JOBSHEET IX
STACK

1. Tujuan Praktikum

Setelah melakukan materi praktikum ini, mahasiswa mampu:

1. Membuat struktur data Stack
2. Menerapkan algoritma Stack ke dalam program Java

2. Praktikum

2.1 Percobaan 1: Mahasiswa Mengumpulkan Tugas

Waktu Percobaan : 90 Menit

Sejumlah mahasiswa mengumpulkan berkas tugas di meja dosen secara ditumpuk dengan menerapkan prinsip stack. Dosen melakukan penilaian secara terurut mulai dari berkas tugas teratas. Perhatikan Class Diagram Mahasiswa berikut.

Mahasiswa<NoAbsen>
nim: String nama: String kelas: String nilai: int
Mahasiswa<NoAbsen>() Mahasiswa<NoAbsen>(nim: String, nama: String, kelas: String) tugasDinilai(nilai: int)

Selanjutnya, untuk mengumpulkan berkas tugas, diperlukan class StackTugasMahasiswa yang berperan sebagai Stack tempat menyimpan data tugas mahasiswa. Atribut dan method yang terdapat di dalam class StackTugasMahasiswa merepresentasikan pengolahan data menggunakan struktur Stack. Perhatikan Class Diagram StackTugasMahasiswa berikut.

StackTugasMahasiswa<NoAbsen>
stack: Mahasiswa[] size: int top: int
StackTugasMahasiswa<NoAbsen>(size: int) isFull(): boolean isEmpty(): boolean push(mhs): void pop(): Mahasiswa peek(): Mahasiswa print(): void

***Catatan**: Tipe data pada variabel **stack** menyesuaikan dengan data yang akan disimpan di dalam Stack. Pada percobaan ini, data yang akan disimpan merupakan array of object dari **Mahasiswa**, sehingga tipe data yang digunakan adalah **Mahasiswa**.*

Berdasarkan dua class diagram tersebut, program menggunakan bahasa Java.

2.1.1 Langkah-langkah Percobaan

A. Class Mahasiswa

1. Buat folder baru bernama **Jobsheet9** di dalam repository **Praktikum ASD**. Buat file baru, beri nama **Mahasiswa<NoAbsen>.java**
2. Lengkapi class **Mahasiswa** dengan atribut yang telah digambarkan di dalam class diagram Mahasiswa, yang terdiri dari atribut **nama**, **nim**, **kelas**, dan **nilai**
3. Tambahkan konstruktor berparameter pada class Mahasiswa sesuai dengan class diagram Mahasiswa. Berikan nilai default nilai = -1 sebagai nilai awal ketika tugas belum dinilai

```
Mahasiswa(String nama, String nim, String kelas) {
    this.nama = nama;
    this.nim = nim;
    this.kelas = kelas;
    nilai = -1;
}
```

4. Tambahkan method **tugasDinilai()** yang digunakan untuk mengeset nilai ketika dilakukan penilaian tugas mahasiswa

```
void tugasDinilai(int nilai) {
    this.nilai = nilai;
}
```

B. Class StackTugasMahasiswa

5. Setelah membuat class Mahasiswa, selanjutnya perlu dibuat class **StackTugasMahasiswa<NoAbsen>.java** sebagai tempat untuk mengelola tumpukan tugas. Class StackTugasMahasiswa merupakan penerapan dari struktur data Stack
6. Lengkapi class **StackTugasMahasiswa** dengan atribut yang telah digambarkan di dalam class diagram StackTugasMahasiswa, yang terdiri dari atribut **stack**, **size**, dan **top**

```
Mahasiswa[] stack;
int top;
int size;
```

7. Tambahkan konstruktor berparameter pada class StackTugasMahasiswa untuk melakukan inisialisasi kapasitas maksimum data tugas mahasiswa yang dapat disimpan di dalam Stack, serta mengeset indeks awal dari pointer **top**

```
public StackTugasMahasiswa(int size) {
    this.size = size;
    stack = new Mahasiswa[size];
    top = -1;
}
```

8. Selanjutnya, buat method **isFull** bertipe boolean untuk mengecek apakah tumpukan tugas mahasiswa sudah terisi penuh sesuai kapasitas

