

Nama: Harry LBI

NIM: A11.2023.15332

LATIHAN 1

```
1  package pratikum5.Latihan1;
2
3  import java.util.Scanner;
4
5  public class Array{
6      private int[] nilai;
7
8      public Array(int jumlahData){
9          this.nilai = new int[jumlahData];
10     }
11 }
```

Konstruktor ini dipanggil saat objek Array dibuat.

jumlahData menentukan ukuran array.

this.nilai = new int[jumlahData]; → Menginisialisasi array dengan ukuran sesuai parameter jumlahData.

```
12     public void isiarray(){
13         Scanner input = new Scanner(System.in);
14         System.out.println("this is nilai array"+ nilai.length);
15         for (int i = 0; i < nilai.length; i++) {
16             System.out.print("data ke- " + (i+1) + " : ");
17             this.nilai[i] = input.nextInt();
18         }
19         input.close();
20
21     }
22
23     public void menampilkanIsiArray(){
24         System.out.println();
25         for (int i = 0; i < nilai.length; i++) {
26             System.out.print("Hasil nilai["+i+"]" + this.nilai[i]);
27         }
28     }
29 }
```

objek Scanner untuk menangani input dari pengguna.

Menampilkan panjang **array** (**nilai.length**).

Mengisi array dengan nilai dari pengguna menggunakan perulangan for:

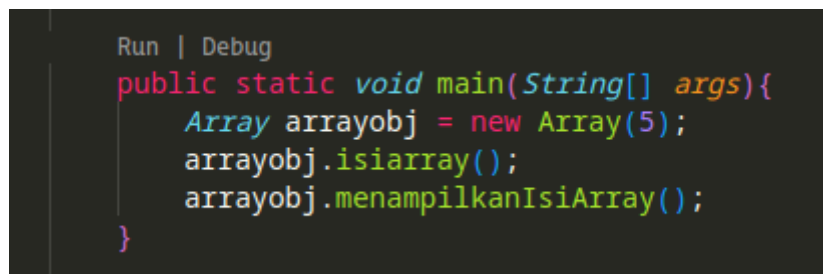
Loop berjalan dari $i = 0$ hingga $i < \text{nilai.length}$.

this.nilai[i] = input.nextInt(); → Menyimpan nilai input ke dalam array.

Menutup **Scanner** (**input.close();**) untuk menghindari kebocoran sumber daya.

Mencetak isi array menggunakan perulangan for.

Setiap elemen array ditampilkan dengan format **Hasil nilai[i]**.

A screenshot of a code editor with a dark background. At the top, there are two tabs labeled 'Run' and 'Debug'. Below the tabs, there is a Java code snippet. The code starts with 'public static void main(String[] args){', followed by 'Array arrayobj = new Array(5);', 'arrayobj.isiarray();', 'arrayobj.menampilkanIsiArray();', and ends with '}'.

```
Run | Debug
public static void main(String[] args){
    Array arrayobj = new Array(5);
    arrayobj.isiarray();
    arrayobj.menampilkanIsiArray();
}
```

Membuat **objek arrayobj** dengan ukuran array 5.

Memanggil **isiarray()** → Meminta input dari user.

Memanggil **menampilkanIsiArray()** → Menampilkan hasil array yang telah diinput.

Latihan 2

```
6 public class HitungNilai {
7     static ArrayList<Mahasiswa> daftarMahasiswa = new ArrayList<>();
8     static Scanner scanner = new Scanner(System.in);
9
10    Run | Debug
11    public static void main(String[] args) {
12        while (true) {
13            System.out.println("1. Tambah Data");
14            System.out.println("2. Tampilkan Daftar Nilai");
15            System.out.println("3. Keluar");
16            System.out.print("Pilih Menu (1-3) : ");
17
18            int pilihan = scanner.nextInt();
19            scanner.nextLine(); // Membuang newline setelah nextInt()
20
21            switch (pilihan) {
22                case 1:
23                    inputData();
24                    break;
25                case 2:
26                    tampilkanDaftarNilai();
27                    break;
28                case 3:
29                    System.out.println("Program Selesai");
30                    System.exit(0);
31                default:
32                    System.out.println("Pilihan tidak valid, silakan coba lagi.");
33            }
34        }
35    }
```

daftarMahasiswa → ArrayList yang menyimpan daftar mahasiswa.

