Nama : Deanissa Sherly Sabilla
Kelas / Absen : SIB 1B / 06

JOBSHEET IV
BRUTE FORCE DAN DIVIDE CONQUER

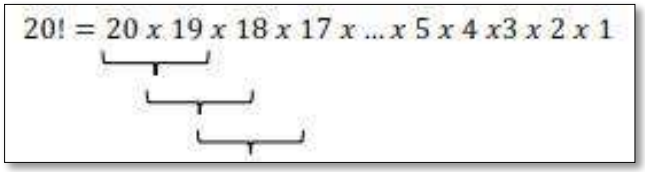
4.2 Menghitung Nilai Faktorial dengan Algoritma Brute Force dan Divide and Conquer

Perhatikan Diagram Class berikut ini :

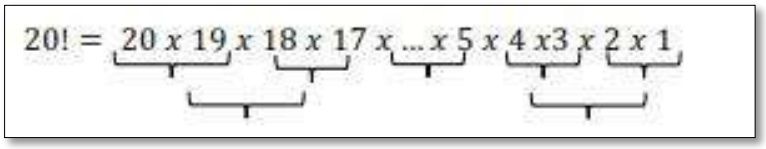
Faktorial
nilai: int
faktorialBF(): int faktorialDC(): int

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, dengan nama "**`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

```
public int nilai;
```

b) Tambahkan method `faktorialBF()` nilai

```
public int faktorialBF(int n){
    int fakto = 1;
    for (int i = 1; i <= n; i++) {
        fakto = fakto * i;
    }
    return fakto;
}
```

c) Tambahkan method `faktorialDC()` nilai

```
public int faktorialDC(int n){
    if (n==1) {
        return  1;
    }
    else
    {
        int fakto = n * faktorialDC(n-1);
        return fakto;
    }
}
```

4. Coba jalankan (Run) class `Faktorial` dengan membuat class baru `MainFaktorial`.

a) Di dalam fungsi main sediakan komunikasi dengan user untuk menginputkan jumlah angka yang akan dicari nilai faktorialnya

```
Scanner sc = new Scanner(System.in);
System.out.println("=======================================================");
System.out.print("Masukkan jumlah elemen yang ingin dihitung : ");
int elemen = sc.nextInt();
```

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

```
Faktorial [] fk = new Faktorial[elemen];
for (int i = 0; i < elemen; i++) {
    fk[i] = new Faktorial();
    System.out.print("Masukkan nilai data ke-"+(i+1)+" : ");
    fk[i].nilai = sc.nextInt();
}
```

c) Tampilkan hasil pemanggilan method `faktorialDC()` dan `faktorialBF()`

```
System.out.println("=========================================================");
System.out.println("Hasil Faktorial dengan Brute Force");
for (int i = 0; i < elemen; i++) {
    System.out.println("Faktorial dari nilai "+fk[i].nilai+" adalah : "+fk[i].faktorialBF(fk[i].nilai));
}
System.out.println("=========================================================");
System.out.println("Hasil Faktorial dengan Divide and Conquer");
for (int i = 0; i < elemen; i++) {
    System.out.println("Faktorial dari nilai "+fk[i].nilai+" adalah : "+fk[i].faktorialDC(fk[i].nilai));
}
System.out.println("=========================================================");
```

d) Pastikan program sudah berjalan dengan baik!

➢ **HASIL INPUT :**

Class Faktorial :

```
package minggu5;
public class Faktorial {

    public int nilai;

    public int faktorialBF(int n) {
        int fakto = 1;
        for (int i = 1; i <= n; i++) {
            fakto = fakto * i;
        }
        return fakto;
    }

    public int faktorialDC (int n){
        if (n==1){
            return 1;
        } else {
            int fakto = n * faktorialDC(n-1);
            return fakto;
        }
    }
}
```

