Nama : Ahmad Franklyn Bima Aquilla
Kelas : SIB 1F – 01
NIM : 2341760027
Matkul : Praktikum Algoritma dan Struktur Data
Github : https://github.com/JustQuill25/Jobsheet15PraktikumASD

PERTEMUAN 15

(ASD – Collection)

Praktikum 1

Class Collection01

```java
import java.util.ArrayList;
import java.util.LinkedList;
import java.util.List;

public class Collection01 {
    Run | Debug
    public static void main(String[] args) {
        // ArrayList
        List<Object> l = new ArrayList<>();
        l.add(e:1);
        l.add(e:2);
        l.add(e:3);
        l.add(e:"Cireng");

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

        // LinkedList
        List<String> names = new LinkedList<>();
        names.add(e:"Noureen");
        names.add(e:"Akhleema");
        names.add(e:"Shannum");
        names.add(e:"Uwais");
        names.add(e:"Al-Qarni");

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

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

Output

```
Elemen 0: 1 total elemen: 4 elemen terakhir: Cireng
Elemen 0: 2 total elemen: 4 elemen terakhir: 4
Elemen 0: Noureen total elemen: 5 elemen terakhir: Al-Qarni
Elemen 0: My kid total elemen: 5 elemen terakhir: Al-Qarni
Names: [My kid, Akhleema, Shannum, Uwais, Al-Qarni]
PS D:\Praktikum Algoritma dan Struktur Data\Jobsheet15>
```

16.2.1. Pertanyaan Percobaan

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

   ➢ Ada 3 alasan mengapa semua jenis data bisa ditampung ke dalam sebuah arrayList :

   1. Generik <Object>: Deklarasi ArrayList<Object> menggunakan generik <Object>, yang berarti kita bisa menyimpan objek dari tipe data apapun di dalamnya. Karena Object adalah superclass dari semua kelas di Java, kita dapat dengan aman menyimpan objek dari berbagai jenis tipe data seperti Integer, String, Double, atau bahkan objek kustom yang kita buat sendiri.

   2. Polimorfisme: Polimorfisme memungkinkan objek dari kelas-kelas yang berbeda untuk dianggap sebagai objek dari superclass mereka. Dalam konteks ArrayList<Object>, semua objek dapat dianggap sebagai objek Object, sehingga kita dapat menambahkan dan mengakses objek dari tipe data apapun.

   3. Dynamic Array: ArrayList di Java adalah implementasi dari struktur data array yang memungkinkan untuk penambahan dan penghapusan elemen secara dinamis. Ini memungkinkan kita untuk secara fleksibel menyesuaikan isi dari ArrayList<Object> sesuai dengan kebutuhan, tanpa batasan terhadap jenis data yang dapat disimpan.

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

```
6   public static void main(String[] args) {
7       // ArrayList yang hanya menampung Integer
8       List<Integer> l = new ArrayList<>();
9       l.add(e:1);
10      l.add(e:2);
11      l.add(e:3);
12      l.add(e:4); // Mengganti "Cireng" dengan angka 4
13
14      System.out.printf(format:"Elemen 0: %d total elemen: %d elemen terakhir: %d\n",
15              l.get(index:0), l.size(), l.get(l.size() - 1));
16
17      l.add(e:5);
18      l.remove(index:0);
19      System.out.printf(format:"Elemen 0: %d total elemen: %d elemen terakhir: %d\n",
20              l.get(index:0), l.size(), l.get(l.size() - 1));
```

Output

```
Elemen 0: 1 total elemen: 4 elemen terakhir: 4
Elemen 0: 2 total elemen: 4 elemen terakhir: 5
Elemen 0: Noureen total elemen: 5 elemen terakhir: Al-Qarni
Elemen 0: My kid total elemen: 5 elemen terakhir: Al-Qarni
Names: [My kid, Akhleema, Shannum, Uwais, Al-Qarni]
PS D:\Praktikum Algoritma dan Struktur Data\Jobsheet15>
```

3. Ubah kode pada baris kode 38 menjadi seperti ini

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

