**Nama : Deanissa Sherly Sabilla**

**Kelas / Absen : SIB 1B / 06**

**JOBSHEET IV**

**BRUTE FORCE DAN DIVIDE CONQUER**

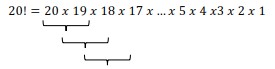
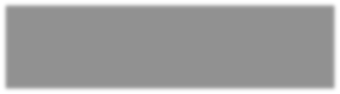
# 4.2 Menghitung Nilai Faktorial dengan Algoritma Brute Force dan Divide and Conquer

Perhatikan Diagram Class berikut ini :

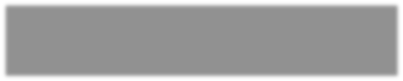
|  |
| --- |
| Faktorial |
| nilai: int |
| faktorialBF(): int faktorialDC(): int |

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 :



## 4.2.1 Langkah-langkah Percobaan

1. Buat Project baru, dengannama “**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()

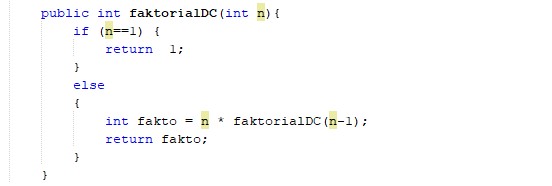
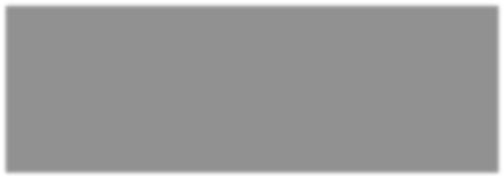
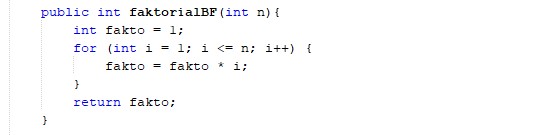
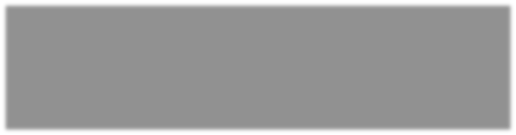
nilai

c)

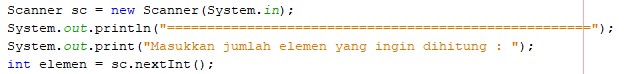
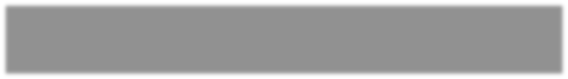
Tambahkan method

faktorialDC()

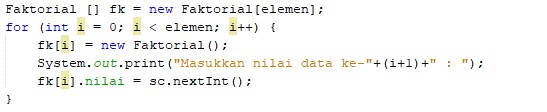
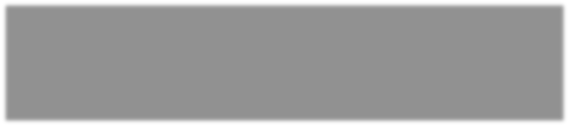
nilai



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



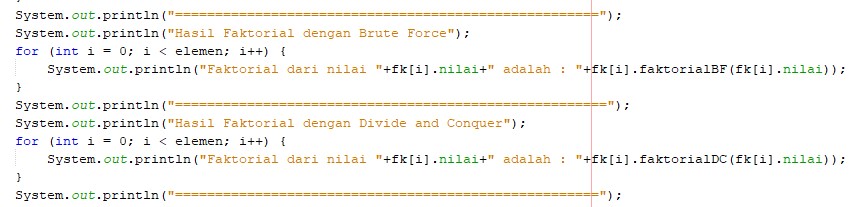
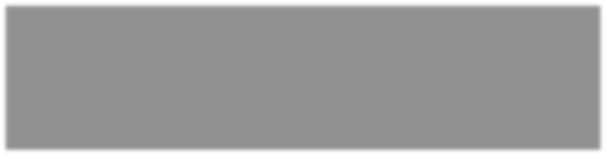
* 1. Buat Array of Objek pada fungsi main, kemudian inputkan beberapa nilai yang akan dihitung faktorialnya



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

d)

Pastikan program sudah berjalan dengan baik!

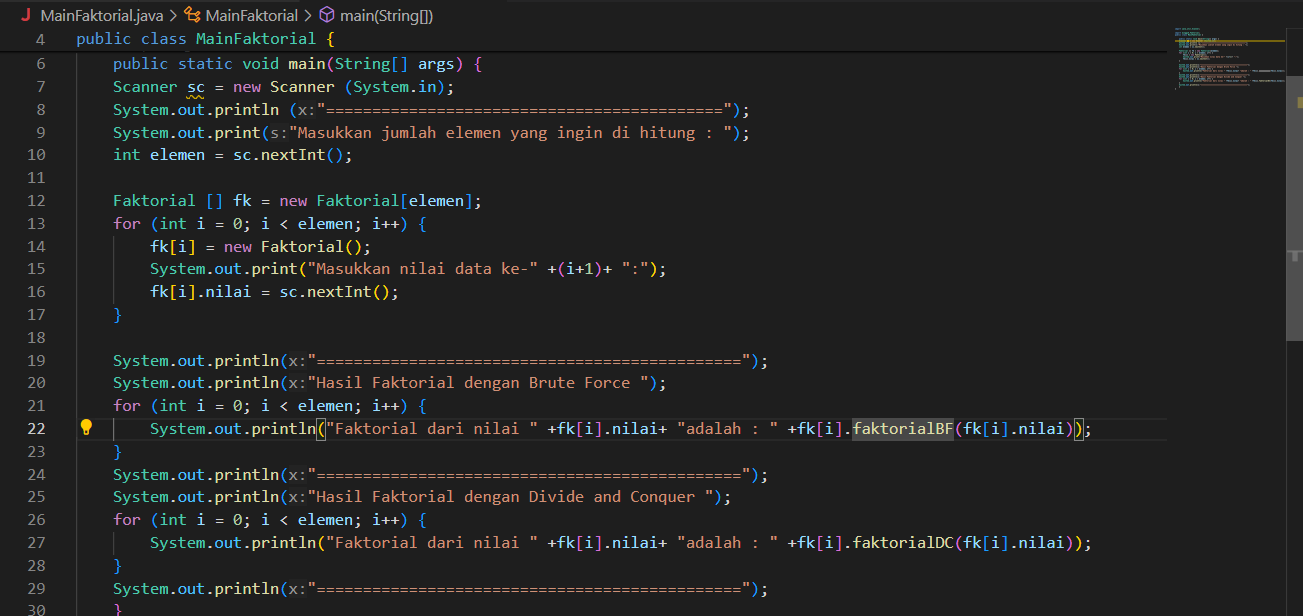


* **HASIL INPUT :**

**Class Faktorial :**

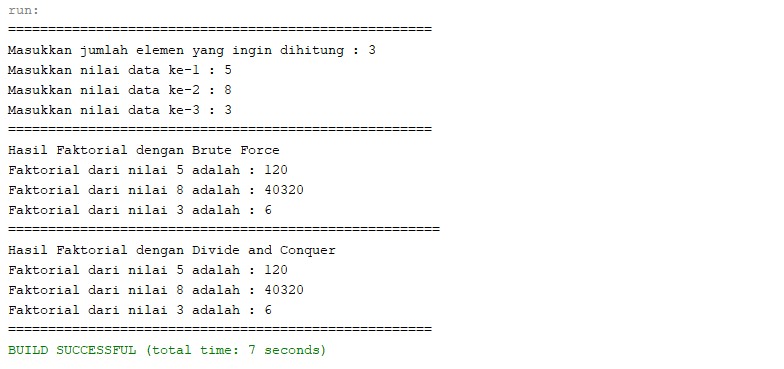
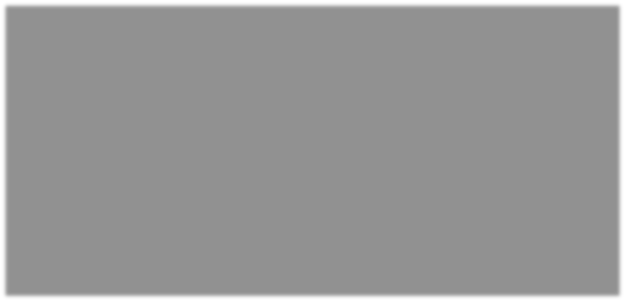


