Jobsheet 8 Stacking

Satria Wiguna/ Ti 1d / Absen 26

Percobaan 1 :

Barang26:

```
public class Barang26 {
    int kode;
    String barang,kategori;

    Barang26(int kode, String barang, String kategori) {
        this.kode = kode;
        this.barang = barang;
        this.kategori = kategori;
}
}
```

Gudang26:

```
Click here to ask Blackbox to help you code faster
public class Gudang26 {

    Barang26[] tumpukan ;
    int size;
    int top;

    public Gudang26(int kapasitas) {
        size = kapasitas;
        tumpukan = new Barang26[size];
        top = -1;

    }

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

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

    public void tambahbarang(Barang26 brg){
        if (!cekpenuh()){
            top++;
            tumpukan[top] = brg;
            System.out.println("Barang " + brg.barang + " Berhasil ditambahkan ke Gudang");
        } else {
            System.out.println(x:"Gagal! Tumpukan barang di gudang sudah penuh");
        }
    }
```

```
public Barang26 ambilbarang() {
    if (!cekkosong()){
        Barang26 delete = tumpukan [top];
        top--;
        System.out.println("Barang " + delete.barang + " Diambil dari Gudang");
        return delete;
    }else{
        System.out.println(x:"Tumpukkan barang kosong");
        return null;
    }
}

public Barang26 lihatbarangteratas(){
    if (!cekkosong()){
        Barang26 barangteratas = tumpukan[top];
        System.out.println("Barang teratas : " +barangteratas.barang);
        return barangteratas;
    } else{
        System.out.println(x:"Tumpukan barang kosong");
        return null;
    }
}

public void tampilkanbarang(){
    if (!cekkosong()){
        System.out.println(x:"Rincian tumpukan barang di Gudang");
            for (int i = 0 ; i <= top ; i++){
                System.out.printf(format:"Kode %d: %s (kategori %s)\n", tumpukan[i].kode, tumpukan[i].barang, tumpukan[i].kategori);
            }
        }else {
            System.out.println(x:"Tumpukkan barang kosong");
        }
    }
}
```

Utama26:

```
import java.util.Scanner;
public class Utama26 {
    Run | Debug
    public static void main(String[] args) {
        Gudang26 gudang = new Gudang26(kapasitas:7);
        Scanner scanner = new Scanner (System.in);

        while (true){
            System.out.println(x:"\nmenu");
            System.out.println(x:"1. Tambah Barang");
            System.out.println(x:"2. Ambil Barang");
            System.out.println(x:"3. Tampilkan Tumpukan Barang");
            System.out.println(x:"4. Keluar");
            System.out.print(s:"Pilih Operasi: ");
            int pilihan = scanner.nextInt();
```

```
        switch (pilihan){
            case 1 :
            System.out.print(s:"Masukkan kode barang: ");
            int kode = scanner.nextInt();
            scanner.nextLine();
            System.out.print(s:"Masukkan nama barang: ");
            String nama = scanner.nextLine();
            System.out.print(s:"Masukkan nama kategori: ");
            String kategori = scanner.nextLine();
            Barang26 barangBaru = new Barang26(kode,nama,kategori);
            gudang.tambahbarang(barangBaru);
            break;

            case 2 :
                gudang.ambilbarang();
                break;
            case 3 :
                gudang.tampilkanbarang();
                break;
            case 4 :
                break;
                default:
                System.out.println(x:"Pilihan tidak valid . Silahkan coba lagi.");
        }
      }
     }


}