```
// LinkedList
LinkedList<String> names = new LinkedList<>();
names.add(e:"Noureen");
names.add(e:"Akhleema");
names.add(e:"Shannum");
names.add(e:"Uwais");
names.add(e:"Al-Qarni");
```

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());
```

```
// Menambahkan elemen ke awal LinkedList menggunakan push
names.push(e:"Mei-mei");
System.out.printf(format:"Elemen 0: %s total elemen: %d 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!
   - **Output**

```
Elemen 0: 1 total elemen: 4 elemen terakhir: Cireng
Elemen 0: 2 total elemen: 4 elemen terakhir: 4
Elemen 0: Noureen total elemen: 5 elemen terakhir: Al-Qarni
Elemen 0: My kid total elemen: 5 elemen terakhir: Al-Qarni
Names: [My kid, Akhleema, Shannum, Uwais, Al-Qarni]
Elemen 0: Mei-mei total elemen: 6 elemen terakhir: Al-Qarni
Names: [Mei-mei, My kid, Akhleema, Shannum, Uwais, Al-Qarni]
PS D:\Praktikum Algoritma dan Struktur Data\Jobsheet15>
```

   - **Penambahan dengan push**

Metode `push` menambahkan elemen baru di awal LinkedList, bukan di akhir atau di posisi lain.

   - **Akses Elemen**

Dengan menggunakan metode getFirst() dan getLast(), kita dapat dengan mudah mengakses elemen pertama dan terakhir dari LinkedList.

   - **Dinamika Ukuran**

LinkedList dapat dengan mudah mengakomodasi penambahan dan penghapusan elemen, dengan ukuran yang menyesuaikan secara dinamis.

Praktikum 2

Class LoopCollection01

```java
import java.util.Iterator;
import java.util.Stack;

public class LoopCollection01 {
    Run | Debug
    public static void main(String[] args) {
        Stack<String> fruits = new Stack<>();
        fruits.push(item:"Banana");
        fruits.add(e:"Orange");
        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());
        }

        // Tambahan potongan kode
        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));
        }
    }
}
```

Output

```
Banana Orange Watermelon Leci Salak
[Banana, Orange, Watermelon, Leci, Salak]
Salak Leci Watermelon Orange Banana
Melon Durian
Melon Durian
Melon Durian
PS D:\Praktikum Algoritma dan Struktur Data\Jobsheet15>
```

16.3.1. Pertanyaan Percobaan

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

   - **Fungsi "Push"** : `push()` adalah metode yang spesifik untuk kelas `Stack`. Metode ini digunakan untuk menambahkan elemen ke bagian atas stack. Ketika `push()` dipanggil, elemen baru akan ditempatkan di posisi paling atas dari stack.
   - **Fungsi "Add()"** : `add()` adalah metode yang digunakan oleh banyak koleksi dalam Java, termasuk `Stack`, `ArrayList`, `LinkedList`, dan lain-lain. Metode ini menambahkan elemen ke akhir koleksi. Ketika `add()` dipanggil pada objek stack, elemen baru akan ditempatkan di akhir stack, meskipun secara konsep, stack seharusnya menggunakan `push()` untuk menambahkan elemen.

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

```
Banana Orange Watermelon Leci Salak
[Banana, Orange, Watermelon, Leci, Salak]
Salak Leci Watermelon Orange Banana


