JOBSHEET 16
Collection

16.1. Tujuan Praktikum

Setelah melakukan praktikum ini, mahasiswa mampu:

1. memahami bentuk-bentuk collection dan hierarkinya;
2. menerapkan collection sesuai dengan fungsi dan jenisnya;
3. menyelesaikan kasus menggunakan collection yang sesuai.

16.2. Kegiatan Praktikum 1

16.2.1. Percobaan 1

Pada percobaan 1 ini akan dicontohkan penggunaan collection untuk menambahkan sebuah elemen, mengakses elemen, dan menghapus sebuah elemen.

1. Buatlah sebuah class ContohList yang main methode berisi kode program seperti di bawah ini

```
25          List l = new ArrayList();
26          l.add(1);
27          l.add(2);
28          l.add(3);
29          l.add("Cireng");
30          System.out.printf("Elemen 0: %d total elemen: %d elemen terakhir: %s\n",
31                  l.get(0), l.size(), l.get(l.size() - 1));
32
33          l.add(4);
34          l.remove(0);
35          System.out.printf("Elemen 0: %d total elemen: %d elemen terakhir: %s\n",
36                  l.get(0), l.size(), l.get(l.size() - 1));
```

2. Tambahkan kode program untuk menggunakan collection dengan aturan penulisan kode program seperti berikut

```
38          List<String> names = new LinkedList<>();
39          names.add("Noureen");
40          names.add("Akhleema");
41          names.add("Shannum");
42          names.add("Uwais");
43          names.add("Al-Qarni");
44
45          System.out.printf("Elemen 0: %s total elemen: %s elemen terakhir: %s\n",
46                  names.get(0), names.size(), names.get(names.size() - 1));
47          names.set(0, "My kid");
48          System.out.printf("Elemen 0: %s total elemen: %s elemen terakhir: %s\n",
49                  names.get(0), names.size(), names.get(names.size() - 1));
50          System.out.println("Names: " + names.toString());
```

16.2.2. Verifikasi Hasil Percobaan

Verifikasi hasil kompilasi kode program Anda dengan gambar berikut ini.

```
run:
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]
```

16.2.3. Pertanyaan Percobaan

1. Perhatikan baris kode 25-36, mengapa semua jenis data bisa ditampung ke dalam sebuah Arraylist?
2. Modifikasi baris kode 25-36 seingga data yang ditampung hanya satu jenis atau spesifik tipe tertentu!
3. Ubah kode pada baris kode 38 menjadi seperti ini

```
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());
```

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

16.3. Kegiatan Praktikum 2

16.3.1. Tahapan Percobaan

Pada praktikum 2 ini akan dibuat beberapa method untuk menampilkan beberapa cara yang dapat dilakukan untuk mengambil/menampilkan elemen pada sebuah collection. Silakan ikutilah Langkah-langkah di bawah ini

1. Buatlah class dengan nama LoopCollection serta tambahkan method main yang isinya adalah sebagai berikut.

```
25          Stack<String> fruits = new Stack<>();
26          fruits.push("Banana");
27          fruits.add("Orange");
28          fruits.add("Watermelon");
29          fruits.add("Leci");
30          fruits.push("Salak");
31
33          for (String fruit : fruits) {
34              System.out.printf("%s ", fruit);
35          }
36
37          System.out.println("\n" + fruits.toString());
38
39          while (!fruits.empty()) {
40              System.out.printf("%s ", fruits.pop());
41          }
```

2. Tambahkan potongan kode berikut ini dari yang sebelumnya agar proses menampilkan elemen pada sebuah stack bervariasi.

```
43              fruits.push("Melon");
44              fruits.push("Durian");
45              System.out.println("");
                for (Iterator<String> it = fruits.iterator(); it.hasNext();) {
47                  String fruit = it.next();
48                  System.out.printf("%s ", fruit);
49              }
50              System.out.println("");
51              fruits.stream().forEach(e -> {
52                  System.out.printf("%s ", e);
53              });
54              System.out.println("");
55              for (int i = 0; i < fruits.size(); i++) {
56                  System.out.printf("%s ", fruits.get(i));
57
58              }
```

