

Nama : Muhammad Zakaria Haniya

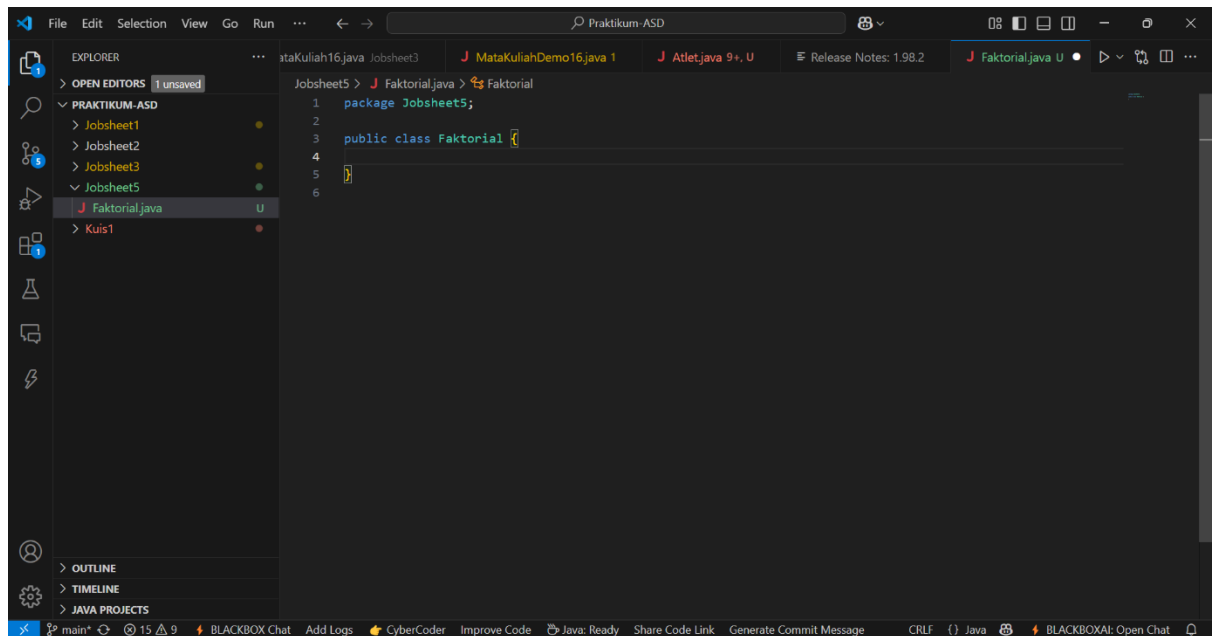
NIM : 244107020135

Kelas : 1B

Absen : 16

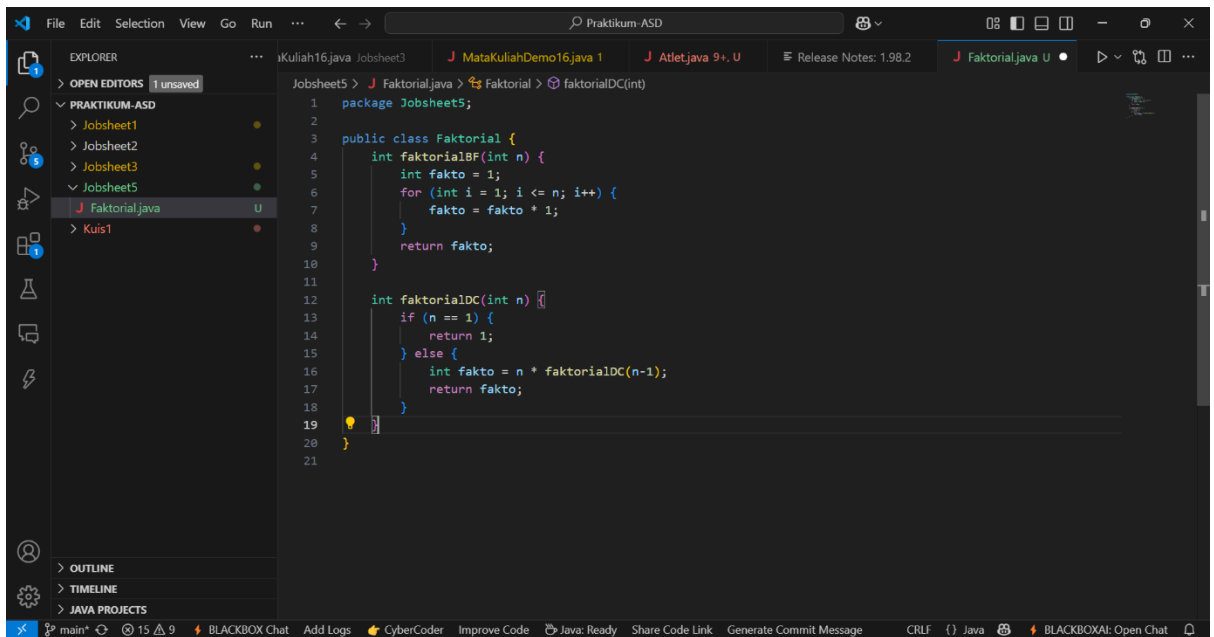
Menghitung Nilai Faktorial dengan Algoritma Brute Force dan Divide and Conquer

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():

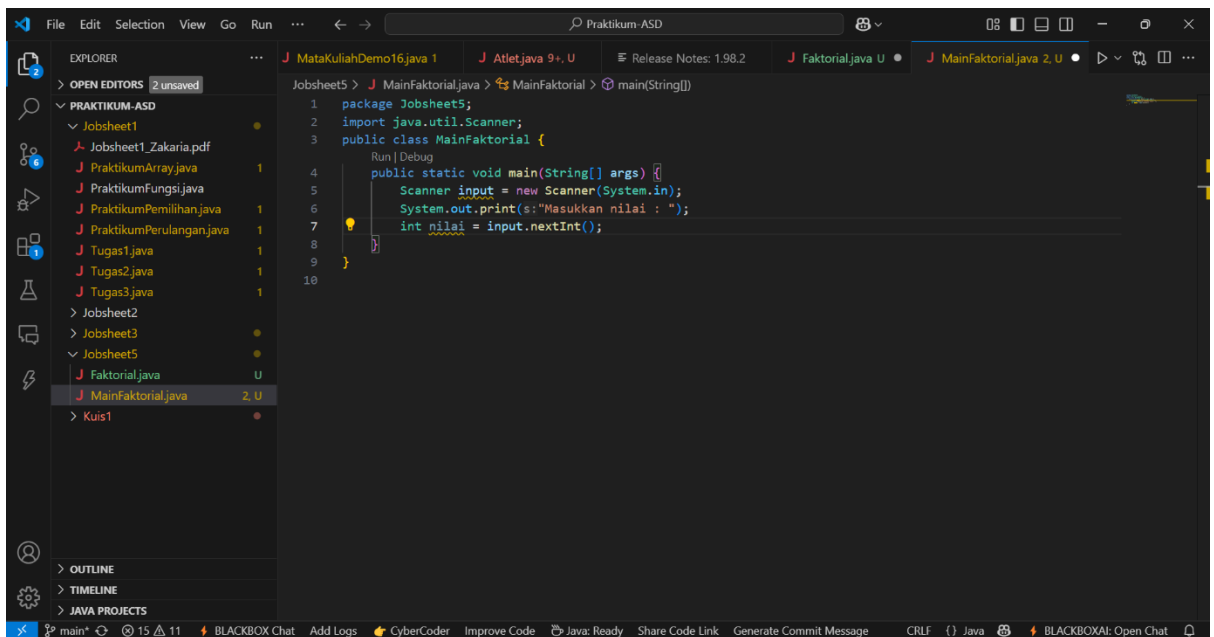
b) Tambahkan method faktorialDC():



```
1 package Jobsheet5;
2
3 public class Faktorial {
4     int faktorialBF(int n) {
5         int fakto = 1;
6         for (int i = 1; i <= n; i++) {
7             fakto = fakto * i;
8         }
9         return fakto;
10    }
11
12    int faktorialDC(int n) {
13        if (n == 1) {
14            return 1;
15        } else {
16            int fakto = n * faktorialDC(n-1);
17            return fakto;
18        }
19    }
20 }
21
```

