

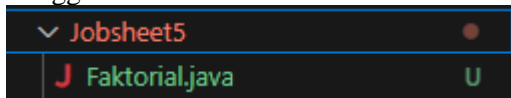
LAPORAN HASIL PRAKTIKUM
ALGORITMA DAN STRUKTUR DATA
JOBSHEET 5



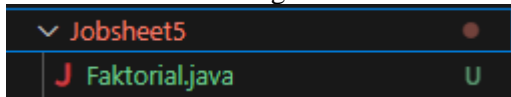
RIFO ANGGI BARBARA DANUARTA
244107020063
TI_1E
PROGRAM STUDI D_IV TEKNIK INFORMATIKA
JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG

Percobaan 1

1. Buat Project baru, dengan nama “BruteForceDivideConquer”. Buat package dengan nama minggu5.



2. Buatlah class baru dengan nama Faktorial



3. Lengkapi class Faktorial dengan atribut dan method yang telah digambarkan di dalam diagram class di atas, sebagai berikut:
 - a) Tambahkan method faktorialBF():

```
int faktorialBF(int n) {  
    int fakto = 1;  
    for (int i = 1; i <= n; i++) {  
        fakto = fakto * i;  
    }  
}
```

- b) Tambahkan method faktorialDC():

```
int faktorialDC (int n) {  
    if (n == 1) {  
        return 1;  
    } else {  
        int fakto = n * faktorialDC(n-1);  
        return fakto;  
    }  
}
```

4. Coba jalankan (Run) class Faktorial dengan membuat class baru MainFaktorial.
 - a) Di dalam fungsi main sediakan komunikasi dengan user untuk memasukkan nilai yang akan dicari faktorialnya

```
Scanner input = new Scanner (System.in);  
System.out.print("Masukan nilai: ");  
int nilai = input.nextInt();
```

- b) Kemudian buat objek dari class Faktorial dan tampilkan hasil pemanggilan method

```
Faktorial fk = new Faktorial();  
System.out.println("Nilai faktorial " + nilai +  
"menggunakan BF: " + fk.faktorialBF(nilai));  
System.out.println("Nilai faktorial " + nilai +  
"menggunakan BF: " + fk.faktorialDC(nilai));
```

- c) Pastikan program sudah berjalan dengan baik!

```

PS E:\Semester 2\Algoritma dan Struktur Data\Praktikum-ASD> &
\jdk-23\bin\java.exe' '--enable-preview' '-XX:+ShowCodeDetails
cp' 'C:\Users\THINKPAD\AppData\Roaming\Code\User\workspaceStorage
00ef0ddb9ec00\redhat.java\jdt_ws\Praktikum-ASD_3c4ba0fc\bin' '.
'

Masukan nilai: 5
Nilai faktorial 5 menggunakan BF: 120
Nilai faktorial 5 menggunakan BF: 120
PS E:\Semester 2\Algoritma dan Struktur Data\Praktikum-ASD>

```

Pertanyaan

1. Pada base line Algoritma Divide Conquer untuk melakukan pencarian nilai faktorial, jelaskan perbedaan bagian kode pada penggunaan if dan else!
2. Apakah memungkinkan perulangan pada method faktorialBF() diubah selain menggunakan for? Buktikan!
3. Jelaskan perbedaan antara fakto *= i; dan int fakto = n * faktorialDC(n-1); !
4. Buat Kesimpulan tentang perbedaan cara kerja method faktorialBF() dan faktorialDC()!

Jawaban

1. if = Menghentikan rekursi ketika mencapai 1, karena $1! = 1$
else = Memecah masalah faktorial menjadi submasalah lebih kecil (rekursi), hingga mencapai base case
2. bisa yaitu menggunakan perulangan while dan do-while

```

3 public class Faktorial {
4     public void faktorialBF() {
5
6     }
7     int faktorialBF(int n) {
8         int fakto = 1;
9         int i = 1;
10        while (i <= n) {
11            fakto *= i;
12            i++;
13        }
14        return fakto;
15    }
16
17    public void faktorialDC() {
18
19    }
20    int faktorialDC (int n) {
21        if (n == 1) {
22            return 1;

```

PROBLEMS 22 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS E:\Semester 2\Algoritma dan Struktur Data\Praktikum-ASD> &
\THINKPAD\AppData\Roaming\Code\User\workspaceStorage\5a9b874ee7
Masukan nilai: 5
Nilai faktorial 5 menggunakan BF: 120
Nilai faktorial 5 menggunakan BF: 120
PS E:\Semester 2\Algoritma dan Struktur Data\Praktikum-ASD>

```

3. fakto *= i;

Digunakan dalam perulangan (loop)

Iteratif: dilakukan langkah demi Langkah

Memperbarui nilai fakto di setiap putaran

Tidak butuh base case

int fakto = n * faktorialDC(n - 1);

Digunakan dalam rekursi (pemanggilan fungsi diri sendiri)

Rekursif: memecah masalah menjadi sub-masalah

Menghitung faktorial dengan pemanggilan diri sendiri

Butuh base case untuk menghentikan rekursi (biasanya if (n == 1))

4. faktorialBF()

Menghitung faktorial dengan mengalikan angka secara berurutan dari 1 sampai n menggunakan loop (for, while, do-while).

faktorialDC()

Menghitung faktorial dengan memecah masalah menjadi lebih kecil ($n * (n-1)!$) hingga mencapai basis ($n == 1$).

Percobaan 2

1. Di dalam paket minggu5, buatlah class baru dengan nama Pangkat. Dan di dalam class Pangkat tersebut, buat atribut angka yang akan dipangkatkan sekaligus dengan angka pangkatnya

