JOBSHEET IV  
BRUTE FORCE DAN DIVIDE CONQUER

Name : Rizqi Dewanto

Number : 17

Class : SIB - 1G

Github : https://github.com/Rizxqi

1. Tujuan Praktikum

Setelah melakukan materi praktikum ini, mahasiswa mampu:

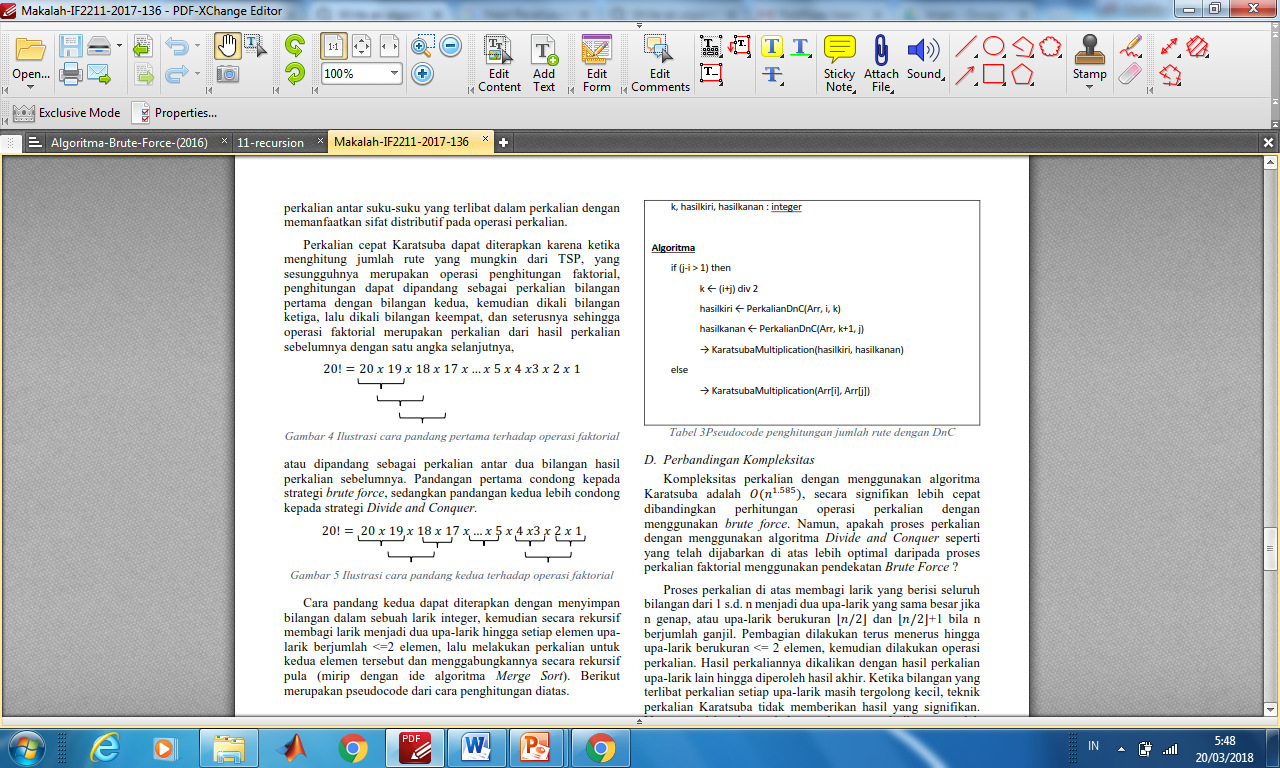
1. Mahasiswa mampu membuat algoritma bruteforce dan divide-conquer
2. Mahasiswa mampu menerapkan penggunaan algorima bruteforce dan divide-conquer
3. Menghitung Nilai Faktorial dengan Algoritma Brute Force dan Divide and Conquer

Perhatikan Diagram Class berikut ini :

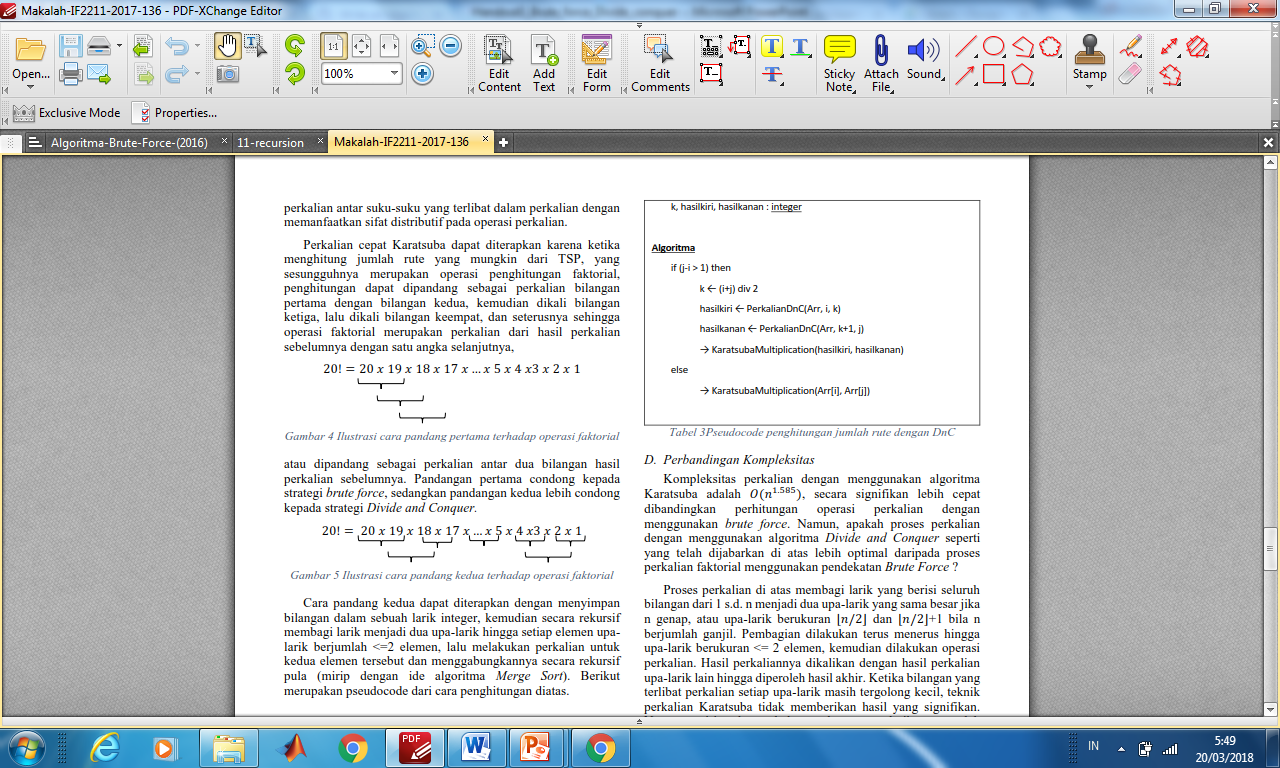
|  |
| --- |
| Faktorial |
| nilai: int |
| faktorialBF(): int  faktorialDC(): int |

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 :

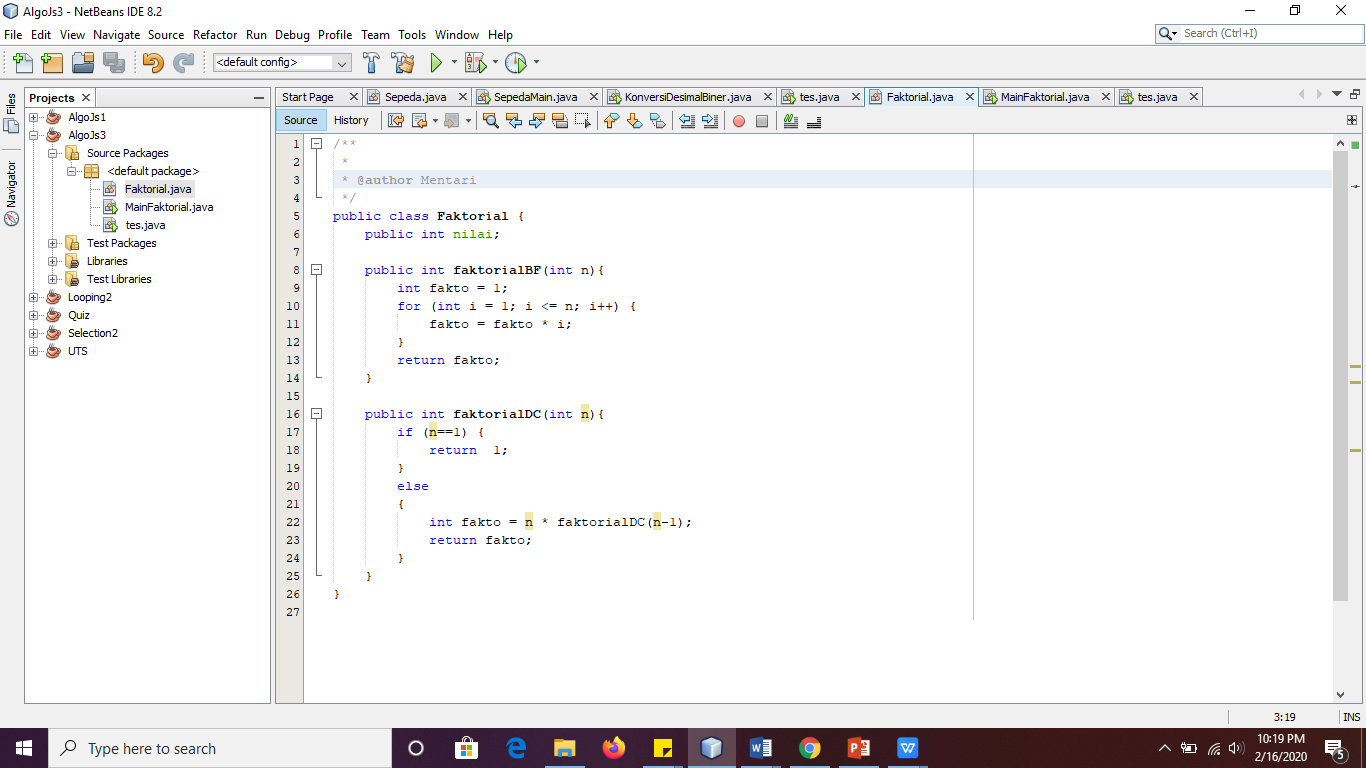
Tahapan pencarian nilai faktorial dengan algoritma Brute Force :



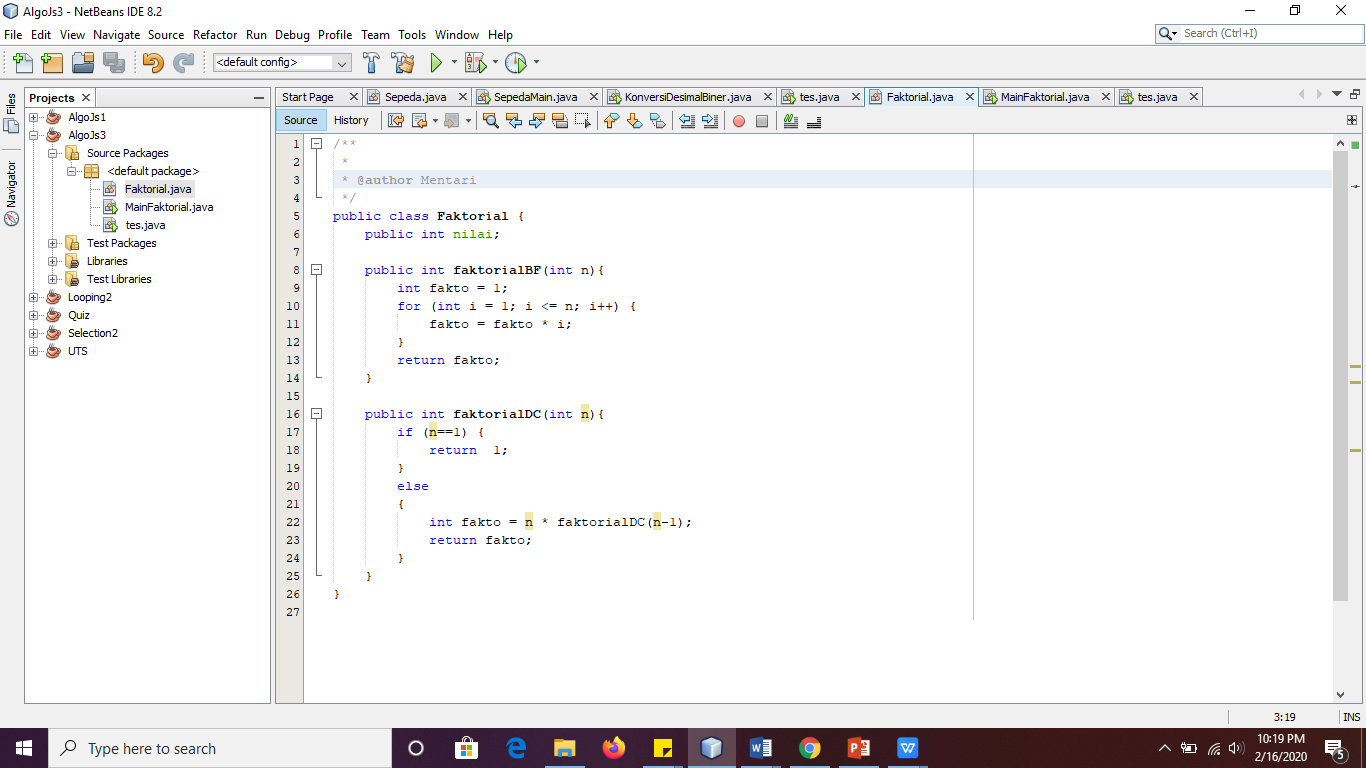
Tahapan pencarian nilai faktorial dengan algoritma Divide and Conquer :