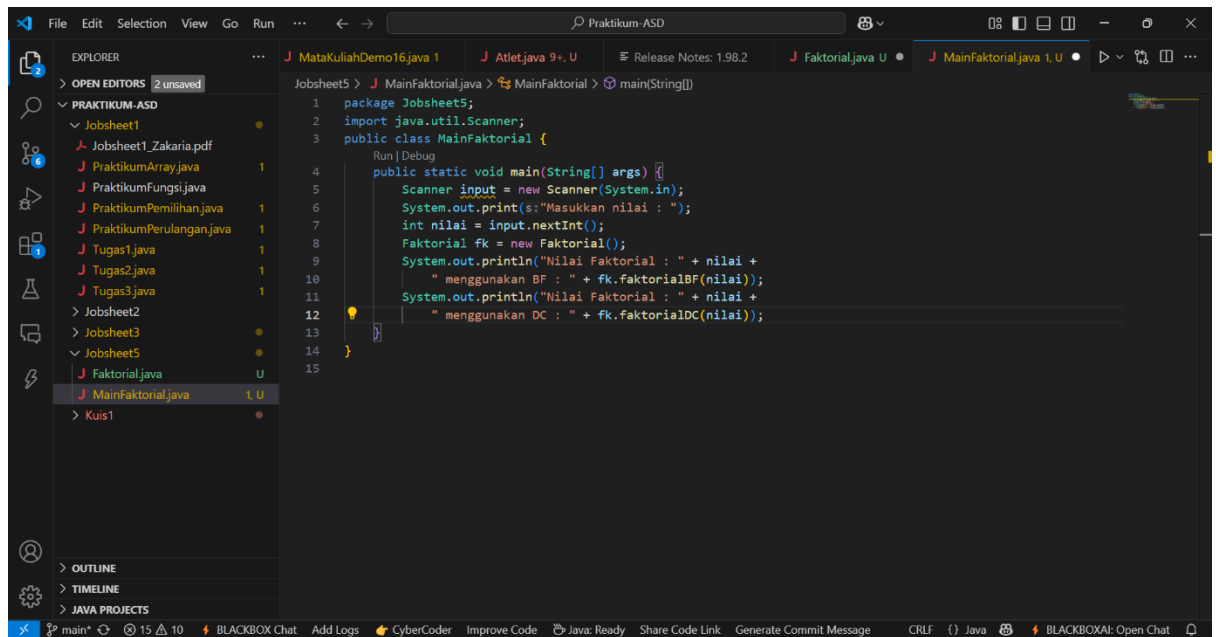
4. Coba jalankan (Run) class Faktorial dengan membuat class baru MainFaktorial.

a) Di dalam fungsi main sediakan komunikasi dengan user untuk memasukkan nilai yang akan dicari faktorialnya



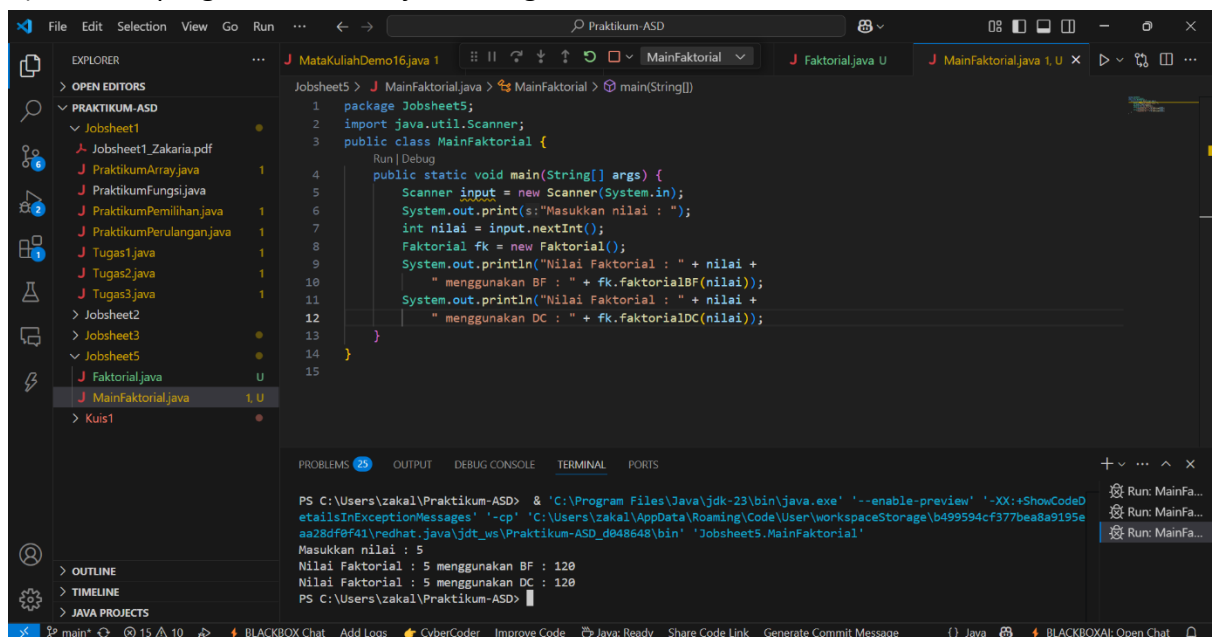
```
1 package Jobsheet5;
2 import java.util.Scanner;
3 public class MainFaktorial {
4     public static void main(String[] args) {
5         Scanner input = new Scanner(System.in);
6         System.out.print("Masukkan nilai : ");
7         int nilai = input.nextInt();
8     }
9 }
10
```

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



```
1 package Jobsheet5;
2 import java.util.Scanner;
3 public class MainFaktorial {
4     public static void main(String[] args) {
5         Scanner input = new Scanner(System.in);
6         System.out.print("Masukkan nilai : ");
7         int nilai = input.nextInt();
8         Faktorial fk = new Faktorial();
9         System.out.println("Nilai Faktorial : " + nilai +
10             " menggunakan BF : " + fk.faktorialBF(nilai));
11         System.out.println("Nilai Faktorial : " + nilai +
12             " menggunakan DC : " + fk.faktorialDC(nilai));
13     }
14 }
15
```

d) Pastikan program sudah berjalan dengan baik!

