**LAPORAN PRAKTIKUM**

**MATA KULIAH ALGORITMA DAN STRUKTUR DATA**

Dosen Pengampu : Triana Fatmawati, S.T, M.T

**PERTEMUAN - 5 Brute Force & Divide Conquer**

****

**Nama : M. Zidna Billah Faza**

**NIM : 2341760030**

**Prodi : D-IV Sistem Informasi Bisnis**

**JURUSAN TEKNOLOGI INFORMASI**

**POLITEKNIK NEGERI MALANG**

**2023**

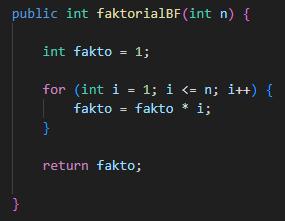
1. **PERCOBAAN 1**
2. Buatlah class baru dengan nama Faktorial



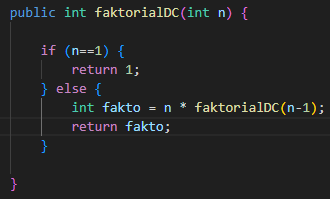
1. Lengkapi class Faktorial dengan atribut dan method yang telah digambarkan di dalam diagram class di atas, sebagai berikut:
2. Tambahkan atribut nilai



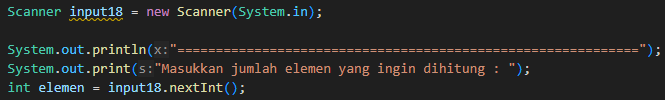
1. Tambahkan method faktorialBF() nilai



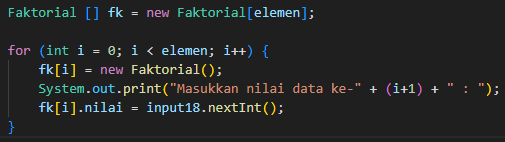
1. Tambahkan method faktorialDC() nilai



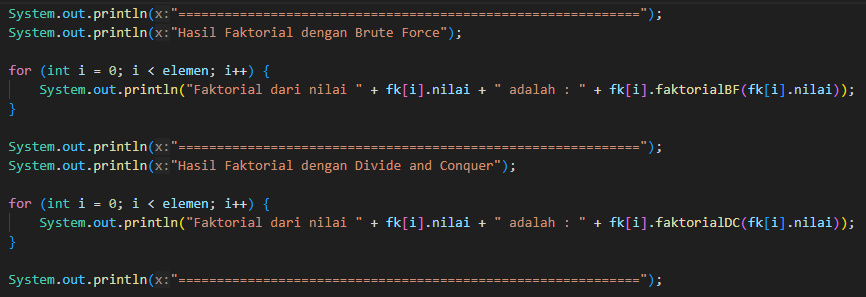
1. Coba jalankan (Run) class Faktorial dengan membuat class baru MainFaktorial.
2. Di dalam fungsi main sediakan komunikasi dengan user untuk menginputkan jumlah angka yang akan dicari nilai faktorialnya



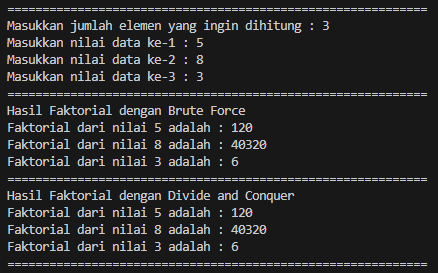
1. Buat Array of Objek pada fungsi main, kemudian inputkan beberapa nilai yang akan dihitung faktorialnya



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



1. Pastikan program sudah berjalan dengan baik!



Pertanyaan Percobaan 1

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

Algoritma Divide-and-Conquer untuk mencari nilai faktor membagi masalah menjadi submasalah yang lebih kecil, menyelesaikan setiap submasalah secara terpisah, dan menggabungkan solusi untuk mendapatkan solusi akhir.

