SAFRIZAL RAHMAN

19/ SIB-1G

JOBSHEET IV  
BRUTE FORCE DAN DIVIDE CONQUER

1. Tujuan Praktikum

Setelah melakukan materi praktikum ini, mahasiswa mampu:

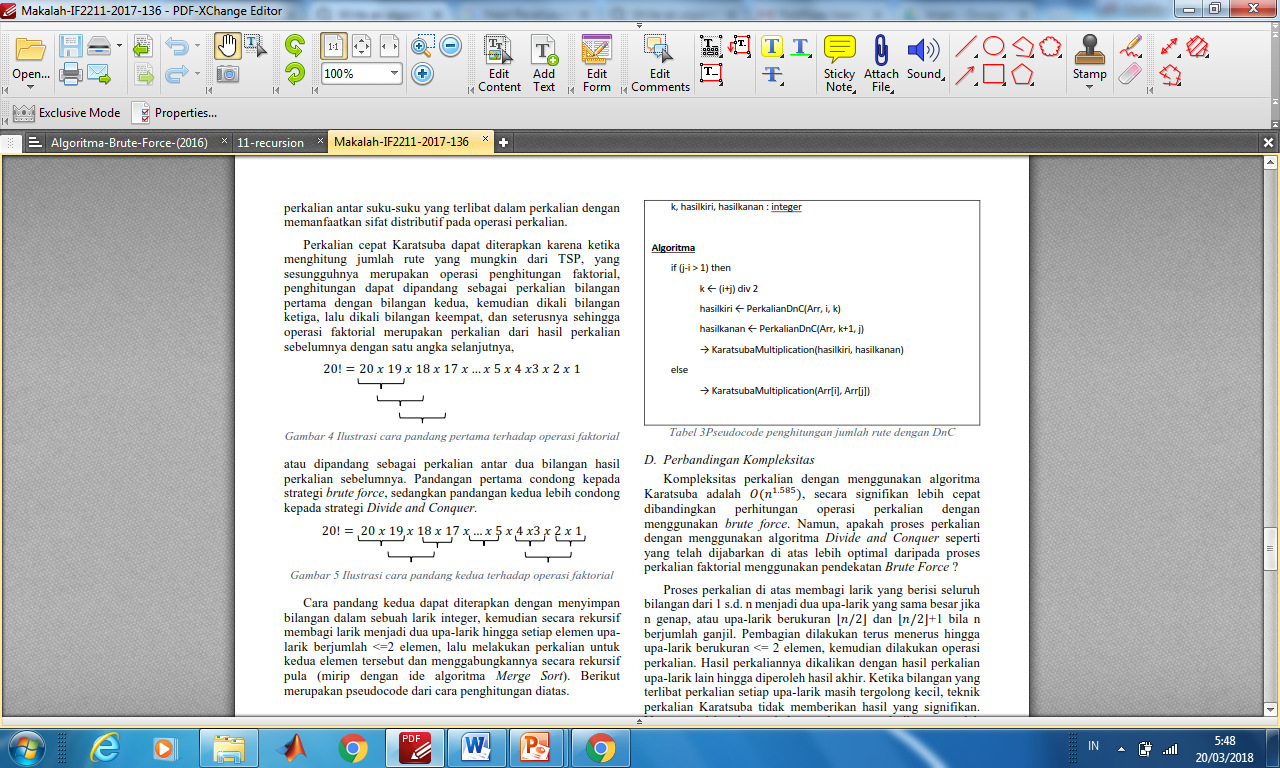
1. Mahasiswa mampu membuat algoritma bruteforce dan divide-conquer
2. Mahasiswa mampu menerapkan penggunaan algorima bruteforce dan divide-conquer
3. Menghitung Nilai Faktorial dengan Algoritma Brute Force dan Divide and Conquer

Perhatikan Diagram Class berikut ini :

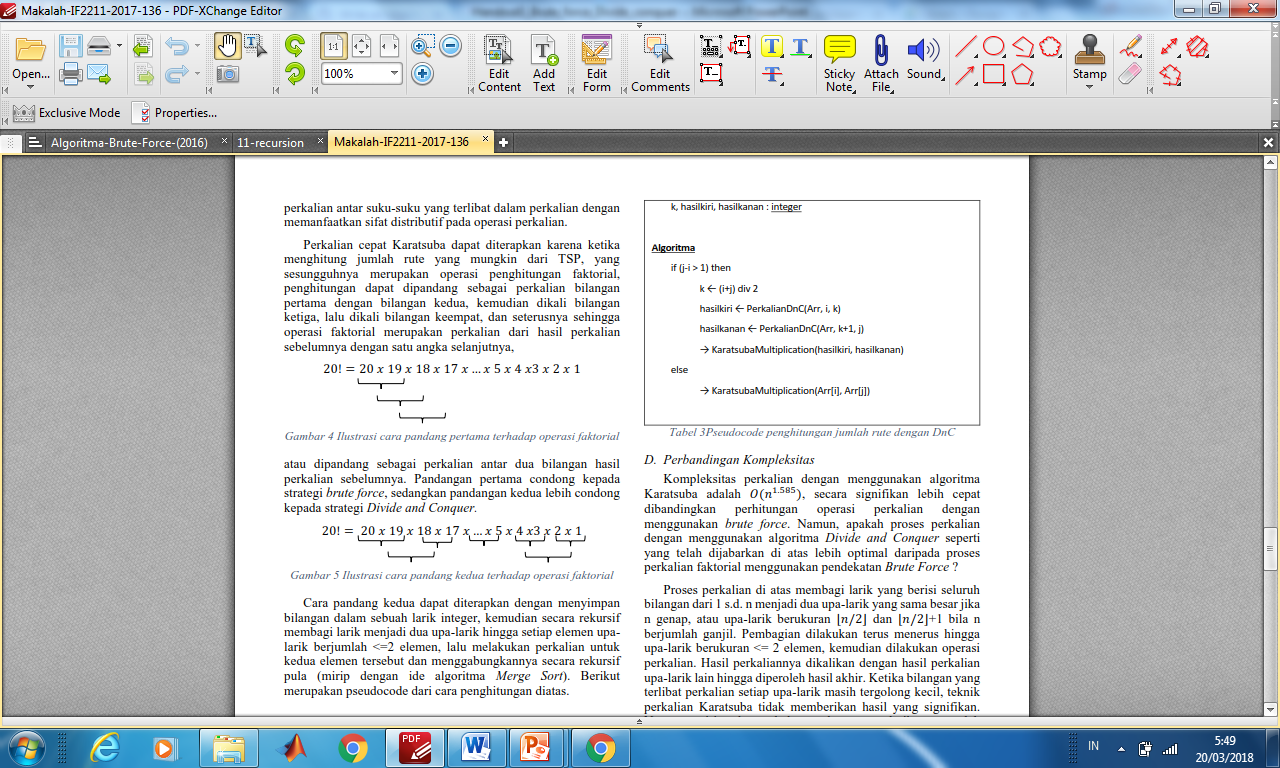
|  |
| --- |
| Faktorial |
| nilai: int |
| faktorialBF(): int  faktorialDC(): int |

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 :

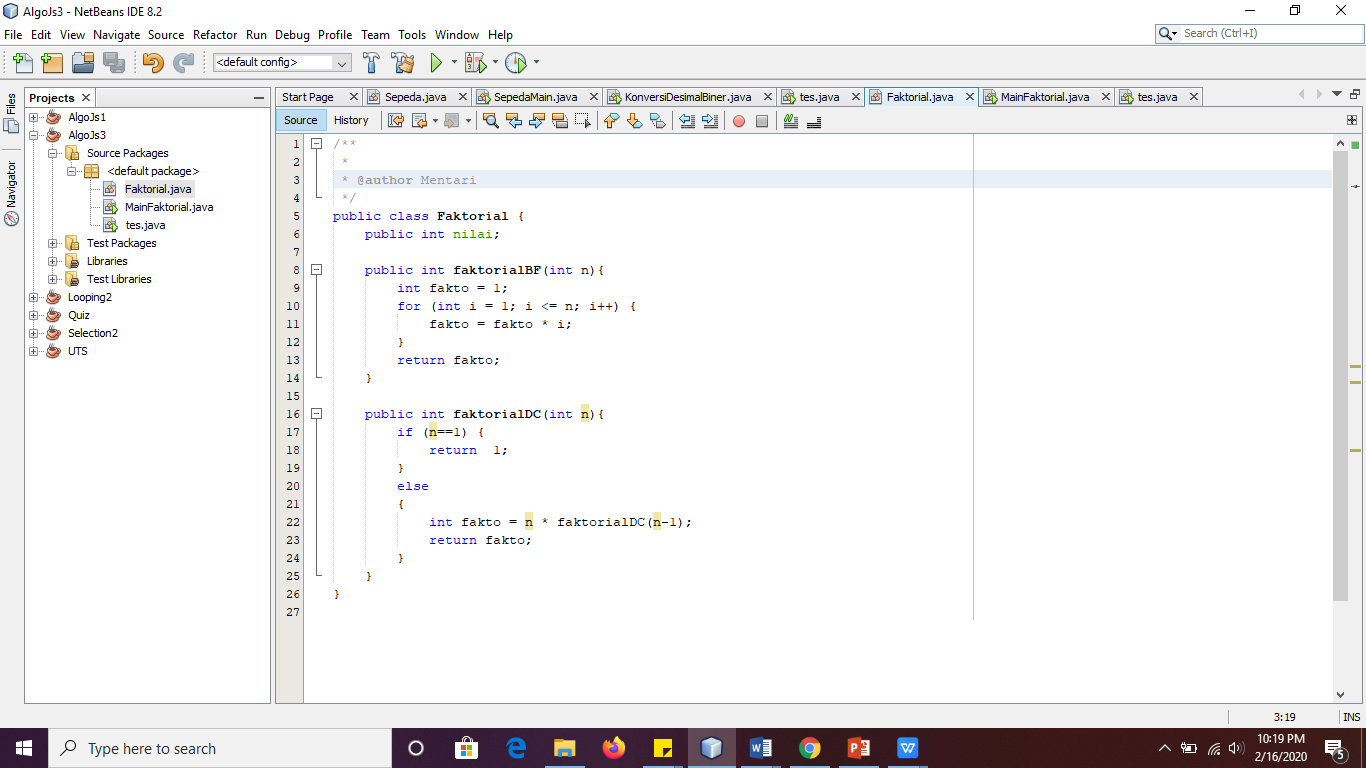
Tahapan pencarian nilai faktorial dengan algoritma Brute Force :



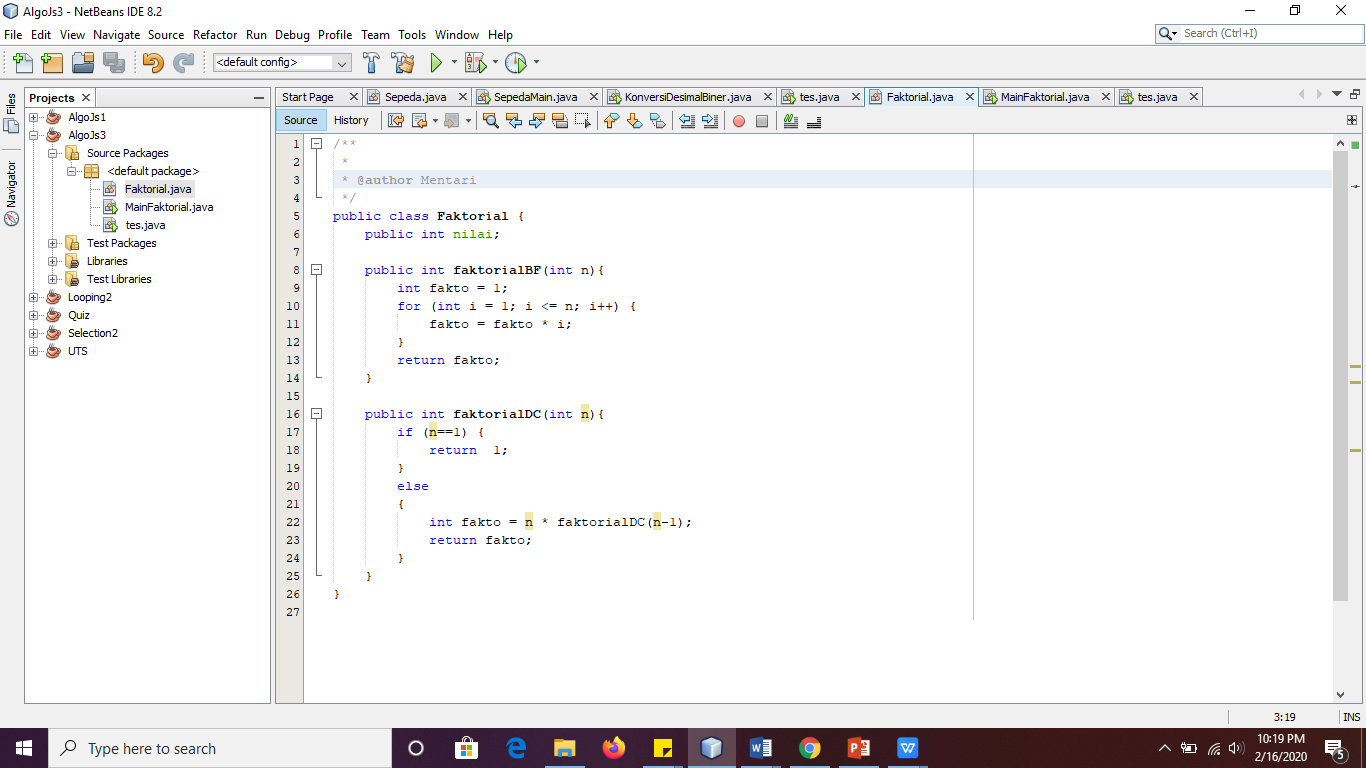
Tahapan pencarian nilai faktorial dengan algoritma Divide and Conquer :



