

Amalia Salsa Lutfiana

1G/04/2141720228

FORCE DAN DIVIDE CONQUER

4.1 Tujuan Praktikum

Setelah melakukan materi praktikum ini, mahasiswa mampu:

1. Mahasiswa mampu membuat algoritma bruteforce dan divide-conquer
2. Mahasiswa mampu menerapkan penggunaan algoritma bruteforce dan divide-conquer

4.2 Menghitung Nilai Faktorial dengan Algoritma Brute Force dan Divide and Conquer

Perhatikan Diagram Class berikut ini :

Berdasarkan diagram class di atas, akan dibuat program class dalam Java. Untuk menghitung nilai faktorial suatu angka menggunakan 2 jenis algoritma, Brute Force dan Divide and Conquer. Jika digambarkan terdapat perbedaan proses perhitungan 2 jenis algoritma tersebut sebagai berikut :

Tahapan pencarian nilai faktorial dengan algoritma Brute Force :

Tahapan pencarian nilai faktorial dengan algoritma Divide and Conquer :

4.2.1 Langkah-langkah Percobaan

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 atribut nilai

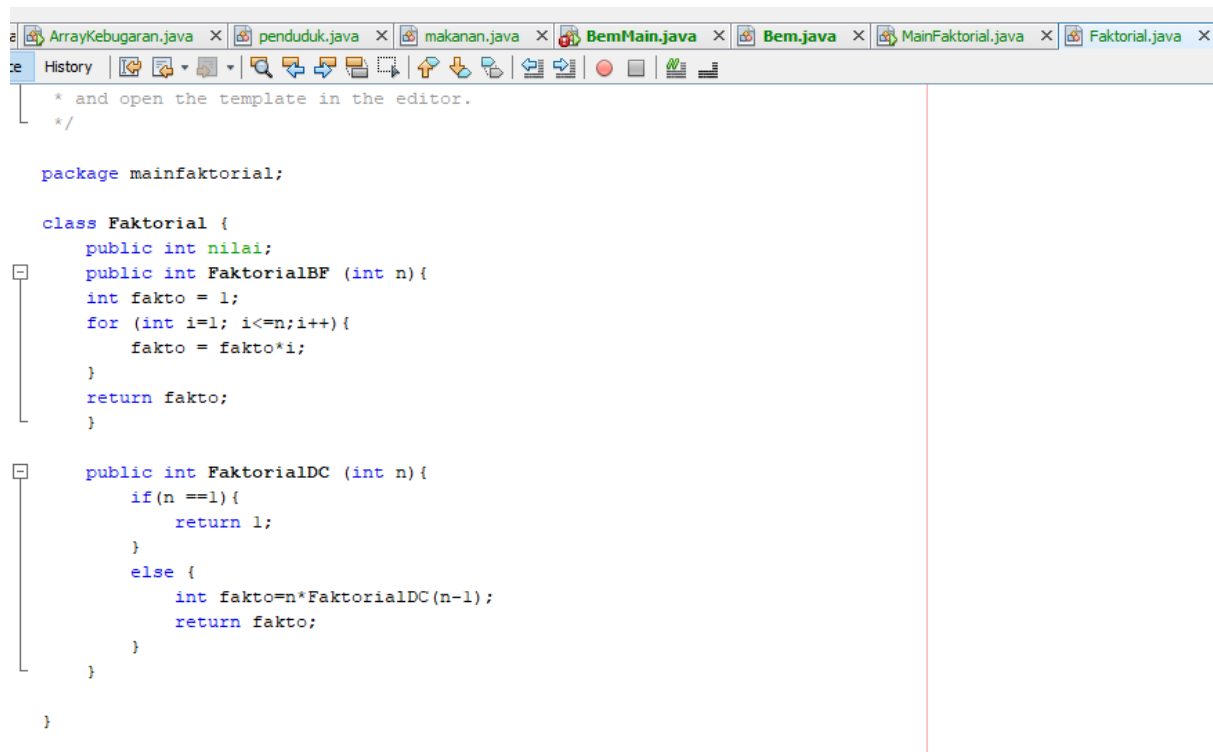
#

b) Tambahkan method faktorialBF() nilai

#

c) Tambahkan method faktorialDC() nilai

#



```

* and open the template in the editor.
*/

package mainfaktorial;

class Faktorial {
    public int nilai;
    public int FaktorialBF (int n){
        int fakto = 1;
        for (int i=1; i<=n;i++){
            fakto = fakto*i;
        }
        return fakto;
    }

    public 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 menginputkan jumlah angka yang akan dicari nilai faktorialnya

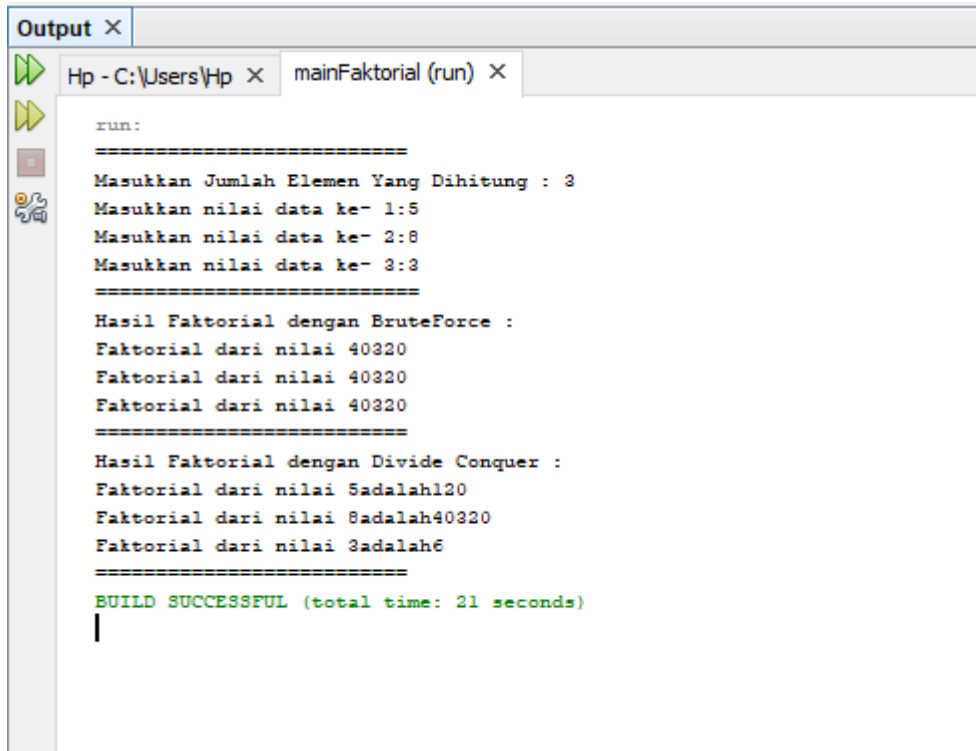
#

b) Buat Array of Objek pada fungsi main, kemudian inputkan beberapa nilai yang akan dihitung faktorialnya

#

c) Tampilkan hasil pemanggilan method faktorialDC() dan faktorialBF()

#