scanner → Objek untuk membaca input dari pengguna.

di metode utama diisi dengan **while loop** untuk perulangan program dan akan berhenti jika kondisi false dan didalam perulangan ini berisi menu

Program menampilkan 3 pilihan:

Tambah Data Mahasiswa

Tampilkan Daftar Nilai

Keluar dari Program

int pilihan = scanner.nextInt();

scanner.nextLine();

scanner.nextInt() digunakan untuk membaca angka yang dipilih pengguna.

scanner.nextLine() digunakan untuk membuang newline (\n) agar input berikutnya tidak bermasalah.

dan disini menggunakan switch case untuk kondisi dalam pemilihan programnya

ika pengguna memilih 1, program akan menjalankan **inputData()**.

Jika memilih 2, program akan menjalankan **tampilkanDaftarNilai()**.

Jika memilih 3, program akan menampilkan pesan dan keluar dengan **System.exit(0)**.

Jika memasukkan angka selain 1-3, akan muncul pesan "Pilihan tidak valid".

```
public static void inputData() {
    System.out.print("Masukkan NIM: ");
    String nim = scanner.nextLine();

    System.out.print("Masukkan Nama: ");
    String nama = scanner.nextLine();

    System.out.print("Nilai Tugas: ");
    double tugas = scanner.nextDouble();

    System.out.print("Nilai UTS: ");
    double uts = scanner.nextDouble();

    System.out.print("Nilai UAS: ");
    double uas = scanner.nextDouble();
    scanner.nextLine(); // Menghindari error saat membaca input selanjutnya

    // Hitung nilai akhir
    double nilaiAkhir = hitungNilaiAkhir(tugas, uts, uas);

    // Tambahkan data ke ArrayList
    daftarMahasiswa.add(new Mahasiswa(nim, nama, tugas, uts, uas, nilaiAkhir));

    System.out.println("Data Mahasiswa Berhasil Ditambahkan!\n");
}
```

metode **inputData()** ini digunakan untuk menginput data pengguna

scanner.nextLine() digunakan untuk membaca NIM dan Nama mahasiswa.

Menggunakan **scanner.nextDouble()** untuk membaca nilai tugas, UTS, dan UAS.

scanner.nextLine() digunakan agar tidak terjadi error saat membaca input berikutnya.

double nilaiAkhir = hitungNilaiAkhir(tugas, uts, uas);

nilaiAkhir ini digunakan untuk Menghitung nilai akhir dengan memanggil method **hitungNilaiAkhir()**.

daftarMahasiswa.add(new Mahasiswa(nim, nama, tugas, uts, uas, nilaiAkhir));

ini untuk Menambahkan data mahasiswa ke dalam **ArrayList<Mahasiswa>**.

System.out.println("Data Mahasiswa Berhasil Ditambahkan!\n");

untuk Menampilkan pesan konfirmasi setelah data berhasil disimpan.

```
62 public static void tampilkanDaftarNilai() {
63     if (daftarMahasiswa.isEmpty()) {
64         System.out.println("Data Mahasiswa Kosong");
65         return;
66     }
67
68     System.out.println("\nDaftar Nilai Mahasiswa:");
69     System.out.printf("| %-10s | %-20s | %-6s | %-6s | %-6s | %-6s |\n",
70         "NIM", "Nama", "Tugas", "UTS", "UAS", "Nilai Akhir");
71
72     for (Mahasiswa mahasiswa : daftarMahasiswa) {
73         System.out.printf("| %-10s | %-20s | %-6.2f | %-6.2f | %-6.2f | %-6.2f |\n",
74             mahasiswa.nim, mahasiswa.nama, mahasiswa.tugas, mahasiswa.uts, mahasiswa.uas, mahasiswa.nilaiAkhir);
75     }
76     System.out.println();
77 }
78
79 public static double hitungNilaiAkhir(double tugas, double uts, double uas) {
80     return (tugas * 0.3) + (uts * 0.3) + (uas * 0.4);
81 }
82 }
```

public static void tampilkanDaftarNilai() {

Method ini digunakan untuk **menampilkan daftar mahasiswa** yang telah dimasukkan.