Sebagai bagian dari pencarian faktorial, algoritma pertama-tama memeriksa apakah nilai yang diinginkan adalah 1. Jika demikian, ia segera mengembalikan nilai 1. Jika tidak, algoritma membagi permasalahan menjadi permasalahan yang lebih kecil dengan mengalikan nilai yangg ditemukan dengan faktorial dari nilai sebelumnya. Proses ini diulangi hingga kasus dasar tercapai, dimana nilai yang diinginkan adalah 1.

Hasil dari setiap langkah rekursi kemudian digabungkan untuk mendapatkan nilai faktor akhir. Oleh karena itu, algoritma ini menggunakan pendekatan rekursif untuk menyelesaikan masalah, menggabungkan solusi hingga sampai pada solusi 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!

Pada percobaan 1 hanya terdapat 2 tahapan yaitu Divide dan Conquer karena tahapan Combine tidak selalu eksplisit dalam implementasi faktorial menggunakan algoritma Divide and Conquer.

* Devide

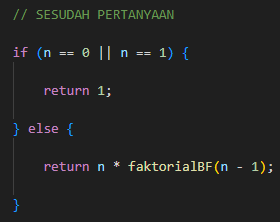
Terjadi ketika pada fungsi faktorialDC(int n), kita membagi masalah menjadi masalah yang lebih kecil dengan mengurangi nilai faktorial yang dicari (n) satu per satu hingga mencapai kasus dasar (n==1).

* Conquer

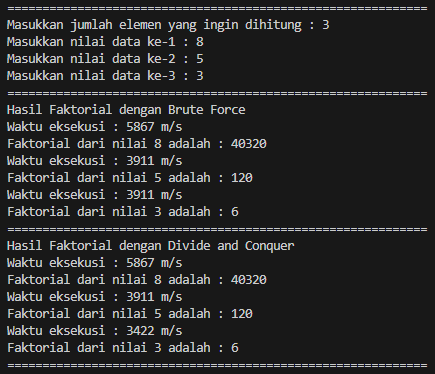
Terjadi di dalam blok else pada fungsi faktorialDC(int n). Pada tahap ini, kita menyelesaikan setiap submasalah secara rekursif dengan mengalikan nilai faktorial yang sedang dicari (n) dengan nilai faktorial dari masalah yang lebih kecil (faktorialDC(n-1)). Setiap langkah rekursif ini akan terus berulang sampai mencapai kasus dasar.

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

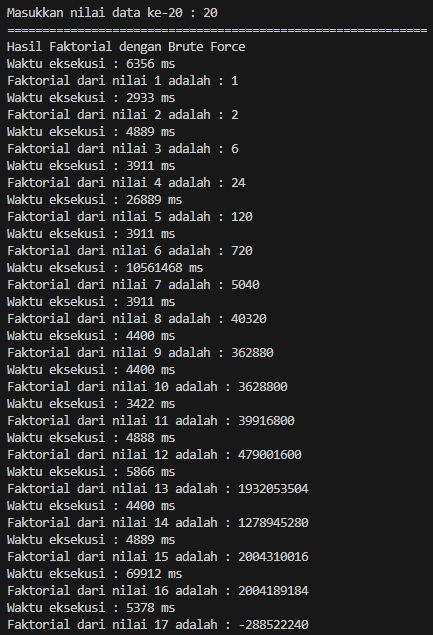
Perulangan pada method faktorialBF() dapat diubah menjadi rekursif dengan menggunakan if else seperti berikut :



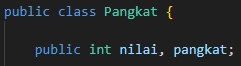
1. Tambahkan pegecekan waktu eksekusi kedua jenis method tersebut!



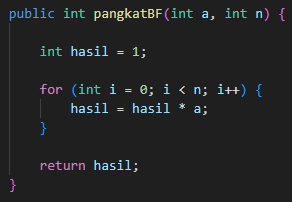
1. Buktikan dengan inputan elemen yang di atas 20 angka, apakah ada perbedaan waktu eksekusi?



