Laporan

Praktikum Mandiri 5

Identitas Mahasiswa

- **Nama**: Aliyah Zahratu Rizqi
- **NIM**: A11.2023.15294
- **Kelas**: A11.4403
- **Link github** : Repository PBO

A. Praktikum Mandiri

1. latihan1

a. Array1

- Membaca input jumlah data dimana meminta user untuk memasukkan jumlah data yang akan disimpan array

```
// Jumlah data yang akan dimasukkan
System.out.print("Masukkan jumlah data: ");
int jumlah = input.nextInt();
```

- Program mengulangi proses meminta input sebanyak **jumlah** kali dan menyimpannya di array **nilai**

```
// Input data
for (int i = 0; i < jumlah; i++) {
    System.out.print("Masukkan data ke-" + (i + 1) + ": ");
    nilai[i] = input.nextInt();
}
```

- Menampilkan Data yang telah dimasukkandan ditampilkan Kembali sesuai urutan inputan

```
// Menampilkan data yang telah dimasukkan
System.out.println("\nData yang dimasukkan:");
for (int i = 0; i < jumlah; i++) {
    System.out.println("Data ke - " + (i + 1) + " = " + nilai[i]);
}
```

- Kemudian data ditampilkan lagi, tetapi dengan indeks array untuk menunjukkan bagaimana data disimpan dalam memori

```
// Menampilkan hasil nilai sesuai indeks
System.out.println("\nHasil nilai:");
for (int i = 0; i < jumlah; i++) {
    System.out.println("Hasil nilai[" + i + "] = " + nilai[i]);
}
```

b. ArrayObjek

- Method tampilkanNilai(int index) menampilkan nilai dari objek Nilai beserta indeksnya

```
// Method untuk menampilkan nilai
public void tampilkanNilai(int index) {
    System.out.println("Hasil nilai[" + index + "] = " + angka);
}
```

- Memasukkan data ke dalam array dengan meggunakan perulangan dan setiap angka disimpan sebagai objek **Nilai** dalam array

```
// Input data
for (int i = 0; i < jumlah; i++) {
    System.out.print("Masukkan data ke-" + (i + 1) + ": ");
    int angka = input.nextInt();
    daftarNilai[i] = new Nilai(angka);
}
```

- Perulangan untuk menampilkan semua data dalam array dengan method tampilkanNilai().

```
// Menampilkan hasil dengan method daftarNilai()
System.out.println("\nHasil nilai:");
for (int i = 0; i < jumlah; i++) {
    daftarNilai[i].tampilkanNilai(i);
}
```

Hasil:

```
A:\SMT 4\PBO\week5\tugas praktikum>java latihan1.Array1
Masukkan jumlah data: 5
Masukkan data ke-1: 68
Masukkan data ke-2: 7
Masukkan data ke-3: 28
Masukkan data ke-4: 90
Masukkan data ke-5: 32

Data yang dimasukkan:
Data ke - 1 = 68
Data ke - 2 = 7
Data ke - 3 = 28
Data ke - 4 = 90
Data ke - 5 = 32

Hasil nilai:
Hasil nilai[0] = 68
Hasil nilai[1] = 7
Hasil nilai[2] = 28
Hasil nilai[3] = 90
Hasil nilai[4] = 32
```

2. latihan2

a. NilaiMahasiswa

- Menambahkan nilai ke dalam **ArrayList** dengan method **add()**

```
public void tambahNilai(int nilai) {
    daftarNilai.add(nilai);
}
```

- Menampilkan daftar nilai mahasiswa menggunakan loop for kemudian **daftarNilai.get(i)** digunakan untuk mengambil nilai dari **ArrayList**

```
public void tampilkanDaftarNilai() {
    System.out.println("Daftar Nilai Mahasiswa:");
    for (int i = 0; i < daftarNilai.size(); i++) {
        System.out.println("Nilai ke-" + (i + 1) + ": " + daftarNilai.get(i));
    }
}
```

- Membuat objek **NilaiMahasiswa (nm)** untuk menyimpan nilai.

```
Run main | Debug main
public static void main(String[] args) {
    NilaiMahasiswa nm = new NilaiMahasiswa();
    Scanner scanner = new Scanner(System.in);
```