```
Output ×
Hp - C:\Users\Hp ×  mainFaktorial (run) ×

run:
=====
Masukkan Jumlah Elemen Yang Dihitung : 3
Masukkan nilai data ke- 1:5
Masukkan nilai data ke- 2:8
Masukkan nilai data ke- 3:3
=====
Hasil Faktorial dengan BruteForce :
Faktorial dari nilai 40320
Faktorial dari nilai 40320
Faktorial dari nilai 40320
=====
Hasil Faktorial dengan Divide Conquer :
Faktorial dari nilai 5adalah120
Faktorial dari nilai 8adalah40320
Faktorial dari nilai 3adalah6
=====
BUILD SUCCESSFUL (total time: 21 seconds)
|
```

4.2.3 Pertanyaan

1. Jelaskan mengenai base line Algoritma Divide Conquer untuk melakukan pencarian nilai faktorial!

Karena jika nilai dari n sama dengan 1, program akan melakukan return 1, dan jika tidak maka nilai pada variable n akan dikurang -1, dimana di setiap perulangan yang di lakukan akan dikalikan dengan nilai sebelumnya.

2. Pada implementasi Algoritma Divide and Conquer Faktorial apakah lengkap terdiri dari 3 tahapan divide, conquer, combine? Jelaskan masing-masing bagiannya pada kode program!

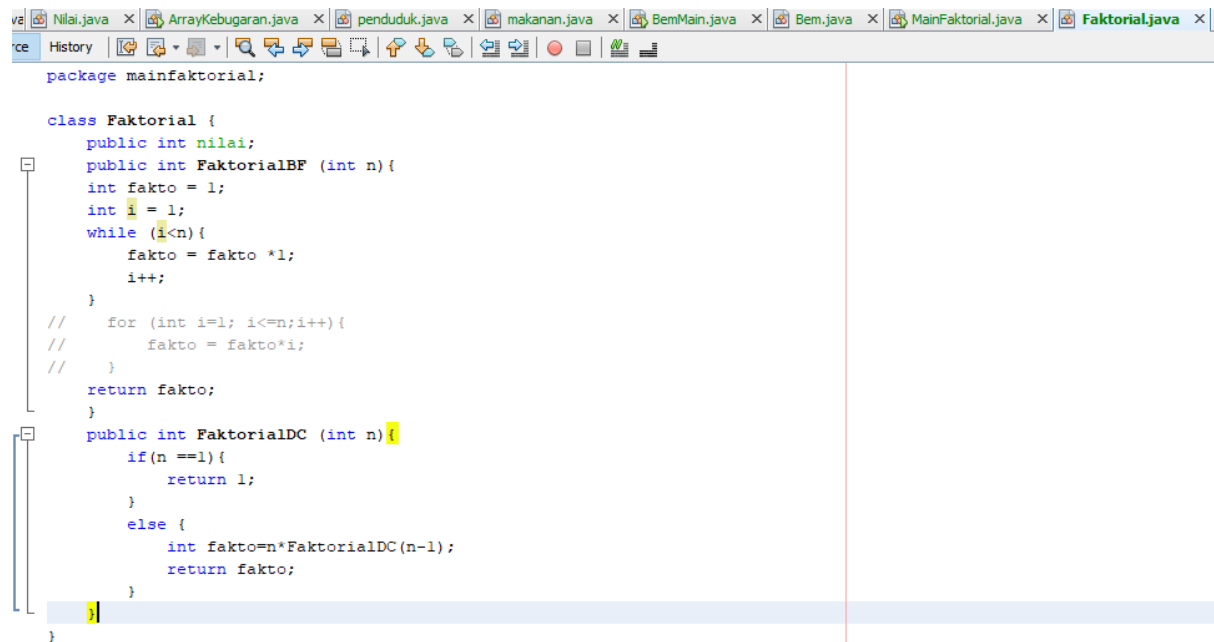
- Divide = faktorialDC($n-1$) pengurangan pada nilai n .
- Conquer = $n \cdot \text{faktorialDC}(n-1)$ penyelesaian dengan rekursif.
- Combine = $n \cdot \text{faktorialDC}(n-1)$ pengombinasian dengan perkalian.

3. Apakah memungkinkan perulangan pada method faktorialBF() dirubah selain menggunakan for?Buktikan!

Ya, mungkin , mungkin saja perulangan pada method faktorialBF() dirubah selain menggunakan for dapat juga di rubah menjadi perulangan jenis while maupun do while.

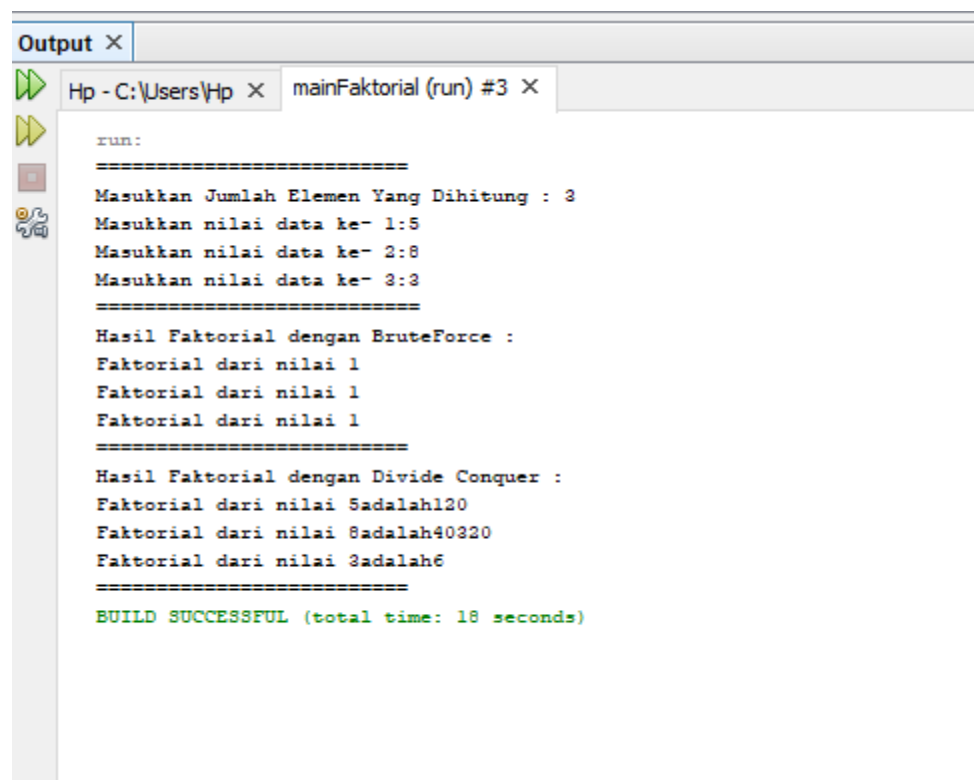
Contohnya pada baris kode di bawah ini :

While