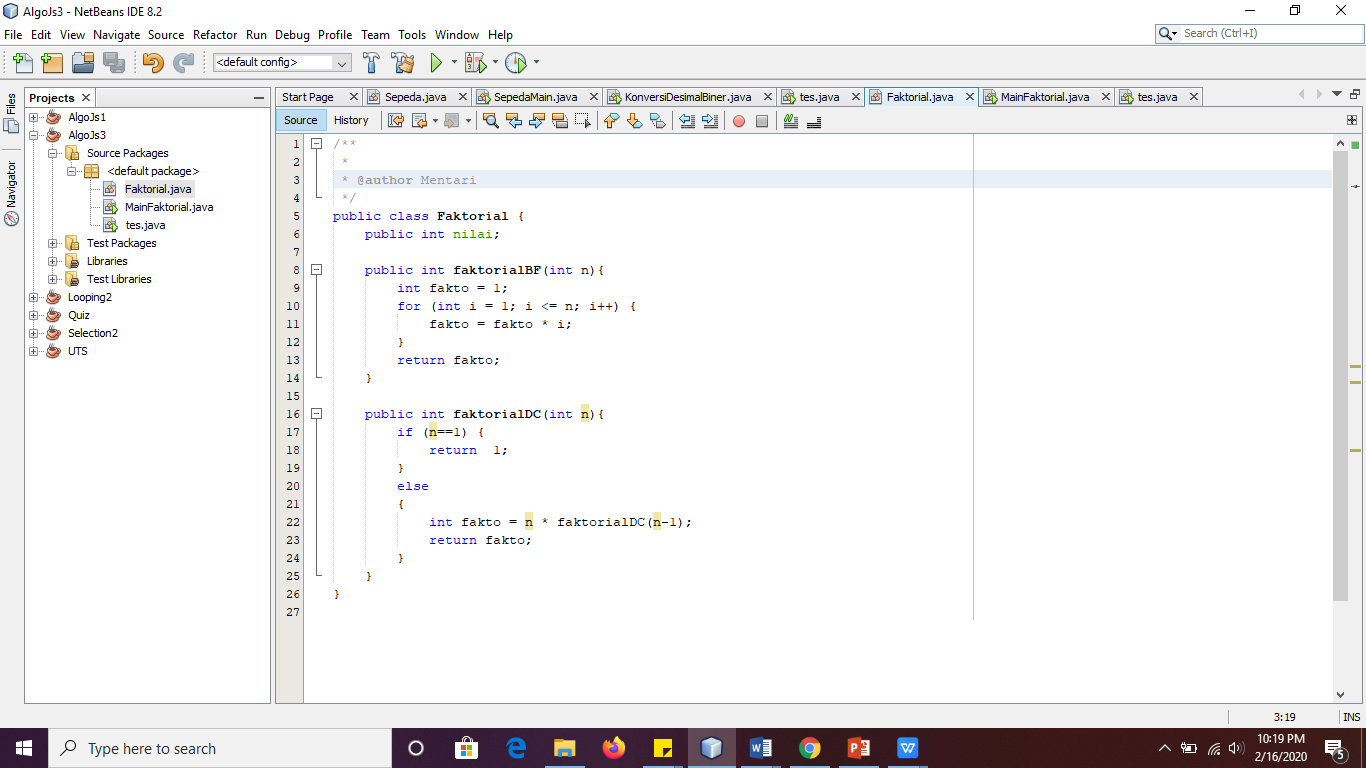
1. Langkah-langkah Percobaan
2. Buat Project baru, dengannama “**BruteForceDivideConquer**”. Buat package dengan nama minggu5.
3. Buatlah class baru dengan nama **Faktorial**
4. Lengkapi class **Faktorial** dengan atribut dan method yang telah digambarkan di dalam diagram class di atas, sebagai berikut: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



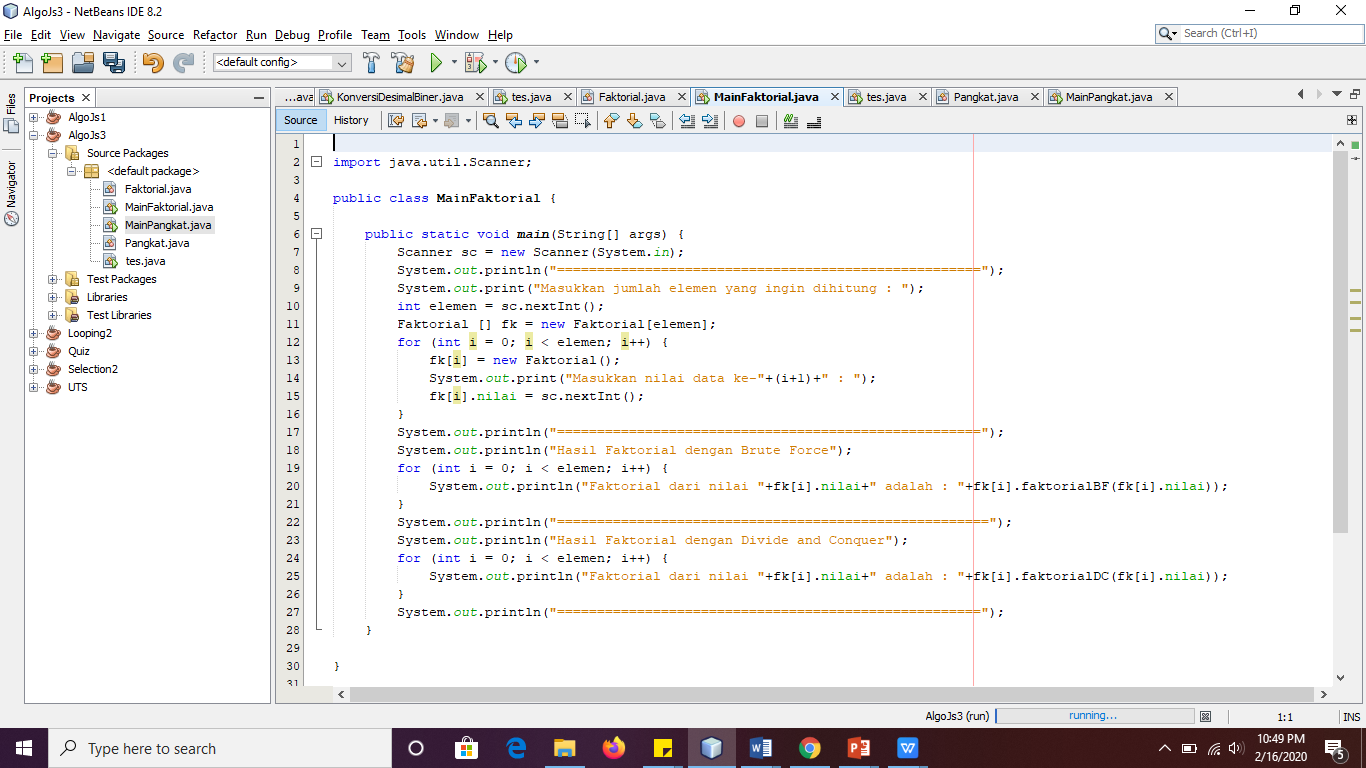
1. Tambahkan method faktorialBF() nilai



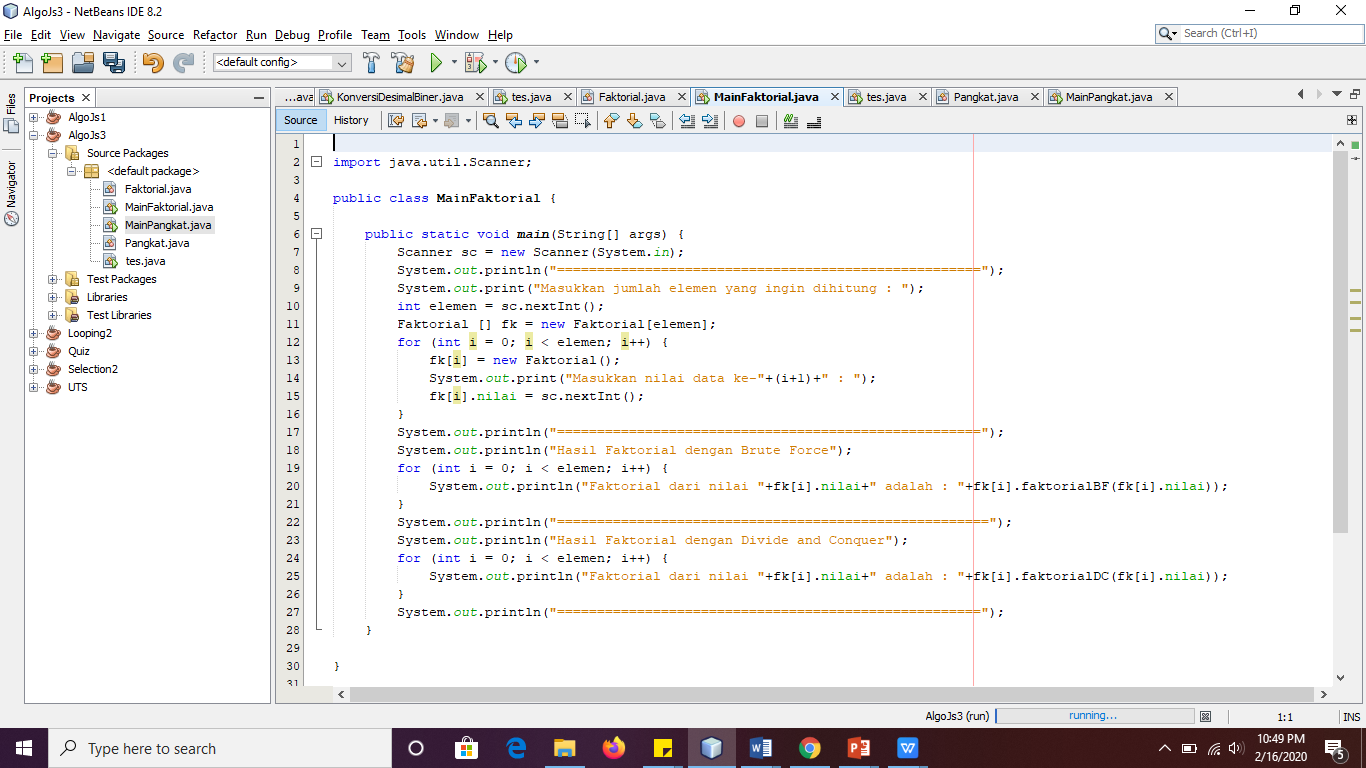
1. Tambahkan method faktorialDC() nilai



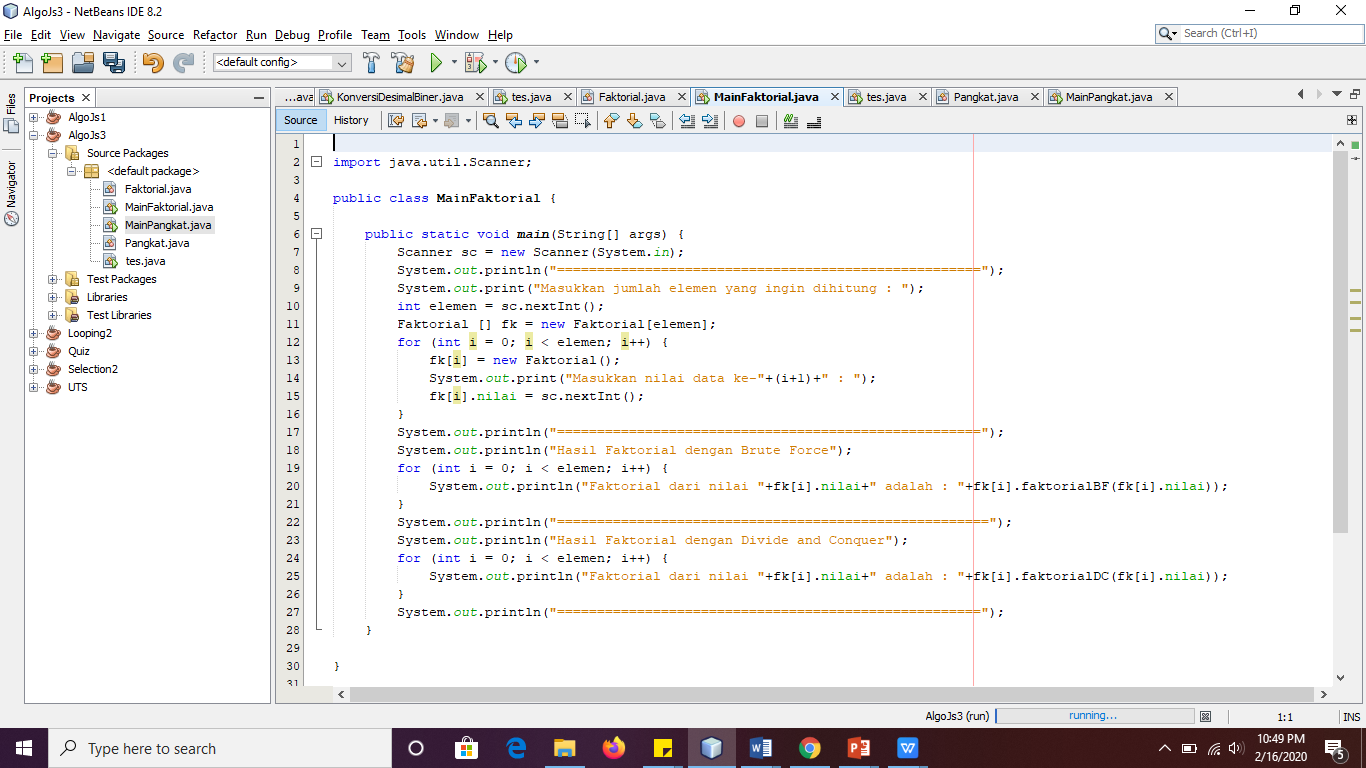
1. Coba jalankan (Run) class Faktorial dengan membuat class baru MainFaktorial.
2. Di dalam fungsi main sediakan komunikasi dengan user untuk menginputkan jumlah angka yang akan dicari nilai faktorialnya



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

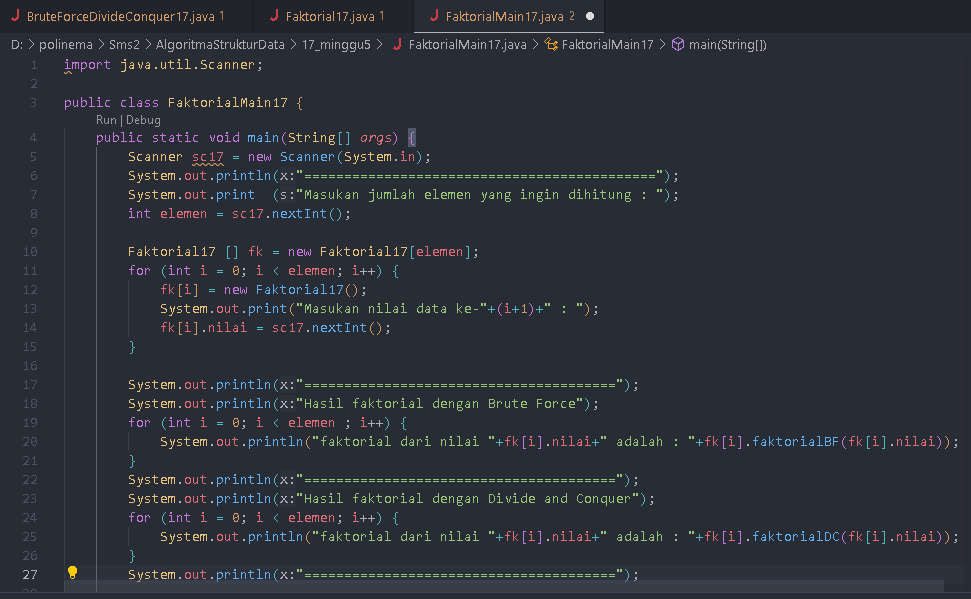


1. Tampilkan hasil pemanggilan method faktorialDC() dan faktorialBF()



* 1. Pastikan program sudah berjalan dengan baik!