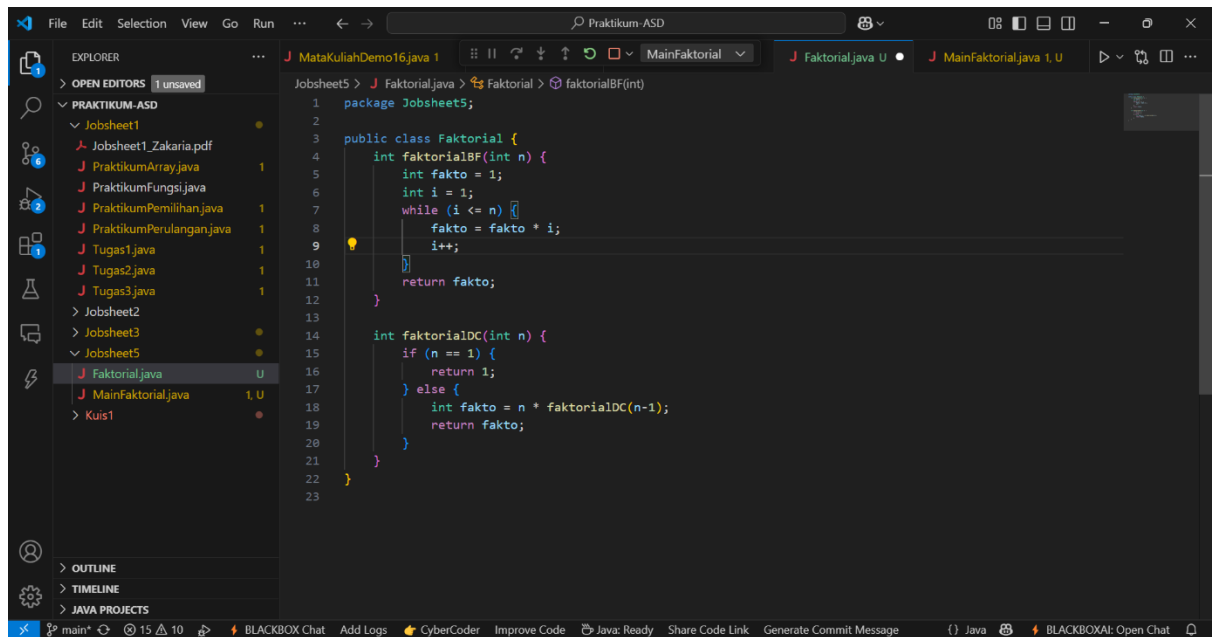


```
PS C:\Users\zakal\Praktikum-ASD> & 'C:\Program Files\Java\jdk-23\bin\java.exe' '-enable-preview' '-XX:+ShowCodeD
etailsInExceptionMessages' '-cp' 'C:\Users\zakal\AppData\Roaming\Code\User\workspaceStorage\b499594cf377bea8a9195e
aa28df0f41\redhat_java_jdt_ws\Praktikum-ASD_d048648\bin\' 'Jobsheet5.MainFaktorial'
Masukkan nilai : 5
Nilai Faktorial : 5 menggunakan BF : 120
Nilai Faktorial : 5 menggunakan DC : 120
PS C:\Users\zakal\Praktikum-ASD>
```

Pertanyaan :

1. Pada base line Algoritma Divide Conquer untuk melakukan pencarian nilai faktorial, jelaskan perbedaan bagian kode pada penggunaan if dan else!
 - if : Ini adalah base case atau kondisi berhenti dalam rekursi, Ketika n mencapai 1, fungsi akan mengembalikan nilai 1 tanpa memanggil dirinya sendiri lagi
 - else : Ini adalah recursive case, di mana fungsi akan terus memanggil dirinya sendiri dengan n-1, Setiap pemanggilan rekursif akan mengalikan n dengan hasil faktorial dari n-1 hingga mencapai base case
2. Apakah memungkinkan perulangan pada method faktorialBF() diubah selain menggunakan for? Buktikan!

- Bisa,

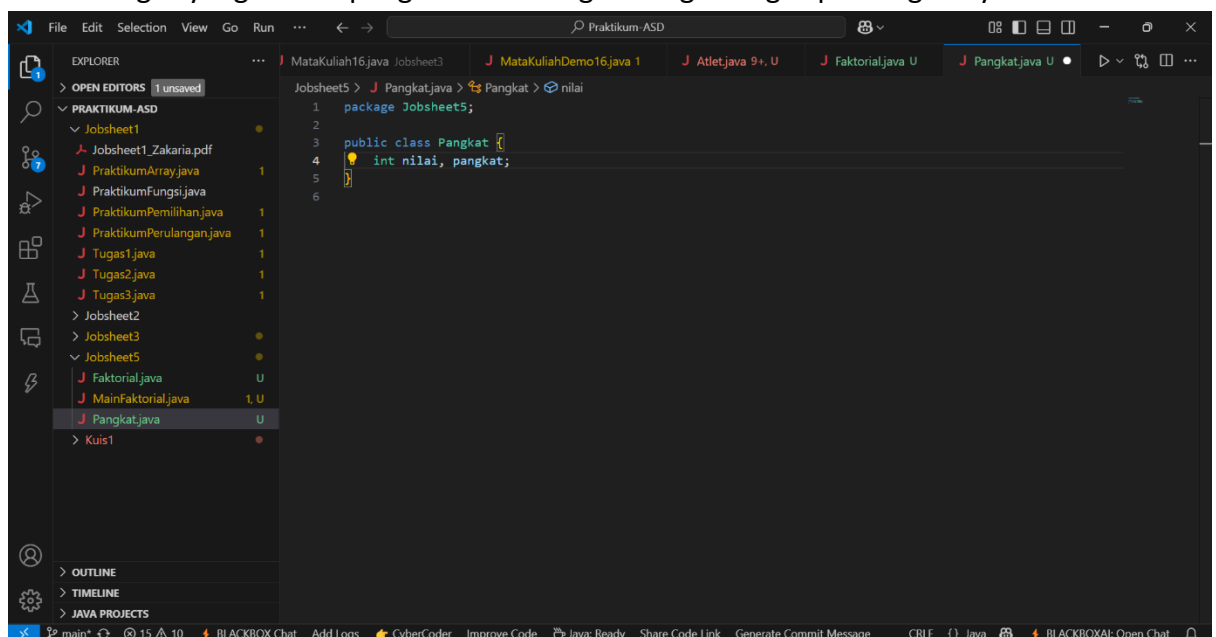


```
1 package Jobsheet5;
2
3 public class Faktorial {
4     int faktorialBF(int n) {
5         int fakto = 1;
6         int i = 1;
7         while (i <= n) {
8             fakto = fakto * i;
9             i++;
10        }
11        return fakto;
12    }
13
14    int faktorialDC(int n) {
15        if (n == 1) {
16            return 1;
17        } else {
18            int fakto = n * faktorialDC(n-1);
19            return fakto;
20        }
21    }
22 }
23
```

3. Jelaskan perbedaan antara fakto *= i; dan int fakto = n * faktorialDC(n-1); !
 - fakto *= i; : mengalikan dengan i dari 1 hingga sebelum n
 - fakto = n * faktorialDC(n-1); : mengalikan n dengan n-1 hingga 1
4. Buat Kesimpulan tentang perbedaan cara kerja method faktorialBF() dan faktorialDC()!
 - Method faktorialBF() menggunakan rekursif
 - Method faktorialDC() menggunakan iteratif

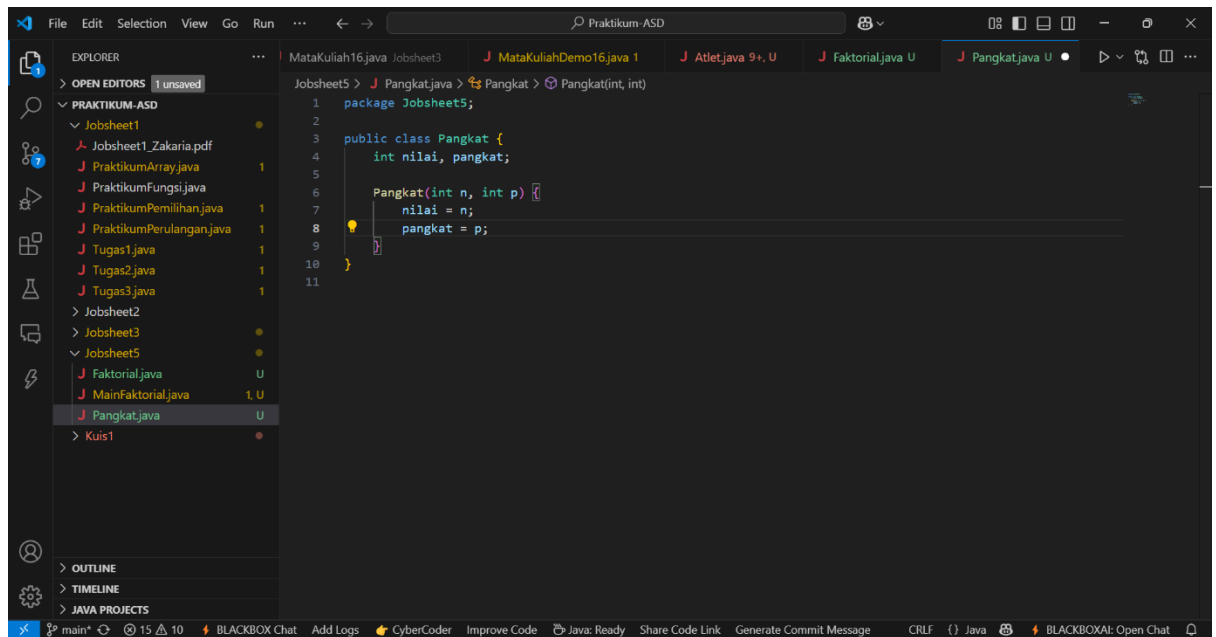
Menghitung Hasil Pangkat dengan Algoritma Brute Force dan Divide and Conquer