```
package mainfaktorial;

class Faktorial {
    public int nilai;
    public int FaktorialBF (int n) {
        int fakto = 1;
        int i = 1;
        while (i < n) {
            fakto = fakto * i;
            i++;
        }
        // for (int i=1; i<=n; i++){
        //     fakto = fakto*i;
        // }
        return fakto;
    }
    public int FaktorialDC (int n) {
        if (n == 1) {
            return 1;
        }
        else {
            int fakto = n * FaktorialDC (n-1);
            return fakto;
        }
    }
}
```



```
Output X
Hp - C:\Users\Hp X mainFaktorial (run) #3 X

run:
=====
Masukkan Jumlah Elemen Yang Dihitung : 3
Masukkan nilai data ke- 1:5
Masukkan nilai data ke- 2:8
Masukkan nilai data ke- 3:3
=====
Hasil Faktorial dengan BruteForce :
Faktorial dari nilai 1
Faktorial dari nilai 1
Faktorial dari nilai 1
=====
Hasil Faktorial dengan Divide Conquer :
Faktorial dari nilai 5 adalah 120
Faktorial dari nilai 8 adalah 40320
Faktorial dari nilai 3 adalah 6
=====
BUILD SUCCESSFUL (total time: 18 seconds)
```

Do-while

```
package mainfaktorial;

class Faktorial {
    public int nilai;
    public int FaktorialBF (int n){
        int fakto = 1;
        int i = 1;

        do{
            fakto = fakto * i;
            i++;
        } while (i <= n);

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

4. Tambahkan pengecekan waktu eksekusi kedua jenis method tersebut!

#

```
package mainfaktorial;

import java.util.Scanner;

public class MainFaktorial {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        Scanner sc = new Scanner (System.in);
        System.out.println("=====");
        System.out.print ("Masukkan Jumlah Elemen Yang Dihitung : ");
        int elemen = sc.nextInt();

        Faktorial [] fk = new Faktorial[elemen];

        for(int i=0; i<elemen; i++){
            fk[i] = new Faktorial();
            System.out.print ("Masukkan nilai data ke- " + (i+1) + ":");
            fk[i].nilai = sc.nextInt();
        }
    }
}
```

```

        long start = System.currentTimeMillis();
        System.out.println("=====");
        System.out.println("Hasil Faktorial dengan BruteForce : ");
        for(int j = 0; j < elemen; j++){
            System.out.println("Faktorial dari nilai " + fk[j].nilai + " adalah : " + fk[j].FaktorialBF(fk[j].nilai));
        }
        long end = System.currentTimeMillis();
        long elapsedtime = end - start;
        System.out.println("Waktu " + String.valueOf(elapsedtime));
        long start1 = System.currentTimeMillis();
        System.out.println("=====");
        System.out.println("Hasil Faktorial dengan Divide dan Conquer");
        for (int k = 0; k < elemen; k++){
            System.out.println("Faktorial dari nilai " + fk[k].nilai + " adalah : " + fk[k].FaktorialDC(fk[k].nilai));
        }

        long end1 = System.currentTimeMillis();
        long elapsedtimel = end1 - start1;
        System.out.println("Waktu " + String.valueOf(elapsedtimel));
        System.out.println("=====");
    }
}