Class Main :

```
public class MainFaktorial {
    public static void main(String[] args) {
    Scanner sc = new Scanner (System.in);
    System.out.println (x:"=========================================");
    System.out.print(s:"Masukkan jumlah elemen yang ingin di hitung : ");
    int elemen = sc.nextInt();

    Faktorial [] fk = new Faktorial[elemen];
    for (int i = 0; i < elemen; i++) {
        fk[i] = new Faktorial();
        System.out.print("Masukkan nilai data ke-" +(i+1)+ ":");
        fk[i].nilai = sc.nextInt();
    }

    System.out.println(x:"=========================================");
    System.out.println(x:"Hasil Faktorial dengan Brute Force ");
    for (int i = 0; i < elemen; i++) {
        System.out.println("Faktorial dari nilai " +fk[i].nilai+ "adalah : " +fk[i].faktorialBF(fk[i].nilai));
    }
    System.out.println(x:"=========================================");
    System.out.println(x:"Hasil Faktorial dengan Divide and Conquer ");
    for (int i = 0; i < elemen; i++) {
        System.out.println("Faktorial dari nilai " +fk[i].nilai+ "adalah : " +fk[i].faktorialDC(fk[i].nilai));
    }
    System.out.println(x:"=========================================");
```

4.2.2 Verifikasi Hasil Percobaan

Cocokkan hasil compile kode program anda dengan gambar berikut ini.

```
run:
=====================================================
Masukkan jumlah elemen yang ingin dihitung : 3
Masukkan nilai data ke-1 : 5
Masukkan nilai data ke-2 : 8
Masukkan nilai data ke-3 : 3
=====================================================
Hasil Faktorial dengan Brute Force
Faktorial dari nilai 5 adalah : 120
Faktorial dari nilai 8 adalah : 40320
Faktorial dari nilai 3 adalah : 6
=====================================================
Hasil Faktorial dengan Divide and Conquer
Faktorial dari nilai 5 adalah : 120
Faktorial dari nilai 8 adalah : 40320
Faktorial dari nilai 3 adalah : 6
=====================================================
BUILD SUCCESSFUL (total time: 7 seconds)
```

➢ **HASIL OUTPUT :**

```
PS C:\Users\TOSHIBA\BruteForceDivideConquer>  & 'C:\Program Files\Java\jdk-19\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp'
'C:\Users\TOSHIBA\AppData\Roaming\Code\User\workspaceStorage\317bade4c7037f6d977d26f2d83e0aa9\redhat.java\jdt_ws\BruteForceDivideConquer_e63
0798f\bin' 'MainFaktorial'
=========================================
Masukkan jumlah elemen yang ingin di hitung : 3
Masukkan nilai data ke-1:5
Masukkan nilai data ke-2:8
Masukkan nilai data ke-3:3
=========================================
Hasil Faktorial dengan Brute Force
Faktorial dari nilai 5adalah : 120
Faktorial dari nilai 8adalah : 40320
Faktorial dari nilai 3adalah : 6
=========================================
Hasil Faktorial dengan Divide and Conquer
Faktorial dari nilai 5adalah : 120
Faktorial dari nilai 8adalah : 40320
Faktorial dari nilai 3adalah : 6
=========================================
```

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.

2. 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.

3. 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** :

```
5
6       public int faktorialBF(int n) {
7           int fakto = 1;
8           int i = 1;
9           while (i <= n) {
10              fakto = fakto * i;
11              i++;
12          }
13          return fakto;
14      }
```

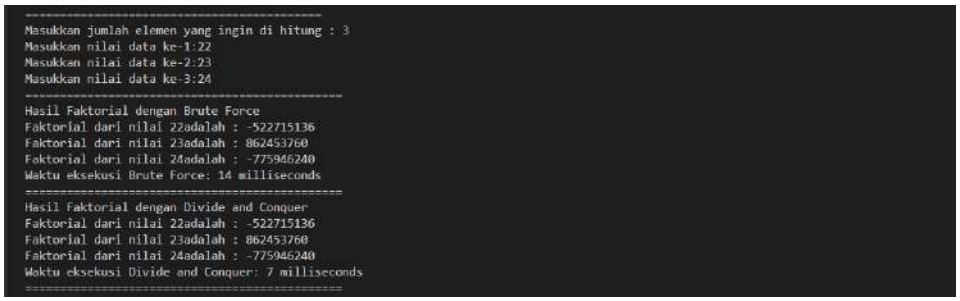
4. Tambahkan pegecekan waktu eksekusi kedua jenis method tersebut!