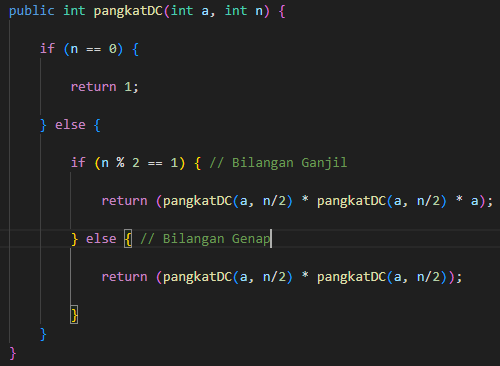
1. **PERCOBAAN 2**
2. Di dalam paket minggu5, buatlah class baru dengan nama Pangkat. Dan di dalam class Pangkat tersebut, buat atribut angka yang akan dipangkatkan sekaligus dengan angka pemangkatnya



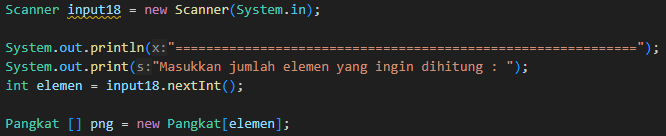
1. Pada class Pangkat tersebut, tambahkan method PangkatBF()



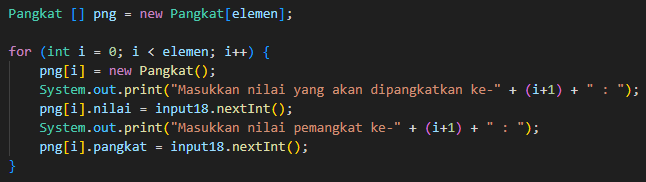
1. Pada class Pangkat juga tambahkan method PangkatDC()



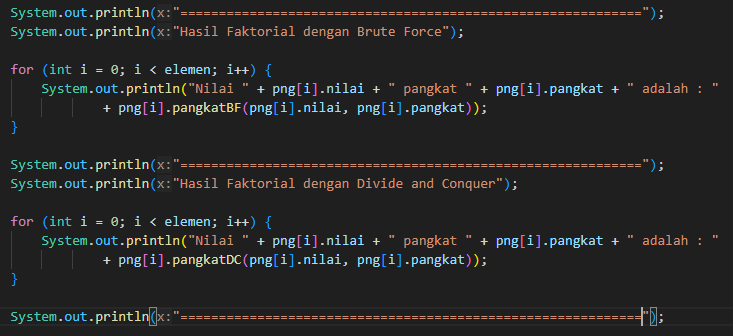
1. Perhatikan apakah sudah tidak ada kesalahan yang muncul dalam pembuatan class Pangkat
2. Selanjutnya buat class baru yang di dalamnya terdapat method main. Class tersebut dapat dinamakan MainPangkat. Tambahkan kode pada class main untuk menginputkan jumlah nilai yang akan dihitung pangkatnya.



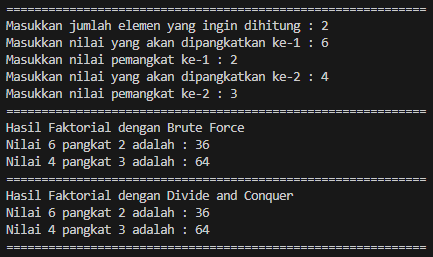
1. Nilai pada tahap 5 selanjutnya digunakan untuk instansiasi array of objek. Di dalam Kode berikut ditambahkan proses pengisian beberapa nilai yang akan dipangkatkan sekaligus dengan pemangkatnya.



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



1. Verifikasi Hasil Percobaan



Pertanyaan Percobaan 2

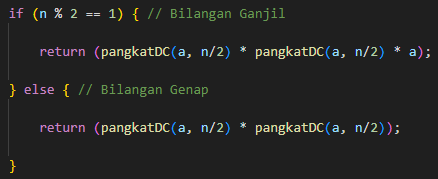
1. Jelaskan mengenai perbedaan 2 method yang dibuat yaitu PangkatBF() dan PangkatDC()!