**Class Main :**

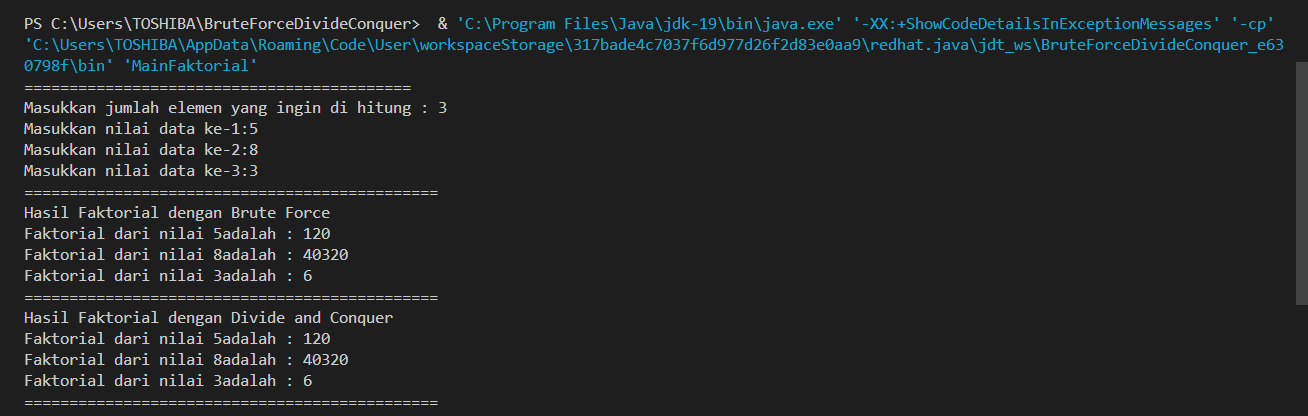


## 4.2.2 Verifikasi Hasil Percobaan

Cocokkan hasil compile kode program anda dengan gambar berikut ini.



* **HASIL OUTPUT :**

****

## 4.2.3 Pertanyaan

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

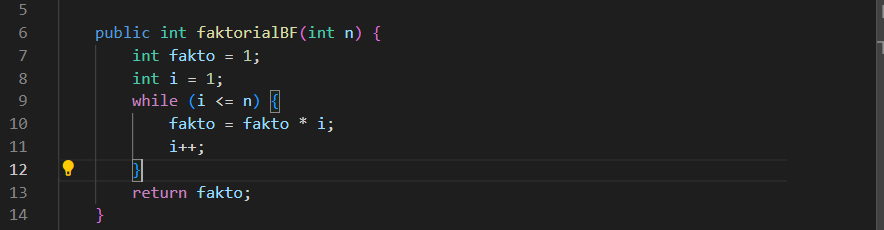
* Nilai n sama dengan 1, Pada saat, hasil faktorial dari 1 adalah 1. Jadi, jika nilai n sama dengan 1, maka langsung mengembalikan nilai 1 sebagai hasil faktorial.
* Ketika nilai n lebih besar dari 1, maka membagi masalah menjadi dua bagian. Dalam kasus ini, dapat membagi masalah menjadi perhitungan faktorial dari n dan perhitungan faktorial dari n-1. Ini dilakukan dengan mengalikan nilai n dengan hasil faktorial dari n-1.
* Langkah ini melibatkan solusi rekursif terhadap setiap submasalah yang dihasilkan. Dalam hal ini, memanggil fungsi faktorialDC secara rekursif untuk menghitung faktorial dari n-1.
* Solusi untuk setiap submasalah telah diperoleh melalui rekursif, dan mengalikan nilai n dengan hasil faktorial dari n-1 untuk mendapatkan hasil akhir.

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** : Saat memanggil metode faktorialDC dengan argumen n, secara tidak langsung membagi masalah menjadi dua bagian: mengalikan n dengan hasil faktorial dari n-1. Ini merupakan konsep pembagian yang mendasari algoritma Divide and Conquer.
* **Conquer** : Saat n tidak sama dengan 1, kita secara rekursif memanggil metode faktorialDC dengan argumen n-1. Dalam konteks ini, penaklukkan terjadi saat kita menyelesaikan submasalah dengan menghitung faktorial dari n-1.
* **Combine** : Setiap pemanggilan rekursif untuk menghitung faktorial dari n-1 menghasilkan solusi untuk submasalah tersebut. Hasil solusi submasalah dikalikan dengan n (bagian penaklukkan) untuk memberikan solusi akhir dari masalah. Meskipun tidak ada langkah penggabungan yang jelas, tetapi hasil penggabungan terjadi melalui proses rekursif dan operasi perkalian.

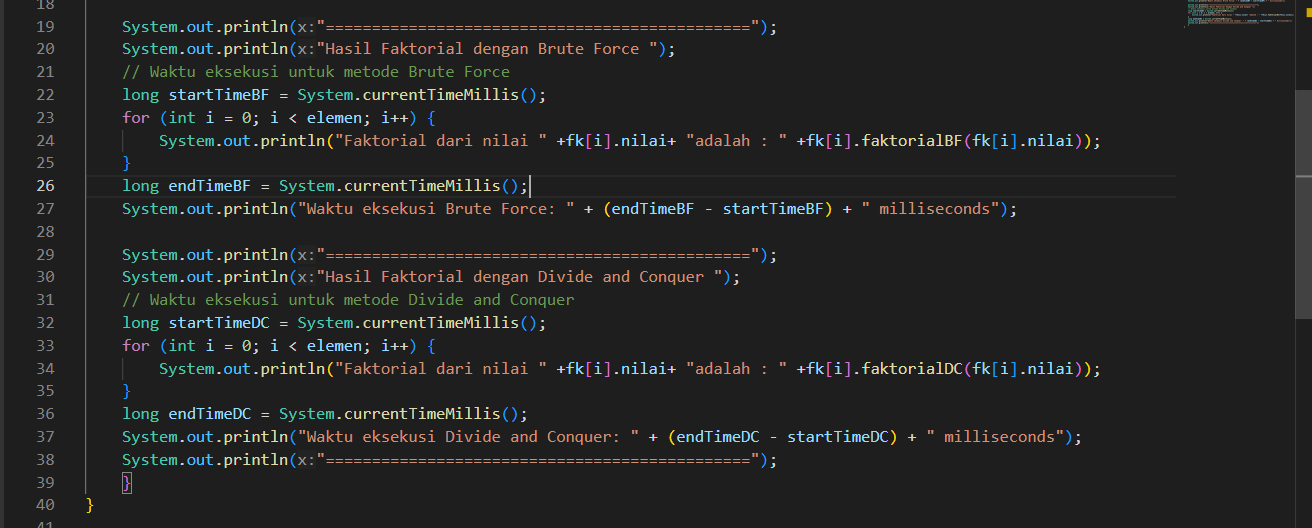
1. Apakah memungkinkan perulangan pada method faktorialBF() dirubah selain menggunakan for?Buktikan!

