Nama : Adellia Salsa Al Barra
Nim : 244107020222
Kelas : TI 1B

PERCOBAAN 5.2.1

1. Buat class Faktorial
2. Melengkapi class dengan atribut dan method

Faktorial.java > Faktorial > faktorialDC(int)

```java
import java.util.Scanner;
public class Faktorial{
    public int nilai;

    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. Buat class baru dengan nama MainFaktorial
4. Di dalam fungsi main sediakan komunikasi dengan user untuk memasukkan nilai yang akan dicari faktorialnya

MainFaktorial.java

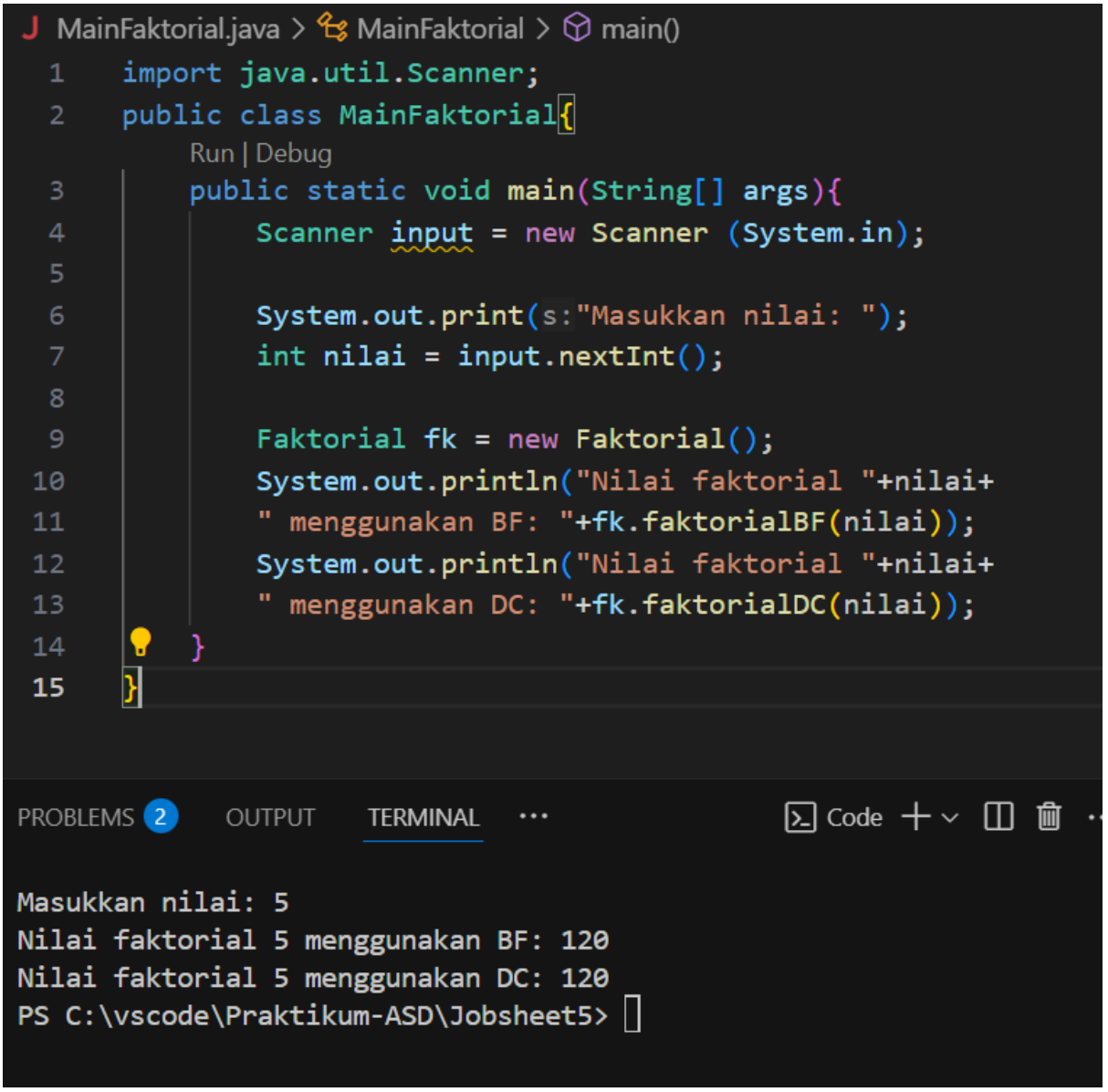
```java
import java.util.Scanner;
public class MainFaktorial{
    public static void main(String[]){
        Scanner input = new Scanner (System.in);