PS D:\Praktikum Algoritma dan Struktur Data\Jobsheet15>
```

3. Jelaskan fungsi dari baris 46-49?
   - Baris 46-49 berfungsi untuk melakukan iterasi melalui semua elemen dalam stack fruits menggunakan Iterator, dan mencetak setiap elemen dengan format tertentu (dalam hal ini, hanya menambahkan spasi di antara elemen-elemen). Ini merupakan salah satu cara yang efektif untuk melakukan iterasi dan pengolahan data dalam sebuah koleksi seperti stack di Java. Dengan menggunakan Iterator, kita dapat mengakses dan memanipulasi elemen-elemen dalam koleksi secara terstruktur dan efisien, tanpa harus mengakses langsung menggunakan indeks seperti pada array. Ini juga memungkinkan kita untuk melakukan operasi seperti pencarian, penghapusan, atau penggantian elemen dengan lebih mudah dalam konteks iteratif.

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

   ➢ Yang terjadi maka kita akan menggunakan antarmuka List untuk deklarasi objek fruits dan menginisialisasinya dengan sebuah ArrayList (atau jenis list lainnya). Mengapa demikian, ulasannya sebagai berikut :

   1. Fleksibilitas Koleksi
      - Dengan menggunakan List<String>, kita dapat menggunakan berbagai implementasi dari antarmuka List seperti ArrayList, LinkedList, Vector, dan lainnya. Ini memberikan fleksibilitas dalam memilih struktur data yang sesuai dengan kebutuhan aplikasi kita.
   2. Operasi Dasar
      - Menggunakan List memungkinkan kita untuk melakukan operasi dasar seperti add() untuk menambahkan elemen, remove() untuk menghapus elemen, dan get() untuk mengakses elemen berdasarkan indeks.
   3. Perubahan pada Implementasi
      - Perubahan ini menggambarkan transisi dari menggunakan struktur data stack yang khusus (dengan operasi push() dan pop()) ke penggunaan koleksi yang lebih umum (dengan berbagai operasi yang dapat dilakukan pada List).
   4. Keterbatasan Perilaku Stack
      - Stack hanya mendukung operasi Last-In-First-Out (LIFO), sementara List memberikan fleksibilitas lebih dalam cara kita mengakses dan memanipulasi elemen-elemen dalam koleksi.

   Dengan demikian, menggunakan List<String> sebagai tipe data koleksi memberikan kemampuan yang lebih luas dalam memanajemen data dan lebih cocok untuk kebutuhan umum dalam pengembangan aplikasi Java.

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

   ➢ **Code (Modify)**

```
// Mengganti elemen terakhir dengan "Strawberry"
fruits.set(fruits.size() - 1, element:"Strawberry");

// Menampilkan isi fruits setelah perubahan
System.out.println(x:"Fruits setelah perubahan:");
for (String fruit : fruits) {
    System.out.printf(format:"%s ", fruit);
}
System.out.println("\n" + fruits.toString());
```

   ➢ **Output (Modify)**

```
Fruits setelah perubahan:
Banana Orange Watermelon Leci Strawberry
[Banana, Orange, Watermelon, Leci, Strawberry]
Banana Orange Watermelon Leci Strawberry
Melon Durian
Melon Durian
Melon Durian
PS D:\Praktikum Algoritma dan Struktur Data\Jobsheet15>
```

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

```
// Menambahkan elemen baru
fruits.add(e:"Mango");
fruits.add(e:"Guava");
fruits.add(e:"Avocado");

// Melakukan sorting (pengurutan)
Collections.sort(fruits);

// Menampilkan isi fruits setelah penambahan dan sorting
System.out.println(x:"Fruits setelah penambahan dan sorting:");
for (String fruit : fruits) {
    System.out.printf(format:"%s ", fruit);
}
System.out.println("\n" + fruits.toString());
```

   - **Output**

```
Fruits setelah penambahan dan sorting:
Avocado Banana Guava Leci Mango Orange Salak Watermelon
[Avocado, Banana, Guava, Leci, Mango, Orange, Salak, Watermelon]
Avocado Banana Guava Leci Mango Orange Salak Watermelon
Melon Durian
Melon Durian
Melon Durian
PS D:\Praktikum Algoritma dan Struktur Data\Jobsheet15>
```

Praktikum 3

Class Mahasiswa01

```java
package Praktikum3;

public class Mahasiswa01 {
    String nim;
    String nama;
    String notelp;

    public Mahasiswa01() {
    }

    public Mahasiswa01(String nim, String nama, String notelp) {
        this.nim = nim;
        this.nama = nama;
        this.notelp = notelp;
    }

    @Override
    public String toString() {
        return "Mahasiswa01{" +
                "nim='" + nim + '\'' +
                ", nama='" + nama + '\'' +
                ", notelp='" + notelp + '\'' +
                '}';
    }
}

```

Class ListMahasiswa01