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

```
System.out.println(x:"=============================================");
System.out.println(x:"Hasil Faktorial dengan Brute Force ");
// Waktu eksekusi untuk metode Brute Force
long startTimeBF = System.currentTimeMillis();
for (int i = 0; i < elemen; i++) {
    System.out.println("Faktorial dari nilai " +fk[i].nilai+ "adalah : " +fk[i].faktorialBF(fk[i].nilai));
}
long endTimeBF = System.currentTimeMillis();
System.out.println("Waktu eksekusi Brute Force: " + (endTimeBF - startTimeBF) + " milliseconds");

System.out.println(x:"=============================================");
System.out.println(x:"Hasil Faktorial dengan Divide and Conquer ");
// Waktu eksekusi untuk metode Divide and Conquer
long startTimeDC = System.currentTimeMillis();
for (int i = 0; i < elemen; i++) {
    System.out.println("Faktorial dari nilai " +fk[i].nilai+ "adalah : " +fk[i].faktorialDC(fk[i].nilai));
}
long endTimeDC = System.currentTimeMillis();
System.out.println("Waktu eksekusi Divide and Conquer: " + (endTimeDC - startTimeDC) + " milliseconds");
System.out.println(x:"=============================================");
}
}
```

5. 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

```
    public int nilai,pangkat;
```

2. Pada class Pangkat tersebut, tambahkan method `PangkatBF()`

```
public int pangkatBF(int a,int n){
    int hasil=1;
    for (int i = 0; i < n; i++) {
        hasil = hasil * a;
    }
    return hasil;
}
```

3. Pada class Pangkat juga tambahkan method `PangkatDC()`

```
public int pangkatDC(int a,int n){
    if (n==0) {
        return  1;
    }
    else
    {
        if(n%2==1)//bilangan ganjil
            return (pangkatDC(a,n/2)*pangkatDC(a,n/2)*a);
        else//bilangan genap
            return (pangkatDC(a,n/2)*pangkatDC(a,n/2));
    }
}
```

4. Perhatikan apakah sudah tidak ada kesalahan yang muncul dalam pembuatan class Pangkat
5. 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.

```
Scanner sc = new Scanner(System.in);
System.out.println("==========================================================");
System.out.print("Masukkan jumlah elemen yang ingin dihitung : ");
int elemen = sc.nextInt();
```

6. 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.

```
Pangkat [] png = new Pangkat[elemen];

for (int i = 0; i < elemen; i++) {
    png[i] = new Pangkat();
    System.out.print("Masukkan nilai yang akan dipangkatkan ke-"+(i+1)+" : ");
    png[i].nilai = sc.nextInt();
    System.out.print("Masukkan nilai pemangkat ke-"+(i+1)+" : ");
    png[i].pangkat = sc.nextInt();
}
```

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

```
System.out.println("=============================================");
System.out.println("Hasil Pangkat dengan Brute Force");
for (int i = 0; i < elemen; i++) {
    System.out.println("Nilai "+png[i].nilai+" pangkat "+png[i].pangkat+" adalah : "+png[i].pangkatBF(png[i].nilai,png[i].pangkat));
}
System.out.println("=============================================");
System.out.println("Hasil Pangkat dengan Divide and Conquer");
for (int i = 0; i < elemen; i++) {
    System.out.println("Nilai "+png[i].nilai+" pangkat "+png[i].pangkat+" adalah : "+png[i].pangkatDC(png[i].nilai,png[i].pangkat));
}
System.out.println("=============================================");
```