```

```

run:
=====
Masukkan Jumlah Elemen Yang Dihitung : 2
Masukkan nilai data ke- 1: 2
Masukkan nilai data ke- 2: 45
=====
Hasil Faktorial dengan BruteForce :
Faktorial dari nilai 2 adalah : 2
Faktorial dari nilai 45 adalah : 0
Waktu 0
=====
Hasil Faktorial dengan Divide dan Conquer
Faktorial dari nilai 2 adalah : 2
Faktorial dari nilai 45 adalah : 0
Waktu 0
=====
BUILD SUCCESSFUL (total time: 12 seconds)

```

5. Buktikan dengan inputan elemen yang di atas 20 angka, apakah ada perbedaan waktu eksekusi?

Ada perbedaan waktu saat eksekusi seperti gambar dibawah ini

#

4.3 Menghitung Hasil Pangkat dengan Algoritma Brute Force dan Divide and Conquer

4.3.1 Langkah-langkah percobaan :

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 pemangkatnya

#

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

#

3. Pada class Pangkat juga tambahkan method PangkatDC

#

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

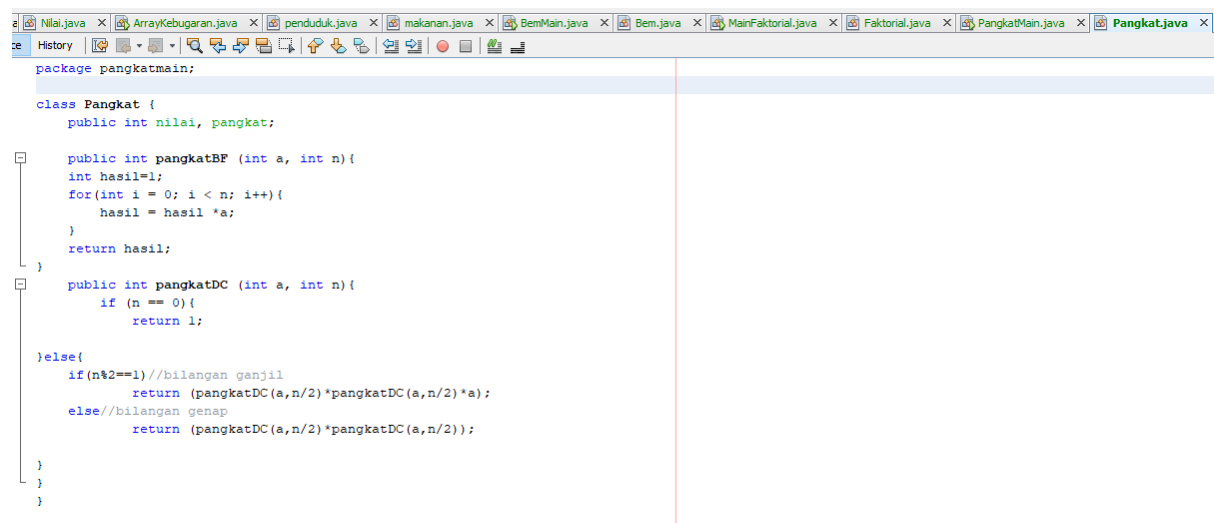
#

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

#

6. 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.

