LAPORAN HASIL PRAKTIKUM
ALGORITMA DAN STRUKTUR DATA

Oleh:
DZULFIKAR MUHAMMAD AL GHIFARI
NIM. 2341760071
SIB-1F / 08
D-IV SISTEM INFORMASI BISNIS
JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG

PRAKTIKUM 15

15.2 Percobaan 1

15.2.1 Langkah Langkah percobaan

1. Membuat Class contohlist

```
public class ContohList08 {
    Run | Debug | Run main | Debug main | Codeium: Refactor | Explain | Generate Javadoc | ×
```

2. Membuat List

```
List l = new ArrayList();
l.add(e:1);
l.add(e:2);
l.add(e:3);
l.add(e:"Cireng");
System.out.printf(format:"Elenen 0: %d total elenen: %d elenen
terakhir: %s\n", l.get(index:0), l.size(), l.get(l.size() - 1));

l.add(e:4);
l.remove (index:0);
System.out.printf(format:"Elenen 0: %d total elenen: 3d elenen
terakhir: %s\n", l.get(index:0), l.size(), l.get(l.size() - 1));
```

3. Membuat list dengan type string

```
List<String> names = new LinkedList<>();
names.add (e:"Noureen");
names.add (e:"Akh leena");
names.add (e:"Shannun");
names.add (e:"Uwais");
names.add (e:"A1-0arni");

System.out.printf(format:"Elenen 0: %s total elenen: %s elenen
terakhir: %s\n", names.get(index:0), names.size(), names.get
(names.size() - 1));

names.set(index:0, element:"My kid");

System.out.printf(format:"Elemen 0: %s total elemen: %s elemen
terakhir: %s\n", names.get(index:0), names.size(), names.get
(names.size() - 1));

System.out.println("Names: " + names.toString());
```

VERIFIKASI HASIL PERCOBAAN 15.2.2

```
Elenen 0: 1 total elenen: 4 elenen terakhir: Cireng
Elenen 0: 2 total elenen: 4 elenen terakhir: 4
Elenen 0: Noureen total elenen: 5 elenen terakhir: A1-0arni
Elemen 0: My kid total elemen: 5 elemen terakhir: A1-0arni
Names: [My kid, Akh leena, Shannun, Uwais, A1-0arni]
dzf@dzf-14ARE05:/media/dzf/DATA/1POLINEMA/2Genap 2023-2024/PraktikumAlgor
itma/pertemuan15$
```

PERTANYAAN 15.2.3

1. Perhatikan baris kode 25-36, mengapa semua jenis data bisa ditampung ke dalam sebuah Arraylist?

Karena tidak ada definisi tipe dari arraylist pada saat instansiasi arraylist

2. Modifikasi baris kode 25-36 seingga data yang ditampung hanya satu jenis atau spesifik tipe tertentu!

```
// List l = new ArrayList();
List<Integer> l = new ArrayList<>();
```

Dengan mengubahnya seperti kode di atas

3. Ubah kode pada baris kode 38 menjadi seperti ini

```
LinkedList<String> names = new LinkedList<>();
```

```
LinkedList<String> names = new LinkedList<>();
```

4. Tambahkan juga baris berikut ini, untuk memberikan perbedaan dari tampilan yang sebelumnya

```
names.push("Mei-mei");
System.out.printf("Elemen 0: %s total elemen: %s elemen terakhir: %s\n",
        names.getFirst(), names.size(), names.getLast());
System.out.println("Names: " + names.toString());
```

```
names. push(e:"Hei-hei");
System.out.printf(format:"Elemen 0: %s total elemen: %s elemen
terakhir: %s\n", names.getFirst(), names.size(), names.getLast());
System.out.println("Names: " + names. toString());
```

5. Dari penambahan kode tersebut, silakan dijalankan dan apakah yang dapat Anda jelaskan!

Dengan menggunakan linkedlist kita dapat melakukan fungsi push, get first dan get last

15.3 Percobaan 2

15.3.1 Langkah-langkah Percobaan

1. Membuat class loopcollection

```
public class LoopCollection08 {
    Run | Debug | Run main | Debug main | Codeium: Refactor | Explain | Generate Javad
    public static void main(String[] args) {
```

