

MODUL & GUIDED
Praktikum Pemrograman Berorientasi Objek 2023



POLIMORFISME

Setelah mempelajari beberapa konsep penting dalam Pemrograman Berorientasi Objek seperti Kelas (Class), Objek (Object), dan Pewarisan (Inheritance), konsep selanjutnya adalah mengenai Polimorfisme (Polymorphism). Polimorfisme dapat diartikan dengan “memiliki banyak bentuk”. Dalam konteks pemrograman berorientasi objek, polimorfisme adalah kemampuan yang dimiliki beberapa objek untuk merespon permintaan yang sama dengan caranya masing-masing. Polimorfisme sendiri dibagi menjadi dua jenis yakni polimorfisme statik dan polimorfisme dinamik yang akan dijelaskan lebih lanjut.

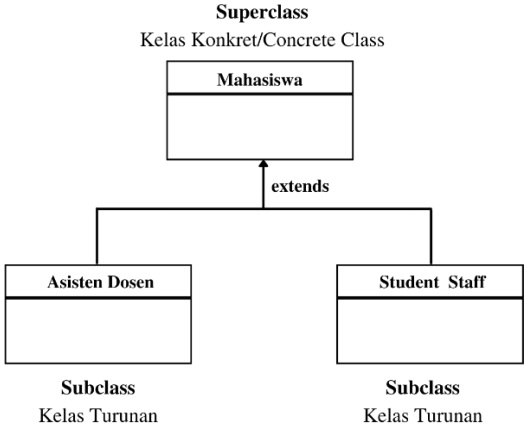
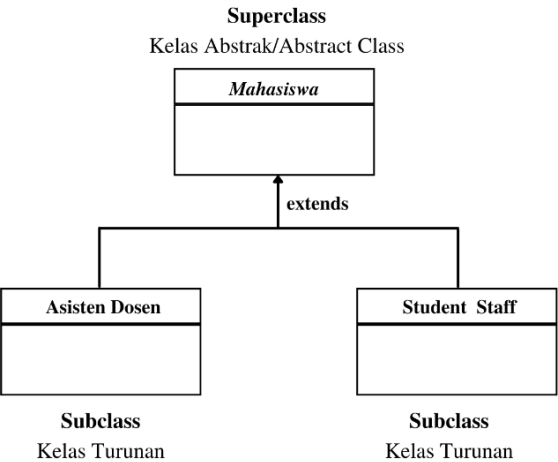
Polimorfisme Statik/Compile Time Polymorphism

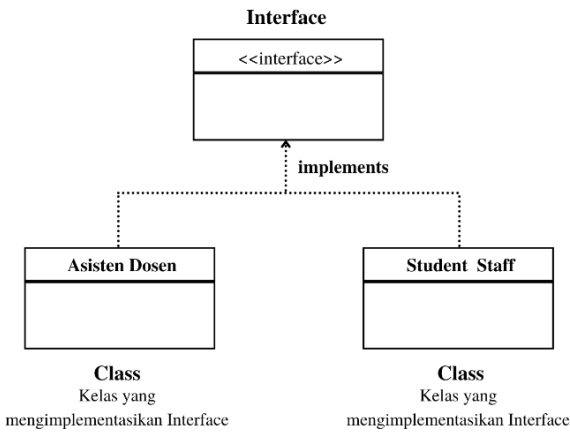
Polimorfisme jenis ini diterapkan dengan melakukan **overloading** methods atau constructors.

<pre>//method 1 public double count(int sks, double hargaSKS){ return sks*hargaSKS; } //method 2 public int count(int sks1, int sks2, int sks3){ return sks1+sks2+sks3; }</pre> <p>Overloading Methods</p>	Methods memiliki nama yang sama dan berada dalam kelas yang sama.
	Jumlah methods dengan nama yang sama dalam satu kelas dapat lebih dari satu.
	Method-method dengan nama yang sama, memiliki parameter yang berbeda tipe dan/atau jumlah dan/atau urutannya.
	Perbedaan tipe balikan/return type tidak berpengaruh.
<pre>//Default Konstruktor public Mahasiswa(){ this.npm="200710716"; this.nama="Novsada Phasa"; } // Konstruktor 1 public Mahasiswa(String npm){ this.npm=npm; this.nama="-"; } //Konstruktor 2 public Mahasiswa(String npm,String nama){ this.npm=npm; this.nama=nama; }</pre> <p>Overloading Constructors</p>	Constructors berada dalam kelas yang sama dan jumlahnya dapat lebih dari satu.
	Setiap constructor yang sama memiliki parameter yang berbeda tipe dan/atau jumlah dan/atau urutannya.