disini ada kondisi dimana

if (daftarMahasiswa.isEmpty()) {

System.out.println("Data Mahasiswa Kosong");

return;

}

Jika **daftarMahasiswa** kosong, program akan menampilkan pesan "Data Mahasiswa Kosong" dan keluar dari method

System.out.println("\nDaftar Nilai Mahasiswa:");

```
System.out.printf("| %-10s | %-20s | %-6s | %-6s | %-6s | %-6s | \n",
```

```
    "NIM", "Nama", "Tugas", "UTS", "UAS", "Nilai Akhir");
```

fungsi dari cetak tersebut untuk Menampilkan header tabel untuk daftar mahasiswa.

```
for (Mahasiswa mahasiswa : daftarMahasiswa) {
```

```
    System.out.printf("| %-10s | %-20s | %-6.2f | %-6.2f | %-6.2f | %-6.2f | \n",
```

```
        mahasiswa.nim, mahasiswa.nama, mahasiswa.tugas, mahasiswa.uts,  
        mahasiswa.uas, mahasiswa.nilaiAkhir);
```

```
}
```

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

Looping ini berguna untuk melalui setiap mahasiswa yang tersimpan dalam ArrayList dan menampilkan data dalam format tabel.

Methode **HitungNilaiAkhir()**

```
public static double hitungNilaiAkhir(double tugas, double uts, double uas) {
```

```
    return (tugas * 0.3) + (uts * 0.3) + (uas * 0.4);
```

```
}
```

methode ini bergyba untuk Menghitung nilai akhir berdasarkan bobot:

Tugas = 30%

UTS = 30%

UAS = 40%

Mengembalikan hasil perhitungan nilai akhir.

```
class Mahasiswa {
```

```
    String nim;
```

```
String nama;  
  
double tugas;  
  
double uts;  
  
double uas;  
  
double nilaiAkhir;  
  
}
```

class ini berguna untuk menyimpan data mahasiswa

```
public Mahasiswa(String nim, String nama, double tugas, double uts, double uas,  
double nilaiAkhir) {  
  
    this.nim = nim;  
  
    this.nama = nama;  
  
    this.tugas = tugas;  
  
    this.uts = uts;  
  
    this.uas = uas;  
  
    this.nilaiAkhir = nilaiAkhir;  
  
}
```

Constructor untuk membuat objek Mahasiswa dengan data yang telah diinputkan.

LATIHAN 3

```

public class Sorting {
    public void bubbleSort(int[] arr) {
        System.out.println("\nBubble Sort");
        int n = arr.length;
        for (int i = 0; i < n; 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;
                    System.out.println(Arrays.toString(arr));
                }
            }
        }
    }
}

```

Bubble sort:

buble sort bekerja dengan menukar elemen berdasarkan jika elemetn pertama lebih besar dari elemeb kedua maka loop i menentukan jumlah iterasi dan loop j memperbandingkan dan menukar elemen

```

public void quickSort(int[] arr, int low, int high) {
    if (low < high) {
        int pivot = partition(arr, low, high);
        quickSort(arr, low, pivot - 1);
        quickSort(arr, pivot + 1, high);
    }
}

```

Quicksort:

cara kerja quick sort memilih elemen pivot dan membagi array menjadi dua bagian

elemen yang lebih kecil akan diletakkan di kiri dan yang lebih besar di kanan

proses ini dilakukan dengan rekursi sampai array terurut


```

public static int partition(int[] arr, int low, int high) {
    int pivot = arr[high];
    int i = low - 1;
    for (int j = low; j < high; j++) {
        if (arr[j] < pivot) {
            i++;
            int temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
            System.out.println(Arrays.toString(arr));
        }
    }
    int temp = arr[i + 1];
    arr[i + 1] = arr[high];
    arr[high] = temp;
    System.out.println(Arrays.toString(arr));
    return i + 1;
}

