

# Jobsheet 7

## Praktikum Algoritma & Struktur Data



Rafi Ody Prasetyo  
(2341720180)

D-IV Teknik Informatika  
Politeknik Negeri Malang  
Semester 2  
2024

## SS Hasil Percobaan 1

```
Menu:
1. Tambah Barang
2. Ambil Barang
3. Tampilkan tumpukan Barang
4. Lihat Barang teratas
0. Keluar
Pilih operasi: 1
Masukkan kode barang: 21
Masukkan nama barang: Majalah
Masukkan kategori: Buku
Barang Majalah berhasil ditambahkan ke gudang.
```

```
Menu:
1. Tambah Barang
2. Ambil Barang
3. Tampilkan tumpukan Barang
4. Lihat Barang teratas
0. Keluar
Pilih operasi: 1
Masukkan kode barang: 26
Masukkan nama barang: Jaket
Masukkan kategori: Pakaian
Barang Jaket berhasil ditambahkan ke gudang.
```

```
Menu:
1. Tambah Barang
2. Ambil Barang
3. Tampilkan tumpukan Barang
4. Lihat Barang teratas
0. Keluar
Pilih operasi: 2
Barang Jaket berhasil diambil dari gudang
```

```
Menu:
1. Tambah Barang
2. Ambil Barang
3. Tampilkan tumpukan Barang
4. Lihat Barang teratas
0. Keluar
Pilih operasi: 1
Masukkan kode barang: 33
Masukkan nama barang: Pizza
Masukkan kategori: Makanan
Barang Pizza berhasil ditambahkan ke gudang.
```

```
Menu:
1. Tambah Barang
2. Ambil Barang
3. Tampilkan tumpukan Barang
4. Lihat Barang teratas
0. Keluar
Pilih operasi: 3
Rincian tumpukkan barang di gudang:
Kode 21: Majalah (Kategori Buku)
Kode 33: Pizza (Kategori Makanan)
```

## Pertanyaan Percobaan 1

1. Lakukan perbaikan pada kode program, sehingga keluaran yang dihasilkan sama dengan verifikasi hasil percobaan! Bagian mana saja yang perlu diperbaiki?

Jawab:

- Menambahkan konstruktor berparameter pada class Barang21, agar dapat menginputkan data barang.

```
public Barang21(int kode, String nama, String kategori) {  
    this.kode = kode;  
    this.nama = nama;  
    this.kategori = kategori;  
}
```

- Menubah !isEmpty menjadi !cekKosong pada method lihatBarangTeratas dikarenakan tidak terdapat method isEmpty.

Before:

```
public Barang lihatBarangTeratas() {  
    if (!isEmpty()) {  
        Barang barangTeratas = tumpukan[top];  
        System.out.println("Barang teratas: " + barangTeratas.nama);  
        return barangTeratas;  
    }
```

After:

```
public Barang21 lihatBarangTeratas()  
{  
    if (!cekKosong()) {  
        Barang21 barangTeratas = tumpukan[top];  
        System.out.println("Barang teratas: " + barangTeratas.nama);  
        return barangTeratas;  
    } else {  
        return null;  
    }  
}
```

2. Berapa banyak data barang yang dapat ditampung di dalam tumpukan?

Tunjukkan potongan kode programnya!

Jawab:

Pada class Gudang21 terdapat konstruktor yang memiliki parameter kapasitas, kemudian konstruktor tersebut dipanggil pada main dan parameter kapasitas tersebut wajib diisi. Di dalam class main sendiri kapasitas memiliki nilai 7, sehingga banyak data yang ditampung adalah 7.

Konstruktor berparameter kapasitas:

```
public Gudang21(int kapasitas)
{
    size = kapasitas;
    tumpukan = new Barang21[size];
    top = -1;
}
```

Kode yang berisi nilai banyaknya data:

```
Gudang21 gudang = new Gudang21(kapasitas:7);
```

3. Mengapa perlu pengecekan kondisi !cekKosong() pada method **tampilkanBarang?** Kalau kondisi tersebut dihapus, apa dampaknya?

