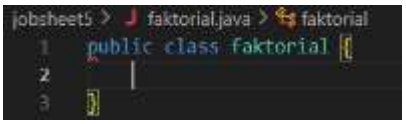
Nama : Siti Nikmatus Sholihah

NIM : 244107020014

Kelas : TI 1B

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

1. Buat folder baru dengan nama jobsheet5 didalam folder Praktikum-ASD
2. Buatlah class baru dengan nama factorial



3. Lengkapi class faktorial dengan atribut dan method yang telah digambarkan didalam diagram class diatas, menggunakan faktorialBF dan faktorialDC.

4. Coba jalankan (run) class faktorial dengan membuat class baru MainFaktorial.

   a) Didalam fungsi main sediakan komunikasi dengan user untuk memasukkan nilai yang akan dicari faktorialnya.

   b) Kemudian buat objek dari class Faktorial dan tampilkan hasil pemanggilan method FaktorialDC() dan FaktorialBF()

c) Pastikan program berjalan dengan baik

5. Verifikasi hasil percobaan

Pertanyaan Percobaan 1

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

   Jawab :

   - Pada bagian if, kondisi n == 1 adalah **base case** yang menghentikan rekursi dan mengembalikan nilai 1.
   - Pada bagian else, rekursi terjadi dengan memanggil fungsi faktorialDC(n-1)

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

   Jawab : Ya, perulangan pada method faktorialBF() bisa diubah menggunakan perulangan while atau do-while. Berikut contohnya menggunakan while:

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

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

   Jawab :

- fakto *= i; adalah operasi iteratif yang dilakukan dalam perulangan untuk menghitung faktorial secara langsung dengan mengalikan
  nilai fakto dengan i pada setiap iterasi.
- int fakto = n * faktorialDC(n-1); adalah operasi rekursif di mana faktorial dihitung dengan memanggil fungsi itu sendiri (faktorialDC(n-1)) dan menggabungkan hasilnya dengan mengalikan n dengan hasil rekursi tersebut.

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

   Jawab :

   - **faktorialBF()** menggunakan pendekatan iteratif dengan perulangan untuk menghitung faktorial secara langsung
   - **faktorialDC()** menggunakan pendekatan rekursif dengan membagi masalah menjadi sub-masalah yang lebih kecil (Divide and Conquer).

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

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

```
jobsheet5 > J PangkatJava > Pangkat
1  public class Pangkat{
2      int nilai, pangkat;
3
4  }
```

2. Tambahkan konstruktor berparameter

```
jobsheet5 > J Pangkat.java > Pangkat
1  public class Pangkat{
2      int nilai, pangkat;
3
4      Pangkat (int n, int p){
5          nilai = n;
6          pangkat = p;
7      }
8  }
```

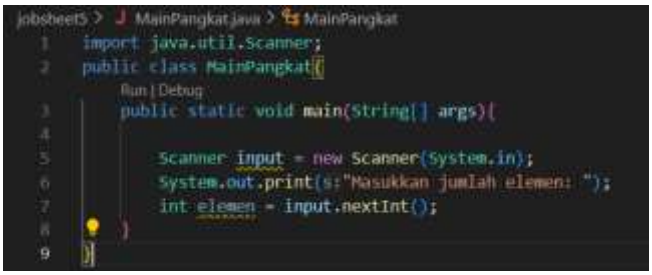
3. Pada class Pangkat tersebut, tambahkan method PangkatBF()

```
jobsheet5 > J Pangkat.java > Pangkat
1  public class Pangkat{
2      int nilai, pangkat;
3
4      Pangkat (int n, int p){
5          nilai = n;
6          pangkat = p;
7      }
8
9      int pangkatBF(int a, int n){
10         int hasil = 1;
11         for (int i = 1; i <= n; i++) {
12             hasil = hasil*a;
13         }
14         return hasil;
15     }
16 }
```

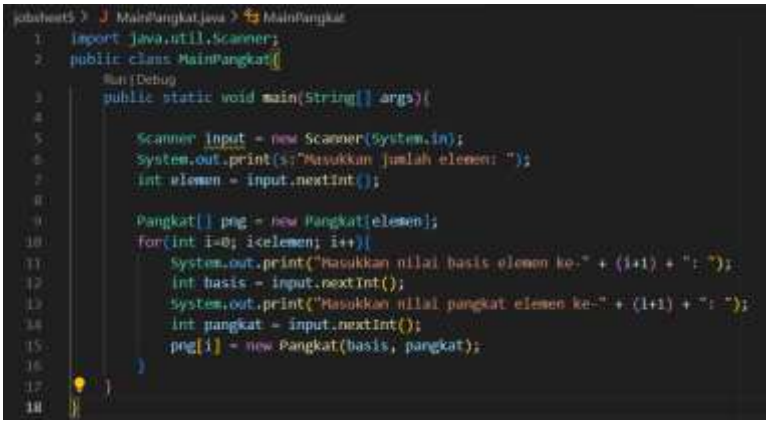
4. Pada class Pangkat juga tambahkan method PangkatDC()

```
jobsheet5 > J Pangkat.java > Pangkat > pangkatDC(int, int)
1  public class Pangkat{
2      int nilai, pangkat;
3
4      Pangkat (int n, int p){
5          nilai = n;
6          pangkat = p;
7      }
8
9      int pangkatBF(int a, int n){
10         int hasil = 1;
11         for (int i = 1; i <= n; i++) {
12             hasil = hasil*a;
13         }
14         return hasil;
15     }
16
17     int pangkatDC(int a, int n){
18         if (n==1 ) {
19             return a;
20         } else {
21             if (n%2==1) {
22                 return(pangkatDC(a, n/2)*pangkatDC(a, n/2)*a);
23             } else {
24                 return(pangkatDC(a, n/2)*pangkatDC(a, n/2));
25             }
26         }
27     }
28 }
```

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.



