Laporan Hasil Praktikum

JOBHSEET 5



Angel Chelssa Leoniy Eka Permatasari

244107020202

1E

Program Studi Teknologi Informasi

Jurusan Teknik Informatika

POLINEMA

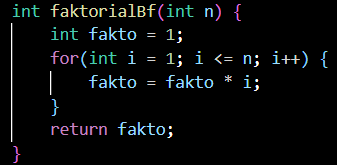
2025

**Percobaan 1**

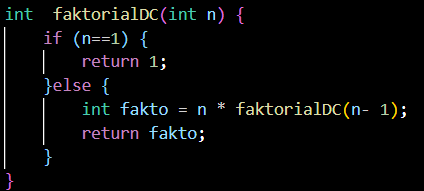
1. Buat class dengan nama faktorial03



1. Lengkapi class dengan atribut dan method yang telah digambarkan digambarkan didalam diagram class
   1. Tambahkan method faktorialBF();



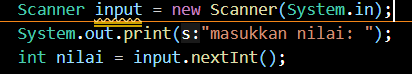
* 1. Tambahkan method faktorialDC();



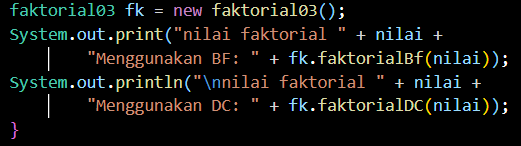
1. Jalankan run class factorial dengan membuat class baru dengan nama mainFaktorial03()



* 1. Dalam fungsi main sediakan komunikasi dengan user untuk memasukkan nilai yang akan dicari faktorialnya



* 1. Buat objek dari class faktorial03 dan tampilkan hasil pemanggilan method faktorialDC() dan faktorialBF();

****

**Kode program:**

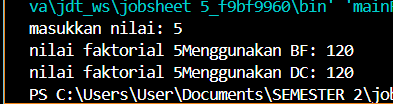
**Class faktorial03**

|  |
| --- |
| public class faktorial03 {      int faktorialBf(int n) {          int faktor = 1;          for(int i = 1; i <= n; i++) {              faktor = faktor \* i;          }          return faktor;      }      Int faktorialDC(int n) {          if (n==1) {              return 1;          }else {              int faktor = n \* faktorialDC(n- 1);              return faktor;          }      }  } |

**Class mainFaktorial03**

|  |
| --- |
| import java.util.Scanner;  public class mainFaktorial03 {      public static void main(String[] args) {      Scanner input = new Scanner(System.in);      System.out.print("masukkan nilai: ");      int nilai = input.nextInt();        faktorial03 fk = new faktorial03();      System.out.print("nilai faktorial " + nilai +              "Menggunakan BF: " + fk.faktorialBf(nilai));      System.out.println("\nnilai faktorial " + nilai +              "Menggunakan DC: " + fk.faktorialDC(nilai));      }  } |

**Hasil kode**



**Pertanyaan**

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

* If (base case) menghentikan rekursi saat n==0 atau 1, langsung return 1
* Else (recursive case) memanggil fungsi Kembali(n \* fatorialDC(n-1))

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

* Bisa menggunakan while atau do while
* Bukti
  + While

|  |
| --- |
| public int faktorialBfWhile(int n) {  int faktor = 1;  int i = 1;  while (i <= n) {  faktor \*= i;  i++;  }  return faktor;  } |

* + Do-while

|  |
| --- |
| public int faktorialBfDoWhile(int n) {  int faktor = 1;  int i = 1;  do {  faktor \*= i;  i++;  } while (i <= n);  return faktor;  } |

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

* Fakto \* = i; → memperbarui nilai yang sudah ada ditiap langkap perulangan
* n \* faktorialDC(n-1); → proses perhitungan dilakukan lewat pemanggilan fungsi secara bertahap ke nilai yang lebih kecil