#



```
package pangkatmain;

class Pangkat {
    public int nilai, pangkat;

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

    public int pangkatDC (int a, int n){
        if (n == 0){
            return 1;
        }else{
            if (n%2==1){//bilangan ganjil
                return (pangkatDC (a,n/2) *pangkatDC (a,n/2) *a);
            }else{//bilangan genap
                return (pangkatDC (a,n/2) *pangkatDC (a,n/2));
            }
        }
    }
}
```

Class main


```

package pangkatmain;

import java.util.Scanner;

public class PangkatMain {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        Scanner sc = new Scanner (System.in);

        System.out.println("=====");
        System.out.print("Masukkan jumlah elemen yang ingin dihitung : ");
        int elemen = sc.nextInt();

        Pangkat[] png = new Pangkat[elemen];

        for (int i = 0; i < elemen; i++){
            png[i] = new Pangkat();
            System.out.print("Masukkan nilai yang akan dipangkatkan ke-"+(i+1)+" : ");
            png[i].nilai = sc.nextInt();
            System.out.print("Masukkan nilai pemangkat ke-"+(i+1)+" : ");
            png[i].pangkat = sc.nextInt();
        }

        System.out.println("=====");
        System.out.println("Hasil Pangkat dengan Brute Force");
        for (int i = 0; i < elemen; i++){
            System.out.println("Nilai "+png[i].nilai+ " pangkat : " + png[i].pangkat+ "adalah : " + png[i].pangkatBF (png[i].nilai,png[i].pangkat));
        }
        System.out.println("=====");
        System.out.println("Hasil Pangkat dengan Divide and Conquer");
        for (int i = 0; i < elemen; i++){
            System.out.println("Nilai "+png[i].nilai+ " pangkat : " + png[i].pangkat+ "adalah : " + png[i].pangkatBF (png[i].nilai,png[i].pangkat));
        }
        System.out.println("=====");
    }
}

```

4.3.2 Verifikasi Hasil Percobaan

```

Output X
pangkatMain (run) X Hp - C:\Users\Hp X

run:
=====
Masukkan jumlah elemen yang ingin dihitung : 2
Masukkan nilai yang akan dipangkatkan ke-1 : 6
Masukkan nilai pemangkat ke-1 : 2
Masukkan nilai yang akan dipangkatkan ke-2 : 4
Masukkan nilai pemangkat ke-2 : 3
=====
Hasil Pangkat dengan Brute Force
Nilai6pangkat : 2adalah : 36
Nilai4pangkat : 3adalah : 64
=====
Hasil Pangkat dengan Divide and Conquer
Nilai6pangkat : 2adalah : 36
Nilai4pangkat : 3adalah : 64
=====
BUILD SUCCESSFUL (total time: 25 seconds)

```

4.3.3 Pertanyaan

1. Jelaskan mengenai perbedaan 2 method yang dibuat yaitu PangkatBF() dan PangkatDC()!

Perbedaan dari method PangkatBF() dan PangkatDC adalah method pangkatBF() menggunakan fungsi yang melakukan proses perulangan terhadap suatu intruksi dan ketika syarat tersebut tidak terpenuhi lagi maka perulangan akan terhenti. Dan pada method pangkatDC() menggunakan fungsi rekursif yang mana perulangan terjadi akibat pengekseskusan suatu fungsi yang mana fungsi tersebut memanggil dirinya sendiri.

2. Pada method PangkatDC() terdapat potongan program sebagai berikut, Jelaskan arti potongan kode tersebut

Maksud dari potongan kode tersebut adalah apabila terdapat suatu bilangan ganjil maka program akan melakukan pembagian terhadap variable n, yakni di bagi 2 kemudian dikalikan dengan method itu sendiri dan kemudian di kalikan lagi dengan variable a. dan apabila bilangan tersebut genap maka program akan melakukan return dan melakukan pembagian terhadap variable n, yang mana variable n di bagi 2 dan kemudian di kalikan dengan method itu sendiri namun tidak dikalikan dengan nilai daripada variable a.

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

Tahap combine sudah terjadi pada kode tersebut, dan lebih tepatnya tahapan tersebut di lakukan pada class Pangkat yang mana lebih tepatnya terletak pada method pangkatDC.

4. Modifikasi kode program tersebut, anggap proses pengisian atribut dilakukan dengan konstruktor.

5. Tambahkan menu agar salah satu method yang terpilih saja yang akan dijalankan!

4.4 Menghitung Sum Array dengan Algoritma Brute Force dan Divide and Conquer

Di dalam percobaan ini, kita akan mempraktekkan bagaimana proses divide, conquer, dan combine diterapkan pada studi kasus penjumlahan keuntungan suatu perusahaan dalam beberapa bulan.

