



## POLITEKNIK MASAMY INTERNASIONAL

SK Menristekdikti RI Nomor: 731/KPT/I/2018

Jalan Ikan Paus No.10-15 Kertosari Banyuwangi - 68411  
Telp (0333) 3384593 – <http://polmain.info>

### PROGRAM STUDI D3 TEKNIK KOMPUTER

Form:  
B.Ak/eva/04/20

Nama Dosen : *Arif Fahmi, S.T.,M.T.*

Mata Kuliah : Pemrograman Berorientasi Object

Semester : 4 (Empat)

Kode Mata Kuliah : TKV4044

Th. Akdm : 2019/2020.

## BAB

### PEMROGRAMAN BERORIENTASI OBJECT

#### BAGIAN III

#### SUB BAB :

1. Polimorfisme
2. Perbedaan Method Overloading dengan Method Overriding
3. Interface

#### TUJUAN MATERI

1. Mahasiswa memiliki kompetensi dalam menerapkan konsep polimorfisme dan class abstrak
2. Mahasiswa mampu membuat program java dengan mengaplikasikan konsep dari polimorfisme dan class abstrak

#### REFERENSI

1. J.Eck, D. (2006). *Introduction to Programming Using Java*. Geneva.
2. An Object-Oriented Approach to Programming Logic and Design, Joyce FarrelUSA, 2013
3. An Introduction to Object Oriented Programming with Java, C. Thomas Wu, McGraw-Hill, New York, 2010.
4. <https://docs.oracle.com/javase/tutorial/java/IandI/polymorphism.html>



## PEMROGRAMAN BERORIENTASI OBJECT BAGIAN III

Setelah mengetahui konsep “method override” pada pertemuan daring minggu lalu, mungkin sebagian dari mahasiswa ada yang bertanya. “Apa sebenarnya kegunaan dari proses override?”, sederhananya proses override dibentuk agar java dapat mendukung proses polimorfisme.

### 1. Polimorfisme

*Poly* artinya banyak, *morfisme* artinya bentuk. Polimorfisme (bahasa inggris *polymorphism*) adalah sebuah prinsip dalam biologi di mana oraganisme atau spesias dapat memiliki banyak bentuk atau tahapan (*stages*)<sup>4</sup>.

Prinsip ini juga diadopsi pada pemrograman berorientasikan objek. Sehingga kita dapat definisikan sebagai berikut: Polimorfisme dalam OOP adalah sebuah prinsip di mana class dapat memiliki banyak “**bentuk**” method yang berbeda-beda meskipun namanya sama. “Bentuk” di sini dapat kita artikan: isinya berbeda, parameternya berbeda, dan tipe datanya berbeda.

Polimorfisme pada Java ada dua macam:

- 1. Static Polymorphism (Polimorfisme statis);
- 2. Dynamic Polymorphism (Polimorfisme dinamis).

Beda dari keduanya terletak pada cara membuat polimorfismenya. Polimorfisme statis menggunakan **method overloading** sedangkan polimorfisme dinamis menggunakan **method overriding**. Karena kamu baru mendengar kedua hal ini, mari kita bahas perbedaanya lebih detail.

### 2. Perbedaan method Overloading dengan method Overriding

Secara garis besar perbedaan method overloading dengan method overriding dapat diklasifikasikan sebagai berikut;

| Overloading  | Overriding   |
|--|--|
| <ul style="list-style-type: none"><li>• Nama method sama.</li></ul>          | <ul style="list-style-type: none"><li>• Nama method sama.</li></ul>                                |
| <ul style="list-style-type: none"><li>• Parameter Berbeda.</li></ul>         | <ul style="list-style-type: none"><li>• Isi / Body method berbeda.</li></ul>                       |
| <ul style="list-style-type: none"><li>• Biasanya dalam satu class.</li></ul> | <ul style="list-style-type: none"><li>• Biasanya berbeda class. interface dan implement.</li></ul> |

Kata kunci yang perlu diingat:

“Dalam satu class” , “Nama method sama”, “Tipe data dan parameter beda”

Itulah method **Overloading**.

Bagaimana contohnya?

Misalkan kita, memiliki “class Lingkaran.java”. Pada class ini terdapat “method luas()”. Nah, si [method luas()] ini bisa saja memiliki parameter yang berbeda. Misalnya kita ingin menghitung luas berdasarkan jari-jari (radius) atau diameter. Maka kita dapat membuat class-nya seperti berikut ini:

```
class Lingkaran {  
  
    // method menghitung luas dengan jari-jari  
    float luas(float r){  
        return (float) (Math.PI * r * r);  
    }  
  
    // method menghitung luas dengan diameter  
    double luas(double d){  
        return (double) (1/4 * Math.PI * d);  
    }  
}
```

Coba perhatikan!

“ Class Lingkaran” memiliki dua method yang namanya sama, yakni “ luas() ”.