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

* Jawaban

|  |  |
| --- | --- |
| **faktorialBF()** | **faktorialDC()** |
| Menggunakan perulangan | Menggunakan rekursi (memanggil diri sendiri |
| Lebih mudah dipahami dan efisien dalam pernggunaan memori | Sesuai definisi matematis |
| Kurang fleksibel | Bisa stack overflow jika besar |

**Percobaan 2**

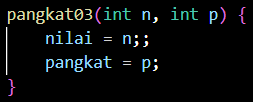
1. Membuat file dengan nama pangkat03;



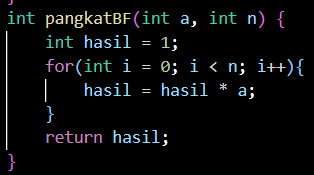
1. Membuat atribut angka yang akan dipangkatkan sekaligus dengan angka pangkatannya



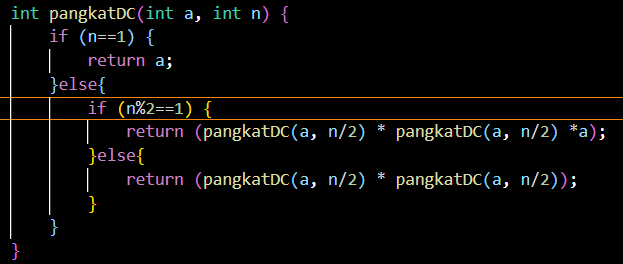
1. Tambahkan konstruktor berparameter



1. Pada class pangkat03 tambahkan method dengan menambahkabn method pangkatBF();



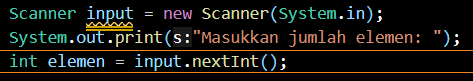
1. Tambahkan method pangkatDC();



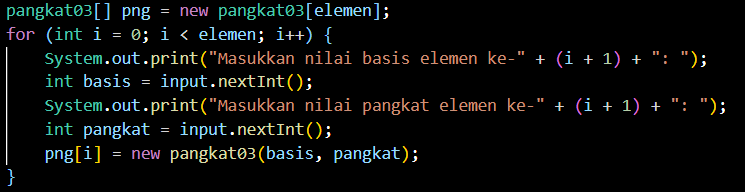
1. Buat class baru dengan nama mainPangkat03()



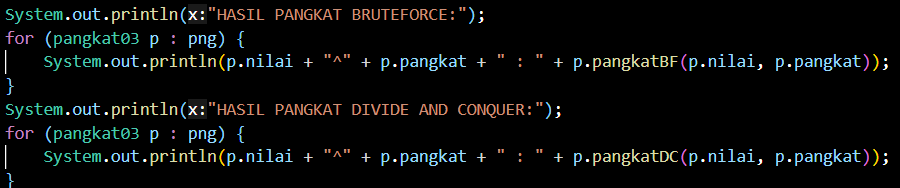
1. Tambahkan kode untuk menginputkan jumlah elemen



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. Panggil hasil dengan mengeluarkan return velue dari method pangkatBF() dan pangkatDC()



**Kode program**

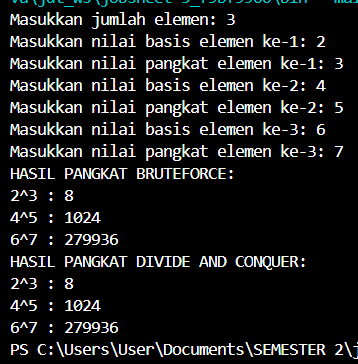
**Kode class pangkat03()**

|  |
| --- |
| public class pangkat03 {      int nilai, pangkat;      public pangkat03(int n, int p) {          nilai = n;;          pangkat = p;      }      int pangkatBF(int a, int n) {          int hasil = 1;          for(int i = 0; i < n; i++){              hasil = hasil \* a;          }          return hasil;      }      int pangkatDC(int a, int n) {          if (n==1) {              return a;          }else{              if (n%2==1) {                  return (pangkatDC(a, n/2) \* pangkatDC(a, n/2) \*a);              }else{                  return (pangkatDC(a, n/2) \* pangkatDC(a, n/2));              }          }      }  } |

