

LAPORAN HASIL PRAKTIKUM
ALGORITMA DAN STRUKTUR DATA
JOBSHEET 4



NAMA :MAULIDYAAFRIANI

NIM: 2441070200559


KELAS : 1E

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG

2025

1.1 PERCOBAAN 1

1. Buatlah class baru dengan nama Faktorial

 faktorial15.java

2. Lengkapi class Faktorial dengan atribut dan method yang telah digambarkan di dalam diagram class, menambahkan method faktorialBF() dan method faktorialDC()

```
import java.util.Scanner;
public class faktorial15 {

    int faktorialBF(int n) {
        int fakto = 1;
        for (int i=1; i<=n; i++ ) {
            fakto = fakto * i;
        }
        return fakto;
    }
    int faktorialDC(int n) {
        if (n==1) {
            return 1;
        }else{
            int fakto = n * faktorialDC(n-1);
            return fakto;
        }
    }
}
```

3. Coba jalankan (Run) class Faktorial dengan membuat class baru MainFaktorial.

```
import java.util.Scanner;

public class MainFaktorial15 {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        System.out.print("Masukkan nilai: ");
        int nilai = input.nextInt();
        faktorial15 fk = new faktorial15();

        System.out.println("Nilai faktorial " + nilai + "Menggunakan
BF: " + fk.faktorialBF(nilai));

        System.out.println("Nilai faktorial " + nilai + "Menggunakan
DC: "+ fk.faktorialDC(nilai));

    }
}
```

4. Run program

```
Masukkan nilai: 5
Nilai faktorial 5Menggunakan BF: 120
Nilai faktorial 5Menggunakan DC: 120
```

PERTANYAAN:

1. Pada base line Algoritma Divide Conquer untuk melakukan pencarian nilai faktorial, jelaskan perbedaan bagian kode pada penggunaan if dan else!

JAWAB:

- if (n == 1) →
- Base case, menghentikan rekursi. else → Recursive case, memanggil fungsi dengan n-1.

2. Apakah memungkinkan perulangan pada method faktorialBF() diubah selain menggunakan for?

Buktikan!

JAWAB: Bisa, Bisa pakai while-loop atau bahkan rekursi.

3. Jelaskan perbedaan antara fakto *= i; dan int fakto = n * faktorialDC(n-1); !

JAWAB:

- fakto *= i; → Iteratif, mengupdate variabel dalam satu loop.
- int fakto = n * faktorialDC(n-1); → Rekursif, menciptakan variabel baru setiap pemanggilan fungsi.

4. Buat Kesimpulan tentang perbedaan cara kerja method faktorialBF() dan faktorialDC()!

JAWAB:

Brute Force (faktorialBF()) → Pakai loop, lebih cepat & efisien dalam memori. Divide & Conquer (faktorialDC()) → Pakai rekursi, lebih elegan tapi ada overhead pemanggilan fungsi.

1.2 PERCOBAAN 2

1. membuat class baru dengan nama Pangkat. Dan di dalam class Pangkat tersebut, buat atribut angka yang akan dipangkatkan sekaligus dengan angka pemangkatnya

```
public class pangkat15 {
    int nilai, pangkat;
```

2. Tambahkan konstruktor berparameter dan Pada class Pangkat tersebut, tambahkan method PangkatBF()

```
    public pangkat15(int n, int p) {
        nilai = n;;
        pangkat = p;
    }
    int pangkatBF(int a, int n) {
        int hasil = 1;
        for(int i = 0; i < n; i++){
            hasil = hasil * a;
        }
        return hasil;
```

```
}
```

3. Pada class Pangkat juga tambahkan method PangkatDC()

```
int pangkatDC(int a, int n) {
    if (n==1) {
        return a;
    }else{
        if (n%2==1) {
            return (pangkatDC(a, n/2) * pangkatDC(a, n/2) *a);
        }else{
            return (pangkatDC(a, n/2) * pangkatDC(a, n/2));
        }
    }
}
```

4. Selanjutnya buat class baru yang di dalamnya terdapat method main. Class tersebut dapat dinamakan MainPangkat. Tambahkan kode pada class main untuk menginputkan jumlah elemen yang akan dihitung pangkatnya.

```
Scanner input = new Scanner(System.in);
System.out.print("Masukkan jumlah elemen: ");
int elemen = input.nextInt();
```