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

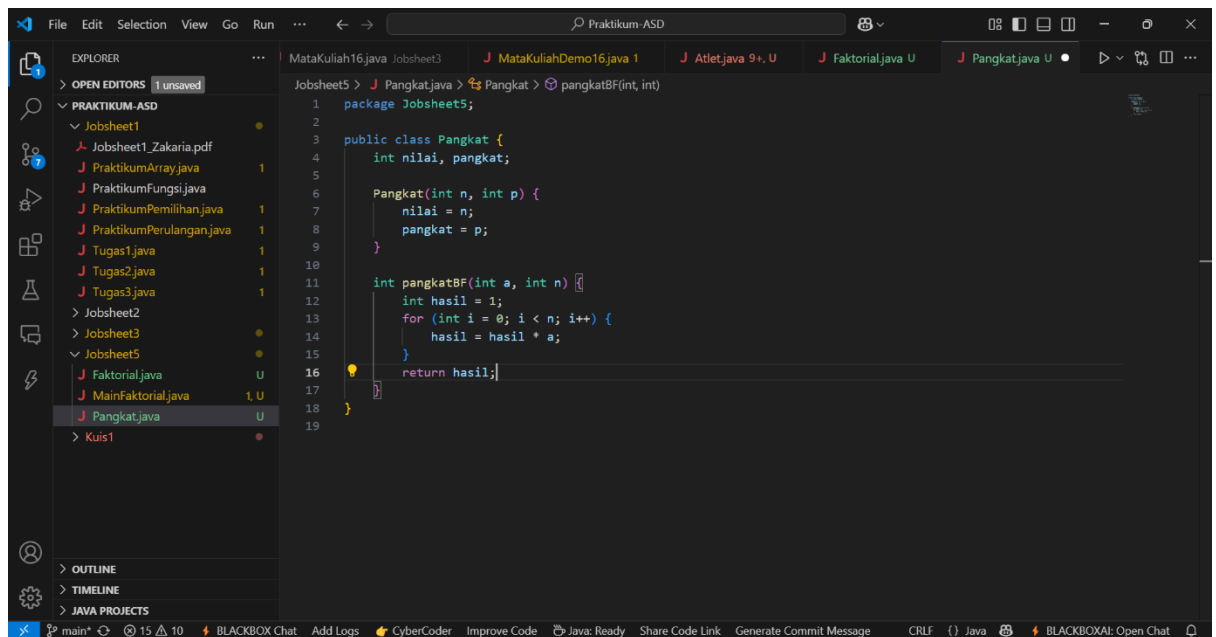


```
1 package Jobsheet5;
2
3 public class Pangkat {
4     int nilai, pangkat;
5 }
6
```