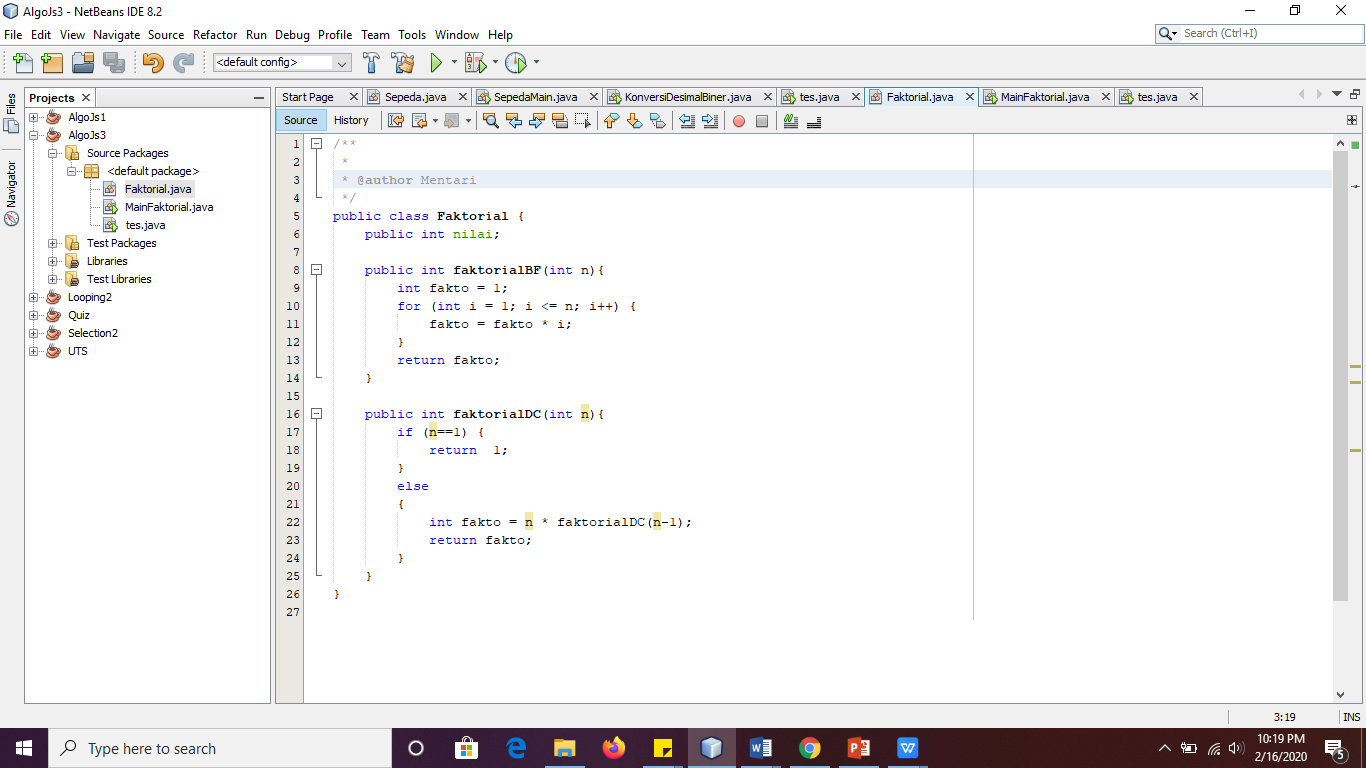
1. Langkah-langkah Percobaan
2. Buat Project baru, dengannama “**BruteForceDivideConquer**”. Buat package dengan nama minggu5.
3. Buatlah class baru dengan nama **Faktorial**
4. Lengkapi class **Faktorial** dengan atribut dan method yang telah digambarkan di dalam diagram class di atas, sebagai berikut:
5. Tambahkan atribut nilai



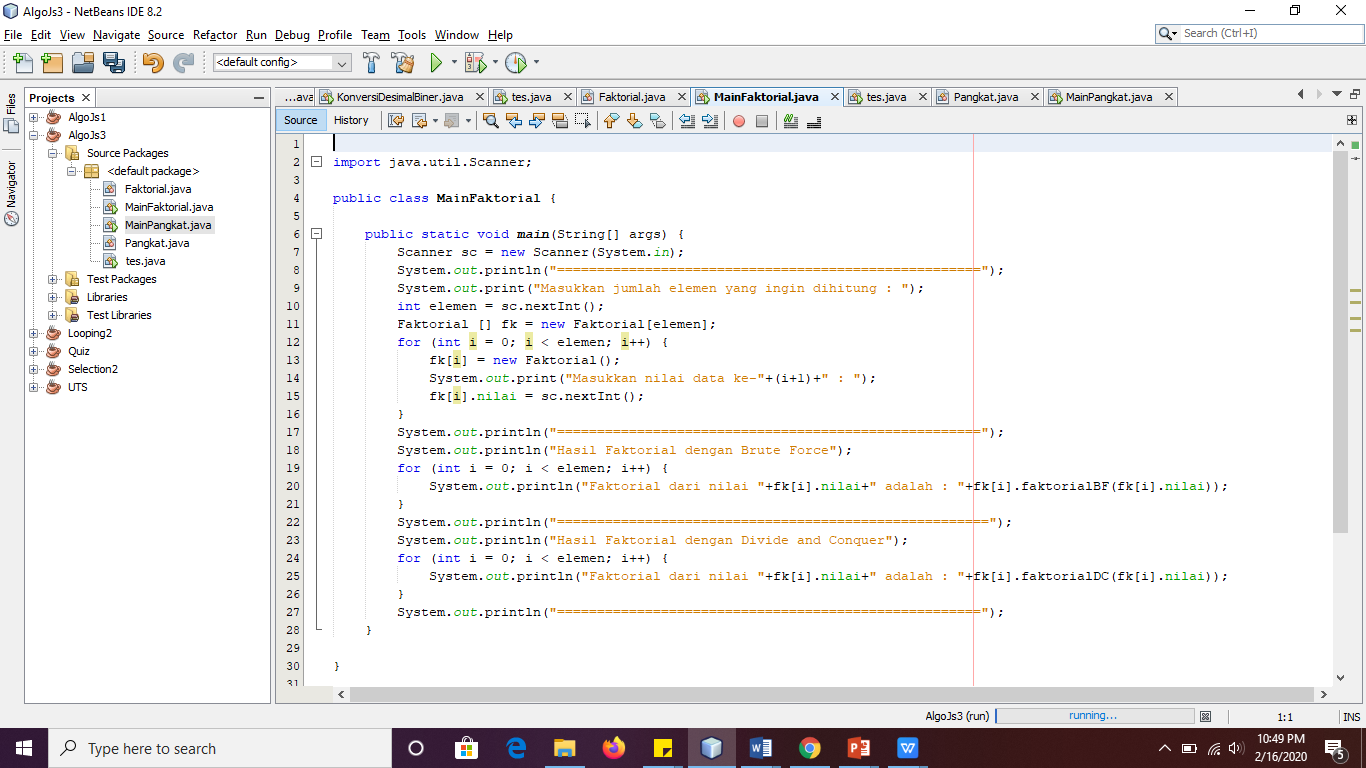
1. Tambahkan method faktorialBF() nilai



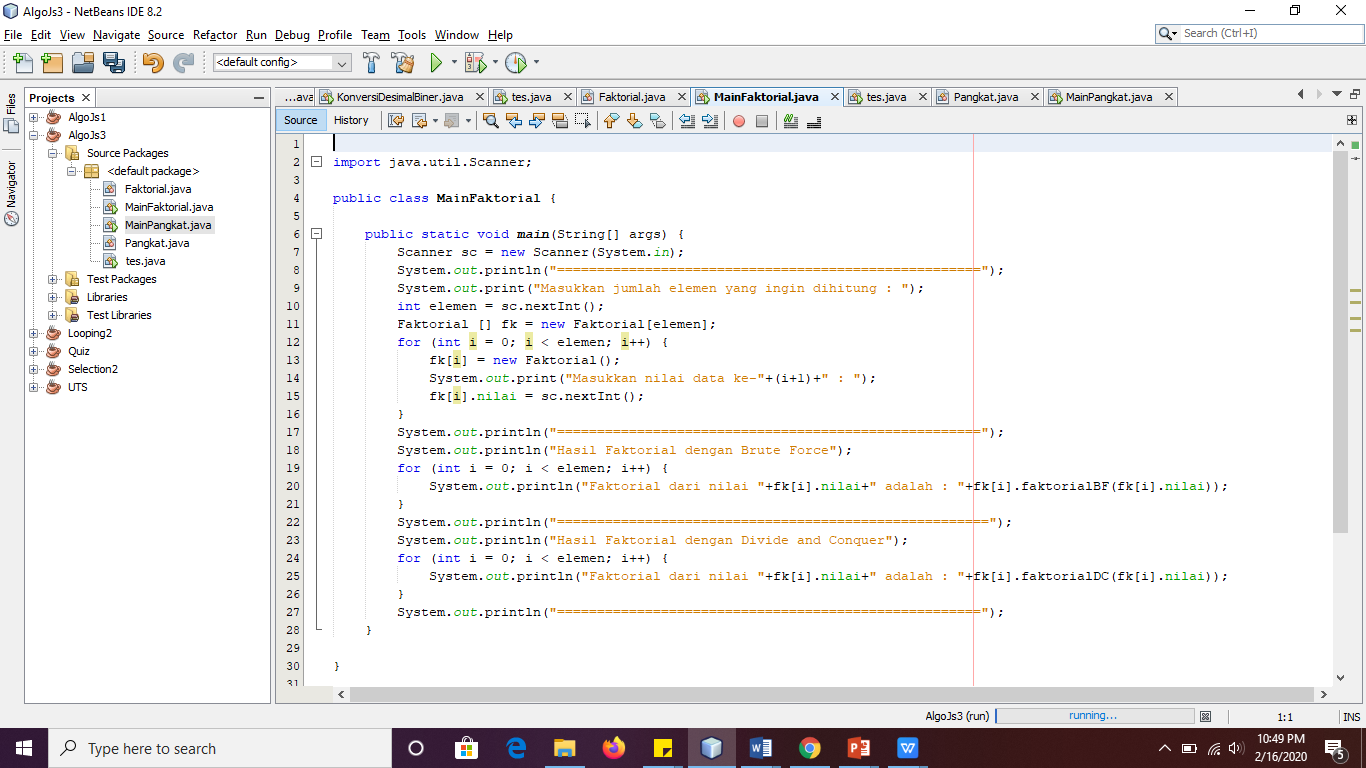
1. Tambahkan method faktorialDC() nilai



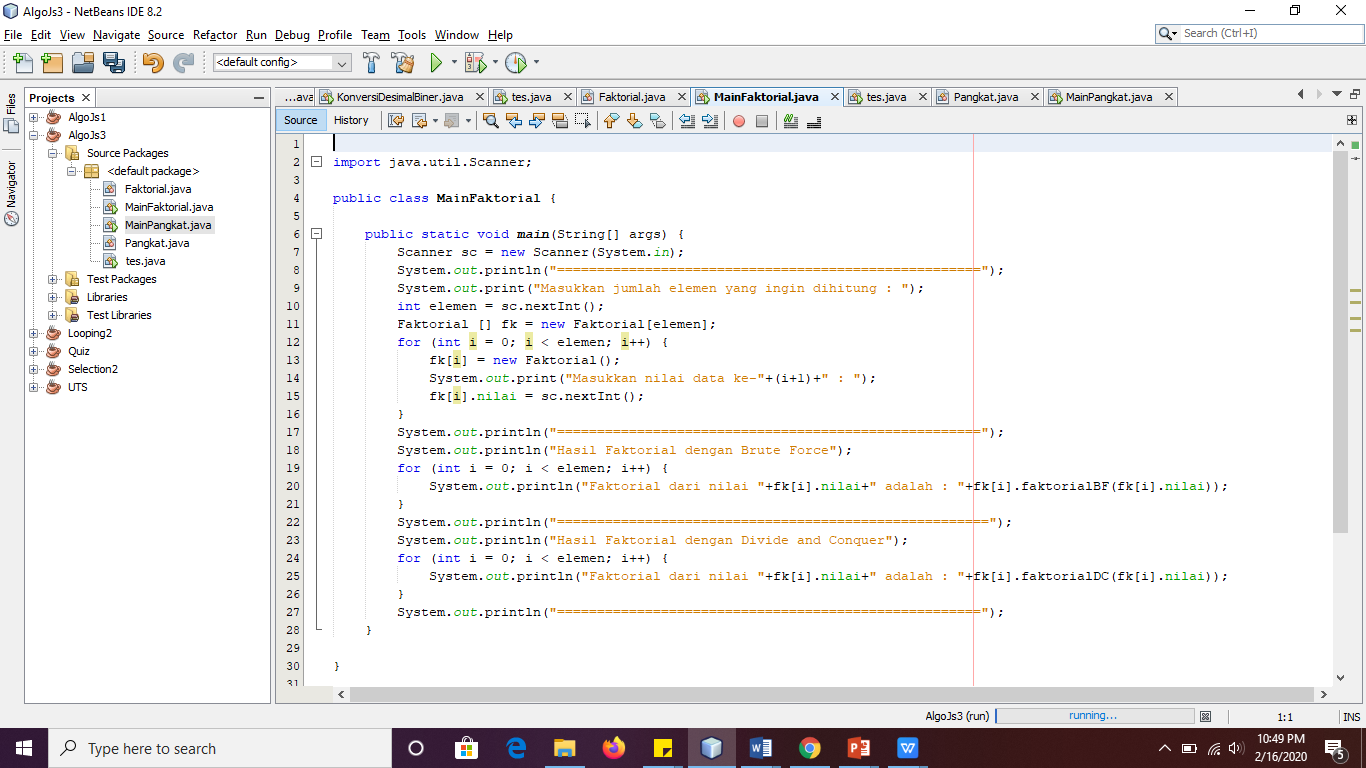
1. Coba jalankan (Run) class Faktorial dengan membuat class baru MainFaktorial.
2. Di dalam fungsi main sediakan komunikasi dengan user untuk menginputkan jumlah angka yang akan dicari nilai faktorialnya



* 1. Buat Array of Objek pada fungsi main, kemudian inputkan beberapa nilai yang akan dihitung faktorialnya