Polimorfisme Dinamik/Runtime Polymorphism

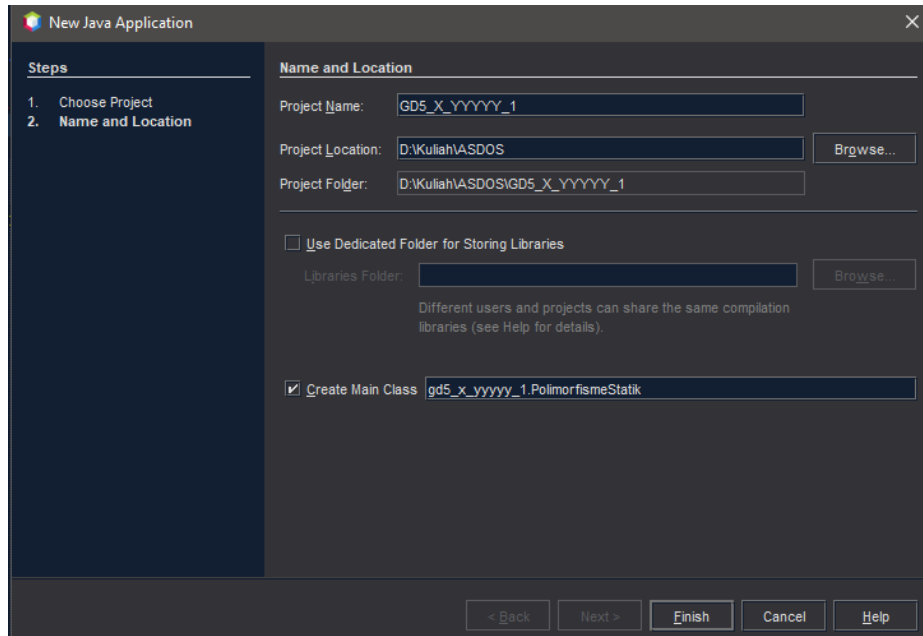
Polimorfisme jenis ini diterapkan dengan melakukan **overriding** methods.

	<p>Menggunakan konsep pewarisan/inheritance untuk menerapkannya. Memperlakukan objek kelas anak dengan tipe kelas induk.</p> <p>Kelas konkret yang menjadi superclass hanya dapat memiliki non-abstract method dan bisa diturunkan kelas turunannya/derived class.</p> <p>Satu kelas turunan (subclass) hanya dapat meng-extends satu kelas induk (superclass). Tidak boleh ada multiple inheritance.</p>
	<p>Menggunakan konsep pewarisan/inheritance untuk menerapkannya. Memperlakukan objek kelas anak dengan tipe kelas induk.</p> <p>Kelas abstrak dapat memiliki atribut, non-abstract method, dan abstract method. Kelas abstrak adalah sebuah kelas yang tidak lengkap, maka kelas ini tidak bisa dibuat objeknya.</p> <p>Semua abstract method dari kelas abstrak induk (superclass) harus dioverride (ditulis implementasinya) di kelas turunannya (subclass).</p> <p>Satu kelas turunan (subclass) hanya dapat meng-extends satu kelas induk (superclass). Tidak boleh ada multiple inheritance.</p>

 <pre> classDiagram class Interface { <<interface>> } class AsistenDosen { } class StudentStaff { } AsistenDosen .. > Interface : implements StudentStaff .. > Interface : implements </pre> <p>The diagram illustrates an interface named 'Interface' (labeled '<<interface>>') at the top. Below it are two classes: 'Asisten Dosen' and 'Student Staff'. Dashed arrows labeled 'implements' point from each class to the interface. Below each class box is the text 'Class' and 'Kelas yang mengimplementasikan Interface'.</p>	<p>Menggunakan konsep interface untuk mengimplementasikannya. Memperlakukan objek kelas dengan tipe interface yang diimplementasikannya.</p>
	<p>Hanya memiliki kumpulan deklarasi unimplemented abstract method dan/atau konstanta. Tidak boleh memiliki atribut dan non-abstract method.</p>
	<p>Semua abstract method dari kelas interface harus dioverride (ditulis implementasinya) di kelas yang mengimplementasikan interface tersebut.</p>
	<p>Satu kelas dapat meng-implements lebih dari satu interface.</p>

GUIDED 1 – POLIMORFISME STATIK

1. Buatlah project baru di Apache Netbeans dengan format nama GD5_X_YYYYY_1 dimana X = kelas dan YYYYYY = 5 digit NPM praktikan. Kemudian, ubah juga nama Main Class menjadi PolimorfismeStatik.