➢ **HASIL INPUT :**

Class Pangkat :

```
1  package minggu5;
2
3  public class Pangkat {
4      public int nilai, pangkat;
5
6      public int pangkatBF (int a, int n){
7          int hasil = 1;
8          for (int i = 0; i < n; i++) {
9              hasil = hasil * a;
10         }
11         return hasil;
12     }
13     public int pangkatDC (int a, int n) {
14         if (n == 0) {
15             return 1;
16         }
17         else
18         {
19             if (n%2==1) //bil ganjil
20             return (pangkatDC(a, n/2) * pangkatDC (a,n/2)*a);
21             else //bil genap
22             return (pangkatDC(a,n/2) * pangkatDC (a,n/2));
23         }
24     }
25 }
```

Class Main :

```java
import java.util.Scanner;

import minggu5.Pangkat;
public class MainPangkat {
    Run | Debug
    public static void main(String[] args) {
        Scanner sc = new Scanner (System.in);

        System.out.println(x:"======================================");
        System.out.print(s:"Masukkan jumlah elemen yang ingin dihitung : ");
        int elemen = sc.nextInt();

        Pangkat [] png = new Pangkat [elemen];
        for (int i = 0; i < elemen; i++) {
            png[i] = new Pangkat ();
            System.out.print("Masukkan nilai yang akan dipangkatkan ke-" +(i+1)+ " : ");
            png[i].nilai = sc.nextInt();
            System.out.print("Masukkan nilai pemangkat ke-" +(i+1)+ " : ");
            png[i].pangkat = sc.nextInt();
        }
        System.out.println(x:"======================================");
        System.out.print(s:"Hasil Pangkat dengan Brute Force");
        for (int i = 0; i < elemen; i++) {
            System.out.println("Nilai "+png[i].nilai+ "Pangkat"+png[i].pangkat+ "adalah : "+png[i].pangkatBF(png[i].ni
        }

        System.out.println(x:"======================================");
        System.out.print(s:"Hasil Pangkat dengan Divide and Conquer");
        for (int i = 0; i < elemen; i++) {
            System.out.println("Nilai "+png[i].nilai+ "Pangkat"+png[i].pangkat+ "adalah : "+png[i].pangkatDC(png[i].ni
        }
        System.out.println(x:"======================================");
    }
}
```

4.3.2 Verifikasi Hasil Percobaan

Pastikan output yang ditampilkan sudah benar seperti di bawah ini.

```
run:
======================================================
Masukkan jumlah elemen yang ingin dihitung : 2
Masukkan nilai yang akan dipangkatkan ke-1 : 6
Masukkan nilai pemangkat ke-1 : 2
Masukkan nilai yang akan dipangkatkan ke-2 : 4
Masukkan nilai pemangkat ke-2 : 3
======================================================
Hasil Pangkat dengan Brute Force
Nilai 6 pangkat 2 adalah : 36
Nilai 4 pangkat 3 adalah : 64
======================================================
Hasil Pangkat dengan Divide and Conquer
Nilai 6 pangkat 2 adalah : 36
Nilai 4 pangkat 3 adalah : 64
======================================================
BUILD SUCCESSFUL (total time: 10 seconds)
```

➢ **HASIL OUTPUT :**

```
PS C:\Users\TOSHIBA\BruteForceDivideConquer>  & 'C:\Program Files\Java\jdk-19\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages'
  '-cp' 'C:\Users\TOSHIBA\AppData\Roaming\Code\User\workspaceStorage\317bade4c7037f6d977d26f2d83e0aa9\redhat.java\jdt_ws\BruteForceDiv
ideConquer_e630798f\bin' 'MainPangkat'
=====================================
Masukkan jumlah elemen yang ingin dihitung : 2
Masukkan nilai yang akan dipangkatkan ke-1 : 6
Masukkan nilai pemangkat ke-1 : 2
Masukkan nilai yang akan dipangkatkan ke-2 : 4
Masukkan nilai pemangkat ke-2 : 3
=======================================
Hasil Pangkat dengan Brute ForceNilai 6Pangkat2adalah : 36
Nilai 4Pangkat3adalah : 64
=======================================
Hasil Pangkat dengan Divide and ConquerNilai 6Pangkat2adalah : 36
Nilai 4Pangkat3adalah : 64
=======================================
PS C:\Users\TOSHIBA\BruteForceDivideConquer>
```

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.