* PangkatBF()

1. Menggunakan iterasi sederhana dengan looping for untuk mengalikan bilangan a sebanyak n kali
2. Algoritma ini efektif untuk nilai n yang relatif kecil, tetapi memiliki kompleksitas waktu O(n) karena membutuhkan waktu linear sesuai dengan nilai eksponen n

* PangkatDC()

1. Membagi masalah menjadi submasalah yang lebih kecil dan menyelesaikannya secara rekursif.
2. Algoritma ini memanfaatkan sifat matematis bahwa a^n dapat dipecah menjadi a^(n/2) \* a^(n/2) untuk eksponen n genap dan (a^(n/2) \* a^(n/2) \* a) untuk eksponen n ganjil.
3. Algoritma ini memiliki kompleksitas waktu yang lebih baik daripada Brute Force, yaitu O(log n), karena melakukan pembagian masalah secara rekursif menjadi submasalah yang lebih kecil
4. Pada method PangkatDC() terdapat potongan program sebagai berikut: Jelaskan arti potongan kode tersebut



* if (n % 2 == 1) { // Bilangan Ganjil

Pernyataan ini memeriksa apakah eksponen n adalah bilangan ganjil. Jika sisa pembagian n dengan 2 sama dengan 1, itu berarti n adalah bilangan ganjil.

* return (pangkatDC(a, n/2) \* pangkatDC(a, n/2) \* a)

Jika eksponen n adalah bilangan ganjil, maka masalah dibagi menjadi dua bagian: a^(n/2) dan a^(n/2). Kedua submasalah ini diselesaikan secara rekursif menggunakan metode pangkatDC(). Kemudian, hasilnya dikalikan dengan a, karena eksponen n ganjil menambahkan satu faktor a tambahan.

* else { // Bilangan Genap

Ini adalah bagian dari struktur if-else yang menangani kasus ketika eksponen n adalah bilangan genap

* return (pangkatDC(a, n/2) \* pangkatDC(a, n/2));

Jika eksponen n adalah bilangan genap, maka masalah dibagi menjadi dua bagian yang sama: a^(n/2) dan a^(n/2). Kedua submasalah ini diselesaikan secara rekursif menggunakan metode pangkatDC(), dan hasilnya dikalikan bersama.