2. Buatlah kelas baru dengan nama Mahasiswa. Kemudian tambahkan code di bawah ini:

```
public class Mahasiswa {
    private String nama;
    private String npm;

    //Default Konstruktor
    public Mahasiswa() {
        this.npm="200710716";
        this.nama="Novsada Phasa";
    }

    // Konstruktor 1
    public Mahasiswa(String npm) {
        this.npm=npm;
        this.nama="-";
    }

    //Konstruktor 2
    public Mahasiswa(String npm,String nama){
        this.npm=npm;
        this.nama=nama;
    }

    public void showMahasiswa() {
        System.err.println("NPM: "+this.npm);
        System.err.println("Nama: "+this.nama);
        System.err.println("-----");
    }
}
```

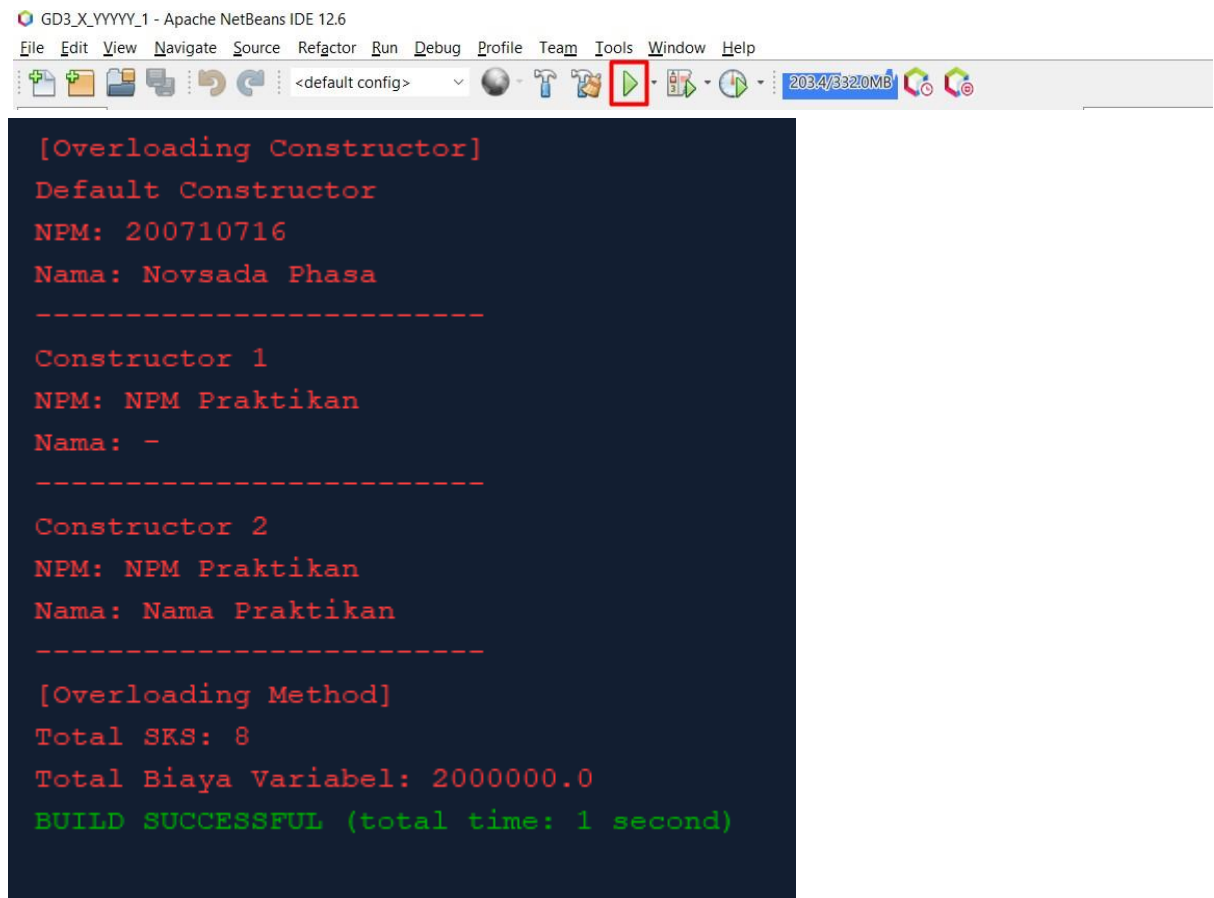
3. Selanjutnya, buatlah kelas baru lagi dengan nama Summary dan tambahkan code di bawah ini:

```
public class Summary {  
  
    //Default Constructor yang dinyatakan secara eksplisit  
    public Summary(){  
  
    }  
    //method 1  
    public double count(int sks, double hargaSKS){  
        return sks*hargaSKS;  
    }  
    //method 2  
    public int count(int sks1, int sks2, int sks3){  
        return sks1+sks2+sks3;  
    }  
}
```

