LAPORAN PRATIKUM STRUKTUR DATA

IMPLEMENTASI STRUKTUR DATA LIST, QUEUE, DAN STACK DALAM BAHASA JAVA

MATA KULIAH STRUKTUR DATA

DOSEN PENGAMPU : Dr. Wahyudi, S.T, M.T

OLEH:

AUFAN TAUFIQURRAHMAN

NIM 2411532011

FAKULTAS TEKNOLOGI INFORMASI

DEPARTEMEN INFORMATIKA

UNIVERSITAS ANDALAS

2025

A. Pendahuluan

Struktur data merupakan inti dari pemrograman komputer modern. Praktikum ini bertujuan untuk memperkenalkan serta mengimplementasikan berbagai jenis struktur data linier seperti ArrayList, Queue, dan Stack dalam bahasa pemrograman Java. Melalui praktik langsung, mahasiswa dapat memahami cara kerja dan perbedaan antar struktur data dalam pengelolaan koleksi data.

B. Tujuan Praktikum

- Mengimplementasikan struktur data ArrayList untuk penyimpanan data dinamis.
- Menggunakan Queue untuk menerapkan konsep antrian (FIFO).
- Menggunakan Stack untuk menerapkan konsep tumpukan (LIFO).
- Memahami penerapan praktis dari masing-masing struktur data dalam konteks program nyata.

C. Prosedur praktikum

1. Class ArrayList1

   Program ini menunjukkan bagaimana menambahkan, menghapus, dan mencetak elemen dalam ArrayList.

   - Buat class ArrayList1 dan inisialisasi objek ArrayList:

```
public class ArrayList1 {
    Run main | Debug main | Run | Debug
    public static void main(String[] args) {
        ArrayList<String> list = new ArrayList<>();
```

   - Tambahkan elemen ke dalam ArrayList menggunakan method add():

```
public class ArrayList1 {
    Run main | Debug main | Run | Debug
    public static void main(String[] args) {
        ArrayList<String> list = new ArrayList<>();
        list.add(e:"Apple");
        list.add(e:"Banana");
```

- Sisipkan elemen dengan memasukkan parameter indeks, lalu elemen

```
public class ArrayList1 {
    Run main | Debug main | Run | Debug
    public static void main(String[] args) {
        ArrayList<String> list = new ArrayList<>();
        list.add(e:"Apple");
        list.add(e:"Banana");
        list.add(index:1, element:"Cheery");
```

- Memasukkan semua elemen list kedalam variabel fruit dan mencetaknya dengan for each

```
public class ArrayList1 {
    Run main | Debug main | Run | Debug
    public static void main(String[] args) {
        ArrayList<String> list = new ArrayList<>();
        list.add(e:"Apple");
        list.add(e:"Banana");
        list.add(index:1, element:"Cheery");
        for (String fruit : list) {
            System.out.println(fruit);
        }
    }
}
```

2. Class Perpustakaan

   Program ini merupakan simulasi sistem perpustakaan yang menggunakan ArrayList untuk menyimpan buku, Queue untuk antrian peminjaman, dan Stack untuk histori pengembalian.

   - Deklarasi class Buku lalu buat contructor dari class tersebut agar dapat menginisialisasi objek buku

```
class Buku {
    String judul, pengarang, isbn;
    Buku(String judul, String pengarang, String isbn) {
        this.judul = judul;
        this.pengarang = pengarang;
        this.isbn = isbn;
    }
}
```

   - Inisilisasi struktur data pada class perpustakaan

```
class Perpustakaan {
    LinkedList<Buku> koleksiBuku = new LinkedList<>();
    Queue<Buku> Peminjaman = new LinkedList<>();
    Stack<Buku> Pengembalian = new Stack<>();
```

- Tambah buku ke LinkedList:

```
void tambahBuku(String judul, String pengarang, String isbn) {
    koleksiBuku.add(new Buku(judul, pengarang, isbn));
}
```

- Tambah peminjam ke Queue:

```
void pinjamBuku (String judul) {
    for (Buku buku : koleksiBuku) {
        if (buku.judul.equals(judul)) {
            Peminjaman.add(buku);
            break;
        }
    }
}
```

- Simpan nama pengembali buku ke dalam Stack:

```
    void kembalikanBuku(String judul) {
        for (Buku buku : Peminjaman){
            if (buku.judul.equals(judul)) {
                Pengembalian.push(buku);
                break;
            }
        }
    }
```

3. Kesimpulan

   Praktikum ini memperlihatkan pentingnya pemilihan struktur data yang tepat dalam menyelesaikan persoalan yang berbeda. ArrayList sangat berguna untuk penyimpanan data dinamis secara berurutan. Queue efektif digunakan untuk antrian layanan (FIFO), dan Stack sangat cocok untuk melacak histori atau kejadian terakhir (LIFO). Dengan memahami sintaks dasar dan perilaku masing-masing struktur data, mahasiswa dapat mengembangkan program yang lebih efisien dan sesuai kebutuhan.