* memungkinkan untuk mengimplementasikan perulangan dalam metode **faktorialBF()** selain menggunakan for loop. Dapat menggunakan pendekatan lain seperti **while loop**. Berikut adalah implementasi menggunakan **while** :

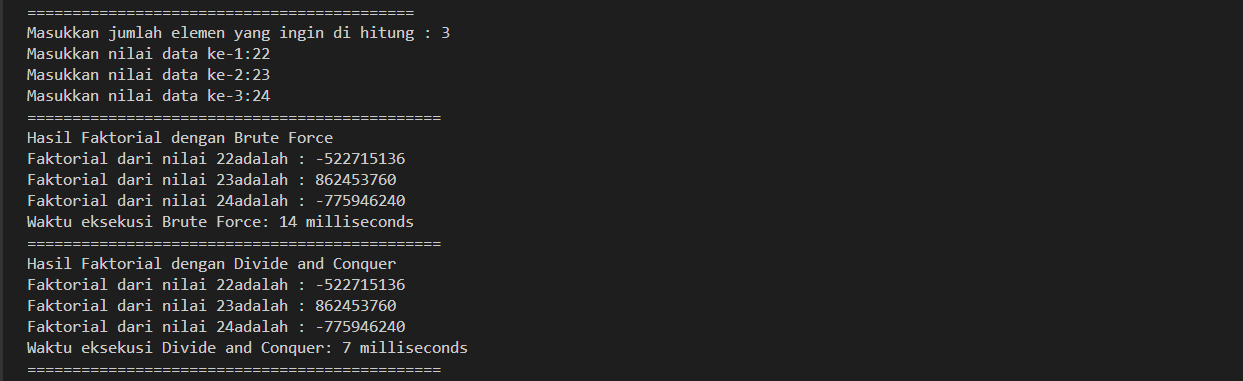


1. Tambahkan pegecekan waktu eksekusi kedua jenis method tersebut!

* Menambahkan pada **class main** untuk pengecekan waktu eksekusi



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

* 

# Dalam contoh output ini, dapat melihat bahwa waktu eksekusi untuk kedua metode faktorial (Brute Force dan Divide and Conquer) menunjukkan perbedaan yang signifikan, bahkan dengan nilai yang besar seperti 22, 23, dan 24.

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

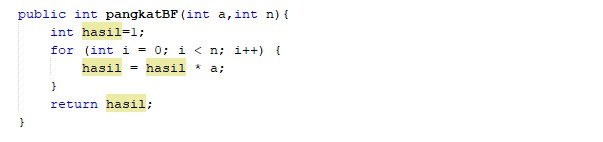
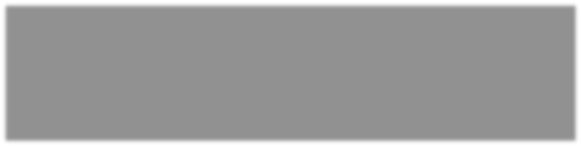
Pada praktikum ini kita akan membuat program class dalam Java. Untuk menghitung nilai pangkat suatu angka menggunakan 2 jenis algoritma, Brute Force dan Divide and Conquer.

## 4.3.1 Langkah-langkah Percobaan

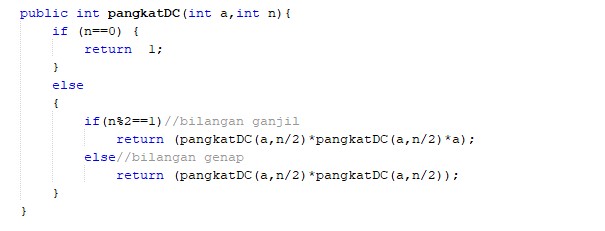
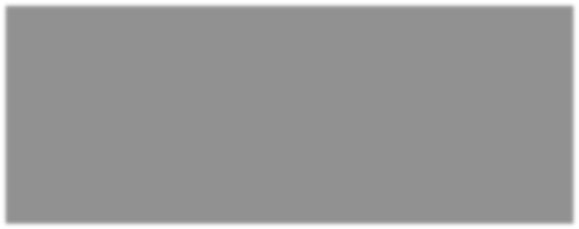
1. Di dalam paket **minggu5**, buatlah class baru dengan nama **Pangkat**. Dan di dalam class **Pangkat** tersebut, buat atribut angka yang akan dipangkatkan sekaligus dengan angka pemangkatnya



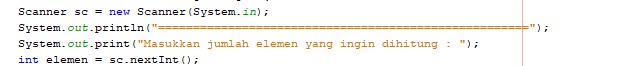
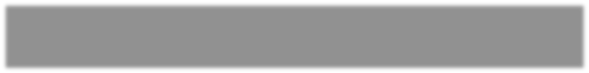
1. Pada class Pangkat tersebut, tambahkan method PangkatBF()



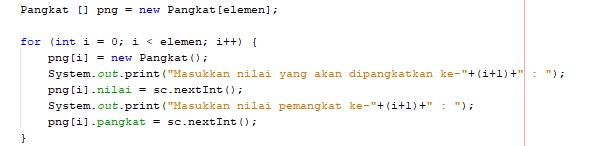
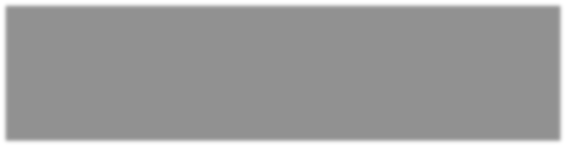
1. Pada class Pangkat juga tambahkan method PangkatDC()



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.



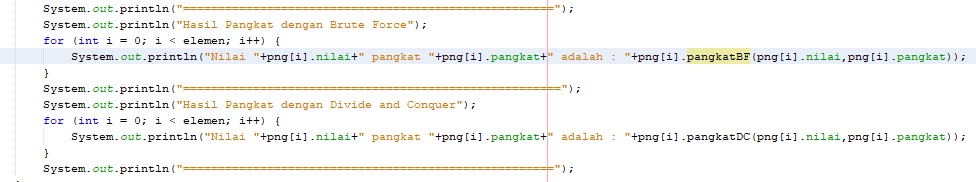
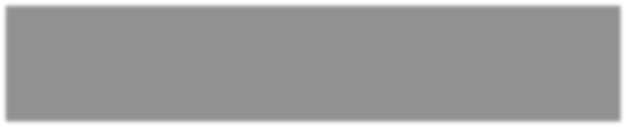
1. Nilai pada tahap 5 selanjutnya digunakan untuk instansiasi array of objek. Di dalam Kode berikut ditambahkan proses pengisian beberapa nilai yang akan dipangkatkan sekaligus dengan pemangkatnya.



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

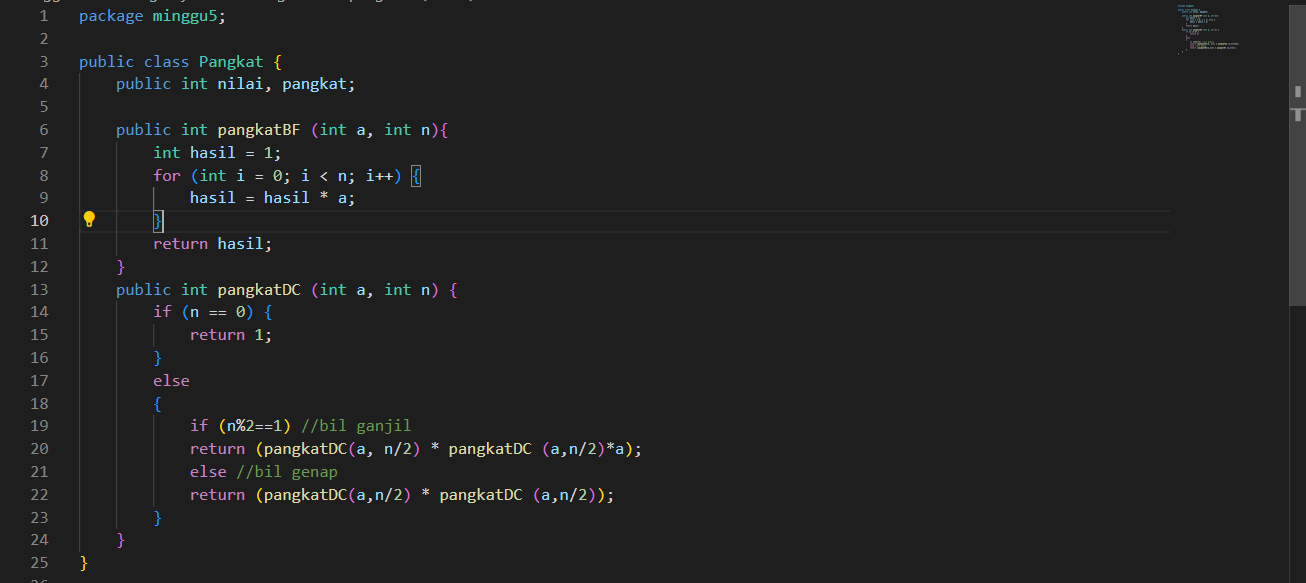
PangkatDC()

.