4. Setelah kedua kelas tersebut selesai dibuat, kita Kembali ke Main untuk menguji bagaimana overloading pada constructors dan methods berjalan dengan menambahkan code berikut:

```
public static void main(String[] args) {  
    Mahasiswa mhs= new Mahasiswa();  
    Mahasiswa mhs1= new Mahasiswa("NPM Praktikan");  
    Mahasiswa mhs2= new Mahasiswa("NPM Praktikan","Nama Praktikan");  
  
    System.err.println("[Overloading Constructor]");  
    System.err.println("Default Constructor");  
    mhs.showMahasiswa();  
  
    System.err.println("Constructor 1");  
    mhs1.showMahasiswa();  
  
    System.err.println("Constructor 2");  
    mhs2.showMahasiswa();  
  
    System.err.println("[Overloading Method]");  
    Summary sum=new Summary();  
    //Silahkan mencoba juga untuk mengganti parameter pada kedua method count dengan nilai lain  
    int totalSKS=sum.count(3, 3, 2);//Method 1  
    System.err.println("Total SKS: "+totalSKS);  
    System.err.println("Total Biaya Variabel: "+ sum.count(totalSKS, 250000.0));//method 2  
}
```

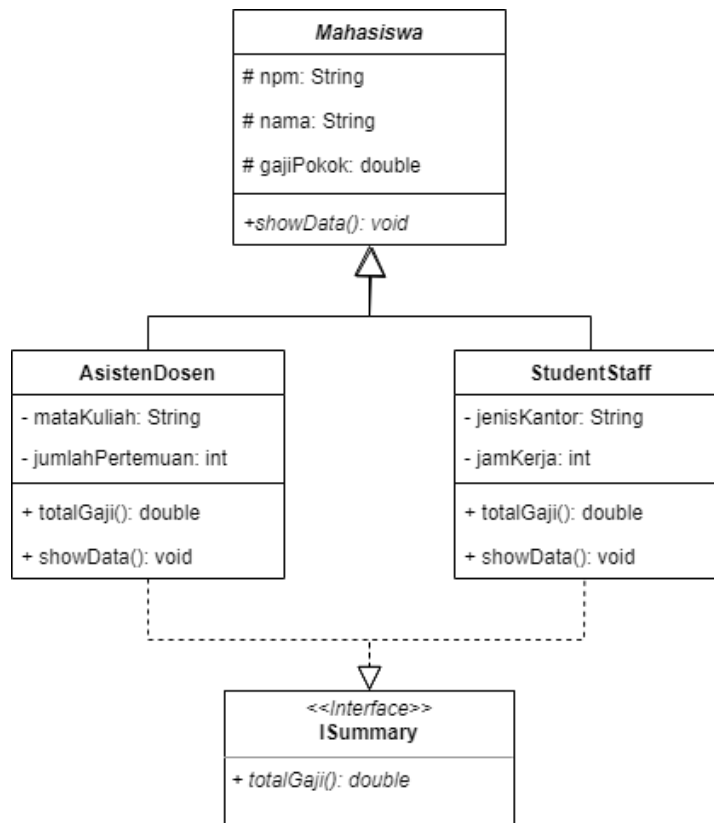
5. Jalankan program dengan menggunakan icon Run atau Fn+F6 / F6 dan lihat outputnya