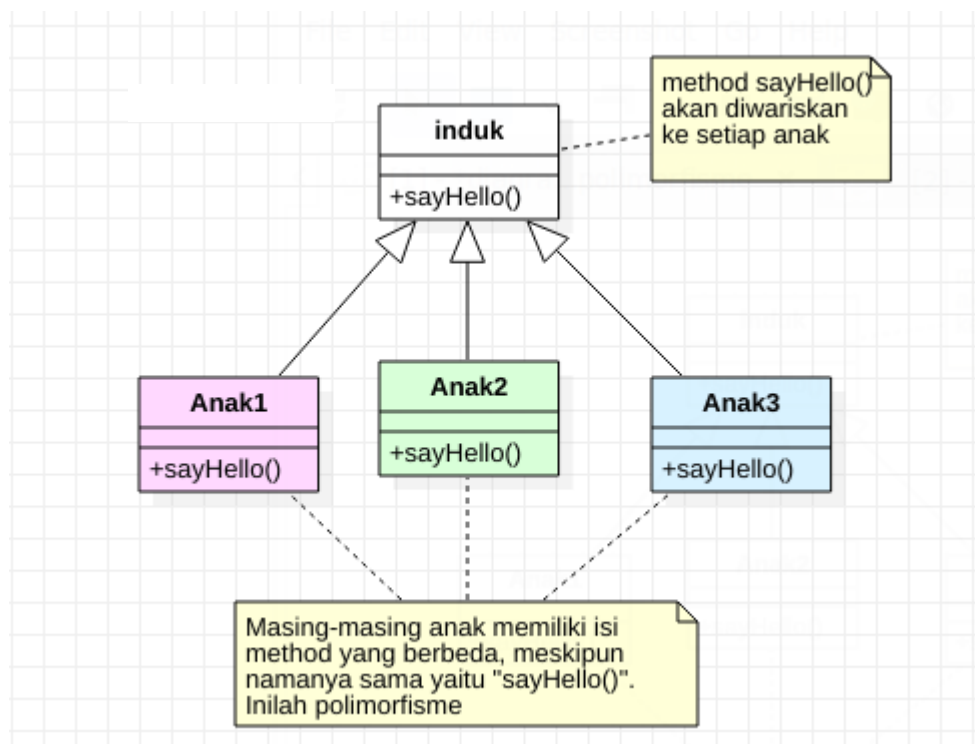
Tapi parameter dan tipe datanya berbeda dan juga isi atau rumus di dalamnya berbeda.

Inilah yang disebut **polimorfisme statis**.

**Plimorfisme dinamis** biasanya terjadi saat kita menggunakan **pewarisan (inheritance)** dan implementasi *interface*.

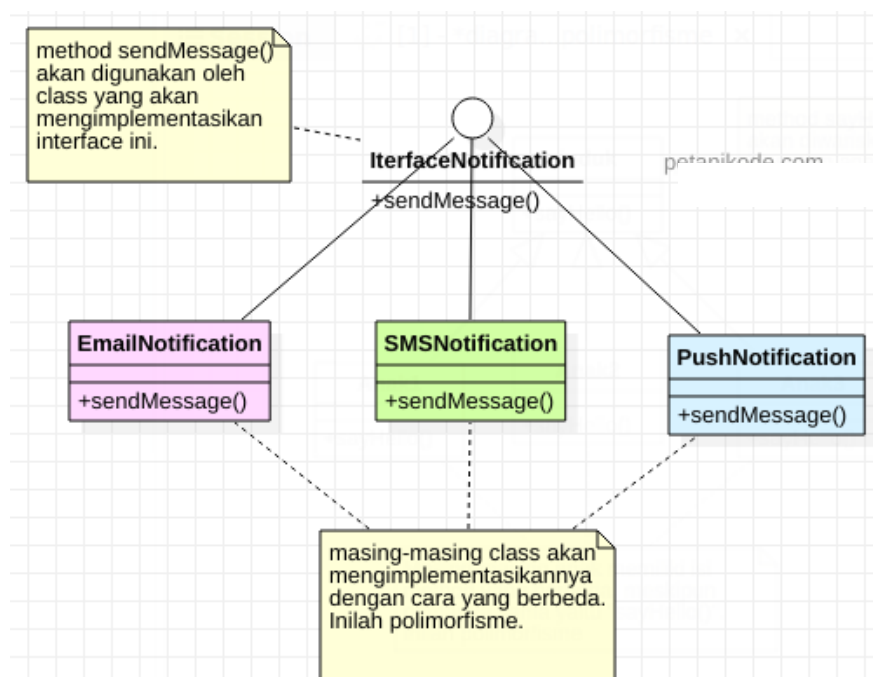
Catatan :

Pada pewarisan, kita bisa mewariskan atribut dan method dari class induk ke class anak. Class anak akan memiliki nama method yang sama dengan class induk dan anak yang lainnya. Nah! Di sinilah akan terjadi **polimorfisme**.



Catatan !

Class anak akan memiliki nama method yang sama, tapi nanti isi dan parameternya bisa berbeda dari class induk. Karena si class anak melakukan **method overriding** (mendingin method) yang diwariskannya. Polimofirme dinamis juga bisa terjadi saat menggunakan *interface*.



### Catatan !

**Interface** adalah *class kosong* yang berisi nama-nama *method* yang nanti harus diimplementasikan pada *class* lain. Dalam pengimplementasiannya bisa saja tiap-tiap *class* akan mengimplementasikan secara berbeda dengan nama *method* yang sama.

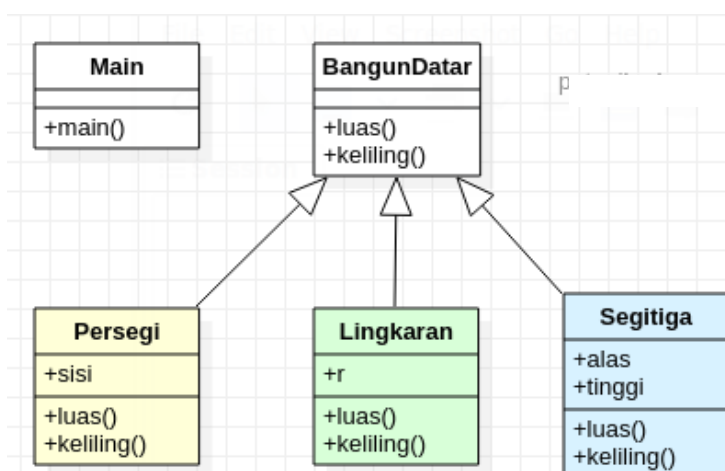
Untuk materi interface akan dibahas pada nomer selanjutnya

Berdasarkan dari pemaparan diatas dapat diketahui bahwa polimorfisme statis hanya terjadi dalam satu class saja. Sedangkan polimorfisme dinamis terjadi pada saat ada hubungan dengan class lain seperti *inheritance*.