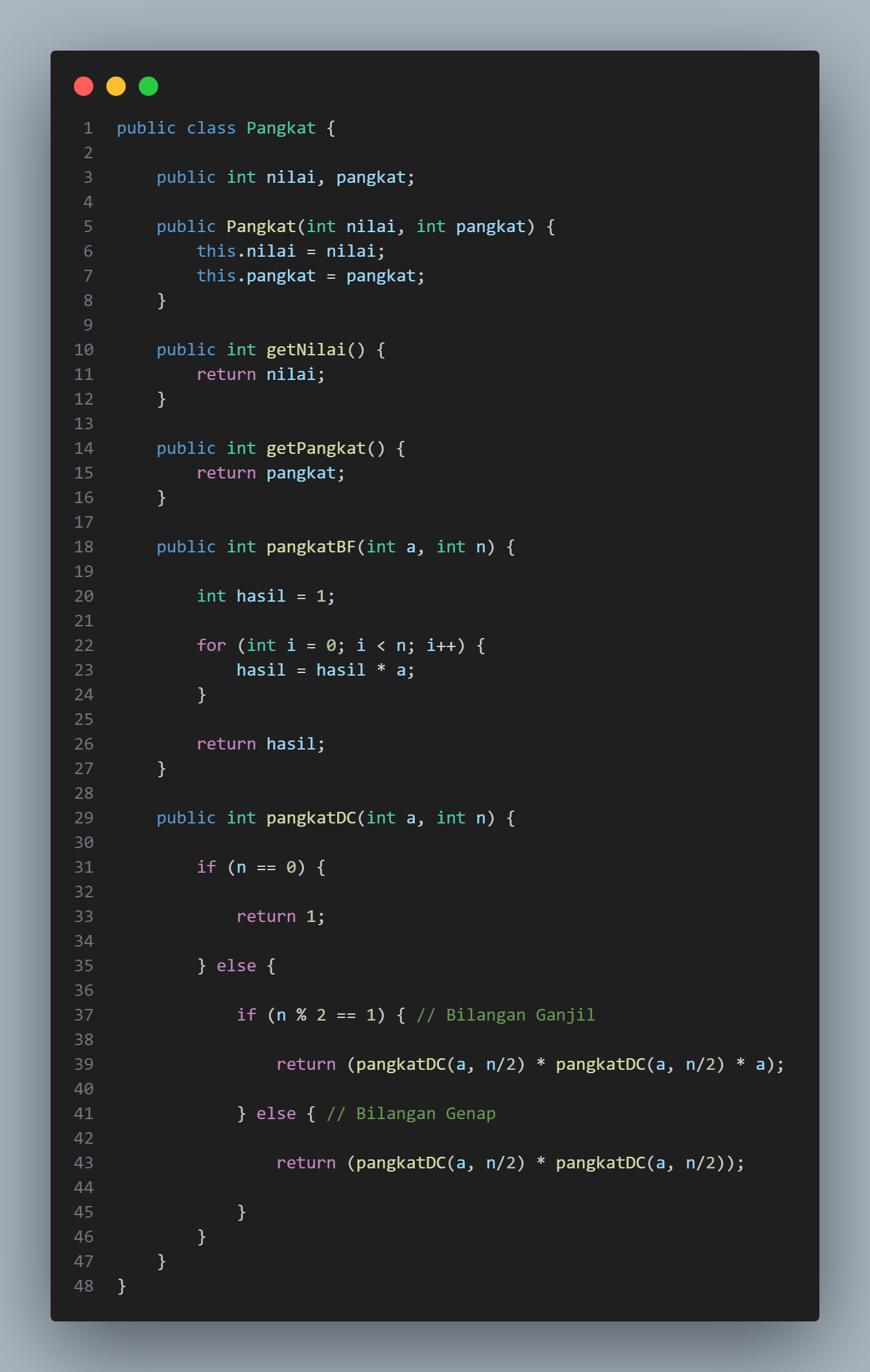
Dengan cara ini, pendekatan Divide and Conquer secara rekursif memecah masalah menjadi submasalah yang lebih kecil dan menyelesaikannya secara efisien.

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



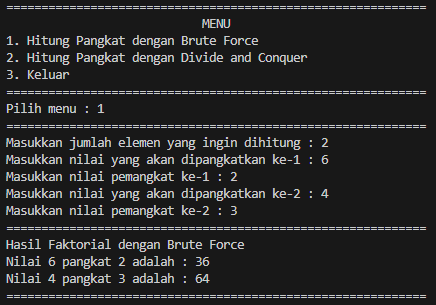


1. Modifikasi kode program tersebut, anggap proses pengisian atribut dilakukan dengan konstruktor.





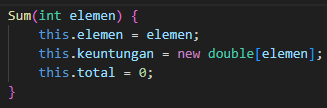
1. Tambahkan menu agar salah satu method yang terpilih saja yang akan dijalankan!



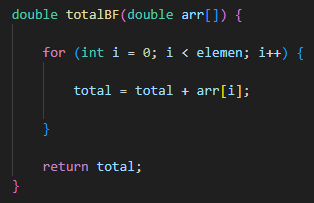


1. **PERCOBAAN 3**
2. Pada paket minggu5. Buat class baru yaitu class Sum. DI salam class tersebut terdapat beberapa atribut jumlah elemen array, array, dan juga total. Tambahkan pula konstruktor pada class Sum.

