

**Nama : Siti Mutmainah**

**NIM : 244107020143**

### **Persiapan**

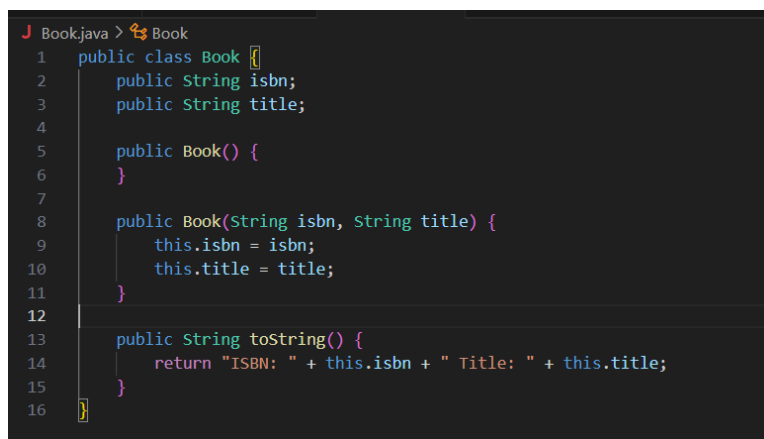
Sebelum mulai mencoba class-class dalam Java Collection Framework, buatlah beberapa class yang akan digunakan dalam praktikum berikutnya:

1. Buat folder Praktikum14
2. Buat class Customer dalam Customer.java



```
Customer.java > Customer > Customer(int, String)
1 public class Customer {
2     public int id;
3     public String name;
4
5     public Customer() {
6     }
7
8     public Customer(int id, String name) {
9         this.id = id;
10        this.name = name;
11    }
12
13    @Override
14    public String toString() {
15        return "ID: " + this.id + " Name: " + this.name;
16    }
17 }
```

3. Buat class Book dalam Book.java



```
Book.java > Book
1 public class Book {
2     public String isbn;
3     public String title;
4
5     public Book() {
6     }
7
8     public Book(String isbn, String title) {
9         this.isbn = isbn;
10        this.title = title;
11    }
12
13    public String toString() {
14        return "ISBN: " + this.isbn + " Title: " + this.title;
15    }
16 }
```

## Praktikum - Implementasi ArrayList

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

4. Buat file DemoArrayList.java. Lakukan import java.util.ArrayList;
5. 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.

```
DemoArray.java > DemoArray > main(String[])
1  import java.util.ArrayList;
2
3  public class DemoArray {
4      public static void main(String[] args) {
5
6          ArrayList<Customer> customers = new ArrayList<>(initialCapacity:2)
7
8          Customer customer1 = new Customer(id:1, name:"Zakia");
9          Customer customer2 = new Customer(id:5, name:"Budi");
10
11          customers.add(customer1);
12          customers.add(customer2);
13      }
```

6. Gunakan looping dengan foreach untuk mencetak data customers

```
3
4      for (Customer cust : customers) {
5          System.out.println(cust.toString());
6      }
7
8  }
```

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

Jawab: Meskipun ArrayList diinisialisasi dengan kapasitas awal 2, object tetap bisa ditambahkan jika melebihi kapasitas awal.

```
customers.add(new Customer(id:4, name:"Cica"));
```

8. Compile dan run kode program, di mana object yang baru ditambahkan? Di awal, di tengah, atau di akhir collection?

Jawab: method add() tanpa index akan menambahkan object di akhir collection

9. Untuk menambahkan object baru pada index tertentu, lakukan sebagai berikut

```
customers.add(index:2, new Customer(id:100, name:"Rosa"));
```

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

Jawab: index pada ArrayList dimulai dari 0.

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

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

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

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

13. Cobalah hapus angka 2 saat instansiasi object `customers`. Apakah `ArrayList` dapat diinstansiasi tanpa harus menentukan size di awal?

Jawab: Iya, `ArrayList` dapat diinstansiasi tanpa menentukan size/ukuran awal.

14. 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(id:201, name:"Della"));  
newCustomers.add(new Customer(id:202, name:"Victor"));  
newCustomers.add(new Customer(id:203, name:"Sarah"));  
  
customers.addAll(newCustomers);  
  
for (Customer cust : customers) {  
    System.out.println(cust.toString());  
}
```

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

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

## Praktikum - Implementasi Stack

Class Stack dalam Java Collection Framework yang menyediakan implementasi struktur data tumpukan/stack.

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

```
StackDemo.java > StackDemo > main(String[])
import java.util.Stack;

public class StackDemo {
    Run | Debug
    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);
    }
}
```

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

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

if (temp != null) {
    System.out.println("Peek: " + temp.toString());
}

Book temp2 = books.pop();

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

20. Mengapa perlu ada pengecekan (temp != null)?

Jawab: Untuk pencegahan NullPointerException meskipun Stack bawaan Java melemparkan EmptyStackException ketika kosong, pengecekan null untuk menghindari crash program jika ada kesalahan implementasi, kompatibilitas dengan kode yang mungkin mengizinkan elemen null.

21. Lakukan looping untuk mencetak data buku pada stack

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

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

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

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

Jawab: search() mencari elemen dalam Stack dan mengembalikan posisi elemen (dimulai dari 1 untuk elemen teratas). Jika elemen tidak ditemukan, mengembalikan -1.

Pencarian dilakukan dari atas (top) ke bawah (bottom) stack

### Praktikum - Implementasi TreeSet

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

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

```
import java.util.TreeSet;

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

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

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

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

26. Cetak data pada ts dengan looping

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

27. Compile dan run program. Mengapa urutan yang ditampilkan berbeda dengan urutan penambahan data ke dalam TreeSet fruits?

Jawab: TreeSet menyimpan elemen secara terurut berdasarkan untuk string diurutkan (A-Z). Secara internal sebagai binary search tree yang selalu menjaga elemen dalam keadaan terurut.

28. Tambahkan kode program sebagai berikut:

```
}
System.out.println("\nFirst: " + fruits.first());
System.out.println("Last: " + fruits.last());

fruits.remove(o:"Jeruk");
System.out.println("\nSetelah remove Jeruk: " + fruits);

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

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

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

Jawab:

First() mengembalikan elemen terkecil

Last() mengembalikan elemen terbesar

Remove() menghapus elemen spesifik dari TreeSet

pollFirst() menghapus dan mengembalikan elemen pertama

pollLast() menghapus dan mengembalikan elemen terakhir

## 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:

```
J SortingDemo.java > SortingDemo
1  import java.util.ArrayList;
2  import java.util.Collections;
3
4  public class SortingDemo {
5      public static void main(String[] args) {
6
7          ArrayList<String> daftarSiswa = new ArrayList<>();
8          daftarSiswa.add(e: "Zainab");
9          daftarSiswa.add(e: "Andi");
10         daftarSiswa.add(e: "Rara");
11
12         Collections.sort(daftarSiswa);
13
14         System.out.println(daftarSiswa);
15     }
16 }
```

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)

```
J CustomerSortingDemo.java > ...
1  import java.util.ArrayList;
2
3  public class CustomerSortingDemo {
4      public static void main(String[] args) {
5
6          ArrayList<Customer> customers = new ArrayList<>();
7
8          customers.add(new Customer(id:3, name:"Zainab"));
9          customers.add(new Customer(id:1, name:"Andi"));
10         customers.add(new Customer(id:2, name:"Rara"));
11
12         customers.sort((c1, c2) -> c1.name.compareTo(c2.name));
13
14         System.out.println(customers);
15     }
16 }
17 |
```

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