```

Vervikasi hasil :

```
menu
1. Tambah Barang
2. Ambil Barang
3. Tampilkan Tumpukan Barang
4. Keluar
Pilih Operasi: 1
Masukkan kode barang: 21
Masukkan nama barang: Majalah
Masukkan nama kategori: Buku
Barang Majalah Berhasil ditambahkan ke Gudang

menu
1. Tambah Barang
2. Ambil Barang
3. Tampilkan Tumpukan Barang
4. Keluar
Pilih Operasi: 1
Masukkan kode barang: 26
Masukkan nama barang: jaket
Masukkan nama kategori: Pakaian
Barang jaket Berhasil ditambahkan ke Gudang

menu
1. Tambah Barang
2. Ambil Barang
3. Tampilkan Tumpukan Barang
4. Keluar
Pilih Operasi: 2
Barang jaket Diambil dari Gudang
```

```
menu
1. Tambah Barang
2. Ambil Barang
3. Tampilkan Tumpukan Barang
4. Keluar
Pilih Operasi: 1
Masukkan kode barang: 33
Masukkan nama barang: Pizza
Masukkan nama kategori: Makanan
Barang Pizza Berhasil ditambahkan ke Gudang

menu
1. Tambah Barang
2. Ambil Barang
3. Tampilkan Tumpukan Barang
4. Keluar
Pilih Operasi: 3
Rincian tumpukan barang di Gudang
Kode 21: Majalah (kategori Buku)
Kode 33: Pizza (kategori Makanan)
```

2.1.3 Pertanyaan

1. Lakukan perbaikan pada kode program, sehingga keluaran yang dihasilkan sama dengan verifikasi hasil percobaan! Bagian mana saja yang perlu diperbaiki?
   perlu diperbaiki pada bagian metode tampilkanBarang, terdapat kesalahan dalam format string pada printf. Format %f digunakan untuk menampilkan tipe data float, sedangkan nama pada Barang10 bertipe data string.
2. Berapa banyak data barang yang dapat ditampung di dalam tumpukan? Tunjukkan potongan kode programnya!
   ```
   Gudang26 gudang = new Gudang26(kapasitas:7);
   ```
3. Mengapa perlu pengecekan kondisi **!cekKosong()** pada method **tampilkanBarang**? Kalau kondisi tersebut dihapus, apa dampaknya?
   untuk memastikan bahwa tumpukan tidak kosong sebelum mencoba menampilkan barang. Jika kondisi tersebut dihapus, maka program akan mencoba menampilkan barang bahkan ketika tumpukan kosong.
4. Modifikasi kode program pada class **Utama** sehingga pengguna juga dapat memilih operasi lihat barang teratas, serta dapat secara bebas menentukan kapasitas gudang!

Utama26:

```
import java.util.Scanner;
public class Utama26 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner (System.in);
        System.out.println("Masukkan Kapasitas gudang : ");
        int kapasitas = scanner.nextInt();
        Gudang26 gudang = new Gudang26(kapasitas);


     boolean ulang = true;
     while (ulang){
        System.out.println("\nmenu");
        System.out.println("1. Tambah Barang");
        System.out.println("2. Ambil Barang");
        System.out.println("3. Tampilkan Tumpukan Barang");
        System.out.println("4. Lihat Barang Teratas");
        System.out.println("5. Keluar");
        System.out.print("Pilih Operasi: ");
        int pilihan = scanner.nextInt();

        switch (pilihan){
            case 1 :
            System.out.print("Masukkan kode barang: ");
            int kode = scanner.nextInt();
            scanner.nextLine();
            System.out.print("Masukkan nama barang: ");
            String nama = scanner.nextLine();
            System.out.print("Masukkan nama kategori: ");
            String kategori = scanner.nextLine();
            Barang26 barangBaru = new Barang26(kode,nama,kategori);
            gudang.tambahbarang(barangBaru);
            break;

            case 2 :
                gudang.ambilbarang();
                break;
            case 3 :
                gudang.tampilkanbarang();
                break;
            case 4 :
                gudang.barangteratas();
            case 5 :
             ulang = false;
            default:
                System.out.println("Pilihan tidak valid . Silahkan coba lagi.");
        }
     }
    }


}
```

Gudang26:

```
void barangteratas (){
   if (!cekkosong()){
    System.out.println(x:"Barang teratas : ");
    System.out.printf(format:"%s (kategori %s) dengan kode %d \n" , tumpukan[top].barang , tumpukan[top].kategori , tumpukan[top].kode);
} else {
    System.out.println(x:"Tumpukkan barang kosong");
```