16.3.2. Verifikasi Hasil Percobaan

Verifikasi hasil kompilasi kode program Anda dengan gambar berikut ini.

```
Banana Orange Watermelon Leci Salak
[Banana, Orange, Watermelon, Leci, Salak]
Salak Leci Watermelon Orange Banana
Melon Durian
Melon Durian
Melon Durian BUILD SUCCESSFUL (total time: 0 seconds)
```

16.3.3. Pertanyaan Percobaan

1. Apakah perbedaan fungsi push() dan add() pada objek *fruits*?
2. Silakan hilangkan baris 43 dan 44, apakah yang akan terjadi? Mengapa bisa demikian?
3. Jelaskan fungsi dari baris 46-49?
4. Silakan ganti baris kode 25, *Stack<String>* menjadi *List<String>* dan apakah yang terjadi? Mengapa bisa demikian?
5. Ganti elemen terakhir dari dari objek fruits menjadi “Strawberry”!
6. Tambahkan 3 buah seperti “Mango”,”guava”, dan “avocado” kemudian dilakukan sorting!

16.4. Kegiatan Praktikum 3

16.4.1. Tahapan Percobaan

Pada praktikum 3 ini dilakukan uji coba untuk mengimplementasikan sebuah collection untuk menampung objek yang dibuat sesuai kebutuhan. Objek tersebut adalah sebuah objek mahasiswa dengan fungsi-fungsi umum seperti menambahkan, menghapus, mengubah, dan mencari.

1. Buatlah sebuah class Mahasiswa dengan attribute, kontruktor, dan fungsi sebagai berikut.

```
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;
}

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

2. Selanjutnya, buatlah sebuah class ListMahasiswa yang memiliki attribute seperti di bawah ini

```
List<Mahasiswa> mahasiswas = new ArrayList<>();
```

3. Method **tambah(), hapus(), update(), dan tampil()** secara berurut dibuat agar bisa melakukan operasi-operasi seperti yang telah disebutkan.

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

4. Untuk proses hapus, update membutuhkan fungsi pencarian terlebih dahulu yang potongan kode programnya adalah sebagai berikut

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

5. Pada class yang sama, tambahkan main method seperti potongan program berikut dan amati hasilnya!

```
ListMahasiswa lm = new ListMahasiswa();
Mahasiswa m = new Mahasiswa("201234", "Noureen", "021xx1");
Mahasiswa m1 = new Mahasiswa("201235", "Akhleema", "021xx2");
Mahasiswa m2 = new Mahasiswa("201236", "Shannum", "021xx3");
  menambahkan objek mahasiswa
lm.tambah(m, m1, m2);
  menampilkan list mahasiswa
lm.tampil();
  update mahasiswa
lm.update(lm.linearSearch("201235"), new Mahasiswa("201235", "Akhleema Lela", "021xx2"));
System.out.println("");
lm.tampil();
```

16.4.2. Verifikasi Hasil Percobaan

Verifikasi hasil kompilasi kode program Anda dengan gambar berikut ini.

```
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}
BUILD SUCCESSFUL (total time: 0 seconds)
```

16.4.3. Pertanyaan Percobaan

1. Pada fungsi tambah() yang menggunakan unlimited argument itu menggunakan konsep apa? Dan kelebihannya apa?
2. Pada fungsi linearSearch() di atas, silakan diganti dengan fungsi binarySearch() dari collection!
3. Tambahkan fungsi sorting baik secara ascending ataupun descending pada class tersebut!

16.5. Kegiatan Praktikum 4

16.5.1 Hashtable

A. hashtable secara procedural