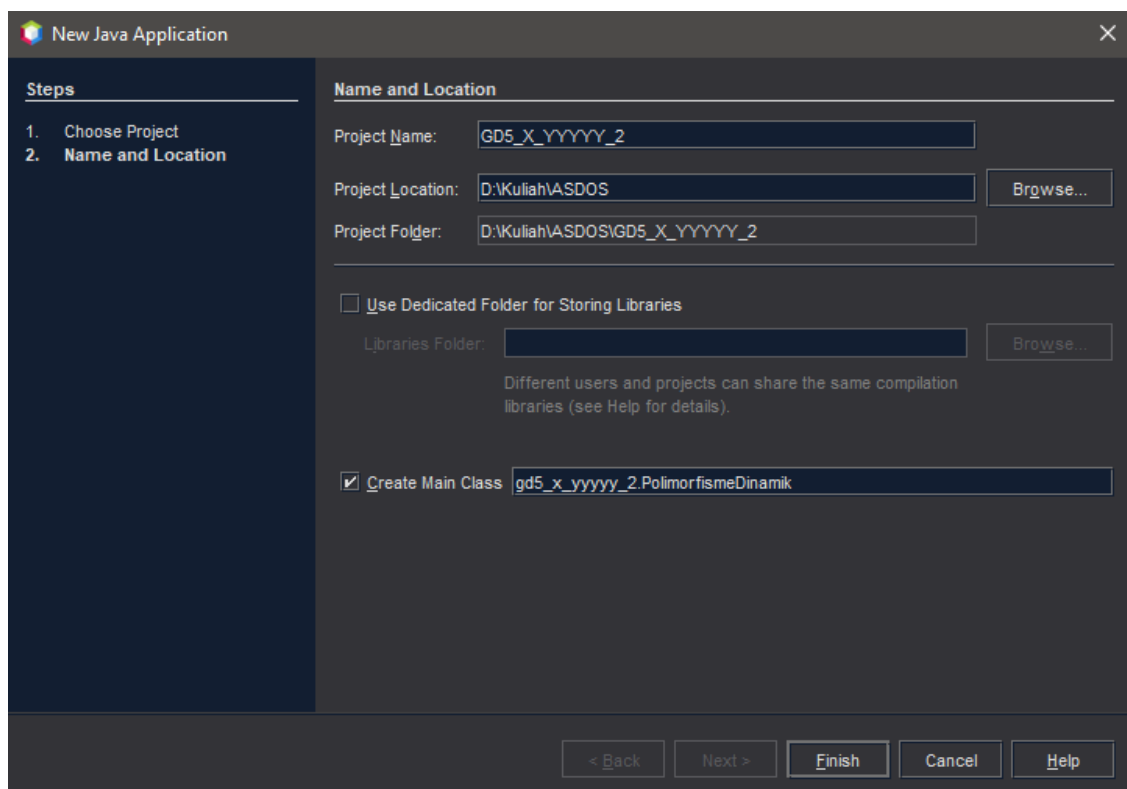
```
GD3_X_YYYY_1 - Apache NetBeans IDE 12.6
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
[Icons] <default config> [Icons] 203.4/532.0MB [Icons]

[Overloading Constructor]
Default Constructor
NPM: 200710716
Nama: Novsada Phasa
-----
Constructor 1
NPM: NPM Praktikan
Nama: -
-----
Constructor 2
NPM: NPM Praktikan
Nama: Nama Praktikan
-----
[Overloading Method]
Total SKS: 8
Total Biaya Variabel: 2000000.0
BUILD SUCCESSFUL (total time: 1 second)
```

GUIDED 2 – POLIMORFISME DINAMIK



1. Buatlah project baru di Apache Netbeans dengan format nama GD5_X_YYYYY_2 dimana X = kelas dan YYYYY = 5 digit NPM praktikan. Kemudian, ubah juga nama Main Class menjadi PolimorfismeDinamik.



2. Dengan menyesuaikan diagram kelas di atas, buatlah kelas baru dengan yakni Mahasiswa dan tambahkan code dalam gambar berikut. Kelas ini nantinya akan berubah menjadi kelas abstrak.

```
public abstract class Mahasiswa {
    protected String npm;
    protected String nama;
    protected double gajiPokok;

    public Mahasiswa(String npm, String nama, double gajiPokok){
        this.npm=npm;
        this.nama=nama;
        this.gajiPokok=gajiPokok;
    }

    //Method abstract ini nantinya harus dioverride/ditulis implementasinya pada kelas turunannya
    public abstract void showData();
}
```

3. Buatlah interface baru dengan nama ISummary. Kemudian, tambahkan code di bawah ini

```
public interface ISummary {
    public double totalGaji();
}
```