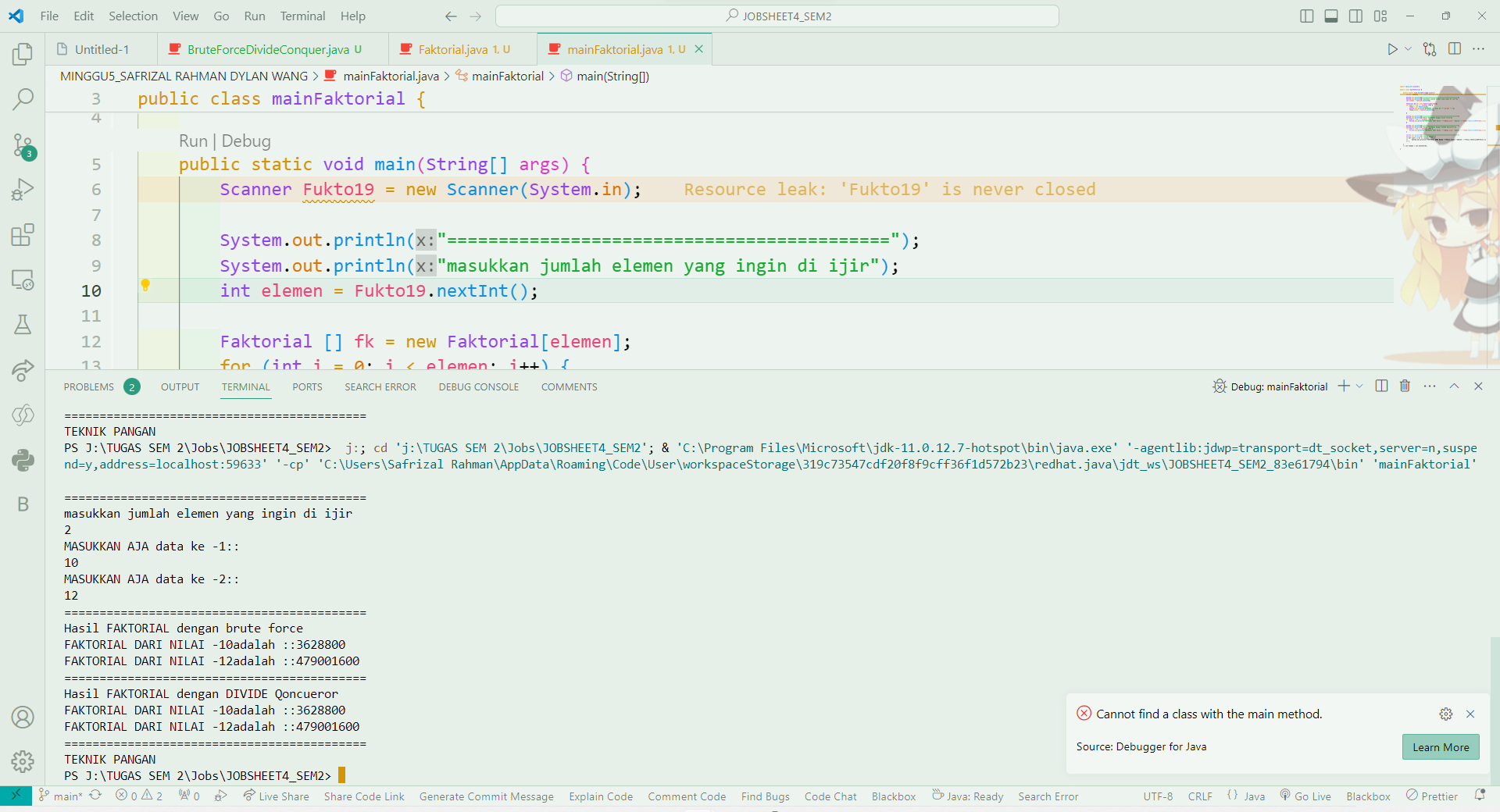


1. Tampilkan hasil pemanggilan method aktorialDC() dan faktorialBF()



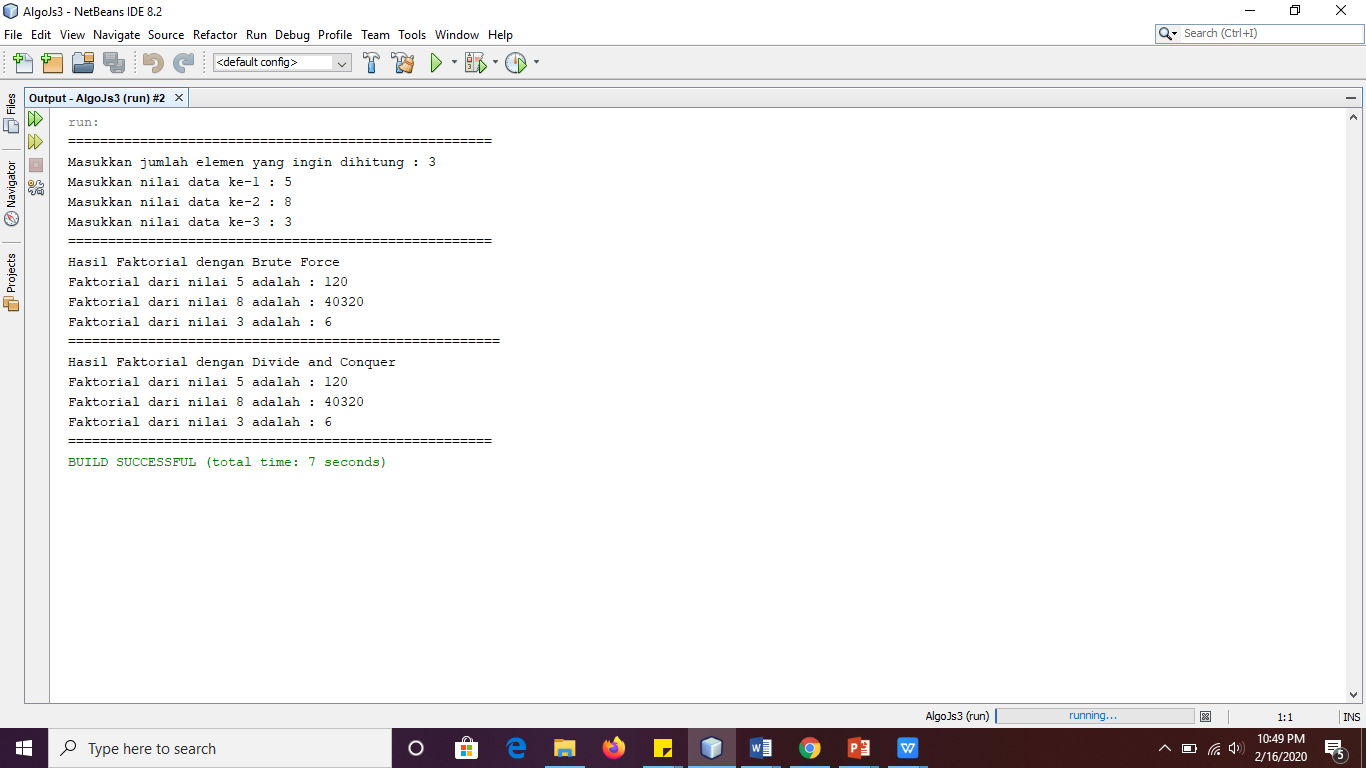
* 1. Pastikan program sudah berjalan dengan baik!

1. import java.util.Scanner;
2. public class mainFaktorial {
4. public static void main(String[] args) {
5. Scanner Fukto19 = new Scanner(System.in);
7. System.out.println("===========================================");
8. System.out.println("masukkan jumlah elemen yang ingin di ijir");
9. int elemen = Fukto19.nextInt();
10. Faktorial [] fk = new Faktorial[elemen];
11. for (int i = 0; i < elemen; i++) {
12. fk[i] = new Faktorial();
13. System.out.println("MASUKKAN AJA data ke -"+ (i+1)+ "::");
14. fk[i].nilai = Fukto19.nextInt();
16. }
18. System.out.println("===========================================");
19. System.out.println("Hasil FAKTORIAL dengan brute force");
20. for (int i = 0; i < elemen; i++) {
21. System.out.println("FAKTORIAL DARI NILAI -"+fk[i].nilai+ "adalah ::"+fk[i].faktorialBF(fk[i].nilai));
22. }
24. System.out.println("===========================================");
25. System.out.println("Hasil FAKTORIAL dengan DIVIDE Qoncueror");
26. for (int i = 0; i < elemen; i++) {
27. System.out.println("FAKTORIAL DARI NILAI -"+fk[i].nilai+ "adalah ::"+fk[i].faktorialDC(fk[i].nilai));
28. }
30. System.out.println("===========================================");
31. System.out.println("TEKNIK PANGAN");
32. *// for (int i = 0; i < elemen; i++) {*
33. *//     System.out.println("FAKTORIAL DARI NILAI -"+fk[i].nilai+ "adalah ::"+fk[i].faktorialBF(fk[i].nilai));*
34. *// }*
35. }
36. *// int elemen = int.nextInt(0);*
38. }



1. Verifikasi Hasil Percobaan

Cocokkan hasil compile kode program anda dengan gambar berikut ini.



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

**Dasar dari Algoritma Divide and Conquer untuk Perhitungan Faktorial: Dasar dari algoritma Divide and Conquer untuk penghitungan faktorial adalah memecah masalah menjadi submasalah yang lebih kecil hingga menjadi cukup sederhana untuk diselesaikan secara langsung. Untuk perhitungan faktorial, masalah menghitung n! (n faktorial) dibagi menjadi masalah yang lebih kecil untuk menghitung (n-1)!, (n-2)!, dan seterusnya, hingga kita mencapai 1! atau 0!, yang keduanya didefinisikan sebagai 1. Solusi untuk submasalah ini kemudian digabungkan untuk memberikan solusi ke masalah asli.**

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

**Implementasi Divide and Conquer dalam Perhitungan Faktorial: Dalam kode yang disediakan, algoritma Divide and Conquer diimplementasikan dalam metode faktorialDC(). Namun, penting untuk dicatat bahwa tiga tahap khas Divide and Conquer (bagi, taklukkan, gabungkan) tidak sepenuhnya berlaku di sini. Hal ini karena perhitungan faktorial pada dasarnya tidak melibatkan langkah 'gabungkan'. Berikut cara kerjanya dalam kode:**

**Membagi: Ini dilakukan dengan memanggil faktorialDC() secara rekursif dengan argumen yang lebih kecil (n-1).**

**Conquer: Ketika kasus dasar (n <= 1) tercapai, fungsi ini mengembalikan 1.**

**Menggabungkan: Seperti yang telah disebutkan, tidak ada langkah 'gabungkan' secara eksplisit. Hasilnya secara implisit digabungkan dengan operasi perkalian dalam pemanggilan rekursif (n faktorialDC(n-1)).**

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

**Ya, Anda dapat menggunakan jenis perulangan lain, seperti perulangan while, untuk mencapai hasil yang sama. Berikut adalah contoh bagaimana Anda dapat memodifikasi metode faktorialBF() untuk menggunakan perulangan while:**

   public long faktorialBF(int n) {

        long result = 1;

        int i = 1;

        while (i <= n) {

            result = i;

            i++;

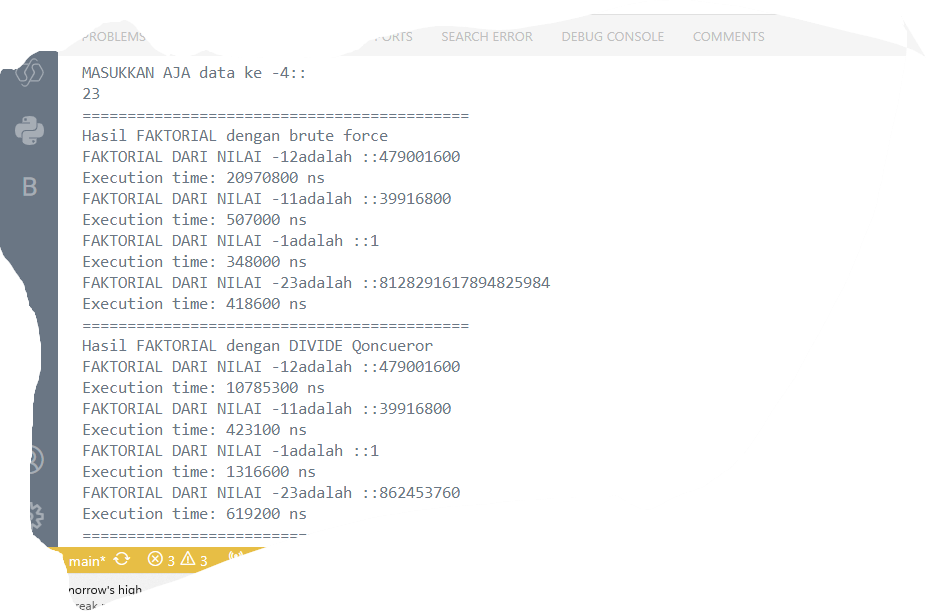
        }

        return result;

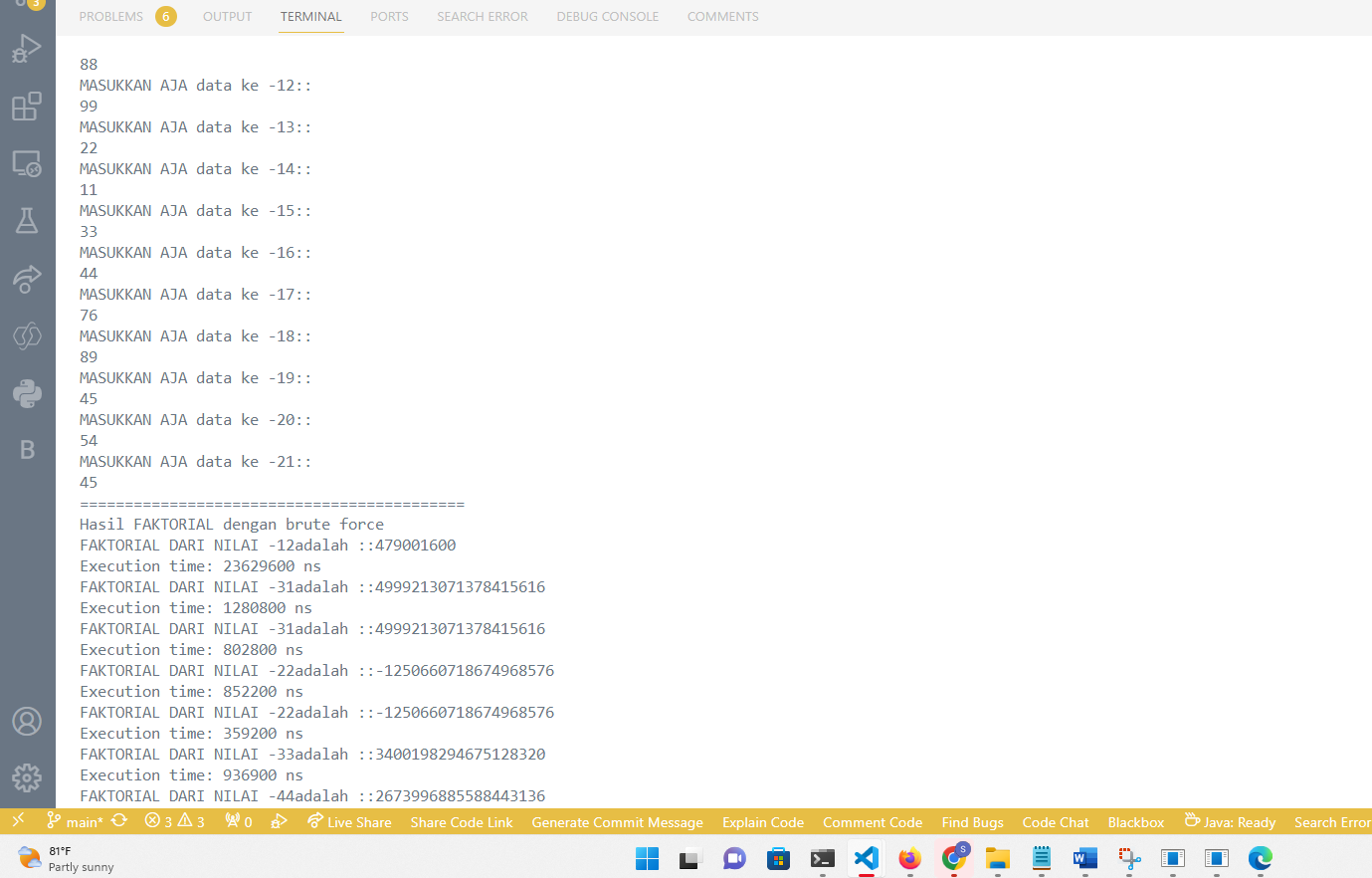
    }

**Kode ini melakukan hal yang sama dengan metode faktorialBF() yang asli: kode ini menghitung faktorial dari n dengan mengalikan hasilnya secara berulang-ulang dengan setiap bilangan bulat dari 1 hingga n.**

1. Tambahkan pegecekan waktu eksekusi kedua jenis method tersebut!
2. public class mainFaktorial {
3. public static void main(String[] args) {
4. Scanner Fukto19 = new Scanner(System.in);
6. System.out.println("===========================================");
7. System.out.println("masukkan jumlah elemen yang ingin di ijir");
8. int elemen = Fukto19.nextInt();
9. Faktorial [] fk = new Faktorial[elemen];
10. for (int i = 0; i < elemen; i++) {
11. fk[i] = new Faktorial();
12. System.out.println("MASUKKAN AJA data ke -"+ (i+1)+ "::");
13. fk[i].nilai = Fukto19.nextInt();
14. }
16. System.out.println("===========================================");
17. System.out.println("Hasil FAKTORIAL dengan brute force");
18. for (int i = 0; i < elemen; i++) {
19. long start = System.nanoTime();
20. System.out.println("FAKTORIAL DARI NILAI -"+fk[i].nilai+ "adalah ::"+fk[i].faktorialBF(fk[i].nilai));
21. long end = System.nanoTime();
22. System.out.println("Execution time: " + (end - start) + " ns");
23. }
25. System.out.println("===========================================");
26. System.out.println("Hasil FAKTORIAL dengan DIVIDE Qoncueror");
27. for (int i = 0; i < elemen; i++) {
28. long start = System.nanoTime();
29. System.out.println("FAKTORIAL DARI NILAI -"+fk[i].nilai+ "adalah ::"+fk[i].faktorialDC(fk[i].nilai));
30. long end = System.nanoTime();
31. System.out.println("Execution time: " + (end - start) + " ns");
32. }
34. System.out.println("===========================================");
35. System.out.println("TEKNIK PANGAN");
36. }
37. }