4.4.1 Langkah-langkah Percobaan

1. Pada paket minggu5. Buat class baru yaitu class Sum. DI salam class tersebut terdapat beberapa atribut jumlah elemen array, array, dan juga total. 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

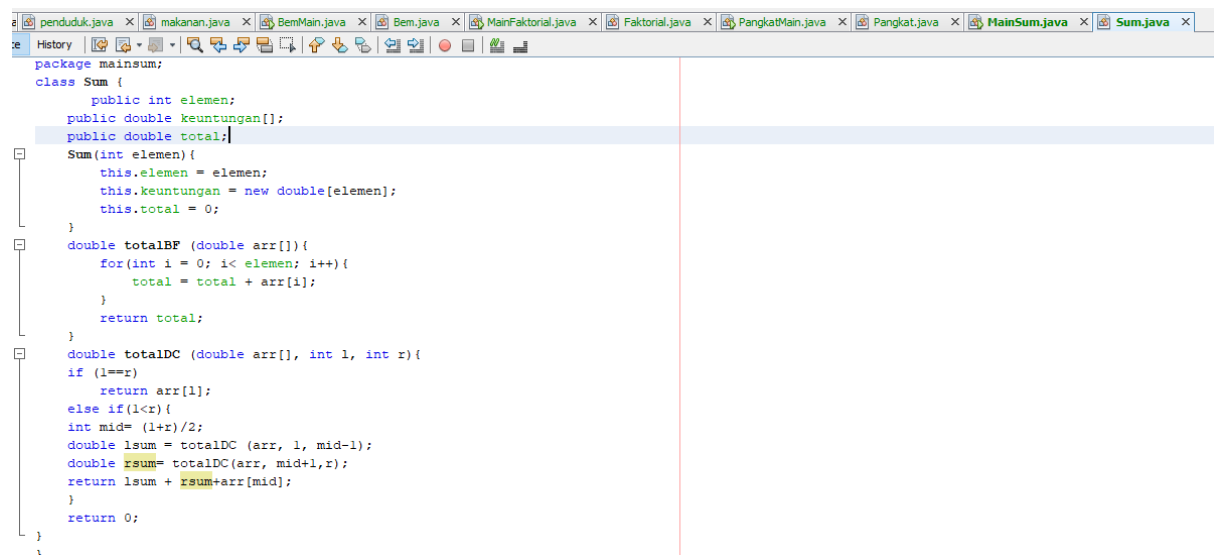
#

5. 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.

#

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

#



```
package mainsum;

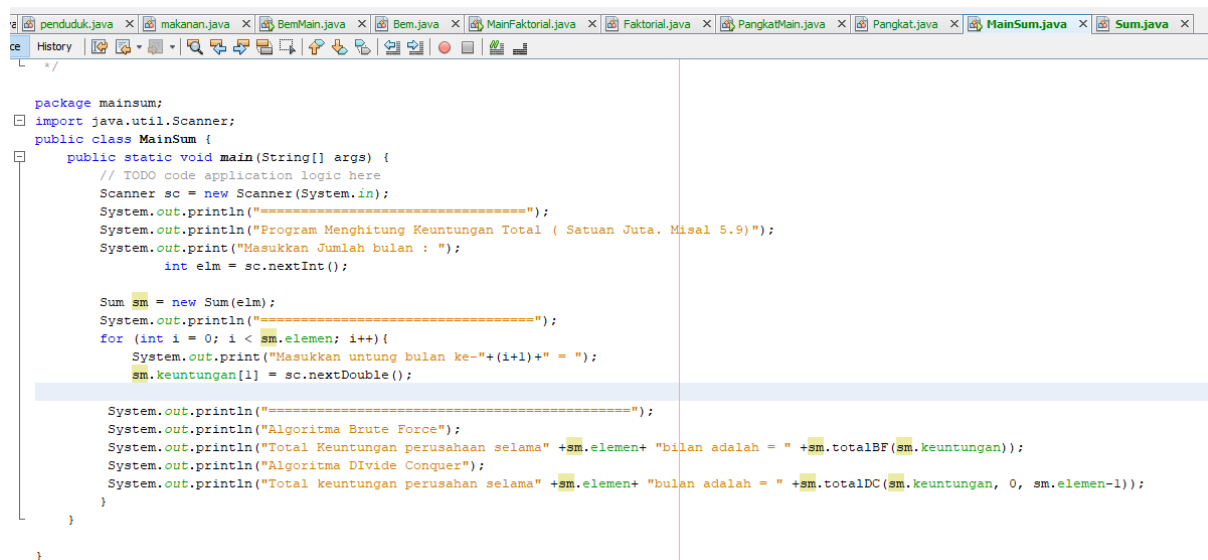
class Sum {
    public int elemen;
    public double keuntungan[];
    public double total;

    Sum(int elemen){
        this.elemen = elemen;
        this.keuntungan = new double[elemen];
        this.total = 0;
    }

    double totalBF (double arr[]){
        for(int i = 0; i< elemen; i++){
            total = total + arr[i];
        }
        return total;
    }

    double totalDC (double arr[], int l, int r){
        if (l==r)
            return arr[l];
        else if (l<r){
            int mid= (l+r)/2;
            double lsum = totalDC (arr, l, mid-1);
            double rsum= totalDC(arr, mid+1,r);
            return lsum + rsum+arr[mid];
        }
        return 0;
    }
}
```

