Pertanyaan

16.2.3. Pertanyaan Percobaan

1. Perhatikan baris kode 2536, mengapa semua jenis data bisa ditampung ke dala m sebuah Arraylist?
   Jawab :
   Karena pada baris kode 25-36 menggunakan list dengan tipe data dinamis (tidak diberi kurung sudut "<>") dan tidak dapat menentukan tipe data. Jadi, arraylist yang ditambahkan secara dinamis, dapat menampung semua data (tipe data).

2. Modifikasi baris kode 25-36 sehingga data yang ditampung hanya satu jenis atau spesifik tipe tertentu!
   Jawab :
   Dengan mengganti intansiasi list array menjadi **List<Integer> l = new ArrayList<>();**

```
l.add(1);
l.add(2);
l.add(3);
System.out.printf("Elemen 0: %d total elemen: %d elemen terakhir: %s\n"
,l.get(0), l.size(), l.get(l.size() - 1));
l.add(4);
l.remove(0);
System.out.printf("Elemen 0: %d total elemen: %d elemen terakhir: %s\n"
,l.get(0), l.size(), l.get(l.size() - 1));
```

3. Ubah kode pada baris kode 38 menjadi seperti ini
   LinkedList<String> names = new LinkedList<>();
   Jawab :
   - Jika diubah menjadi kode baris seperti di atas, maka akan terjadi error, seperti gambar di bawah ini:

```
LinkedList<String> names = new LinkedList<>();
names.add(1);
names.add(2);
names.add(3);
names.add("Cireng");
System.out.printf("Elemen 0: %d total elemen: %d elemen terakhir: %s\n"
,names.get(0), names.size(), names.get(names.size() - 1));
```

   - Dan untuk perbaikannya seperti berikut disertai dengan hasil compilenya

```java
public class Pertanyaan1_ContohList{
    public static void main(String[] args) {

        List<Integer> l = new ArrayList<>();
        l.add(1);
        l.add(2);
        l.add(3);
        System.out.printf("Elemen 0: %d total elemen: %d elemen terakhir: 
        ,l.get(0), l.size(), l.get(l.size() - 1));

        l.add(4);
        l.remove(0);

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

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

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

        names.set(0, "My kid");

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

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

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());
   Jawab :

```java
List<Integer> l = new ArrayList<>();
l.add(1);
l.add(2);
l.add(3);
System.out.printf("Elemen 0: %d total elemen: %d elemen terakhir: %s\n"
,l.get(0), l.size(), l.get(l.size() - 1));

l.add(4);
l.remove(0);

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

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

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

names.set(0, "My kid");

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

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

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());
```

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

   Jawab :

```
Elemen 0: 1 total elemen: 3 elemen terakhir: 3
Elemen 0: 2 total elemen: 3 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]
```

16.3.3 Pertanyaan Percobaan

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

Jawab :

Yang membedakan keduanya ialah pada jenis struktur data yang digunakan. Jika fungsi push() merupakan stack, sedangkan fungsi add() merupakan list.

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

Jawab :

Melon dan Durian akan terhapus karena tidak ada penambahan data (push) Melon dan Durian.

- Kondisi sebelum dihapus:

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

- Kondisi setelah dihapus:

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

3. Jelaskan fungsi dari baris 46-49?

Jawab :

Fungsi dari baris 46-49 ialah untuk menginisialisasi fungsi iterator dengan tipe data String. `hashNext()` digunakan untuk mengecek apakah iterator memiliki elemet berikutnya atau tidak. Jika bernilai ****true****, maka variabel fruit akan menyimpan nilai berikutnya. Kemudian ditampilkan elemen tersebut.

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

Jawab :

Akan terjadi error, karena list tidak compatible dengan fungsi-fungsi dan inisialisasi dengan stack.

```
List<String> fruits = new Stack<>();
fruits.push("Banana");
fruits.add("Orange");
fruits.add("Watermelon");
fruits.add("Leci");
fruits.add("Salak");
for(String fruit : fruits){
    System.out.printf("%s ", fruit);
}
System.out.println("\n" + fruits.toString());
while(!fruits.empty()){
    System.out.printf("%s ", fruits.pop());
```

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

Jawab :

- Code:

```
Stack<String> fruits = new Stack<>();
fruits.push("Banana");
fruits.add("Orange");
fruits.add("Watermelon");
fruits.add("Leci");
fruits.add("Salak");
for(String fruit : fruits){
    System.out.printf("%s ", fruit);
}
System.out.println("\n" + fruits.toString());
while(!fruits.empty()){
    System.out.printf("%s ", fruits.pop());
}
fruits.push("Melon");
fruits.push("Strawberry");
```

- Output:

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

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

Jawab :

- Code:

```
fruits.push("Mango");
fruits.push("Guava");
fruits.push("Avocado");
fruits.sort((String i, String j) -> i.compareTo(j));
System.out.println("");
System.out.println("\nSetelah data telah di sorting");
for(int i=0; i<fruits.size(); i++){
System.out.printf("%s ", fruits.get(i));
    String fruit = it.next();
    System.out.printf("%s ", fruit);
}
```

- Output:

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

Setelah Data telah di Sorting
Avocado Guava Mango Melon Strawberry
```

16.4.3. Pertanyaan Percobaan

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

Jawab :

ArrayList, alasannya ialah karena dengan konsep itu dapat memungkinkan metode untuk mengambil sejumlah argumen, dapat diakses sebagai array dalam method.