```
int nilai, pangkat;
```

2. Tambahkan konstruktor berparameter

```
Pangkat(int n, int p) {  
    nilai = n;  
    pangkat = p;  
}
```

3. Pada class Pangkat tersebut, tambahkan method PangkatBF()

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

4. 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));  
        }  
    }  
}
```

5. Perhatikan apakah sudah tidak ada kesalahan yang muncul dalam pembuatan class Pangkat
6. 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("Masukan jumlah elemen: ");  
int elemen = input.nextInt();
```

7. 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.

```
Pangkat[] png = new Pangkat[elemen];
for (int i = 0; i < elemen; i++) {
    System.out.print("Masukan nilai basis elemen ke-" +
(i + 1) + ": ");
    int basis = input.nextInt();
    System.out.print("Masukan nilai pangkat elemen ke-"
+ (i + 1) + ": ");
    int pangkat = input.nextInt();
    png[i] = new Pangkat(basis, pangkat);
}
```

8. Kemudian, panggil hasilnya dengan mengeluarkan return value dari method PangkatBF() dan PangkatDC().

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

9. Pastikan output yang ditampilkan sudah benar seperti di bawah ini.

```
PS E:\Semester 2\Algoritma dan Struktur Data\Praktikum-ASD> &
'jdk-23\bin\java.exe' '--enable-preview' '-XX:+ShowCodeDetails
cp' 'C:\Users\THINKPAD\AppData\Roaming\Code\User\workspaceStorage
00ef0ddb9ec00\redhat.java\jdt_ws\Praktikum-ASD_3c4ba0fc\bin'
Masukan jumlah elemen: 3
Masukan nilai basis elemen ke-1: 2
Masukan nilai pangkat elemen ke-1: 3
Masukan nilai basis elemen ke-2: 4
Masukan nilai pangkat elemen ke-2: 5
Masukan nilai basis elemen ke-3: 6
Masukan nilai pangkat elemen ke-3: 7
HASIL PANGKAT BRUTEFORCE:
2^3: 8
4^5: 1024
6^7: 279936
HASIL PANGKAT DIVEDE AND CONQUER:
2^3: 8
4^5: 1024
6^7: 279936
PS E:\Semester 2\Algoritma dan Struktur Data\Praktikum-ASD>
```

Pertanyaan

1. Jelaskan mengenai perbedaan 2 method yang dibuat yaitu pangkatBF() dan pangkatDC()!
2. Apakah tahap combine sudah termasuk dalam kode tersebut? Tunjukkan!
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?
4. Tarik tentang cara kerja method pangkatBF() dan pangkatDC()!

Jawaban

1. pangkatBF = Melakukan perkalian berulang sebanyak n kali
pangkatDC = Membagi pangkat menjadi setengahnya tiap langkah
2. ya, karena dalam kode tersebut hasil dari dua panggilan rekursif dikalikan Kembali Menyusun Kembali hasil sub masalah menjadi solusi lengkap.

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

3. Method pangkatBF() tetap relevan meski memiliki parameter sendiri, akan tetapi nilai yang diinputkan pada parameter harus sama dengan nilai pada atribut nilai dan pangkat agar bisa tetap relevan. Jadi menurut saya menurut saya method tersebut kurang relevan untuk memiliki parameter, karena diperlukan kondisi tertentu agar method tersebut dikatakan cukup relevan. Method pangkatBF() dapat atau lebih baik dibuat tanpa parameter
4. pangkatBF() melakukan proses perhitungan pangkat nilai a dikalikan dengan nilai a hingga n kali secara iteratif. Sedangkan pangkatDC() melakukan proses perhitungan pangkat n nilai a dengan metode divide and conquer, yang dimana membagi proses menghitung pangkat menjadi lebih kecil, dan menyatukannya kembali.

Percobaan 3

1. Pada paket minggu5. Buat class baru yaitu class Sum. Tambahkan pula konstruktor pada class Sum.