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

9. Pada class StackTugasMahasiswa, buat method **isEmpty** bertipe boolean untuk mengecek apakah tumpukan tugas masih kosong

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

10. Untuk dapat menambahkan berkas tugas ke dalam tumpukan Stack, maka buat method **push**. Method ini menerima parameter **mhs** yang berupa object dari class **Mahasiswa**

```
public void push(Mahasiswa mhs) {
    if (!isFull()) {
        top++;
        stack[top] = mhs;
    } else {
        System.out.println("Stack penuh! Tidak bisa menambahkan tugas lagi.");
    }
}
```

11. Penilaian tugas mahasiswa yang dilakukan oleh dosen dilakukan dengan menggunakan method **pop** untuk mengeluarkan tugas yang akan dinilai. Method ini tidak menerima parameter apapun namun mempunyai nilai kembalian berupa object dari class Mahasiswa

```
public Mahasiswa pop() {
    if (!isEmpty()) {
        Mahasiswa m = stack[top];
        top--;
        return m;
    } else {
        System.out.println("Stack kosong! Tidak ada tugas untuk dinilai.");
        return null;
    }
}
```

Catatan: Apabila diperlukan informasi mengenai data mahasiswa yang diambil, maka tipe kembalian harus berupa object Mahasiswa. Sebaliknya, tipe kembalian void dapat digunakan jika data mahasiswa yang dikeluarkan tidak akan diolah atau digunakan lagi

12. Buat method **peek** untuk dapat mengecek tumpukan tugas mahasiswa yang berada di posisi paling atas

```
public Mahasiswa peek() {
    if (!isEmpty()) {
        return stack[top];
    } else {
        System.out.println("Stack kosong! Tidak ada tugas yang dikumpulkan");
        return null;
    }
}
```

13. Tambahkan method **print** untuk dapat menampilkan semua daftar tugas mahasiswa pada Stack

```
public void print() {
    for (int i = 0; i <= top; i++) {
        System.out.println(stack[i].nama + "\t" + stack[i].nim + "\t" + stack[i].kelas);
    }
    System.out.println("");
}
```

C. Class Utama

14. Buat file baru, beri nama **MahasiswaDemo<NoAbsen>.java**
15. Tuliskan struktur dasar bahasa pemrograman Java yang terdiri dari fungsi **main**
16. Di dalam fungsi **main,** lakukan instansiasi object StackTugasMahasiswa bernama **stack** dengan nilai parameternya adalah 5.

```
StackTugasMahasiswa stack = new StackTugasMahasiswa(5);
```

17. Deklarasikan Scanner dengan nama variabel **scan** dan variabel **pilih** bertipe int
18. Tambahkan menu untuk memfasilitasi pengguna dalam memilih operasi Stack dalam mengelola data tugas mahasiswa menggunakan struktur perulangan **do-while**

```
do {
    System.out.println("\nMenu:");
    System.out.println("1. Mengumpulkan Tugas");
    System.out.println("2. Menilai Tugas");
    System.out.println("3. Melihat Tugas Teratas");
    System.out.println("4. Melihat Daftar Tugas");
    System.out.print("Pilih: ");
    pilih = scan.nextInt();
    scan.nextLine();
    switch (pilih) {
        case 1:
            System.out.print("Nama: ");
            String nama = scan.nextLine();
            System.out.print("NIM: ");
            String nim = scan.nextLine();
            System.out.print("Kelas: ");
            String kelas = scan.nextLine();
            Mahasiswa mhs = new Mahasiswa(nama, nim, kelas);
            stack.push(mhs);
            System.out.printf("Tugas %s berhasil dikumpulkan\n", mhs.nama);
            break;
        case 2:
            Mahasiswa dinilai = stack.pop();
            if (dinilai != null) {
                System.out.println("Menilai tugas dari " + dinilai.nama);
                System.out.print(s:"Masukkan nilai (0-100): ");
                int nilai = scan.nextInt();
                dinilai.tugasDinilai(nilai);
                System.out.printf(format:"Nilai Tugas %s adalah %d\n", dinilai.nama, nilai);
            }
            break;
        case 3:
            Mahasiswa lihat = stack.peek();
            if (lihat != null) {
                System.out.println("Tugas terakhir dikumpulkan oleh " + lihat.nama);
            }
            break;
        case 4:
            System.out.println(x:"Daftar semua tugas");
            System.out.println(x:"Nama\tNIM\tKelas");
            stack.print();
            break;
        default:
            System.out.println(x:"Pilihan tidak valid.");
    }
} while (pilih >= 1 && pilih <= 4);
```

19. **Commit dan push kode program ke Github**
20. Compile dan run program.