```java
package Praktikum3;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class ListMahasiswa01 {
    List<Mahasiswa01> mahasiswas = new ArrayList<>();

    public void tambah(Mahasiswa01... mahasiswa) {
        mahasiswas.addAll(Arrays.asList(mahasiswa));
    }

    public void hapus(int index) {
        mahasiswas.remove(index);
    }

    public void update(int index, Mahasiswa01 mhs) {
        mahasiswas.set(index, mhs);
    }

    public void tampil() {
        mahasiswas.forEach(mhs -> {
            System.out.println("" + mhs.toString());
        });
    }

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

```java
    Run | Debug
    public static void main(String[] args) {
        ListMahasiswa01 lm = new ListMahasiswa01();
        Mahasiswa01 m = new Mahasiswa01(nim:"201234", nama:"Noureen", notelp:"021xx1");
        Mahasiswa01 m1 = new Mahasiswa01(nim:"201235", nama:"Akhleema", notelp:"021xx2");
        Mahasiswa01 m2 = new Mahasiswa01(nim:"201236", nama:"Shannum", notelp:"021xx3");

        lm.tambah(m, m1, m2);

        System.out.println(x:"Setelah penambahan:");
        lm.tampil();

        lm.update(lm.linearSearch(nim:"201235"), new Mahasiswa01(nim:"201235", nama:"Akhleema Lela", notelp:"021xx2"));
        System.out.println(x:"\nSetelah update:");
        lm.tampil();
    }
}
```

Output

```
Setelah penambahan:
Mahasiswa01{nim='201234', nama='Noureen', notelp='021xx1'}
Mahasiswa01{nim='201235', nama='Akhleema', notelp='021xx2'}
Mahasiswa01{nim='201236', nama='Shannum', notelp='021xx3'}

Setelah update:
Mahasiswa01{nim='201234', nama='Noureen', notelp='021xx1'}
Mahasiswa01{nim='201235', nama='Akhleema Lela', notelp='021xx2'}
Mahasiswa01{nim='201236', nama='Shannum', notelp='021xx3'}
PS D:\Praktikum Algoritma dan Struktur Data\Jobsheet15>
```

16.4.1. Pertanyaan Percobaan

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

➢ Fungsi tambah() yang menggunakan unlimited argument (varargs) pada metode tambah(Mahasiswa01... mahasiswa) menggunakan konsep **varargs** (variable-length arguments). Konsep ini memungkinkan sebuah metode untuk menerima jumlah argumen yang tidak terbatas dari jenis yang sama.

➢ Kelebihan dan Konsep Penggunaan Varargs :

1. **Fleksibilitas**

- Varargs memungkinkan kita untuk memanggil metode dengan jumlah argumen yang bervariasi, mulai dari tidak ada argumen sama sekali hingga banyak argumen sekaligus. Misalnya, dalam kasus `tambah()` ini, kita bisa menambahkan satu, dua, atau lebih objek `Mahasiswa01` secara langsung dalam satu pemanggilan metode.

2. **Sederhana dalam Implementasi**

- Daripada harus membuat array atau daftar terlebih dahulu sebelum memanggil metode, varargs memungkinkan kita untuk langsung menyediakan argumen yang dibutuhkan tanpa perlu melakukan inisialisasi array secara eksplisit.

3. **Meningkatkan Keterbacaan**

- Penggunaan varargs juga dapat meningkatkan keterbacaan kode karena menghilangkan kebutuhan untuk menyusun atau memformat argumen dalam array sebelum mengirimkannya ke metode.