        System.out.print("Masukkan nilai: ");
        int nilai = input.nextInt();
    }
}
```

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

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

6. Hasil Run



PERTANYAAN

1. Pada base line Algoritma Divide Conquer untuk melakukan pencarian nilai faktorial, jelaskan perbedaan bagian kode pada penggunaan if dan else!
   if : Base Case => menghentikan rekursi (n == 1 → return 1)
   else : Recursive Case => memecah masalah (n * faktorialDC(n - 1))
2. Apakah memungkinkan perulangan pada method faktorialBF() diubah selain menggunakan for? Buktikan!
   Bisa diganti dengan while atau do-while

```java
public class Faktorial{
    public int nilai;

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

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

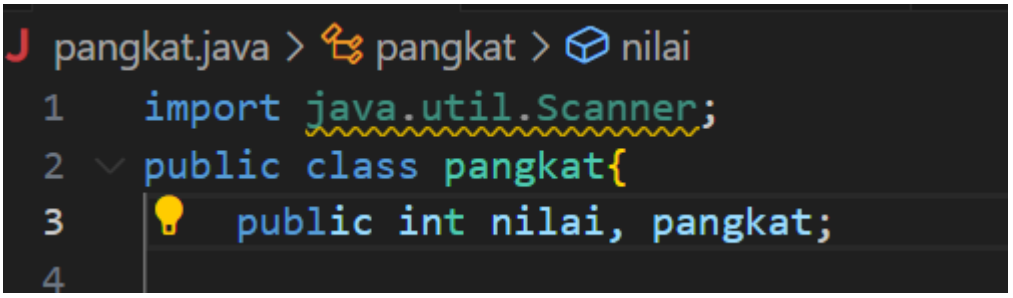
```
PROBLEMS 2   OUTPUT   TERMINAL   ...   Code

Masukkan nilai: 5
Nilai faktorial 5 menggunakan BF: 120
Nilai faktorial 5 menggunakan DC: 120
PS C:\vscode\Praktikum-ASD\Jobsheet5> cd "c:\vscode\Praktiku
" ; if ($?) { javac MainFaktorial.java } ; if ($?) { java Ma
Masukkan nilai: 5
Nilai faktorial 5 menggunakan BF: 120
Nilai faktorial 5 menggunakan DC: 120
PS C:\vscode\Praktikum-ASD\Jobsheet5>
```