5. **Commit dan push kode program ke Github**

Percobaan 2

```
    public String konversidesimalkebiner(int kode ){
     StackKonversi26 stack = new StackKonversi26();
     while (kode > 0){
         int sisa = kode % 2;
         stack.push(sisa);
         kode = kode/2;
     }
     String biner = new String();
     while (!stack.isEmpty()){
         biner += stack.pop();
     }
     return biner;
    }

}
```

```java
public class StackKonversi26 {
    int size, top;
    int [] tumpukanBiner;
    StackKonversi26 (){
        this.size = 32;
        tumpukanBiner = new int[size];
        top = -1;
    }
    boolean isEmpty(){
        return top == -1;
    }
    boolean isfull(){
        return top == size - 1;
    }
    void push(int data){
        if (isfull()) {
            System.out.println(x:"Stack penuh");
        }else{
            top++;
            tumpukanBiner[top] = data;
        }
    }
    int pop(){
        if (isEmpty()) {
            System.out.println(x:"Stack Kosong");
            return -1;
        }else{
            int data = tumpukanBiner[top];
            top--;
            return data;
        }


}
}
```

```java
public Barang26 ambilbarang() {
    if (!cekkosong()){
        Barang26 delete = tumpukan [top];
        top--;
        System.out.println("Barang " + delete.barang + " Diambil dari Gudang");
        System.out.println("Kode unik dalam biner: " + konversidesimalkebiner(delete.kode));
        return delete;
    }else{
        System.out.println(x:"Tumpukkan barang kosong");
        return null;
```

```
menu
1. Tambah Barang
2. Ambil Barang
3. Tampilkan Tumpukan Barang
4. Lihat Barang Teratas
5. Keluar
Pilih Operasi: 1
Masukkan kode barang: 13
Masukkan nama barang: Setrika
Masukkan nama kategori: Elektronik
Barang Setrika Berhasil ditambahkan ke Gudang

menu
1. Tambah Barang
2. Ambil Barang
3. Tampilkan Tumpukan Barang
4. Lihat Barang Teratas
5. Keluar
Pilih Operasi: 2
Barang Setrika Diambil dari Gudang
Kode unik dalam biner: 1101
```

Pertanyaan

1. Pada method **konversiDesimalKeBiner**, ubah kondisi perulangan menjadi **while (kode != 0)**, bagaimana hasilnya? Jelaskan alasannya!

Setelah diubah menjadi while (kode != 0), perulangan akan dilakukan selama kode tidak sama dengan 0. dan hasilnya pun sama ketika menggunakan while (kode != 0)

2. Jelaskan alur kerja dari method **konversiDesimalKeBiner**!

- objek StackKonversi10 dibuat untuk menyimpan sisa pembagian bilangan desimal dengan 2, Selama kode tidak sama dengan 0 (while (kode != 0)):
- kemudian menghitung sisa pembagian kode dengan 2 (int sisa = kode % 2).
- lalu memasukkan sisa tersebut ke dalam stack menggunakan metode push. setelah itu membagi kode dengan 2 untuk memperoleh nilai kode baru yang akan digunakan pada iterasi berikutnya.
- Setelah semua sisa pembagian dimasukkan ke dalam stack, lakukan iterasi melalui stack untuk mengambil nilai sisa dan tambahkan ke string biner menggunakan StringBuilder.
- lalu mengembalikan string biner yang berisi representasi biner dari bilangan desimal.