5. Nilai pada tahap 5 selanjutnya digunakan untuk instansiasi array of objek. Di dalam Kode berikut ditambahkan proses pengisian beberapa nilai yang akan dipangkatkan sekaligus dengan pemangkatnya

```
pangkat15[] png = new pangkat15[elemen];

for (int i = 0; i < elemen; i++) {
    System.out.print("Masukkan nilai basis elemen ke-" + (i
+ 1) + ": ");
    int basis = input.nextInt();
    System.out.print("Masukkan nilai pangkat elemen ke-" +
(i + 1) + ": ");
    int pangkat = input.nextInt();

    png[i] = new pangkat15(basis, pangkat);
}
```

6. Kemudian, panggil hasil nya dengan mengeluarkan return value dari method PangkatBF() dan PangkatDC().

```
System.out.println("HASIL PANGKAT BRUTEFORCE: ");
for (pangkat15 p : png) {
    System.out.println(p.nilai+"^"+p.pangkat+":
"+p.pangkatBF(p.nilai, p.pangkat));
}
System.out.println("HASIL PANGKAT DIVIDE AND CONQUER: ");
for (pangkat15 p : png) {
    System.out.println(p.nilai+"^"+p.pangkat+":
"+p.pangkatDC(p.nilai, p.pangkat));
}

}
```

```
}
```

7. Hasil run

```
Masukkan jumlah elemen: 3
Masukkan nilai basis elemen ke-1: 2
Masukkan nilai pangkat elemen ke-1: 3
Masukkan nilai basis elemen ke-2: 4
Masukkan nilai pangkat elemen ke-2: 5
Masukkan nilai basis elemen ke-3: 6
Masukkan nilai pangkat elemen ke-3: 7
HASIL PANGKAT BRUTEFORCE:
2^3: 8
4^5: 1024
6^7: 279936
HASIL PANGKAT DIVIDE AND CONQUER:
2^3: 8
4^5: 1024
6^7: 279936
```

PERTANYAAN:

1. Jelaskan mengenai perbedaan 2 method yang dibuat yaitu pangkatBF() dan pangkatDC()!

JAWAB: ☐ pangkatBF(): Menggunakan perulangan (*looping*), kompleksitas **O(n)**.

☐ pangkatDC(): Menggunakan rekursi (*Divide and Conquer*), lebih cepat dengan kompleksitas **O(log n)**.

2. Apakah tahap *combine* sudah termasuk dalam kode tersebut? Tunjukkan!

JAWAB: iyaa, tahap *combine* terjadi dalam pangkatDC(), saat hasil rekursi dikalikan kembali untuk mendapatkan hasil akhir.

3. Pada method pangkatBF() terdapat parameter untuk melewati nilai yang akan dipangkatkan dan pangkat berapa, padahal di sisi lain di class Pangkat telah ada atribut nilai dan pangkat, apakah menurut Anda method tersebut tetap relevan untuk memiliki parameter? Apakah bisa jika method tersebut dibuat dengan tanpa parameter? Jika bisa, seperti apa method pangkatBF() yang tanpa parameter?

JAWAB: Tidak perlu. Bisa dibuat tanpa parameter dengan langsung memakai atribut objek (nilai dan pangkat).

4. Tarik tentang cara kerja method pangkatBF() dan pangkatDC()!

JAWAB: ☐ pangkatBF(): Mengalikan nilai sebanyak pangkat kali (**O(n)**).

☐ pangkatDC(): Membagi masalah menjadi submasalah lebih kecil, menggunakan rekursi untuk mempercepat perhitungan (**O(log n)**)

1.3 PERCOBAAN 3

1. Buat class baru yaitu class **Sum**. Tambahkan pula konstruktor pada class Sum.

```
public class Sum15 {
    double keuntungan[];

    Sum15(int el){
        keuntungan = new double[el];
    }
}
```

2. Tambahkan method TotalBF() yang akan menghitung total nilai array dengan cara *iterative*.

```
double totalBF() {
    double total = 0;
}
```

```

        for(int i=0;i<keuntungan.length;i++){
            total = total+keuntungan[i];
        }
        return total;
    }
}

```

3. Tambahkan pula method TotalDC() untuk implementasi perhitungan nilai total array menggunakan algoritma Divide and Conquer

```

double totalDC(double arr[], int l, int r){
    if(l==r){
        return arr[l];
    }
    int mid = (l+r)/2;
    double lsum = totalDC(arr, l, mid);
    double rsum = totalDC(arr, mid+1, r);
    return lsum+rsum;
}
}