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

```
if(n%2==1)//bilangan ganjil
    return (pangkatDC(a,n/2)*pangkatDC(a,n/2)*a);
else//bilangan genap
    return (pangkatDC(a,n/2)*pangkatDC(a,n/2));
```

   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.

3. 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.

```
    if (n%2==1) //bil ganjil
    return (pangkatDC(a, n/2) * pangkatDC (a,n/2)*a);
    else //bil genap
    return (pangkatDC(a,n/2) * pangkatDC (a,n/2));
}
```

4. Modifikasi kode program tersebut, anggap proses pengisian atribut dilakukan dengan konstruktor.
   - Dengan menambah kontruktor pada **class Pangkat**

```
4       public int nilai, pangkat;
5
6       // Konstruktor untuk menginisialisasi nilai dan pangkat
7       public Pangkat (int nilai, int pangkat){
8           this.nilai = nilai;
9           this.pangkat = pangkat;
10      }
11
```

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

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

Class Main :

```
public class MainPangkat {
    Run | Debug
    public static void main(String[] args) {
        Scanner sc = new Scanner (System.in);

        System.out.println("=======================================");
        System.out.print("Masukkan jumlah elemen yang ingin dihitung : ");
        int elemen = sc.nextInt();

        Pangkat [] png = new Pangkat [elemen];
        for (int i = 0; i < elemen; i++) {
            //png[i] = new Pangkat ();
            System.out.print("Masukkan nilai yang akan dipangkatkan ke-" +(i+1)+ " : ");
            int nilai = sc.nextInt();
            System.out.print("Masukkan nilai pemangkat ke-" +(i+1)+ " : ");
            int pangkat = sc.nextInt();
            png[i] = new Pangkat(nilai, pangkat);
        }

        System.out.println("=======================================");
        System.out.println("Pilih metode:");
        System.out.println("1. Pangkat dengan Brute Force");
        System.out.println("2. Pangkat dengan Divide and Conquer");
        System.out.print("Masukkan pilihan: ");
        int pilihan = sc.nextInt();

        switch (pilihan){
        case 1 :
        System.out.println("=======================================");
        System.out.print("Hasil Pangkat dengan Brute Force ");
        for (int i = 0; i < elemen; i++) {
            System.out.println("Nilai "+png[i].nilai+ " Pangkat "+png[i].pangkat+ " adalah : "+png[i].pangkatBF(png[i]
        } break;

        case 2:
        System.out.println("=======================================");
        System.out.print("Hasil Pangkat dengan Divide and Conquer ");
        for (int i = 0; i < elemen; i++) {
            System.out.println("Nilai "+png[i].nilai+ " Pangkat "+png[i].pangkat+ " adalah : "+png[i].pangkatDC(png[i]
        } break;
        default:
                System.out.println("Pilihan tidak valid");
        }
        System.out.println("=======================================");
    }
}
```

- **Hasil Output :**

```
PS C:\Users\TOSHIBA\BruteForceDivideConquer>  c:; cd 'c:\Users\TOSHIBA\BruteForceDivideCon
java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\TOSHIBA\AppData\Roamin
6d977d26f2d83e0aa9\redhat.java\jdt_ws\BruteForceDivideConquer_e630798f\bin' 'MainPangkat'
=======================================
Masukkan jumlah elemen yang ingin dihitung : 1
Masukkan nilai yang akan dipangkatkan ke-1 : 20
Masukkan nilai pemangkat ke-1 : 2
=======================================
Pilih metode:
1. Pangkat dengan Brute Force
2. Pangkat dengan Divide and Conquer
Masukkan pilihan: 1
=======================================
Hasil Pangkat dengan Brute Force Nilai 20 Pangkat 2 adalah : 400
=======================================
```

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`.

```
public int elemen;
public double keuntungan[];
public double total;
```

```
Sum(int elemen){
    this.elemen = elemen;
    this.keuntungan=new double[elemen];
    this.total = 0;
}
```

