

Nama : Siti Nikmatu Sholihah

NIM : 244107020014

Kelas : TI1B

JOBSHEET 9 – STACK

Percobaan 1: Mahasiswa Mengumpulkan Tugas

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.

Mahasiswa22.java
nim: String nama: String kelas: String nilai: int
Mahasiswa22() Mahasiswa22(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.

StackTugasMahasiswa22.java
stack: Mahasiswa[] size: int top: int
StackTugasMahasiswa(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.

A. Class Mahasiswa

1. Buat folder baru dengan nama **jobsheet9** didalam repository **Praktikum-ASD**.
Buat file baru untuk class Mahasiswa dengan nama **Mahasiswa22.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.

```
jobsheet9 > Mahasiswa22.java > Mahasiswa22 > Mahasiswa22(String, String, String)
1  public class Mahasiswa22 {
2
3  String nama, nim, kelas;
4  int nilai;
5
6  Mahasiswa22(String nama, String nim, String kelas) {
7      this.nama = nama;
8      this.nim = nim;
9      this.kelas = kelas;
10     nilai = -1;
11 }
12 }
```

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

```
jobsheet9 > Mahasiswa22.java > Mahasiswa22 > tugasDinilai(int)
1  public class Mahasiswa22 {
2
3  String nama, nim, kelas;
4  int nilai;
5
6  Mahasiswa22(String nama, String nim, String kelas) {
7      this.nama = nama;
8      this.nim = nim;
9      this.kelas = kelas;
10     nilai = -1;
11 }
12
13 void tugasDinilai(int nilai) {
14     this.nilai = nilai;
15 }
16 }
```

B. Class StackTugasMahasiswa

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

```
jobsheet9 > StackTugasMahasiswa22.java > StackTugasMahasiswa22 > size
1 public class StackTugasMahasiswa22 {
2
3     Mahasiswa22[] stack;
4     int top;
5     int size;
6 }
```

3. 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.

```
jobsheet9 > StackTugasMahasiswa22.java > StackTugasMahasiswa22 >
1 public class StackTugasMahasiswa22 {
2
3     Mahasiswa22[] stack;
4     int top;
5     int size;
6
7     public StackTugasMahasiswa22(int size){
8         this.size = size;
9         stack = new Mahasiswa22[size];
10        top = -1;
11    }
12 }
```

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

```
jobsheet9 > StackTugasMahasiswa22.java > StackTugasMahasiswa22 > isFull()
1 public class StackTugasMahasiswa22 {
2
3     Mahasiswa22[] stack;
4     int top;
5     int size;
6
7     public StackTugasMahasiswa22(int size){
8         this.size = size;
9         stack = new Mahasiswa22[size];
10        top = -1;
11    }
12
13    public boolean isFull(){
14        if (top == size - 1) {
15            return true;
16        } else {
17            return false;
18        }
19    }
20 }
```

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

```
jobsheet9 > StackTugasMahasiswa22.java
1 public class StackTugasMahasiswa22 {
2
3     Mahasiswa22[] stack;
4     int top;
5     int size;
6
7     public StackTugasMahasiswa22(int size){
8         this.size = size;
9         stack = new Mahasiswa22[size];
10        top = -1;
11    }
12
13    public boolean isFull(){
14        if (top == size -1) {
15            return true;
16        } else {
17            return false;
18        }
19    }
20
21    public boolean isEmpty(){
22        if (top == -1) {
23            return true;
24        } else {
25            return false;
26        }
27    }
28 }
```

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

```
jobsheet9 > StackTugasMahasiswa22.java > StackTugasMahasiswa22 > isEmpty()
1 public class StackTugasMahasiswa22 {
2
3     Mahasiswa22[] stack;
4     int top;
5     int size;
6
7     public StackTugasMahasiswa22(int size){
8         this.size = size;
9         stack = new Mahasiswa22[size];
10        top = -1;
11    }
12
13    public boolean isFull(){
14        if (top == size -1) {
15            return true;
16        } else {
17            return false;
18        }
19    }
20
21    public boolean isEmpty(){
22        if (top == -1) {
23            return true;
24        } else {
25            return false;
26        }
27    }
28
29    public void push (Mahasiswa22 mhs){
30        if (!isFull()) {
31            top++;
32            stack[top] = mhs;
33        } else {
34            System.out.println("Stack penuh! tidak bisa menambahkan tugas lagi.");
35        }
36    }
37 }
```

7. 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.

```
jobsheet9 > StackTugasMahasiswa22.java > StackTugasMahasiswa22 > push(Mahasiswa22)
1 public class StackTugasMahasiswa22 {
38     public Mahasiswa22 pop(){
39         if (!isEmpty()) {
40             Mahasiswa22 m = stack[top];
41             top--;
42             return m;
43         } else {
44             System.out.println("Stack kosong! Tidak ada tugas untuk dinilai.");
45             return null;
46         }
47     }
48 }
```