Percobaan 3 :

Postfix26 :

```
public class Postfix26 {
    int n, top;
    char [] stack;

    Postfix26(int kapasitas) {
        n = kapasitas;
        stack = new char[n];
        top = -1;
        push (c:'(');
    }
    void push (char c){
        top++;
        stack [top] = c;
    }
    char pop (){
        char item = stack[top];
        top--;
        return item;
    }
    boolean isOperand(char c){
        if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == ' ' ||c =='.'){
            return true;
        }else{
            return false;
        }
    }
    boolean isOperator(char c){
        if(c == '^' || c == '%' || c == '/' || c == '*' || c == '-' || c == '+'){
            return true;
        }else {
            return false;
        }
    }
    int derajat (char c){
        switch (c) {
            case '^':
                return 3;
            case '%':
```

```
            return 2;
        case '/':
            return 2;
        case '*':
            return 2;
        case '+':
            return 1;
        case '-':
            return 2;
        default:
            return 0;
    }
}
String konversi (String Q){
    String p = "";
    char c;
    for (int i = 0; i < n; i++) {
        c = Q.charAt(i);
        if(isOperand(c)){
            p += c;
        }
        if ( c == '(') {
            push(c);
        }
        if (c == ')') {
            while (stack[top] != '(') {
                p += pop();
            }
            pop();
        }
        if (isOperator(c)) {
            while (derajat(stack[top])>= derajat(c)) {
                p += pop();
            }
            push(c);
        }
    }
    return p;
}

}
```

PostfixMain26 :

```java
import java.util.Scanner;
public class PostfixMain26 {
    Run | Debug
    public static void main(String[] args) {
        Scanner sc = new Scanner (System.in);

        String P, Q;
        System.out.println(x:"Masukkan Ekspresi Matematika (infix) :");
        Q = sc.nextLine();
        Q = Q.trim();
        Q = Q + ")";

        int total = Q.length();

        Postfix26 post = new Postfix26(total);
        P = post.konversi(Q);
        System.out.println("Postfix : "+P);

        sc.close();
    }
}
```

Pertanyaan

1. Pada method **derajat**, mengapa return value beberapa case bernilai sama? Apabila return value diubah dengan nilai berbeda-beda setiap case-nya, apa yang terjadi?

Operator dengan prioritas lebih tinggi harus dievaluasi terlebih dahulu. Jika mengubah return value menjadi angka yang berbeda untuk setiap kasus, urutan prioritas dalam mengevaluasi operator akan terganggu

Contoh kita mengubah menjadi

```
int derajat (char c){
    switch (c) {
        case '^':
            return 3;
        case '%':
            return 2;
        case '*':
            return 2;
        case '/':
            return 2;
        case '+':
            return 1;
        case '-':
            return 2;
        default:
            return 0;
    }
```

```
Masukkan Ekspresi Matematika (infix) :
a+b*(c+d-e)/f
Postfix : abcde-+*f/+
```

Hasil ahan terlihat menjadi

2. Jelaskan alur kerja method **konversi**!

- Method konversi mengubah ekspresi infix menjadi ekspresi postfix menggunakan stack, serta menginisialisasi string kosong p untuk menyimpan ekspresi postfix.
- Method melakukan looping untuk setiap karakter dalam ekspresi infix Q.
- Jika karakter adalah operand (huruf, digit, spasi, atau titik), karakter tersebut langsung ditambahkan ke string postfix p.
- Jika karakter adalah tanda kurung buka (, ia dipush ke stack.
- Jika karakter adalah tanda kurung tutup), operator-oprator dipop dari stack dan ditambahkan ke p sampai menemukan tanda kurung buka.
- Jika karakter adalah operator (^, %, /, *, -, +), ia membandingkan prioritasnya dengan operator teratas di stack.
- Jika prioritas operator saat ini lebih rendah atau sama dengan operator teratas di stack, operator teratas dipop dan ditambahkan ke p. Proses ini terus berlanjut sampai stack kosong atau operator teratas memiliki prioritas yang lebih rendah. Kemudian, operator saat ini dipush ke stack.

- Setelah semua karakter dalam Q diproses, operator-operator yang tersisa di stack dipop dan ditambahkan ke p untuk melengkapi ekspresi postfix.
- Method mereturn value postfix p.