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

```
double totalBF(double arr[]){
    for (int i = 0; i < elemen; i++) {
        total = total + arr[i];
    }
    return total;
}
```

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

```
double totalDC(double arr[], int l, int r){
   if(l==r)
       return arr[l];
   else if(l<r){
    int mid=(l+r)/2;
    double lsum=totalDC(arr,l,mid-1);
    double rsum=totalDC(arr,mid+1,r);
    return lsum+rsum+arr[mid];
   }

   return 0;
}
```

4. 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`

```
Scanner sc = new Scanner(System.in);
System.out.println("================================================================");
System.out.println("Program Menghitung Keuntungan Total (Satuan Juta. Misal 5.9)");
System.out.print("Masukkan jumlah bulan : ");
int elm = sc.nextInt();
```

5. 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.

```
Sum sm = new Sum(elm);
System.out.println("================================================================");
for (int i = 0; i < sm.elemen; i++) {
    System.out.print("Masukkan untung bulan ke - "+(i+1)+" = ");
    sm.keuntungan[i] = sc.nextDouble();
}
```

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

```
System.out.println("============================================================");
System.out.println("Algoritma Brute Force");
System.out.println("Total keuntungan perusahaan selama " + sm.elemen + " bulan adalah = "+sm.totalBF(sm.keuntungan));
System.out.println("============================================================");
System.out.println("Algoritma Divide Conquer");
System.out.println("Total keuntungan perusahaan selama " + sm.elemen + " bulan adalah = "+sm.totalDC(sm.keuntungan, 0,sm.elemen-1));
```

➢ **HASIL INPUT :**

Class Sum :

```
1  public class Sum {
2      public int elemen;
3      public double keuntungan[];
4      public double total;
5
6      Sum (int elemen){
7          this.elemen = elemen;
8          this.keuntungan = new double [elemen];
9          this.total = 0;
10     }
11         double totalBF (double arr[]) {
12             for (int i = 0; i < elemen; i++) {
13                 total = total + arr [i];
14             }
15             return total;
16         }
17         double totalDC (double arr[], int l, int r){
18             if (l==r)
19             return arr[l];
20             else if (l<r){
21                 int mid = (l+r)/2;
22                 double lsum = totalDC(arr, l, mid-1);
23                 double rsum = totalDC(arr, mid+1, r);
24                 return lsum+rsum+arr [mid];
25             }
26             return 0;
```

Class Main :

```java
import java.util.Scanner;
public class MainSum {
    Run | Debug
    public static void main(String[] args) {
    Scanner sc = new Scanner (System.in);
    System.out.println(x:"=============================================================");
    System.out.println(x:"Program Menghitung Keuntungan Total (Saruan Juta. Misal 5.9) ");
    System.out.print(s:"Masukkan Jumlah Bulan : ");
    int elm = sc.nextInt();

    Sum sm = new Sum (elm);
    System.out.println(x:"=============================================");
    for (int i = 0; i < sm.elemen; i++) {
        System.out.print("Masukkan untung bulan ke - " +(i+1) +" = ");
        sm.keuntungan[i] = sc.nextDouble();
    }
    System.out.println(x:"=============================================");
    System.out.println(x:"Algoritma Brute Force ");
    System.out.println("Total keuntungan selama " +sm.elemen+ " bulan adalah = " +sm.totalBF (sm.keuntungan));
    System.out.println(x:"=============================================");
    System.out.println(x:"Algoritma Divide Conquer ");
    System.out.println("Total keuntungan selama " +sm.elemen+ " bulan adalah = " +sm.totalDC (sm.keuntungan, 1:0, sm.e

    }
}
```