```

Partition():

di fungsi partition ini berkonsep elemen terakhir(arr[high]) sebagai pivot atau bilangan terakhir di array. perulangan di kode tersebut dimana j akan berjalan dari indeks awal (low) ke sebelum pivot (high -1). jika menemukan element yang lebih kecil dari pivot akan ditukar dengan element di posisi i+1

```

48 public void insertionSort(int[] arr) {
49     System.out.println("\nInsertion Sort");
50     for (int i = 1; i < arr.length; i++) {
51         int key = arr[i];
52         int j = i - 1;
53         while (j >= 0 && arr[j] > key) {
54             arr[j + 1] = arr[j];
55             j = j - 1;
56             System.out.println(Arrays.toString(arr));
57         }
58         arr[j + 1] = key;
59         System.out.println(Arrays.toString(arr));
60     }
61 }
62

```

insertionSort ini algoritma yang membandingkan elemen dengan elemen sebelumnya.

di kode tersebut melakukan perulangan untuk mengetahui isi arr dan int key akan mengambil elemen ke dua untuk dibandingkan dengan element yang pertama dengan menggunakan perulangan while untuk menggesernya cara kerjanya

while (j>=0 && arr[j] > key) j>=0 untuk memastikan tidak keluar **array**. **arr[j]>key** akan mengecek apakah elemen sebelum nya lebih besar dengan key jika lebih besar maka **arr[j]**

akan digeser ke kanan dan menggantikan key dan $j=j-1$ menggeser j kekiri untuk mengecek elemen sebelumnya.

```
64 public void selectionSort(int[] arr) {
65     System.out.println("\nSelection Sort");
66     int n = arr.length;
67     for (int i = 0; i < n - 1; i++) {
68         int minIdx = i;
69         for (int j = i + 1; j < n; j++) {
70             if (arr[j] < arr[minIdx]) {
71                 minIdx = j;
72             }
73         }
74         int temp = arr[minIdx];
75         arr[minIdx] = arr[i];
76         arr[i] = temp;
77         System.out.println(Arrays.toString(arr));
78     }
79 }
80
```

SelectionSort

Pilih elemen pertama sebagai minIdx (elemen terkecil sementara).

Bandingkan minIdx dengan elemen lain dalam array untuk mencari elemen terkecil.

Jika ditemukan elemen yang lebih kecil, perbarui minIdx. dengan menggunakan **for(int j =i +1; j<n; j++)** mencari element terkecil untuk dalam array yang belum diurutkan. dan **if (arr[j] < arr[minIdx])** Jika menemukan elemen yang lebih kecil dari **arr[minIdx]**, update **minIdx** dengan posisi elemen baru.

int temp = arr[minIdx]; Simpan elemen terkecil di variabel **temp**.

arr[minIdx] = arr[i]; Tukar elemen di posisi **minIdx** dengan elemen di posisi **i**.

arr[i] = temp;

```

public void mergeSort(int[] arr, int left, int right) {
    if (left < right) {
        int mid = left + (right - left) / 2;
        mergeSort(arr, left, mid);
        mergeSort(arr, mid + 1, right);
        merge(arr, left, mid, right);
    }
}

```

if (left < right)

left adalah indeks awal bagian array yang sedang diproses.

right adalah indeks akhir bagian array yang sedang diproses.

Jika **left** lebih kecil dari **right**, artinya array tersebut memiliki lebih dari satu elemen. Jadi, kita perlu melakukan pemisahan lebih lanjut.

int mid = left + (right - left) / 2;

mid adalah indeks tengah dari array yang sedang diproses.

Ini digunakan untuk membagi array menjadi dua bagian yang lebih kecil: satu bagian kiri dan satu bagian kanan.

`mergeSort(arr, left, mid);` // Menyortir bagian kiri

`mergeSort(arr, mid + 1, right);` // Menyortir bagian kanan

Fungsi **mergeSort** dipanggil secara rekursif pada dua bagian array:

Bagian kiri array: dari indeks **left** hingga **mid**.