2. Menambahkan foreach pada stack untuk menampilkan stack

```
Stack<String> fruits = new Stack<>();
fruits.push (item:"Banana");
fruits.add(e:"0range");
fruits.add(e:"Watermelon");
fruits.add (e:"Leci");
fruits.push(item:"salak");

for (String fruit : fruits) {
System.out.printf (format:"%s " , fruit);
}

System.out.println("\n" + fruits.toString());

while (!fruits.empty()) {
    System.out.printf (format:"%s ", fruits.pop());
}
```

3. Menambahkan for pada stack untuk menampilkan stack

```
fruits.push(item:"Melon");
fruits.push(item:"Durian");
System.out.println(x:"");

for(Iterator<String> it = fruits.iterator(); it.hasNext();) {
    String fruit = it.next();
    System.out.printf (format:"%s" , fruit);
}

System.out.println(x:"");
fruits. stream().forEach(e -> {
    System.out.printf(format:"%s" , e);
});

System.out.println(x:"");
for (int i=0; i < fruits.size(); i++) {
    System.out.printf(format:"%s" , fruits.get(i));
}
```

VERIFIKASI HASIL PERCOBAAN 15.3.2

```
Banana 0range Watermelon Leci salak
[Banana, 0range, Watermelon, Leci, salak]
salak Leci Watermelon 0range Banana
Melon Durian
Melon Durian
Melon Durian dzf@dzf-14ARE05:/media/dzf/DATA/
```

PERTANYAAN 15.3.3

1. Apakah perbedaan fungsi push() dan add() pada objek fruits?

add menambahkan elemen baru ke ujung (rear) dari objek Stack

Push mirip dengan add, namun menggunakan konsep lifo

2. Silakan hilangkan baris 43 dan 44, apakah yang akan terjadi? Mengapa bisa demikian?

ketika kode program dijalankan tidak mengeluarkan output “Melon Durian”, karena pada perulangan terakhir yang ada di kode program, method get tidak menjalankan statement apapun

3. Jelaskan fungsi dari baris 46-49?

untuk menampilkan seluruh element pada stack.

4. Silakan ganti baris kode 25, Stack<String> menjadi List<String> dan apakah yang terjadi?

Mengapa bisa demikian?

eror, hal itu dikarenakan pada kode program tidak dituliskan “import java.util.List”

5. Ganti elemen terakhir dari dari objek fruits menjadi “Strawberry”!

```
fruits.push(item:"Melon");
fruits.push(item:"Durian");
System.out.println(x:"");

fruits.pop(); // Remove the last element
fruits.push(item:"Strawberry");
```

```
Banana 0range Watermelon Leci salak
[Banana, 0range, Watermelon, Leci, salak]
salak Leci Watermelon 0range Banana
Melon Strawberry
Melon Strawberry
Melon Strawberry dzf@dzf-14ARE05:/media/dz
```

6. Tambahkan 3 buah seperti “Mango”,”guava”, dan “avocado” kemudian dilakukan sorting!

```
// Add new fruits (Mango, Guava, Avocado)
fruits.addAll(java.util.Arrays.asList(...a:"Mango", "Guava",
"Avocado"));

// Sort the fruits (ascending order)
fruits.sort(java.util.Comparator.naturalOrder());
```

```
Banana 0range Watermelon Leci salak
[Banana, 0range, Watermelon, Leci, salak]
salak Leci Watermelon 0range Banana
Avocado Guava Mango Melon Strawberry
Avocado Guava Mango Melon Strawberry
Avocado Guava Mango Melon Strawberry dzf@dz
```

15.4 Percobaan 3

15.4.1 Langkah-langkah Percobaan

1. Menambahkan class mahasiswa

```
Codeium: Refactor | Explain
public class Mahasiswa08 {
```

2. Menambahkan attr dan konstruktor

```
String nim;
String nama;
String notelp;

public Mahasiswa08() {
}

public Mahasiswa08(String nim, String nama, String notelp) {
    this.nim = nim;
    this.nama = nama;
    this.notelp = notelp;
}

Codeium: Refactor | Explain | Generate Javadoc | ×
@Override
public String toString() {
    return "Mahasiswa{" + "nim=" + nim +", nama=" + nama + ",
    notelp=" + notelp + "}";
}
```