```
double keuntungan[];

Sum(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

```
public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    System.out.print("Masukan Jumlah Elemen: ");
    int elemen = input.nextInt();
}
```

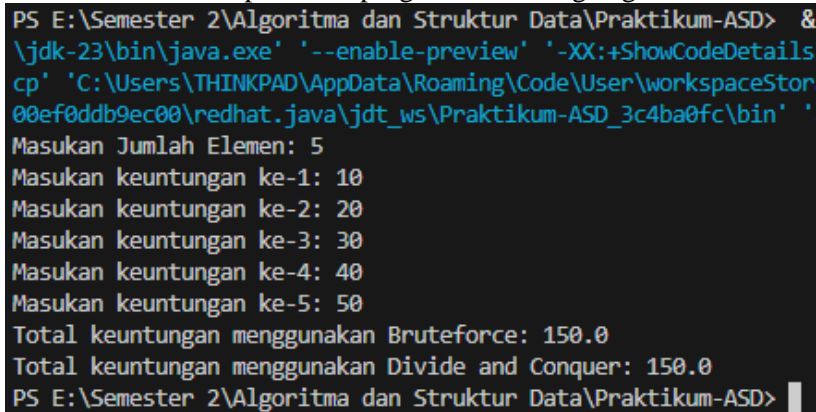
5. Karena yang akan dihitung adalah total nilai keuntungan, maka ditambahkan pula pada method main mana array yang akan dihitung. Array tersebut merupakan atribut yang terdapat di class Sum, maka dari itu dibutuhkan pembuatan objek Sum terlebih dahulu.

```
Sum sm = new Sum(elemen);
for (int i = 0; i < elemen; i++) {
    System.out.print("Masukan 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  
Bruteforce: " + sm.totalBF());  
System.out.println("Total keuntungan menggunakan Divide  
and Conquer: " + sm.totalDC(sm.keuntungan,0 ,elemen-1));
```

7. Cocokkan hasil compile kode program anda dengan gambar berikut ini.



```
PS E:\Semester 2\Algoritma dan Struktur Data\Praktikum-ASD> &  
jdk-23\bin\java.exe' --enable-preview' -XX:+ShowCodeDetails  
cp' 'C:\Users\THINKPAD\AppData\Roaming\Code\User\workspaceStor  
00ef0ddb9ec00\redhat.java\jdt_ws\Praktikum-ASD_3c4ba0fc\bin' '  
Masukan Jumlah Elemen: 5  
Masukan keuntungan ke-1: 10  
Masukan keuntungan ke-2: 20  
Masukan keuntungan ke-3: 30  
Masukan keuntungan ke-4: 40  
Masukan keuntungan ke-5: 50  
Total keuntungan menggunakan Bruteforce: 150.0  
Total keuntungan menggunakan Divide and Conquer: 150.0  
PS E:\Semester 2\Algoritma dan Struktur Data\Praktikum-ASD>
```

Pertanyaan

1. Kenapa dibutuhkan variable mid pada method TotalDC()?
2. Untuk apakah statement di bawah ini dilakukan dalam TotalDC()?
3. Kenapa diperlukan penjumlahan hasil lsum dan rsum seperti di bawah ini?
4. Apakah base case dari totalDC()?
5. Tarik Kesimpulan tentang cara kerja totalDC()

Jawaban

1. Variabel mid diperlukan pada method TotalDC(), variabel mid digunakan untuk mengetahui indeks titik tengah (median) dari atribut array keuntungan yang akan digunakan sebagai pembatas dari perhitungan kiri (leftsum) dan sebagai awalan dari perhitungan kanan (rightsum)
2. Statement `-- double lsum = totalDC(arr,l,mid);` berfungsi untuk menghitung penjumlahan array pada indeks 0 – mid (indeks tengah pada array) atau proses penjumlahan pada sisi kiri array secara rekursif. `double rsum = totalDC(arr,mid+1,r);` sedangkan rsum berfungsi untuk menghitung proses penjumlahan array pada indeks tengah (mid+1) hingga batas indeks dari array atau bisa dibilang proses penjumlahan pada sisi kanan array secara rekursif.
3. Penjumlahan hasil lsum dan rsum dilakukan untuk menggabungkan solusi dari permasalahan yang telah dipecah. Hasil lsum merupakan solusi dari sub masalah pertama (perhitungan dari indeks 0 sampai median) dan hasil rsum merupakan solusi dari sub masalah kedua (perhitungan dari tengah hingga indeks terakhir). Untuk menemukan solusi akhir maka diperlukan untuk menjumlahkan lsum dan rsum.
4. Base case dari method totalDC yaitu jika posisi dari indeks l dan indeks r bernilai sama maka akan mengembalikan nilai array pada indeks l.
5. Method totalDC() merupakan method untuk mengetahui jumlah dari seluruh elemen pada array dengan metode divide and conquer. Pada method totalDC solusi untuk menghitung jumlah seluruh elemen pada array dilakukan dengan membagi proses perhitungan menjadi dua bagian, yaitu sisi kiri dan sisi kanan hingga mencapai basis rekursi, yaitu hingga posisi indeks kiri dan kanan bernilai sama sehingga menunjuk elemen pada indeks yang sama. Setelah itu menemukan hasil dari sisi kiri dan kanan, maka return penjumlahan dari hasil sisi kiri dan sisi kanan untuk mendapatkan hasil akhirnya.