Nama : ANGGA SAPUTRA RAMADHAN

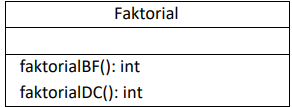
Kelas : TI\_1B

NIM : 244107020065

5.1 Tujuan Praktikum Setelah melakukan materi praktikum ini, mahasiswa mampu:

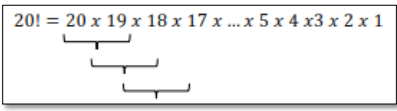
1. Mahasiswa mampu membuat algoritma bruteforce dan divide-conquer
2. Mahasiswa mampu menerapkan penggunaan algorima bruteforce dan divide-conquer

5.2 Menghitung Nilai Faktorial dengan Algoritma Brute Force dan Divide and Conquer Perhatikan Diagram Class berikut ini:

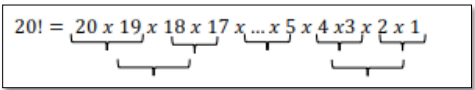


Berdasarkan diagram class di atas, akan dibuat program class dalam Java. Untuk menghitung nilai faktorial suatu angka menggunakan 2 jenis algoritma, Brute Force dan Divide and Conquer. Jika digambarkan terdapat perbedaan proses perhitungan 2 jenis algoritma tersebut sebagai berikut :

Tahapan pencarian nilai faktorial dengan algoritma Brute Force:

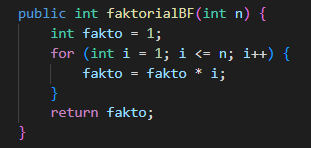


Tahapan pencarian nilai faktorial dengan algoritma Divide and Conquer:

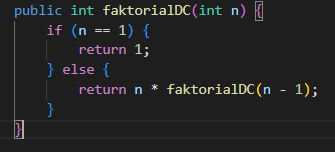


5.2.1. Langkah-langkah Percobaan

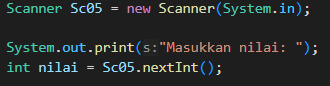
1. Buat folder baru bernama Jobsheet5 di dalam repository Praktikum ASD
2. Buatlah class baru dengan nama Faktorial
3. Lengkapi class Faktorial dengan atribut dan method yang telah digambarkan di dalam diagram class di atas, sebagai berikut:
4. Tambahkan method faktorialBF():



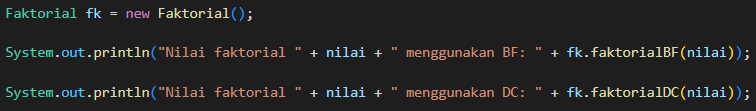
1. Tambahkan method faktorialDC():



1. Coba jalankan (Run) class Faktorial dengan membuat class baru MainFaktorial.
2. Didalam fungsi main sediakan komunikasi dengan user untuk memasukan nilai yang akan dicari faktorialnya



1. Kemudian buat objek dari class Faktorial dan tampilkan hasil pemanggilan method faktorialDC() dan faktorialBF()



1. Pastikan program sudah berjalan dengan baik!

5.2.2. Verifikasi Hasil Percobaan

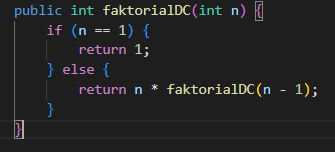
Cocokkan hasil compile kode program anda dengan gambar berikut ini.



5.2.3. Pertanyaan

* 1. Pada base line Algoritma Divide Conquer untuk melakukan pencarian nilai faktorial, jelaskan perbedaan bagian kode pada penggunaan if dan else!

Jwb.

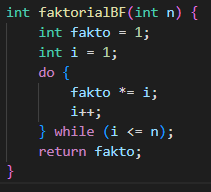


if saat n == 1fungsi akan langsung mengembalikan 1sehingga rekursif tidak berjalan terus menerus tanpa akhir sedangkan else adalah fungsi untuk memanggil dirinya sendiri dengan nilai n-1 hingga mencapai base case ( n == 1 )

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

Jwb.

Ya, perulangan pada metode faktorialBF() bisa diubah menggunakan while atau do-while loop



* 1. Jelaskan perbedaan antara fakto \*= i; dan int fakto = n \* faktorialDC(n-1); !

Jwb.

fakto \*= i; digunakan dalam fungsi iterative dan memperbarui nilai fakto setiap iterasi dengan mengalihkan nilai sebelumnya dengan i misalnya n=5 : 1 \* 1 – 1 \* 2 – 2 \* 3 – 6 \* 4 – 24 \* 5 = 120 sedangkan int fakto = n \* faktorialDC(n-1); digunakan dalam metode rekursif dan memanggil dirinya sendiri misalnya jika n = 5 : 5 \* faktorialDC(4) – 4 \* faktorialDC(3) – 3 \* faktorialDC(2) – 2 \* faktorialDC(1) - 1

* 1. Buat Kesimpulan tentang perbedaan cara kerja method faktorialBF() dan faktorialDC()!