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

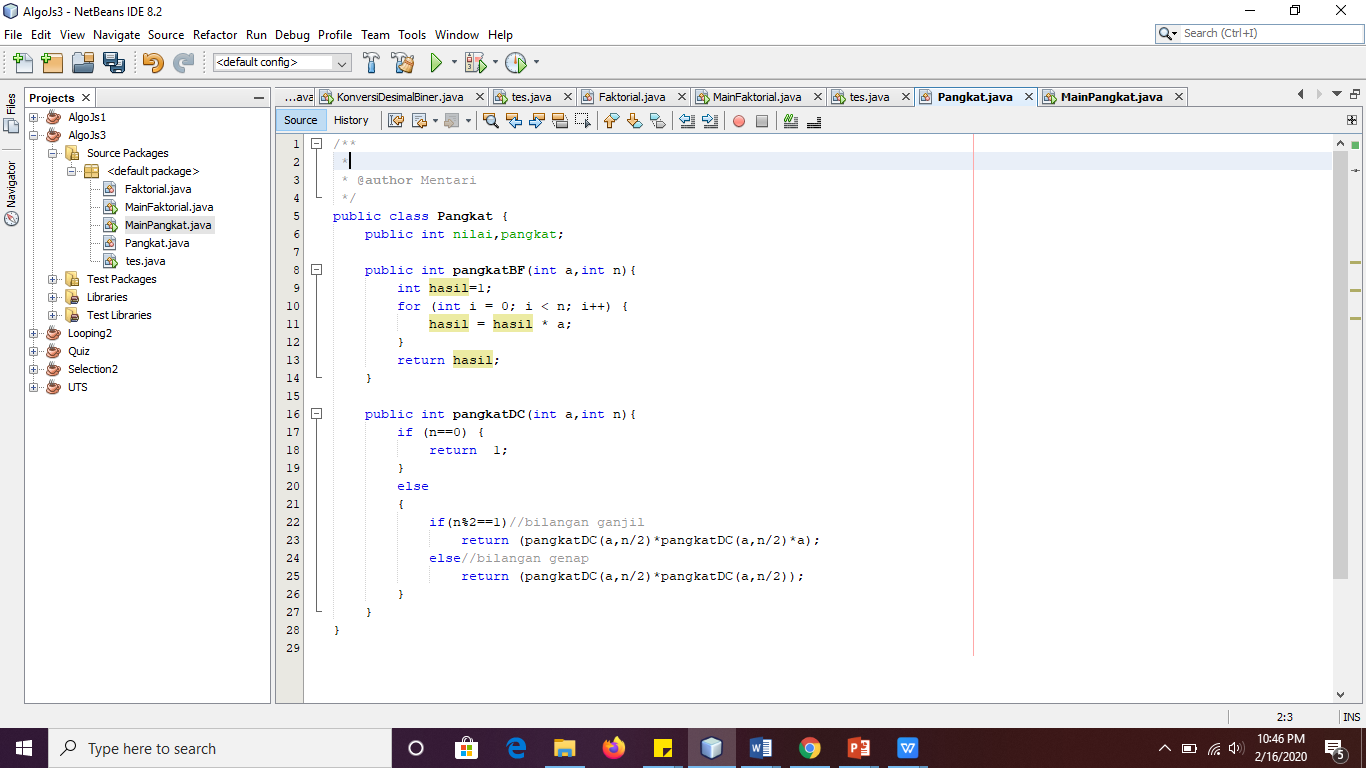


1. Menghitung Hasil Pangkat dengan Algoritma Brute Force dan Divide and Conquer

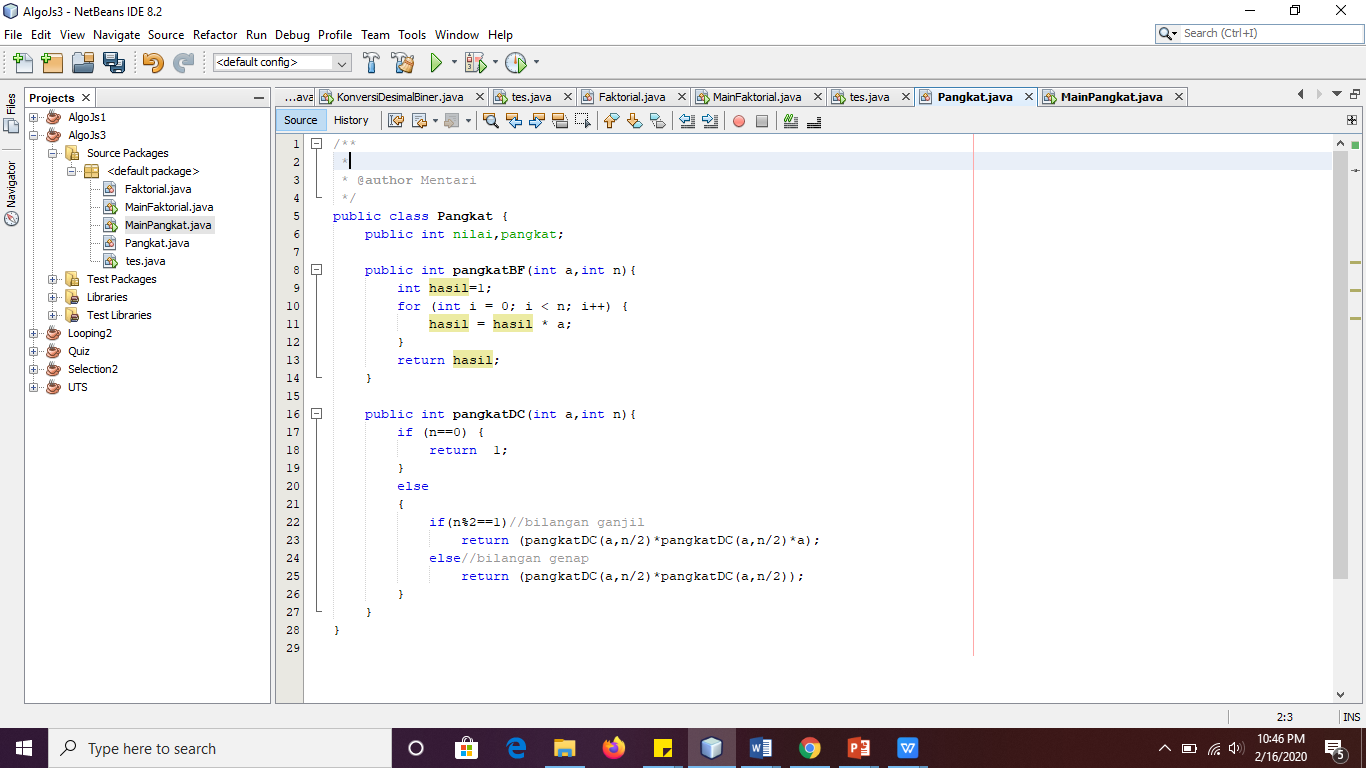
Pada praktikum ini kita akan membuat program class dalam Java. Untuk menghitung nil ai pangkat suatu angka menggunakan 2 jenis algoritma, Brute Force dan Divide and Conquer.

* + 1. Langkah-langkah Percobaan

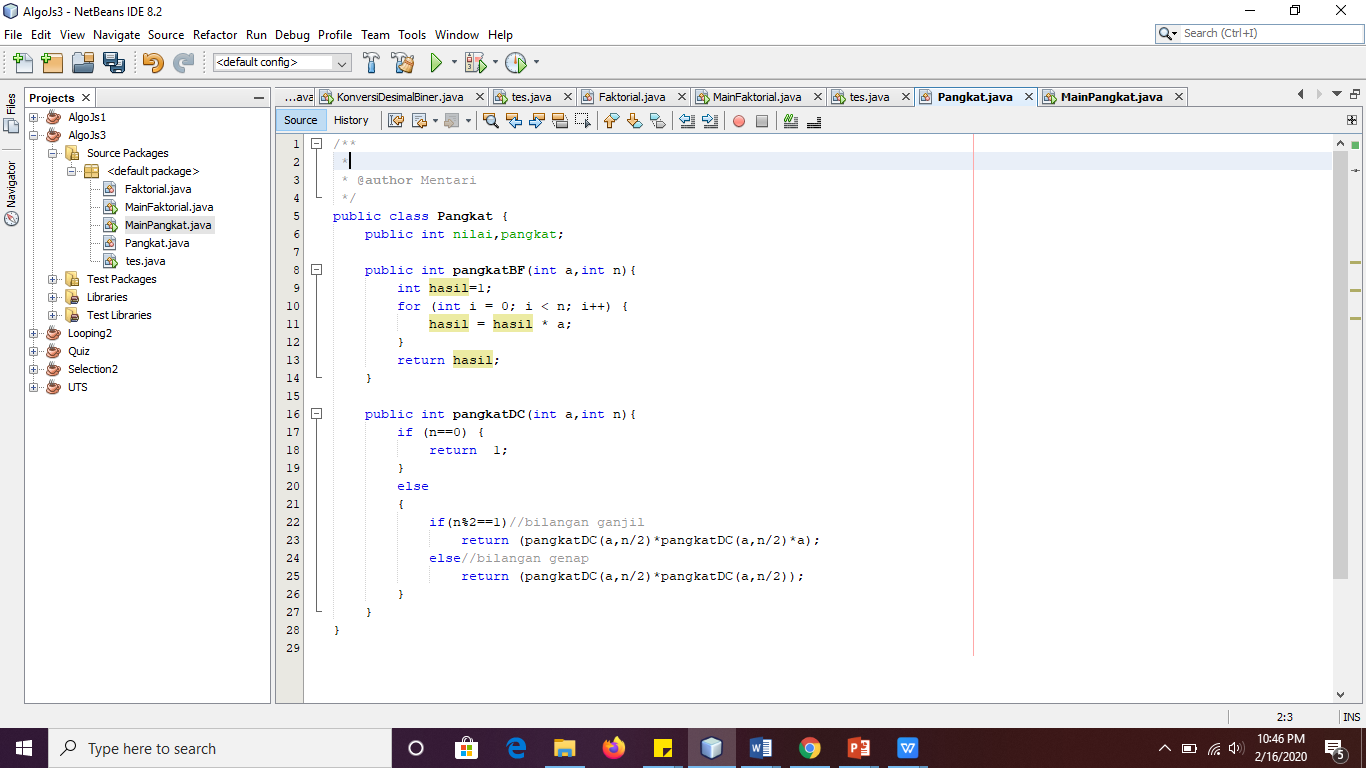
1. Di dalam paket **minggu5**, buatlah class baru dengan nama **Pangkat**. Dan di dalam class **Pangkat** tersebut, buat atribut angka yang akan dipangkatkan sekaligus dengan angka pemangkatnya



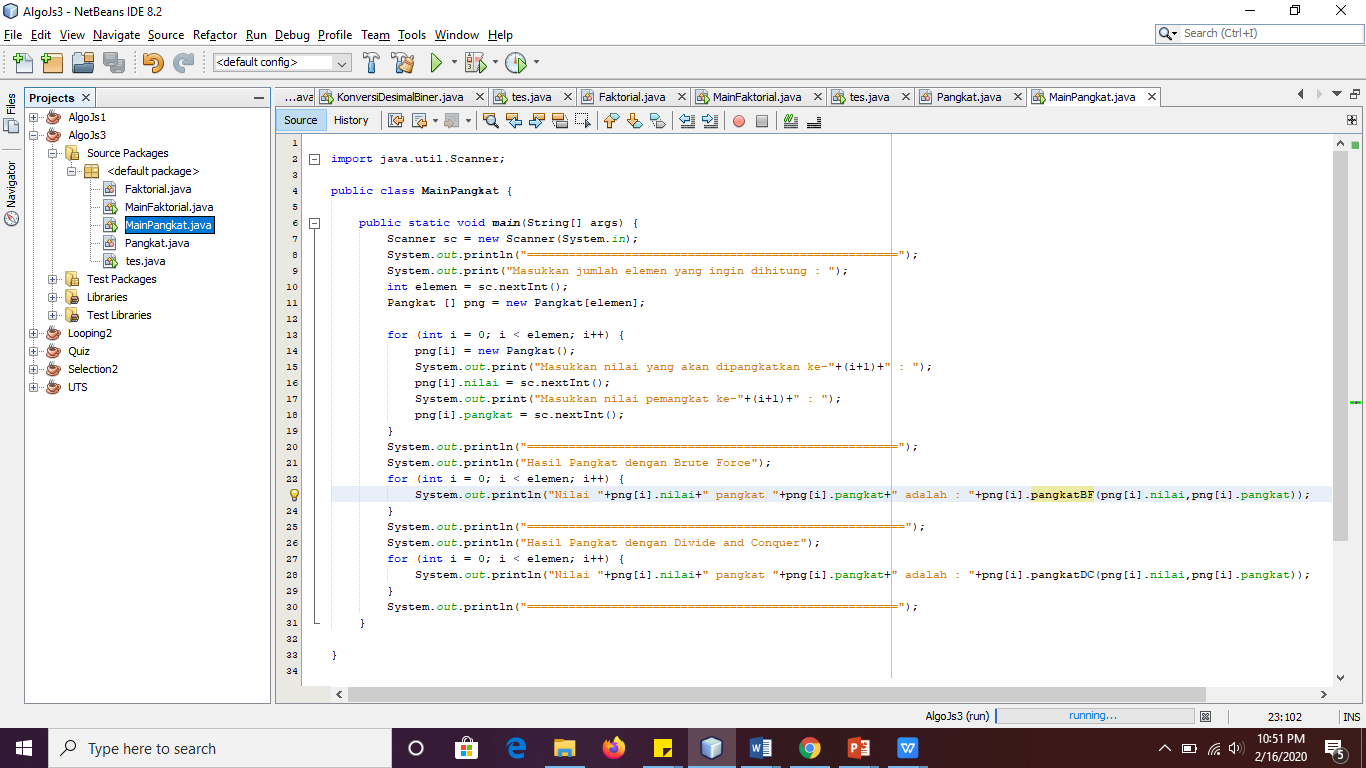
1. Pada class Pangkat tersebut, tambahkan method PangkatBF()



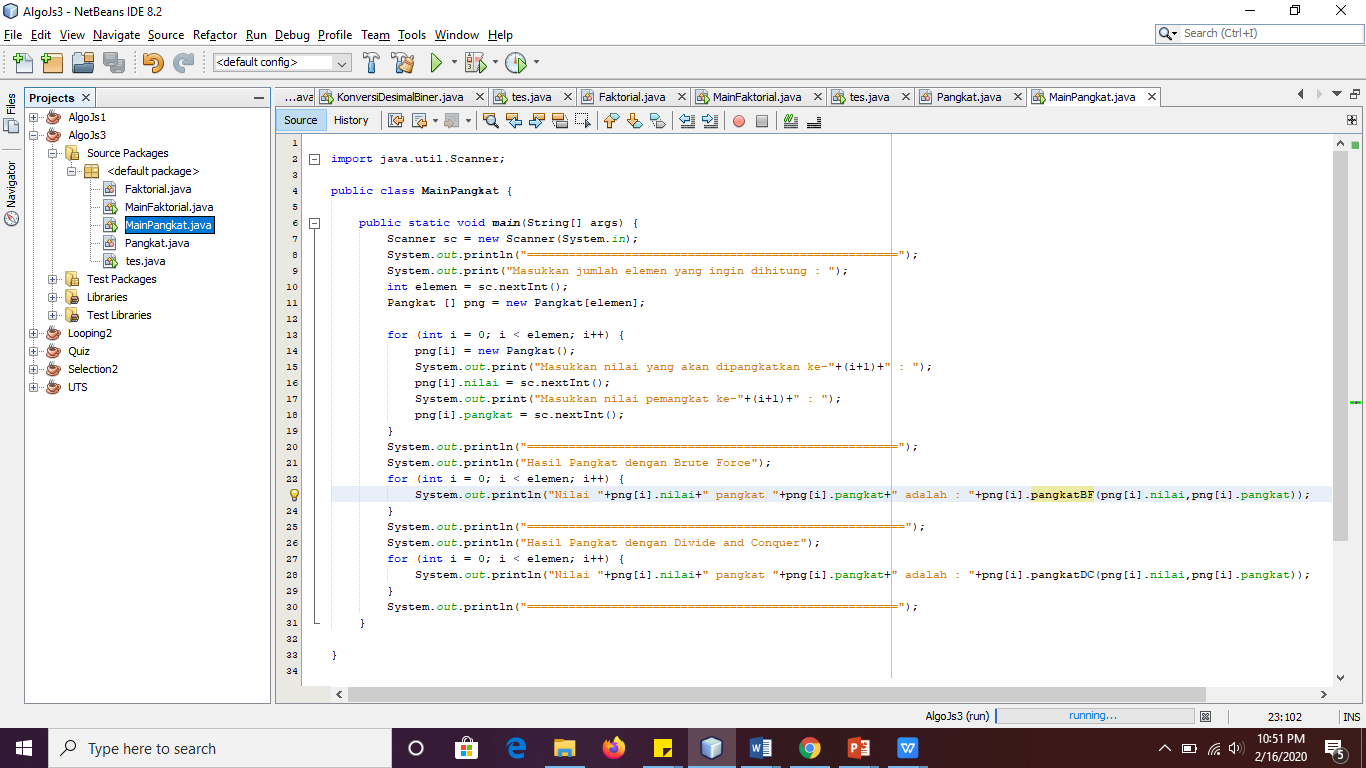
1. Pada class Pangkat juga tambahkan method PangkatDC()