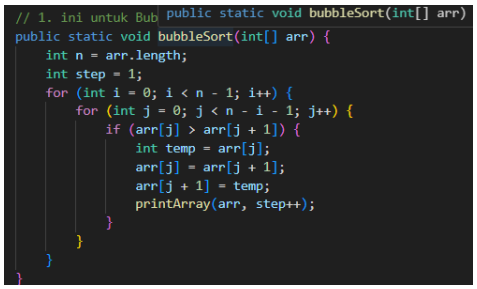
Hasil :

```
A:\SMT 4\PBO\week5\tugas praktikum>java latihan2.NilaiMahasiswa
Masukkan nilai (atau ketik 0 untuk berhenti): 80
Masukkan nilai (atau ketik 0 untuk berhenti): 90
Masukkan nilai (atau ketik 0 untuk berhenti): 100
Masukkan nilai (atau ketik 0 untuk berhenti): 75
Masukkan nilai (atau ketik 0 untuk berhenti): 0
Daftar Nilai Mahasiswa:
Nilai ke-1: 80
Nilai ke-2: 90
Nilai ke-3: 100
Nilai ke-4: 75
```

3. latihan3

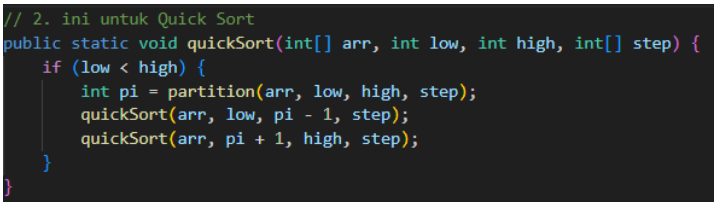
a. SortingAlgorithm

- **Bubble Sort** untuk membandingkan elemen bersebelahan dan menukar jika tidak berurutan atau elemen terbesar dipindahkan ke akhir array secara bertahap



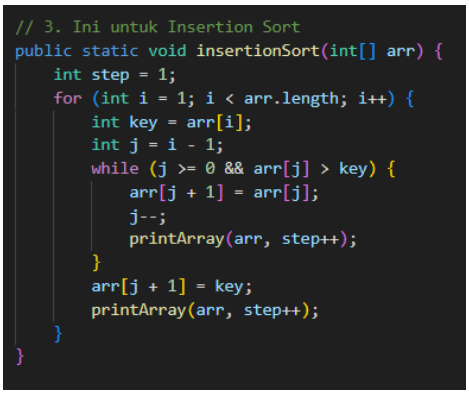
```
// 1. ini untuk Bub public static void bubbleSort(int[] arr)
public static void bubbleSort(int[] arr) {
    int n = arr.length;
    int step = 1;
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
                printArray(arr, step++);
            }
        }
    }
}
```

- **Quick Sort** yaitu untuk membagi array menjadi 2 bagian berdasarkn pivot dan sorting di lakukan secara rekursif untuk setiap bagian



```
// 2. ini untuk Quick Sort
public static void quickSort(int[] arr, int low, int high, int[] step) {
    if (low < high) {
        int pi = partition(arr, low, high, step);
        quickSort(arr, low, pi - 1, step);
        quickSort(arr, pi + 1, high, step);
    }
}
```

- **Insertion Sort** berfungsi untuk memisahkan array menjadi 2 bagian yang sudah diurutkan dan yang belum dan juga memasukkan elemen satu per sat uke bagian yang sudah diurutkan



```
// 3. Ini untuk Insertion Sort
public static void insertionSort(int[] arr) {
    int step = 1;
    for (int i = 1; i < arr.length; i++) {
        int key = arr[i];
        int j = i - 1;
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j--;
            printArray(arr, step++);
        }
        arr[j + 1] = key;
        printArray(arr, step++);
    }
}
```