3. Membuat class list mahasiswa

```
import java.util.ArrayList;
import java.util.List;
Codeium: Refactor | Explain
public class ListMahasiswa08 {
    List<Mahasiswa08> mahasiswa08 = new ArrayList<>();
}
```

4. Menambahkan method tambah hapus update dan tampil

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
Codeium: Refactor | Explain
public class ListMahasiswa08 {
    List<Mahasiswa08> mahasiswas = new ArrayList<>();

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public void tambah(Mahasiswa08... mahasiswa) {
        mahasiswas.addAll(Arrays.asList(mahasiswa));
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public void hapus(int index) {
        mahasiswas.remove(index);
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public void update(int index, Mahasiswa08 mhs) {
        mahasiswas.set(index, mhs);
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public void tampil() {
        mahasiswas.stream().forEach(mhs -> {
            System.out.println("" + mhs.toString());
        });
    }
}
```

5. Menambahkan method linear search

```
int linearSearch(String nim) {
    for(int i=0; i< mahasiswas.size(); i++){
        if(nim.equals(mahasiswas.get(i).nim)){
            return i;
        }
    }
    return -1;
}
```

6. Menambahkan method main

```
public static void main(String[] args) {
    ListMahasiswa08 lm = new ListMahasiswa08();
    Mahasiswa08 m = new Mahasiswa08(nim:"201234", nama:"Noureen",
    notelp:"021xx1");
    Mahasiswa08 m1 = new Mahasiswa08(nim:"201235", nama:"Akhleena",
    notelp:"021xx2");
    Mahasiswa08 m2 = new Mahasiswa08(nim:"201236", nama:"Shannun",
    notelp:"021xx3");

    // menambahkan objek mahasiswa
    lm.tambah(m, m1, m2);

    // enanpilkan list mahasiswa
    lm.tampil();

    // Update mahasiswa
    lm.update(lm.linearSearch(nim:"201235"), new Mahasiswa08
    (nim:"201235",nama:"Aknhleena Lela", notelp:"021xx2"));
    System.out.println(x:"");

    lm.tampil();


}
```

VERIFIKASI HASIL PERCOBAAN 15.4.2

```
Mahasiswa{nim=201234, nama=Noureen, notelp=021xx1}
Mahasiswa{nim=201235, nama=Akhleena, notelp=021xx2}
Mahasiswa{nim=201236, nama=Shannun, notelp=021xx3}

Mahasiswa{nim=201234, nama=Noureen, notelp=021xx1}
Mahasiswa{nim=201235, nama=Aknhleena Lela, notelp=021xx2}
Mahasiswa{nim=201236, nama=Shannun, notelp=021xx3}
dzf@dzf-14ARE05:/media/dzf/DATA/1POLINEMA/2Genap 2023-2024/
```

PERTANYAAN 15.4.3

1. Pada fungsi tambah() yang menggunakan unlimited argument itu menggunakan konsep apa?Dan kelebihannya apa?

Menggunakan konsep stdarg (Variable Arguments) yang memungkinkan menerima jumlah argumen yang tidak terdefinisi.Kelebihan dari konsep ini yaitu memungkinkan fungsi menerima sejumlah argumen yang tidak terdefinisi, meningkatkan fleksibilitas kode.

2. Pada fungsi linearSearch() di atas, silakan diganti dengan fungsi binarySearch() dari collection!

```
Codeium: Refactor | Explain | Generate Javadoc | X
int binarySearch(String nim) {
    int low = 0;
    int high = mahasiswas.size() - 1;

    while (low <= high) {
        int mid = low + (high - low) / 2;
        Mahasiswa08 mhs = mahasiswas.get(mid);

        if (mhs.nim.equals(nim)) {
            return mid;
        } else if (mhs.nim.compareTo(nim) < 0) {
            low = mid + 1;
        } else {
            high = mid - 1;
        }
    }

    return -1;
}
```

3. Tambahkan fungsi sorting baik secara ascending ataupun descending pada class tersebut!

```
public void sortByNimAsc() {
    Collections.sort(mahasiswas, (mhs1, mhs2) -> mhs1.nim.compareTo
    (mhs2.nim));
}

Codeium: Refactor | Explain | Generate Javadoc | ×
public void sortByNimDesc() {
    Collections.sort(mahasiswas, (mhs1, mhs2) -> mhs2.nim.compareTo
    (mhs1.nim));
}
```

8.4 TUGAS

1. Buatlah implementasi program daftar nilai mahasiswa semester, minimal memiliki 3 class yaitu Mahasiswa, Nilai, dan Mata Kuliah. Data Mahasiswa dan Mata Kuliah perlu melalui penginputan data terlebih dahulu.