**Kode class mainPangkat03()**

|  |
| --- |
| import java.util.Scanner;  public class mainPangkat03 {      public static void main(String[] args) {          Scanner input = new Scanner(System.in);          System.out.print("Masukkan jumlah elemen: ");          int elemen = input.nextInt();          pangkat03[] png = new pangkat03[elemen];          for (int i = 0; i < elemen; i++) {              System.out.print("Masukkan nilai basis elemen ke-" + (i + 1) + ": ");              int basis = input.nextInt();              System.out.print("Masukkan nilai pangkat elemen ke-" + (i + 1) + ": ");              int pangkat = input.nextInt();              png[i] = new pangkat03(basis, pangkat);          }          System.out.println("HASIL PANGKAT BRUTEFORCE:");          for (pangkat03 p : png) {              System.out.println(p.nilai + "^" + p.pangkat + " : " + p.pangkatBF(p.nilai, p.pangkat));          }          System.out.println("HASIL PANGKAT DIVIDE AND CONQUER:");          for (pangkat03 p : png) {              System.out.println(p.nilai + "^" + p.pangkat + " : " + p.pangkatDC(p.nilai, p.pangkat));          }      }  } |

**Hasil kode**



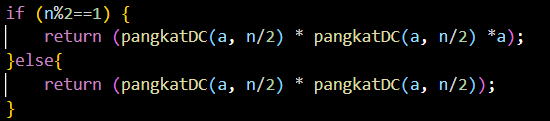
**Pertanyaan**

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

* Pada method pangkatBF() operasi mencari hitung hasil pangkat dilakukan dengan cara buteforce yang dilakukan dengan iteratife/prulangan/looping dan lagoritma buteforcenya adalah mendeklarasikan dahulu hasil = 1 lalu melakukan perulanga dengan batas n(pangkatnya) dan dalam perulangan tersebut dilakukan looping dari hasil tadi dikali dengan a(bilangan yang akan dipangkatkan) dan perulangan akan terus berlanjut hingga < n sehingga a akan menghasilkan nilai hasil dari pemangakatnya
* Pada method pangkatDC() operasi mencari hitung hasil pangkat dilakukan dengan cara divide conquer yang dilakukan dengan rekursif dan algoritma divide conquer yang dilakukan terbagi dalam 3 tahap yaitu : divide, conquer, combine.

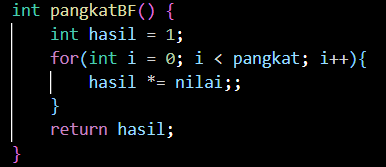
1. Apakah tahap combine sudah termasuk dalam kode tersebut?Tunjukkan!

* combine ada pada method pangkatDC()



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?

* Relevan jika ingin fleksibel, agar method bisa digunakan dengan nilai lain.
* Tidak wajib menggunakan parameter karena sudah ada atribut nilai dan pangkat di dalam class.
* Bisa juga dibuat versi tanpa parameter



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

* pangkatBF();

1. hasil awal = 1
2. ulangi sebanyak pangkat kali
3. disetiap iterasi, hasil dikali dengan nilai
4. setelah perulangan selesai, hasil berisi hasil akhir dari perpangkatan

* pangkatDC();

1. jika pangkat = 0 → hasil = 1 → hasil = nilai
2. untuk pangkat > 1
   1. jika genap



* 1. jika ganjil

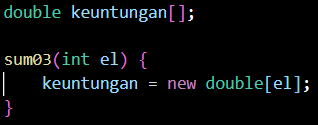


**Percobaan 3**

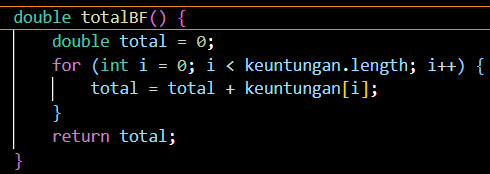
1. buat class baru dengan nama sum03()



