JOBSHEET XV

Collection

13.1 Tujuan Praktikum

Setelah melakukan praktikum ini, mahasiswa mampu:

1. memahami manfaat Java Collection Framework;
2. mengimplementasikan class dalam Java Collection Framework sesuai kasus;

13.2 Persiapan

Sebelum mulai mencoba class-class dalam Java Collection Framework, buatlah beberapa class yang akan digunakan dalam praktikum berikutnya:

1. Buat folder Praktikum14

2. Buat class Customer dalam Customer.java

```java
public class Customer {
    public int id;
    public String name;

    public Customer() {
    }

    public Customer(int id, String name) {
        this.id = id;
        this.name = name;
    }

    public String toString() {
        return "ID: " + this.id + " Nama: " + this.name;
    }
}
```

3. Buat class Book dalam Book.java

```java
public class Book {
    public String isbn;
    public String title;

    public Book() {
    }

    public Book(String isbn, String title) {
        this.isbn = isbn;
        this.title = title;
    }

    public String toString() {
        return "ISBN: " + this.isbn + " Title: " + this.title;
    }
}
```

13.3 Praktikum - Implementasi ArrayList

ArrayList merupakan sebuah class dari Java Collection Framework yang menyediakan implementasi struktur data serupa dengan array tetapi dengan ukuran dinamis.

1. Buat file DemoArrayList.java. Lakukan import java.util.ArrayList;

```java
import java.util.ArrayList;

public class DemoArrayList {
```

2. Pada fungsi main(), instansiasi collection baru dengan nama customers bertipe ArrayList of Customer dengan size 2. Selanjutnya, buat object customer1 dan customer2 kemudian tambahkan ke dalam ArrayList customers dengan method add.

```java
ArrayList<Customer> customers = new ArrayList<>(2);
Customer customer1 = new Customer(1, "Zakia");
Customer customer2 = new Customer(5, "Budi");

customers.add(customer1);
customers.add(customer2);
```

3. Gunakan looping dengan foreach untuk mencetak data customers

```java
for (Customer cust : customers) {
    System.out.println(cust.toString());
}
```

4. Cobalah tambahkan object customer baru ke dalam customers. Apakah object dapat ditambahkan meskipun melebihi kapasitas?

```
ArrayList<Customer> customers = new ArrayList<>(2);

Customer customer1 = new Customer(1, "Zakia");
Customer customer2 = new Customer(5, "Budi");

customers.add(customer1);
customers.add(customer2);

customers.add(new Customer(4, "Cica"));

for (Customer cust : customers) {
    System.out.println(cust.toString());
}
```

5. Compile dan run kode program, di mana object yang baru ditambahkan? Di awal, di tengah, atau di akhir collection?
6. Untuk menambahkan object baru pada index tertentu, lakukan sebagai berikut

```
ArrayList<Customer> customers = new ArrayList<>(2);

Customer customer1 = new Customer(1, "Zakia");
Customer customer2 = new Customer(5, "Budi");

customers.add(customer1);
customers.add(customer2);

customers.add(new Customer(4, "Cica"));

customers.add(2, new Customer(100, "Rosa"));

for (Customer cust : customers) {
    System.out.println(cust.toString());
}
```

7. Compile dan run kode program. Index pada ArrayList dimulai dari 0 atau 1?

8. Untuk mengetahui posisi dari suatu objek, gunakan method indexOf()

```
ArrayList<Customer> customers = new ArrayList<>(2);

Customer customer1 = new Customer(1, "Zakia");
Customer customer2 = new Customer(5, "Budi");

customers.add(customer1);
customers.add(customer2);

customers.add(new Customer(4, "Cica"));

customers.add(2, new Customer(100, "Rosa"));

System.out.println(customers.indexOf(customer2));

for (Customer cust : customers) {
    System.out.println(cust.toString());
}
```

9. Untuk mengembalikan object pada index tertentu, gunakan method get()

```
ArrayList<Customer> customers = new ArrayList<>(2);

Customer customer1 = new Customer(1, "Zakia");
Customer customer2 = new Customer(5, "Budi");

customers.add(customer1);
customers.add(customer2);

customers.add(new Customer(4, "Cica"));

customers.add(2, new Customer(100, "Rosa"));

System.out.println(customers.indexOf(customer2));

Customer customer = customers.get(1);
System.out.println(customer.name);
customer.name = "Budi Utomo";

for (Customer cust : customers) {
    System.out.println(cust.toString());
}
```

10. Cobalah hapus angka 2 saat instansiasi object customers. Apakah ArrayList dapat diinstansiasi tanpa harus menentukan size di awal?
11. Anda juga dapat menambahkan sekumpulan customer baru ke dalam ArrayList secara sekaligus. Misalnya terdapat ArrayList newCustomers. Tambahkan seluruh object customer sekaligus ke dalam customers.

```
ArrayList<Customer> newCustomers = new ArrayList<>();
newCustomers.add(new Customer(201, "Della"));
newCustomers.add(new Customer(202, "Victor"));
newCustomers.add(new Customer(203, "Sarah"));

customers.addAll(newCustomers);

for (Customer cust : customers) {
    System.out.println(cust.toString());
}
```