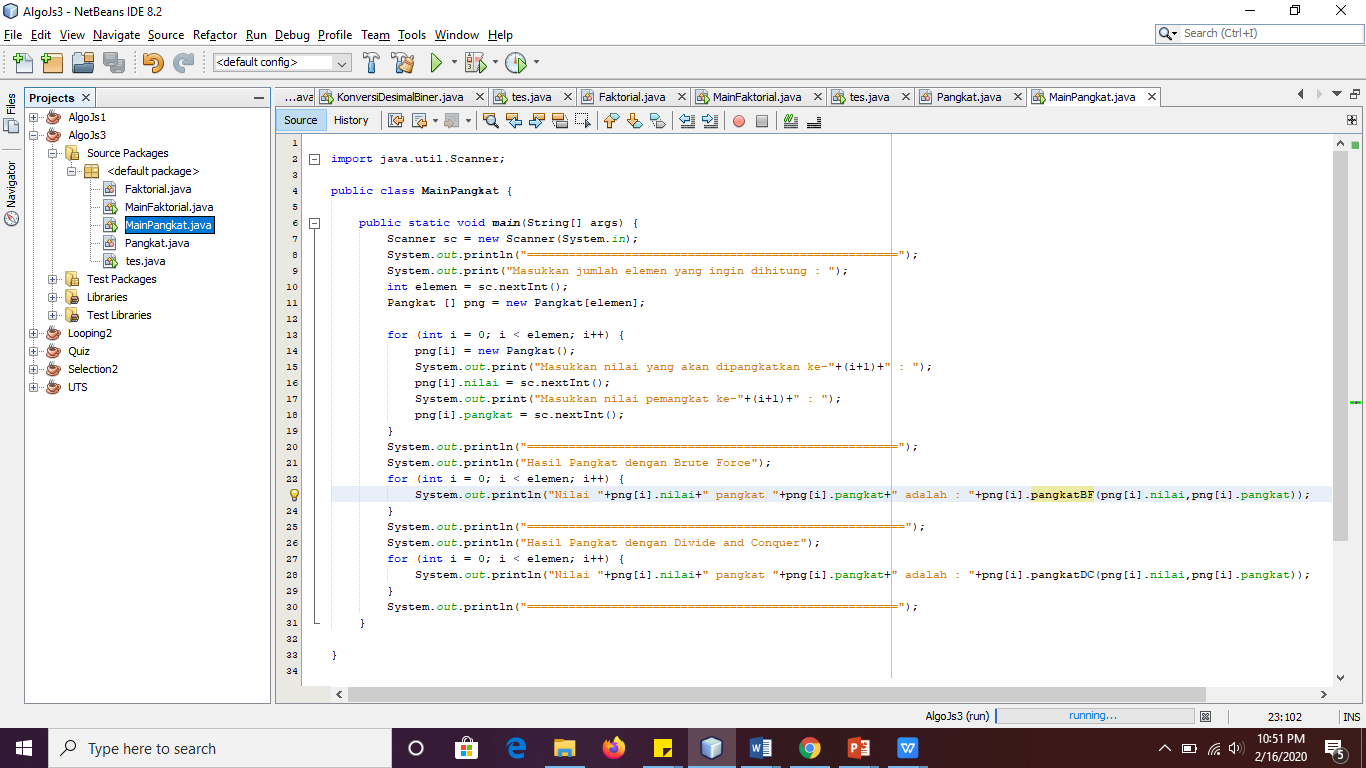
1. Perhatikan apakah sudah tidak ada kesalahan yang muncul dalam pembuatan class Pangkat
2. Selanjutnya buat class baru yang di dalamnya terdapat method main. Class tersebut dapat dinamakan MainPangkat. Tambahkan kode pada class main untuk menginputkan jumlah nilai yang akan dihitung pangkatnya.



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



import java.util.Scanner;

public class mainPangkat {

    public static void main(String[] args) {

        Scanner pangto19 = new Scanner(System.in);

        System.out.println("===========================================");

        System.out.println("masukkan jumlah elemen yang ingin di ijir");

        int elemen = pangto19.nextInt();

        pangkatpangkatan [] punk = new pangkatpangkatan[elemen];

        for (int i = 0; i < elemen;i++) {

            punk[i] = new pangkatpangkatan();

            System.out.println("Masukkan nilai yang akan dipangkatkan ke "+ (i+1)+ "::::");

            punk[i].nilai = pangto19.nextInt();

            System.out.println("Masukkan nilai yang PEMANGKAT KE"+ (i+1)+ "::::");

            punk[i].pangkat = pangto19.nextInt(); // This line was corrected

        }

        System.out.println("===========================================");

        System.out.println("Hasil pangkat dengan brute force");

        for (int i = 0; i < elemen; i++) {

            long start = System.nanoTime();

            System.out.println("pangkat DARI NILAI -"+punk[i].nilai+"pangkat " +punk[i].pangkat +"adalah ::"+punk[i].pangkatBF(punk[i].nilai,punk[i].pangkat));

            long end = System.nanoTime();

            System.out.println("Execution time: " + (end - start) + " ns");

        }

        System.out.println("===========================================");

        System.out.println("Hasil pangkat dengan DIVIDEN force");

        for (int i = 0; i < elemen; i++) {

            long start = System.nanoTime();

            System.out.println("pangkat DARI NILAI -"+punk[i].nilai+"pangkat " +punk[i].pangkat +"adalah ::"+punk[i].pangkatDC(punk[i].nilai,punk[i].pangkat));

            long end = System.nanoTime();

            System.out.println("Execution time: " + (end - start) + " ns");

        }

        System.out.println("===========================================");

        System.out.println("TEKNIK PANGAN");

    }

    //     System.out.println("===========================================");

    //     System.out.println("masukkan jumlah elemen yang ingin di ijir");

    //     int elemen = pangto19.nextInt();

    //     pangkatpangkatan [] punk = new pangkatpangkatan[elemen];

    //     for (int i = 0; i < elemen;i++) {

    //         punk[i] = new pangkatpangkatan();

    //         System.out.println("Masukkan nilai yang akan dipangkatkan ke "+ (i+1)+ "::::");

    //         punk[i].nilai = pangto19.nextInt();

    //         System.out.println("Masukkan nilai yang PEMANGKAT KE"+ (i+1)+ "::::");

    //         punk[i].nilai = pangto19.nextInt();

    //         //#endregion

    //     }

    //     System.out.println("===========================================");

    //     System.out.println("Hasil pangkat dengan brute force");

    //     for (int i = 0; i < elemen; i++) {

    //         long start = System.nanoTime();

    //         System.out.println("pangkat DARI NILAI -"+punk[i].nilai+"pangkat " +punk[i].pangkat +"adalah ::"+punk[i].pangkatBF(punk[i].nilai,punk[i].pangkat));

    //         long end = System.nanoTime();

    //         System.out.println("Execution time: " + (end - start) + " ns");

    //     }

    //     System.out.println("===========================================");

    //     System.out.println("Hasil pangkat dengan DIVIDEN force");

    //     for (int i = 0; i < elemen; i++) {

    //         long start = System.nanoTime();

    //         System.out.println("pangkat DARI NILAI -"+punk[i].nilai+"pangkat " +punk[i].pangkat +"adalah ::"+punk[i].pangkatDC(punk[i].nilai,punk[i].pangkat));

    //         long end = System.nanoTime();

    //         System.out.println("Execution time: " + (end - start) + " ns");

    //     }

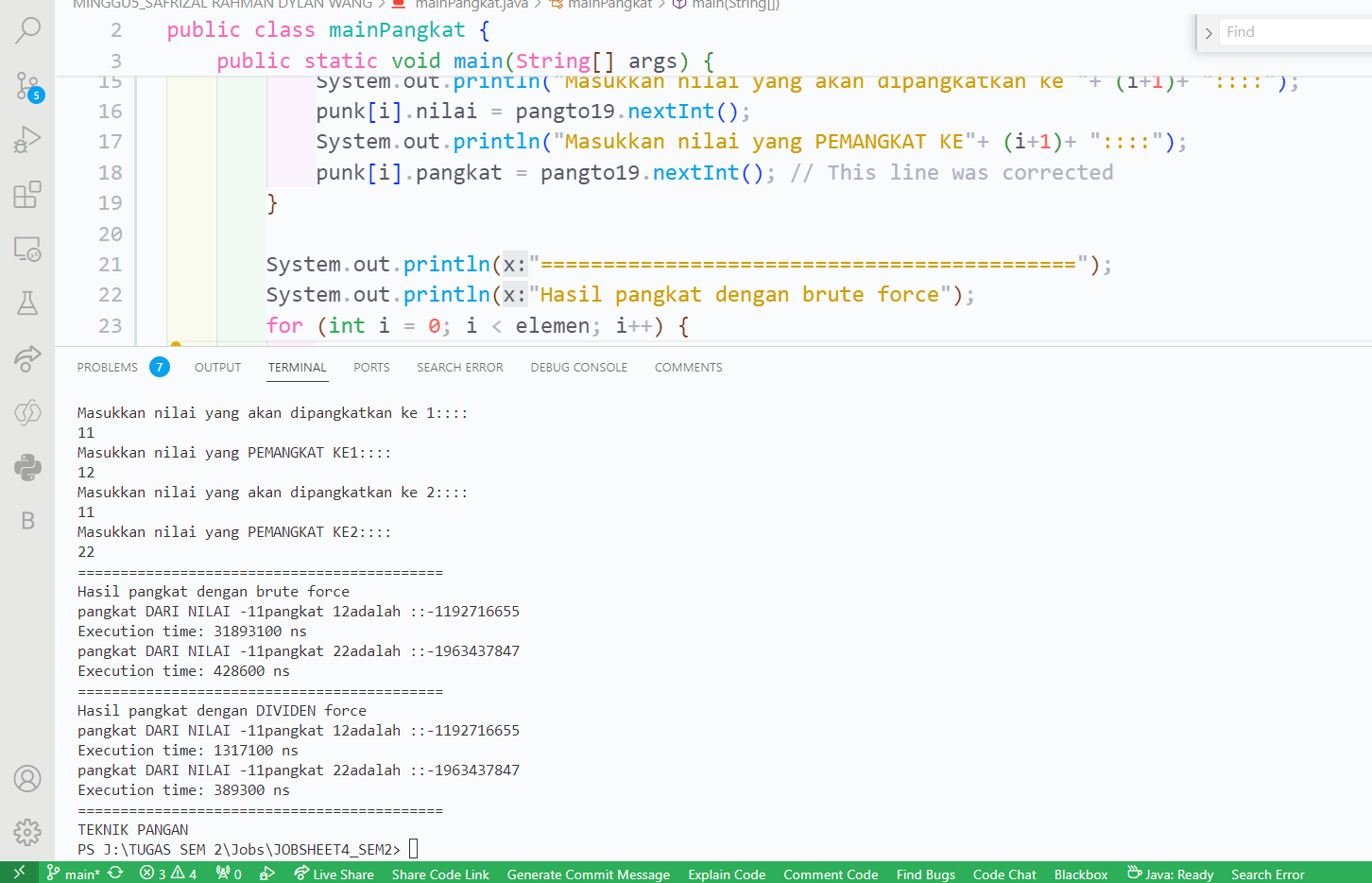
    //     System.out.println("===========================================");

    //     System.out.println("TEKNIK PANGAN");

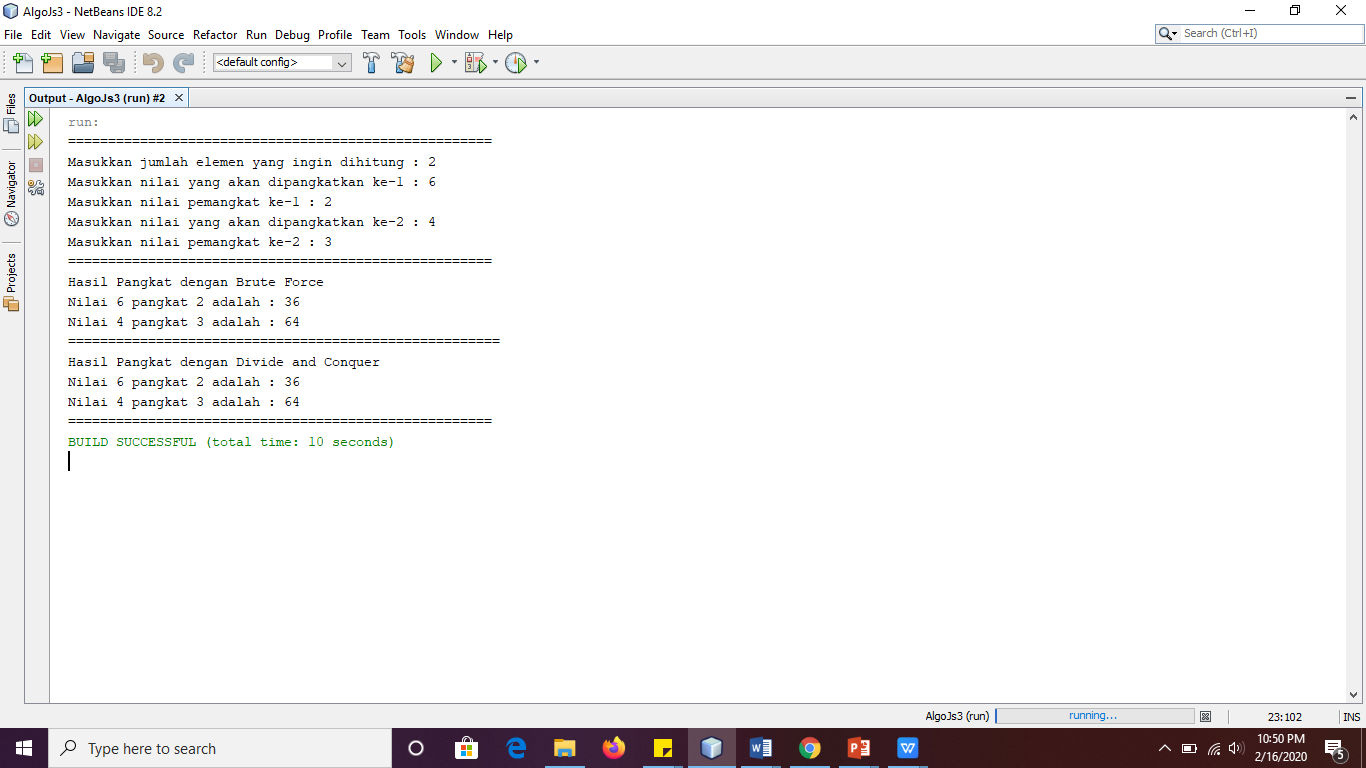
    // }

    }

* + 1. Verifikasi Hasil Percobaan



Pastikan output yang ditampilkan sudah benar seperti di bawah ini.