Jwb.

faktorialBF() lebih cepat dan hemat memori sedangkan faktorialDC() lebih elegan dan menggunakan konsep divide & conquer tetapi bisa menyebabkan stack overflow untuk nilai besar

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

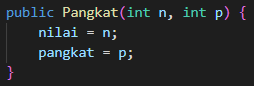
Pada praktikum ini kita akan membuat program class dalam Java, untuk menghitung nilai pangkat suatu angka menggunakan 2 jenis algoritma, Brute Force dan Divide and Conquer. Pada praktikum ini akan digunakan Array of Object untuk mengelola beberapa objek yang akan dibuat, berbeda dengan praktikum tahap sebelumnya yang hanya berfokus pada 1 objek factorial saja.

5.3.1. Langkah-langkah Percobaan

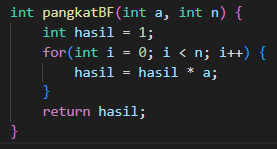
1. Buatlah class baru dengan nama Pangkat, dan di dalam class Pangkat tersebut, buat atribut angka yang akan dipangkatkan sekaligus dengan angka pemangkatnya



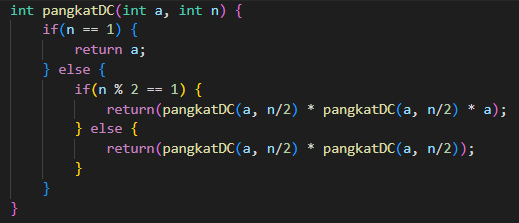
1. Tambahkan konstruktor berparameter



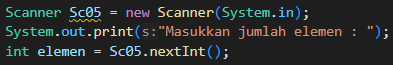
1. Pada class Pangkat tersebut, tambahkan method PangkatBF()



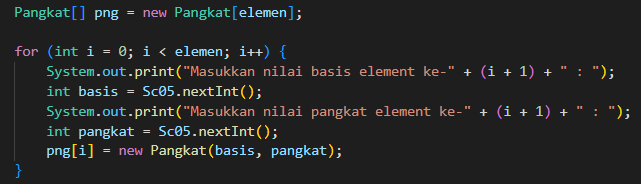
1. Pada class Pangkat juga tambahkan method PangkatDC()



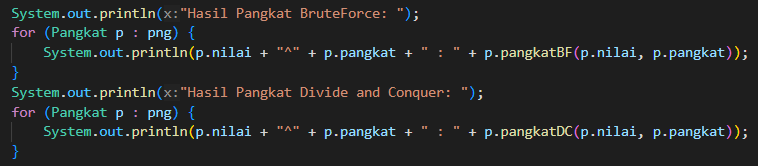
1. Perhatikan apakah sudah tidak ada kesalahan yang muncul dalam pembuatan class Pangkat
2. Selanjutnya buat class baru yang di dalamnya terdapat method main. Class tersebut dapat dinamakan MainPangkat. Tambahkan kode pada class main untuk menginputkan jumlah elemen yang akan dihitung pangkatnya.



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

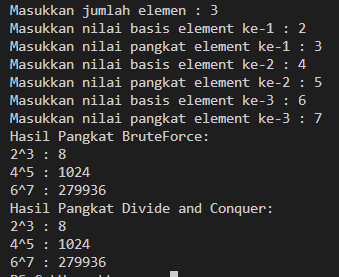


1. Kemudian, panggil hasil nya dengan mengeluarkan return value dari method PangkatBF() dan PangkatDC().



5.3.2. Verifikasi Hasil Percobaan

Pastikan output yang ditampilkan sudah benar seperti di bawah ini.



5.3.3. Pertanyaan

1. Jelaskan mengenai perbedaan 2 method yang dibuat yaitu pangkatBF() dan pangkatDC()!

Jwb.

Pada method pangkatBF() menggunakan perulangan for untuk mengalika angka secara berulang sebanyak nilai pangkat.sedangkan pangkatDC() melakukan perhitungan secara rekursif dengan membagi pangkat menjadi setengah untuk mengurangi jumlah perhitungan.

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

Jwb.

Ya, terdapat pada method pangkatDC() Ketika rekursi dijalankan nilai pangkat dibagi menjadi 2(n/2), sehingga terdapat dua hasil perhitungan.

1. Pada method pangkatBF()terdapat parameter untuk melewatkan nilai yang akan dipangkatkan dan pangkat berapa, padahal di sisi lain di class Pangkat telah ada atribut nilai dan pangkat, apakah menurut Anda method tersebut tetap relevan untuk memiliki parameter? Apakah bisa jika method tersebut dibuat dengan tanpa parameter? Jika bisa, seperti apa method pangkatBF() yang tanpa parameter?

Jwb.