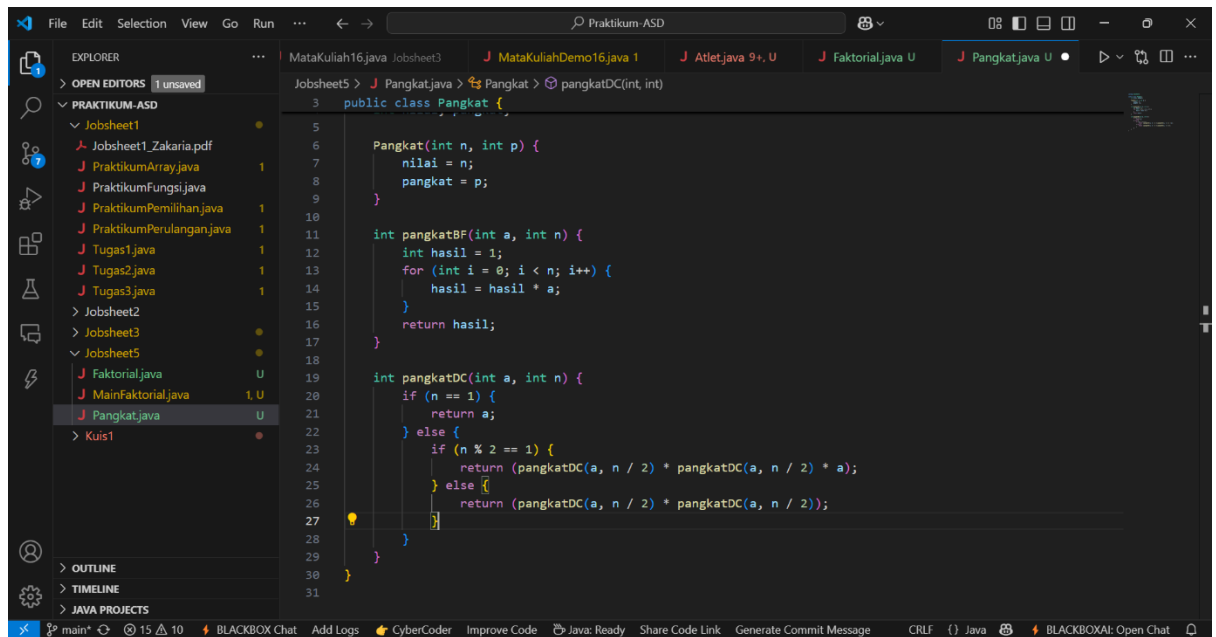
2. Tambahkan konstruktor berparameter



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

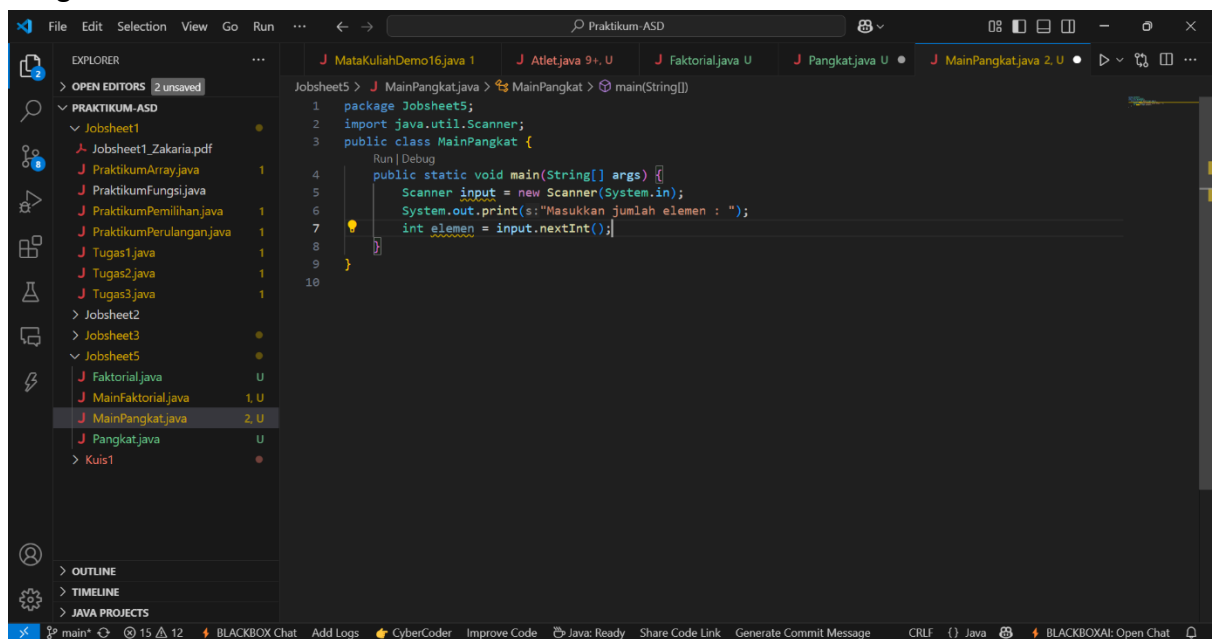


4. Pada class Pangkat juga tambahkan method PangkatDC()



```
public class Pangkat {  
    5  
    6  
    Pangkat(int n, int p) {  
        7  
        nilai = n;  
        8  
        pangkat = p;  
    }  
    9  
    10  
    int pangkatBF(int a, int n) {  
        11  
        int hasil = 1;  
        12  
        for (int i = 0; i < n; i++) {  
            13  
            hasil = hasil * a;  
        }  
        14  
        return hasil;  
    }  
    15  
    16  
    int pangkatDC(int a, int n) {  
        17  
        if (n == 1) {  
            18  
            return a;  
        }  
        19  
        else {  
            20  
            if (n % 2 == 1) {  
                21  
                return (pangkatDC(a, n / 2) * pangkatDC(a, n / 2) * a);  
            }  
            22  
            else {  
                23  
                return (pangkatDC(a, n / 2) * pangkatDC(a, n / 2));  
            }  
        }  
        24  
    }  
    25  
    26  
    27  
    28  
    29  
    30  
    31  
}
```

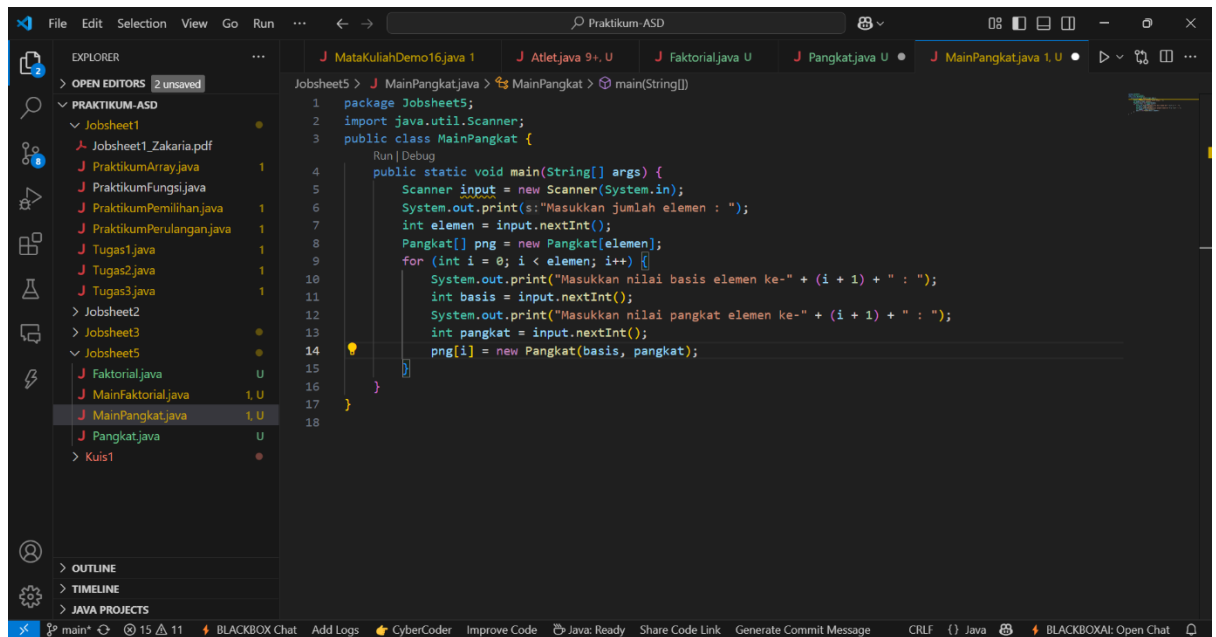
5. Perhatikan apakah sudah tidak ada kesalahan yang muncul dalam pembuatan class Pangkat



```
package Jobsheet5;  
import java.util.Scanner;  
public class MainPangkat {  
    1  
    2  
    3  
    Run | Debug  
    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  
    9  
    10  
}
```

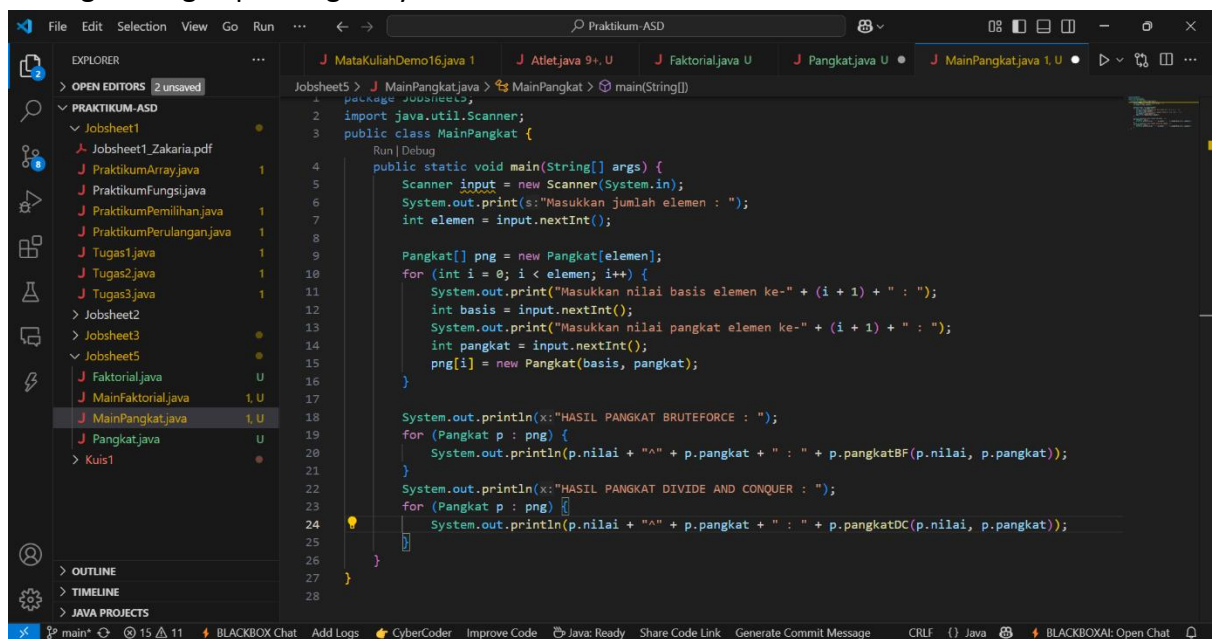
6. Selanjutnya buat class baru yang di dalamnya terdapat method main. Class tersebut dapat dinamakan MainPangkat. Tambahkan kode pada class main untuk

menginputkan jumlah elemen yang akan dihitung pangkatnya.