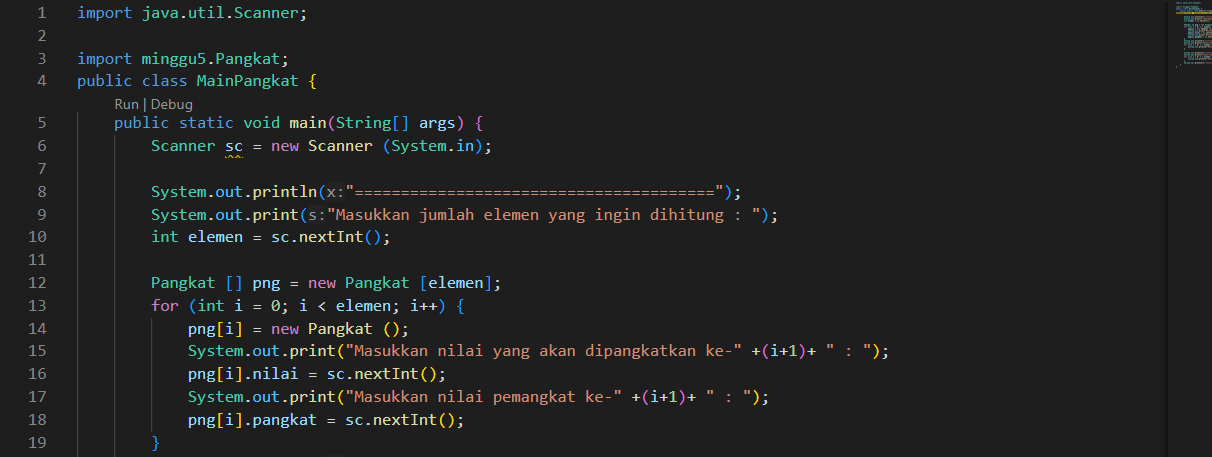


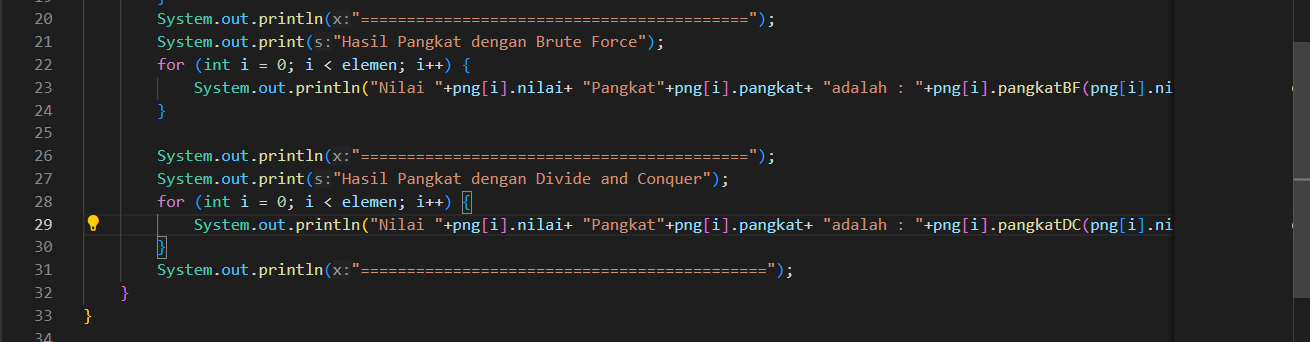
* **HASIL INPUT :**

**Class Pangkat :**



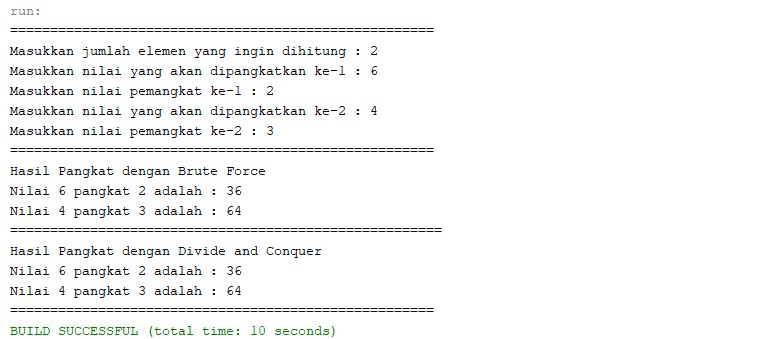
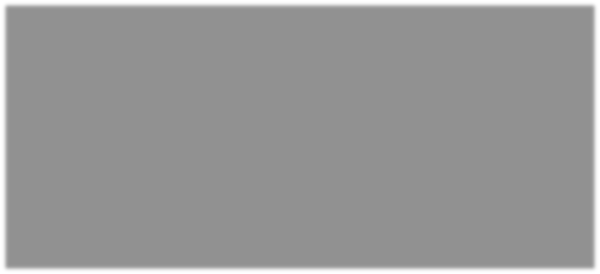
**Class Main :**



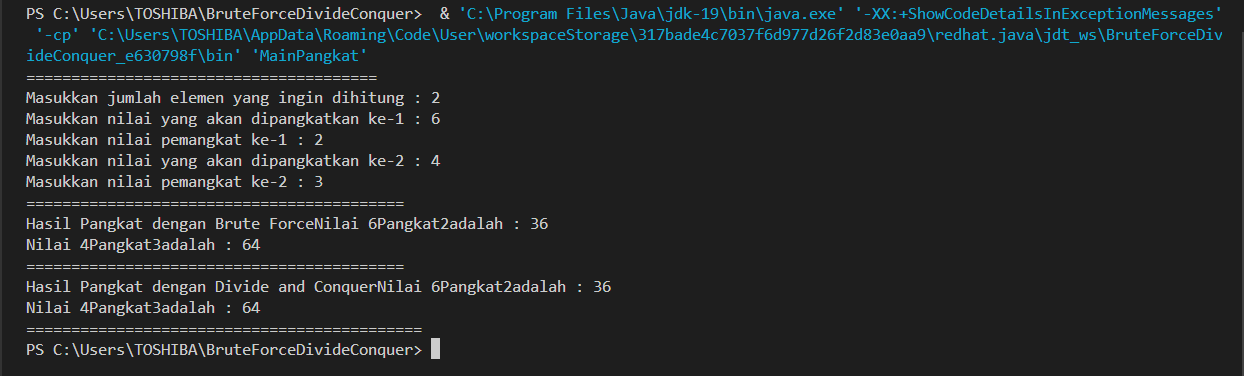


## 4.3.2 Verifikasi Hasil Percobaan

Pastikan output yang ditampilkan sudah benar seperti di bawah ini.



* **HASIL OUTPUT :**

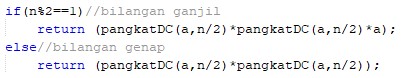
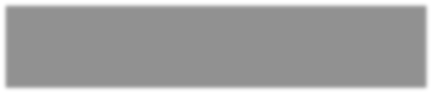
****

## 4.3.3 Pertanyaan

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

* **PangkatBF :**
* Metode ini menggunakan perulangan untuk menghitung pangkat dari sebuah bilangan.
* Dalam implementasinya, metode ini melakukan perulangan sebanyak n kali, di mana n adalah pangkat yang diinginkan.
* Pada setiap iterasi, metode ini mengalikan hasil sebelumnya dengan bilangan basis.
* Kompleksitas waktu dari metode ini adalah O(n), di mana n adalah pangkat yang diinginkan.
* **PangkatDC :**
* Metode ini menggunakan rekursif untuk menghitung pangkat dari sebuah bilangan.
* Dalam implementasinya, metode ini membagi perhitungan pangkat menjadi dua bagian yang lebih kecil dan menggabungkan hasilnya.
* Kompleksitas waktu dari metode ini adalah O(log n), di mana n adalah pangkat yang diinginkan.