Class main



```
package mainsum;
import java.util.Scanner;
public class MainSum {
    public static void main(String[] args) {
        // TODO code application logic here
        Scanner sc = new Scanner(System.in);
        System.out.println("=====");
        System.out.println("Program Menghitung Keuntungan Total (Satuan Juta, Misal 5.9)");
        System.out.print("Masukkan Jumlah bulan : ");
        int elm = sc.nextInt();

        Sum sm = new Sum(elm);
        System.out.println("=====");
        for (int i = 0; i < sm.elemen; i++) {
            System.out.print("Masukkan untung bulan ke-" + (i+1) + " = ");
            sm.keuntungan[i] = sc.nextDouble();

            System.out.println("=====");
            System.out.println("Algoritma Brute Force");
            System.out.println("Total Keuntungan perusahaan selama " + sm.elemen + " bulan adalah = " + sm.totalBF(sm.keuntungan));
            System.out.println("Algoritma DIvide Conquer");
            System.out.println("Total keuntungan perusahaan selama " + sm.elemen + " bulan adalah = " + sm.totalDC(sm.keuntungan, 0, sm.elemen-1));
        }
    }
}
```

4.4.2 Verifikasi Hasil Percobaan

Cocokkan hasil compile kode program anda dengan gambar berikut ini.

4.4.3 Pertanyaan

1. Berikan ilustrasi perbedaan perhitungan keuntungan dengan method TotalBF() ataupun TotalDC()

a. Perbedaan perhitungan yang terjadi pada method TotalBF() adalah, kode program terkesan lebih ringkas dan sederhana, dan menjadikannya lebih mudah untuk di pahami.

b. Perbedaan perhitungan yang terjadi pada method TotalDC() adalah, kode program lebih panjang daripada kode program pada method TotalBF(),method ini juga memiliki keuntungan dapat memecahkan masalah yang sulit.

2. Perhatikan output dari kedua jenis algoritma tersebut bisa jadi memiliki hasil berbeda di belakang koma. Bagaimana membatasi output di belakang koma agar menjadi standar untuk kedua jenis algoritma tersebut

Dengan cara mengganti sintaks println menjadi sintaks printf dan juga menggunakan metode pembatasan karakter.

3. Mengapa terdapat formulasi return value berikut?Jelaskan

Return value tersebut berguna untuk mengembalikan nilai dari variable lsum ,variable rsum dan juga arr [mid] yang mana masing-masing nya di jumlah kan terlebih dahulu.

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

Di butuhkan variable mid pada method TotalDC()dikarenakan variable ini dapat menampng nilai dari perhitungan $(1+r)/2$ yang akan digunakan sebagai pengisian nilai pada parameter di method TotalDC()

5. Program perhitungan keuntungan suatu perusahaan ini hanya untuk satu perusahaan saja. Bagaimana cara menghitung sekaligus keuntungan beberapa bulan untuk beberapa perusahaan.(Setiap perusahaan bisa saja memiliki jumlah bulan berbeda-beda)? Buktikan dengan program!

Hasil Kode program akan menjadi seperti ini: