JOBSHEET IX STACK

Ka Abi Muhammad R.F./12/TI-1B

1. Buat folder baru bernama Jobsheet9 di dalam repository Praktikum ASD. Buat file baru, beri nama Mahasiswa.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

```
Mahasiswa12(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 B. Class StackTugasMahasiswa

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

5. Setelah membuat class Mahasiswa, selanjutnya perlu dibuat class StackTugasMahasiswa.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

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 StackTugasMahasiswa12(int size){
    this.size = size;
    stack = new Mahasiswa12[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(Mahasiswa12 mhs){
    if (!isFull()) {
        top++;
        stack[top] = mhs;
    } else{
        System.out.println(x:"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 Mahasiswa12 pop(){
    if (!isEmpty()) {
        Mahasiswa12 m = stack[top];
        top--;
        return m;
    }else{
        System.out.println(x:"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 Mahasiswa12 peek(){
    if (!isEmpty()) {
        return stack[top];
    }else {
        System.out.println(x:"Stack kosong! Tidak ada tugas yang dikumpulkan");
        return null;
    }
}
```

13. Tambahkan method print untuk dapat menampilkan semua daftar tugas mahasiswa pada Stack C. Class Utama

```
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(x:"");
}
```

14. Buat file baru, beri nama MahasiswaDemo.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.

```
public static void main(String[] args) {
    StackTugasMahasiswa12 stack = new StackTugasMahasiswa12(size:5);
    Scanner scan = new Scanner (System.in);
    int pilih;
```

17. Deklarasikan Scanner dengan nama variabel scan dan variabel pilih bertipe int

```
public static void main(String[] args) {
    StackTugasMahasiswa12 stack = new StackTugasMahasiswa12(size:5);
    Scanner scan = new Scanner (System.in);
    int pilih;
```

18. Tambahkan menu untuk memfasilitasi pengguna dalam memilih operasi Stack dalam mengelola data tugas mahasiswa menggunakan struktur perulangan do-while

```
do {
    System.out.println(x:"\nMenu");
    System.out.println(x:"1. Mengumpulkan Tugas");
    System.out.println(x:"2. Menilai Tugas");
    System.out.println(x:"3. Melihat Tugas Teratas");
    System.out.println(x:"4. Melihat Daftar Tugas");
    System.out.print(s:"Pilih: ");
    pilih = scan.nextInt();
    scan.nextLine();
    switch (pilih) {
        case 1:
            System.out.print(s:"Nama: ");
            String nama = scan.nextLine();
            System.out.print(s:"NIM: ");
            String nim = scan.nextLine();
            System.out.print(s:"Kelas: ");
            String kelas = scan.nextLine();
            Mahasiswa12 mhs = new Mahasiswa12(nama, nim, kelas);
            stack.push(mhs);
            System.out.printf(format:"Tugas %s berhasil dikumpulkan\n", mhs.nama);
            break;
        case 2:
            Mahasiswa12 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);
```

```
            case 3:
                Mahasiswa12 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: 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
Dila    1001    1A
Erik    1002    1B
Tika    1003    1C


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
Dila    1001    1A
Erik    1002    1B
```

2.1.3 Pertanyaan

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

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

    }
    System.out.println(x:"");
}
```

Ganti inisialisasi, kondisi, dan update pada fori

2. Berapa banyak data tugas mahasiswa yang dapat ditampung di dalam Stack? Tunjukkan potongan kode programnya!

```
ublic static void main(String[] args) {
  StackTugasMahasiswa12 stack = new StackTugasMahasiswa12(size:5);
  Scanner scan = new Scanner (System.in);
  int pilih;
```

aad 5

3. Mengapa perlu pengecekan kondisi !isFull() pada method push? Kalau kondisi if-else tersebut dihapus, apa dampaknya?

Tidak bisa membaca Ketika stack full/akan terus meminta input/terjadi error

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!

```
public Mahasiswa12 peekbottom(){
    if (!isEmpty()) {
        return stack[bottom];
    }else{
        System.out.println(x:"Stack kosong! Tidak ada tugas yang dikumpulkan");
        return null;
    }
}
```

```
case 3:
    Mahasiswa12 lihat = stack.peek();
    Mahasiswa12 lihatbottom = stack.peekbottom();
    if (lihat != null) {
        System.out.println("Tugas terakhir dikumpulkan oleh " +lihat.nama);
        System.out.println("Tugas pertama dikumpulkan oleh " +lihatbottom.nama);
    }
    break;
```

5. Tambahkan method untuk dapat menghitung berapa banyak tugas yang sudah dikumpulkan saat ini, serta tambahkan operasi menunya!

```
public int hitungJmlTugas(){
    int count = 0;
    if (!isEmpty()) {
        for (int i = top; i >= 0; i--) {
            count++;
        }
    }else {
        System.out.println(x:"Tidak ada tugas yang ditambahkan");
    }
    return count;
}
```

```
System.out.println(x:"\nMenu");
System.out.println(x:"1. Mengumpulkan Tugas");
System.out.println(x:"2. Menilai Tugas");
System.out.println(x:"3. Melihat Tugas Teratas");
System.out.println(x:"4. Melihat Daftar Tugas");
System.out.println(x:"5. Lihat Jumlah Tugas Terkumpul");
System.out.print(s:"Pilih: ");
pilih = scan.nextInt();
```

```
case 5:
    System.out.println("Jumlah tugas dikumpulkan: " + stack.hitungJmlTugas());
    break;
```

6. Commit dan push kode program ke Github

2.2 Percobaan 2: Konversi Nilai Tugas ke Biner

1. Tambahkan method konversiDesimalKeBiner dengan menerima parameter kode bertipe int (pada class stacktugasmahasiswa12)

```
public String konversiDesimalKeBIner(int nilai){
        StackKonversi12 stack = new StackKonversi12();
        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;
    }
```

2. buat class stack konversi

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

```
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:
    Mahasiswa12 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);
        String biner = stack.konversiDesimalKeBIner(nilai);
        System.out.println("Nilai Biner Tugas: " + biner);
    }
    break;
```

2.2.2 Verifikasi Hasil Percobaan

```
NIM: 2
Kelas: 2
Tugas Tika berhasil dikumpulkan

Menu
1. Mengumpulkan Tugas
2. Menilai Tugas
3. Melihat Tugas Teratas
4. Melihat Daftar Tugas
5. Lihat Jumlah Tugas Terkumpul
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!

Sisa pembagian 2 selalu menghasilkan 0 atau 1, inisialisasi while untuk memastikan nilai positif, lalu simpan nilai sisa dalam stack (push), lalu nilai dibagi 2 (karena modulus 2 juga), perulangan terus berjalan sampai nilai menyentuh nilai minimum (sekurang kurangnya 1 karena kondisi >0)

Lalu pada string biner Mengeluarkan isi stack satu per satu, menyusun digit-digit biner dari atas stack ke bawah (karena digit paling awal harus ditampilkan terakhir).

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

Sama saja, karena ujung ujungnya nilai akan menyentuh 1 atau 0 dan Ketika dibagi 2 (lagi) akan bernilai 0 yang mana (tidak) memenuhi kondisi while, dan akan berhenti