3. Pada method konversi, apa fungsi dari potongan kode berikut?

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

kode Q.length() berfungsi untuk mendapatkan panjang dari string Q, Panjang string ini digunakan untuk menginisialisasi variabel total, yang nantinya digunakan untuk mengembalikan jumlah karakter dalam string Q,

Latihan Praktikum
Waktu : 60 Menit
Perhatikan dan gunakan kembali kode program pada Percobaan 1. Tambahkan dua method berikut pada class Gudang:
• Method **lihatBarangTerbawah** digunakan untuk mengecek barang pada tumpukan terbawah

```
    void barangTerbawah (){
     if (!cekkosong()) {
         System.out.println(x:"Barang Terbawah adalah:");
         System.out.printf(format:"%s (kategori %s) dengan kode %d\n", tumpukan[0].barang, tumpukan[0].kategori, tumpukan[0].kode);
     }else{
         System.out.println(x:"Tumpukan Barang Kosong");
     }
 }

 void cariBarangnama (String nama){
     if (!cekkosong()){
         for (int i = 0; i < size; i++) {
             if (tumpukan[i].barang.equalsIgnoreCase(nama)) {
                 System.out.printf(format:"%s (kategori %s) dengan kode %d\n", tumpukan[i].barang, tumpukan[i].kategori, tumpukan[i].kode);
             }
         }
     }else{
         System.out.println(x:"Tumpukan Barang Kosong");
     }

 }
 void cariBarangkode (int kode){
     if (!cekkosong()) {
         for (int i = 0; i < size; i++) {
             if (tumpukan[i].kode == kode) {
                 System.out.printf(format:"%s (kategori %s) dengan kode %d\n", tumpukan[i].barang, tumpukan[i].kategori, tumpukan[i].kode);
             }
         }
     }else {
         System.out.println(x:"Tumpukan Barang Kosong");
     }

 }
```

• Method **cariBarang** digunakan untuk mencari ada atau tidaknya barang berdasarkan kode barangnya atau nama barangnya

```java
import java.util.Scanner;
public class Utama26 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner (System.in);

        System.out.print("Masukkan kapasitas Gudang: ");
        int kapasitas = scanner.nextInt();
        Gudang26 gudang = new Gudang26(kapasitas);

        boolean ulang = true;
        while (ulang) {
            System.out.println("====================================");
            System.out.println("\t Menu :");
            System.out.println("1. Tambah Barang");
            System.out.println("2. Ambil Barang");
            System.out.println("3. Tampilan tumpukan Barang");
            System.out.println("4. Lihat Barang");
            System.out.println("5. Keluar");
            System.out.print("Pilih Operasi :");
            int pilihan = scanner.nextInt();

            switch (pilihan) {
                case 1:
                    System.out.print("Masukkan Kode Barang : ");
                    int kode = scanner.nextInt();
                    System.out.print("Masukkan nama Barang : ");
                    String nama = scanner.next();
                    System.out.print("Masukkan nama Kategori : ");
                    String kategori = scanner.next();
                    Barang26 barangbaru = new Barang26(kode, nama, kategori);
                    gudang.tambahbarang(barangbaru);
                    break;
                case 2:
                    gudang.ambilbarang();
                    break;
                case 3:
                    gudang.tampilkanbarang();
                    break;
                case 4:
                    System.out.println();
                    System.out.println("Pilih Barang berdasarkan : ");
                    System.out.println("1. Teratas");
                    System.out.println("2. Terbawah");
                    System.out.println("3. Nama Barang");
                    System.out.println("4. Kode Barang");
                    System.out.println("5. Keluar");
                    System.out.print("Pilih : ");
                    byte plh = scanner.nextByte();
                    if (plh != 5) {
                        switch (plh) {
                            case 1:
                                gudang.barangteratas();
                                break;
                            case 2:
                                gudang.barangTerbawah();
                                break;
                            case 3:
                                System.out.print("Masukkan Nama Barang : ");
                                String nm = scanner.next();
                                gudang.cariBarangnama(nm);
                                break;
                            case 4:
                                System.out.print("Masukkan Kode Barang : ");
                                int kd = scanner.nextInt();
                                gudang.cariBarangkode(kd);
                                break;
                        }
                    }else{
                            System.out.println("Pilihan Tidak Valid ");
                    }
                    break;
                case 5:
                    ulang = false;
                default:
                    System.out.println("Pilihan tidak valid. Silahkan coba lagi");
                    break;
            }
        }
        scanner.close();
    }
}
```

```
Masukkan kapasitas Gudang: 4
=====================================
         Menu :
1. Tambah Barang
2. Ambil Barang
3. Tampilan tumpukan Barang
4. Lihat Barang
5. Keluar
Pilih Operasi :1
Masukkan Kode Barang : 134
Masukkan nama Barang : Sapi
Masukkan nama Kategori : Hewan
Barang Sapi Berhasil ditambahkan ke Gudang
=====================================
         Menu :
1. Tambah Barang
2. Ambil Barang
3. Tampilan tumpukan Barang
4. Lihat Barang
5. Keluar
Pilih Operasi :1
Masukkan Kode Barang : 234
Masukkan nama Barang : kambing
Masukkan nama Kategori : Hewan
Barang kambing Berhasil ditambahkan ke Gudang
=====================================
```