Bagian kanan array: dari indeks **mid + 1** hingga **right**.

merge(arr, left, mid, right); setelah kanan dan kiri terurut secara rekursif maka akan digabungkan

```

91 private void merge(int[] arr, int left, int mid, int right) {
92     int n1 = mid - left + 1;
93     int n2 = right - mid;
94     int[] L = new int[n1];
95     int[] R = new int[n2];
96
97     for (int i = 0; i < n1; i++) L[i] = arr[left + i];
98     for (int j = 0; j < n2; j++) R[j] = arr[mid + 1 + j];
99
100    int i = 0, j = 0, k = left;
101    while (i < n1 && j < n2) {
102        if (L[i] <= R[j]) {
103            arr[k] = L[i];
104            i++;
105        } else {
106            arr[k] = R[j];
107            j++;
108        }
109        k++;
110    }
111    while (i < n1) {
112        arr[k] = L[i];
113        i++;
114        k++;
115    }
116    while (j < n2) {
117        arr[k] = R[j];
118        j++;
119    }

```

L[] untuk menyimpan elemen dari bagian kiri.

R[] untuk menyimpan elemen dari bagian kanan.

n1 = mid - left + 1 adalah ukuran array kiri.

n2 = right - mid adalah ukuran array kanan.

untuk menyimpan data ke array sementara

for (int i = 0; i < n1; i++) L[i] = arr[left + i];

for (int j = 0; j < n2; j++) R[j] = arr[mid + 1 + j];

while (i < n1 && j < n2) dengan ini untuk membandingkan element dari **L[]** dan **R[]**
 Element yang lebih kecil akan dimasukkan ke **arr[]**

while (j < n2) {

```
arr[k] = R[j];
```

```
j++;
```

```
k++;
```

```
} Jika masih ada sisa elemen di L, masukkan ke arr
```

```
while (j < n2) {
```

```
arr[k] = R[j];
```

```
j++;
```

```
k++;
```

```
} // Jika masih ada sisa elemen di R, masukkan ke arr
```

```
6 public class Main {
  Run | Debug
7   public static void main(String[] args) {
8       Scanner scanner = new Scanner(System.in);
9       System.out.print("Masukkan angka (pisahkan dengan spasi): ");
10      String input = scanner.nextLine();
11      String[] strArr = input.split(" ");
12      int[] arr = new int[strArr.length];
13
14      for (int i = 0; i < strArr.length; i++) {
15          arr[i] = Integer.parseInt(strArr[i]);
16      }
17      Sorting sorting = new Sorting();
18      // Membuat salinan array untuk setiap algoritma
19      int[] bubbleArr = arr.clone();
20      int[] quickArr = arr.clone();
21      int[] insertionArr = arr.clone();
22      int[] selectionArr = arr.clone();
23      int[] mergeArr = arr.clone();
24      // Menjalankan semua algoritma sorting
25      sorting.bubbleSort(bubbleArr);
26      System.out.println("\nQuick Sort:");
27      sorting.quickSort(quickArr, 0, quickArr.length - 1);
28      System.out.println(Arrays.toString(quickArr));
29      sorting.insertionSort(insertionArr);
30      sorting.selectionSort(selectionArr);
31      System.out.println("\nMerge Sort:");
32      sorting.mergeSort(mergeArr, 0, mergeArr.length - 1);
33      System.out.println(Arrays.toString(mergeArr));
34  }
```

Program ini membaca angka dari pengguna, lalu **menjalankan lima algoritma sorting** secara independen.

Menggunakan .clone() untuk memastikan setiap algoritma bekerja dengan array asli.

Menampilkan hasil sorting setelah masing-masing metode selesai.

LATIHAN 4

```
public class ArrayMatrix {  
    Run | Debug  
    public static void main(String[] args) {  
        Scanner scanner = new Scanner(System.in);  
  
        // Menu pilihan operasi  
        System.out.println("Pilih Operasi Matrix:");  
        System.out.println("1. Penjumlahan");  
        System.out.println("2. Pengurangan");  
        System.out.println("3. Perkalian");  
        System.out.println("4. Pembagian");  
        System.out.print("Masukkan pilihan (1-4): ");  
        int choice = scanner.nextInt();  
  
        // Input ukuran matrix  
        System.out.print("Masukkan jumlah data: ");  
        int size = scanner.nextInt();  
  
        // Deklarasi ArrayList  
        ArrayList<Integer> matrix1 = new ArrayList<>();  
        ArrayList<Integer> matrix2 = new ArrayList<>();  
        ArrayList<Integer> result = new ArrayList<>();  
  
        // Input matrix pertama  
        System.out.println("\nMasukkan nilai Matrix 1:");  
        for (int i = 0; i < size; i++) {  
            System.out.print("Nilai ke-" + (i + 1) + ": ");  
            matrix1.add(scanner.nextInt());  
        }  
    }  
}
```

import java.util.ArrayList;

import java.util.Scanner;

ArrayList: Digunakan untuk menyimpan elemen-elemen dari dua matriks yang akan dioperasikan.