```
do {
    System.out.println
    (x:"=======================================");
    System.out.println(x:" SISTEM PENGOLAHAN DATA NILAI MAHASISWA
    [dzulfikar / 1FSIB - 08] ");
    System.out.println
    (x:"=======================================");
    System.out.println(x:"1. Input Nilai");
    System.out.println(x:"2. Tampil Nilai");
    System.out.println(x:"3. Mencari Nilai Mahasiswa");
    System.out.println(x:"4. Urut Data Nilai");
    System.out.println(x:"5. Keluar");

    System.out.print(s:"Pilih: ");
    pilihan = input.nextInt();
    input.nextLine();
```

```
switch (pilihan) {
    case 1:
        System.out.print(s:"Masukkan NIM: ");
        String nim = input.nextLine();
        System.out.print(s:"Masukkan Kode Mata Kuliah: ");
        String kodeMk = input.nextLine();
        System.out.print(s:"Masukkan Nilai: ");
        double nilai = input.nextDouble();
        sistem.inputNilai(nim, kodeMk, nilai);
        break;
    case 2:
        sistem.tampilNilai();
        break;
    case 3:
        System.out.print(s:"Masukkan NIM: ");
        nim = input.nextLine();
        sistem.cariNilaiMahasiswa(nim);
        break;
    case 4:
        sistem.urutDataNilai();
        break;
    case 5:
        return;
```

```
Codeium: Refactor | Explain | Generate Javadoc | ×
void tambahMahasiswa(String nim, String nama, String telf) {
    Mahasiswa mahasiswa = new Mahasiswa(nim, nama, telf);
    daftarMahasiswa.add(mahasiswa);
}

Codeium: Refactor | Explain | Generate Javadoc | ×
void tambahMataKuliah(String kode, String nama, int sks) {
    Matakuliah matakuliah = new Matakuliah(kode, nama, sks);
    daftarMataKuliah.add(matakuliah);
}
```

```
void tampilNilai() {
    System.out.println(x:"=======================================");
    System.out.println(x:"         DAFTAR NILAI MAHASISWA        ");
    System.out.println(x:"=======================================");
    for (Nilai nilai : daftarNilai) {
        System.out.println("NIM         : " + nilai.mahasiswa.nim);
        System.out.println("Nama        : " + nilai.mahasiswa.nama);
        System.out.println("Mata Kuliah : " + nilai.matakuliah.nama);
        System.out.println("SKS         : " + nilai.matakuliah.sks);
        System.out.println("Nilai       : " + nilai.nilai);
        System.out.println
        (x:"---------------------------------------");
    }
}
```

```
void inputNilai(String nim, String kodeMk, double nilai) {
    Mahasiswa mahasiswa = daftarMahasiswa.stream().filter(m -> m.nim.
    equals(nim)).findFirst().orElse(other:null);
    Matakuliah matakuliah = daftarMataKuliah.stream().filter(mk -> mk.
    kode.equals(kodeMk)).findFirst().orElse(other:null);

    if (mahasiswa != null && matakuliah != null) {
        Nilai nilaiObj = new Nilai(mahasiswa, matakuliah, nilai);
        daftarNilai.add(nilaiObj);
    } else {
        if (mahasiswa == null) {
            System.out.println("Mahasiswa dengan NIM " + nim + "
            tidak ditemukan.");
        } else {
            System.out.println("Mata kuliah dengan kode " + kodeMk +
            " tidak ditemukan.");
        }
    }
}
```

```
void cariNilaiMahasiswa(String nim) {
    System.out.println(x:"=======================================");
    System.out.println("       NILAI MAHASISWA NIM: " + nim);
    System.out.println(x:"=======================================");
    for (Nilai nilai : daftarNilai) {
        if (nilai.mahasiswa.nim.equals(nim)) {
            System.out.println("Nama        : " + nilai.mahasiswa.
            nama);
            System.out.println("Mata Kuliah : " + nilai.matakuliah.
            nama);
            System.out.println("SKS         : " + nilai.matakuliah.
            sks);
            System.out.println("Nilai       : " + nilai.nilai);
            System.out.println
            (x:"---------------------------------------");
        }
    }
}
```