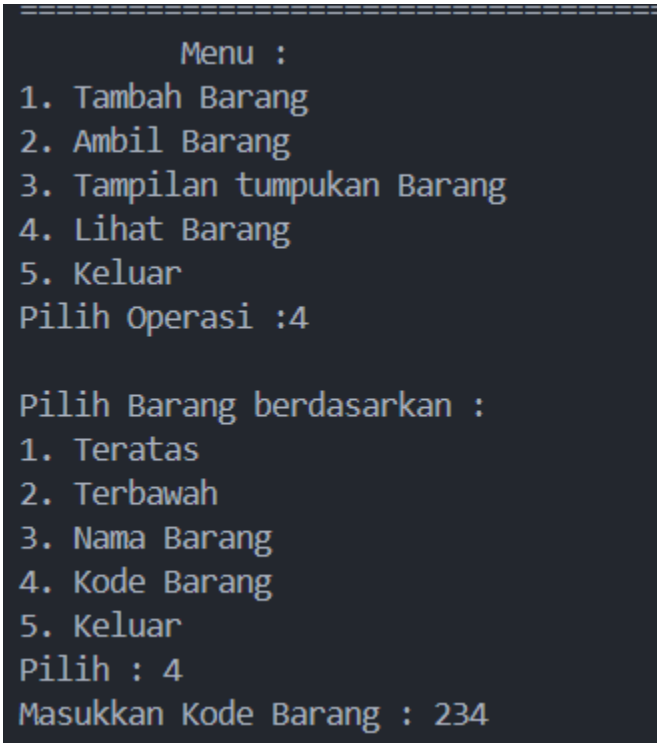
Berdasarkan terbawah

```
         Menu :
1. Tambah Barang
2. Ambil Barang
3. Tampilan tumpukan Barang
4. Lihat Barang
5. Keluar
Pilih Operasi :4

Pilih Barang berdasarkan :
1. Teratas
2. Terbawah
3. Nama Barang
4. Kode Barang
5. Keluar
Pilih : 2
Barang Terbawah adalah:
Sapi (kategori Hewan) dengan kode 134
=====================================
```

Berdasarkan nama

```
         Menu :
1. Tambah Barang
2. Ambil Barang
3. Tampilan tumpukan Barang
4. Lihat Barang
5. Keluar
Pilih Operasi :4

Pilih Barang berdasarkan :
1. Teratas
2. Terbawah
3. Nama Barang
4. Kode Barang
5. Keluar
Pilih : 3
Masukkan Nama Barang : Sapi
Sapi (kategori Hewan) dengan kode 134
```

Berdasarkan kode :



Link github :

https://github.com/AuroraSauces/Praktikum-algoritma-dan-sistem-data/tree/main