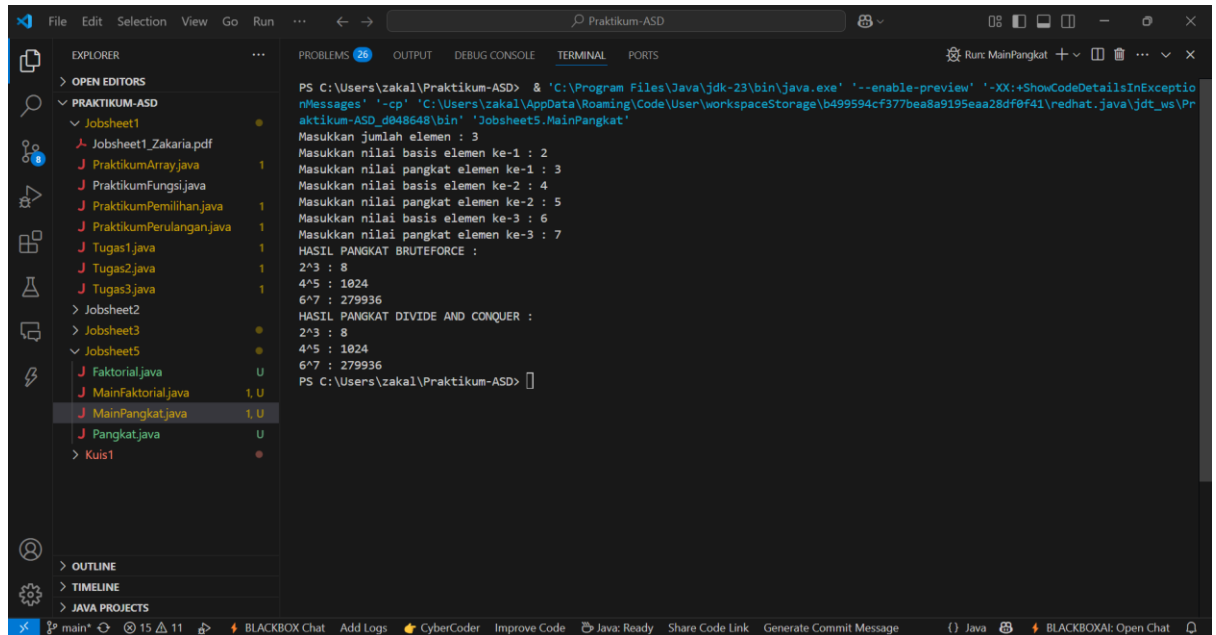
```
1 package Jobsheet5;
2 import java.util.Scanner;
3 public class MainPangkat {
4     public static void main(String[] args) {
5         Scanner input = new Scanner(System.in);
6         System.out.print("Masukkan jumlah elemen : ");
7         int elemen = input.nextInt();
8         Pangkat[] png = new Pangkat[elemen];
9         for (int i = 0; i < elemen; i++) {
10             System.out.print("Masukkan nilai basis elemen ke-" + (i + 1) + " : ");
11             int basis = input.nextInt();
12             System.out.print("Masukkan nilai pangkat elemen ke-" + (i + 1) + " : ");
13             int pangkat = input.nextInt();
14             png[i] = new Pangkat(basis, pangkat);
15         }
16     }
17 }
18
```

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



```
1 package Jobsheet5;
2 import java.util.Scanner;
3 public class MainPangkat {
4     public static void main(String[] args) {
5         Scanner input = new Scanner(System.in);
6         System.out.print("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("HASIL PANGKAT BRUTEFORCE : ");
19        for (Pangkat p : png) {
20            System.out.println(p.nilai + "^" + p.pangkat + " : " + p.pangkatBF(p.nilai, p.pangkat));
21        }
22        System.out.println("HASIL PANGKAT DIVIDE AND CONQUER : ");
23        for (Pangkat p : png) {
24            System.out.println(p.nilai + "^" + p.pangkat + " : " + p.pangkatDC(p.nilai, p.pangkat));
25        }
26    }
27 }
28
```

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



Pertanyaan

1. Jelaskan mengenai perbedaan 2 method yang dibuat yaitu pangkatBF() dan pangkatDC()!
 - pangkatBF() : menggunakan rekursif
 - pangkatDC() : menggunakan iteratif
2. Apakah tahap combine sudah termasuk dalam kode tersebut? Tunjukkan!

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

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?
 - Jika pangkatBF() harus bisa menerima angka berbeda setiap pemanggilan, maka parameter tetap relevan


```

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

```

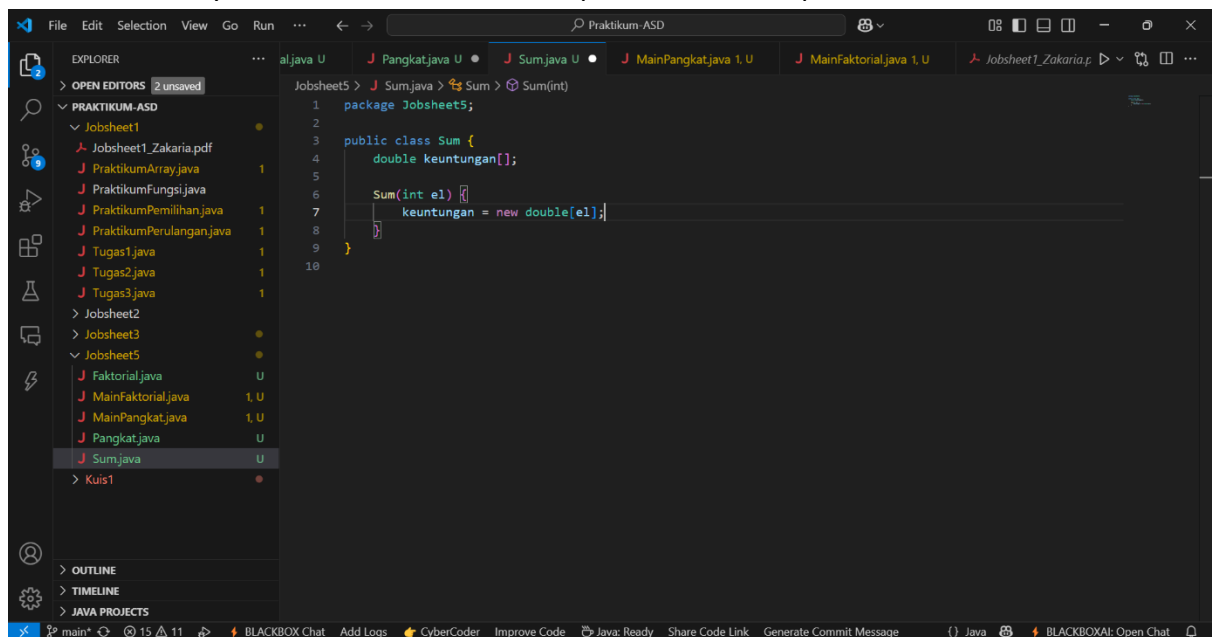
- Bisa,

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

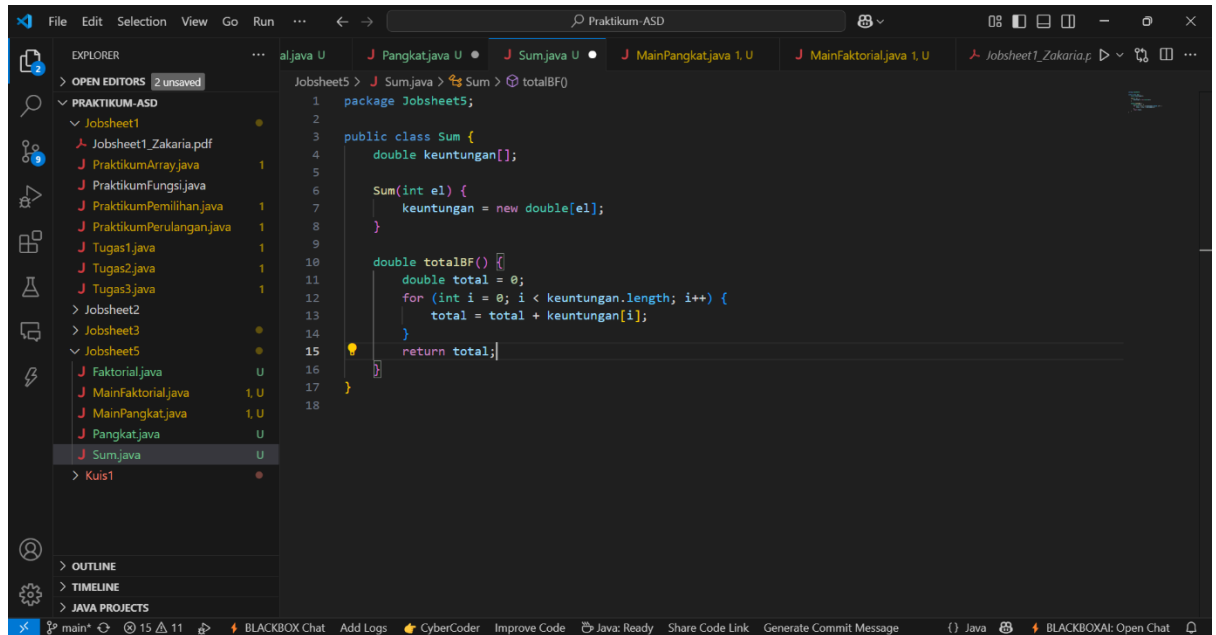
- pangkatBF() : menggunakan perulangan untuk menghitung nilai pangkat
- pangkatDC() : menggunakan rekursi untuk membagi masalah menjadi lebih kecil

