Nama : Dicky Darmawan
Kelas : TI 1B
No. Absen : 08
NIM : 244107020037

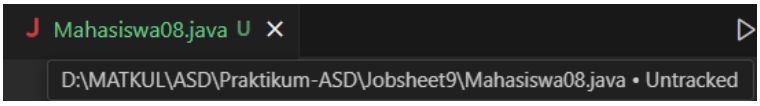
Percobaan 1: Mahasiswa Mengumpulkan Tugas

Waktu Percobaan: 90 Menit

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

```
public class Mahasiswa08 {
    String nim, nama, kelas;
    int nilai;

    Mahasiswa08() {

    }

    Mahasiswa08(String nim, String nama, String kelas)

    }

    void tugasDinilai(int 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

```
Mahasiswa08(String nim, String nama, 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

```
public class StackTugasMahasiswa08 {

    Mahasiswa08[] stack;
    int top, 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 StackTugasMahasiswa08(int size) {
    this.size = size;
    stack = new Mahasiswa08[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 (Mahasiswa08 mhs) {
    if (!isFull()) {
        top++;
        stack[top] = mhs;
    } else {
        System.out.println(x:"Stack pennuh! Tidak bisa menambahkan tug
    }
}
```

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 Mahasiswa08 pop() {
    if (!isEmpty()) {
        Mahasiswa08 m = stack[top];
        top--;
        return m;
    } else {
        System.out.println(x:"Stack kosong! Tidak ada tugas untuk dinila
        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 Mahasiswa08 peek(){
    if (!isEmpty()){
        return stack[top];
    } else {
        System.out.println(x:"Stack kosong! Tidak ada tugas yang dikum
        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" +
    }
    System.out.println(x:"");
}
```

C. Class Utama

14. Buat file baru, beri nama MahasiswaDemo<NoAbsen>.java
15. Tuliskan struktur dasar bahasa pemrograman Java yang terdiri dari fungsi main

```
public class MahasiswaDemo08 {

    public static void main(String[] args) {

    }
}
```

16. Di dalam fungsi main, lakukan instansiasi object StackTugasMahasiswa bernama stack dengan nilai parameternya adalah 5.

```
StackTugasMahasiswa08 stack = new StackTugasMahasiswa08(size:5);
```

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

```
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();
            Mahasiswa08 mhs = new Mahasiswa08(nama, nim, kelas);
            stack.push(mhs);
            System.out.printf(format:"Tugas %s berhasil dikumpulkan\n", mhs.nama);
            break;
        case 2:
            Mahasiswa08 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("Nilai Tugas %s adalah %d\n" + dinilai.nama, nilai);
            }
            break;
        case 3:
            Mahasiswa08 lihat = stack.peek();
            if (lihat != null) {
```

```
            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.

Verifikasi Hasil Percobaan

```
Menu:
1. Mengumpulkan Tugas
2. Menilai Tugas
3. Melihat Tugas Teratas
4. Melihat Daftar Tugas
Pilih: 1
Nama: Dicky
NIM: 101
Kelas: 1B
Tugas Dicky berhasil dikumpulkan

Menu:
1. Mengumpulkan Tugas
2. Menilai Tugas
3. Melihat Tugas Teratas
4. Melihat Daftar Tugas
Pilih: 1
Nama: Ziyan
NIM: 102
Kelas: 1B
Tugas Ziyan berhasil dikumpulkan

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

Menu:
1. Mengumpulkan Tugas
2. Menilai Tugas
3. Melihat Tugas Teratas
4. Melihat Daftar Tugas
Pilih: 1
Nama: Bela
NIM: 104
Kelas: 1A
Tugas Bela 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
Dicky   101     1B
Ziyan   102     1B
Bela    104     1A


Menu:
1. Mengumpulkan Tugas
2. Menilai Tugas
3. Melihat Tugas Teratas
4. Melihat Daftar Tugas
Pilih: 2
Menilai tugas dari Bela
Masukkan nilai (0-100): 90
Nilai Tugas Bela adalah 90

Menu:
1. Mengumpulkan Tugas
2. Menilai Tugas
3. Melihat Tugas Teratas
4. Melihat Daftar Tugas
Pilih: 4
Daftar semua tugas
Nama    NIM     Kelas
Dicky   101     1B
Ziyan   102     1B
```

Pertanyaan

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

   Saya tidak menemukan adanya perbedaan dari keluaran program saya dengan keluaran verifikasi

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

   ```
   StackTugasMahasiswa08 stack = new StackTugasMahasiswa08(size:5);
   ```

   Hanya menampung sebanyak 5 data mahasiswa

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

Fungsi dari pengecekan kondisi !isFull() adalah untuk memastikan apakah masih ada tempat untuk menambahkan elemen yang berrtujuan untuk menghindari terjadinya stack overflow, sehingga pada saat menambahkan elemen pada stack tidak terjadi hal tersebut.