Scanner: Digunakan untuk menerima input dari pengguna.

Scanner scanner = new Scanner(System.in);

Membuat objek **Scanner** untuk membaca input dari pengguna.

ArrayList<Integer> matrix1 = new ArrayList<>();

ArrayList<Integer> matrix2 = new ArrayList<>();

ArrayList<Integer> result = new ArrayList<>();

matrix1: Menyimpan elemen-elemen dari matriks pertama.

matrix2: Menyimpan elemen-elemen dari matriks kedua.

result: Menyimpan hasil operasi pada dua matriks

dan setelah kode ini ada kode loop untuk menginput berapa banyak bilangan untuk matrix

```
// Operasi berdasarkan pilihan
switch (choice) {
    case 1:
        result = addMatrix(matrix1, matrix2);
        System.out.println("\nHasil Penjumlahan:");
        break;
    case 2:
        result = subtractMatrix(matrix1, matrix2);
        System.out.println("\nHasil Pengurangan:");
        break;
    case 3:
        result = multiplyMatrix(matrix1, matrix2);
        System.out.println("\nHasil Perkalian:");
        break;
    case 4:
        result = divideMatrix(matrix1, matrix2);
        System.out.println("\nHasil Pembagian:");
        break;
    default:
        System.out.println("Pilihan tidak valid!");
        return;
}

// Tampilkan hasil
for (int i = 0; i < size; i++) {
    System.out.println("Index " + i + ": " + result.get(i));
}

scanner.close();
```

terdapat program switch case untuk metode operasi yang sesuai dengan input pengguna.

Jika pengguna memasukkan angka di luar rentang 1-4, program akan menampilkan pesan kesalahan dan berhenti (**return**). dan fungsi perulangan tersebut untuk menampilkan hasilnya

```

72
73 // Method operasi matrix
74 // Penjumlahan
75 public static ArrayList<Integer> addMatrix(ArrayList<Integer> m1, ArrayList<Integer> m2) {
76     ArrayList<Integer> result = new ArrayList<>();
77     for (int i = 0; i < m1.size(); i++) {
78         result.add(m1.get(i) + m2.get(i));
79     }
80     return result;
81 }
82
83 // Pengurangan
84 public static ArrayList<Integer> subtractMatrix(ArrayList<Integer> m1, ArrayList<Integer> m2) {
85     ArrayList<Integer> result = new ArrayList<>();
86     for (int i = 0; i < m1.size(); i++) {
87         result.add(m1.get(i) - m2.get(i));
88     }
89     return result;
90 }
91
92 // Perkalian
93 public static ArrayList<Integer> multiplyMatrix(ArrayList<Integer> m1, ArrayList<Integer> m2) {
94     ArrayList<Integer> result = new ArrayList<>();
95     for (int i = 0; i < m1.size(); i++) {
96         result.add(m1.get(i) * m2.get(i));
97     }
98     return result;
99 }

```

Disinii ada metode operasi untuk dua matriks

public static ArrayList<Integer> addMatrix() untuk melakukan penjumlahan

public static ArrayList<Integer> subtractMatrix() untuk melakukan pengurangan matriks 1 dengan matriks 2

public static ArrayList<Integer> multiplyMatrix() Mengalikan setiap elemen dari matriks 1 dengan elemen yang sesuai di matriks 2.

public static ArrayList<Integer> divideMatrix() program mengecek apakah pembagi (m2.get(i)) bernilai nol untuk menghindari ArithmeticException.

Jika pembagi bernilai nol, akan ditampilkan peringatan, dan hasilnya dianggap **0**.