**Jawab:**

Pengecekan kondisi !cekKosong pada method **tampilkanBarang()** diperlukan untuk memastikan bahwa tumpukan barang tidak kosong sebelum mencoba untuk menampilkan barang-barangnya. Dampak ketika kita hapus kondisi tersebut adalah program akan mencoba melakukan iterasi melalui tumpukkan array dan mencoba untuk mengakses atribut dari **Barang21**.

4. Modifikasi kode program pada class **Utama** sehingga pengguna juga dapat memilih operasi lihat barang teratas, serta dapat secara bebas menentukan kapasitas gudang!

**Jawab:**

Code:

```
System.out.print(s:"Masukkan kapasitas gudang: ");
int kapasitas = scanner.nextInt();
Gudang21 gudang = new Gudang21(kapasitas);
```

```
case 4:
    gudang.lihatBarangTeratas();
    break;
```

Output:

```

0. Keluar
Pilih operasi: 1
Masukkan kode barang: 1
Masukkan nama barang: Kemeja
Masukkan kategori: Pakaian
Barang Kemeja berhasil ditambahkan ke gudang.

Menu:
1. Tambah Barang
2. Ambil Barang
3. Tampilkan tumpukan Barang
4. Lihat Barang teratas
0. Keluar
Pilih operasi: 1
Masukkan kode barang: 4
Masukkan nama barang: Cincin
Masukkan kategori: Aksesoris
Barang Cincin berhasil ditambahkan ke gudang.

Menu:
1. Tambah Barang
2. Ambil Barang
3. Tampilkan tumpukan Barang
4. Lihat Barang teratas
0. Keluar
Pilih operasi: 4
Barang teratas: Cincin

```

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

<https://github.com/rafiody16/Praktikum-Algoritma-dan-Struktur-Data--smt-2-/tree/main/Jobsheet%207/Percobaan%201>

## SS Hasil Percobaan 2

```

Menu:
1. Tambah Barang
2. Ambil Barang
3. Tampilkan tumpukan Barang
4. Lihat Barang teratas
0. Keluar
Pilih operasi: 1
Masukkan kode barang: 13
Masukkan nama barang: Setrika
Masukkan kategori: ELEktronik
Barang Setrika berhasil ditambahkan ke gudang.

```

```

Menu:
1. Tambah Barang
2. Ambil Barang
3. Tampilkan tumpukan Barang
4. Lihat Barang teratas
0. Keluar
Pilih operasi: 2
Barang Setrika berhasil diambil dari gudang
Kode unik dari biner: 1101

```

## Pertanyaan Percobaan 2

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

**Jawab:**

Hasilnya akan tetap sama. Alasan utama mengapa menggunakan kondisi while (kode != 0) lebih fleksibel adalah karena itu memungkinkan penanganan nilai kode yang negatif.

2. Jelaskan alur kerja dari method konversiDesimalKeBiner!

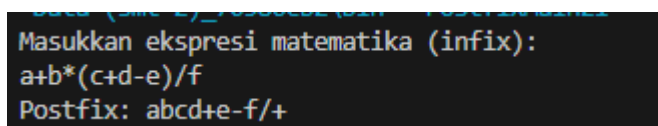
**Jawab:**

Program akan melakukan operasi modulo 2 dengan variabel int kode, kemudian hasil sisa modulo 1/0 akan disimpan di stack sedangkan hasil modulo akan disimpan di variabel kode. Dan program akan terus melakukan iterasi hingga variabel kode bernilai 0.

3. Push & commit github.

<https://github.com/rafiody16/Praktikum-Algoritma-dan-Struktur-Data--smt-2-/tree/main/Jobsheet%207/Percobaan%202>

## SS Hasil Percobaan 3



```
data (smt 2)_Percobaan2 (infix -> postfix)
Masukkan ekspresi matematika (infix):
a+b*(c+d-e)/f
Postfix: abcd+e-f/+
```

## Pertanyaan Percobaan 3

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?