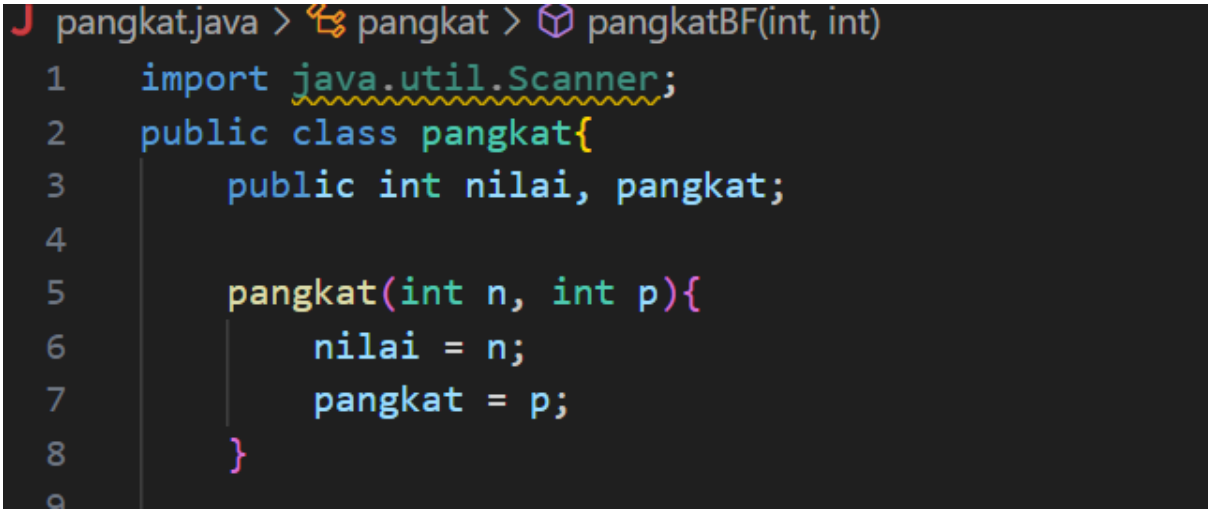
3. Jelaskan perbedaan antara fakto *= i; dan int fakto = n * faktorialDC(n-1); !
   fakto *= i; => digunakan dalam perulangan
   n * faktorialDC(n-1); => digunakan untuk memanggil dirinya sendiri
4. Buat Kesimpulan tentang perbedaan cara kerja method faktorialBF() dan faktorialDC()!
   faktorialBF() : menggunakan perulangan lebih cepat & hemat memori
   faktorialDC() : menggunakan rekursi, lebih ringkas tetapi membutuhkan lebih banyak memori

PERCOBAAN 5.3.1

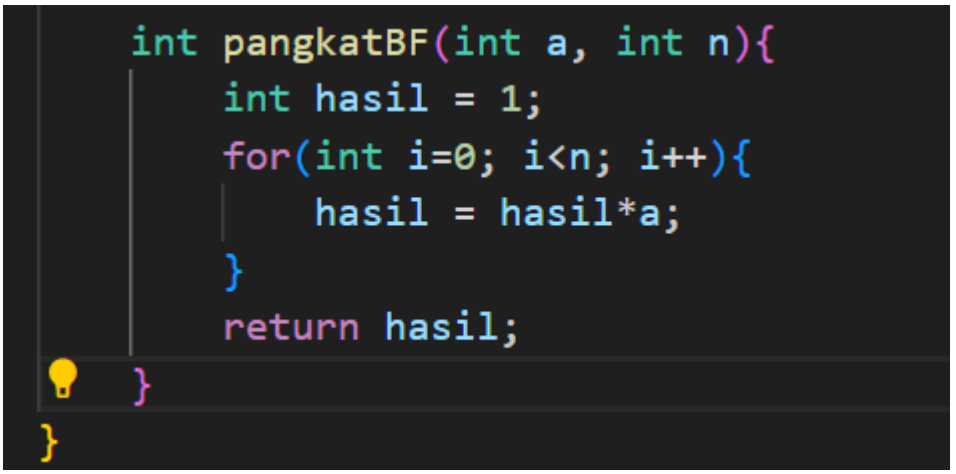
1. Buat class baru dengan nama pangkat
2. buat atribut angka yang akan dipangkatkan sekaligus dengan angka pemangkatnya



3. Menambahkan konstruktor berparameter



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



5. Pada class Pangkat juga tambahkan method PangkatDC()

```java
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));
        }
    }
}
```

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

pangkatMain.java

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

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

9. Run program

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

        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);
        }

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

```
PROBLEMS 4   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

PS C:\vscode\Praktikum-ASD\Jobsheet5> cd "c:\vscode\Praktikum-ASD\Jobsheet5\" ; if ($?) { javac pangkatMain.java } ; if ($?
Masukkan jumlah elemen: 3
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
PS C:\vscode\Praktikum-ASD\Jobsheet5>
```