### Contoh Program Polimorfisme Dinamis

Contoh program polimorfisme dinamis sebenarnya sudah pernah kita buat pada [pembahasan inheritance](#).

Perhatikan diagram ini:



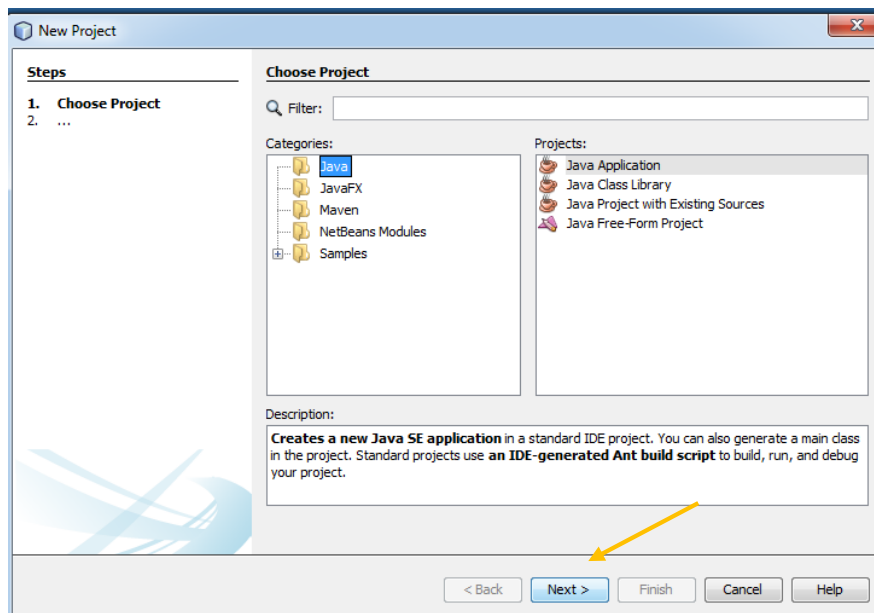
Pada diagram tersebut, terdapat “class BanungDatar” yang memiliki tiga *subclass*, yaitu: [Persegi, Lingkaran, dan Segitiga].

Setiap class memiliki method yang sama yaitu “luas()” dan “keliling()”. Akan tetapi *method-method ini memiliki isi rumus yang berbeda*.

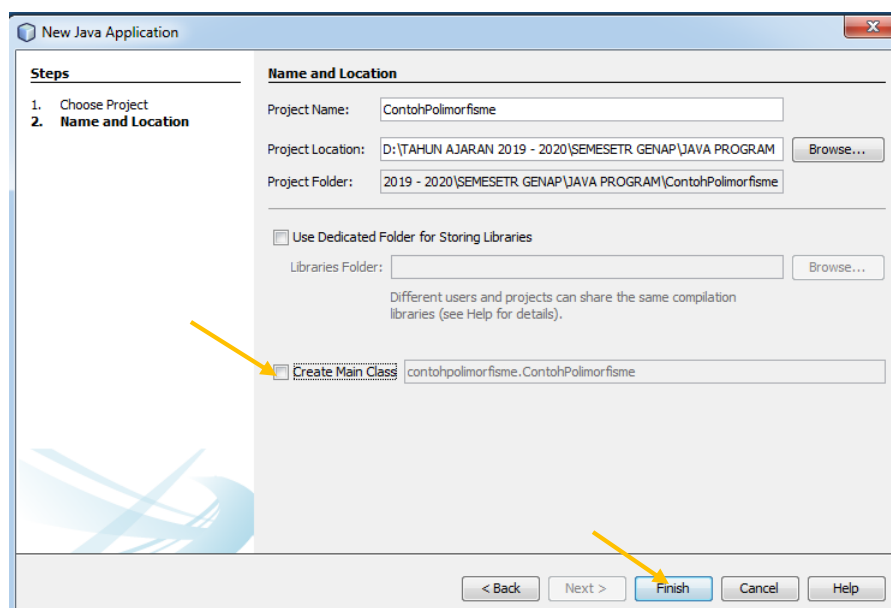


Mari kita coba buat dalam progarm Java.

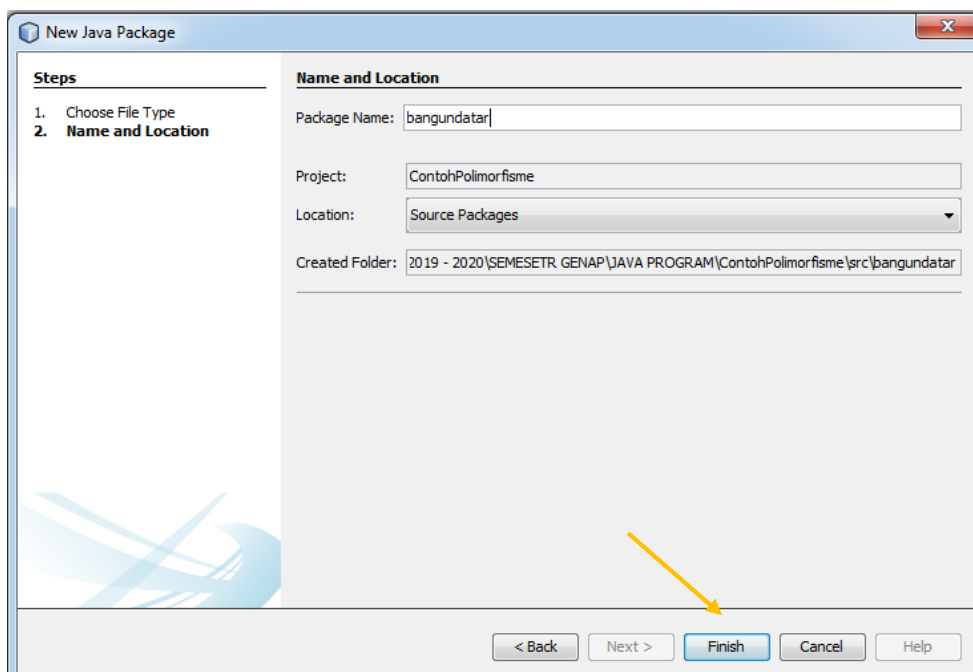
A. Buat project baru di software IDE Netbeans