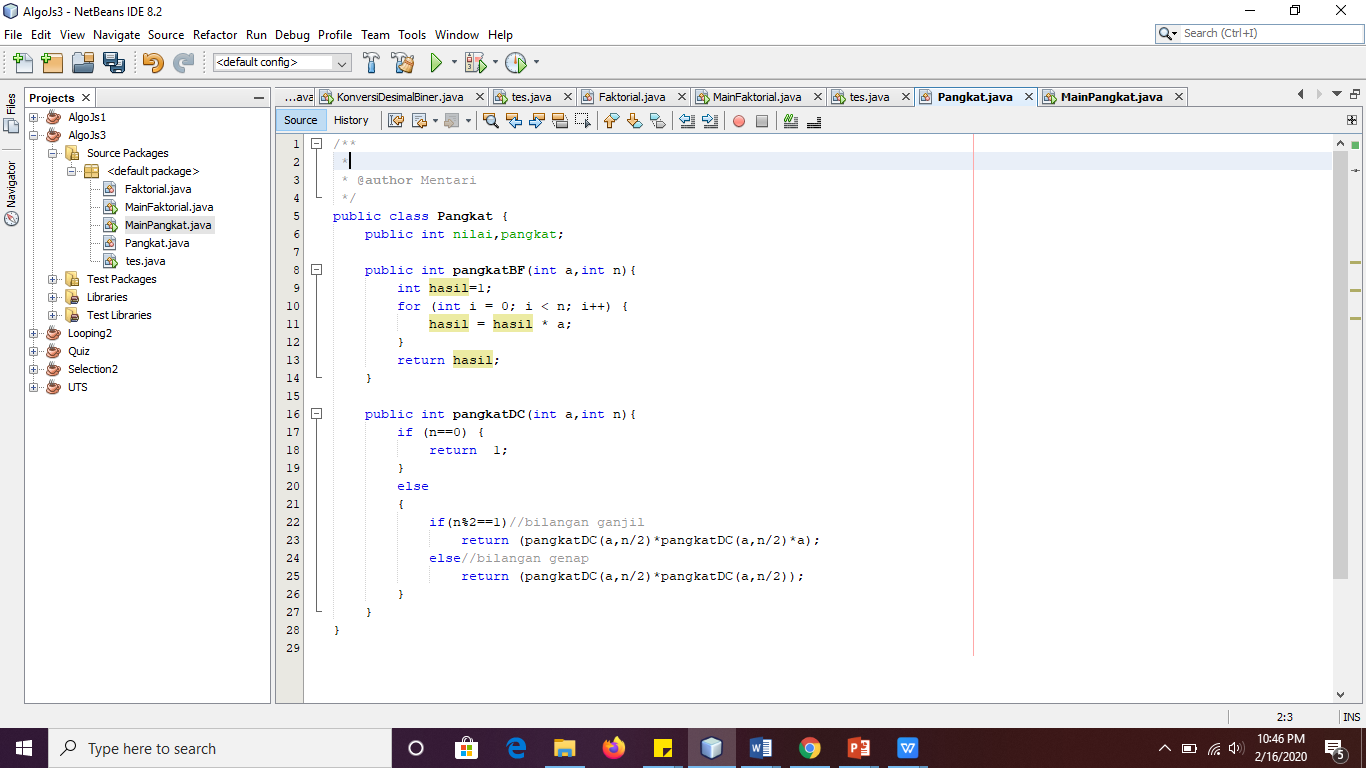


* + 1. Pertanyaan

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

**PangkatBF() vs PangkatDC(): Metode `PangkatBF () ` menghitung pangkat dari sebuah angka menggunakan pendekatan brute force. Metode ini mengalikan angka dasar dengan sendirinya untuk beberapa kali yang ditentukan oleh eksponen. Di sisi lain, `PangkatDC()` menggunakan pendekatan Bagi dan Taklukkan untuk menghitung pangkat suatu bilangan. Metode ini memecah masalah menjadi submasalah yang lebih kecil dan menggabungkan hasilnya untuk mendapatkan jawaban akhir. Metode ini lebih efisien karena mengurangi jumlah perkalian.**

1. Pada method PangkatDC() terdapat potongan program sebagai berikut:



Jelaskan arti potongan kode tersebut

**Potongan Kode di PangkatDC(): Potongan kode ini memeriksa apakah eksponen `n` genap atau ganjil. Jika `n` ganjil, maka ia akan menghitung pangkat setengah dari `n`, mengkuadratkannya, lalu mengalikannya dengan basis `a`. Jika `n` genap, maka ia akan menghitung pangkat setengah dari `n` dan mengkuadratkannya. Pendekatan ini mengurangi jumlah perkalian.**

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

**Ya, langkah penggabungan disertakan di dalam kode. Hasil dari sub-sub masalah digabungkan ketika metode ini mengembalikan `(pangkatDC(a, n/2)pangkatDC(a, n/2)a)` atau `(pangkatDC(a, n/2)pangkatDC(a, n/2))`.**

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

**`pangkatpangkat` untuk menyertakan konstruktor yang menginisialisasi atribut `nilai` dan `pangkat`. Berikut adalah cara untuk melakukannya:**

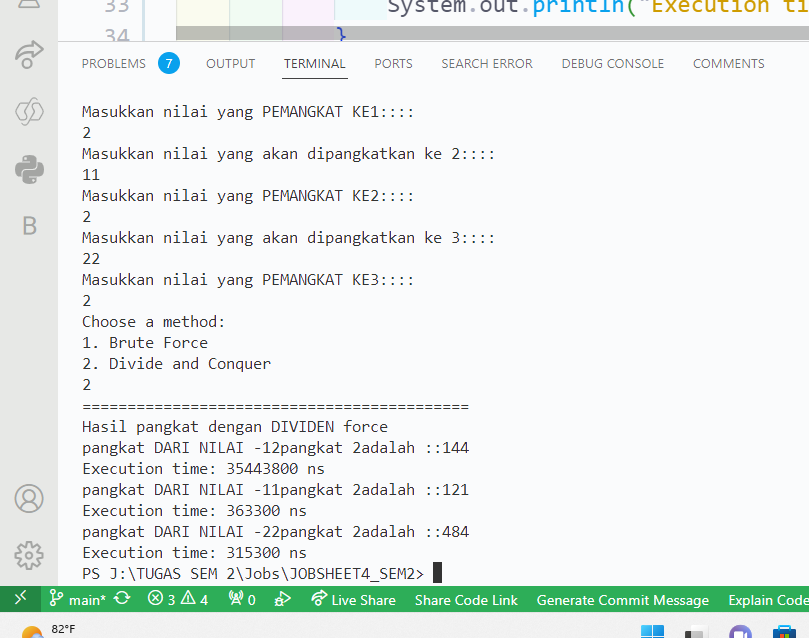
   public pangkatpangkatan(int nilai, int pangkat) {

        this.nilai = nilai;

        this.pangkat = pangkat;

    }

1. Tambahkan menu agar salah satu method yang terpilih saja yang akan dijalankan!
2. System.out.println("Choose a method:");
3. System.out.println("1. Brute Force");
4. System.out.println("2. Divide and Conquer");
5. int choice = pangto19.nextInt();
6. if (choice == 1) {
7. System.out.println("===========================================");
8. System.out.println("Hasil pangkat dengan brute force");
9. for (int i = 0; i < elemen; i++) {
10. long start = System.nanoTime();
11. System.out.println("pangkat DARI NILAI -"+punk[i].nilai+"pangkat " +punk[i].pangkat +"adalah ::"+punk[i].pangkatBF(punk[i].nilai,punk[i].pangkat));
12. long end = System.nanoTime();
13. System.out.println("Execution time: " + (end - start) + " ns");
14. }
15. // Run brute force method
16. } else if (choice == 2) {
17. System.out.println("===========================================");
18. System.out.println("Hasil pangkat dengan DIVIDEN force");
19. for (int i = 0; i < elemen; i++) {
20. long start = System.nanoTime();
21. System.out.println("pangkat DARI NILAI -"+punk[i].nilai+"pangkat " +punk[i].pangkat +"adalah ::"+punk[i].pangkatDC(punk[i].nilai,punk[i].pangkat));
22. long end = System.nanoTime();
23. System.out.println("Execution time: " + (end - start) + " ns");
24. }
25. // Run divide and conquer method
26. } else {
27. System.out.println("Invalid choice");
28. System.out.println("===========================================");
29. System.out.println("TEKNIK PANGAN");
30. }

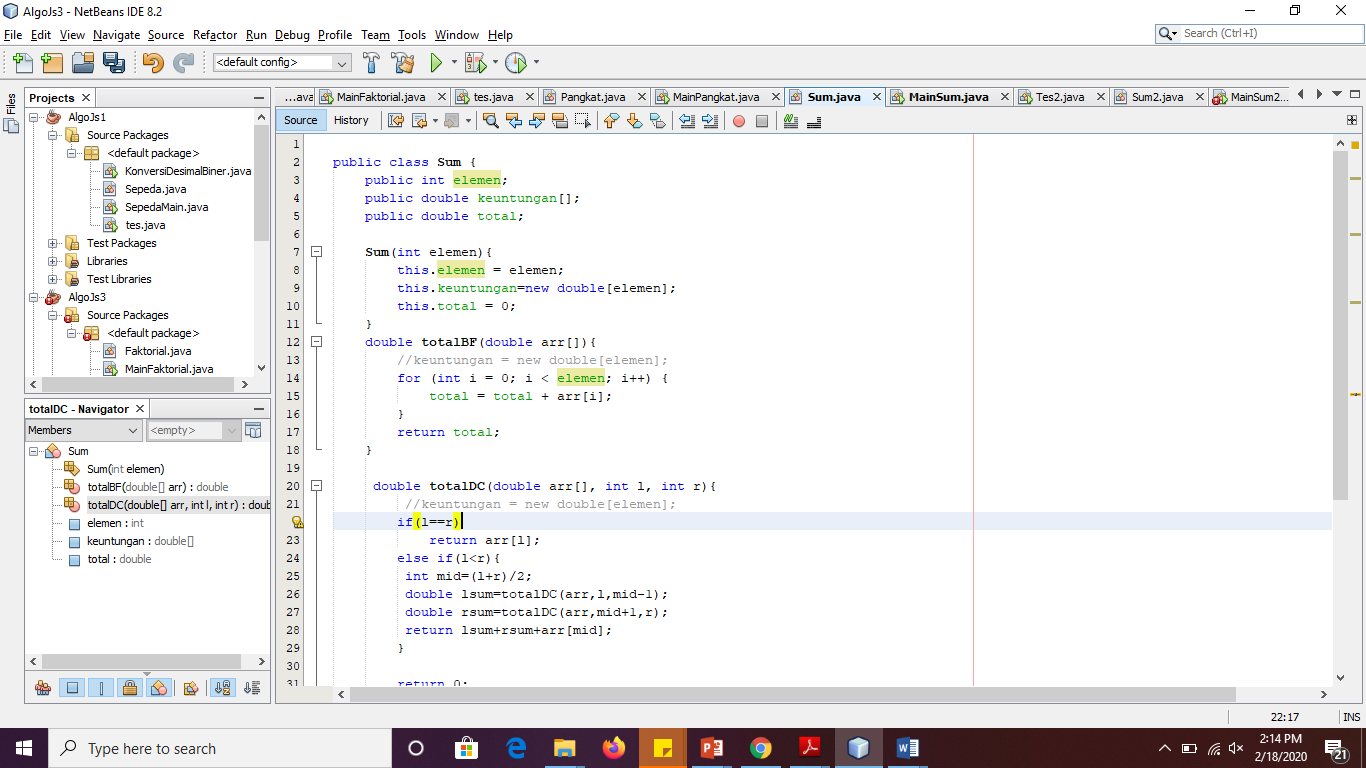


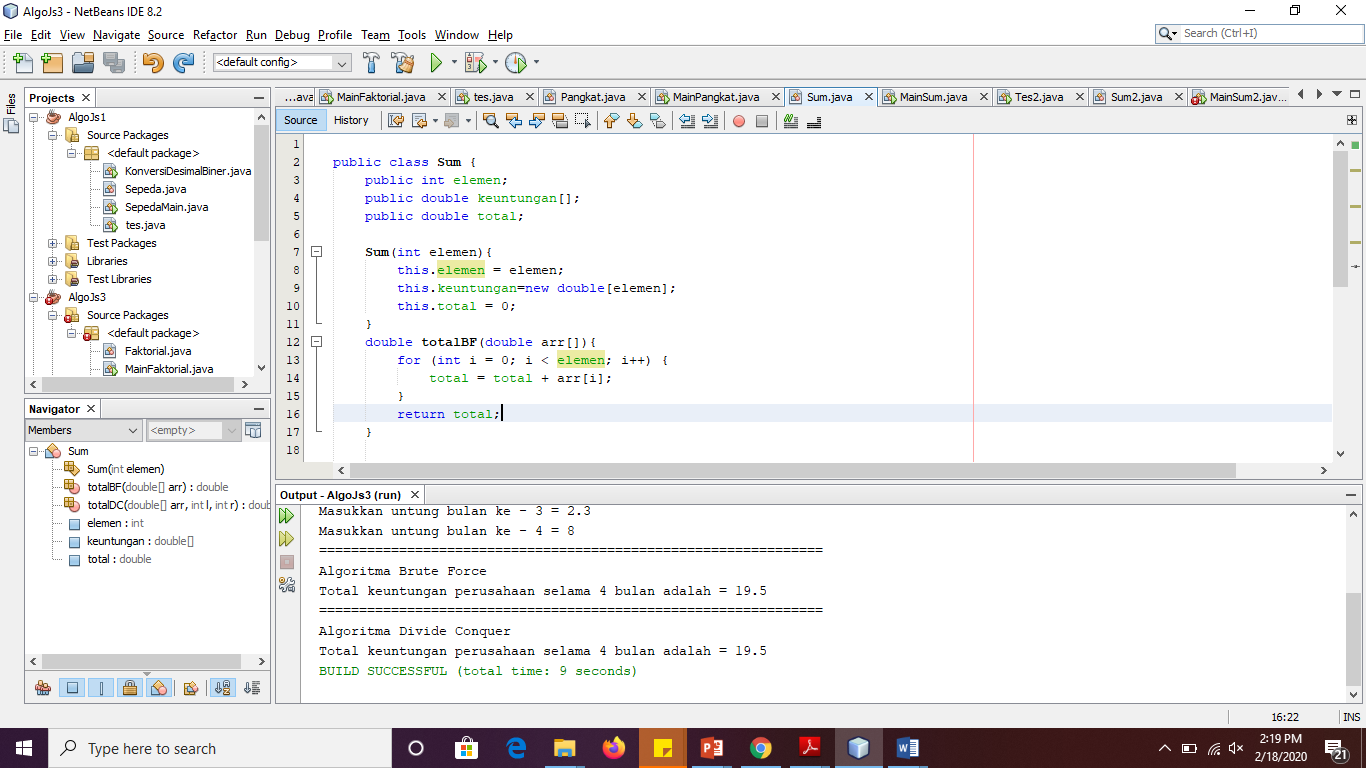
1. Menghitung Sum Array dengan Algoritma Brute Force dan Divide and Conquer