```
void urutDataNilai() {
    daftarNilai.sort(Comparator.comparing(n -> n.mahasiswa.nama));
    tampilNilai();
}
```

```
=====================================
 SISTEM PENGOLAHAN DATA NILAI MAHASISWA [dzulf
8]
=====================================
1. Input Nilai
2. Tampil Nilai
3. Mencari Nilai Mahasiswa
4. Urut Data Nilai
5. Keluar
Pilih: 1
Masukkan NIM: 001
Masukkan Kode Mata Kuliah: 01
Masukkan Nilai: 90
-------------------------------------
```

```
Pilih: 2
=====================================
        DAFTAR NILAI MAHASISWA
=====================================
NIM         : 001
Nama        : abdur
Mata Kuliah : Internet of Things
SKS         : 3
Nilai       : 90.0
-------------------------------------
```

```
Pilih: 3
Masukkan NIM: 001
=====================================
       NILAI MAHASISWA NIM: 001
=====================================
Nama        : abdur
Mata Kuliah : Internet of Things
SKS         : 3
Nilai       : 90.0
-------------------------------------
```

```
Pilih: 4
=====================================
        DAFTAR NILAI MAHASISWA
=====================================
NIM         : 001
Nama        : abdur
Mata Kuliah : Internet of Things
SKS         : 3
Nilai       : 90.0
```

```
1. Input Nilai
2. Tampil Nilai
3. Mencari Nilai Mahasiswa
4. Urut Data Nilai
5. Keluar
Pilih: 5
dzf@dzf-14ARE05:/media/dzf/DATA/1POLINEMA/
ktikumAlgoritma/pertemuan15$
file /media/dzf/DATA/1POLINEMA/2Genap 20  Java: Ready
```

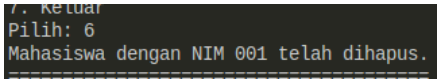
2. Tambahkan prosedur hapus data mahasiswa melalui implementasi Queue pada collections

```
Codeium: Refactor | Explain | Generate Javadoc | ×
void antrianHapusMahasiswa(String nim) {
    try {
        Mahasiswa mahasiswa = daftarMahasiswa.stream().filter(m -> m.
        nim.equals(nim)).findFirst().orElse(other:null);
        if (mahasiswa != null) {
            antrianHapus.add(mahasiswa);
            System.out.println("Mahasiswa dengan NIM " + nim + "
            ditambahkan ke dalam antrian penghapusan.");
        } else {
            System.out.println("Mahasiswa dengan NIM " + nim + "
            tidak ditemukan.");
        }
    } catch (NullPointerException e) {
        System.out.println(x:"Terjadi kesalahan saat mencari
        mahasiswa. Silakan periksa input NIM.");
    }
}

Codeium: Refactor | Explain | Generate Javadoc | ×
void hapusMahasiswa() {
    Mahasiswa mahasiswa = antrianHapus.poll();
    if (mahasiswa != null) {
        daftarMahasiswa.remove(mahasiswa);
        daftarNilai.removeIf(nilai -> nilai.mahasiswa.equals
        (mahasiswa));
        System.out.println("Mahasiswa dengan NIM " + mahasiswa.nim +
        " telah dihapus.");
    } else {
        System.out.println(x:"Tidak ada mahasiswa dalam antrian
        penghapusan.");
    }
}
```

```
=====================================
 SISTEM PENGOLAHAN DATA NILAI MAHASISWA [dzulf
8]
=====================================
1. Input Nilai
2. Tampil Nilai
3. Mencari Nilai Mahasiswa
4. Urut Data Nilai
5. Tambah Mahasiswa ke Antrian Penghapusan
6. Hapus Mahasiswa dari Antrian
7. Keluar
Pilih:
```

```
7. Keluar
Pilih: 5
Masukkan NIM Mahasiswa yang akan ditambahkan ke antrian pengha
pusan: 001
Mahasiswa dengan NIM 001 ditambahkan ke dalam antrian penghapu
san.
=====================================
```


7. Keluar
Pilih: 6
Mahasiswa dengan NIM 001 telah dihapus.