```

4. Buat class baru yaitu MainSum. Di dalam kelas ini terdapat method main. Pada method ini user dapat menuliskan berapa bulan keuntungan yang akan dihitung. Dalam kelas ini sekaligus dibuat instansiasi objek untuk memanggil atribut ataupun fungsi pada class Sum

```

import java.util.Scanner;
public class MainSum15 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Masukkan jumlah elemen: ");
        int elemen = input.nextInt();
    }
}

```

5. Karena yang akan dihitung adalah total nilai keuntungan, maka ditambahkan pula pada method main mana array yang akan dihitung. Array tersebut merupakan atribut yang terdapat di class Sum, maka dari itu dibutuhkan pembuatan objek Sum terlebih dahulu.

```

Sum15 sm = new Sum15(elemen);
for (int i = 0; i < elemen; i++) {
    System.out.print("Masukkan keuntungan ke-" + (i + 1)
+ ": ");
    sm.keuntungan[i] = input.nextDouble();
}
}

```

6. Tampilkan hasil perhitungan melalui objek yang telah dibuat untuk kedua cara yang ada (Brute Force dan Divide and Conquer)

```

System.out.println("Total keuntungan menggunakan Brute force: " +
sm.totalBF());
System.out.println("Total keuntungan menggunakan Divide
and Conquer: " +sm.totalDC(sm.keuntungan, 0, elemen - 1));
}
}

```

7. Hasil run

```
Masukkan jumlah elemen: 5
Masukkan keuntungan ke-1: 10
Masukkan keuntungan ke-2: 20
Masukkan keuntungan ke-3: 30
Masukkan keuntungan ke-4: 40
Masukkan keuntungan ke-5: 50
Total keuntungan menggunakan Bruteforce: 100.0
Total keuntungan menggunakan Divide and Conquer: 100.0
```

PERTANYAAN:

1. Kenapa dibutuhkan variable mid pada method TotalDC()?
JAWAB: mid digunakan untuk membagi array menjadi dua bagian dalam pendekatan *Divide and Conquer*.
2. Untuk apakah statement di bawah ini dilakukan dalam TotalDC()?

```
double lsum = totalDC(arr, l, mid);
double rsum = totalDC(arr, mid+1, r);
```

JAWAB: Memecah masalah menjadi dua submasalah lebih kecil untuk dihitung secara rekursif.

3. Kenapa diperlukan penjumlahan hasil lsum dan rsum seperti di bawah ini?

```
return lsum+rsum;
```

Apakah base case dari totalDC()?

JAWAB: Menggabungkan hasil dari dua submasalah untuk mendapatkan total keseluruhan.

4. Apakah base case dari totalDC()?

JAWAB: Saat hanya ada satu elemen ($l == r$), fungsi langsung mengembalikan nilai elemen tersebut.

5. Tarik Kesimpulan tentang cara kerja totalDC()

JAWAB: ☐ Menggunakan rekursi untuk membagi array menjadi dua bagian, menghitung totalnya, lalu menggabungkan hasilnya.

☐ Efisiensi lebih baik dibandingkan metode *Brute Force* karena memiliki kompleksitas **$O(\log n)$** .

1.4 LATIHAN PRAKTIKUM

1. Nilai UTS tertinggi menggunakan Divide and Conquer!

```
class Mahasiswa {
    String nama;
    String NIM;
    int tahunMasuk;
    int nilaiUTS;
    int nilaiUAS;

    Mahasiswa(String nama, String NIM, int tahunMasuk, int
nilaiUTS, int nilaiUAS) {
        this.nama = nama;
        this.NIM = NIM;
        this.tahunMasuk = tahunMasuk;
        this.nilaiUTS = nilaiUTS;
        this.nilaiUAS = nilaiUAS;
    }
}
```

```

public class NilaiUTSTertinggi15{

    public static int cariMaxUTS(Mahasiswa[] mhs, int left, int
right) {
        if (left == right) {
            return mhs[left].nilaiUTS;
        }
        int mid = (left + right) / 2;
        int maxLeft = cariMaxUTS(mhs, left, mid);
        int maxRight = cariMaxUTS(mhs, mid + 1, right);
        return Math.max(maxLeft, maxRight);
    }

    public static void main(String[] args) {
        Mahasiswa[] mahasiswaList = {
            new Mahasiswa("Ahmad", "220101001", 2022, 78, 82),
            new Mahasiswa("Budi", "220101002", 2022, 85, 88),
            new Mahasiswa("Cindy", "220101003", 2021, 90, 87),
            new Mahasiswa("Dian", "220101004", 2021, 76, 79),
            new Mahasiswa("Eko", "220101005", 2023, 92, 95),
            new Mahasiswa("Fajar", "220101006", 2020, 88, 85),
            new Mahasiswa("Gina", "220101007", 2023, 80, 83),
            new Mahasiswa("Hadi", "220101008", 2020, 82, 84)
        };

        int maxUTS = cariMaxUTS(mahasiswaList, 0,
mahasiswaList.length - 1);
        System.out.println("Nilai UTS tertinggi: " + maxUTS);
    }
}