Bisa dibuat tanpaparameter karena bisa langsung menggunakan atribut nilai dan pangkat

1. Tarik tentang cara kerja method pangkatBF() dan pangkatDC()!

Jwb.

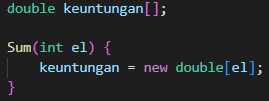
Metode pangkatBF() bekerja secara iteratif dengan kompleksitas O(n) yang kurang optimal untuk pangkat besar, sedangkan pangkatDC() menggunakan pendekatan rekursif dengan kompleksitas O(log n) yang lebih efisien karena membagi masalah menjadi bagian-bagian lebih kecil.

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

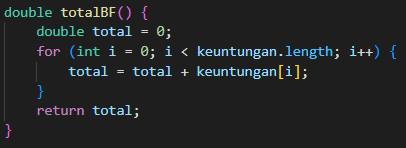
Di dalam percobaan ini, kita akan mempraktekkan bagaimana proses divide, conquer, dan combine diterapkan pada studi kasus penjumlahan keuntungan suatu perusahaan dalam beberapa bulan.

5.4.1. Langkah-langkah Percobaan

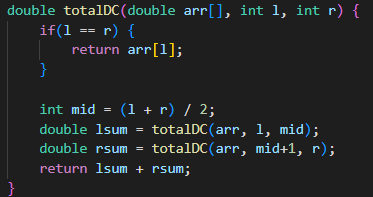
1. Buat class baru yaitu class Sum. Tambahkan pula konstruktor pada class Sum.



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



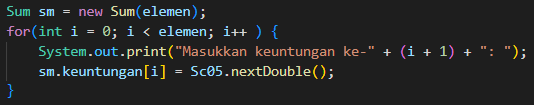
1. Tambahkan pula method TotalDC() untuk implementasi perhitungan nilai total array menggunakan algoritma Divide and Conquer



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. Buat objek dari class Sum. Lakukan perulangan untuk mengambil input nilai keuntungan dan masukkan ke atribut keuntungan dari objek yang baru dibuat tersebut!

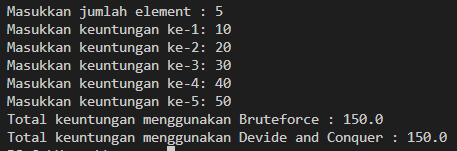


1. Tampilkan hasil perhitungan melalui objek yang telah dibuat untuk kedua cara yang ada (Brute Force dan Divide and Conquer)



5.4.2. Verifikasi Hasil Percobaan

Cocokkan hasil compile kode program anda dengan gambar berikut ini.



5.4.3. Pertanyaan

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

Jwb.

Untuk membagi masalah menjadi dua bagian agar bisa diselesaikan dengan cepat

1. Untuk apakah statement di bawah ini dilakukan dalam TotalDC()?



Jwb.

Membagi menjadi dua tempat lsum untuk menyelesaikan permasalahan dari l sampai mid dan untuk rsum menyelesaikan masalah dari mid+1 sampai r

1. Kenapa diperlukan penjumlahan hasil lsum dan rsum seperti di bawah ini?



Jwb.

Untuk menggabungkan / combine hasil dari pemecahan masalah.

1. Apakah base case dari totalDC()?

Jwb.



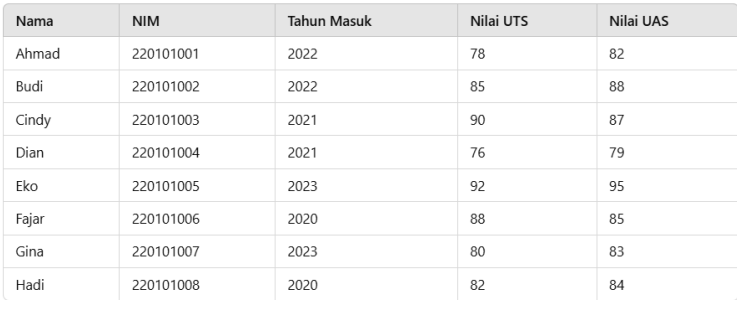
1. Tarik Kesimpulan tentang cara kerja totalDC()?

Jwb.

Fungsi totalDC() menggunakan metode Divide and Conquer untuk membagi array menjadi dua bagian hingga mencapai elemen tunggal, lalu menggabungkan hasilnya secara rekursif, sehingga menghasilkan perhitungan yang lebih efisien dengan kompleksitas O(n log n) dibandingkan metode iteratif

* 1. Latihan Praktikum

1. Sebuah kampus memiliki daftar nilai mahasiswa dengan data sesuai tabel di bawah ini



Tentukan:

1. Nilau UTS tertinggi tertinggi menggunakan Divide and Conquer!
2. Nilai UTS terendah menggunakan Divide and Conquer!
3. Rata-rata nilai UAS dari semua mahasiswa menggunakan Brute Force!

Jwb.