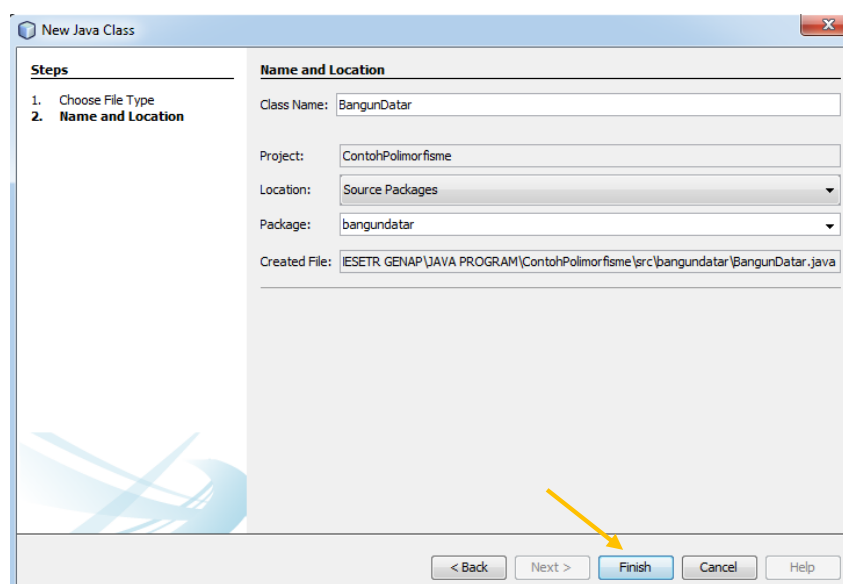
B. Berikan nama “ContohPolimorfisme”:



C. Setelah itu, buat package baru dengan nama “bangundatar”.



D. Buatlah class baru dengan “BangunDatar”.



E. kemudian isi dengan kode berikut:

```
1
2 package bangundatar;
3
4 /**
5  *
6  * @author ARIF FAHMI
7  */
8 public class BangunDatar {
9     float luas() {
10         System.out.println("Menghitung luas bangun datar");
11         return 0;
12     }
13
14     float keliling() {
15         System.out.println("Menghitung keliling bangun datar");
16         return 0;
17     }
18 }
```

F. Berikutnya buat class lagi dengan nama “Persegi” dan isi kodenya seperti ini:

```
1
2 package bangundatar;
3
4 /**
5  *
6  * @author ARIF FAHMI
7  */
8 public class Persegi extends BangunDatar {
9     int sisi;
10
11     public Persegi(int sisi) {
12         this.sisi = sisi;
13     }
14
15     @Override
16     public float luas() {
17         return this.sisi * this.sisi;
18     }
19
20     @Override
21     public float keliling() {
22         return this.sisi * 4;
23     }
24 }
```

G. Berikutnya buat class “Segitiga” dengan isi sebagai berikut:

```
1
2 package bangundatar;
3
4 /**
5  *
6  * @author ARIF FAHMI
7  */
8 public class Segitiga extends BangunDatar{
9     int alas;
10    int tinggi;
11
12    public Segitiga(int alas, int tinggi) {
13        this.alas = alas;
14        this.tinggi = tinggi;
15    }
16
17
18    @Override
19    public float luas(){
20        return this.alas * this.tinggi;
21    }
22 }
```

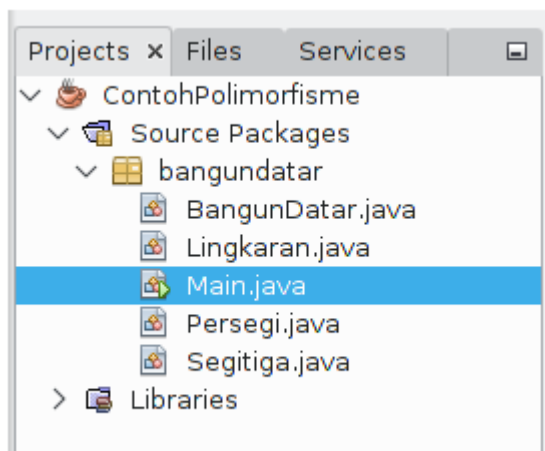
H. Berikutnya buat class “Lingkaran” dengan isi sebagai berikut:

```
1
2 package bangundatar;
3
4 /**
5  *
6  * @author ARIF FAHMI
7  */
8 public class Lingkaran extends BangunDatar {
9     int r;
10
11    public Lingkaran(int r) {
12        this.r = r;
13    }
14
15    @Override
16    public float luas(){
17        return (float) (Math.PI * r * r);
18    }
19
20    @Override
21    public float keliling(){
22        return (float) (2 * Math.PI * r);
23    }
24
25 }
```