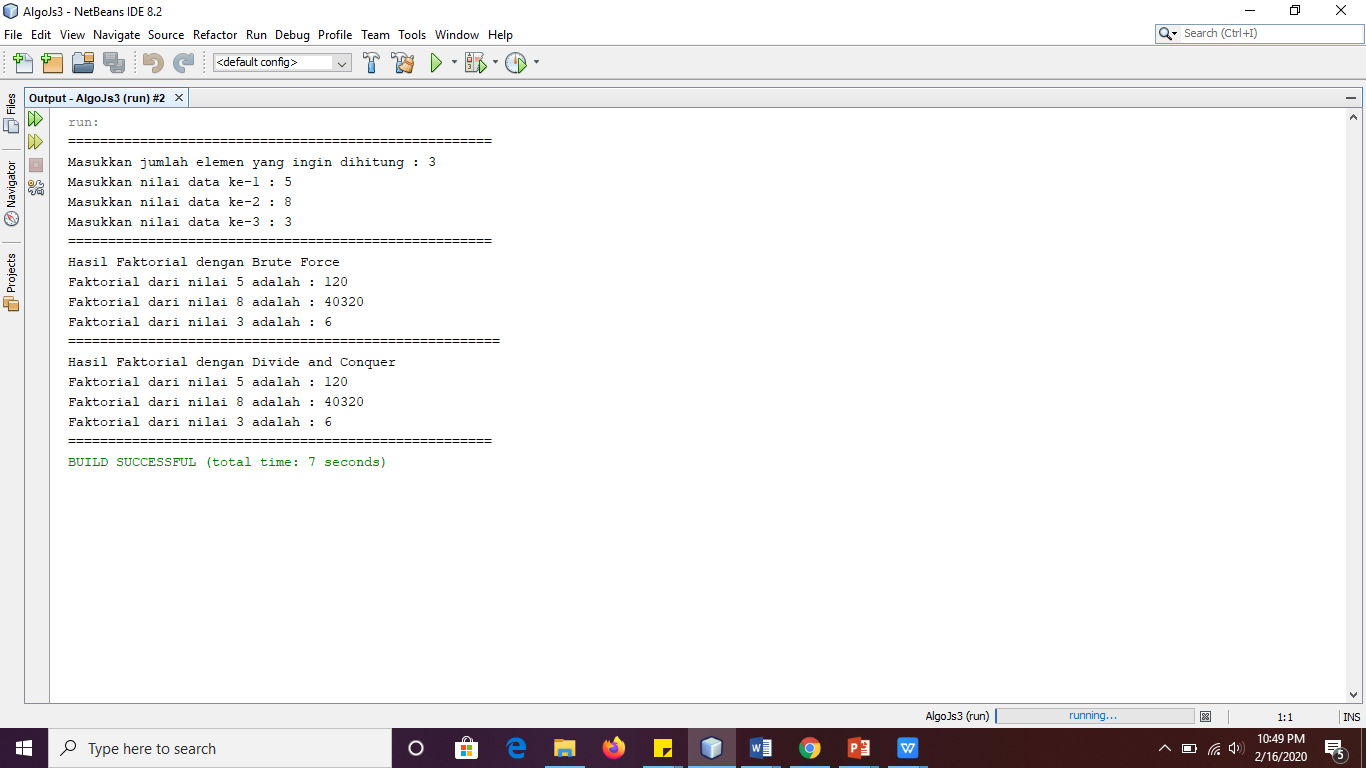


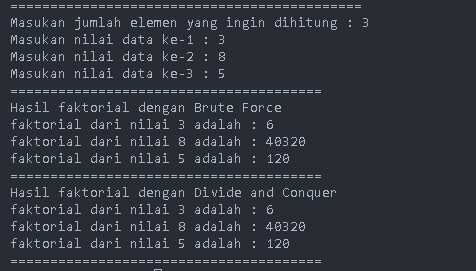


I

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!

Untuk pencarian nilai faktorial, baseline terjadi ketika bilangan yang akan dicari faktorialnya adalah 1. Ini karena definisi faktorial menyatakan bahwa 1!=1. Dengan demikian, pada titik ini, tidak perlu lagi melakukan rekursi untuk mencari nilai faktorial, karena sudah mengetahui hasilnya secara langsung.

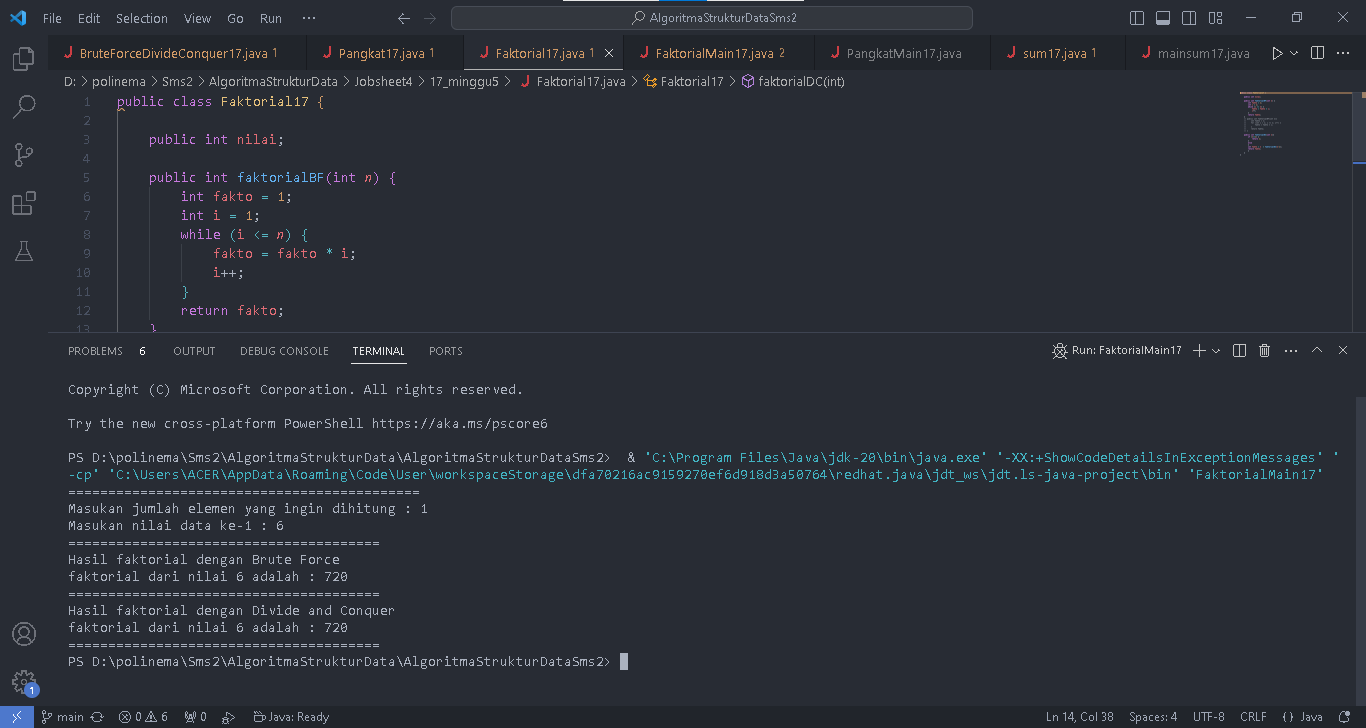
1. Pada implementasi Algoritma Divide and Conquer Faktorial apakah lengkap terdiri dari 3 tahapan divide, conquer, combine? Jelaskan masing-masing bagiannya pada kode program!

* Divide:Pada tahap ini, masalah dibagi menjadi submasalah yang lebih kecil. Dalam kasus faktorial, kita membagi perhitungan faktorial dari n menjadi perhitungan faktorial dari n-1.
* Conquer : , tahap conquer terjadi ketika kita mengalikan n dengan hasil dari faktorial

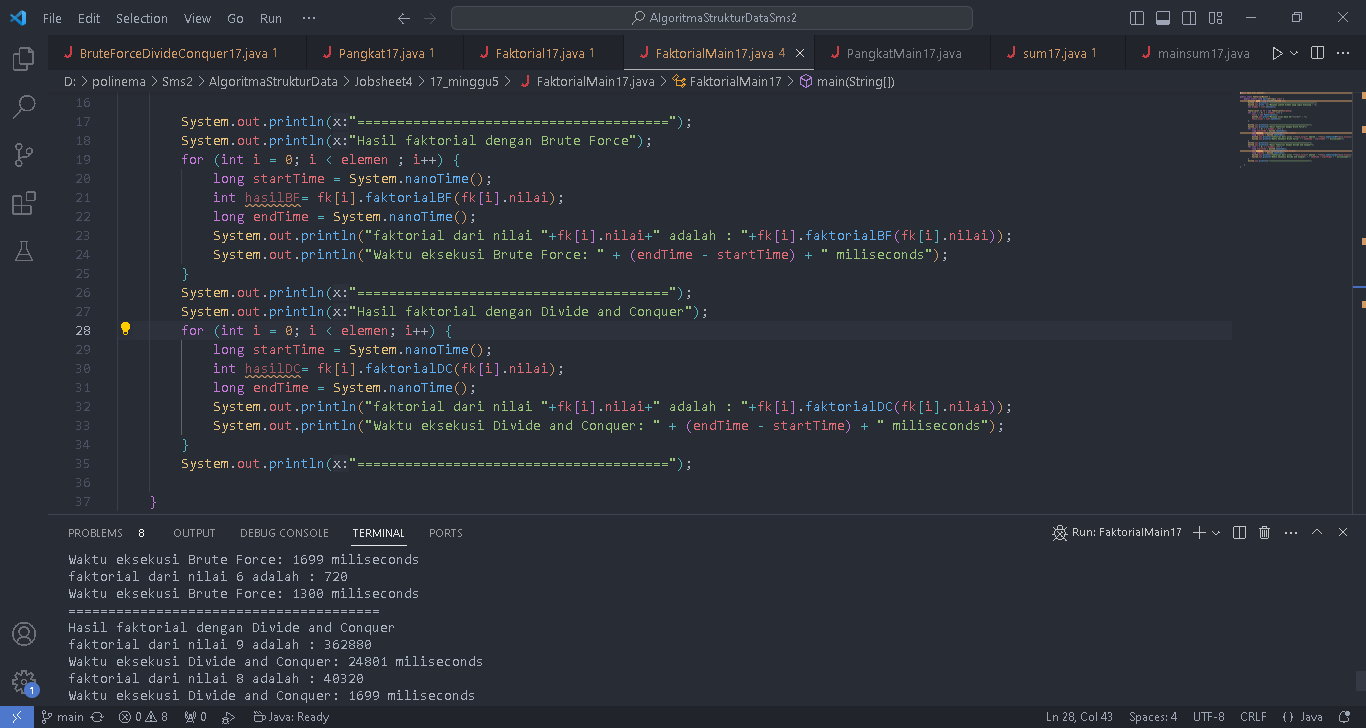
DC(n-1).

* Combine : Pada tahap ini, kita menggabungkan solusi dari submasalah menjadi solusi utama.

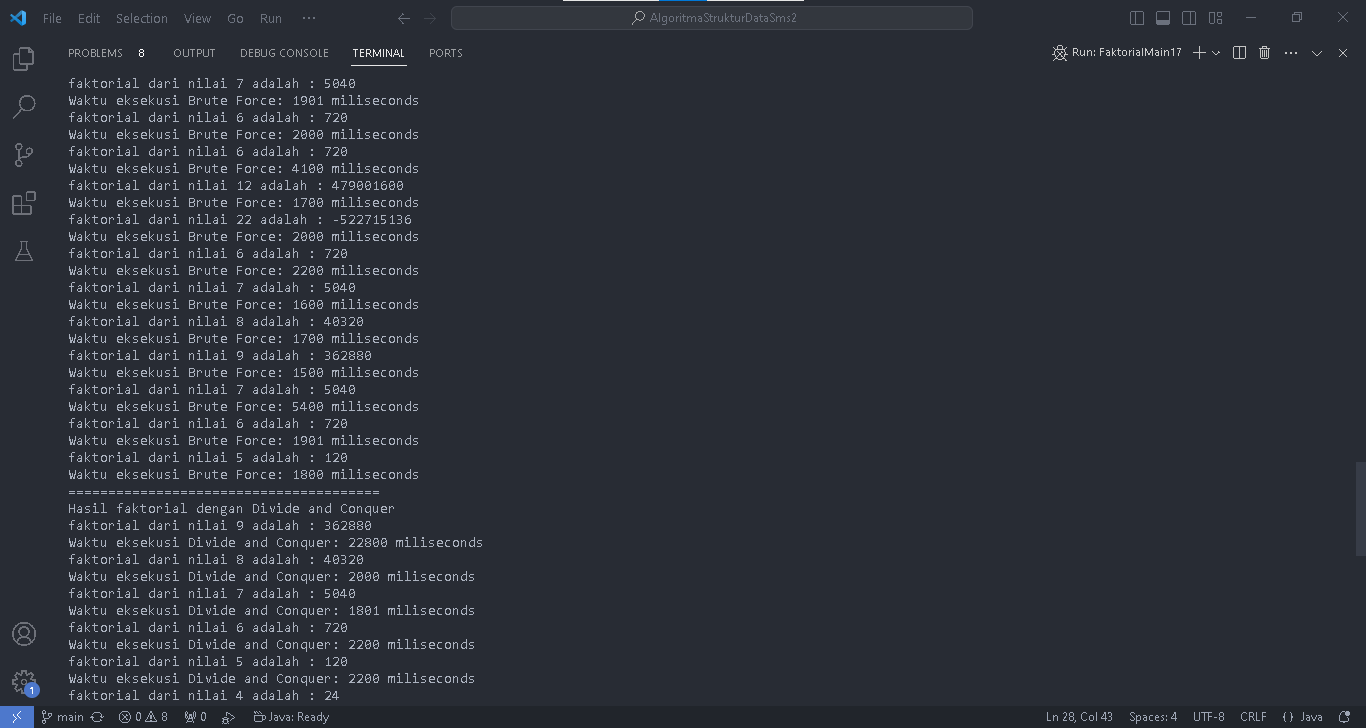
1. Apakah memungkinkan perulangan pada method faktorialBF() dirubah selain menggunakan for?Buktikan!