Matrix.java

```
// Input Matrix A
System.out.print("input baris matrix A = ");
int rowsA = scanner.nextInt();
System.out.print("input kolom matrix A = ");
int colsA = scanner.nextInt();
int[][] matrixA = new int[rowsA][colsA];

for (int i = 0; i < rowsA; i++) {
    for (int j = 0; j < colsA; j++) {
        System.out.printf("input elemen matrix A [%d,%d] = ", i, j);
        matrixA[i][j] = scanner.nextInt();
    }
}
```

Meminta pengguna memasukkan jumlah baris dan kolom matriks A.

Menggunakan nested loop untuk mengisi elemen matriks satu per satu.

```
23 // Input Matrix B
24 System.out.print("\ninput baris matrix B = ");
25 int rowsB = scanner.nextInt();
26 System.out.print("input kolom matrix B = ");
27 int colsB = scanner.nextInt();
28 int[][] matrixB = new int[rowsB][colsB];
29
30 for (int i = 0; i < rowsB; i++) {
31     for (int j = 0; j < colsB; j++) {
32         System.out.printf("input elemen matrix B [%d,%d] = ", i, j);
33         matrixB[i][j] = scanner.nextInt();
34     }
35 }
36
```

Mirip dengan input matriks A, tetapi untuk matriks B.

```

63 // Method Penjumlahan Matrix
64 public static int[][] addMatrices(int[][] a, int[][] b) {
65     int rows = a.length;
66     int cols = a[0].length;
67     int[][] result = new int[rows][cols];
68
69     for (int i = 0; i < rows; i++) {
70         for (int j = 0; j < cols; j++) {
71             result[i][j] = a[i][j] + b[i][j];
72         }
73     }
74     return result;
75 }

```

Mengiterasi setiap elemen matriks **A** dan **B** lalu menjumlahkan elemen yang sesuai.

Mengembalikan hasil dalam bentuk matriks **result**.

Syarat: Ukuran **baris** dan **kolom** harus **sama** untuk bisa melakukan penjumlahan.

```

84 // Method Transpose Matrix
85 public static int[][] transposeMatrix(int[][] matrix) {
86     int rows = matrix.length;
87     int cols = matrix[0].length;
88     int[][] result = new int[cols][rows];
89
90     for (int i = 0; i < rows; i++) {
91         for (int j = 0; j < cols; j++) {
92             result[j][i] = matrix[i][j];
93         }
94     }
95     return result;
96 }
97

```

Menukar posisi **baris** dan **kolom**.

Matriks yang **awal baris** m , **kolom** n akan menjadi **baris** n , **kolom** m .

```

67 // Method Perkalian Matrix
68 public static int[][] multiplyMatrices(int[][] a, int[][] b) {
69     int rowsA = a.length;
70     int colsA = a[0].length;
71     int colsB = b[0].length;
72     int[][] result = new int[rowsA][colsB];
73
74     for (int i = 0; i < rowsA; i++) {
75         for (int j = 0; j < colsB; j++) {
76             for (int k = 0; k < colsA; k++) {
77                 result[i][j] += a[i][k] * b[k][j];
78             }
79         }
80     }
81     return result;
82 }

```

perkalian matriks:

Jumlah kolom pada matriks A harus sama dengan jumlah baris pada matriks B.

Misal A ($m \times n$) \times B ($n \times p$) \rightarrow hasilnya ($m \times p$).

```

98 // Method untuk Menampilkan Matrix
99 public static void printMatrix(int[][] matrix) {
100     for (int[] row : matrix) {
101         for (int value : row) {
102             System.out.print(value + " ");
103         }
104         System.out.println();
105     }
106 }

```

Mencetak isi matriks dengan format yang rapi.

Queue.java

```
5 public class Queue {
6     private int[] queueArray;
7     private int front;
8     private int rear;
9     private int size;
10    private int capacity;
11
12    public Queue(int capacity) {
13        this.capacity = capacity;
14        queueArray = new int[capacity];
15        front = 0;
16        rear = -1;
17        size = 0;
18    }
19
20    public void insert(int item) {
21        if (isFull()) {
22            System.out.println("Queue is full. Cannot insert " + item);
23            return;
24        }
25        rear = (rear + 1) % capacity;
26        queueArray[rear] = item;
27        size++;
28        System.out.println(item + " inserted to queue");
29    }
```

queueArray → Array untuk menyimpan elemen queue.

front → Menunjuk elemen depan (elemen pertama dalam antrian).

rear → Menunjuk elemen belakang (elemen terakhir yang masuk).

size → Jumlah elemen dalam antrian.

capacity → Kapasitas maksimum queue.

konstruktor Queue untuk menginisialisasi queue dengan kapasitas tertentu.

rear = -1 karena queue awalnya kosong.

metode insert untuk menambah elemen
dan ada kondisi fungsi Jika penuh, tidak bisa menambah elemen.