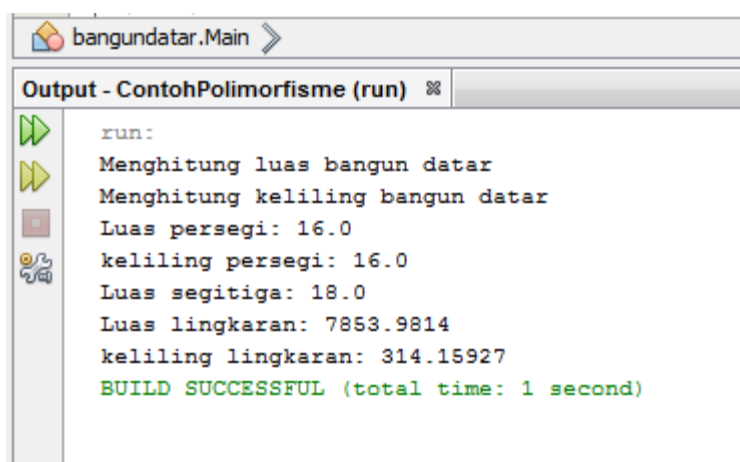
I. Terakhir, buat class “Main” dengan isi sebagai berikut:

```
1
2 package bangundatar;
3
4 /**
5  *
6  * @author ARIF FAHMI
7  */
8 public class Main {
9     public static void main(String[] args) {
10
11        BangunDatar bangunDatar = new BangunDatar();
12        Persegi persegi = new Persegi(4);
13        Segitiga segitiga = new Segitiga(6, 3);
14        Lingkaran lingkaran = new Lingkaran(50);
15
16        // memanggil method luas dan keliling
17        bangunDatar.luas();
18        bangunDatar.keliling();
19
20        System.out.println("Luas persegi: " + persegi.luas());
21        System.out.println("keliling persegi: " + persegi.keliling());
22        System.out.println("Luas segitiga: " + segitiga.luas());
23        System.out.println("Luas lingkaran: " + lingkaran.luas());
24        System.out.println("keliling lingkaran: " + lingkaran.keliling());
25    }
26 }
```

Sekarang kita sudah punya lima class di dalam “package bangundatar”.

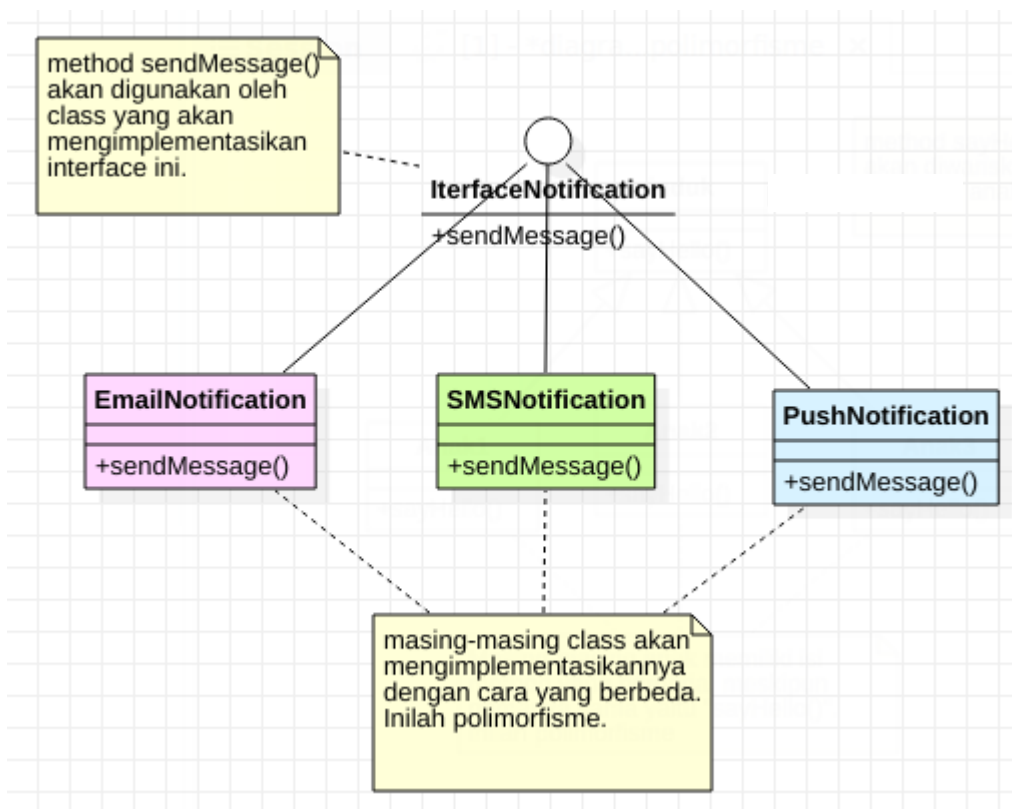


Class yang bisa dijalankan hanyalah class Main, karena ia memiliki method main. Untuk menjalankannya, silahkan klik kanan pada class main.. lalu pilih **Run File**.