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

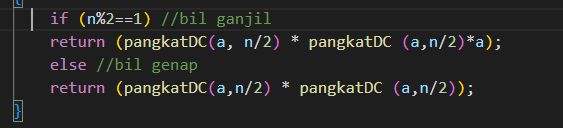


Jelaskan arti potongan kode tersebut

* **if (n % 2 == 1):** Ini adalah kondisi untuk memeriksa apakah pangkat n ganjil atau genap. Jika sisa pembagian n dengan 2 akan menghasilkan 1 jika n ganjil, dan 0 jika n genap.
* **(pangkatDC (a, n/2) \* pangkatDC(a, n/2) \* a)**: Jika pangkat n adalah ganjil, maka metode pangkatDC akan memanggil dirinya sendiri dua kali dengan pangkat yang lebih kecil, yaitu n/2, kemudian hasilnya dikalikan dengan basis a.
* **(pangkatDC (a, n/2) \* pangkatDC(a, n/2))**: Jika pangkat n adalah genap, maka metode pangkatDC akan memanggil dirinya sendiri dua kali dengan pangkat yang lebih kecil, yaitu n/2, dan hasilnya dikalikan bersama.

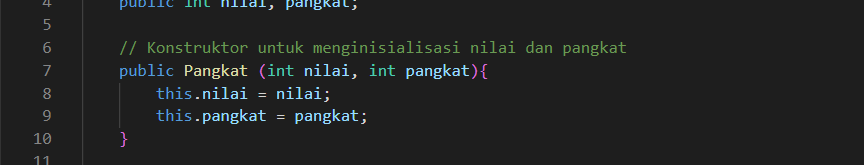
1. Apakah tahap *combine* sudah termasuk dalam kode tersebut?Tunjukkan!

* Dalam **pangkatDC**, tahap combine a terjadi pada proses pengalihan hasil dari sub-problem yang lebih kecil menjadi hasil akhir. Ini terjadi ketika mengalikan hasil dari kedua pemanggilan rekursif dengan basis a untuk pangkat ganjil dan hanya mengalikan hasilnya bersama untuk pangkat genap.



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

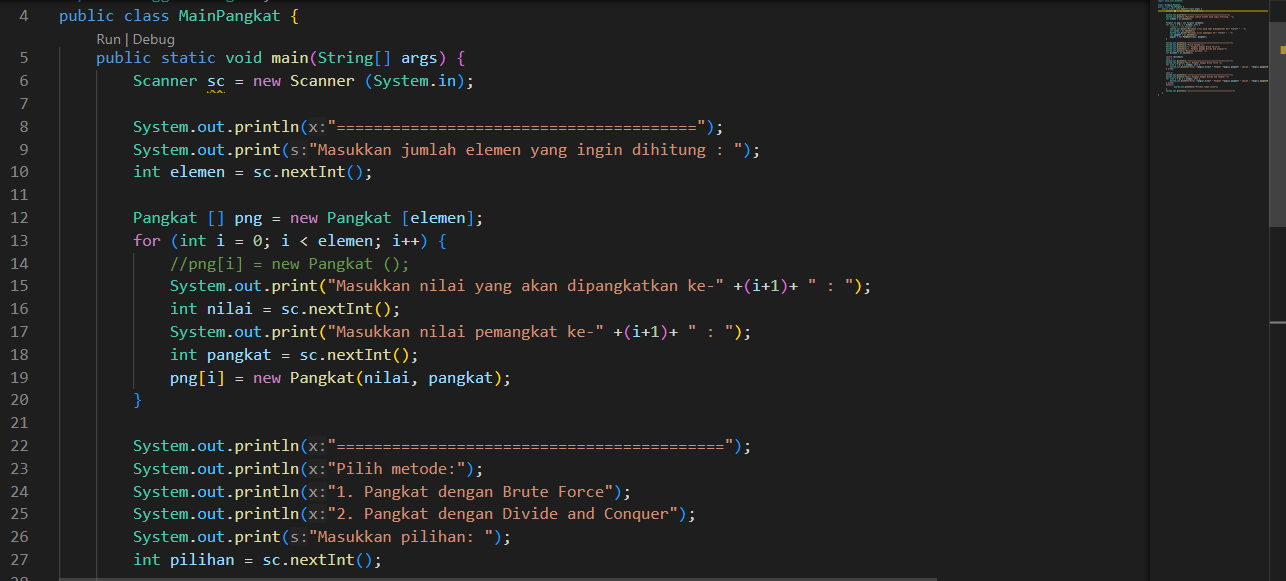
* Dengan menambah kontruktor pada **class Pangkat**

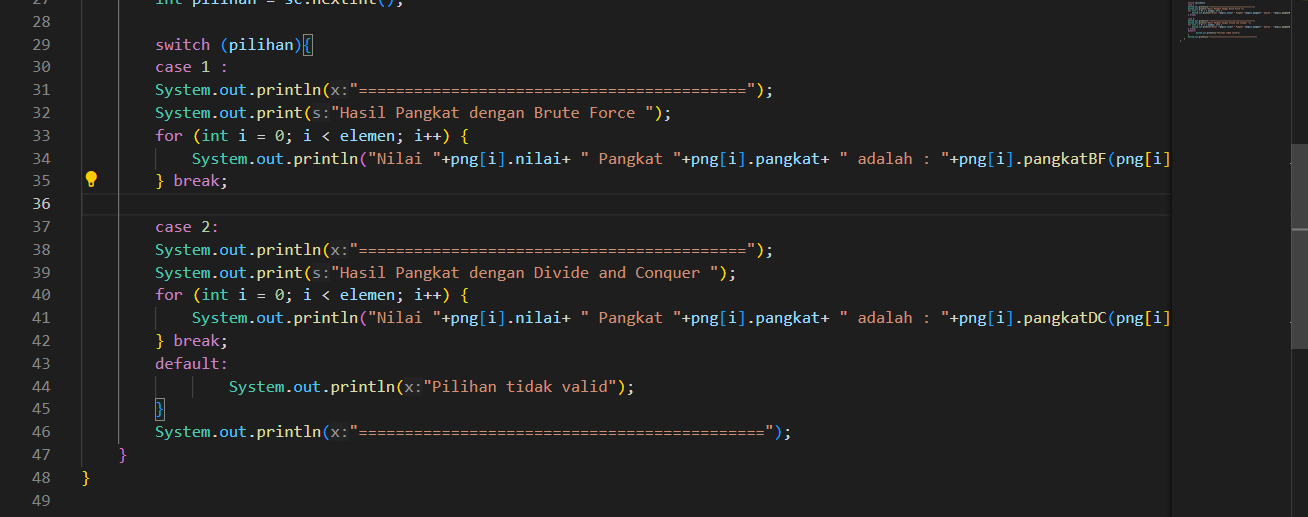


1. Tambahkan menu agar salah satu method yang terpilih saja yang akan dijalankan!

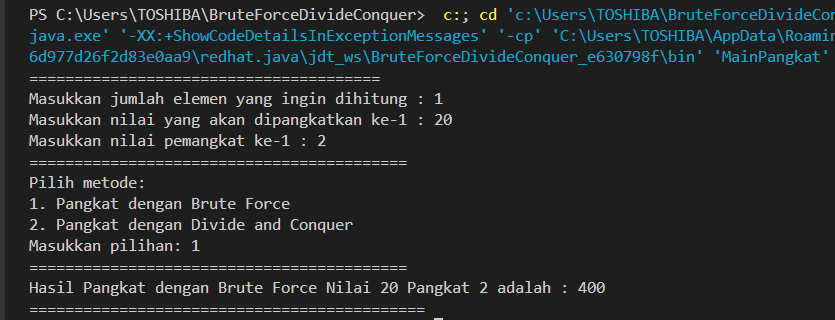
* **Input** : (Dengan menambahkan kode program **pilihan** metode di class main)