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

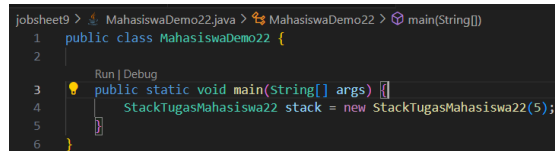
```
jobsheet9 > StackTugasMahasiswa22.java > StackTugasMahasiswa22 > peek()
1 public class StackTugasMahasiswa22 {
38     public Mahasiswa22 pop(){
39         if (!isEmpty()) {
40             Mahasiswa22 m = stack[top];
41             top--;
42             return m;
43         } else {
44             System.out.println("Stack kosong! Tidak ada tugas untuk dinilai.");
45             return null;
46         }
47     }
48
49     public Mahasiswa22 peek(){
50         if (!isEmpty()) {
51             return stack[top];
52         } else {
53             System.out.println("Stack kosong! tidak ada tugas yang dikumpulkan");
54             return null;
55         }
56     }
57 }
```

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

```
jobsheet9 > StackTugasMahasiswa22.java > StackTugasMahasiswa22 > print()
1 public class StackTugasMahasiswa22 {
38     public Mahasiswa22 pop(){
39         if (!isEmpty()) {
40             Mahasiswa22 m = stack[top];
41             top--;
42             return m;
43         } else {
44             System.out.println("stack kosong! Tidak ada tugas untuk dinilai.");
45             return null;
46         }
47     }
48
49     public Mahasiswa22 peek(){
50         if (!isEmpty()) {
51             return stack[top];
52         } else {
53             System.out.println("stack kosong! tidak ada tugas yang dikumpulkan");
54             return null;
55         }
56     }
57
58     public void print(){
59         for (int i = 0; i <= top; i++) {
60             System.out.println(stack[i].nama + "\t" + stack[i].nim + "\t" + stack[i].kelas);
61         }
62         System.out.println("");
63     }
64 }
```

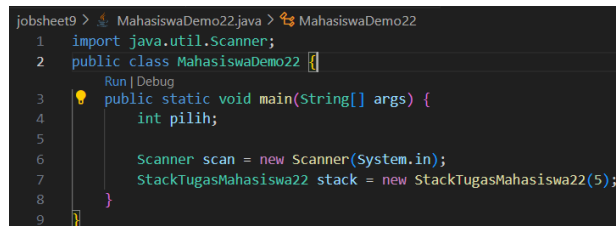
C. Class Utama

1. Buat file baru, beri nama **MahasiswaDemo.java**.
2. Tuliskan struktur dasar bahasa pemrograman Java yang terdiri dari fungsi main.
3. Di dalam fungsi main, lakukan instansiasi object **StackTugasMahasiswa** bernama stack dengan nilai parameternya adalah 5.



```
jobsheet9 > MahasiswaDemo22.java > MahasiswaDemo22 > main(String[])
1 public class MahasiswaDemo22 {
2
3     public static void main(String[] args) {
4         StackTugasMahasiswa22 stack = new StackTugasMahasiswa22(5);
5     }
6 }
```

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



```
jobsheet9 > MahasiswaDemo22.java > MahasiswaDemo22
1 import java.util.Scanner;
2 public class MahasiswaDemo22 {
3     public static void main(String[] args) {
4         int pilih;
5
6         Scanner scan = new Scanner(System.in);
7         StackTugasMahasiswa22 stack = new StackTugasMahasiswa22(5);
8     }
9 }
```