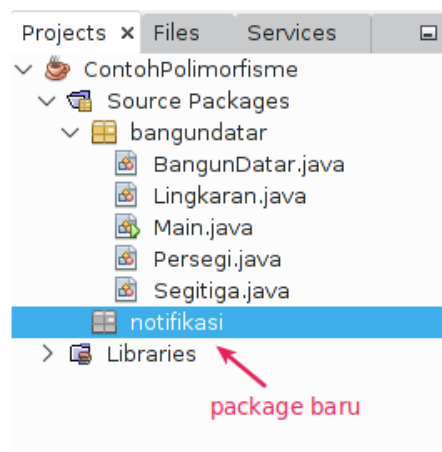
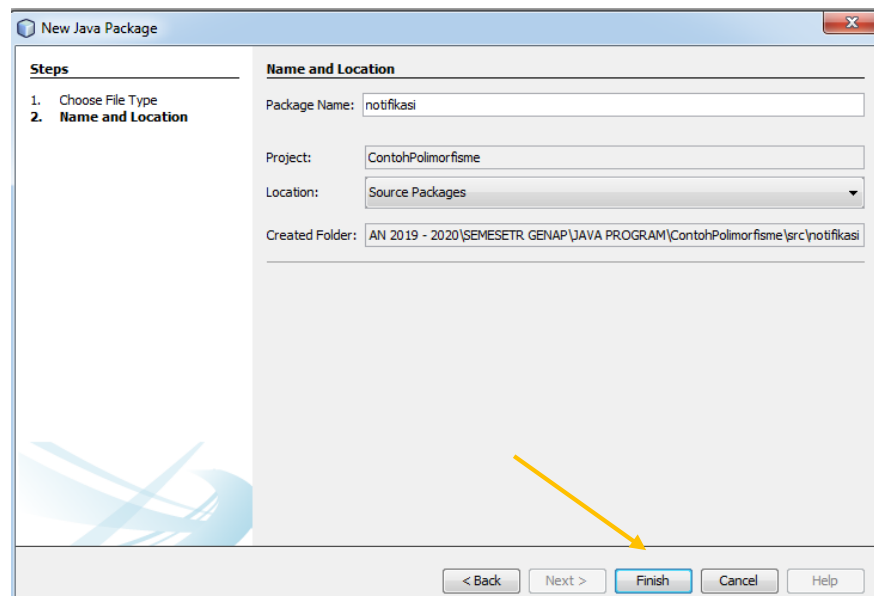
### Contoh Polimorfisme selanjutnya.

kita akan membuat aplikasi untuk mengirim notifikasi dengan Interface. Ini adalah bentuk diagramnya:





A. Pada proyek “ContohPolimorfisme”, buatlah package baru dengan nama “notifikasi”.



B. Setelah itu, di dalam package notifikasi. buatlah class baru dengan nama “InterfaceNotifikasi” dan isi kodenya seperti ini:

```
1
2 package notifikasi;
3
4 /**
5  *
6  * @author ARIF FAHMI
7  */
8 public interface InterfaceNotifikasi {
9     void sendMessage(String receiver, String content);
10 }
11
```

C. Berikutnya, di dalam package yang sama. buatlah class baru dengan nama “EmailNotification” dan isi seperti ini:

```
1
2 package notifikasi;
3
4 /**
5  *
6  * @author ARIF FAHMI
7  */
8 public class EmailNotification implements InterfaceNotifikasi{
9
10     @Override
11     public void sendMessage(String receiver, String content) {
12         System.out.println("Mengirim email ke " + receiver + " dengan isi:");
13         System.out.println(content);
14     }
15
16 }
```

D. Berikutnya, di dalam package yang sama. buatlah class baru dengan nama “SMSNotification” dan isi seperti ini:

```
1
2 package notifikasi;
3
4 /**
5  *
6  * @author ARIF FAHMI
7  */
8 public class SMSNotification implements InterfaceNotifikasi{
9
10     @Override
11     public void sendMessage(String receiver, String content) {
12         System.out.println("Mengirim SMS ke " + receiver + " dengan isi:");
13         System.out.println(content);
14     }
15
16 }
```

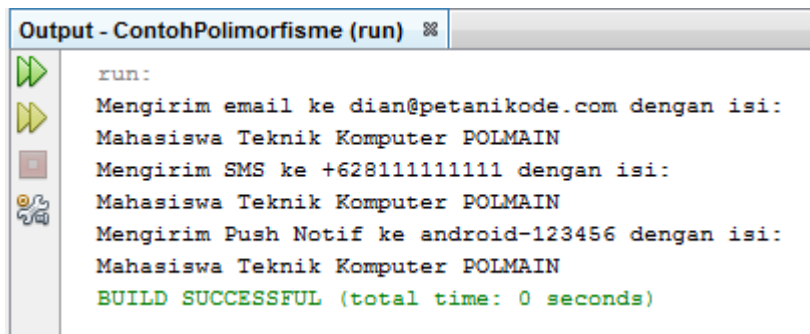
E. Berikutnya, di dalam package yang sama.. buatlah class baru dengan nama “PushNotification” dan isi seperti ini:

```
1
2 package notifikasi;
3
4 /**
5  *
6  * @author ARIF FAHMI
7  */
8 public class PushNotification implements InterfaceNotifikasi{
9
10     @Override
11     public void sendMessage(String receiver, String content) {
12         System.out.println("Mengirim Push Notif ke " + receiver + " dengan isi:");
13         System.out.println(content);
14     }
15
16 }
```

F. Terakhir, buatlah class “Main” di dalam package notifikasi dan isi kodenya seperti ini:

```
1
2 package notifikasi;
3
4 /**
5  *
6  * @author ARIF FAHMI
7  */
8 public class Main {
9
10     public static void main(String[] args) {
11         String emailPenerima = "dian@petanikode.com";
12         String nomerHp = "+628111111111";
13         String mobileId = "android-123456";
14
15         EmailNotification emailNotif = new EmailNotification();
16         SMSNotification smsNotif = new SMSNotification();
17         PushNotification pushNotif = new PushNotification();
18
19         String message = "Mahasiswa Teknik Komputer POLMAIN";
20
21         emailNotif.sendMessage(emailPenerima, message);
22         smsNotif.sendMessage(nomerHp, message);
23         pushNotif.sendMessage(mobileId, message);
24     }
25 }
```

### G. outputnya



```
run:
Mengirim email ke dian@petanikode.com dengan isi:
Mahasiswa Teknik Komputer POLMAIN
Mengirim SMS ke +628111111111 dengan isi:
Mahasiswa Teknik Komputer POLMAIN
Mengirim Push Notif ke android-123456 dengan isi:
Mahasiswa Teknik Komputer POLMAIN
BUILD SUCCESSFUL (total time: 0 seconds)
```

Walapun semua isi “**method sendMessage()**” sama, namun nanti pada implementasinya di dunia nyata akan beda-beda. “**EmailNotification**” biasanya akan mengirim email dengan metode SMTP atau API. Sedangkan SMS bisa melalui SMS Gateway. Begitu juga dengan “**PushNotofication**”, nanti akan menggunakan API yang berbeda.