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.



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

```
jobsheet5 > J MainPangkat.java > MainPangkat > main(String[])
1   import java.util.Scanner;
2   public class MainPangkat{
        Run | Debug
3       public static void main(String[] args){
4
5           Scanner input = new Scanner(System.in);
6           System.out.print(s:"Masukkan jumlah elemen: ");
7           int elemen = input.nextInt();
8
9           Pangkat[] png = new Pangkat[elemen];
10          for(int i=0; i<elemen; i++){
11              System.out.print("Masukkan nilai basis elemen ke-" + (i+1) + ": ");
12              int basis = input.nextInt();
13              System.out.print("Masukkan nilai pangkat elemen ke-" + (i+1) + ": ");
14              int pangkat = input.nextInt();
15              png[i] = new Pangkat(basis, pangkat);
16          }
17
18          System.out.println(x:"HASIL PANGKAT BRUTH FORCE:");
19          for (Pangkat p : png) {
20              System.out.println(p.nilai+"^"+p.pangkat+": "+p.pangkatBF(p.nilai, p.pangkat));
21          }
22          System.out.println(x:"HASIL PANGKAT DIVINE CONQUER:");
23          for (Pangkat p : png) {
24              System.out.println(p.nilai+"^"+p.pangkat+": "+p.pangkatDC(p.nilai, p.pangkat));
25          }
26      }
27  }
```

9. Verifikasi Hasil Percobaan

```
d:\Praktikum-ASD\jobsheet5>cd "d:\Praktikum-ASD\jobsheet5\"
Picked up JAVA_TOOL_OPTIONS: -Dstdout.encoding=UTF-8 -Dstde
Picked up JAVA_TOOL_OPTIONS: -Dstdout.encoding=UTF-8 -Dstde
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 BRUTH FORCE:
2^3: 8
4^5: 1024
6^7: 279936
HASIL PANGKAT DIVINE 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 pendekatan iteratif dengan perulangan untuk menghitung pangkat secara langsung.
   - **pangkatDC()** menggunakan pendekatan rekursif dengan membagi masalah menjadi sub-masalah yang lebih kecil (Divide and Conquer).
2. Apakah tahap *combine* sudah termasuk dalam kode tersebut? Tunjukkan!

- Jawab : Ya, tahap **combine** terdapat pada bagian:

```
return (pangkatDC(a, n/2) * pangkatDC(a, n/2) * a);
```

atau

```
return (pangkatDC(a, n/2) * pangkatDC(a, n/2));
```

Di sini, hasil dari dua pemanggilan rekursif digabungkan dengan operasi perkalian.

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?

   Jawab : Parameter pada method pangkatBF() tetap relevan karena memungkinkan fleksibilitas dalam menghitung pangkat untuk nilai dan pangkat yang berbeda. Namun, jika method tersebut dibuat tanpa parameter, kita bisa menggunakan atribut nilai dan pangkat yang sudah ada di class Pangkat. Contohnya:

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

4. Tarik tentang cara kerja method pangkatBF() dan pangkatDC()!
   Jawab :
   - **pangkatBF()** menggunakan pendekatan iteratif dengan perulangan untuk menghitung pangkat secara langsung.
   - **pangkatDC()** menggunakan pendekatan rekursif dengan membagi masalah menjadi sub-masalah yang lebih kecil (Divide and Conquer).

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

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

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

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

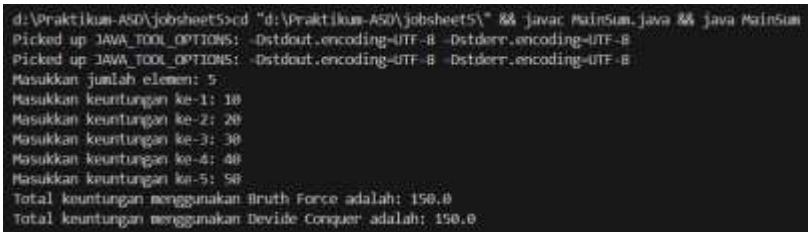
4. Buat class baru yaitu MainSum. Di dalam kelas ini terdapat method main. Pada method ini user dapat menuliskan berapa bulan keuntungan yang

akan dihitung. Dalam kelas ini sekaligus dibuat instansiasi objek untuk memanggil atribut ataupun fungsi pada class Sum

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

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

7. Verifikasi Hasil Percobaan



Pertanyaan

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

- Jawab : Variabel mid digunakan untuk membagi array menjadi dua bagian yang lebih kecil. Ini adalah bagian dari proses **Divide** dalam algoritma 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 : Statement tersebut digunakan untuk membagi array menjadi dua bagian dan menghitung jumlah masing-masing bagian secara rekursif. lsum adalah jumlah dari bagian kiri array, sedangkan rsum adalah jumlah dari bagian kanan array.

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

```
return lsum+rsum;
```

Jawab : Penjumlahan lsum dan rsum adalah bagian dari proses **Combine** dalam algoritma Divide and Conquer. Setelah masalah dibagi dan dihitung secara terpisah, hasilnya digabungkan untuk mendapatkan solusi akhir.

4. Apakah base case dari totalDC()?

- Jawab : Base case dari totalDC() adalah ketika l == r, yaitu ketika array hanya memiliki satu elemen. Pada kondisi ini, fungsi mengembalikan nilai elemen tersebut:

```
if (l == r) {
    return arr[l];
}
```

5. Tarik Kesimpulan tentang cara kerja totalDC()

Jawab : **totalDC()** menggunakan pendekatan Divide and Conquer dengan membagi array menjadi dua bagian, menghitung jumlah masing-masing bagian secara rekursif, dan menggabungkan hasilnya.