- **Selection Sort** berfungsi untuk mencari elemen terkecil dari array dan menukar elemen terkecil dengan posisi awal yang belum diurutkan

```java
// 4. ini Untuk Selection Sort
public static void selectionSort(int[] arr) {
    int step = 1;
    for (int i = 0; i < arr.length - 1; i++) {
        int minIndex = i;
        for (int j = i + 1; j < arr.length; j++) {
            if (arr[j] < arr[minIndex]) {
                minIndex = j;
            }
        }
        int temp = arr[minIndex];
        arr[minIndex] = arr[i];
        arr[i] = temp;
        printArray(arr, step++);
    }
}
```

- **Merge Sort** membagi array menjadi 2 bagian rekursif dan menggabungkan Kembali dengan cara yang sudah di urutkan

```java
// 5. Ini untuk Merge Sort
public static void mergeSort(int[] arr, int left, int right, int[] step) {
    if (left < right) {
        int mid = left + (right - left) / 2;
        mergeSort(arr, left, mid, step);
        mergeSort(arr, mid + 1, right, step);
        merge(arr, left, mid, right, step);
    }
}
```

Hasil :

```
A:\SMT 4\PBO\week5\tugas praktikum>java latihan3.SortingAlgorithms
Bubble Sort:
Iterasi 1: [1, 7, 3, 5, 2]
Iterasi 2: [1, 3, 7, 5, 2]
Iterasi 3: [1, 3, 5, 7, 2]
Iterasi 4: [1, 3, 5, 2, 7]
Iterasi 5: [1, 3, 2, 5, 7]
Iterasi 6: [1, 2, 3, 5, 7]

Quick Sort:
Iterasi 1: [1, 7, 3, 5, 2]
Iterasi 2: [1, 2, 3, 5, 7]
Iterasi 3: [1, 2, 3, 5, 7]
Iterasi 4: [1, 2, 3, 5, 7]
Iterasi 5: [1, 2, 3, 5, 7]
Iterasi 6: [1, 2, 3, 5, 7]
Iterasi 7: [1, 2, 3, 5, 7]

Insertion Sort:
Iterasi 1: [7, 7, 3, 5, 2]
```

4. latihan4

a. Array & MainArray

- **Class untuk operasi matematika** method **hitung()** untuk melakukan operasi matematika pada setiap indeks

```
public void hitung() {
    for (int i = 0; i < nilai1.size(); i++) {
        jumlah.add(nilai1.get(i) + nilai2.get(i));
        pengurangan.add(nilai1.get(i) - nilai2.get(i));
        perkalian.add(nilai1.get(i) * nilai2.get(i));
        if (nilai2.get(i) != 0) {
            pembagian.add(nilai1.get(i) / nilai2.get(i));
        } else {
            pembagian.add(0); // Atau bisa ditangani dengan cara lain
        }
    }
}
```

- **Input data dan eksekusi perhitungan** kemudian memasukkan data ke **ArrayList** dan menghubungkannya ke objek **Array**

```
// Input jumlah data yang diinginkan
System.out.print("Masukkan jumlah data: ");
int jumlahData = scanner.nextInt();

// Buat objek untuk mengelola operasi ArrayList
Array arrayOps = new Array();  // Pastikan ini ada
ArrayList<Integer> nilai1 = new ArrayList<>();
ArrayList<Integer> nilai2 = new ArrayList<>();

// Input nilai untuk nilai1
System.out.println("Nilai 1:");
for (int i = 0; i < jumlahData; i++) {
    System.out.print("Index ke-" + i + " = ");
    nilai1.add(scanner.nextInt());
}

// Input nilai untuk nilai2
System.out.println("Nilai 2:");
for (int i = 0; i < jumlahData; i++) {
    System.out.print("Index ke-" + i + " = ");
    nilai2.add(scanner.nextInt());
}
```

- **Memanggil metode** untuk perhitungan dan menampilkan hasil

```
// Set nilai ke objek operasi
arrayOps.setNilai1(nilai1);
arrayOps.setNilai2(nilai2);

// Hitung semua operasi
arrayOps.hitung();

// Tampilkan hasil
arrayOps.tampilkanHasil();
```