```

2. Nilai UTS terendah menggunakan Divide and Conquer!

```

class Mahasiswa {
    String nama;
    String NIM;
    int tahunMasuk;
    int nilaiUTS;
    int nilaiUAS;

    Mahasiswa(String nama, String NIM, int tahunMasuk, int
nilaiUTS, int nilaiUAS) {
        this.nama = nama;
        this.NIM = NIM;
        this.tahunMasuk = tahunMasuk;
        this.nilaiUTS = nilaiUTS;
        this.nilaiUAS = nilaiUAS;
    }
}

public class NilaiUTSTerenda15 {

    public static int cariMinUTS(Mahasiswa[] mhs, int left, int
right) {
        if (left == right) {
            return mhs[left].nilaiUTS;
        }
        int mid = (left + right) / 2;
        int minLeft = cariMinUTS(mhs, left, mid);

```



```

        int minRight = cariMinUTS(mhs, mid + 1, right);
        return Math.min(minLeft, minRight);
    }

    public static void main(String[] args) {
        Mahasiswa[] mahasiswaList = {
            new Mahasiswa("Ahmad", "220101001", 2022, 78, 82),
            new Mahasiswa("Budi", "220101002", 2022, 85, 88),
            new Mahasiswa("Cindy", "220101003", 2021, 90, 87),
            new Mahasiswa("Dian", "220101004", 2021, 76, 79),
            new Mahasiswa("Eko", "220101005", 2023, 92, 95),
            new Mahasiswa("Fajar", "220101006", 2020, 88, 85),
            new Mahasiswa("Gina", "220101007", 2023, 80, 83),
            new Mahasiswa("Hadi", "220101008", 2020, 82, 84)
        };

        int minUTS = cariMinUTS(mahasiswaList, 0,
mahasiswaList.length - 1);
        System.out.println("Nilai UTS terendah: " + minUTS);
    }
}

```

3. Rata-rata nilai UAS dari semua mahasiswa menggunakan Brute Force!

```

class Mahasiswa {
    String nama;
    String NIM;
    int tahunMasuk;
    int nilaiUTS;
    int nilaiUAS;

    Mahasiswa(String nama, String NIM, int tahunMasuk, int
nilaiUTS, int nilaiUAS) {
        this.nama = nama;
        this.NIM = NIM;
        this.tahunMasuk = tahunMasuk;
        this.nilaiUTS = nilaiUTS;
        this.nilaiUAS = nilaiUAS;
    }
}

public class RataRataUAS15 {

    public static double hitungRataUAS(Mahasiswa[] mhs) {
        int total = 0;
        for (Mahasiswa mahasiswa : mhs) {
            total += mahasiswa.nilaiUAS;
        }
        return (double) total / mhs.length;
    }

    public static void main(String[] args) {
        Mahasiswa[] mahasiswaList = {
            new Mahasiswa("Ahmad", "220101001", 2022, 78, 82),
            new Mahasiswa("Budi", "220101002", 2022, 85, 88),
            new Mahasiswa("Cindy", "220101003", 2021, 90, 87),
            new Mahasiswa("Dian", "220101004", 2021, 76, 79),
            new Mahasiswa("Eko", "220101005", 2023, 92, 95),
            new Mahasiswa("Fajar", "220101006", 2020, 88, 85),

```

```
        new Mahasiswa("Gina", "220101007", 2023, 80, 83),  
        new Mahasiswa("Hadi", "220101008", 2020, 82, 84)  
    };  
  
    double rataUAS = hitungRataUAS(mahasiswaList);  
    System.out.println("Rata-rata nilai UAS: " + rataUAS);  
}  
}
```