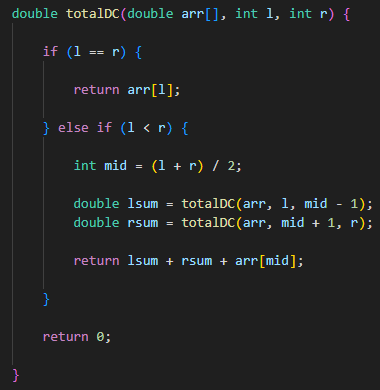




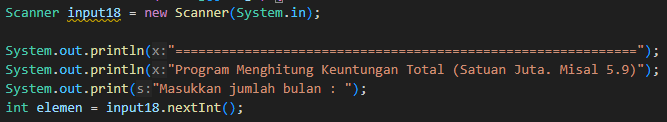
1. Tambahkan method TotalBF() yang akan menghitung total nilai array dengan cara iterative.



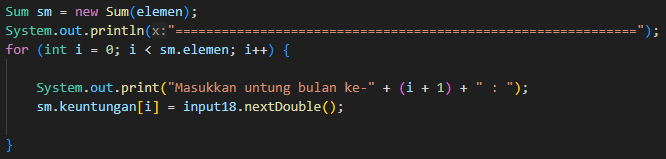
1. Tambahkan pula method TotalDC() untuk implementasi perhitungan nilai total array menggunakan algoritma Divide and Conquer.



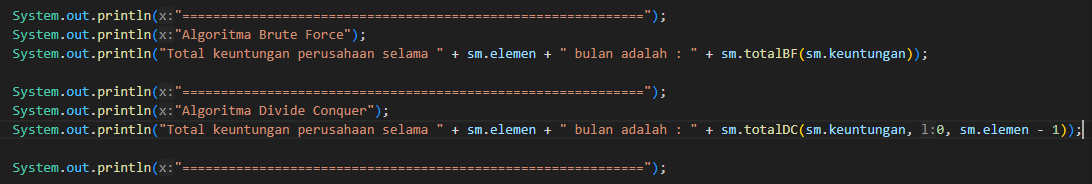
1. Buat class baru yaitu MainSum. Di dalam kelas ini terdapat method main. Pada method ini user dapat menuliskan berapa bulan keuntungan yang akan dihitung. Dalam kelas ini sekaligus dibuat instansiasi objek untuk memanggil atribut ataupun fungsi pada class Sum.



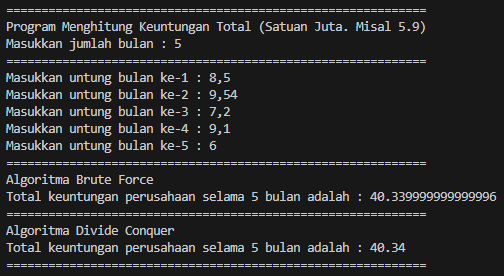
1. Karena yang akan dihitung adalah total nilai keuntungan, maka ditambahkan pula pada method main mana array yang akan dihitung. Array tersebut merupakan atribut yang terdapat di class Sum, maka dari itu dibutuhkan pembuatan objek Sum terlebih dahulu.



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



1. Verifikasi hasil percobaan



Pertanyaan Percobaan 3

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

* totalBF()

1. Pada metode ini akan menelusuri seluruh elemen dalam array keuntungan
2. Setiap elemen ditambahkan ke total secara berurutan dengan kompleksitas waktu O(n), di mana n adalah jumlah elemen dalam array

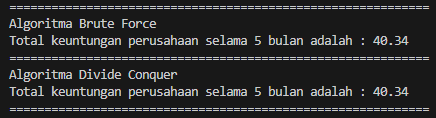
* totalDC()

1. Metode ini membagi masalah menjadi submasalah yang lebih kecil, menyelesaikan submasalah tersebut secara rekursif, dan menggabungkan hasilnya
2. Ketika mencapai kasus dasar (basis), yaitu ketika kisaran (range) dari l dan r hanya satu elemen, maka nilai dari elemen tersebut dikembalikan
3. Jika kisaran lebih dari satu elemen, kisaran dibagi menjadi dua bagian pada tengahnya (mid), dan dua submasalah dipecahkan secara rekursif untuk setiap bagian kiri dan kanan
4. Kemudian, total dari kiri, kanan, dan elemen tengah dihitung dan digabungkan untuk memberikan total keseluruhan
5. Proses ini memiliki kompleksitas waktu O(n log n) karena membagi masalah menjadi dua setengah pada setiap tingkat rekursi.