Nah itulah contoh polimorfisme dengan interface

#### Catatan :

Platform java memiliki dua komponen, yaitu :

1. Java Virtual Machine (JVM) merupakan dasar bagi platform Java yang berjalan di atas sistem operasi.
2. Java Application Programming Interface (API) merupakan kumpulan kode java siap pakai untuk berbagai keperluan seperti untuk User Interface, networking dan lain sebagainya.

### 3. Interface

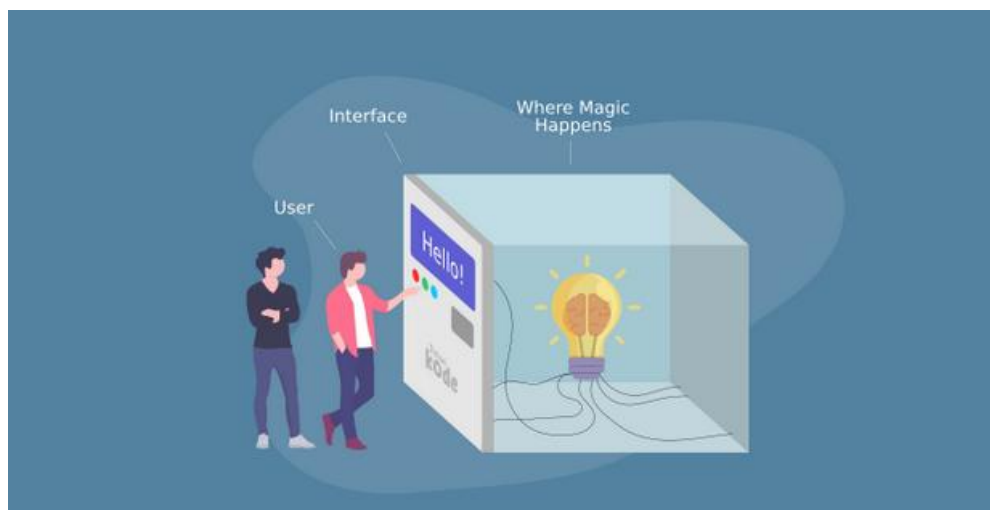
Interface terdiri dari dua kata yakni, inter (antar) dan face (muka). Jadi interface adalah antarmuka. Kata ini mungkin sering kamu dengar. Contohnya seperti: GUI (*Graphical User Interface*) dan CLI (*Command Line Interface*).

Secara umum, *interface* berfungsi sebagai **penghubung** antara sesuatu yang 'abstrak' dengan sesuatu yang nyata.

#### Contoh Analogi Interface

Coba perhatikan handphone-mu, ada berapa tombol di sana?

Kalau HP zaman sekarang, paling cuma ada dua. yakni tombol power dan tombol volume. Tombol-tombol inilah yang dimaksud interface. Sementara isi dalam HP-nya, kamu tak peduli atau tak akan tahu seberapa kompleks dan abstrak sistem yang ada di sana.



Saat kamu menekan tombol di HP mu, ada beberapa hal yang mungkin terjadi:

- HP menyala;
- HP mati;
- Suara HP tambah besar;
- Suara HP mengecil;
- Mengambil gambar dari kamera;
- dll.

Ini tergantung bagaimana implementasi yang diberikan untuk tombol-tombol tersebut. Beda merek HP, beda lagi cara mereka mengimplementasikannya. Misal pada HP tertentu, jika menekan tombol volume 3x. maka akan mengambil screenshot. Sementara di HP merek lain, kita tidak bisa melakukan ini.

Nah, konsep ini juga ada di dalam OOP.