2.1.2 Verifikasi Hasil Percobaan

Cocokkan hasil compile kode program Anda dengan gambar berikut ini.

```
Menu:
1. Mengumpulkan Tugas
2. Menilai Tugas
3. Melihat Tugas Teratas
4. Melihat Daftar Tugas
Pilih: 1
Nama: Dila
NIM: 1001
Kelas: 1A
Tugas Dila berhasil dikumpulkan

Menu:
1. Mengumpulkan Tugas
2. Menilai Tugas
3. Melihat Tugas Teratas
4. Melihat Daftar Tugas
Pilih: 1
Nama: Erik
NIM: 1002
Kelas: 1B
Tugas Erik berhasil dikumpulkan

Menu:
1. Mengumpulkan Tugas
2. Menilai Tugas
3. Melihat Tugas Teratas
4. Melihat Daftar Tugas
Pilih: 3
Tugas terakhir dikumpulkan oleh Erik

Menu:
1. Mengumpulkan Tugas
2. Menilai Tugas
3. Melihat Tugas Teratas
4. Melihat Daftar Tugas
Pilih: 1
Nama: Tika
NIM: 1003
Kelas: 1C
Tugas Tika berhasil dikumpulkan

Menu:
1. Mengumpulkan Tugas
2. Menilai Tugas
3. Melihat Tugas Teratas
4. Melihat Daftar Tugas
Pilih: 4
Daftar semua tugas
Nama    NIM     Kelas
Tika    1003    1C
Erik    1002    1B
Dila    1001    1A
```

```
Menu:
1. Mengumpulkan Tugas
2. Menilai Tugas
3. Melihat Tugas Teratas
4. Melihat Daftar Tugas
Pilih: 2
Menilai tugas dari Tika
Masukkan nilai (0-100): 87
Nilai Tugas Tika adalah 87

Menu:
1. Mengumpulkan Tugas
2. Menilai Tugas
3. Melihat Tugas Teratas
4. Melihat Daftar Tugas
Pilih: 4
Daftar semua tugas
Nama    NIM     Kelas
Erik    1002    1B
Dila    1001    1A
```

2.1.3 Pertanyaan

1. Lakukan perbaikan pada kode program, sehingga keluaran yang dihasilkan sama dengan verifikasi hasil percobaan! Bagian mana yang perlu diperbaiki?
2. Berapa banyak data tugas mahasiswa yang dapat ditampung di dalam Stack? Tunjukkan potongan kode programnya!
3. Mengapa perlu pengecekan kondisi **!isFull()** pada method **push**? Kalau kondisi if-else tersebut dihapus, apa dampaknya?
4. Modifikasi kode program pada class **MahasiswaDemo** dan **StackTugasMahasiswa** sehingga pengguna juga dapat melihat mahasiswa yang pertama kali mengumpulkan tugas melalui operasi lihat tugas terbawah!
5. Tambahkan method untuk dapat menghitung berapa banyak tugas yang sudah dikumpulkan saat ini, serta tambahkan operasi menunya!
6. **Commit dan push kode program ke Github**

2.2 Percobaan 2: Konversi Nilai Tugas ke Biner

Waktu Percobaan: 60 Menit

Sampai tahap ini, proses pengelolaan data tugas mahasiswa menggunakan konsep Stack telah berhasil dibuat pada Percobaan 1. Selanjutnya, pada Percobaan 2 ini ditambahkan method baru yang berfungsi untuk mengonversi nilai tugas bertipe int ke dalam bentuk biner setelah tugas tersebut diberi nilai dan dikeluarkan dari Stack.