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

➢ **Modify Code ListMahasiswa01**

```
public int binarySearch(String nim) {
    // Memastikan list diurutkan berdasarkan nim sebelum menggunakan binarySearch
    Collections.sort(mahasiswas, (m1, m2) -> m1.nim.compareTo(m2.nim));

    // Melakukan binary search
    Mahasiswa01 key = new Mahasiswa01(nim, nama:"", notelp:"");
    int index = Collections.binarySearch(mahasiswas, key, (m1, m2) -> m1.nim.compareTo(m2.nim));

    return index;
}
```

```
// Menggunakan binarySearch untuk mencari mahasiswa berdasarkan nim
int index = lm.binarySearch(nim:"201235");
if (index >= 0) {
    System.out.println("\nMahasiswa ditemukan pada indeks: " + index);
    System.out.println("Detail mahasiswa: " + lm.mahasiswas.get(index));
} else {
    System.out.println(x:"\nMahasiswa tidak ditemukan.");
}
```

3. Tambahkan fungsi sorting baik secara ascending ataupun descending pada class tersebut!
   - **Modify Code**

```
// Menambahkan fungsi sorting ascending
public void sortAscending() {
    Collections.sort(mahasiswas, Comparator.comparing(m -> m.nim));
}

// Menambahkan fungsi sorting descending
public void sortDescending() {
    Collections.sort(mahasiswas, (m1, m2) -> m2.nim.compareTo(m1.nim));
}
```

   - **Output**

```
Setelah penambahan:
Mahasiswa01{nim='201234', nama='Noureen', notelp='021xx1'}
Mahasiswa01{nim='201235', nama='Akhleema', notelp='021xx2'}
Mahasiswa01{nim='201236', nama='Shannum', notelp='021xx3'}

Mahasiswa ditemukan pada indeks: 1
Detail mahasiswa: Mahasiswa01{nim='201235', nama='Akhleema', notelp='021xx2'}

Sorting ascending:
Mahasiswa01{nim='201234', nama='Noureen', notelp='021xx1'}
Mahasiswa01{nim='201235', nama='Akhleema', notelp='021xx2'}
Mahasiswa01{nim='201236', nama='Shannum', notelp='021xx3'}

Sorting descending:
Mahasiswa01{nim='201236', nama='Shannum', notelp='021xx3'}
Mahasiswa01{nim='201235', nama='Akhleema', notelp='021xx2'}
Mahasiswa01{nim='201234', nama='Noureen', notelp='021xx1'}
PS D:\Praktikum Algoritma dan Struktur Data\Jobsheet15>
```

Tugas Praktikum

Class Mahasiswa01

```java
package tugas;

public class Mahasiswa01 {
    String nim;
    String nama;
    String telepon;

    public Mahasiswa01(String nim, String nama, String telepon) {
        this.nim = nim;
        this.nama = nama;
        this.telepon = telepon;
    }

    @Override
    public String toString() {
        return "Mahasiswa{" +
                "nim='" + nim + '\'' +
                ", nama='" + nama + '\'' +
                ", telepon='" + telepon + '\'' +
                '}';
    }
}
```

Class MataKuliah01

```java
package tugas;

public class MataKuliah01 {
    String kodeMK;
    String namaMK;
    int sks;

    public MataKuliah01(String kodeMK, String namaMK, int sks) {
        this.kodeMK = kodeMK;
        this.namaMK = namaMK;
        this.sks = sks;
    }

    @Override
    public String toString() {
        return "MataKuliah{" +
                "kodeMK='" + kodeMK + '\'' +
                ", namaMK='" + namaMK + '\'' +
                ", sks=" + sks +
                '}';
    }
}
```

Class Nilai01

```java
package tugas;

public class Nilai01 {
    String nim;
    String kodeMK;
    double nilai;

    public Nilai01(String nim, String kodeMK, double nilai) {
        this.nim = nim;
        this.kodeMK = kodeMK;
        this.nilai = nilai;
    }

    @Override
    public String toString() {
        return "Nilai{" +
                "nim='" + nim + '\'' +
                ", kodeMK='" + kodeMK + '\'' +
                ", nilai=" + nilai +
                '}';
    }
}
```