Menghitung Sum Array dengan Algoritma Brute Force dan Divide and Conquer

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

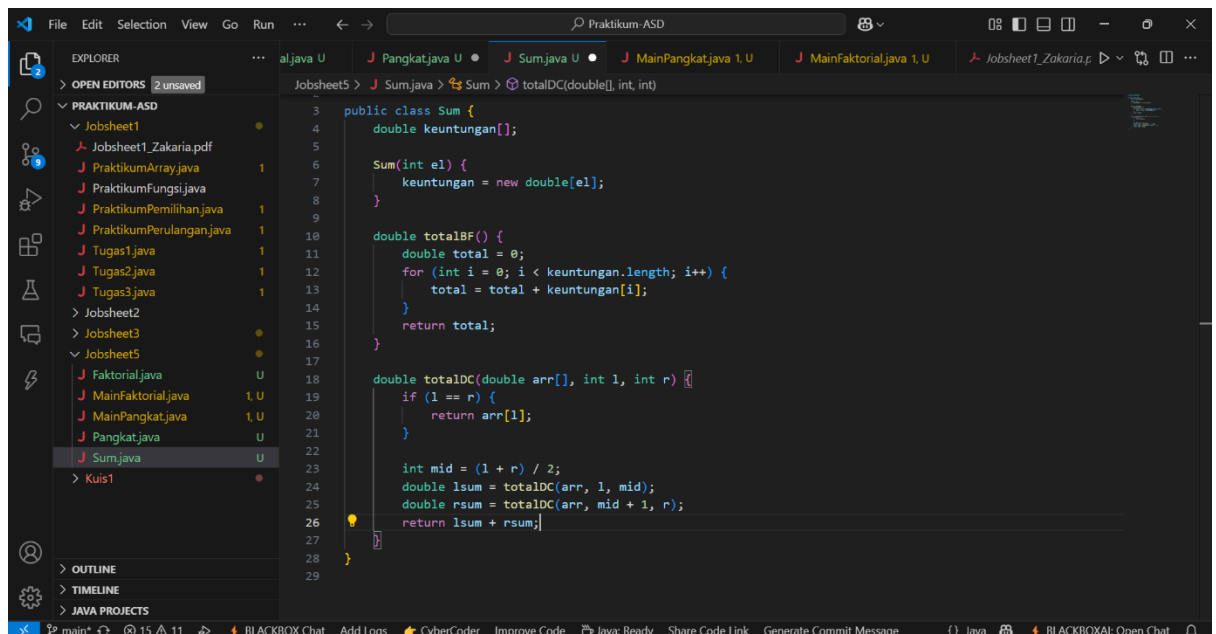


2. Tambahkan method TotalBF() yang akan menghitung total nilai array dengan cara iterative



```
1 package Jobsheet5;
2
3 public class Sum {
4     double keuntungan[];
5
6     Sum(int e1) {
7         keuntungan = new double[e1];
8     }
9
10    double totalBF() {
11        double total = 0;
12        for (int i = 0; i < keuntungan.length; i++) {
13            total = total + keuntungan[i];
14        }
15        return total;
16    }
17 }
18
```

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

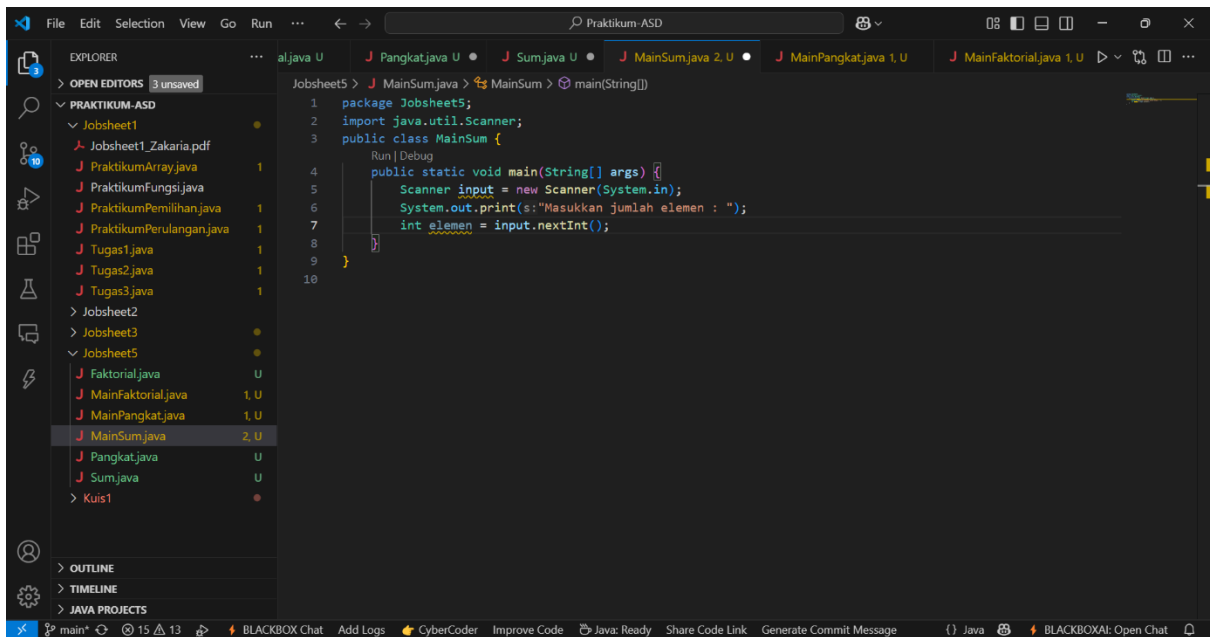


```
1 package Jobsheet5;
2
3 public class Sum {
4     double keuntungan[];
5
6     Sum(int e1) {
7         keuntungan = new double[e1];
8     }
9
10    double totalBF() {
11        double total = 0;
12        for (int i = 0; i < keuntungan.length; i++) {
13            total = total + keuntungan[i];
14        }
15        return total;
16    }
17
18    double totalDC(double arr[], int l, int r) {
19        if (l == r) {
20            return arr[l];
21        }
22
23        int mid = (l + r) / 2;
24        double lsum = totalDC(arr, l, mid);
25        double rsum = totalDC(arr, mid + 1, r);
26        return lsum + rsum;
27    }
28 }
29
```