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

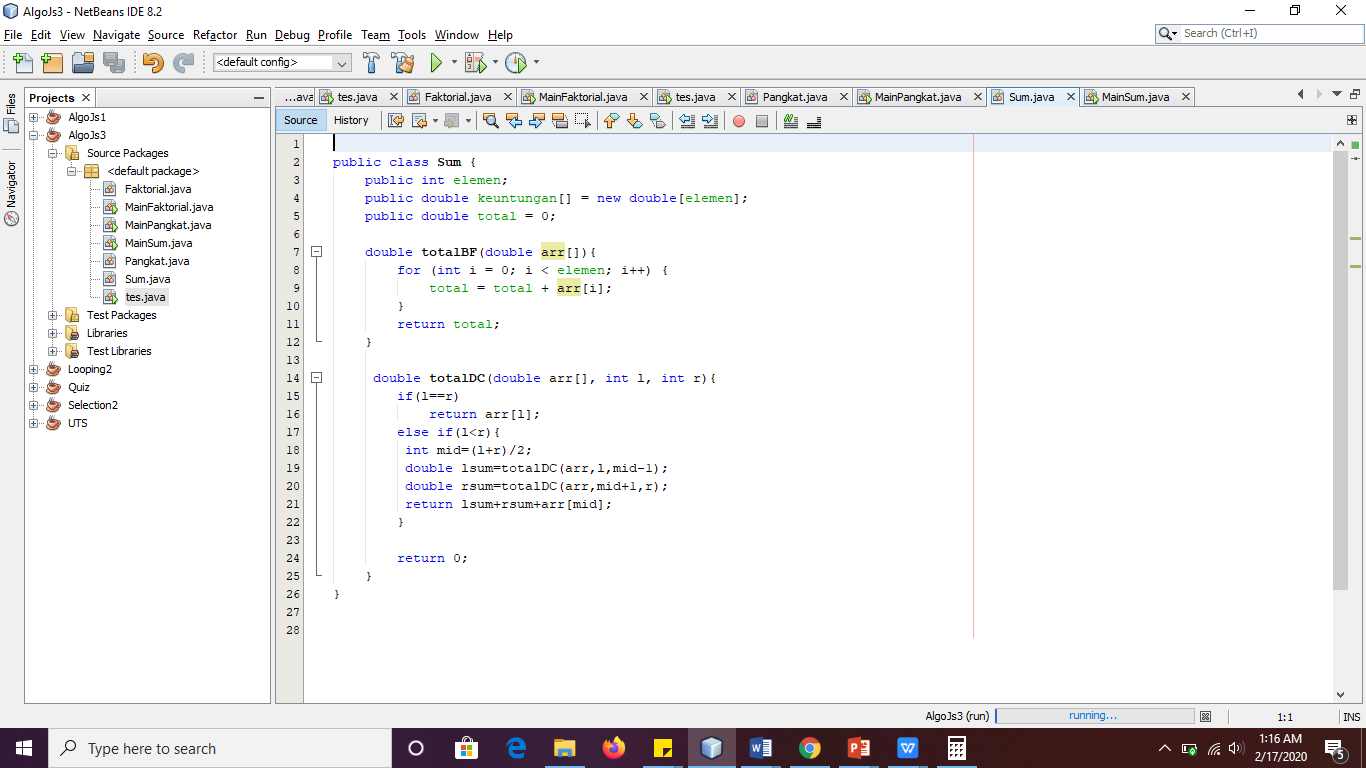
* + 1. Langkah-langkah Percobaan

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.

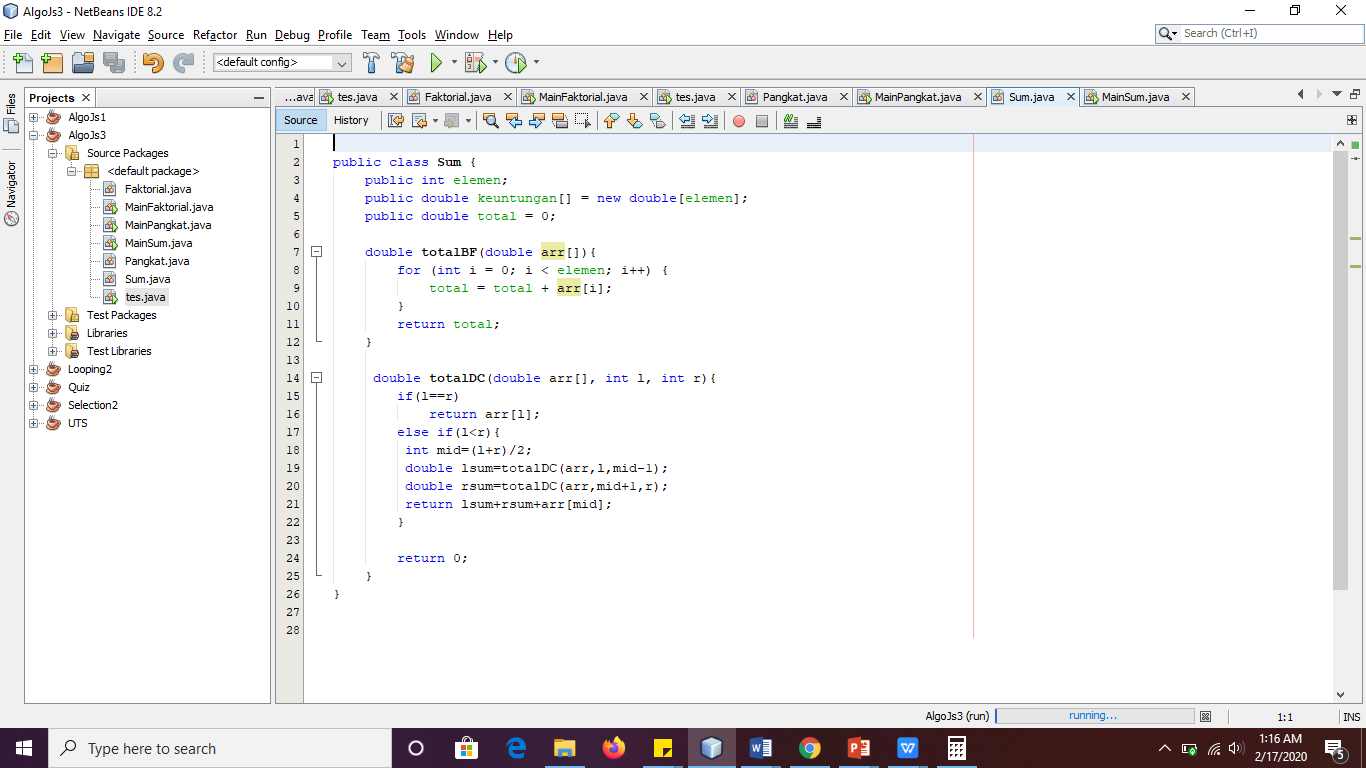




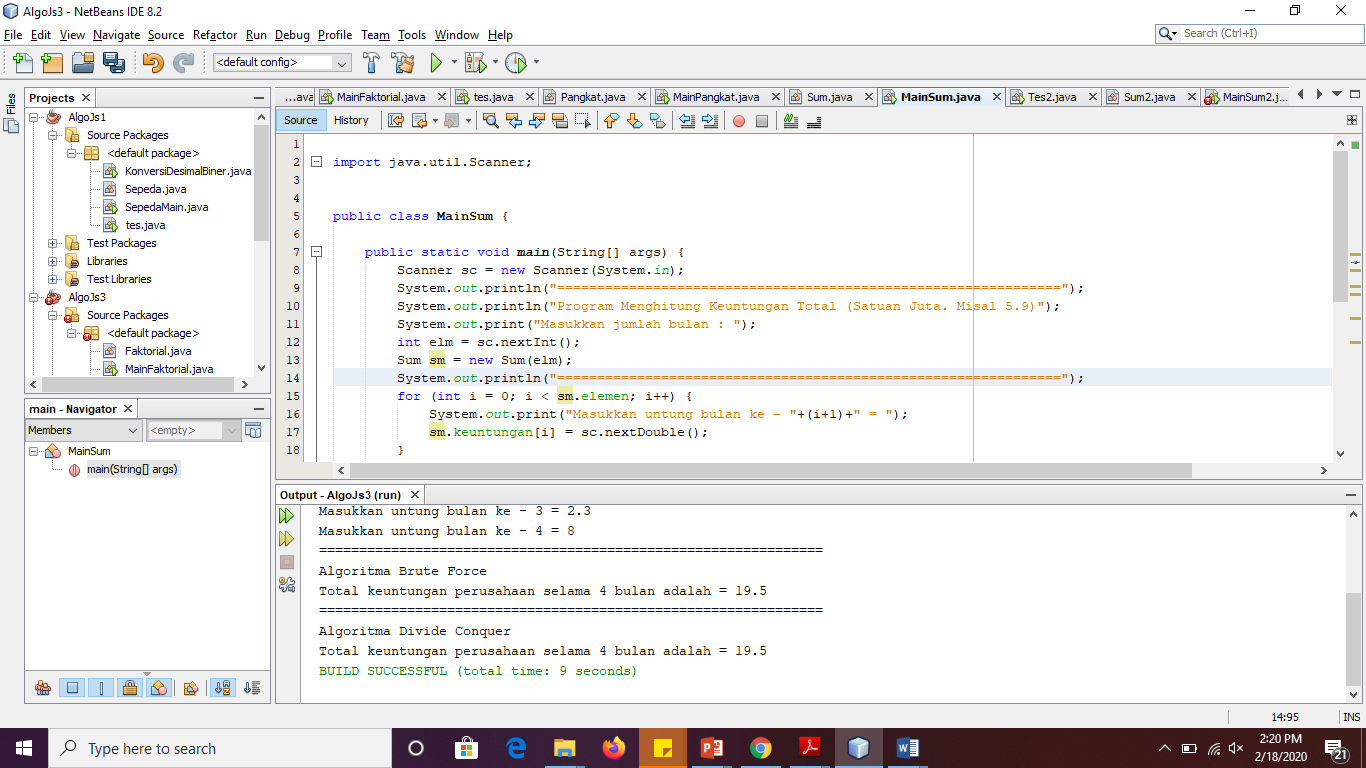
1. Tambahkan method TotalBF() yang akan menghitung total nilai array dengan cara *iterative*.



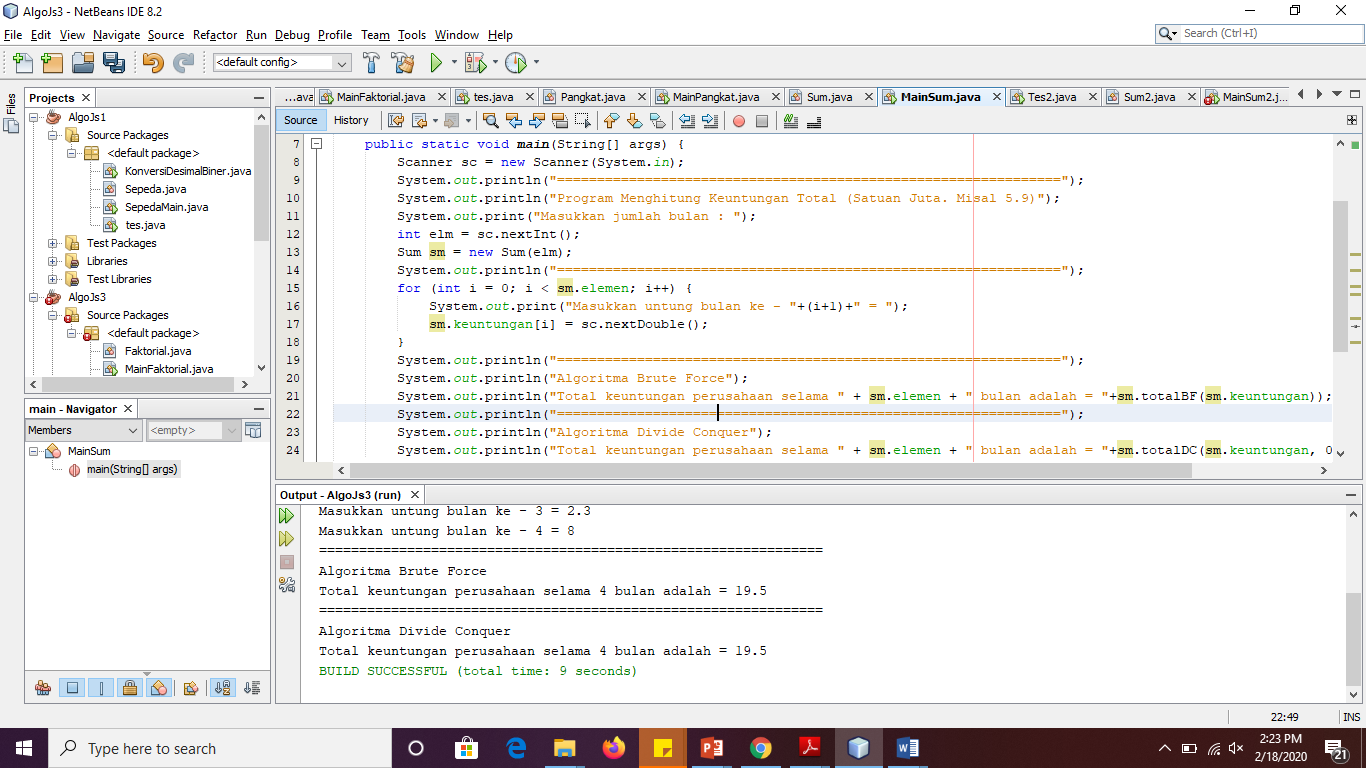
1. Tambahkan pula method TotalDC() untuk implementasi perhitungan nilai total array menggunakan algoritma Divide and Conquer



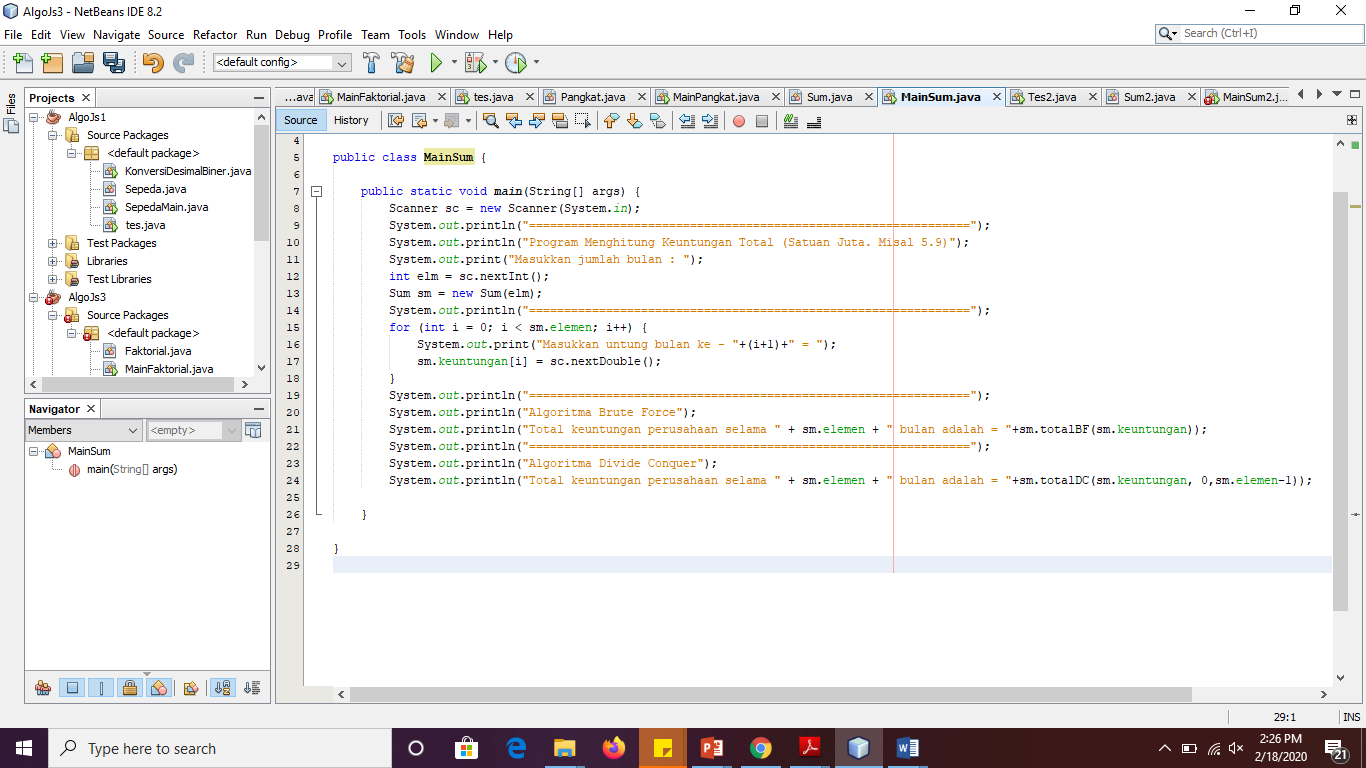
1. Buat class baru yaitu MainSum. Di dalam kelas ini terdapat method main. Pada method ini user dapat menuliskan berapa bulan keuntungan yang akan dihitung. Dalam kelas ini sekaligus dibuat instansiasi objek untuk memanggil atribut ataupun fungsi pada class Sum



1. 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.