12. Karena sudah menyediakan method toString(), pengecekan data customers untuk proses debugging juga dapat dilakukan lebih sederhana dengan cara berikut

```
System.out.println(customers);
```

```java
public static void main(String[] args) {
    ArrayList<Customer> customers = new ArrayList<>(2);
    Customer customer1 = new Customer(1, "Zakia");
    Customer customer2 = new Customer(5, "Budi");

    customers.add(customer1);
    customers.add(customer2);

    System.out.println("Initial customers:");
    for (Customer cust : customers) {
        System.out.println(cust.toString());
    }
    System.out.println();

    customers.add(new Customer(4, "Cica"));
    System.out.println("Customers after adding 'Cica':");
    for (Customer cust : customers) {
        System.out.println(cust.toString());
    }
    System.out.println();

    customers.add(2, new Customer(100, "Rosa"));
    System.out.println("Customers after adding 'Rosa' at index 2:");
    for (Customer cust : customers) {
        System.out.println(cust.toString());
    }
    System.out.println();

    System.out.println("Index of customer2: " + customers.indexOf(customer2));
    System.out.println();

    Customer customer = customers.get(1);
    System.out.println("Customer at index 1: " + customer.name);
    customer.name = "Budi Utomo";
    System.out.println("Customers after changing name of customer at index 1:");
    for (Customer cust : customers) {
        System.out.println(cust.toString());
    }
    System.out.println();

    ArrayList<Customer> customersNoInitialSize = new ArrayList<>();
    customersNoInitialSize.add(new Customer(10, "NoSizeCustomer1"));
    System.out.println("Customers in ArrayList instantiated without initial size:");
    System.out.println(customersNoInitialSize);
    System.out.println();

    ArrayList<Customer> newCustomers = new ArrayList<>();
    newCustomers.add(new Customer(201, "Della"));
    newCustomers.add(new Customer(202, "Victor"));
    newCustomers.add(new Customer(203, "Sarah"));

    customers.addAll(newCustomers);
    System.out.println("Customers after adding newCustomers collection:");
    for (Customer cust : customers) {
        System.out.println(cust.toString());
    }
    System.out.println();

    System.out.println("Customers directly printed (for debugging): " + customers);
}
```

13.4 Praktikum - Implementasi Stack

Class Stack dalam Java Collection Framework yang menyediakan implementasi struktur

data tumpukan/stack.

1. Buat file StackDemo.java. Lakukan import java.util.Stack;
2. Pada fungsi main() buat beberapa object bertipe Book.
3. Instansiasi object books bertipe Stack of Book kemudian tambahkan object yang sudah dibuat ke dalamnya menggunakan method push()

```
Book book1 = new Book("1234", "Dasar Pemrograman");
Book book2 = new Book("7145", "Hafalah Shalat Delisa");
Book book3 = new Book("3562", "Muhammad Al-Fatih");

Stack<Book> books = new Stack<>();
books.push(book1);
books.push(book2);
books.push(book3);
```

4. Class Stack juga sudah memiliki method pop() dan peek() seperti yang Anda diimpelementasikan secara manual pada praktikum sebelumnya

```
Book temp = books.peek();

if (temp != null) {
    System.out.println(temp.toString());
}
```

```
Book temp2 = books.pop();

if (temp2 != null) {
    System.out.println(temp2.toString());
}
```

5. Mengapa perlu ada pengecekan (temp != null)?
6. Lakukan looping untuk mencetak data buku pada stack

```
for (Book book : books) {
    System.out.println(book.toString());
}
```

7. Jika diperlukan pada proses debugging, books juga dapat dicetak dengan cara berikut

```
System.out.println(books);
```

8. Bagaimana cara melakukan pencarian elemen pada stack menggunakan method search()?

```java
import java.util.Stack;

public class StackDemo {
    public static void main(String[] args) {
        Book book1 = new Book.Book(String isbn, String title)
        Book book2 = new Book("7145", "Hafalah Shalat Delisa");
        Book book3 = new Book("3562", "Muhammad Al-Fatih");

        Stack<Book> books = new Stack<>();
        books.push(book1);
        books.push(book2);
        books.push(book3);

        Book temp = books.peek();
        if (temp != null) {
            System.out.println("Peeked book: " + temp.toString());
        }

        Book temp2 = books.pop();
        if (temp2 != null) {
            System.out.println("Popped book: " + temp2.toString());
        }
        System.out.println();

        System.out.println("Books remaining in stack:");
        for (Book book : books) {
            System.out.println(book.toString());
        }
        System.out.println();

        System.out.println("Books directly printed (for debugging): " + books);
        System.out.println();

        System.out.println("Position of book1 in stack (from top, 1-based): " + books.search(book1));
        System.out.println("Position of book2 in stack (from top, 1-based): " + books.search(book2));
        System.out.println("Position of book3 in stack (from top, 1-based): " + books.search(book3));
        System.out.println();
    }
}
```