1. Tambahkan pegecekan waktu eksekusi kedua jenis method tersebut!



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



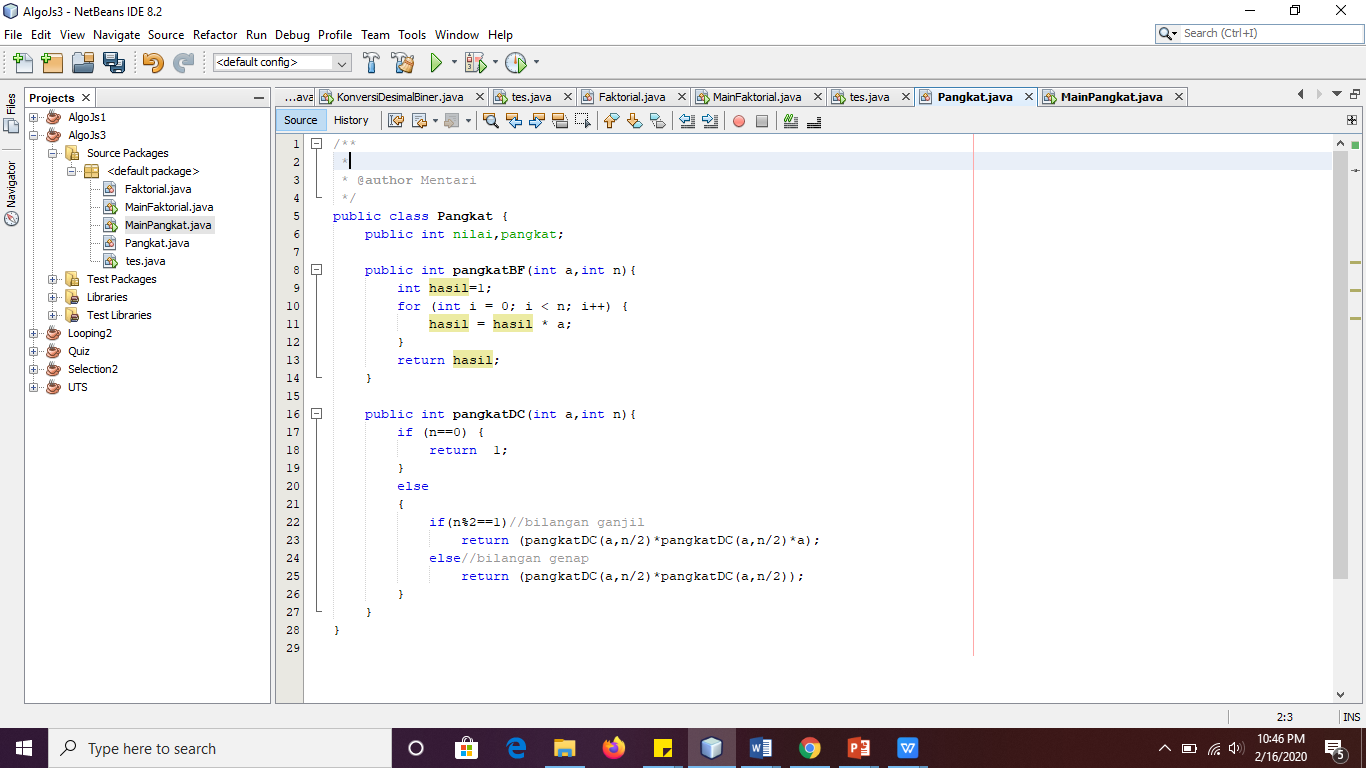
Ada, saat jumlah elemen semakin besar, metode Divide and Conquer cenderung lebih efisien daripada metode Brute Force. Namun, ini bisa bervariasi tergantung pada implementasi dan lingkungan eksekusi.

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

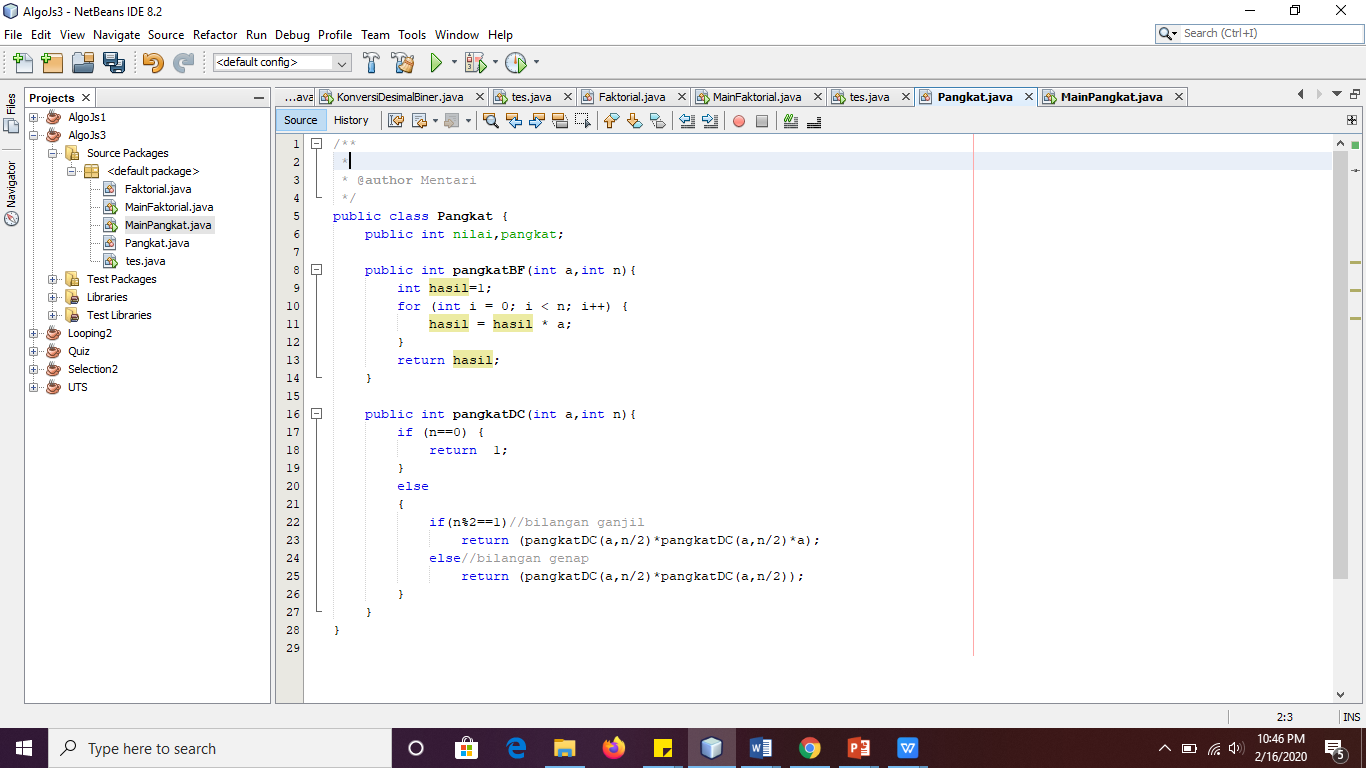
Pada praktikum ini kita akan membuat program class dalam Java. Untuk menghitung nilai pangkat suatu angka menggunakan 2 jenis algoritma, Brute Force dan Divide and Conquer.

* + 1. Langkah-langkah Percobaan

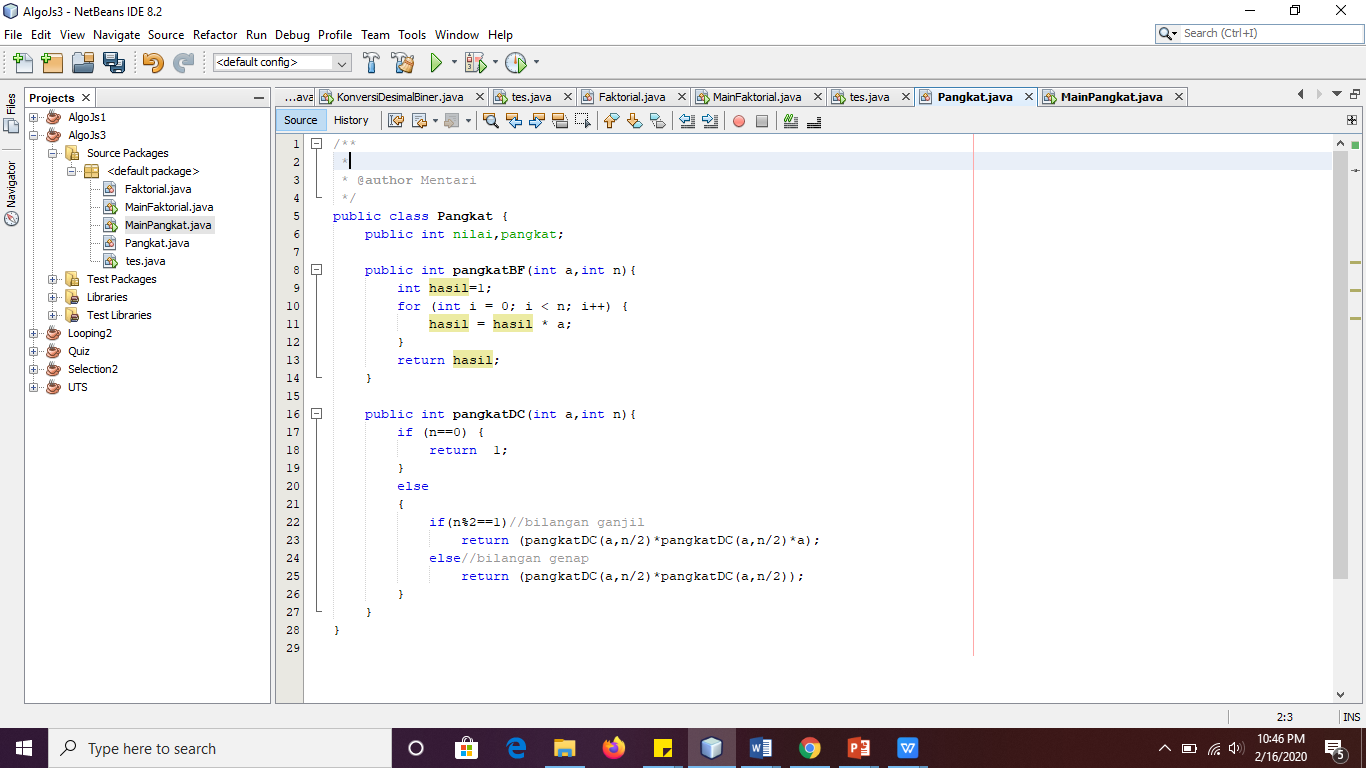
1. Di dalam paket **minggu5**, buatlah class baru dengan nama **Pangkat**. Dan di dalam class **Pangkat** tersebut, buat atribut angka yang akan dipangkatkan sekaligus dengan angka pemangkatnya



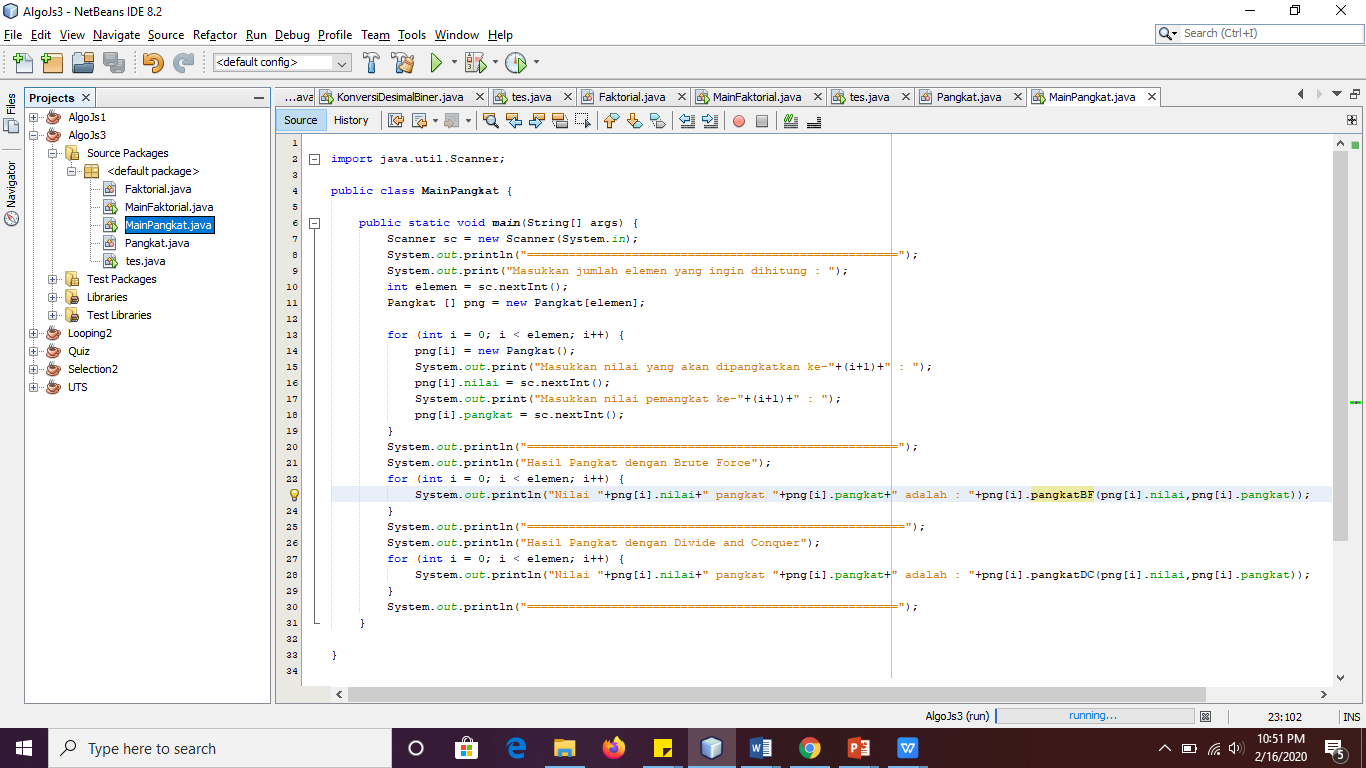
1. Pada class Pangkat tersebut, tambahkan method PangkatBF()