**Jawab:**

Return value bernilai sama yang menentukan derajatnya berarti sama, jika setiap nilai return berbeda, susah untuk mengelompokkannya untuk membandingkan inputan char dan operasi.

2. Jelaskan alur kerja method konversi!

**Jawab:**

- Jika 'c' operand maka, menambahkan c ke P.
- Jika '(', panggil fungsi push( c ) untuk memasukannya ke stack.
- Jika ')', melakukan iterasi selama karakter teratas stack bukan '(', mengeluarkan karakter teratas dari stack menggunakan pop() dan menambahkannya ke P.
- Jika 'c' operator maka, melakukan iterasi selama prioritas lebih besar sama dengan antara derajat c dengan derajat stack teratas dan mengeluarkan karakter teratas dari stack menggunakan pop() dan tambahkan ke P. Setelah itu memasukkan karakter c ke stack.
- Akhirnya mengembalikan nilai P.

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

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

#### Jawab:

Potongan kode tersebut digunakan untuk mengambil karakter pada posisi indeks ke-i dari string Q dan menyimpannya ke dalam variabel c.

### 4. Push & Commit Github

<https://github.com/rafiody16/Praktikum-Algoritma-dan-Struktur-Data--smt-2-/tree/main/Jobsheet%207/Percobaan%203>

## Latihan Praktikum

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.
- Method cariBarang digunakan untuk mencari ada atau tidaknya barang berdasarkan kode barangnya atau nama barangnya.

Source Code:

<https://github.com/rafiody16/Praktikum-Algoritma-dan-Struktur-Data--smt-2-/tree/main/Jobsheet%207/Latihan%20Praktikum>

Output:

Input data barang:

```
Masukkan kapasitas gudang: 2

Menu:
1. Tambah Barang
2. Ambil Barang
3. Tampilkan tumpukan Barang
4. Lihat Barang teratas
5. Lihat barang terbawah
6. Cari
0. Keluar
Pilih operasi: 1
Masukkan kode barang: 12
Masukkan nama barang: Pensil
Masukkan kategori: Alat Tulis
Barang Pensil berhasil ditambahkan ke gudang.

Menu:
1. Tambah Barang
2. Ambil Barang
3. Tampilkan tumpukan Barang
4. Lihat Barang teratas
5. Lihat barang terbawah
6. Cari
0. Keluar
Pilih operasi: 1
Masukkan kode barang: 4
Masukkan nama barang: Pisau
Masukkan kategori: Alat Masak
Barang Pisau berhasil ditambahkan ke gudang.
```

Lihat barang terbawah:

```
Menu:
1. Tambah Barang
2. Ambil Barang
3. Tampilkan tumpukan Barang
4. Lihat Barang teratas
5. Lihat barang terbawah
6. Cari
0. Keluar
Pilih operasi: 5
Barang teratas: Pensil
```

Mencari barang berdasarkan kode dan nama barang:

<pre>Menu: 1. Tambah Barang 2. Ambil Barang 3. Tampilkan tumpukan Barang 4. Lihat Barang teratas 5. Lihat barang terbawah 6. Cari 0. Keluar Pilih operasi: 6 Masukan Kode Barang atau Nama Barang: 4 Barang dengan kode 4 dan nama Pisau ditemukan di Gudang</pre>	<pre>Menu: 1. Tambah Barang 2. Ambil Barang 3. Tampilkan tumpukan Barang 4. Lihat Barang teratas 5. Lihat barang terbawah 6. Cari 0. Keluar Pilih operasi: 6 Masukan Kode Barang atau Nama Barang: Pensil Barang dengan kode 12 dan nama Pensil ditemukan di Gudang</pre>
--	---