Class MainProgram01

```java
package tugas;

import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class MainProgram01 {
    static List<Mahasiswa01> daftarMahasiswa = new ArrayList<>();
    static List<MataKuliah01> daftarMataKuliah = new ArrayList<>();
    static List<Nilai01> daftarNilai = new ArrayList<>();
    static Scanner scanner = new Scanner(System.in);

    Run | Debug
    public static void main(String[] args) {
```

```java
        daftarMahasiswa.add(new Mahasiswa01(nim:"20001", nama:"Thalhah", telepon:"021123"));
        daftarMahasiswa.add(new Mahasiswa01(nim:"20002", nama:"Zubair", telepon:"021124"));
        daftarMahasiswa.add(new Mahasiswa01(nim:"20003", nama:"AbdurRahman", telepon:"021125"));
        daftarMahasiswa.add(new Mahasiswa01(nim:"20004", nama:"Sa'ad", telepon:"021126"));
        daftarMahasiswa.add(new Mahasiswa01(nim:"20005", nama:"Sa'id", telepon:"021127"));
        daftarMahasiswa.add(new Mahasiswa01(nim:"20006", nama:"Ubaidah", telepon:"021128"));

        daftarMataKuliah.add(new MataKuliah01(kodeMK:"00001", namaMK:"Internet of Things", sks:3));
        daftarMataKuliah.add(new MataKuliah01(kodeMK:"00002", namaMK:"Algoritma dan Struktur Data", sks:2));
        daftarMataKuliah.add(new MataKuliah01(kodeMK:"00003", namaMK:"Algoritma dan Pemrograman", sks:2));
        daftarMataKuliah.add(new MataKuliah01(kodeMK:"00004", namaMK:"Praktikum Algoritma dan Struktur Data", sks:3));
        daftarMataKuliah.add(new MataKuliah01(kodeMK:"00005", namaMK:"Praktikum Algoritma dan Pemrograman", sks:3));

        // Memanggil menu utama
        menuUtama();
    }
```

```java
    public static void menuUtama() {
        int pilihan;
        do {
            System.out.println(x:"=========================================");
            System.out.println(x:"SISTEM PENGOLAHAN NILAI MAHASISWA PUNYA BIMA");
            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.println(x:"=========================================");
            System.out.print(s:"Pilih\t\t: ");
            pilihan = scanner.nextInt();
            switch (pilihan) {
                case 1:
                    inputNilai();
                    break;
                case 2:
                    tampilNilai();
                    break;
                case 3:
                    cariNilaiMahasiswa();
                    break;
                case 4:
                    urutDataNilai();
                    break;
                case 5:
                    System.out.println(x:"Terima kasih telah menggunakan program ini.");
                    break;
                default:
                    System.out.println(x:"Pilihan tidak valid.");
            }
        } while (pilihan != 5);
    }
```

```java
    public static void inputNilai() {
        scanner.nextLine();
        System.out.println(x:"\n====  Masukan data  =====");
        System.out.print(s:"Kode\t\t: ");
        String kodeMK = scanner.nextLine();
        System.out.print(s:"Nilai\t\t: ");
        double nilai = scanner.nextDouble();

        System.out.println(x:"\nDAFTAR MAHASISWA");
        System.out.println(x:"=========================================");
        System.out.println(x:"NIM\t\tNama\t\tTelp");
        for (Mahasiswa01 mahasiswa : daftarMahasiswa) {
            System.out.println(mahasiswa.nim + "\t\t" + mahasiswa.nama + "\t\t" + mahasiswa.telepon);
        }

        scanner.nextLine();
        System.out.print(s:"\nPilih mahasiswa by nim : ");
        String nim = scanner.nextLine();

        System.out.println(x:"\nDAFTAR MATA KULIAH");
        System.out.println(x:"=========================================");
        System.out.println(x:"Kode\t\tMata Kuliah\t\t\tSKS");
        for (MataKuliah01 mataKuliah : daftarMataKuliah) {
            System.out.println(mataKuliah.kodeMK + "\t\t" + mataKuliah.namaMK + "\t\t\t\t" + mataKuliah.sks);
        }

        System.out.print(s:"\nPilih MK by Kode\t: ");
        String kodeMataKuliah = scanner.nextLine();

        // Menambahkan nilai ke daftarNilai
        daftarNilai.add(new Nilai01(nim, kodeMataKuliah, nilai));
        System.out.println(x:"\nData nilai berhasil dimasukkan.\n");
    }
```

```java
    public static void tampilNilai() {
        System.out.println(x:"\nDAFTAR NILAI MAHASISWA");
        System.out.println(x:"========================================");
        System.out.println(x:"NIM\t\tNama\t\tMata Kuliah\t\t\t\tNilai");
        for (Nilai01 nilai : daftarNilai) {
            String namaMahasiswa = getNamaMahasiswa(nilai.nim);
            String namaMataKuliah = getNamaMataKuliah(nilai.kodeMK);
            System.out.printf(format:"%s\t\t%s\t\t%s\t\t\t\t%.2f\n", nilai.nim, namaMahasiswa, namaMataKuliah, nilai.nilai);
        }
        System.out.println();
    }

    public static void cariNilaiMahasiswa() {
        scanner.nextLine();
        System.out.print(s:"\nMasukkan NIM Mahasiswa : ");
        String nim = scanner.nextLine();

        boolean found = false;
        System.out.println(x:"\nDAFTAR NILAI MAHASISWA");
        System.out.println(x:"========================================");
        System.out.println(x:"NIM\t\tNama\t\tMata Kuliah\t\t\t\tNilai");
        for (Nilai01 nilai : daftarNilai) {
            if (nilai.nim.equals(nim)) {
                String namaMahasiswa = getNamaMahasiswa(nilai.nim);
                String namaMataKuliah = getNamaMataKuliah(nilai.kodeMK);
                System.out.printf(format:"%s\t\t%s\t\t%s\t\t\t\t%.2f\n", nilai.nim, namaMahasiswa, namaMataKuliah,
                        nilai.nilai);
                found = true;
            }
        }
        if (!found) {
            System.out.println("Nilai tidak ditemukan untuk mahasiswa dengan NIM " + nim);
        }
        System.out.println();
    }

    public static void urutDataNilai() {

    }

    public static String getNamaMahasiswa(String nim) {
        for (Mahasiswa01 mahasiswa : daftarMahasiswa) {
            if (mahasiswa.nim.equals(nim)) {
                return mahasiswa.nama;
            }
        }
        return "N/A";
    }

    public static String getNamaMataKuliah(String kodeMK) {
        for (MataKuliah01 mataKuliah : daftarMataKuliah) {
            if (mataKuliah.kodeMK.equals(kodeMK)) {
                return mataKuliah.namaMK;
            }
        }
        return "N/A";
    }
}
```

