

**LAPORAN PRAKTIKUM ALGORITMA DAN STRUKTUR
COLLECTION**



ABIM MUSTAWA

244107020078

KELAS TI-1B

**PRODI D-IV TEKNIK
INFORMATIKA JURUSAN
TEKNOLOGI INFORMASI
POLITEKNIK NEGERI
MALANG 2025**

1. Percobaan Praktikum

1.1 Persiapan

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

1. Buat folder Praktikum15
2. Buat class Customer dalam Customer.java

```
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     public String toString() {  
14         return " ID: " + this.id + " Name: " + this.name;  
15     }  
16 }
```

3. Buat class Book dalam Book.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;  
    }  
}
```

1.2 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; 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.

```
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;
    }
}
```

3. Gunakan looping dengan foreach untuk mencetak data customers

```
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?

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

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

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

7. Compile dan run kode program. Index pada ArrayList dimulai dari 0 atau 1?
8. Untuk mengetahui posisi dari suatu objek, gunakan method indexOf()

```
System.out.println("\nIndex dari Budi: " + customers.indexOf(customer2));
```

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

```
Customer customer = customers.get(index:1);
System.out.println("\nCustomer di index 1 sebelum diubah: " + customer.name);
customer.name = "Budi Utomo";
```

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(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());
}
```

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

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

1.3 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(isbn:"1234", title:"Dasar Pemrograman");
Book book2 = new Book(isbn:"7145", title:"Hafalah Shalat Delisa");
Book book3 = new Book(isbn:"3562", title:"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 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());
}
```

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

Jawab : Untuk menghindari NullPointerException ketika stack kosong.

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("\nPosisi buku 'Dasar Pemrograman': " +  
    books.search(book1));
```

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

Jawab : Menggunakan Method search().

```
Stack<Book> books = new Stack<>();
```

```
books.push(book1);
```

```
int posisi = books.search(book1);
```

1.4 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

```
TreeSet<String> fruits = new TreeSet<>();  
  
fruits.add(e:"Mangga");  
fruits.add(e:"Apel");  
fruits.add(e:"Jeruk");  
fruits.add(e:"Jambu");  
  
System.out.println(x:"Daftar buah (terurut):");  
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?

Jawab : Karena TreeSet menyimpan data secara terurut berdasarkan nilai, bukan urutan insert.

5. 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 poll first: " + fruits);

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

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

Jawab :

- first(): Mengambil elemen pertama (terkecil)
- last(): Mengambil elemen terakhir (terbesar)
- remove(): Menghapus elemen tertentu
- pollFirst(): Menghapus dan mengembalikan elemen pertama
- pollLast(): Menghapus dan mengembalikan elemen terakhir

1.5 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(e:"Zainab");
daftarSiswa.add(e:"Andi");
daftarSiswa.add(e:"Rara");

Collections.sort(daftarSiswa);
System.out.println("Daftar siswa terurut: " + 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)

```
ArrayList<Object> ArrayList(Collection):  
ArrayList<Object> ArrayList(Collection, Comparator):  
Collections.sort((C1, C2) -> C1.compareTo(C2)):
```

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