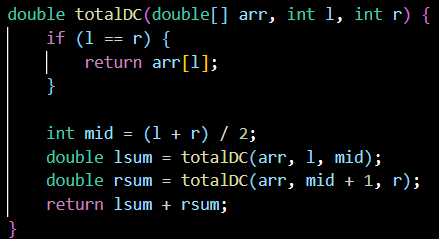
1. tambahkan konstruktor pada class tersebut



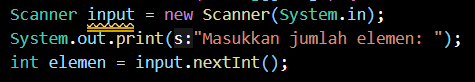
1. tambhakan method totalBF() yang akan menghhitung total nilai array dengan cara iterative



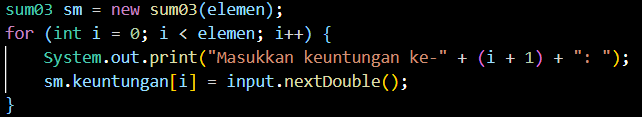
1. Tambahkan pula method TotalDC() untuk implementasi perhitungan nilai total array menggunakan algoritma Divide and Conquer



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 kode**

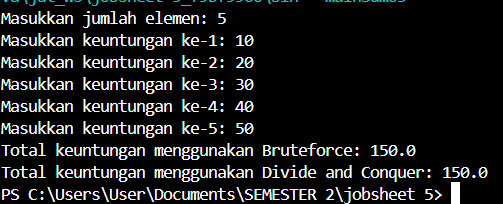
**Kode class pangkat03();**

|  |
| --- |
| public class sum03 {      double keuntungan[];      sum03(int el) {          keuntungan = new double[el];      }      double totalBF() {          double total = 0;          for (int i = 0; i < keuntungan.length; i++) {              total = total + keuntungan[i];          }          return total;      }      double totalDC(double[] arr, int l, int r) {          if (l == r) {              return arr[l];          }            int mid = (l + r) / 2;          double lsum = totalDC(arr, l, mid);          double rsum = totalDC(arr, mid + 1, r);          return lsum + rsum;      }  } |

**Kode class mainpangkat03();**

|  |
| --- |
| import java.util.Scanner;  public class mainSum03 {      public static void main(String[] args) {          Scanner input = new Scanner(System.in);          System.out.print("Masukkan jumlah elemen: ");          int elemen = input.nextInt();          sum03 sm = new sum03(elemen);          for (int i = 0; i < elemen; i++) {              System.out.print("Masukkan keuntungan ke-" + (i + 1) + ": ");              sm.keuntungan[i] = input.nextDouble();          }          System.out.println("Total keuntungan menggunakan Bruteforce: " + sm.totalBF());          System.out.println("Total keuntungan menggunakan Divide and Conquer: " +sm.totalDC(sm.keuntungan, 0, elemen - 1));      }  } |

**Hasil kode program**

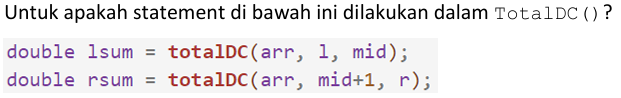


**Pertanyaan**

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

* Karena mid digunakan untuk mambgai array menjadi dua bagian, yaitu bagian kiri (l sampai mid) dan bagian kanan (mid+1 sampai r)

1. Untuk apakah stetment dibawah ini dilakukan dalam totalDC();



* Kedua statement tersebut digunakan untuk menghitung total elemen bagian kiri (lsum) dan kanan (rsum) array secara rekursif.

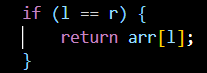
1. Kenapa diperlukan penjumlahan hasil lsum dan rsum seperti di bawah ini?



* Karena total keseluruhan adalah penjumlahan dari bagian kiri dan kanan array. menggabungkan solusi dari bagian-bagian kecil.

1. Apakah base case dari totalDC()?

* jika **hanya satu elemen yang tersisa**, maka langsung dikembalikan nilainya tanpa dibagi lagi.



1. Tarik Kesimpulan tentang cara kerja totalDC()

* membagi array menjadi dua bagian (Divide), menghitung total masing-masing bagian secara rekursif (Conquer), lalu menjumlahkan hasilnya (Combine).