4. Buatlah kelas baru dengan nama AsistenDosen dan tambahkan code dalam gambar. Kelas ini berbentuk kelas konkret dan merupakan subclass dari kelas Mahasiswa. Kelas ini akan melakukan overriding method dari kelas abstrak Mahasiswa dan interface ISummary.

```
public class AsistenDosen extends Mahasiswa implements ISummary {
    private String mataKuliah;
    private int jmlPertemuan;

    public AsistenDosen(String npm, String nama, double gajiPokok, String mataKuliah, int jmlPertemuan){
        super(npm, nama, gajiPokok);
        this.mataKuliah=mataKuliah;
        this.jmlPertemuan=jmlPertemuan;
    }

    public double totalGaji() {
        return gajiPokok+(jmlPertemuan*20000);
    }

    public void showData() {
        System.err.println("== Asisten Dosen ==");
        System.err.println("NPM: "+npm);
        System.err.println("Nama: "+nama);
        System.err.println("Gaji Pokok: Rp."+gajiPokok);
        System.err.println("Mata Kuliah: "+mataKuliah);
        System.err.println("Jumlah Pertemuan: "+jmlPertemuan);
        System.err.println("=====");
        System.err.println("Total Gaji: Rp."+totalGaji());
    }
}
```

5. Buatlah kelas baru dengan nama StudentStaff dan tambahkan code dalam gambar. Kelas ini berbentuk konkret dan merupakan subclass dari kelas Mahasiswa. Kelas ini akan melakukan overriding method dari kelas abstrak Mahasiswa dan interface ISummary.

```
public class StudentStaff extends Mahasiswa implements ISummary{
    private String jnsKantor;
    private int jamKerja;
    public StudentStaff(String npm, String nama, double gajiPokok, String jnsKantor, int jamKerja){
        super(npm,nama,gajiPokok);
        this.jnsKantor=jnsKantor;
        this.jamKerja=jamKerja;
    }
    @Override
    public void showData() {
        System.err.println("== Student Staff ==");
        System.err.println("NPM: "+npm);
        System.err.println("Nama: "+nama);
        System.err.println("Gaji Pokok: RP."+gajiPokok);
        System.err.println("Jenis Kantor: "+jnsKantor);
        System.err.println("Jam Kerja: "+jamKerja);
        System.err.println("=====");
        System.err.println("Total Gaji: Rp."+totalGaji());
    }

    @Override
    public double totalGaji() {
        return gajiPokok+(jamKerja*30000);
    }
}
```

6. Selanjutnya, kita Kembali ke Main dan tambahkan code berikut:

```
//TODO 1: Tambahkan impot statement berikut:
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.ArrayList;
```

```
public static BufferedReader br= new BufferedReader(new InputStreamReader(System.in));
```

```

public static void main(String[] args) {
    int jmlPer,jamK,i,count;
    String npm,nama,matkul,kantor;
    double gajiP;
    ArrayList<Mahasiswa> mhs = new ArrayList<>();
    try{
        System.err.println("Banyak data Asisten Dosen yang akan dibuat: "); count = Integer.parseInt(br.readLine());
        for(i=0;i<count;i++){
            System.err.println("[Input data Asisten Dosen "+(i+1)+"]");
            System.err.println("Nomor Pokok Mahasiswa: "); npm=br.readLine();
            System.err.println("Nama Mahasiswa: "); nama=br.readLine();
            System.err.println("Gaji Pokok: "); gajiP=Double.parseDouble(br.readLine());
            System.err.println("Mata Kuliah: "); matkul=br.readLine();
            System.err.println("Jumlah Pertemuan: "); jmlPer=Integer.parseInt(br.readLine());
            mhs.add(new AsistenDosen(npm,nama,gajiP,matkul,jmlPer));
        }
        System.err.println("-----");
        System.err.println("Banyak data Student Staff yang akan dibuat: "); count = Integer.parseInt(br.readLine());
        for(i=0;i<count;i++){
            System.err.println("[Input data Student Staff "+(i+1)+"]");
            System.err.println("Nomor Pokok Mahasiswa: "); npm=br.readLine();
            System.err.println("Nama Mahasiswa: "); nama=br.readLine();
            System.err.println("Gaji Pokok: "); gajiP=Double.parseDouble(br.readLine());
            System.err.println("Jenis Kantor: "); kantor=br.readLine();
            System.err.println("Jam Kerja: "); jamK=Integer.parseInt(br.readLine());
            mhs.add(new StudentStaff(npm,nama,gajiP,kantor,jamK));
        }
        System.err.println("-----");
        for(i=0;i<mhs.size();i++){
            System.err.println("[ Data Mahasiswa yang Bekerja "+(i+1)+"]");
            mhs.get(i).showData();
            System.err.println("-----");
        }
        System.err.println("-----");
    }catch (Exception e){}
}