Hasil :

```
A:\SMT 4\PBO\week5\tugas praktikum>java latihan4.MainArray
Masukkan jumlah data: 2
Nilai 1:
Index ke-0 = 1
Index ke-1 = 2
Nilai 2:
Index ke-0 = 2
Index ke-1 = 3
Hasil Jumlah index ke-0 = 3
Hasil Pengurangan index ke-0 = -1
Hasil Perkalian index ke-0 = 2
Hasil Pembagian index ke-0 = 0
Hasil Jumlah index ke-1 = 5
Hasil Pengurangan index ke-1 = -1
Hasil Perkalian index ke-1 = 6
Hasil Pembagian index ke-1 = 0
```

b. MainQueue

- **Membuat Queue** menggunakan LinkedList untuk antrian

```
Queue<Integer> queue = new LinkedList<>();
int choice;
```

- **Menambahkan Elemen (insert)**

```
System.out.print("Enter value to insert: ");
int value = scanner.nextInt();
queue.add(value);
break;
```

- **Menghapus Elemen (remove)**

```
if (!queue.isEmpty()) {
    System.out.println("Removed: " + queue.poll());
} else {
```

- **Melihat Elemen Depan (peek)**

```
if (!queue.isEmpty()) {
    System.out.println("Front: " + queue.peek());
} else {
    System.out.println("Queue is empty!");
}
```

- **Cek Kosong (check empty)**

```
ystem.out.println("Queue empty: " + queue.isEmpty(
reak;
```

- **Menampilkan Ukuran (size)**

```
System.out.println("Size = " + queue.size());
System.out.println("Queue = " + queue);
break;
```

Hasil :

```
A:\SMT 4\PBO\week5\tugas praktikum>java latihan4.MainQueue

Queue Operations
1. insert
2. remove
3. peek
4. check empty
5. check full
6. size
Your Choice ? 1
Enter value to insert: 5
Do you want to continue (Type y or n): n
```

c. Matriks

- **Deklarasi Matrix**

```
public class Matriks {
    private int[][] matriksA;
    private int[][] matriksB;
    private int[][] hasilPenjumlahan;
    private int[][] hasilPerkalian;
```

- **Penjumlahan Matriks**

```
// Method untuk menjumlahkan matriks
private void jumlahMatriks() {
    for (int i = 0; i < matriksA.length; i++) {
        for (int j = 0; j < matriksA[i].length; j++) {
            hasilPenjumlahan[i][j] = matriksA[i][j] + matriksB[i][j];
        }
    }
}
```

- **Mengalikan Matriks**

```java
// Method untuk mengalikan matriks
private void kaliMatriks() {
    for (int i = 0; i < matriksA.length; i++) {
        for (int j = 0; j < matriksB[0].length; j++) {
            hasilPerkalian[i][j] = 0; // Inisialisasi hasil
            for (int k = 0; k < matriksA[0].length; k++) {
                hasilPerkalian[i][j] += matriksA[i][k] * matriksB[k][j];
            }
    latihan4
    public class Matriks
}
```

Hasil :

```
A:\SMT 4\PBO\week5\tugas praktikum>java latihan4.Matriks
Input baris matriks A= 2
Input kolom matriks A= 2
Input elemen A:
input A [0][0] = 10
input A [0][1] = 17
input A [1][0] = 18
input A [1][1] = 20
Input baris matriks B= 2
Input kolom matriks B= 2
Input elemen B:
input B [0][0] = 6
input B [0][1] = 7
input B [1][0] = 8
input B [1][1] = 9
Hasil penjumlahan matriks C:
16 24
26 29
Hasil perkalian matriks A dengan matriks B:
196 223
268 306
```

Kesimpulan

Saya telah menyelesaikan Praktikum Mandiri pertemuan 4 untuk mata kuliah PBO, di mana semua program berhasil dijalankan dengan output yang sesuai. Pada praktikum ini, saya mempelajari operasi ArrayList untuk penjumlahan, pengurangan, perkalian, dan pembagian dua ArrayList integer, implementasi Queue menggunakan LinkedList untuk operasi insert, remove, peek, dan cek ukuran, serta operasi matriks untuk penjumlahan dan perkalian dengan pengecekan ukuran terlebih dahulu. Pengalaman ini membantu saya memahami struktur data dan logika pemrograman Java lebih dalam.