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



* + 1. Verifikasi Hasil Percobaan

Cocokkan hasil compile kode program anda dengan gambar berikut ini.

import java.util.Scanner;

public class mainSumpitak {

    public static void main(String[] args) {

        Scanner pangkat19 = new Scanner(System.in);

        System.out.println("===========================================");

        System.out.println("masukkan jumlah elemen yang ingin di ijir SUMPITAK");

        int elm = pangkat19.nextInt();

        sumpitak sumpitak = new sumpitak(elm);

        System.out.println("===============================================================");

        for (int i = 0; i < sumpitak.elemen; i++) {

            System.out.println("masukkan untung bulan ke - "+(i+1)+"===");

            sumpitak.keuntungan[i] = pangkat19.nextDouble();

        }

        System.out.println("===============================================================");

        System.out.println("ALGORITMA BRUTE FORCE");

        System.out.println("");

        System.out.println("Total Keuntungan PERUSAHAAN selama "+ sumpitak.elemen + "bulan adalah" + sumpitak.totalBF(sumpitak.keuntungan));

        System.out.println("===============================================================");

        System.out.println("ALGORITMA Dividen conquer");

        System.out.println("");

        System.out.println("Total Keuntungan PERUSAHAAN selama "+ sumpitak.elemen + "bulan adalah" + sumpitak.totalDC(sumpitak.keuntungan, 0, sumpitak.elemen-1));

    }

    //     Scanner pangkat19 = new Scanner(System.in);

    //     System.out.println("===========================================");

    //     System.out.println("masukkan jumlah elemen yang ingin di ijir SUMPITAK");

    //     int elm = pangkat19.nextInt();

    //     sumpitak sumpitak = new sumpitak(elm);

    //     System.out.println("===============================================================");

    //     for (int i = 0; i < sumpitak.elemen; i++) {

    //         System.out.println("masukkan untung bulan ke - "+(i+1)+"===");

    //         sumpitak.keuntungan[i] = pangkat19.nextDouble();

    //     }

    //     System.out.println("===============================================================");

    //     System.out.println("ALGORITMA BRUTE FORCE");

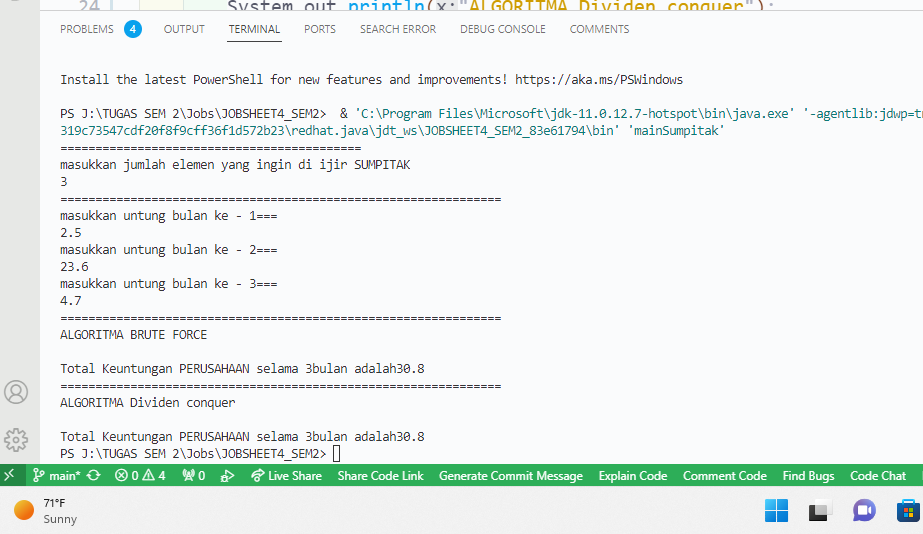
    //     System.out.println("Total Keuntungan PERUSAHAAN selama "+ sumpitak.elemen + "bulan adalah" + sumpitak.totalBF(sumpitak.keuntungan));

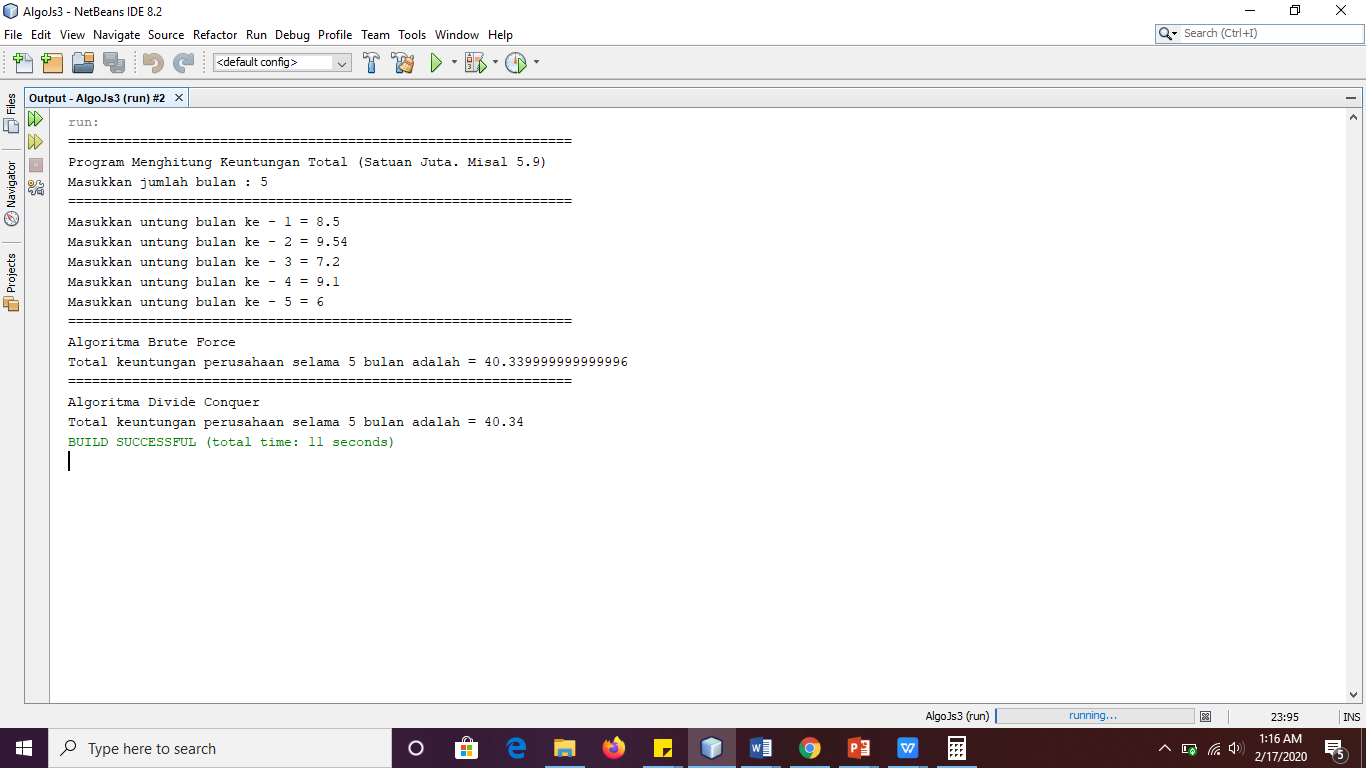
    //     System.out.println("ALGORITMA Dividen conquer");

    //     System.out.println("Total Keuntungan PERUSAHAAN selama "+ sumpitak.elemen + "bulan adalah" + sumpitak.totalDC(sumpitak.keuntungan, 0, sumpitak.elemen-1));

    // }

}





* + 1. Pertanyaan

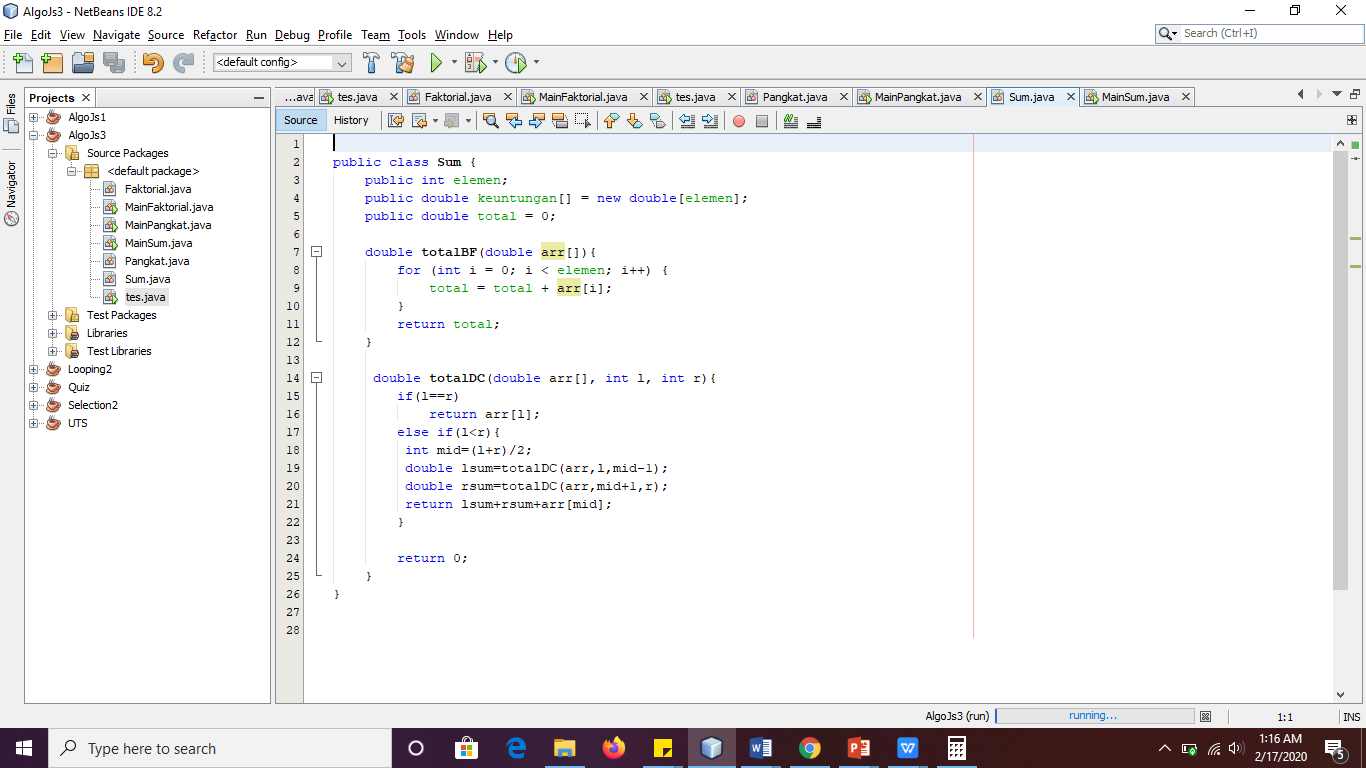
1. Berikan ilustrasi perbedaan perhitungan keuntungan dengan method TotalBF() ataupun TotalDC()

**Perbedaan TotalBF() dan TotalDC(): Metode `TotalBF()` menghitung total keuntungan dengan cara menjumlahkan semua elemen secara langsung, satu per satu. Ini seperti menambahkan setiap angka pada baris angka. Sebaliknya, metode `TotalDC()` menggunakan pendekatan Divide and Conquer. Ia membagi array keuntungan menjadi dua bagian yang sama, menghitung total masing-masing bagian secara rekursif, dan kemudian menambahkan dua total tersebut.**

1. 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.

**Untuk membatasi jumlah digit di belakang koma, Anda bisa menggunakan fungsi pembulatan atau format dalam bahasa pemrograman Anda. Misalnya, dalam Java, Anda bisa menggunakan `String.format("%.2f", yourValue)` untuk membatasi output hingga dua angka di belakang koma.**

1. Mengapa terdapat formulasi *return* *value* berikut?Jelaskan!



**Formulasi `return lsum+rsum+arr[mid];` digunakan dalam metode `TotalDC()`. `lsum` adalah total dari setengah bagian kiri array, `rsum` adalah total dari setengah bagian kanan array, dan `arr[mid]` adalah elemen tengah array. Formulasi ini mengembalikan total dari seluruh array.**

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

**Variabel `mid` digunakan untuk membagi array menjadi dua bagian yang hampir sama besar. Ini adalah bagian penting dari pendekatan Divide and Conquer.**

1. 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!

**Untuk menghitung keuntungan beberapa perusahaan sekaligus, Anda bisa membuat array dari objek `sumpitak`, di mana setiap objek mewakili satu perusahaan. Anda bisa mengulangi proses pengisian data dan perhitungan keuntungan untuk setiap perusahaan.**

* 1. Latihan Praktikum

Buatlah kode program untuk menghitung nilai akar dari suatu bilangan dengan algoritma Brute Force dan Divide Conquer! *Jika bilangan tersebut bukan merupakan kuadrat sempurna, bulatkan angka ke bawah.*

import java.util.Scanner;

public class mainsquar {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter a number: ");

        int x = scanner.nextInt();

        scanner.close();

        // int x = 10;

        System.out.println("Square root of " + x + " using Brute Force: " + SquareRoot.sqrtBF(x));

        System.out.println("Square root of " + x + " using Divide and Conquer: " + SquareRoot.sqrtDC(x));

    }

}

public class SquareRoot {

    public static int sqrtBF(int x) {

        if (x < 0) return -1; // Invalid input

        int i = 0;

        while (i \* i <= x) {

            i++;

        }

        return i - 1; // Return the floor value of the square root

    }

    // Divide and Conquer method

    public static int sqrtDC(int x) {

        if (x < 0) return -1; // Invalid input

        int start = 0, end = x;

        while (start <= end) {

            int mid = start + (end - start) / 2;

            if (mid \* mid == x) {

                return mid;

            } else if (mid \* mid < x) {

                start = mid + 1;

            } else {

                end = mid - 1;

            }

        }

        return end; // Return the floor value of the square root

    }

}