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

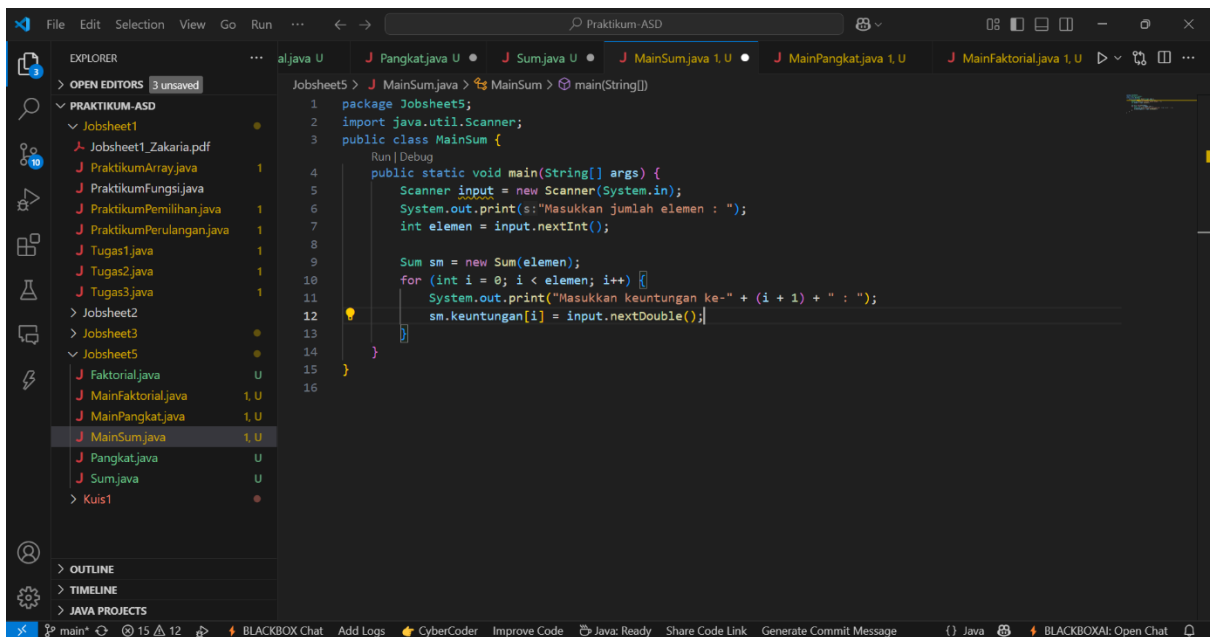
class

Sum



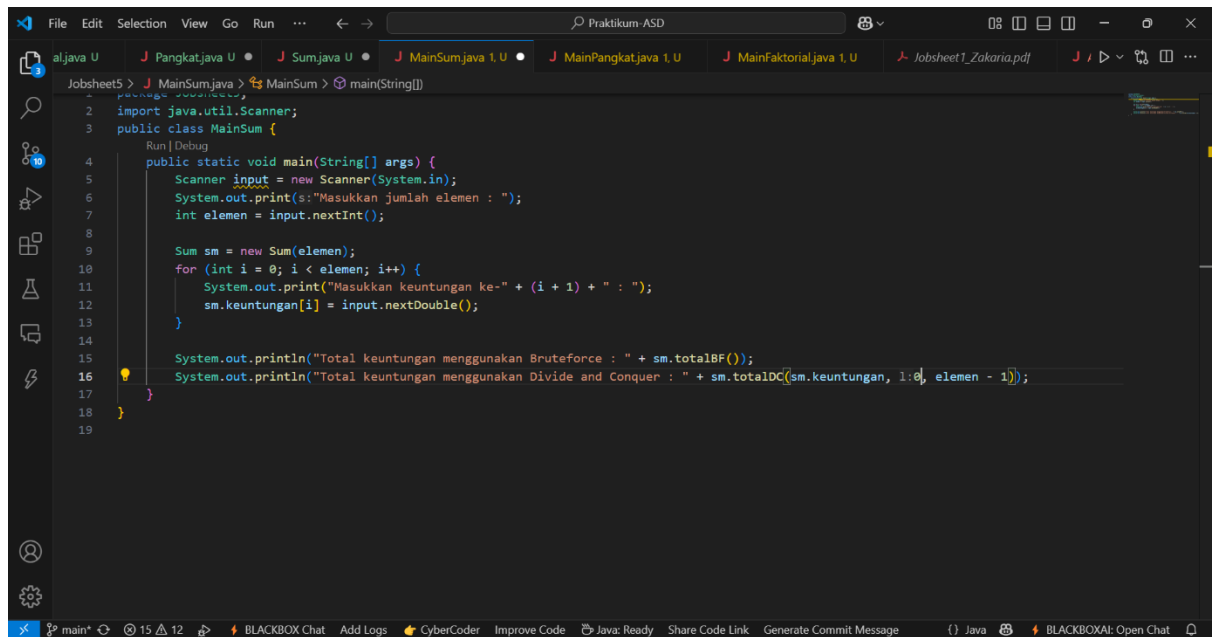
```
1 package Jobsheet5;
2 import java.util.Scanner;
3 public class MainSum {
4     public static void main(String[] args) {
5         Scanner input = new Scanner(System.in);
6         System.out.print(s:"Masukkan jumlah elemen : ");
7         int elemen = input.nextInt();
8     }
9 }
10
```

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



```
1 package Jobsheet5;
2 import java.util.Scanner;
3 public class MainSum {
4     public static void main(String[] args) {
5         Scanner input = new Scanner(System.in);
6         System.out.print(s:"Masukkan jumlah elemen : ");
7         int elemen = input.nextInt();
8
9         Sum sm = new Sum(elemen);
10        for (int i = 0; i < elemen; i++) {
11            System.out.print("Masukkan keuntungan ke-" + (i + 1) + " : ");
12            sm.keuntungan[i] = input.nextDouble();
13        }
14    }
15 }
16
```

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



```
File Edit Selection View Go Run ... < -> Praktikum ASD
al.java U J Pangkat.java U J Sum.java U J MainSum.java 1.U J MainPangkat.java 1.U J MainFaktorial.java 1.U Jobsheet1_Zakaria.pdf
Jobsheet5 > J MainSum.java > MainSum > main(String[])
1 package praktikum;
2 import java.util.Scanner;
3 public class MainSum {
4     public static void main(String[] args) {
5         Scanner input = new Scanner(System.in);
6         System.out.print("Masukkan jumlah elemen : ");
7         int elemen = input.nextInt();
8
9         Sum sm = new Sum(elemen);
10        for (int i = 0; i < elemen; i++) {
11            System.out.print("Masukkan keuntungan ke-" + (i + 1) + " : ");
12            sm.keuntungan[i] = input.nextDouble();
13        }
14
15        System.out.println("Total keuntungan menggunakan Bruteforce : " + sm.totalBF());
16        System.out.println("Total keuntungan menggunakan Divide and Conquer : " + sm.totalDC(sm.keuntungan, 1, elemen - 1));
17    }
18 }
19
```

Pertanyaan

1. Kenapa dibutuhkan variable mid pada method TotalDC()?
 - Variabel mid berfungsi untuk menentukan titik tengah array sehingga kita dapat membagi array menjadi dua bagian yang lebih kecil sebelum menggabungkan hasilnya kembali
2. Untuk apakah statement di bawah ini dilakukan dalam TotalDC()?
 - Statement tersebut untuk rekursi dihitung untuk dua bagian array secara terpisah
3. Kenapa diperlukan penjumlahan hasil lsum dan rsum seperti di bawah ini?
 - Penjumlahan diperlukan agar mendapatkan hasil total dari dua bagian
4. Apakah base case dari totalDC()?
 - If (l==r) , adalah base case atau kondisi berhentinya
5. Tarik Kesimpulan tentang cara kerja totalDC()
 - Implementasi perhitungan nilai total array menggunakan algoritma Divide and Conquer dilakukan dengan method totalDC(), yang mana method berikut menentukan kondisi berhenti (base case) pada bagian if, yaitu if (l==r). method ini mempunyai cara kerja dengan menentukan titik Tengah array dengan tujuan membagi array menjadi dua bagian yang lebih kecil sebelum menggabungkan Kembali hasilnya.