4.4.2 Verifikasi Hasil Percobaan

Cocokkan hasil compile kode program anda dengan gambar berikut ini.

```
run:
=================================================================
Program Menghitung Keuntungan Total (Satuan Juta. Misal 5.9)
Masukkan jumlah bulan : 5
=================================================================
Masukkan untung bulan ke - 1 = 8.5
Masukkan untung bulan ke - 2 = 9.54
Masukkan untung bulan ke - 3 = 7.2
Masukkan untung bulan ke - 4 = 9.1
Masukkan untung bulan ke - 5 = 6
=================================================================
Algoritma Brute Force
Total keuntungan perusahaan selama 5 bulan adalah = 40.339999999999996
=================================================================
Algoritma Divide Conquer
Total keuntungan perusahaan selama 5 bulan adalah = 40.34
BUILD SUCCESSFUL (total time: 11 seconds)
```

➢ **HASIL OUTPUT :**

```
=================================================
Program Menghitung Keuntungan Total (Saruan Juta. Misal 5.9)
Masukkan Jumlah Bulan : 5
=============================================
Masukkan untung bulan ke - 1 = 8.5
Masukkan untung bulan ke - 2 = 9.54
Masukkan untung bulan ke - 3 = 7.2
Masukkan untung bulan ke - 4 = 9.1
Masukkan untung bulan ke - 5 = 6
=============================================
Algoritma Brute Force
Total keuntungan selama 5 bulan adalah = 40.339999999999996
=============================================
Algoritma Divide Conquer
Total keuntungan selama 5 bulan adalah = 40.34
PS C:\Users\TOSHIBA\BruteForceDivideConquer>
```

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.

2. 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 :

```
==================================================
Algoritma Brute Force
Total keuntungan selama 5 bulan adalah = 40.34
==================================================
Algoritma Divide Conquer
Total keuntungan selama 5 bulan adalah = 40.34
PS C:\Users\TOSHIBA\BruteForceDivideConquer>
```

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

```
return lsum+rsum+arr[mid];
```

   - 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.

4. 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.

5. 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 :**

MainSum.java > MainSum

```
import java.text.DecimalFormat;
import java.util.Scanner;
public class MainSum {
    Run | Debug
    public static void main(String[] args) {
    Scanner sc = new Scanner (System.in);
    //System.out.println("=============================================================");
    //System.out.println("Program Menghitung Keuntungan Total (Satuan Juta. Misal 5.9) ");
    //System.out.print("Masukkan Jumlah Bulan : ");
    System.out.println(x:"=============================================================");
    System.out.println(x:"Program Menghitung Keuntungan Total untuk Beberapa Perusahaan");
    System.out.println(x:"=============================================================");

    // Input jumlah perusahaan
    System.out.print(s:"Masukkan jumlah perusahaan: ");
    int jmlPerusahaan = sc.nextInt();

    // Inisialisasi array 2 dimensi untuk menyimpan keuntungan perusahaan
    double[][] keuntungan = new double[jmlPerusahaan][];

    // Input jumlah bulan dan keuntungan perusahaan
    for (int i = 0; i < jmlPerusahaan; i++) {
        System.out.println(x:"-------------------------------------------------------------");
        System.out.print("Masukkan jumlah bulan untuk perusahaan ke-" + (i + 1) + ": ");
        int jumlahBulan = sc.nextInt();
```

```
    keuntungan[i] = new double[jumlahBulan];
    System.out.println(x:"=============================================");
    for (int j = 0; j < jumlahBulan; j++) {
        System.out.print("Masukkan keuntungan bulan ke-" + (j + 1) + " untuk perusahaan ke-" + (i + 1) + ": ");
        keuntungan[i][j] = sc.nextDouble();
    }
}
    // Hitung total keuntungan untuk setiap perusahaan
    for (int i = 0; i < jmlPerusahaan; i++) {
        Sum sm = new Sum(keuntungan[i].length);
            System.out.println(x:"=============================================");
            System.out.println(x:"Algoritma Brute Force ");
            //Membatasi jumlah angka decimal dibelakang koma
            DecimalFormat df = new DecimalFormat(pattern:"#.##");
            System.out.println("Total keuntungan perusahaan ke- " +(i+1)+ " bulan adalah = " +df.format(sm.totalBF(keu
            System.out.println(x:"=============================================");
            System.out.println(x:"Algoritma Divide Conquer ");
            System.out.println("Total keuntungan perusahaan ke- " +(i+1)+ " bulan adalah = " +df.format(sm.totalDC(keu
            }
    }
}
```