Jika if-else dihapus kemungkinan akan error, karena variable top tidak ditambahkan, tetap pada index -1 sehingga di luar dari array yang dibuat.

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!

Class StackTugasMahasiswa:

```
public Mahasiswa08 awal () {
    if (!isEmpty()) {
        return stack[0];
    } else {
        System.out.println(x:"Stack kosong! Tidak ada tugas yang dikumpul
        return null;
    }
}
```

Class MahasiswaDemo:

```
System.out.println(x:"4. Melihat Daftar Tugas");
System.out.println(x:"5. Melihat Mahasiswa Pertama");
```

```
        case 5:
            Mahasiswa08 awal = stack.awal();
            if(awal!=null) {
                System.out.println(awal.nama + "adalah mahasiswa yang pert
            }
            break;
        default:
            System.out.println(x:"Pilihan tidak valid");
    }
} while (pilih >= 1 && pilih <= 5);
```

Outputnya:

```
Menu:
1. Mengumpulkan Tugas
2. Menilai Tugas
3. Melihat Tugas Teratas
4. Melihat Daftar Tugas
5. Melihat Mahasiswa Pertama
Pilih: 1
Nama: Dicky
NIM: 101
Kelas: TI 1B
Tugas Dicky berhasil dikumpulkan

Menu:
1. Mengumpulkan Tugas
2. Menilai Tugas
3. Melihat Tugas Teratas
4. Melihat Daftar Tugas
5. Melihat Mahasiswa Pertama
Pilih: 1
Nama: Adam
NIM: 102
Kelas: TI 1A
Tugas Adam berhasil dikumpulkan

Menu:
1. Mengumpulkan Tugas
2. Menilai Tugas
3. Melihat Tugas Teratas
4. Melihat Daftar Tugas
5. Melihat Mahasiswa Pertama
Pilih: 5
Dicky adalah mahasiswa yang pertama mengumpulkan
```

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

   Tambahan kode

```
public int hitungTugas() {
    return top+1;
}
```

```
System.out.println(x:"6. Hitung Jumlah Tugas");
System.out.print(s:"Pilih: ");
```

```
case 6:
    System.out.println("Tugas yang telah dikumpulkan sebanyak " + stack.hi
    break;
```

   Output

```
Menu:
1. Mengumpulkan Tugas
2. Menilai Tugas
3. Melihat Tugas Teratas
4. Melihat Daftar Tugas
5. Melihat Mahasiswa Pertama
6. Hitung Jumlah Tugas
Pilih: 1
Nama: Dicky
NIM: 101
Kelas: TI 1B
Tugas Dicky berhasil dikumpulkan

Menu:
1. Mengumpulkan Tugas
2. Menilai Tugas
3. Melihat Tugas Teratas
4. Melihat Daftar Tugas
5. Melihat Mahasiswa Pertama
6. Hitung Jumlah Tugas
Pilih: 1
Nama: Adam
NIM: 102
Kelas: Ti 1A
Tugas Adam berhasil dikumpulkan
```

```
Menu:
1. Mengumpulkan Tugas
2. Menilai Tugas
3. Melihat Tugas Teratas
4. Melihat Daftar Tugas
5. Melihat Mahasiswa Pertama
6. Hitung Jumlah Tugas
Pilih: 1
Nama: Sam
NIM: 103
Kelas: TI 1C
Tugas Sam berhasil dikumpulkan

Menu:
1. Mengumpulkan Tugas
2. Menilai Tugas
3. Melihat Tugas Teratas
4. Melihat Daftar Tugas
5. Melihat Mahasiswa Pertama
6. Hitung Jumlah Tugas
Pilih: 1
Nama: Bagas
NIM: 104
Kelas: TI 1F
Tugas Bagas berhasil dikumpulkan

Menu:
1. Mengumpulkan Tugas
2. Menilai Tugas
3. Melihat Tugas Teratas
4. Melihat Daftar Tugas
5. Melihat Mahasiswa Pertama
6. Hitung Jumlah Tugas
Pilih: 6
Tugas yang telah dikumpulkan sebanyak 4
```

6. Commit dan push kode program ke Github

Percobaan 2: Konversi Nilai Tugas ke Biner

Waktu Percobaan: 60 Menit

Langkah-langkah Percobaan

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

   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

```
public String konversiDesimalKeBiner(int nilai) {
    StackKonversi08 stack = new StackKonversi08();
    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;
}
```