**Class Main :**

****

****

* **Hasil Output :**

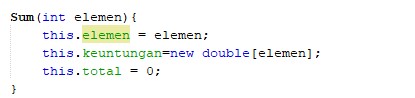
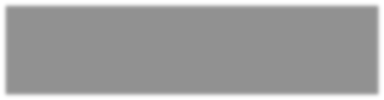
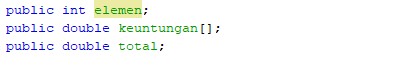
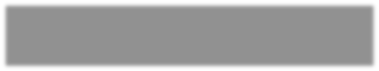


# 4.4 Menghitung Sum Array dengan Algoritma Brute Force dan Divide and Conquer

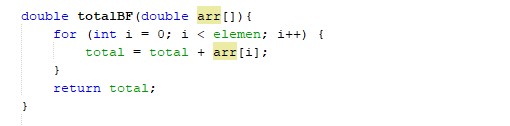
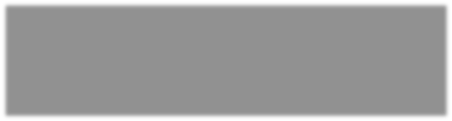
Di dalam percobaan ini, kita akan mempraktekkan bagaimana proses *divide, conquer*, dan *combine* diterapkan pada studi kasus penjumlahan keuntungan suatu perusahaan dalam beberapa bulan.

## 4.4.1 Langkah-langkah Percobaan

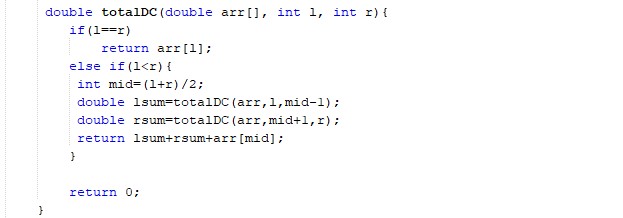
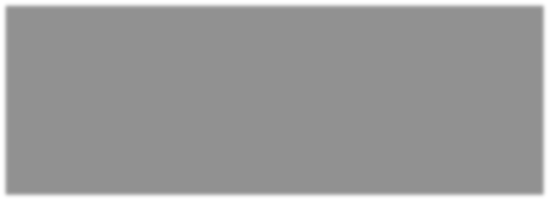
1. Pada paket minggu5. Buat class baru yaitu class Sum. DI salam class tersebut terdapat beberapa atribut jumlah elemen array, array, dan juga total. Tambahkan pula konstruktor pada class Sum.



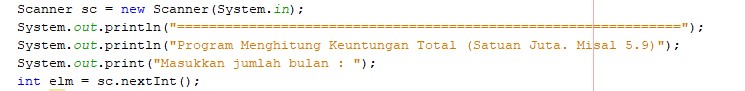
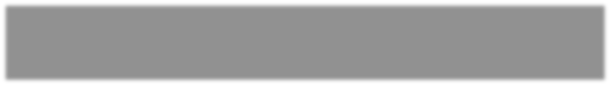
1. Tambahkan method TotalBF() yang akan menghitung total nilai array dengan cara *iterative*.



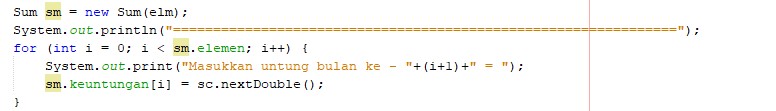
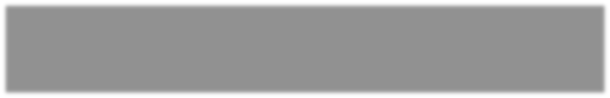
1. Tambahkan pula method TotalDC() untuk implementasi perhitungan nilai total array menggunakan algoritma Divide and Conquer



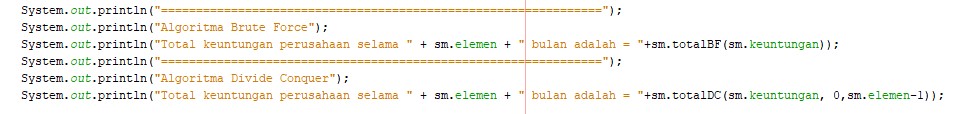
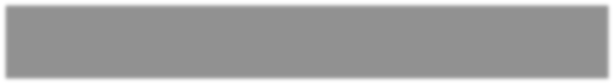
1. Buat class baru yaitu MainSum. Di dalam kelas ini terdapat method main. Pada method ini user dapat menuliskan berapa bulan keuntungan yang akan dihitung. Dalam kelas ini sekaligus dibuat instansiasi objek untuk memanggil atribut ataupun fungsi pada class Sum



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.

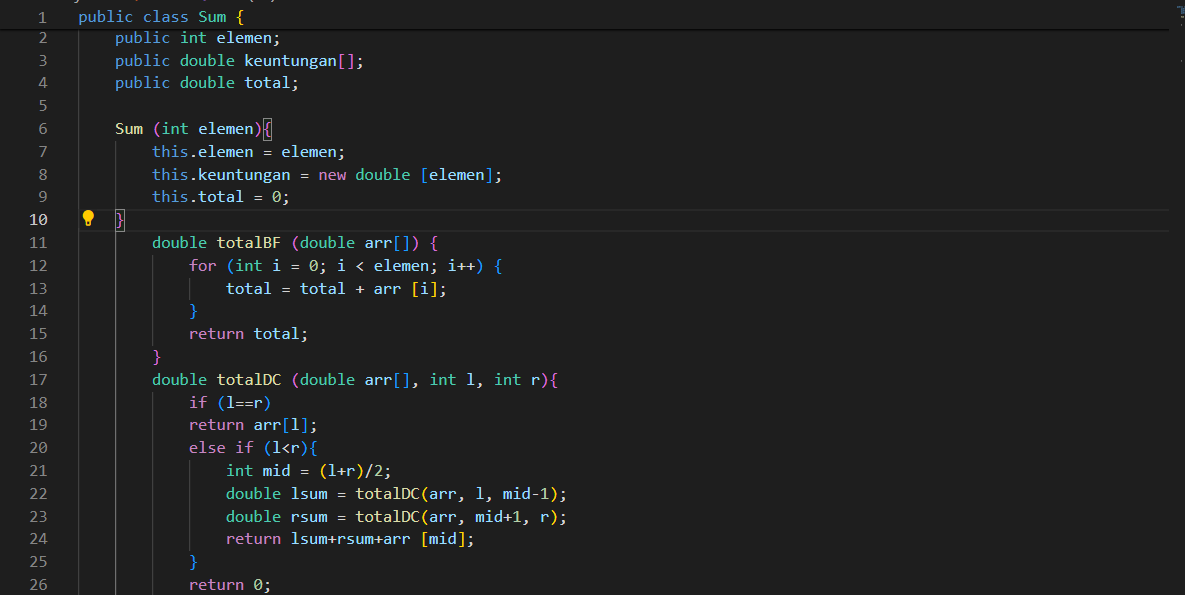


1. Tampilkan hasil perhitungan melalui objek yang telah dibuat untuk kedua cara yang ada (Brute Force dan Divide and Conquer)



* **HASIL INPUT :**

**Class Sum :**

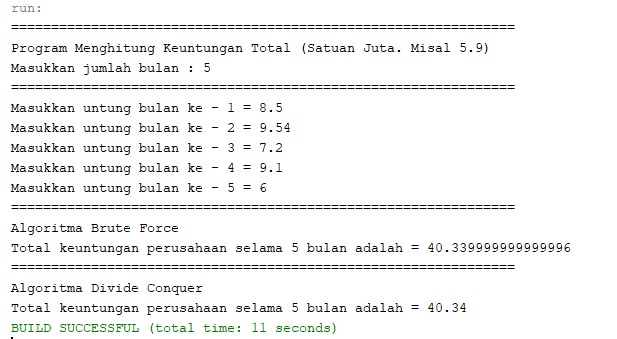
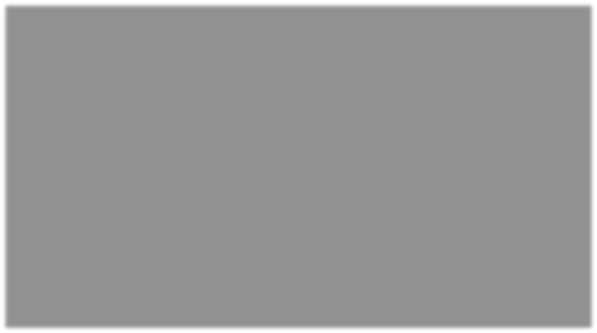
****