- **Hasil Output :**

```
============================================================
Program Menghitung Keuntungan Total untuk Beberapa Perusahaan
============================================================
Masukkan jumlah perusahaan: 2
------------------------------------------------------------
Masukkan jumlah bulan untuk perusahaan ke-1: 2
==================================================
Masukkan keuntungan bulan ke-1 untuk perusahaan ke-1: 6
Masukkan keuntungan bulan ke-2 untuk perusahaan ke-1: 7.1
------------------------------------------------------------
Masukkan jumlah bulan untuk perusahaan ke-2: 3
==================================================
Masukkan keuntungan bulan ke-1 untuk perusahaan ke-2: 8.5
Masukkan keuntungan bulan ke-2 untuk perusahaan ke-2: 9.1
Masukkan keuntungan bulan ke-3 untuk perusahaan ke-2: 9.54
==================================================
Algoritma Brute Force
Total keuntungan perusahaan ke- 1 bulan adalah = 13.1
==================================================
Algoritma Divide Conquer
Total keuntungan perusahaan ke- 1 bulan adalah = 13.1
==================================================
Algoritma Brute Force
Total keuntungan perusahaan ke- 2 bulan adalah = 27.14
==================================================
Algoritma Divide Conquer
Total keuntungan perusahaan ke- 2 bulan adalah = 27.14
PS C:\Users\TOSHIBA\BruteForceDivideConquer>
```

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 :

```java
public class Akar {

    public int bruteForce(int x) {
        if (x == 0 || x == 1)
            return x;
        int i = 1, result = 1;
        while (result <= x) {
            i++;
            result = i * i;
        }
        return i - 1;
    }
    public int divideConquer(int x) {
        if (x == 0 || x == 1)
            return x;
        int start = 1, end = x, ans = 0;
        while (start <= end) {
            int mid = (start + end) / 2; // Jika x adalah kuadrat dari mid
            if (mid * mid == x)
                return mid;
            if (mid * mid < x) { // Jika kuadrat dari lebih kecil dari x, pencarian ke kanan
                start = mid + 1;
                ans = mid;
            } else { // Jika kuadrat dari lebih besar dari x, pencarian ke kiri
                end = mid - 1;
            }
        }
        return ans;
    }
}
```

Class Main :

```java
import java.util.Scanner;
public class AkarMain {
    Run | Debug
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print(s:"Masukkan jumlah bilangan : ");
        int jmlBilangan = sc.nextInt();

        Akar akar = new Akar();

        for (int i = 1; i <= jmlBilangan; i++) {
            System.out.print("Masukkan bilangan ke-" + i + ": ");
            int bilangan = sc.nextInt();

            // Menggunakan algoritma Brute Force
            int akarBruteForce = akar.bruteForce(bilangan);
            System.out.println("Akar dari " + bilangan + " (Brute Force) adalah: " + akarBruteForce);

            // Menggunakan algoritma Divide Conquer
            int akarDivideConquer = akar.divideConquer(bilangan);
            System.out.println("Akar dari " + bilangan + " (Divide Conquer) adalah: " + akarDivideConquer);
        }

        sc.close();
    }
```

➢ **HASIL OUTPUT :**

```
PS C:\Users\TOSHIBA\BruteForceDivideConquer>  & 'C:\Program Files\Java\jdk-19\b
 '-cp' 'C:\Users\TOSHIBA\AppData\Roaming\Code\User\workspaceStorage\317bade4c70
ideConquer_e630798f\bin' 'AkarMain'
Masukkan jumlah bilangan : 2
Masukkan bilangan ke-1: 31
Akar dari 31 (Brute Force) adalah: 5
Akar dari 31 (Divide Conquer) adalah: 5
Masukkan bilangan ke-2: 16
Akar dari 16 (Brute Force) adalah: 4
Akar dari 16 (Divide Conquer) adalah: 4
PS C:\Users\TOSHIBA\BruteForceDivideConquer>
```