2.2.1 Langkah-langkah Percobaan

1. Buka kembali file **StackTugasMahasiswa<NoAbsen>.java**
2. Tambahkan method **konversiDesimalKeBiner** dengan menerima parameter **kode** bertipe int

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

Pada method ini, terdapat penggunaan **StackKonversi** yang merupakan penerapan Stack, sama halnya dengan class **StackTugasMahasiswa**. Hal ini bertujuan agar Stack untuk mahasiswa berbeda dengan Stack yang digunakan untuk biner karena tipe data yang digunakan berbeda. Oleh karena itu, buat file baru bernama **StackKonversi<NoAbsen>.java**

Catatan: Perlu diingat bahwa pada dasarnya semua class Stack mempunyai operasi (method) yang sama. Hal yang membedakan adalah aktivitas spesifik yang perlu dilakukan, misalnya setelah menambah atau mengeluarkan data.

3. Tambahkan empat method yaitu **isEmpty**, **isFull**, **push**, dan **pull** sebagai operasi utama Stack pada class **StackKonversi**

```
int[] tumpukanBiner;
int size;
int top;

public StackKonversi() {
    this.size = 32; //asumsi 32 bit
    tumpukanBiner = new int[size];
    top = -1;
}
public boolean isEmpty() {
    return top == -1;
}

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

```
public void push(int data) {
    if (isFull()) {
        System.out.println(x:"Stack penuh");
    } else {
        top++;
        tumpukanBiner[top] = data;
    }
}

public int pop() {
    if (isEmpty()) {
        System.out.println(x:"Stack kosong.");
        return -1;
    } else {
        int data = tumpukanBiner[top];
        top--;
        return data;
    }
}
```

4. Agar nilai tugas mahasiswa dikonversi ke dalam bentuk biner setelah dilakukan penilaian, maka tambahkan baris kode program pada method **pop** di class **MahasiswaDemo**

```
case 2:
    Mahasiswa dinilai = stack.pop();
    if (dinilai != null) {
        System.out.println("Menilai tugas dari " + dinilai.nama);
        System.out.print("Masukkan nilai (0-100): ");
        int nilai = scan.nextInt();
        dinilai.tugasDinilai(nilai);
        System.out.printf("Nilai Tugas %s adalah %d\n", dinilai.nama, nilai);
        String biner = stack.konversiDesimalKeBiner(nilai);
        System.out.println("Nilai Biner Tugas: " + biner);
    }
    break;
```

5. Compile dan run program.
6. **Commit dan push kode program ke Github**

2.2.2 Verifikasi Hasil Percobaan

Cocokkan hasil compile kode program Anda dengan gambar berikut ini.

```
Menu:
1. Mengumpulkan Tugas
2. Menilai Tugas
3. Melihat Tugas Teratas
4. Melihat Daftar Tugas
Pilih: 2
Menilai tugas dari Tika
Masukkan nilai (0-100): 87
Nilai Tugas Tika adalah 87
Nilai Biner Tugas: 1010111
```

2.2.3 Pertanyaan

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

2.4 Latihan Praktikum

Waktu : 90 Menit

Mahasiswa mengajukan surat izin (karena sakit atau keperluan lain) setiap kali tidak mengikuti perkuliahan. Surat terakhir yang masuk akan diproses atau divalidasi lebih dulu oleh admin Prodi. Perhatikan class diagram berikut.

Surat<**NoAbsen**>
idSurat: String namaMahasiswa: String kelas: String jenisIzin: char durasi: int
Surat<**NoAbsen**>() Surat<**NoAbsen**>(idSurat: String, namaMahasiswa: String, kelas: String, jenisIzin: char, durasi: int)

Atribut **jenisIzin** digunakan untuk menyimpan keterangan ijin mahasiswa (S: sakit atau I: izin keperluan lain) dan **durasi** untuk menyimpan lama waktu izin.

Berdasarkan class diagram tersebut, implementasikan class **Surat** dan tambahkan class **StackSurat** untuk mengelola data Surat. Pada class yang memuat method main, buat pilihan menu berikut:

1. **Terima Surat Izin** untuk memasukkan data surat
2. **Proses Surat Izin** untuk memproses atau memverifikasi surat
3. **Lihat Surat Izin Terakhir** untuk melihat surat teratas
4. **Cari Surat** untuk mencari ada atau tidaknya surat izin berdasarkan nama mahasiswa