Circular queue: rear akan kembali ke awal (0) saat mencapai batas akhir.

```

31     public int remove() {
32         if (isEmpty()) {
33             System.out.println("Queue is empty");
34             return Integer.MIN_VALUE;
35         }
36         int item = queueArray[front];
37         front = (front + 1) % capacity;
38         size--;
39         System.out.println(item + " removed from queue");
40         return item;
41     }

```

metode remove untuk menghapus elemen di **front**.

Circular queue: **front** kembali ke indeks awal setelah mencapai batas akhir

```

public int peek() {
    if (isEmpty()) {
        System.out.println("Queue is empty");
        return Integer.MIN_VALUE;
    }
    int item = queueArray[front];
    System.out.println("Front element = " + item);
    return item;
}

```

metode peek berguna untuk melihat elemen terdepan tanpa menghapusnya

```

public boolean isEmpty() {
    return size == 0;
}

public boolean isFull() {
    return size == capacity;
}

public int size() {
    System.out.println("Size = " + size);
    return size;
}

```

disini terdapat metode untuk mengecek status :

terdiri dari

isEmpty() → Mengecek apakah queue kosong.

isFull() → Mengecek apakah queue penuh.

size() → Mengembalikan jumlah elemen dalam queue.

```
66     public void display() {
67         System.out.print("Queue = ");
68         if (isEmpty()) {
69             System.out.println("empty");
70             return;
71         }
72         int count = 0;
73         int index = front;
74         while (count < size) {
75             System.out.print(queueArray[index] + " ");
76             index = (index + 1) % capacity;
77             count++;
78         }
79         System.out.println();
80     }
```

Metode display untuk menampilkan Queue dan didalam metode ini

Jika queue kosong, cetak "empty".

Jika ada elemen, cetak dari front ke rear dengan **circular traversal**.

```
Run | Debug
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter the capacity of the queue: ");
    int capacity = scanner.nextInt();
    scanner.nextLine(); // Membersihkan newline

    Queue queue = new Queue(capacity);

    char choice;
```

dalam main kode diatas berfungsi untuk Meminta kapasitas queue dari pengguna dan Membuat objek queue dengan kapasitas tersebut.

```

    char choice;
    do {
        System.out.println("\nQueue Operations");
        System.out.println("1. tambah");
        System.out.println("2. hapus");
        System.out.println("3. peek");
        System.out.println("4. cek kosong");
        System.out.println("5. check full");
        System.out.println("6. size");
        System.out.print("\npilihan ");

        int operation = scanner.nextInt();
        scanner.nextLine(); // Membersihkan newline setelah nextInt()
    }

```

disini ada do while dengan do untuk menampilkan operasi Queue

```

104         switch (operation) {
105             case 1:
106                 System.out.print("Masukan data: ");
107                 int value = scanner.nextInt();
108                 scanner.nextLine();
109                 queue.insert(value);
110                 break;
111             case 2:
112                 queue.remove();
113                 break;
114             case 3:
115                 queue.peek();
116                 break;
117             case 4:
118                 System.out.println("Queue empty? " + queue.isEmpty());
119                 break;
120             case 5:
121                 System.out.println("Queue full? " + queue.isFull());
122                 break;
123             case 6:
124                 queue.size();
125                 break;
126             default:
127                 System.out.println("pilihan invalid");
128         }
129
130         queue.display();

```

Switch case menangani pilihan pengguna

```

130         queue.display();
131
132         System.out.print("\nmau lanjut atau tidak ");
133         choice = scanner.nextLine().charAt(0);
134     } while (Character.toLowerCase(choice) == 'y');
135
136     scanner.close();
137 }

```

Loop berjalan selama pengguna memilih 'y'.

Queue ditampilkan setelah setiap operasi.