**Class Main :**

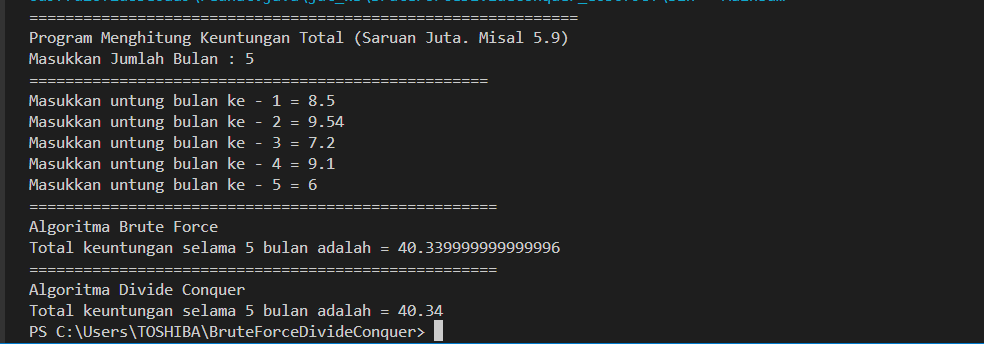
****

## 4.4.2 Verifikasi Hasil Percobaan

Cocokkan hasil compile kode program anda dengan gambar berikut ini.



* **HASIL OUTPUT :**

****

## 4.4.3 Pertanyaan

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

TotalDC()

* **TotalBF() :** Dalam metode ini, secara langsung menjumlahkan semua elemen array.

Total keuntungan dihitung dengan rumus:

Total = keuntungan[0] + keuntungan[1] + ... + keuntungan[n-1]

* **TotalDC() :** Dalam metode ini, membagi array menjadi dua bagian, menghitung total keuntungan untuk setiap bagian, dan kemudian menggabungkan hasilnya.

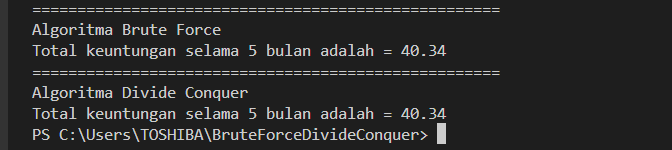
1. Perhatikan output dari kedua jenis algoritma tersebut bisa jadi memiliki hasil berbeda di belakang koma. Bagaimana membatasi output di belakang koma agar menjadi standar untuk kedua jenis algoritma tersebut.

* Untuk membatasi output di belakang koma agar menjadi standar untuk kedua jenis algoritma, dapat menggunakan **format angka desimal** dengan jumlah digit yang diinginkan.

***DecimalFormat df = new DecimalFormat("#.##");***

Dengan menggunakan DecimalFormat dan pola "#.##", memastikan bahwa hasil akan ditampilkan dengan tepat dua digit di belakang koma.

Maka hasil output :



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



* Pada kode **return lsum + rsum + arr[mid];** formulasi tersebut digunakan untuk menggabungkan hasil dari bagian kiri, bagian kanan, dan elemen tengah (pada indeks mid) dari array keuntungan. Dengan menggabungkan ketiga komponen ini, mendapatkan total keuntungan untuk seluruh array, dan kemudian dikembalikan sebagai hasil dari totalDC.

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

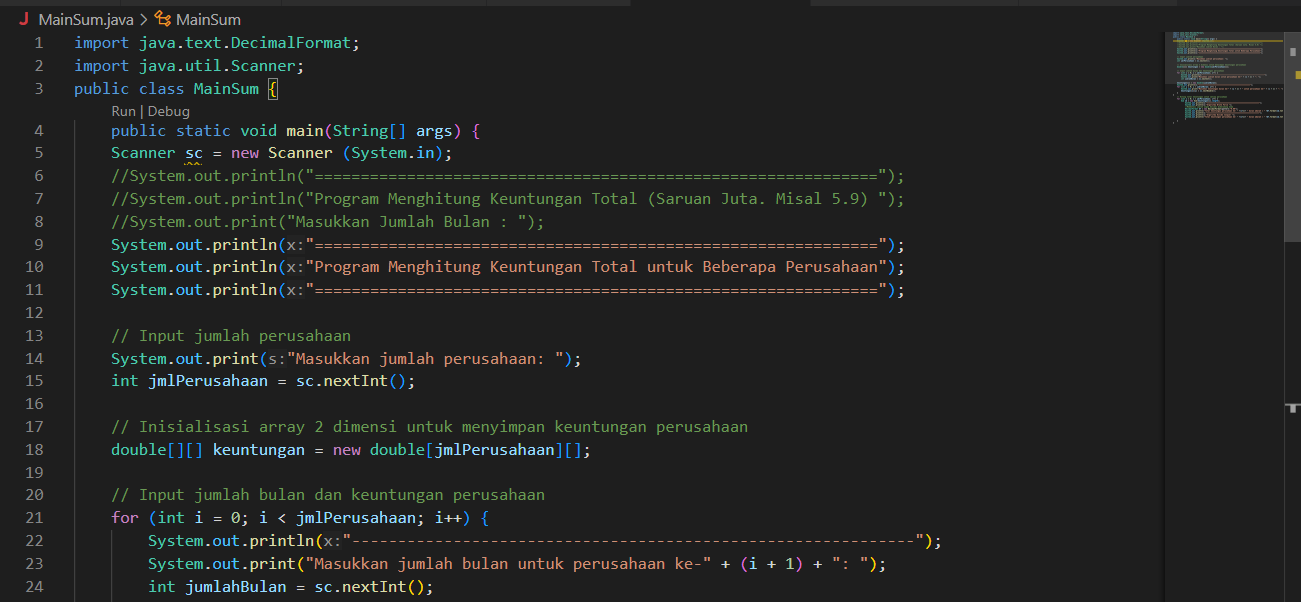
* Variabel **mid** digunakan dalam **totalDC()** karena itu adalah titik tengah yang digunakan untuk membagi array menjadi dua bagian saat menerapkan pendekatan Divide and Conquer.

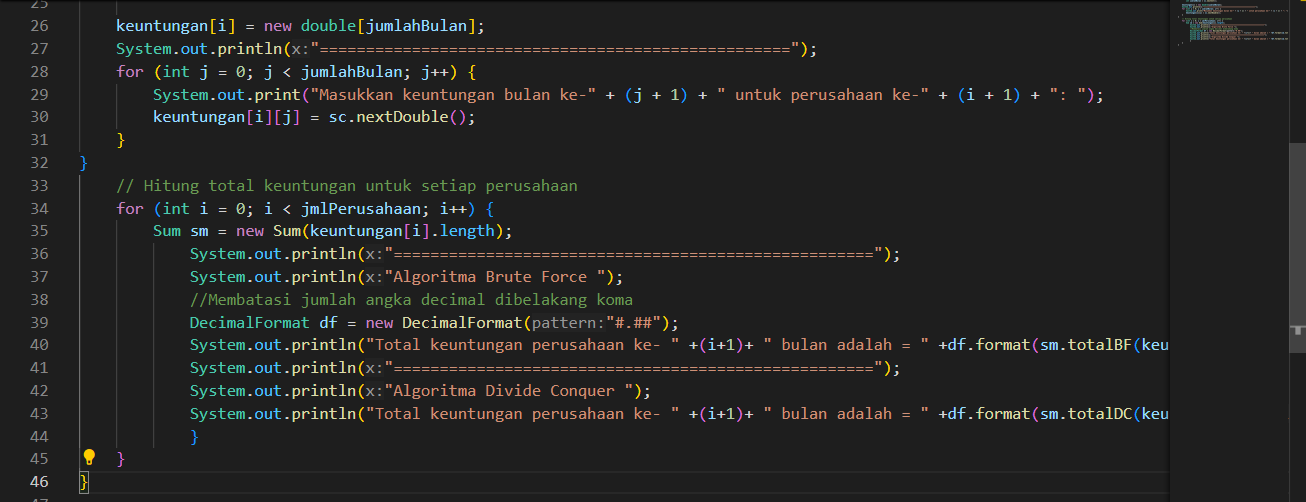
1. Program perhitungan keuntungan suatu perusahaan ini hanya untuk satu perusahaan saja. Bagaimana cara menghitung sekaligus keuntungan beberapa bulan untuk beberapa perusahaan.(Setiap perusahaan bisa saja memiliki jumlah bulan berbeda-beda)? Buktikan dengan program!