```
public class Hashtable<K, V> {

    private Node<K, V>[] table;
    private int size;

    public Hashtable(int capacity) {
        // Initialize the table with empty nodes
        table = new Node[capacity];
        size = 0;
    }

    public int size() {
        return size;
    }

    public boolean isEmpty() {
        return size == 0;
    }
```

```
    public V put(K key, V value) {
        int hash = hash(key);
        int index = indexFor(hash, table.length);

        Node<K, V> node = table[index];
        while (node != null) {
            if (node.key.equals(key)) {
                // Update existing value
                V oldValue = node.value;
                node.value = value;
                return oldValue;
            }
            node = node.next;
        }

        // Add new entry
        table[index] = new Node<>(key, value, table[index]);
        size++;
        return null;
    }

    public V get(K key) {
        int hash = hash(key);
        int index = indexFor(hash, table.length);

        Node<K, V> node = table[index];
        while (node != null) {
            if (node.key.equals(key)) {
                return node.value;
            }
            node = node.next;
        }

        return null;
    }

    public V remove(K key) {
        int hash = hash(key);
        int index = indexFor(hash, table.length);

        Node<K, V> node = table[index];
        Node<K, V> prev = null;
        while (node != null) {
            if (node.key.equals(key)) {
                if (prev == null) {
                    table[index] = node.next;
                } else {
                    prev.next = node.next;
                }
                size--;
                return node.value;
            }
            prev = node;
            node = node.next;
        }

        return null;
    }

    private int hash(K key) {
        // Hash function chosen based on your needs
        int h = key.hashCode();
        return h ^ (h >>> 16);
    }

    private int indexFor(int hash, int length) {
        return hash % length;
    }
```

```
    private static class Node<K, V> {
        K key;
        V value;
        Node<K, V> next;

        public Node(K key, V value, Node<K, V> next) {
            this.key = key;
            this.value = value;
            this.next = next;
        }
    }
}
```

B. Hashtable dengan pendekatan OOP

```
import java.util.Map;
import java.util.HashMap;

public class Hashtable<K, V> {

    private final Map<K, V> map;

    public Hashtable() {
        map = new HashMap<>();
    }

    public int size() {
        return map.size();
    }

    public boolean isEmpty() {
        return map.isEmpty();
    }

    public V put(K key, V value) {
        return map.put(key, value);
    }

    public V get(K key) {
        return map.get(key);
    }

    public V remove(K key) {
        return map.remove(key);
    }

    public boolean containsKey(K key) {
        return map.containsKey(key);
    }

    // Additional methods specific to Hashtable

    public void putAll(Map<? extends K, ? extends V> m) {
        map.putAll(m);
    }

    public void clear() {
        map.clear();
    }

    public Set<K> keySet() {
```

```
        return map.keySet();
    }

    public Collection<V> values() {
        return map.values();
    }

    public Set<Map.Entry<K, V>> entrySet() {
        return map.entrySet();
    }

    // Implement any custom methods specific to your use case
}
```

16.5.2 Heap

Penerapan Heap pada Java collection Framework tidak dapat dilakukan secara langsung. Berikut contoh penerapan dalam bentuk kode program Java.

```
import java.util.PriorityQueue;
import java.util.Comparator;

public interface Heap {

    int getSize();

    boolean isEmpty();

    void insert(int element);

    int extractRoot();
}

public class MinHeap implements Heap {

    private final PriorityQueue<Integer> pq;

    public MinHeap() {
        this.pq = new PriorityQueue<>();
    }

    @Override
    public int getSize() {
        return pq.size();
    }
```

```
    @Override
    public boolean isEmpty() {
        return pq.isEmpty();
    }

    @Override
    public void insert(int element) {
        pq.add(element);
    }

    @Override
    public int extractRoot() {
        return pq.remove();
    }
}

public class MaxHeap implements Heap {

    private final PriorityQueue<Integer> pq;

    public MaxHeap() {
        this.pq = new PriorityQueue<>(Comparator.reverseOrder());
    }