Jadi, TotalBF() sederhana dan langsung menambahkan keuntungan secara berurutan, sementara TotalDC() membagi masalah menjadi submasalah lebih kecil, menyelesaikannya secara rekursif, dan kemudian menggabungkan hasilnya untuk mendapatkan total keseluruhan

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.

Dibulatkan menjadi 2 angka setelah koma, maka menghasilkan output seperti ini







1. Mengapa terdapat formulasi return value berikut?Jelaskan!



* lsum

Ini adalah hasil dari pemanggilan rekursif totalDC() pada setengah bagian kiri array (dari indeks l hingga mid - 1). Metode ini mengembalikan total keuntungan dari setengah bagian kiri

* rsum

Ini adalah hasil dari pemanggilan rekursif totalDC() pada setengah bagian kanan array (dari indeks mid + 1 hingga r). Metode ini mengembalikan total keuntungan dari setengah bagian kanan

* arr[mid]

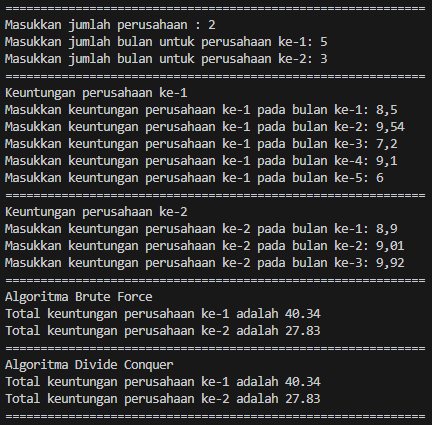
Ini adalah nilai keuntungan pada indeks mid, yang merupakan nilai keuntungan tengah dari array.

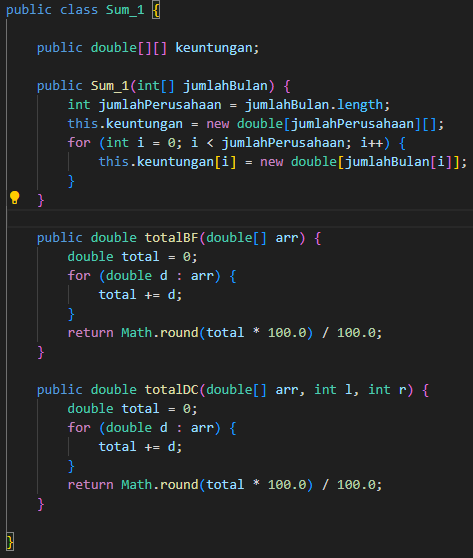
Jadi, return lsum + rsum + arr[mid]; menggabungkan total keuntungan dari setengah bagian kiri, setengah bagian kanan, dan nilai keuntungan tengah, yang merupakan total keseluruhan keuntungan dari seluruh array. Ini adalah langkah combine dalam algoritma Divide and Conquer, di mana hasil dari dua submasalah yang lebih kecil digabungkan bersama-sama untuk membentuk solusi untuk masalah yang lebih besar.

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

Variabel mid dibutuhkan dalam metode totalDC() karena itu merupakan indeks yang menandai titik tengah dari array yang sedang diproses. Dalam pendekatan Divide and Conquer, array dibagi menjadi dua bagian setiap kali metode rekursif dipanggil. Variabel mid digunakan untuk menandai pembagian tersebut

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. **TUGAS**
2. 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.

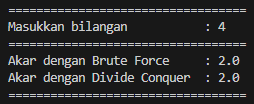
* Program HitungAkar.java



* Program MainHitungAkar.java



* Output



**Link Repository : <https://github.com/zidnafaz/Praktikum-Algoritma-Struktur-Data>**