PERTANYAAN

1. Jelaskan mengenai perbedaan 2 method yang dibuat yaitu pangkatBF() dan pangkatDC()!
   pangkatBF() : Iteratif, menggunakan for untuk perkalian berulang
   pangkatDC() : Rekursif, membagi masalah jadi lebih kecil

2. Apakah tahap combine sudah termasuk dalam kode tersebut? Tunjukkan!
   ya, sudah termasuk di dalam method 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?
   Tidak perlu karena atribut pangkat dan nilai sudah ada

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

4. Tarik tentang cara kerja method pangkatBF() dan pangkatDC()!
   pangkatBF() : cocok untuk nilai kecil
   pangkatDC(): lebih efisien untuk pangkat besar

PERCOBAAN 5.4.1

1. Buat class baru dengan nama Sum
2. Tambahkan konstruktor

```
1  import java.util.Scanner;
2  public class Sum {
3
4      double keuntungan[];
5      Sum(int el){
6          keuntungan = new double[el];
7      }
8
```

3. 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;
    }
```

4. 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;
    }
}
```

5. 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

```
MainSum.java > MainSum > main(String[])
1   import java.util.Scanner;
2   public class MainSum{
        Run | Debug
3       public static void main(String[] args){
4           Scanner input = new Scanner(System.in);
5           System.out.print(s:"Masukkan jumlah elemen: ");
6           int elemen = input.nextInt();
```

6. 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();
        }
```

7. 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.print("Total keuntungan menggunakan Divide and Conquer: "+sm.totalDC(sm.keuntungan,1:0,elemen-1));
    }
```

8. Run program

MainSum.java > MainSum

```
1   import java.util.Scanner;
2   public class MainSum{
        Run | Debug
3       public static void main(String[] args){
4           Scanner input = new Scanner(System.in);
5           System.out.print(s:"Masukkan jumlah elemen: ");
6           int elemen = input.nextInt();
7
8           Sum sm = new Sum(elemen);
9           for(int i=0;i<elemen;i++){
10              System.out.print("Masukkan keuntungan ke-"+(i+1)+": ");
11              sm.keuntungan[i] = input.nextDouble();
12          }
13
14          System.out.println("Total keuntungan menggunakan Bruteforce: "+sm.totalBF());
15          System.out.print("Total keuntungan menggunakan Divide and Conquer: "+sm.totalDC(sm.
16      }
17  }
```

PROBLEMS 6 OUTPUT DEBUG CONSOLE TERMINAL PORTS Code

```
PS C:\vscode\Praktikum-ASD\Jobsheet5> cd "c:\vscode\Praktikum-ASD\Jobsheet5\" ; if ($?) { javac Ma
} ; if ($?) { java MainSum }
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: 150.0
Total keuntungan menggunakan Divide and Conquer: 150.0
PS C:\vscode\Praktikum-ASD\Jobsheet5>
```

PERTANYAAN

1. Kenapa dibutuhkan variable mid pada method TotalDC()?
   Untuk membagi array menjadi dua bagian dalam Divide and Conquer
2. Untuk apakah statement di bawah ini dilakukan dalam TotalDC()?
   lsum = totalDC(arr, l, mid); => Menghitung total keuntungan dari bagian kiri array

rsum = totalDC(arr, mid+1, r); => Menghitung total keuntungan dari bagian kanan array

3. Kenapa diperlukan penjumlahan hasil lsum dan rsum seperti di bawah ini?
   Untuk menggabungkan hasil perhitungan dari dua bagian array agar mendapatkan total keseluruhan
4. Apakah base case dari totalDC()?
   Base case totalDC() terjadi saat hanya ada satu elemen (l == r)
5. Tarik Kesimpulan tentang cara kerja totalDC()
   totalDC() membagi array, menghitung rekursif, lalu menggabungkan hasilnya. Lebih efisien dari Brute Force