   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

```java
public class StackKonversi08 {

    int[] tumpukanBiner;
    int size, top;

    public StackKonversi08() {
        this.size = 32;
        tumpukanBiner = new int[size];
        top = -1;
    }

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

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

```java
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

```java
System.out.printf(format:"Nilai Tugas %s adalah %d\n", dinilai.n
String biner = stack.konversiDesimalKeBiner(nilai);
System.out.println("Nilai Biner Tugas: " + biner);
```

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

Verifikasi Hasil Percobaan

```
Menu:
1. Mengumpulkan Tugas
2. Menilai Tugas
3. Melihat Tugas Teratas
4. Melihat Daftar Tugas
5. Melihat Mahasiswa Pertama
6. Hitung Jumlah Tugas
Pilih: 2
Menilai tugas dari Dicky
Masukkan nilai (0-100): 95
Nilai Tugas Dicky adalah 95
Nilai Biner Tugas: 1011111
```

Pertanyaan

1. Jelaskan alur kerja dari method konversiDesimalKeBiner!

```java
StackKonversi08 stack = new StackKonversi08();
```

Menginisialisasi stack terlebih dahulu yang digunakan untuk menyimpan sisa hasil bagi nilai dibagi 2

```
while (nilai > 0) {
    int sisa = nilai % 2;
    stack.push(sisa);
    nilai = nilai / 2;
}
```

Membuat perulangan pembagian dengan 2, nilai % 2 nantinya akan menghasilkan sisa yaitu bilangan biner (0 atau 1), lalu sisa dari pembagian tadi di push ke stack agar urutannya bisa dibalik, nilai = nilai /2 untuk mengurangi nilai/bilangan yang diubah ke biner sampai 0

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

```
String biner = new String();
while (!stack.isEmpty()) {
    biner += stack.pop();
}
return biner;
```

Membuat string biner, isi stack akan di-pop satu persatu dari atas dan disimpan ke dalam strig binerr secara berurutan, lalu mengembalikan biner sebagai hasil dari konversi

Latihan Praktikum

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

```
Menu:
1. Terima Surat Izin
2. Proses Surat Izin
3. Lihat Surat Izin Terakhir
4. Cari Surat Berdasarkan Nama
Pilih: 1
ID Surat: 123
Nama: Dicky
Kelas: TI 1B
Jenis Izin: Sakit
Durasi: 6
Surat berhasil ditambahkan

Menu:
1. Terima Surat Izin
2. Proses Surat Izin
3. Lihat Surat Izin Terakhir
4. Cari Surat Berdasarkan Nama
Pilih: 1
ID Surat: 234
Nama: Abi
Kelas: TI 1B
Jenis Izin: Sakit
Durasi: 6
Surat berhasil ditambahkan
```

```
Menu:
1. Terima Surat Izin
2. Proses Surat Izin
3. Lihat Surat Izin Terakhir
4. Cari Surat Berdasarkan Nama
Pilih: 1
ID Surat: 345
Nama: Azmi
Kelas: TI 1A
Jenis Izin: Izin
Durasi: 3
Surat berhasil ditambahkan

Menu:
1. Terima Surat Izin
2. Proses Surat Izin
3. Lihat Surat Izin Terakhir
4. Cari Surat Berdasarkan Nama
Pilih: 1
ID Surat: 456
Nama: Ahmad
Kelas: TI 1G
Jenis Izin: Sakit
Durasi: 2
Surat berhasil ditambahkan
```

2. Proses Surat Izin untuk memproses atau memverifikasi surat

```
Menu:
1. Terima Surat Izin
2. Proses Surat Izin
3. Lihat Surat Izin Terakhir
4. Cari Surat Berdasarkan Nama
Pilih: 2
Surat dari Ahmad berhasil diproses
```

3. Lihat Surat Izin Terakhir untuk melihat surat teratas

```
Menu:
1. Terima Surat Izin
2. Proses Surat Izin
3. Lihat Surat Izin Terakhir
4. Cari Surat Berdasarkan Nama
Pilih: 3
Surat teratas adalah milik Azmi
```

4. Cari Surat untuk mencari ada atau tidaknya surat izin berdasarkan nama mahasiswa

```
Menu:
1. Terima Surat Izin
2. Proses Surat Izin
3. Lihat Surat Izin Terakhir
4. Cari Surat Berdasarkan Nama
Pilih: 4
Masukkan nama Mahasiswa: Abi
Surat atas nama Abi berhasil ditemukan
```