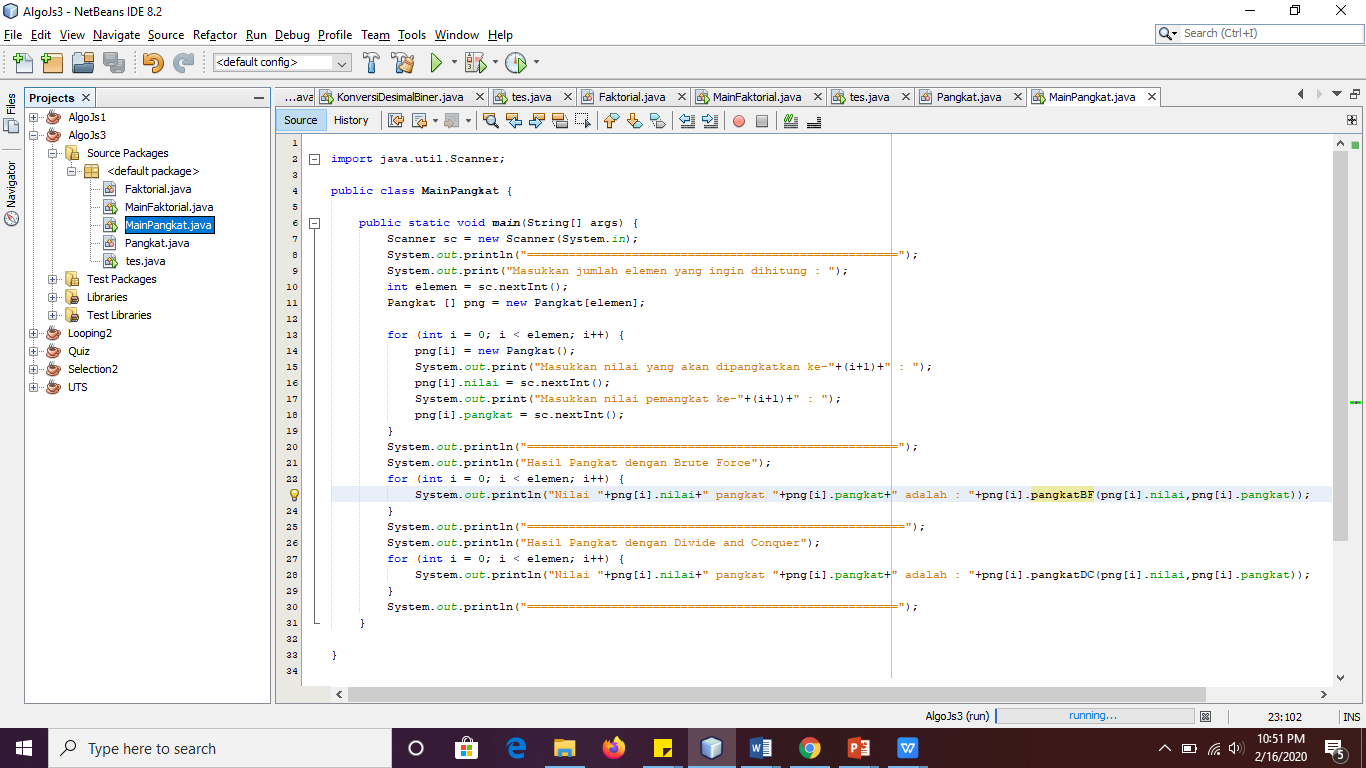
1. Pada class Pangkat juga tambahkan method PangkatDC()



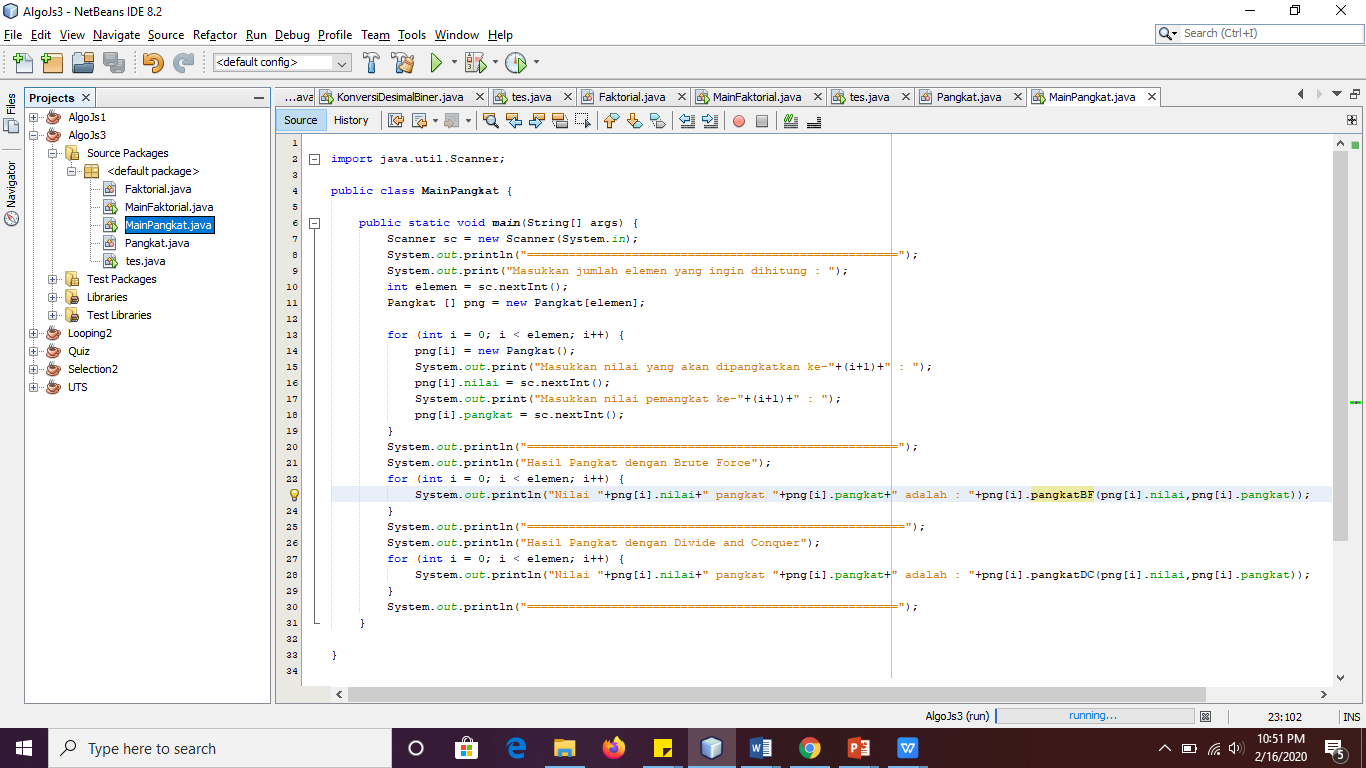
1. Perhatikan apakah sudah tidak ada kesalahan yang muncul dalam pembuatan class Pangkat
2. Selanjutnya buat class baru yang di dalamnya terdapat method main. Class tersebut dapat dinamakan MainPangkat. Tambahkan kode pada class main untuk menginputkan jumlah nilai yang akan dihitung pangkatnya.



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.

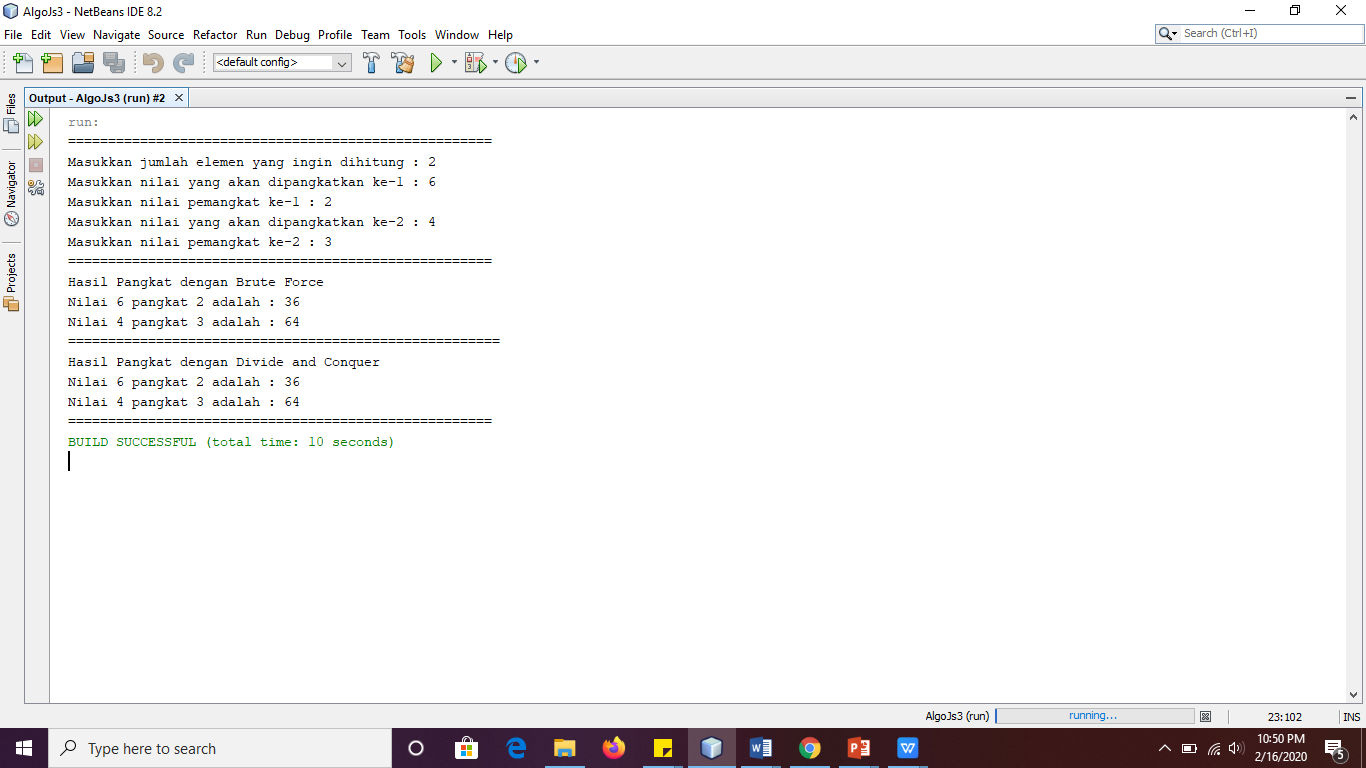


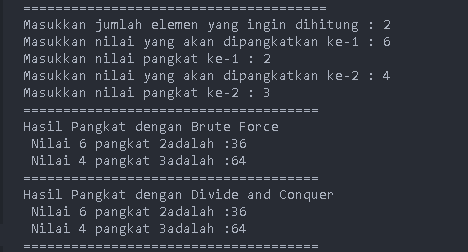
1. Kemudian, panggil hasil nya dengan mengeluarkan return value dari method PangkatBF() dan PangkatDC().



* + 1. Verifikasi Hasil Percobaan

Pastikan output yang ditampilkan sudah benar seperti di bawah ini.



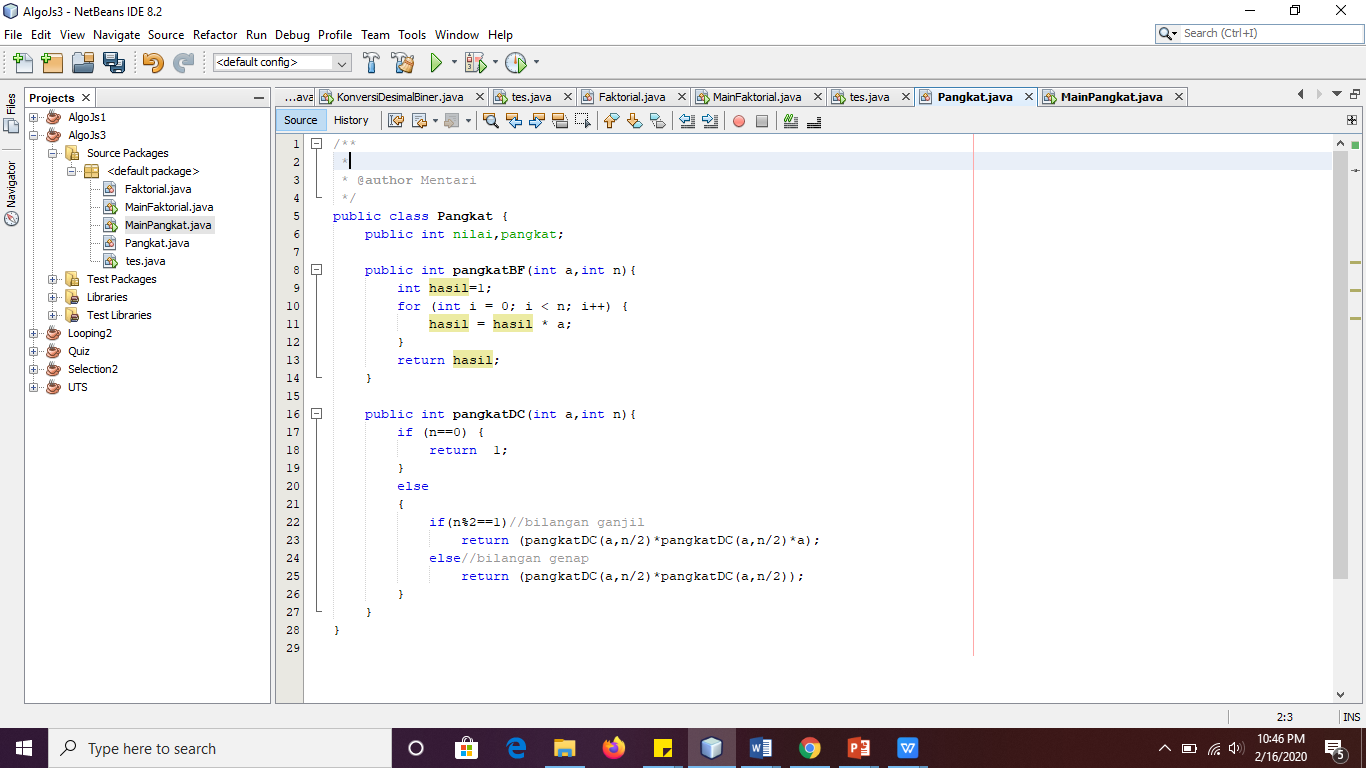


* + 1. Pertanyaan

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

Perbedaan antarra PangkatBf() dan PangkatDC() terletak pada if else atau divide memecah masalah menjadi submasalah sedangkan bruteforce lebih ke trial error

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



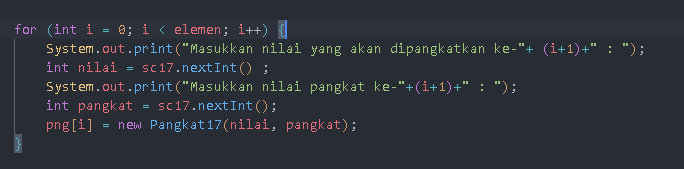
Jelaskan arti potongan kode tersebut

Jadi apabila nilai n sisa baginya 1 termasuk bilangan ganjil semisal 5/2 sisa satu jadi 5 termasuk bilangan ganjil apabila habis dibagi 2 berarti genap dan potonga kode diatas termasuk bagian combine

