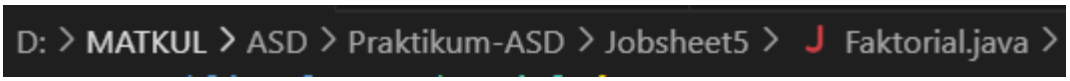
Nama : Dicky Darmawan
Kelas : TI 1B
NIM : 244107020037

Percobaan 1: Menghitung Nilai Faktorial dengan Algoritma Brute Force dan Divide and Conquer

Langkah-langkah:

1. Buat folder baru bernama Jobsheet5 di dalam repository Praktikum ASD
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;
    }
    return fakto;
}
```

    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

```
Run main | Debug main | Run | Debug
public static void main(String[] args) {
    Scanner input = new Scanner (System.in);
    System.out.print(s:"Masukkan nilai: ");
    int nilai = input.nextInt();
```

b) Kemudian buat objek dari class Faktorial dan tampilkan hasil pemanggilan method faktorialDC() dan faktorialBF()

```
Faktorial fk = new Faktorial();
System.out.println("Nilai faktorial " + nilai
System.out.println("Nilai faktorial " + nilai
```

c) Pastikan program sudah berjalan dengan baik!

Verifikasi Hasil Percobaan

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

Pertanyaan

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

   Pada if (base case) jika n bernilai 1 maka akan langsung mengembalikan 1 tanpa melakukan rekursif, pada else jika n lebih besar dari 1 maka akan melakukan rekursif n-1 hingga mencapai batas

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

   Bisa

```
int faktorialBF(int n) {
    int fakto = 1;
    int i = 1;
    while (i <= n) {
        fakto *= i;
    }
    return fakto;
}
```

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

   Fakto *= I operasi yang memperbarui nilai fakto selama iterasi berjalan

   fakto = n * faktorialDC(n-1) melakukan rekursif hingga mencapai base case

4. Buat Kesimpulan tentang perbedaan cara kerja method faktorialBF() dan faktorialDC()!
    - faktorialBF():
        a. Iterasi menggunakan perulangan
        b. Terkadang lebih mudah untuk dipahami prosesnya
        c. Lebih cepat dan efisien
    - faktorialDC():
        a. Bersifat rekursif yang memanggil dirinya sendiri
        b. Prosesnya lebih kompleks
        c. Lebih lambar karena melakukan banyak pemanggilan

Percobaan 2: Menghitung Hasil Pangkat dengan Algoritma Brute Force dan Divide and Conquer

Langkah-langkah:

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

```
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.

```
iblic static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    System.out.print(s:"Masukkan 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("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 Pangkat(basis, pangkat);
}
```

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

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

Verifikasi Hasil Percobaan

```
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()!

   pangkatBF() menggunakan perulangan untuk mengalikan a sebanyak n, sedangkan pangkatDC() menggunaakn rekursif

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

   Iya pada metode pangkatDC()

```
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. Pada method pangkatBF()terdapat parameter untuk melewatkan 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?

   Dengan menggunakan parameter akan lebih fleksibel karena bisa digunakan untuk menghitung pangkat angka lain tanpa membuat objek baru. Bisa juga tanpa menggunakan parameter tetapi harus mengubah pemanggilan pangkatBF() di main

```
int pangkatBF() {
    int hasil = 1;
    for (int i = 0; i < pangkat; i++) {
        hasil *= nilai;
    }
    return hasil;
}
```

```
for (Pangkat p : png) {
    System.out.println(p.nilai + "^" + p.pangkat + ": " + p.pangkatBF());
}
```

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

   pangkatBF()menggunakan pendekatan iteratif sehingga kurang efisien untuk nilai pangkat yang besar.

   pangkatDC()) menggunakan pendekatan rekursif yang lebih optimal karena mengurangi jumlah perkalian secara signifikan.

Percobaan 3: Menghitung Sum Array dengan Algoritma Brute Force dan Divide and Conquer

Langkah-langkah:

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

```
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

```
Scanner input = new Scanner (System.in);

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

5. Buat objek dari class Sum. Lakukan perulangan untuk mengambil input nilai keuntungan dan masukkan ke atribut keuntungan dari objek yang baru dibuat tersebut!

```
Sum sm = new Sum(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 Bruteforce: " +sm
System.out.println("Total keuntungan menggunakan Divide and Conque
```

Verifikasi Hasil Percobaan

Pertanyaan

1. Kenapa dibutuhkan variable mid pada method TotalDC()?

   Variabel mid digunakan untuk membagi array menjadi dua bagian pada setiap pemanggilan rekursif. Tanpa variabel mid, program tidak bisa membagi array menjadi lebih kecil sehingga tidak dapat menggunakan Divide and Conquer
2. Untuk apakah statement di bawah ini dilakukan dalam TotalDC()?

   lsum = totalDC(arr, l, mid) menghitung total keuntungan pada bagian kiri dari array.

   rsum = totalDC(arr, mid+1, r) menghitung total keuntungan pada bagian kanan dari array
3. Kenapa diperlukan penjumlahan hasil lsum dan rsum seperti di bawah ini?

   untuk menggabungkan hasil dari kedua bagian array yang telah dihitung sebelumnya
4. Apakah base case dari totalDC()?

   ketika hanya ada satu elemen dalam array (l == r), pada kondisi ini langsung mengembalikan nilai elemen tersebut, karena tidak ada lagi yang harus dihitung atau dibagi
5. Tarik Kesimpulan tentang cara kerja totalDC()

   totalDC() adalah algoritma rekursif yang menggunakan Divide and Conquer untuk menghitung total keuntungan.

   Divide: memisahkan array dibagi menjadi dua bagian menggunakan variabel mid.

   Conquer: menyelesaikan dengan setiap bagian dihitung secara rekursif menggunakan totalDC().

   Combine: menggabungkan hasil dari dua bagian array dengan menjumlahkan (lsum + rsum).