13.5 Praktikum - Implementasi TreeSet

Class TreeSet pada Java Collection Framework menyediakan implementasi binary search tree. Lakukan langkah percobaan berikut:

1. Buat file TreeSetDemo.java kemudian import java.util.TreeSet;
2. Tambahkan fungsi main() kemudian instansiasi object TreeSet of String. Tambahkan beberapa nilai bertipe String ke dalam TreeSet

```java
TreeSet<String> fruits = new TreeSet<>();

fruits.add("Mangga");
fruits.add("Apel");
fruits.add("Jeruk");
fruits.add("Jambu");

for (String temp : fruits) {
    System.out.println(temp);
}
```

3. Cetak data pada ts dengan looping

```
for (String temp : fruits) {
    System.out.println(temp);
}
```

4. Compile dan run program. Mengapa urutan yang ditampilkan berbeda dengan urutan penambahan data ke dalam TreeSet fruits?
5. Tambahkan kode program sebagai berikut:

```
System.out.println("First: " + fruits.first());
System.out.println("Last: " + fruits.last());

fruits.remove("Jeruk");
System.out.println("Setelah remove " + fruits);

fruits.pollFirst();
System.out.println("Setelah poll first " + fruits);

fruits.pollLast();
System.out.println("Setelah poll last " + fruits);
```

6. Apa yang dilakukan oleh method first(), last(), remove(), pollFirst(), dan pollLast()?

```
import java.util.Stack;

public class StackDemo {
    public static void main(String[] args) {
        Book book1 = new Book("1234", "Dasar Pemrograman");
        Book book2 = new Book("7145", "Hafalah Shalat Delisa");
        Book book3 = new Book("3562", "Muhammad Al-Fatih");

        Stack<Book> books = new Stack<>();
        books.push(book1);
        books.push(book2);
        books.push(book3);

        Book temp = books.peek();
        if (temp != null) {
            System.out.println("Peeked book: " + temp.toString());
        }

        Book temp2 = books.pop();
        if (temp2 != null) {
            System.out.println("Popped book: " + temp2.toString());
        }
        System.out.println();

        System.out.println("Books remaining in stack:");
        for (Book book : books) {
            System.out.println(book.toString());
        }
        System.out.println();

        System.out.println("Books directly printed (for debugging): " + books);
        System.out.println();

        System.out.println("Position of book1 in stack (from top, 1-based): " + books.search(book1));
        System.out.println("Position of book2 in stack (from top, 1-based): " + books.search(book2));
        System.out.println("Position of book3 in stack (from top, 1-based): " + books.search(book3));
        System.out.println();
    }
}
```

13.6 Praktikum – Sorting

Selain pencarian data menggunakan method get() atau search(), Java Collection Framework juga menyediakan fungsi untuk melakukan pengurutan data.

Untuk melakukan pengurutan data String, int, atau primitive data type lain, Anda dapat melakukan cara berikut:

```
ArrayList<String> daftarSiswa = new ArrayList<>();
daftarSiswa.add("Zainab");
daftarSiswa.add("Andi");
daftarSiswa.add("Rara");
Collections.sort(daftarSiswa);

System.out.println(daftarSiswa);
```

Khusus untuk collection of objects, pengurutan data harus menentukan berdasarkan atribut mana pengurutan dilakukan. Misalnya ingin dilakukan pengurutan data customer pada praktikum di atas berdasarkan atribut name. Gunakan kode program berikut (berikan nama variabel c1 dan c2 secara random. Anda bisa menggunakan nama variabel lainnya)

```
customers.sort((c1,c2)->c1.name.compareTo(c2.name));

System.out.println(customers);
```

Kedua cara di atas dapat digunakan untuk mengurutkan data pada jenis collection lain yang tidak melakukan pengurutan secara otomatis seperti TreeSet.

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.TreeSet;

public class SortingDemo {
    public static void main(String[] args) {
        ArrayList<String> daftarSiswa = new ArrayList<>();
        daftarSiswa.add("Zainab");
        daftarSiswa.add("Andi");
        daftarSiswa.add("Rara");
        Collections.sort(daftarSiswa);

        System.out.println("Sorted student list: " + daftarSiswa);
        System.out.println();

        ArrayList<Customer> customers = new ArrayList<>();
        customers.add(new Customer(1, "Zakia"));
        customers.add(new Customer(5, "Budi"));
        customers.add(new Customer(4, "Cica"));

        customers.sort((c1, c2) -> c1.name.compareTo(c2.name));
        System.out.println("Sorted customers by name: " + customers);
        System.out.println();

        TreeSet<Customer> sortedCustomersTreeSet = new TreeSet<>((c1, c2) -> c1.name.compareTo(c2.name));
        sortedCustomersTreeSet.add(new Customer(1, "Zakia"));
        sortedCustomersTreeSet.add(new Customer(5, "Budi"));
        sortedCustomersTreeSet.add(new Customer(4, "Cica"));
        System.out.println("Customers in TreeSet sorted by name: " + sortedCustomersTreeSet);
    }
}
```