* Memodifikasi program agar dapat menghitung keuntungan beberapa bulan untuk beberapa perusahaan yang memiliki jumlah bulan berbeda-beda, dapat menggunakan struktur data yang sesuai untuk menyimpan data keuntungan untuk setiap perusahaan. Salah satu pendekatan yang bisa digunakan adalah menggunakan array 2 dimensi, di mana setiap baris mewakili satu perusahaan dan setiap kolom mewakili keuntungan bulanan.

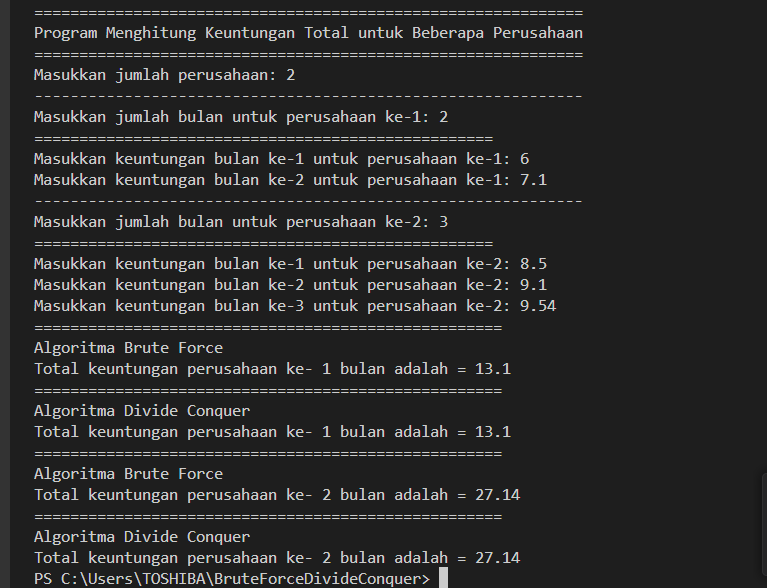
Berikut merupakan untuk **kode program :**

* **Hasil Input :**





* **Hasil Output :**



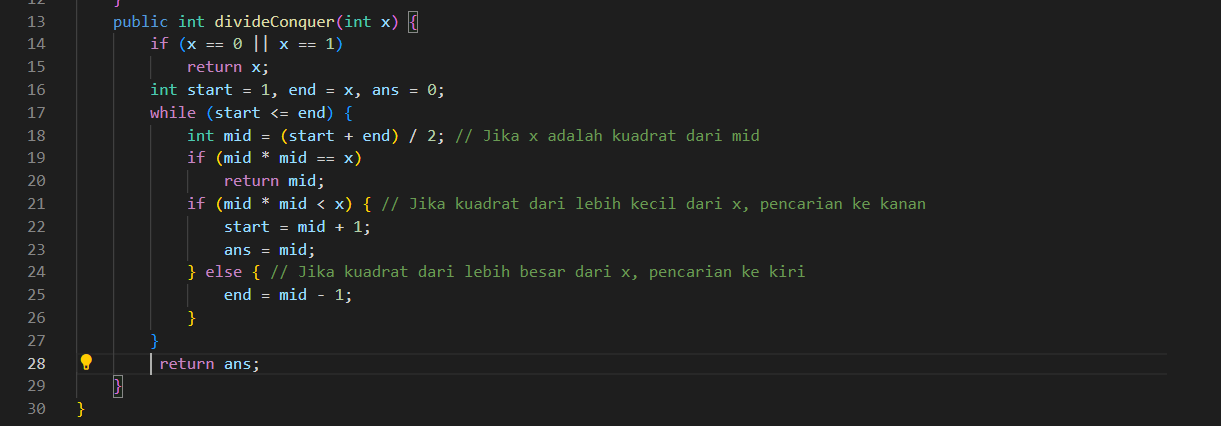
# 4.5 Latihan Praktikum

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.*

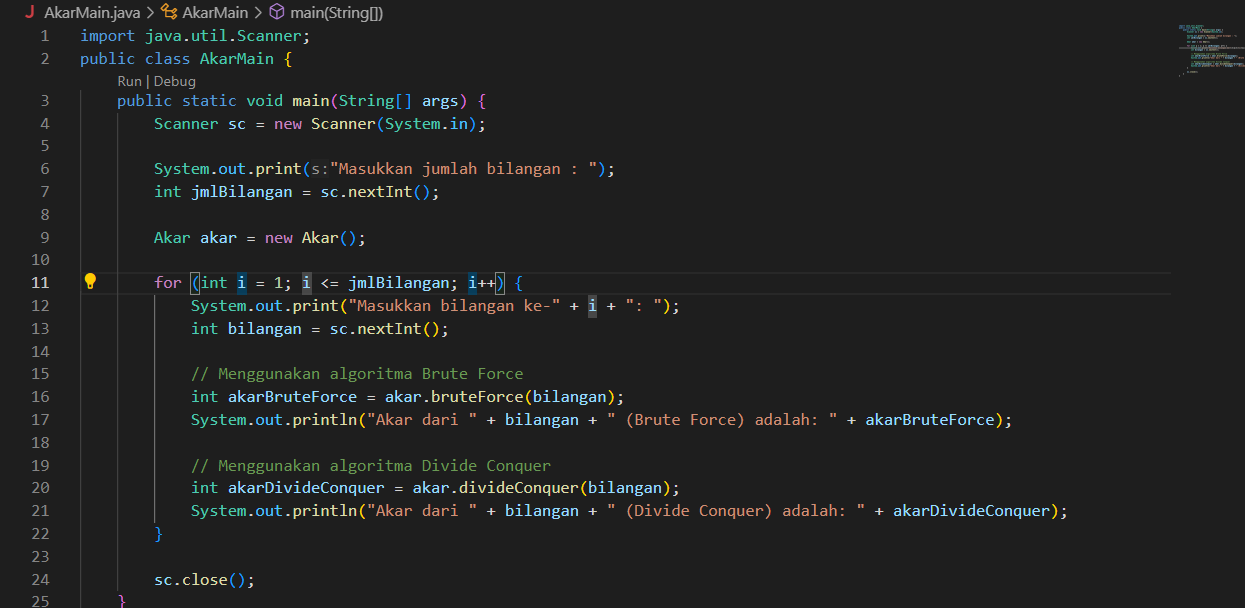
* **HASIL INPUT :**

**Class Akar :**

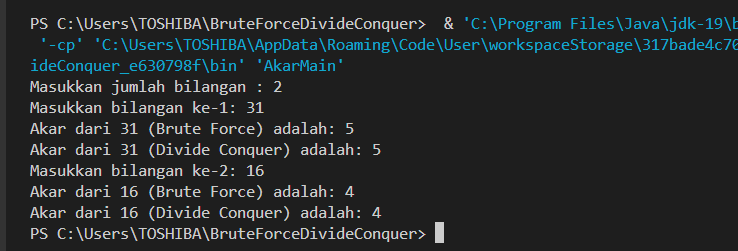
****

****

**Class Main :**

****

* **HASIL OUTPUT :**

****