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

```

jobsheet9 > MahasiswaDemo22.java > MahasiswaDemo22
1 import java.util.Scanner;
2 public class MahasiswaDemo22 {
3     Run | Debug
4     public static void main(String[] args) {
5         int pilih;
6
7         Scanner scan = new Scanner(System.in);
8         StackTugasMahasiswa22 stack = new StackTugasMahasiswa22(5);
9
10        do {
11            System.out.println("\nMenu:");
12            System.out.println("1. Mengumpulkan Tugas");
13            System.out.println("2. Menilai Tugas");
14            System.out.println("3. Melihat Tugas Teratas");
15            System.out.println("4. Melihat Daftar Tugas");
16            System.out.print("Pilih: ");
17            pilih = scan.nextInt();
18            scan.nextLine();
19
20            switch (pilih) {
21                case 1:
22                    System.out.print("Nama: ");
23                    String nama = scan.nextLine();
24                    System.out.print("nim: ");
25                    String nim = scan.nextLine();
26                    System.out.print("Kelas: ");
27                    String kelas = scan.nextLine();
28                    Mahasiswa22 mhs = new Mahasiswa22(nama, nim, kelas);
29                    stack.push(mhs);
30                    System.out.printf("Tugas %s berhasil dikumpulkan\n", mhs.nama);
31                    break;
32                case 2:
33                    Mahasiswa22 dinilai = stack.pop();
34                    if (dinilai != null) {
35                        System.out.println("Menilai tugas dari " + dinilai.nama);
36                        System.out.print("Masukkan nilai (0-100): ");
37                        int nilai = scan.nextInt();
38                        scan.nextLine();
39                        dinilai.tugasDinilai(nilai);
40                        System.out.printf("Nilai tugas %s berhasil dikumpulkan\n", dinilai.nama);
41                    } else {
42                        System.out.println("Tidak ada tugas yang bisa dinilai.");
43                    }
44                    break;
45                case 3:
46                    Mahasiswa22 lihat = stack.peek();
47                    if (lihat != null) {
48                        System.out.println("Tugas terakhir dikumpulkan oleh " + lihat.nama);
49                    } else {
50                        System.out.println("Belum ada tugas yang dikumpulkan.");
51                    }
52                    break;
53                case 4:
54                    System.out.println("Daftar semua tugas:");
55                    System.out.println("nama\tNIM\tKelas");
56                    stack.print();
57                    break;
58                default:
59                    System.out.println("Pilihan tidak valid.");
60            }
61        } while (pilih >= 1 && pilih <= 4);
62    }
63 }

```

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

8. Verifikasi hasil percobaan

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


Pertanyaan

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

Jawab:

Yang perlu diperbaiki ada pada bagian ini.

```
58 public void print() {
59     for (int i = 0; i <= top; i++) {
60         System.out.println(stack[i].nama + "\t" + stack[i].nim + "\t" + stack[i].kelas);
61     }
62     System.out.println("");
63 }
```

Untuk kode setelah diperbaiki adalah sebagai berikut:

```
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("");
}
```

Output:

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

Jawab:

Data yang dapat diinputkan sebanyak 5 data

```
StackTugasMahasiswa22 stack = new StackTugasMahasiswa22(5);
```

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

Jawab:

Karena **`isFull()`** berguna untuk melakukan pengecekan sebelum melakukan penginputan data, apakah masih ada tempat yang tersisa untuk menampung data. Jika kondisi `if else` dihapus maka akan berdampak pada `push` yakni tidak adanya Batasan saat menginputkan data dan data yang diinputkan tidak terdata di dalam **`size(kapasitas yang sebelumnya di deklarasikan)`** pada bagian **`isFull()`**.

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!

Jawab:

Kode setelah dilakukan perubahan:

```
public Mahasiswa22 peek(){
    if (isEmpty()) {
        return stack[0];
    } else {
        System.out.println("Stack kosong! tidak ada tugas yang dikumpulkan");
        return null;
    }
}
```

```
do {
    System.out.println("\nMenu:");
    System.out.println("1. Mengumpulkan Tugas");
    System.out.println("2. Menilai Tugas");
    System.out.println("3. Melihat Tugas Terbawah");
    System.out.println("4. Melihat Daftar Tugas");
    System.out.println("5. Lihat Jumlah Tugas yang Sudah Dimasukkan");
    System.out.print("Pilih: ");
    pilih = scan.nextInt();
    scan.nextLine();
}
```

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

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

Menu:
1. Mengumpulkan Tugas
2. Menilai Tugas
3. Melihat Tugas Terbawah
4. Melihat Daftar Tugas
Pilih: 3
Tugas pertama dikumpulkan oleh Dila
```

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

Jawab:

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

```
do {
    System.out.println("\nMenu:");
    System.out.println("1. Mengumpulkan Tugas");
    System.out.println("2. Menilai Tugas");
    System.out.println("3. Melihat Tugas Terbawah");
    System.out.println("4. Melihat Daftar Tugas");
    System.out.println("5. Lihat Jumlah Tugas yang Sudah Dimasukkan");
    System.out.print("Pilih: ");
    pilih = scan.nextInt();
    scan.nextLine();
}
```

```
case 5:
    int jumlah = stack.jumlahTugas();
    System.out.println("Jumlah tugas yang sudah dikumpulkan: " + jumlah);
    break;
```

6. Commit dan push kode program ke Github

Jawab:

Selesai

Percobaan 2: Koversi Nilai Tugas ke Biner

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.

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

```
jobsheet9 > StackTugasMahasiswa22.java > StackTugasMahasiswa22 > top
3  public class StackTugasMahasiswa22 {
71      public String konversiDesimalKeBiner(int nilai){
72          StackKonversi22 binerStack = new StackKonversi22();
73          while (nilai > 0) {
74              int sisa = nilai % 2;
75              binerStack.push(sisa);
76              nilai /= 2;
77          }
78          String biner = "";
79          while (!binerStack.isEmpty()) {
80              biner += binerStack.pop();
81          }
82          return biner;
83      }
```

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.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**.