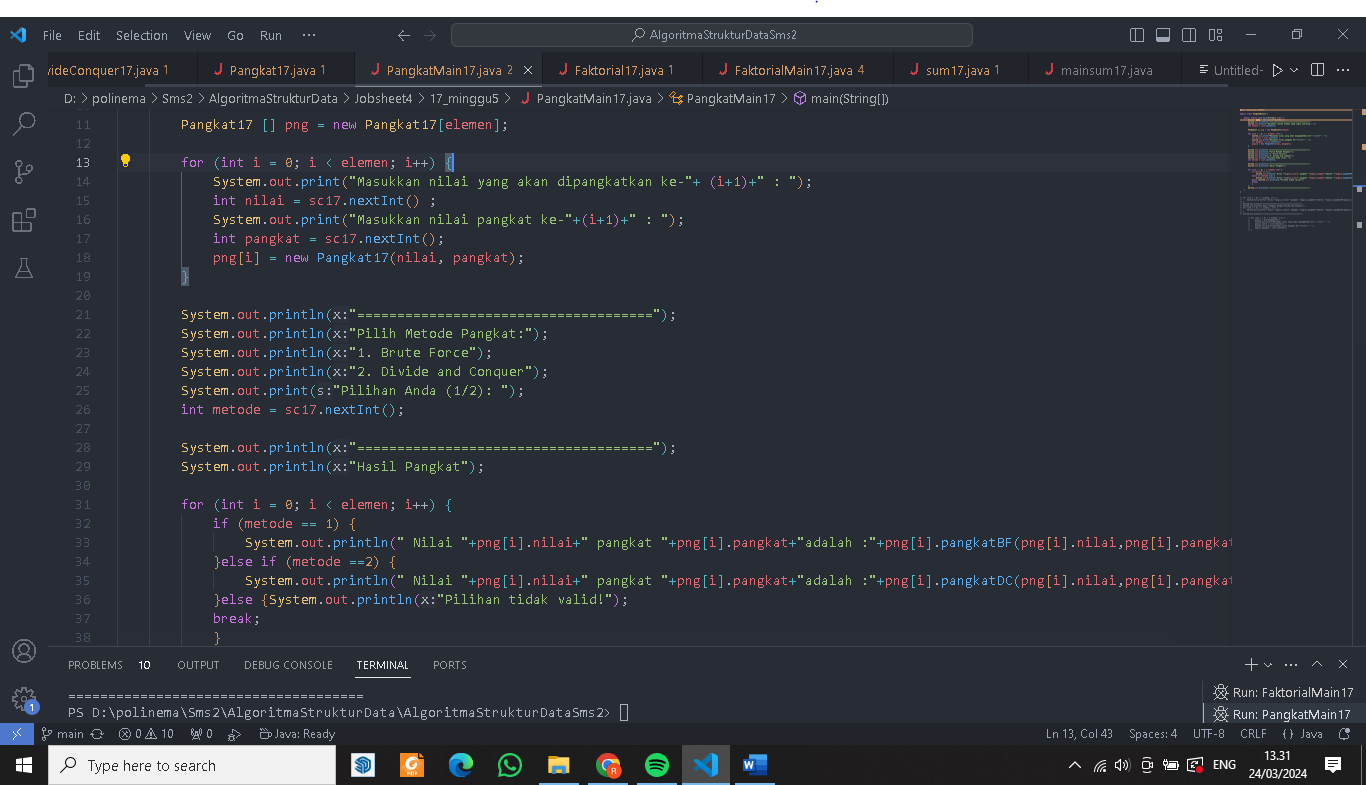
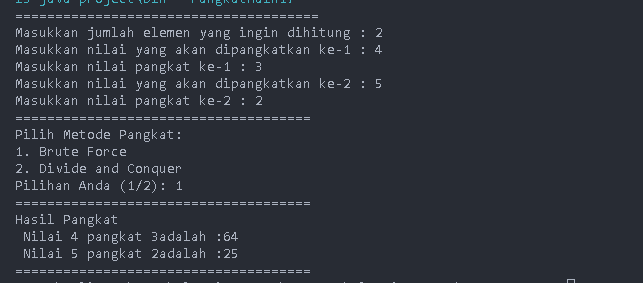
1. Apakah tahap *combine* sudah termasuk dalam kode tersebut?Tunjukkan!

Ketika n ganjil, baris return (pangkatDC(a, n / 2) \* pangkatDC(a, n / 2) \* a); adalah tahap combine. Di sini, kita memanggil pangkatDC(a, n/2) dua kali untuk menghitung nilai pangkat dari a dengan pangkat yang lebih kecil (n/2), dan hasilnya dikalikan dengan a.

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



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

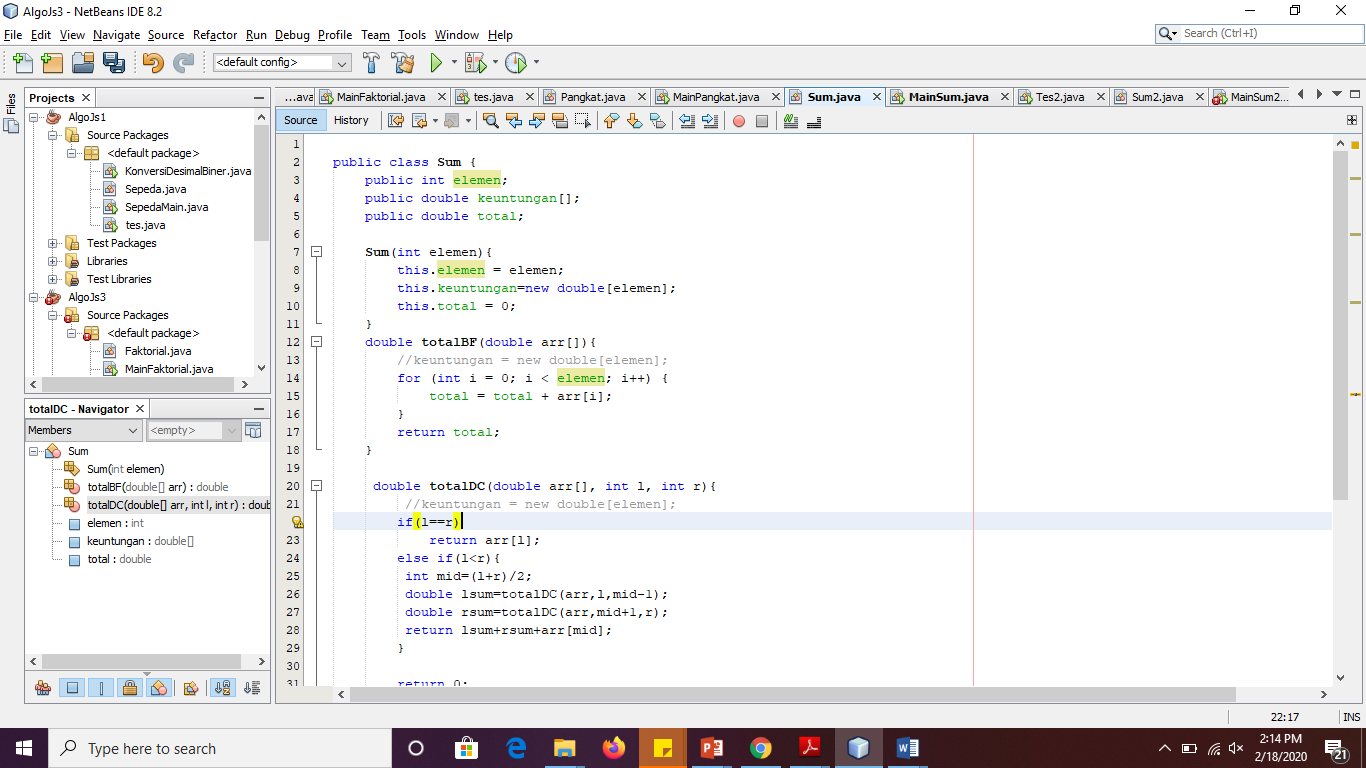


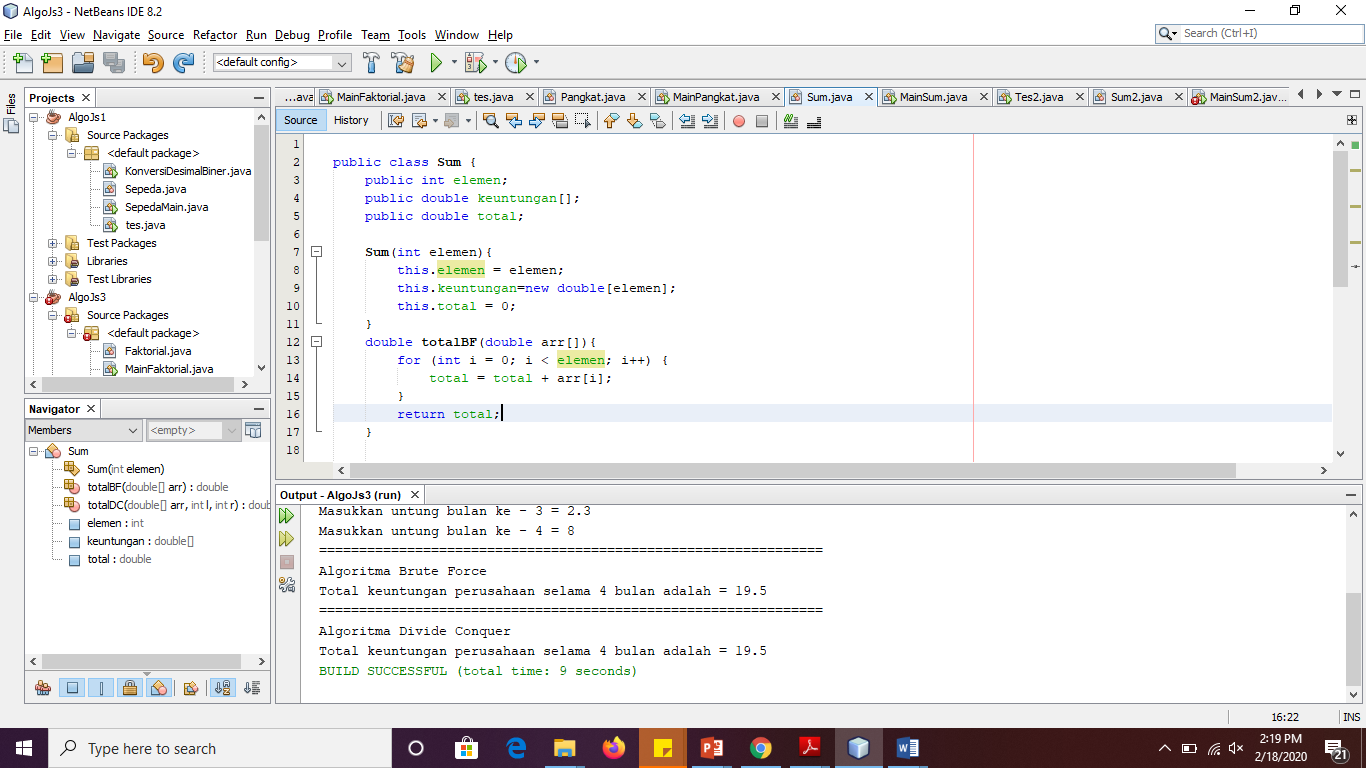
1. Menghitung Sum Array dengan Algoritma Brute Force dan Divide and Conquer

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

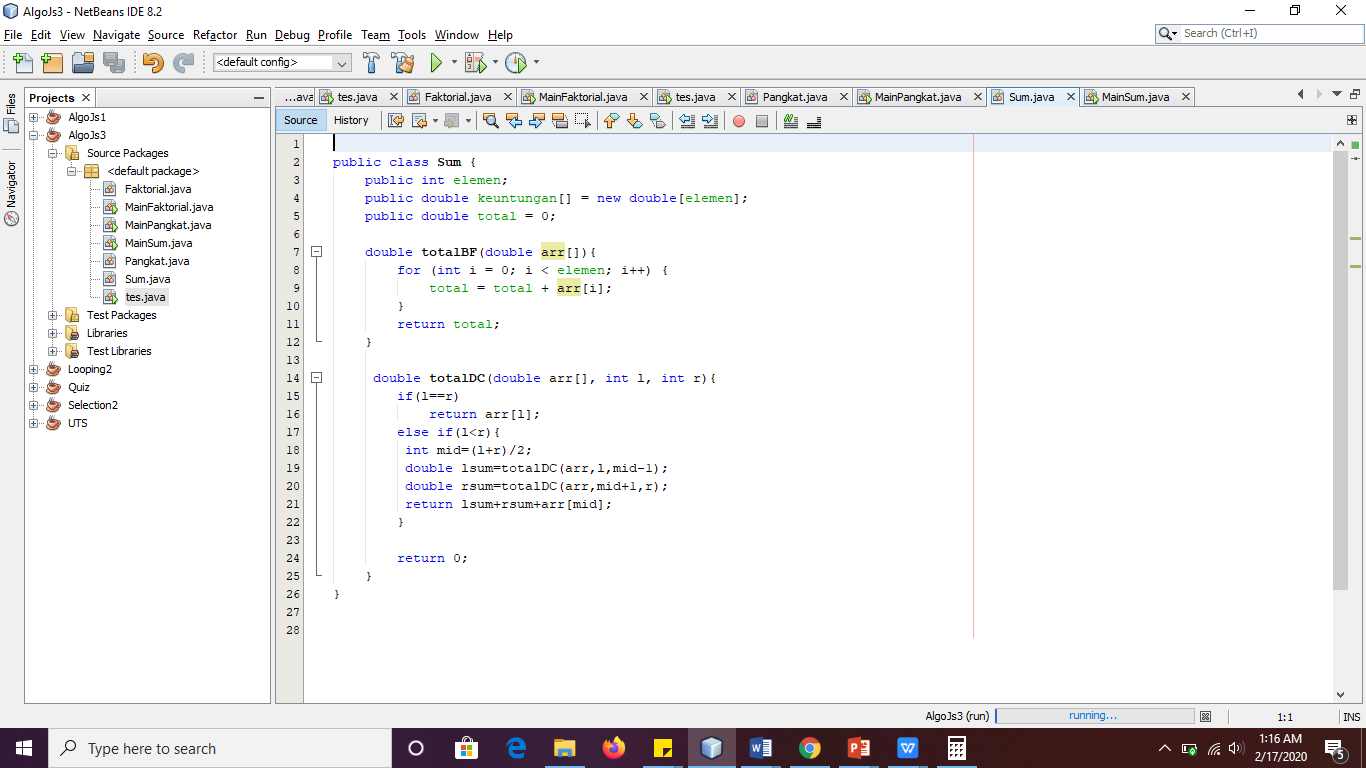
* + 1. Langkah-langkah Percobaan

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.