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

Jawab :

- Code:

```
package Pertemuan_16.Praktikum_3;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.Comparator;
import java.util.LinkedList;
import java.util.List;
public class Mahasiswa_Pertanyaan_2 {

        String nim;
        String nama;
        String notelp;
        public Mahasiswa_Pertanyaan_2 () {
        }
        public Mahasiswa_Pertanyaan_2(String nim, String nama, String notelp){
            this.nim = nim;
            this.nama = nama;
            this.notelp = notelp;
        }
        public String toString(){
            return "Mahasiswa{" + "nim= " + nim + ", nama=" + nama + ", notelp=" +
notelp + '}';
        }
    }
    class ListMahasiswa {
        List<Mahasiswa> mahasiswas = new ArrayList<>();
        public void tambah(Mahasiswa... mahasiswa) {
           mahasiswas.addAll(Arrays.asList(mahasiswa));
        }
        public void hapus(int index) {
           mahasiswas.remove(index);
        }
        public void update(int index, Mahasiswa mhs) {
           mahasiswas.set(index, mhs);
        }
        public void tampil() {
           mahasiswas.stream().forEach(mhs -> {
              System.out.println("" + mhs.toString());
```

```
            });
        }
        int linearSearch(String nim) {
            for (int i = 0; i < mahasiswas.size(); i++) {
                if (nim.equals(mahasiswas.get(i).nim)) {
                    return i;
                }
            }
            return -1;
        }
        public static void main(String[] args) {
            ListMahasiswa lm = new ListMahasiswa();
            Mahasiswa m = new Mahasiswa("201234", "Noureen","021xx1");
            Mahasiswa m1 = new Mahasiswa("201235", "Akhleena", "021xx2");
            Mahasiswa m2 = new Mahasiswa("201236", "Shannum", "021xx3");
            lm.tambah(m, m1, m2);
            lm.tampil();
            lm.update(lm.linearSearch("201235"), new Mahasiswa("201235", "Akhleena
Lela", "021xx2"));
            System.out.println("");
            lm.tampil();
            Mahasiswa key = new Mahasiswa("201235", null, null);
            lm.update(Collections.binarySearch(lm.mahasiswas, key, new
MhsComparator()),
                    new Mahasiswa("201235", "Akhleema", "021xx2"));
            System.out.println("");
            lm.tampil();
        }
    }
class MhsComparator implements Comparator<Mahasiswa> {
        public int compare(Mahasiswa mhs1, Mahasiswa mhs2) {
            if (mhs1.nim == mhs2.nim) {
                return 0;
            } else {
                return -1;
            }
        }
        public Comparator<Mahasiswa> reversed() {
            return Comparator.super.reversed();
        }
    }
```

- Output:

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

Mahasiswa{nim= 201234, nama=Noureen, notelp=021xx1}
Mahasiswa{nim= 201235, nama=Akhleema Lela, notelp=021xx2}
Mahasiswa{nim= 201236, nama=Shannum, notelp=021xx3}
```

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

Jawab :

- Code:

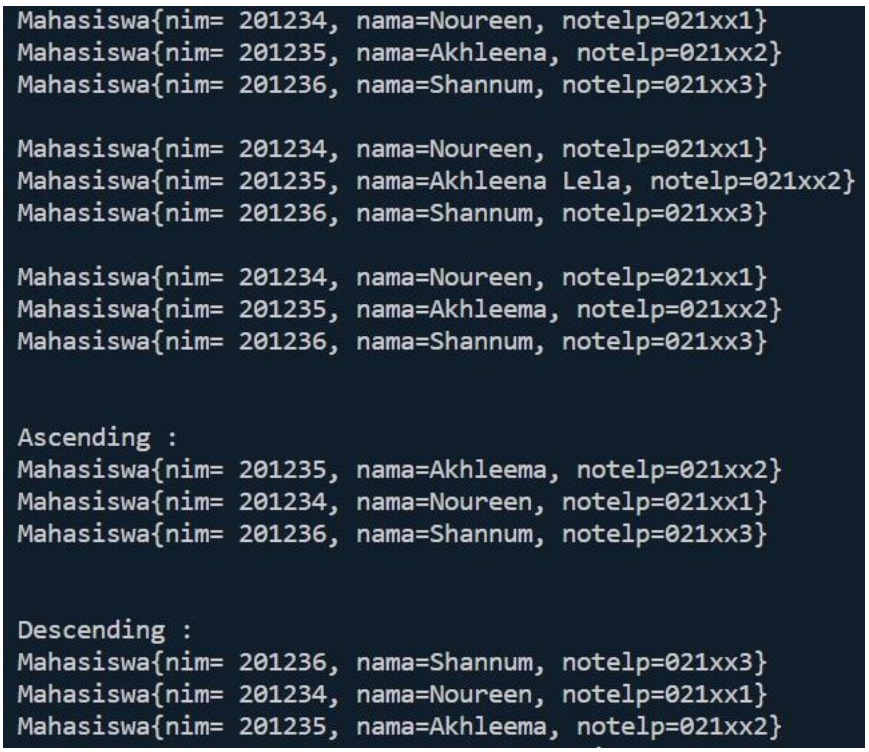
```
package Pertemuan_16.Praktikum_3;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.Comparator;
import java.util.LinkedList;
import java.util.List;
class Mahasiswa {
        String nim;
        String nama;
        String notelp;
        public Mahasiswa () {
        }
        public Mahasiswa(String nim, String nama, String notelp){
            this.nim = nim;
            this.nama = nama;
            this.notelp = notelp;
        }
        public String toString(){
            return "Mahasiswa{" + "nim= " + nim + ", nama=" + nama + ", notelp=" +
notelp + '}';
        }
}
public class ListMahasiswa_No_3 {
    List<Mahasiswa> mahasiswas = new ArrayList<>();
    public void tambah(Mahasiswa... mahasiswa) {
        mahasiswas.addAll(Arrays.asList(mahasiswa));
    }
    public void hapus(int index) {
        mahasiswas.remove(index);
    }
    public void update(int index, Mahasiswa mhs) {
        mahasiswas.set(index, mhs);
    }
    public void tampil() {
        mahasiswas.stream().forEach(mhs -> {
            System.out.println("" + mhs.toString());
        });
    }
    void ascendingSort() {
        this.mahasiswas.sort((Mahasiswa d1, Mahasiswa d2) ->
d1.nama.compareTo(d2.nama));
    }
    void descending() {
```

```
            this.mahasiswas.sort((Mahasiswa d1, Mahasiswa d2) ->
d2.nama.compareTo(d1.nama));
        }
        int linearSearch(String nim) {
            for (int i = 0; i < mahasiswas.size(); i++) {
                if (nim.equals(mahasiswas.get(i).nim)) {
                    return i;
                }
            }
            return -1;
        }
        public static void main(String[] args) {
            ListMahasiswa_No_3 lm = new ListMahasiswa_No_3();
            Mahasiswa m = new Mahasiswa("201234", "Noureen", "021xx1");
            Mahasiswa m1 = new Mahasiswa("201235", "Akhleena", "021xx2");
            Mahasiswa m2 = new Mahasiswa("201236", "Shannum", "021xx3");
            lm.tambah(m, m1, m2);
            lm.tampil();
            lm.update(lm.linearSearch("201235"), new Mahasiswa("201235", "Akhleena Lela",
"021xx2"));
            System.out.println("");
            lm.tampil();
            Mahasiswa key = new Mahasiswa("201235", null, null);
            lm.update(Collections.binarySearch(lm.mahasiswas, key, new MhsComparator()),
                  new Mahasiswa("201235", "Akhleema", "021xx2"));
            System.out.println("");
            lm.tampil();
            System.out.println("\n");
            System.out.println("Ascending : ");
            lm.ascendingSort();
            lm.tampil();
            System.out.println("\n");
            System.out.println("Descending : ");
            lm.descending();
            lm.tampil();
        }
    }
    class MhsComparator implements Comparator<Mahasiswa> {
        public int compare(Mahasiswa mhs1, Mahasiswa mhs2) {
            if (mhs1.nim == mhs2.nim) {
                return 0;
            } else {

                return -1;
            }
        }
        public Comparator<Mahasiswa> reversed() {
            return Comparator.super.reversed();
```

```
    }
}
```

- Output:



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

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

Mahasiswa{nim= 201234, nama=Noureen, notelp=021xx1}
Mahasiswa{nim= 201235, nama=Akhleema, notelp=021xx2}
Mahasiswa{nim= 201236, nama=Shannum, notelp=021xx3}


Ascending :
Mahasiswa{nim= 201235, nama=Akhleema, notelp=021xx2}
Mahasiswa{nim= 201234, nama=Noureen, notelp=021xx1}
Mahasiswa{nim= 201236, nama=Shannum, notelp=021xx3}


Descending :
Mahasiswa{nim= 201236, nama=Shannum, notelp=021xx3}
Mahasiswa{nim= 201234, nama=Noureen, notelp=021xx1}
Mahasiswa{nim= 201235, nama=Akhleema, notelp=021xx2}
```