```

6. Jalankan program dengan menggunakan icon Run atau Fn+F6 / F6 dan lihat outputnya



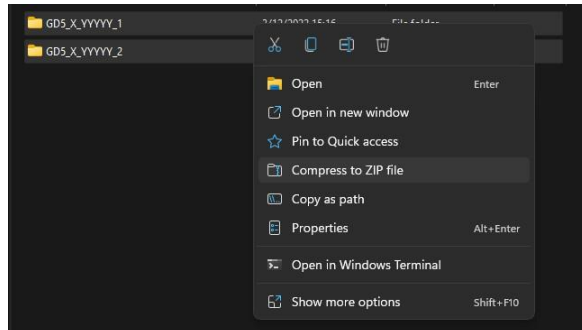
```
Banyak data Asisten Dosen yang akan dibuat:
1
[Input data Asisten Dosen 1]
Nomor Pokok Mahasiswa:
0716
Nama Mahasiswa:
Novsada Phasa
Gaji Pokok:
1000000
Mata Kuliah:
PBO
Jumlah Pertemuan:
14
-----
Banyak data Student Staff yang akan dibuat:
1
[Input data Student Staff 1]
Nomor Pokok Mahasiswa:
0717
Nama Mahasiswa:
Dion Putra
Gaji Pokok:
1500000
Jenis Kantor:
KKACM
Jam Kerja:
4
-----
[ Data Mahasiswa yang Bekerja 1]
== Asisten Dosen ==
NPM: 0716
Nama: Novsada Phasa
Gaji Pokok: RP.1000000.0
Mata Kuliah: PBO
Jumlah Pertemuan: 14
=====
Total Gaji: Rp.1020014.0
-----
[ Data Mahasiswa yang Bekerja 2]
== Student Staff ==
NPM: 0717
Nama: Dion Putra
Gaji Pokok: RP.1500000.0
Jenis Kantor: KKACM
Jam Kerja: 4
=====
Total Gaji: Rp.1620000.0
-----
-----
BUILD SUCCESSFUL (total time: 2 minutes 0 seconds)
```

Catatan tambahan

BufferedReader InputStreamReader	Keduanya digunakan untuk menerima inputan dan untuk menggunakannya, kita perlu melakukan import dengan: <pre>import java.io.BufferedReader; import java.io.InputStreamReader;</pre>
ArrayList	Perbedaan mendasar keduanya adalah Array bersifat static dimana ukuran tetap sesuai deklarasi awal. Sedangkan, ArrayList bersifat dinamis dimana ukurannya akan menyesuaikan dan untuk menggunakannya kita perlu melakukan import dengan: <pre>import java.util.ArrayList;</pre> Beberapa method yang digunakan dalam penggunaan ArrayList: add(), untuk menambah elemen. get(), untuk mengakses. set(), untuk mengubah elemen. remove(), untuk menghapus elemen tertentu. clear(), untuk menghapus semua elemen. size(), untuk mengetahui ukuran ArrayList/banyak elemen.
System.out.print() System.err.print()	Salah satu contoh penggunaan System.out.print() adalah untuk mencetak hasil operasi atau data yang dihasilkan oleh program. Sedangkan, System.err.print() dapat digunakan untuk mencetak pesan kesalahan, seperti ketika terjadi kesalahan pembacaan file atau ketika ada input yang tidak valid. Secara teknis, keduanya memiliki perbedaan dalam cara output di-stream dan di-buffer, tetapi untuk penggunaan umum, perbedaan utama adalah pada tujuan dan konteks penggunaannya.

Pengumpulan Guided

1. Kedua folder GD5_X_YYYYY_1 dan GD5_X_YYYYY_2 di zip menjadi satu dengan nama GD5_X_YYYYY.zip (X = Kelas; Y= 5 digit terakhir NPM)



2. Jangan lupa untuk dikumpulkan dan jika masih ada yang bingung bisa hubungi pemegang modul Sada (WA: 081227115899).