```

jobsheet9 > StackKonversi22.java > StackKonversi22
1 public class StackKonversi22 {
2
3     int[] tumpukanBiner;
4     int size;
5     int top;
6
7     public StackKonversi22(){
8         this.size = 32;
9         tumpukanBiner = new int[size];
10        top = -1;
11    }
12
13    public boolean isEmpty(){
14        return top == -1;
15    }
16
17    public boolean isFull(){
18        return top == size -1;
19    }
20
21    public void push (int data){
22        if (isFull()) {
23            System.out.println("Stack penuh");
24        } else {
25            top++;
26            tumpukanBiner[top] = data;
27        }
28    }
29
30    public int pop(){
31        if (isEmpty()) {
32            System.out.println("Stack kosong");
33        } else {
34            int data = tumpukanBiner[top];
35            top --;
36            return data;
37        }
38    }
39
40 }

```

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:
    Mahasiswa22 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();
        scan.nextLine();
        dinilai.tugasDinilai(nilai);
        System.out.printf("Nilai Tugas %s berhasil dikumpulkan\n", dinilai.nama);
        String biner = stack.konversiDesimalKeBiner(nilai);
        System.out.println("Nilai biner tugas: " + biner);
    } else {
        System.out.println("Tidak ada tugas yang bisa dinilai.");
    }
    break;

```

5. Compile dan run program.
6. Commit dan push kode program ke Github
7. Verifikasi hasil percobaan.

```

Menu:
1. Mengumpulkan Tugas
2. Menilai Tugas
3. Melihat Tugas Terbawah
4. Melihat Daftar Tugas
5. Lihat Jumlah Tugas yang Sudah Dimasukkan
Pilih: 2
Menilai tugas dari Tika
Masukkan nilai (0-100): 87
Nilai Tugas Tika berhasil dikumpulkan
Nilai biner tugas: 1010111

```

Pertanyaan

1. Jelaskan alur kerja dari method konversiDesimalKeBiner!

Jawab:

Di dalam method **konversiDesimalKeBiner**, terdapat sebuah instansiasi stack yang berfungsi untuk membuat objek stack guna menyimpan hasil sisa pembagian bilangan desimal dengan 2.

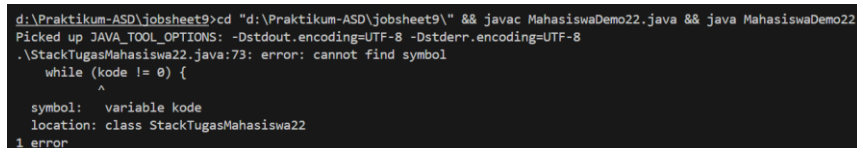
Setelah itu, dilakukan perulangan menggunakan **while** dengan kondisi selama nilai tidak sama dengan 0. Dalam perulangan ini, akan dihitung sisa pembagian **nilai % 2**, lalu disimpan ke dalam variabel **sisa** yang bertipe data **int**. Nilai **sisa** tersebut kemudian dimasukkan ke dalam stack menggunakan operasi **push**, dan nilai akan dibagi 2 secara bertahap.

Selanjutnya, terdapat perulangan kedua yang berfungsi untuk menyusun bilangan biner. Dengan kondisi selama stack belum kosong, maka angka paling atas akan diambil menggunakan **pop**, dan hasilnya ditambahkan ke dalam string **biner**.

Setelah semua proses selesai, nilai string **biner** akan dikembalikan ke pemanggil method menggunakan **return**.

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

Jawab:



```
d:\Praktikum-ASD\jobsheet9>cd "d:\Praktikum-ASD\jobsheet9\" && javac MahasiswaDemo22.java && java MahasiswaDemo22
Picked up JAVA_TOOL_OPTIONS: -Dstdout.encoding=UTF-8 -Dstderr.encoding=UTF-8
.\StackTugasMahasiswa22.java:73: error: cannot find symbol
    while (kode != 0) {
           ^
    symbol:   variable kode
    location: class StackTugasMahasiswa22
1 error
```

Hasilnya akan error saat kita melakukan kompilasi dikarenakan variabel yang dideklarasikan adalah nilai bukan kode. Maka variabel kode tidak dikenal, karena java itu sifatnya *statically typed* dan sangat ketat dalam hal deklarasi variabel.