pakaian yang tersimpan di dalam stack!

   ```
   public void getMax() {
       if (isEmpty()) {
           System.out.println("Stack masih kosong");
       } else {
           Pakaian15 maxPakaian = data[0];

           for (int i = 1; i <= top; i++) {
               if (data[i].harga > maxPakaian.harga) {
                   maxPakaian = data[i];
               }
           }

           System.out.println("Data pakaian dengan harga tertinggi: " + maxPakaian.jenis + " " + maxPakaian.warna + " " + maxPakaian.merk + " " + maxPakaian.ukuran + " " + maxPakaian.harga);
       }
   }
   ```

   ```
   case 5:
       stk.getMax();
       break;
   ```

2. Setiap hari Minggu, Dewi pergi berbelanja ke salah satu supermarket yang berada di area rumahnya. Setiap kali selesai berbelanja, Dewi menyimpan struk belanjaannya di dalam laci. Setelah dua bulan, ternyata Dewi sudah mempunyai delapan struk belanja. Dewi berencana mengambil lima struk belanja untuk ditukarkan dengan voucher belanja.

   Buat sebuah program stack untuk menyimpan data struk belanja Dewi, kemudian lakukan juga proses pengambilan data struk belanja sesuai dengan jumlah struk yang akan ditukarkan dengan voucher. Informasi yang tersimpan pada struk belanja terdiri dari:

- Nomor transaksi
- Tanggal pembelian
- Jumlah barang yang dibeli
- Total harga bayar

Tampilkan informasi struk belanja yang masih tersimpan di dalam stack!

```java
public class Struk15 {
    int noTransaksi, jumlahBarang;
    String tglBeli;
    int totalBayar;
    int size;
    int top;
    Struk15 data[];
    Struk15[] stk;

    Struk15(int no, String tgl, int jb, int tb){
        noTransaksi = no;
        tglBeli = tgl;
        jumlahBarang = jb;
        totalBayar = tb;
    }

    public Struk15(int size){
        this.size = size;
        data = new Struk15[size];
        top = -1;
    }

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

    public boolean isFull(){
        return top == size - 1;
    }


    public void push(Struk15 dt){
        if (!isFull()) {
            top++;
            data[top] = dt;
        } else {
            System.out.println("Isi Stack Penuh!");
        }
    }

    public void pop(){
        if (!isEmpty()) {
            Struk15 x = data[top];
            top--;
            System.out.println("Data yang keluar: " + x.noTransaksi + " " + x.tglBeli + " " + x.jumlahBarang + " " + x.totalBayar + " ");
        } else {
            System.out.println("Stack masih kosong");
        }
    }

    public void peek(){
        System.out.println("Elemen teratas: " + data[top].noTransaksi + " " + " " + data[top].tglBeli + " " + data[top].jumlahBarang + " " + data[top].totalBayar);
    }

    public void print(){
        System.out.println("Isi stack: ");
        for (int i = top; i >= 0; i--) {
            System.out.println(data[i].noTransaksi + " " + data[i].tglBeli + " " + data[i].jumlahBarang + " " + data[i].totalBayar + " ");
        }
        System.out.println("");
    }

    public void clear(){
        if (!isEmpty()) {
            for (int i = top; i >= 0; i--) {
                top--;
            }
            System.out.println("Stack sudah dikosongkan");
        } else {
            System.out.println("Gagal! Stack masih kosong");
        }
    }
}
```

```java
import java.util.Scanner;

public class StrukMain15 {
    Scanner sc = new Scanner(System.in);

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Struk15 stk = new Struk15(20);
        StrukMain15 main = new StrukMain15();

        int pil;
        while (true) {
            System.out.println("\nPilih Operasi:");
            System.out.println("1. Push");
            System.out.println("2. Pop");
            System.out.println("3. Peek");
            System.out.println("4. Print");
            System.out.println("5. Kupon");
            System.out.println("6. Exit");
            System.out.print("Masukkan pilihan: ");
            pil = sc.nextInt();
            switch (pil) {
                case 1:
                    main.pushPakaian(stk);
                    break;
                case 2:
                    stk.pop();
                    break;
                case 3:
                    stk.peek();
                    break;
                case 4:
                    stk.print();
                    break;
                case 5:
                    main.kupon(stk);
                    break;
                case 6:
                    return;
                default:
                    System.out.println("Pilihan tidak tersedia. ");
                    break;
            }
        }
    }

    public void pushPakaian(Struk15 stk) {
        char pilih;
        do {
            System.out.print("No Belanja: ");
            int noTra = sc.nextInt();
            System.out.print("Tanggal (dd/mm/yyyy) : ");
            String tanggal = sc.next();
            System.out.print("Jumlah: ");
            int jumlah = sc.nextInt();
            System.out.print("Total Bayar: ");
            int total = sc.nextInt();

            Struk15 d = new Struk15(noTra, tanggal, jumlah, total);

            System.out.print("Apakah Anda akan menambahkan data baru ke stack (y/n) ? ");
            pilih = sc.next().charAt(0);
            stk.push(d);
        } while (pilih == 'y');
    }

    public void kupon(Struk15 stk){
        System.out.println("Data yang diambil");
        int i = 0;
        while (i < 5) {
            stk.pop();
            i++;
        }

        System.out.println();
        System.out.println("Data yang tersisa");
        stk.print();
    }
}
```