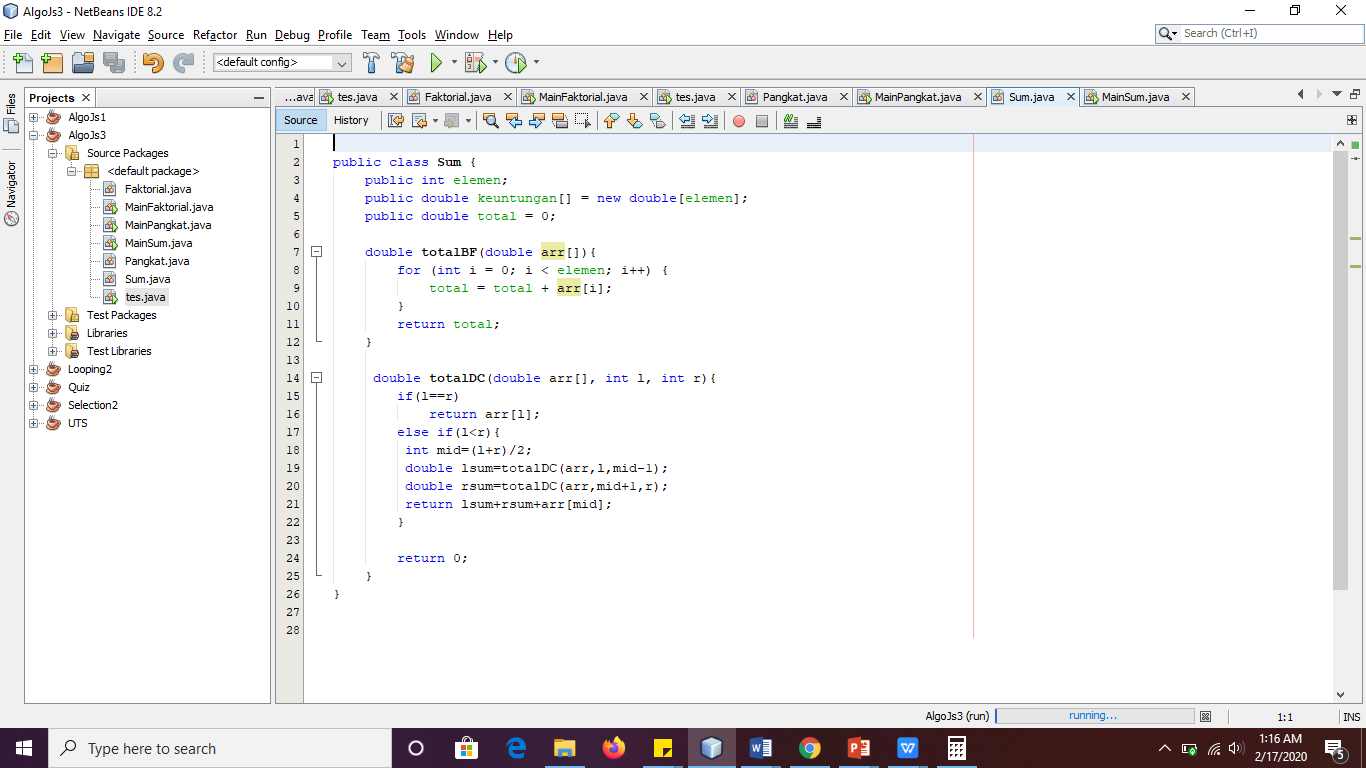




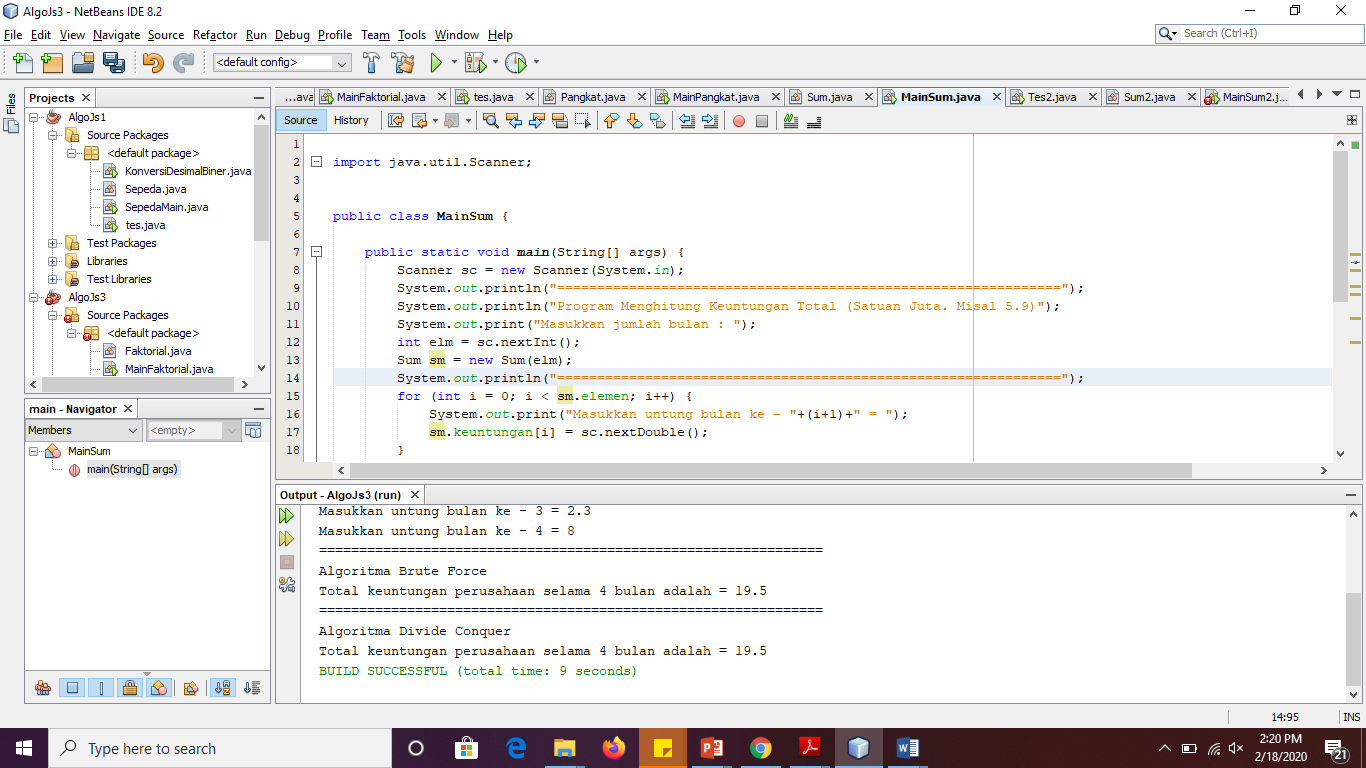
1. Tambahkan method TotalBF() yang akan menghitung total nilai array dengan cara *iterative*.



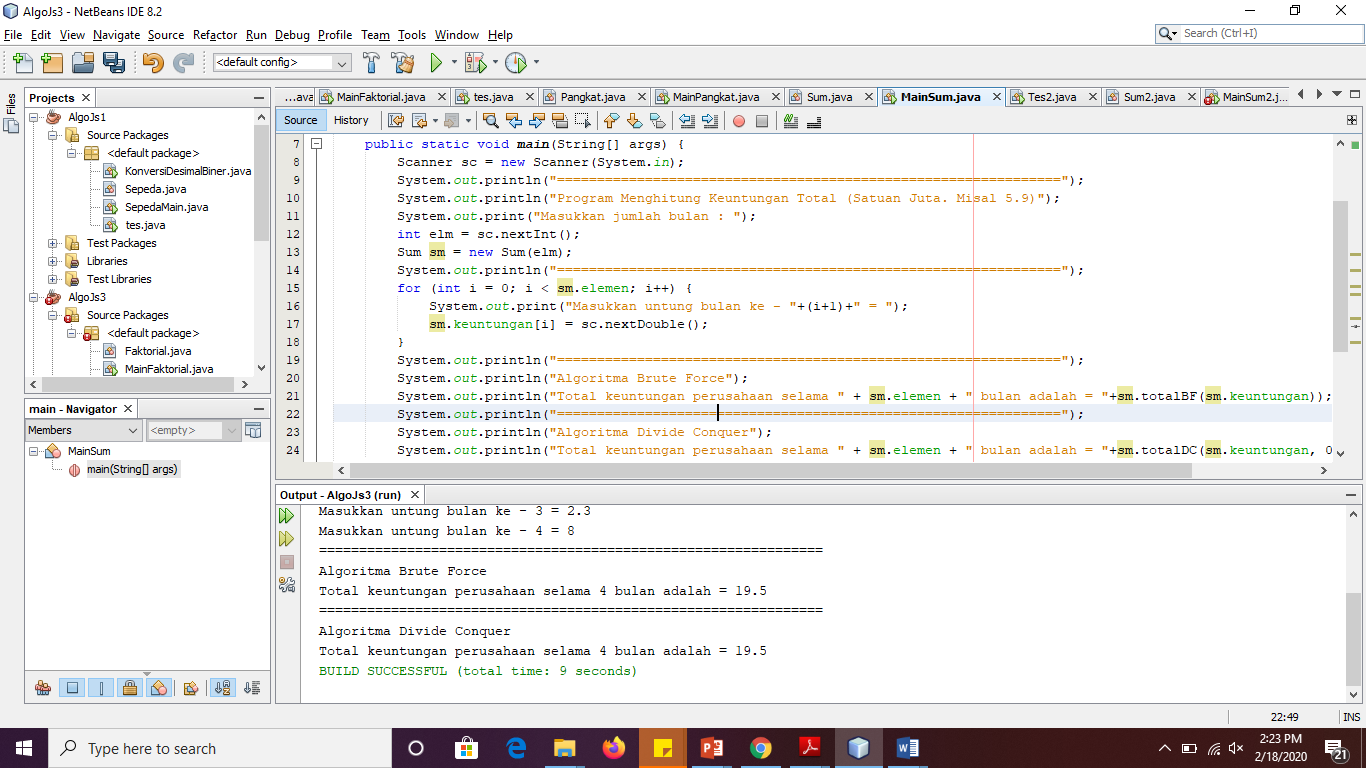
1. Tambahkan pula method TotalDC() untuk implementasi perhitungan nilai total array menggunakan algoritma Divide and Conquer



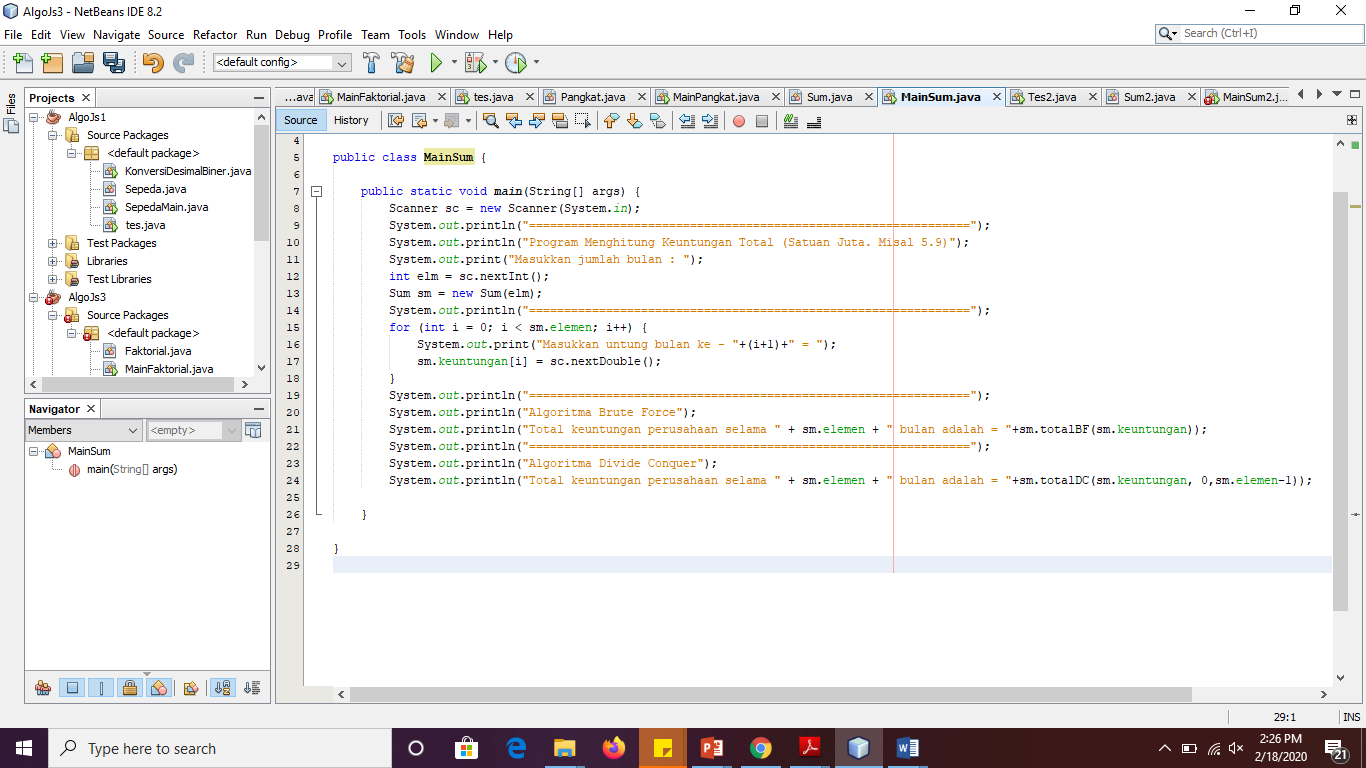
1. Buat class baru yaitu MainSum. Di dalam kelas ini terdapat method main. Pada method ini user dapat menuliskan berapa bulan keuntungan yang akan dihitung. Dalam kelas ini sekaligus dibuat instansiasi objek untuk memanggil atribut ataupun fungsi pada class Sum



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.