    @Override
    public int getSize() {
        return pq.size();
    }

    @Override
    public boolean isEmpty() {
        return pq.isEmpty();
    }

    @Override
    public void insert(int element) {
        pq.add(element);
    }

    @Override
    public int extractRoot() {
```

```
        return pq.remove();
    }
}

public class Main {

    public static void main(String[] args) {
        MinHeap minHeap = new MinHeap();
        MaxHeap maxHeap = new MaxHeap();

        minHeap.insert(10);
        minHeap.insert(5);
        minHeap.insert(20);

        System.out.println("Ukuran heap-min: " + minHeap.getSize());
        System.out.println("Elemen minimum: " + minHeap.extractRoot());
        System.out.println("Elemen tersisa: " + minHeap.pq);

        maxHeap.insert(10);
        maxHeap.insert(5);
        maxHeap.insert(20);

        System.out.println("Ukuran heap-max: " + maxHeap.getSize());
        System.out.println("Elemen max: " + maxHeap.extractRoot());
        System.out.println("Elemen tersisa: " + maxHeap.pq);
    }
}
```

16.6. Tugas Praktikum

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.
   Ilustrasi Program
   Menu Awal dan Penambahan Data

```
  ************************************************
  SISTEM PENGOLAHAN DATA NILAI MAHASISWA SEMESTER
  ************************************************

  1. Input Nilai
  2. Tampil Nilai
  3. Mencari Nilai Mahasiswa
  4. Urut Data Nilai
  5. Keluar
  ************************************************
  Pilih     :
```

```
Pilih     : 1
Masukan data
Kode      : 0001
Nilai     : 80.75

DAFTAR MAHASISWA
*********************************************
NIM             Nama            Telf
20001           Thalhah         021xxx
20002           Zubair          021xxx
20003           Abdur-Rahman    021xxx
20004           Sa'ad           021xxx
20005           Sa'id           021xxx
20006           Ubaidah         021xxx
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
Pilih MK by kode: 00001
```

Tampil Nilai

```
**********************************************
SISTEM PENGOLAHAN DATA NILAI MAHASISWA SEMESTER
**********************************************

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                          SKS       Nilai
20001          Thalhah        Internet of Things                   3         80.75
```

Pencarian Data Mahasiswa

```
**********************************************
SISTEM PENGOLAHAN DATA NILAI MAHASISWA SEMESTER
**********************************************

1. Input Nilai
2. Tampil Nilai
3. Mencari Nilai Mahasiswa
4. Urut Data Nilai
5. Keluar
**********************************************
Pilih     : 3

DAFTAR NILAI MAHASISWA
**********************************************
Nim            Nama           Mata Kuliah                          SKS       Nilai
20001          Thalhah        Internet of Things                   3         90.00
20002          Zubair         Praktikum Algoritma dan Pemrograman  3         80.75
Masukkan data mahasiswa[nim] :20002
Nim            Nama           Mata Kuliah                          SKS       Nilai
20002          Zubair         Praktikum Algoritma dan Pemrograman  3         80.75
Total SKS 3 telah diambil.
```

Pengurutan Data Nilai

```
**********************************************
SISTEM PENGOLAHAN DATA NILAI MAHASISWA SEMESTER
**********************************************

1. Input Nilai
2. Tampil Nilai
3. Mencari Nilai Mahasiswa
4. Urut Data Nilai
5. Keluar
**********************************************
Pilih     : 4

DAFTAR NILAI MAHASISWA
**********************************************
Nim            Nama           Mata Kuliah                          SKS       Nilai
20002          Zubair         Praktikum Algoritma dan Pemrograman  3         80.75
20001          Thalhah        Internet of Things                   3         90.00
```

2. Tambahkan prosedur hapus data mahasiswa melalui implementasi Queue pada collections Tugas nomor 1!

--- *** ---