Output

```
==========================================
SISTEM PENGOLAHAN NILAI MAHASISWA PUNYA BIMA
==========================================
1. Input Nilai
2. Tampil Nilai
3. Mencari Nilai Mahasiswa
4. Urut Data Nilai
5. Keluar
==========================================
Pilih           :
```

```
==========================================
SISTEM PENGOLAHAN NILAI MAHASISWA PUNYA BIMA
==========================================
1. Input Nilai
2. Tampil Nilai
3. Mencari Nilai Mahasiswa
4. Urut Data Nilai
5. Keluar
==========================================
Pilih           : 2

DAFTAR NILAI MAHASISWA
==========================================
NIM             Nama            Mata Kuliah
20001           Thalhah         Internet of Things
```

```
==========================================
Pilih           : 1

=====  Masukan data  =====
Kode            : 0001
Nilai           : 80.75

DAFTAR MAHASISWA
==========================================
NIM             Nama            Telp
20001           Thalhah         021123
20002           Zubair          021124
20003           AbdurRahman             021125
20004           Sa'ad           021126
20005           Sa'id           021127
20006           Ubaidah         021128

Pilih mahasiswa by nim : 20001

DAFTAR MATA KULIAH
==========================================
Kode      Mata Kuliah                           SKS
------------------------------------------
00001     Internet of Things                    3
00002     Algoritma dan Struktur Data           2
00003     Algoritma dan Pemrograman             2
00004     Praktikum Algoritma dan Struktur Data 3
00005     Praktikum Algoritma dan Pemrograman   3
```