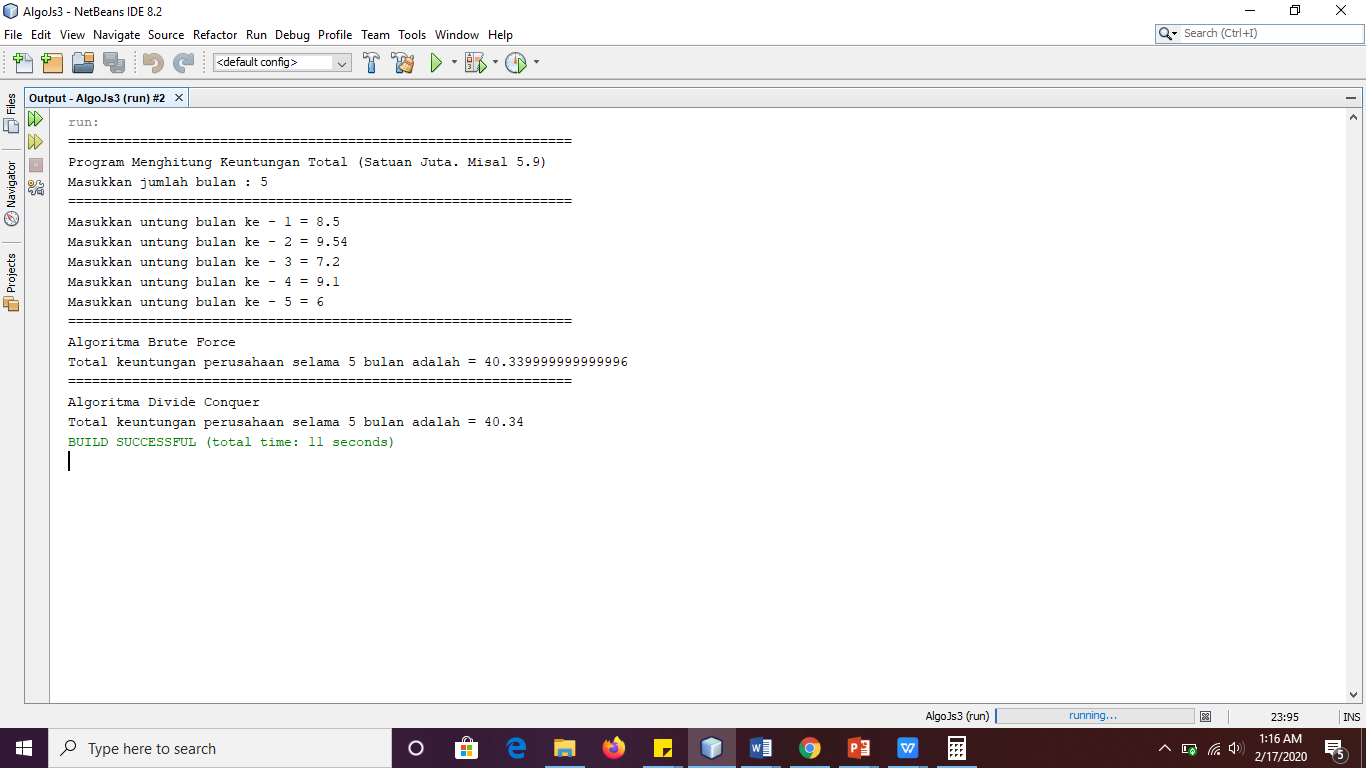


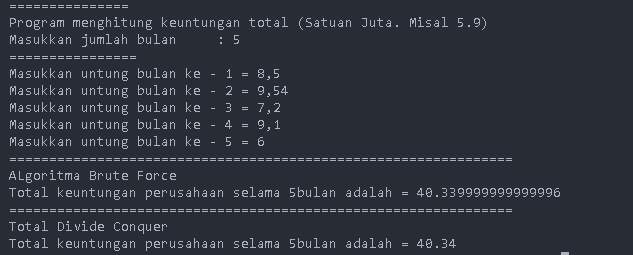
1. Tampilkan hasil perhitungan melalui objek yang telah dibuat untuk kedua cara yang ada (Brute Force dan Divide and Conquer)



* + 1. Verifikasi Hasil Percobaan

Cocokkan hasil compile kode program anda dengan gambar berikut ini.



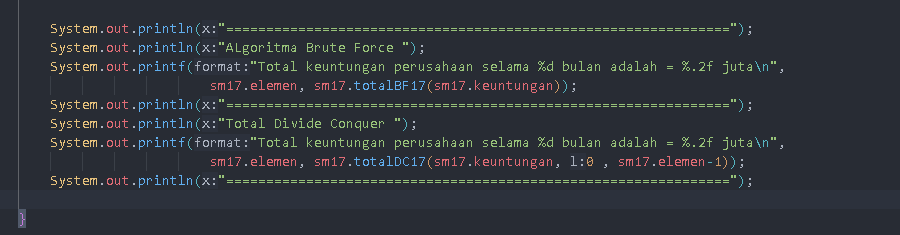
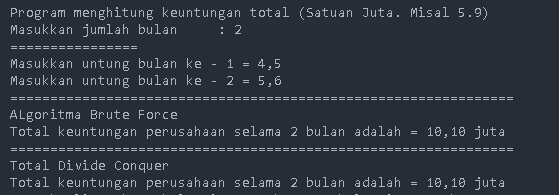


Pertanyaan

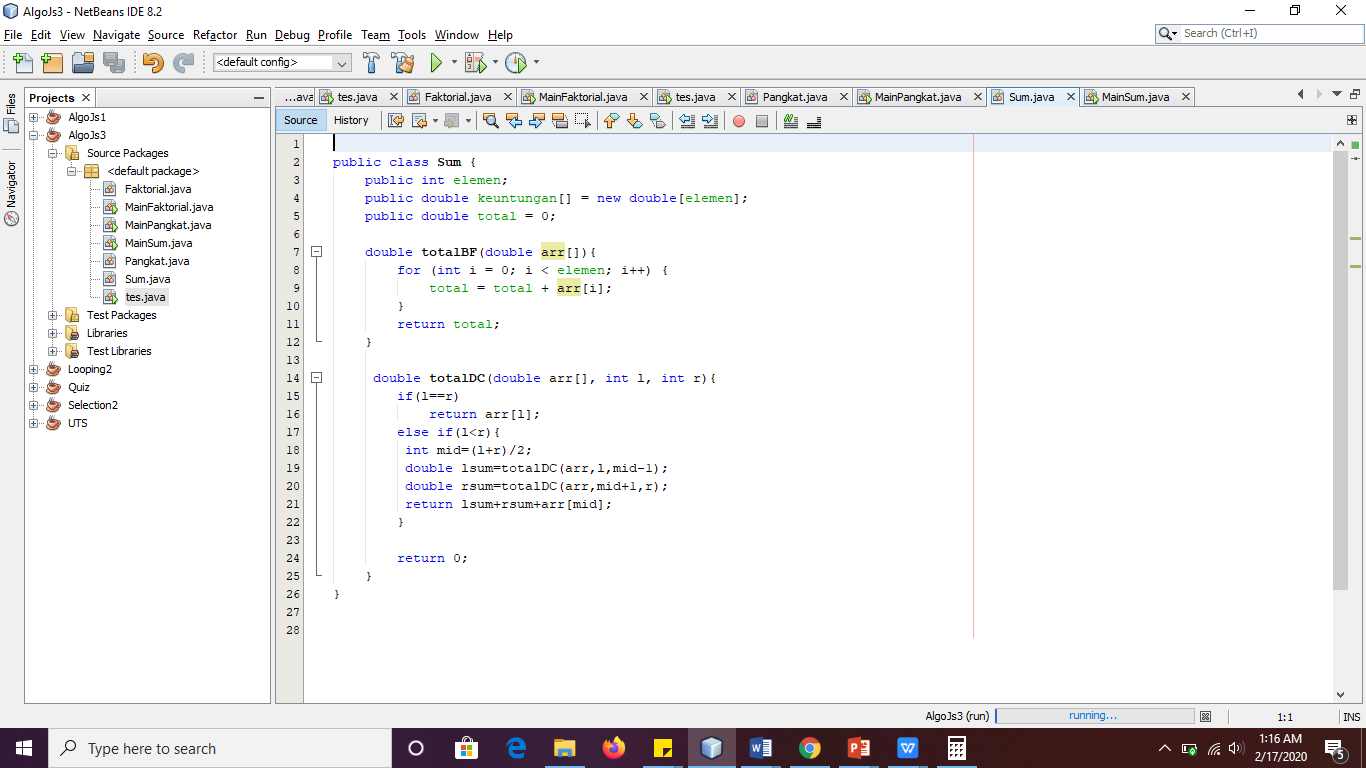
1. Berikan ilustrasi perbedaan perhitungan keuntungan dengan method TotalBF() ataupun TotalDC()

* TotalBF (Brute Force): Dalam metode Brute Force, total keuntungan dihitung dengan menjumlahkan semua nilai keuntungan dari setiap bulan secara berurutan.
* TotalDC (Divide and Conquer): Dalam metode Divide and Conquer, total keuntungan dihitung dengan membagi masalah menjadi submasalah yang lebih kecil dan menggabungkan hasilnya. Pada kasus ini, submasalah adalah membagi daftar keuntungan menjadi dua bagian dan menghitung total untuk setiap bagian, kemudian menggabungkan total tersebut dengan nilai tengah (mid).

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.



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

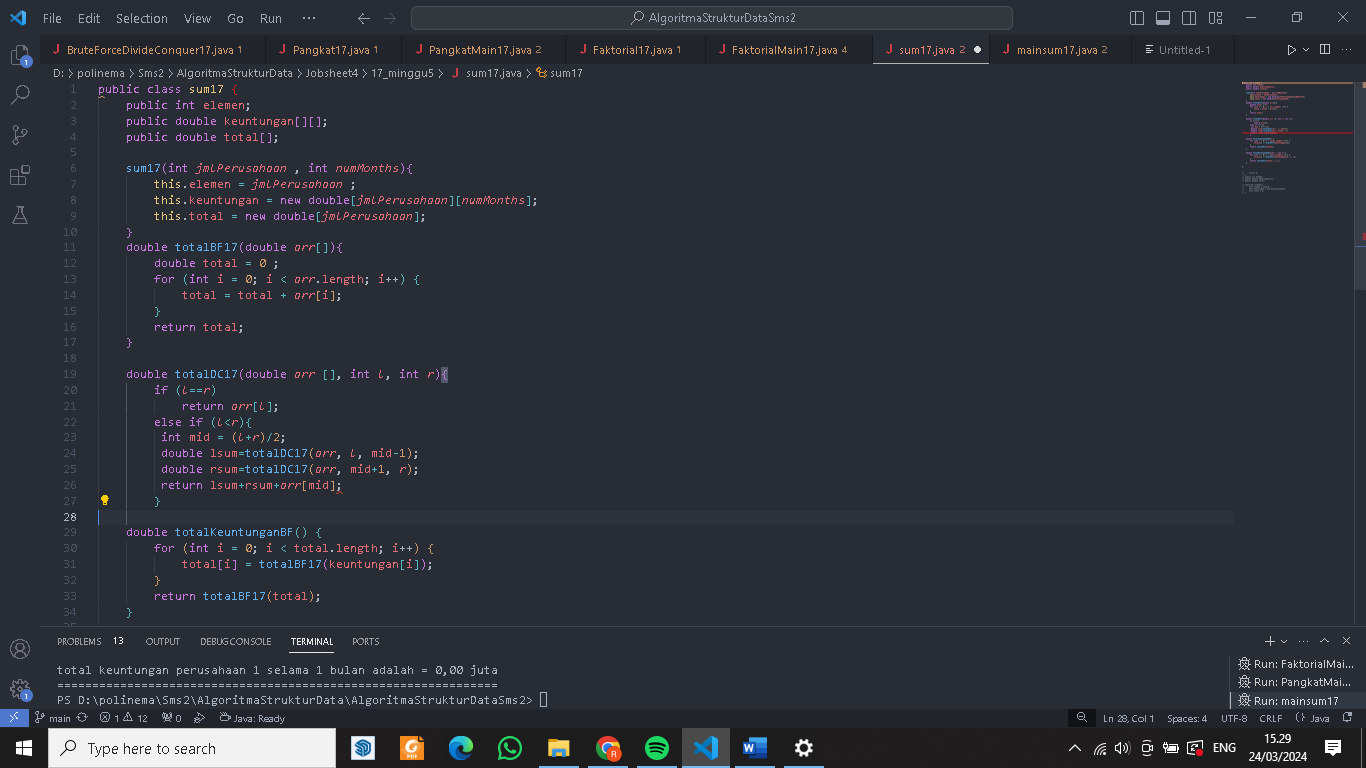


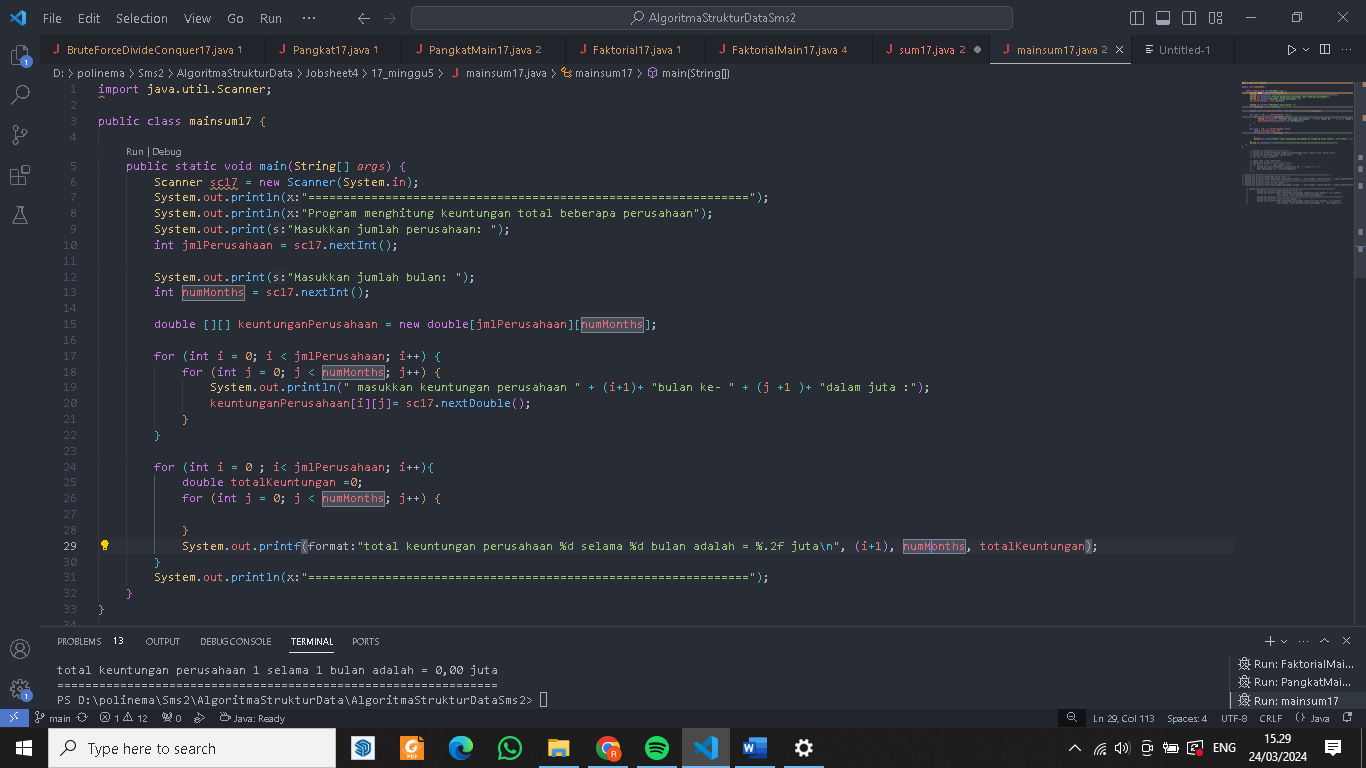
Pada totalDC17, formulasi return value yang menggunakan lsum + rsum + arr[mid] adalah untuk menggabungkan total keuntungan dari submasalah kiri (lsum), submasalah kanan (rsum), dan nilai tengah (arr[mid]). Ini mengikuti konsep dari Divide and Conquer di mana masalah dibagi menjadi submasalah yang lebih kecil dan kemudian digabungkan kembali.

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

Variabel mid pada totalDC17 digunakan untuk menentukan nilai tengah dari subarray yang sedang diproses. Hal ini penting karena dalam Divide and Conquer, kita perlu membagi masalah menjadi submasalah yang lebih kecil, dan nilai tengah ini digunakan untuk membagi submasalah menjadi dua bagian yang seimbang.

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!





* 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.*