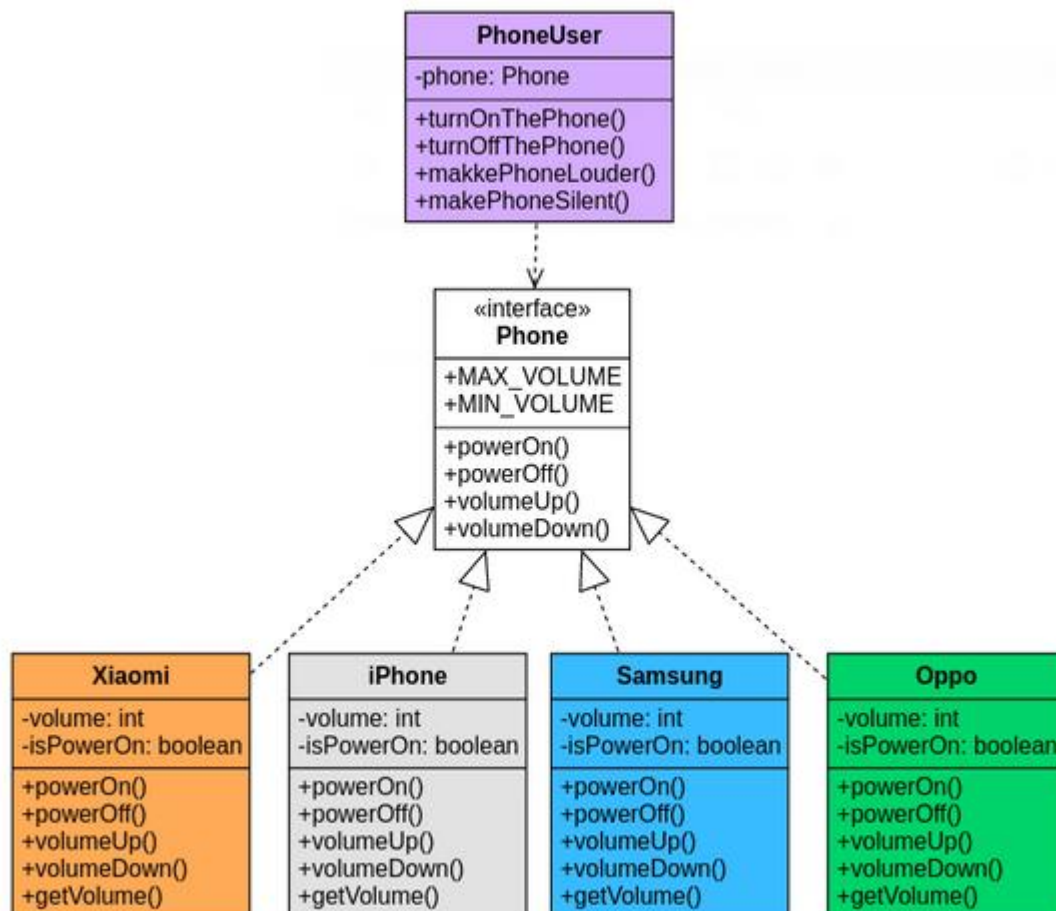
## Contoh Program Interface

Biar nyambung dengan penjelasan saya yang tadi, kita akan pakai contoh Handphone.

Jadi. ada dua objek yang akan kita hubungkan dengan interface.

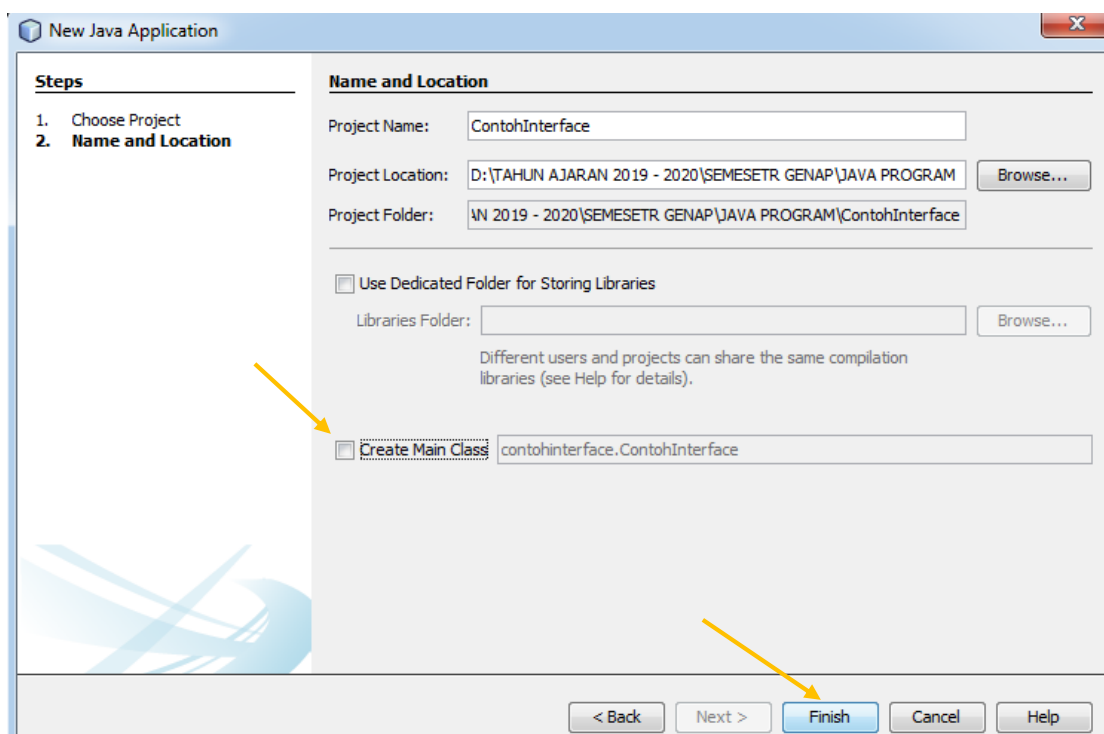
Yakni manusia dan handphone. Manusianya kita sebut saja sebagai PhoneUser. Karena dia akan menggunakan objek HP.

Maka dapat kita buat class diagramnya seperti berikut ini:



Mari kita coba buat dengan software IDE Netbeans.

A. Buatlah proyek baru dengan nama “ContohInterface”.



- B. Setelah itu, buat class baru di dalam <default package> dengan nama Phone dan isi kodenya seperti ini:

```
1  /**
2   *
3   * @author ARIFFAHMMI
4   */
5  public interface Phone {
6      int MAX_VOLUME = 100;
7      int MIN_VOLUME = 0;
8
9      void powerOn();
10     void powerOff();
11     void volumeUp();
12     void volumeDown();
13 }
```

- C. Berikutnya, buat lagi class baru dengan nama PhoneUser dan isi kodenya seperti ini:

```
1  /**
2   *
3   * @author ARIF FAHMI
4   */
5  public class PhoneUser {
6      private Phone phone;
7
8      public PhoneUser(Phone phone) {
9          this.phone = phone;
10     }
11
12     void turnOnThePhone() {
13         this.phone.powerOn();
14     }
15
16     void turnOffThePhone() {
17         this.phone.powerOff();
18     }
19
20     void makePhoneLouder() {
21         this.phone.volumeUp();
22     }
23
24     void makePhoneSilent() {
25         this.phone.volumeDown();
26     }
27 }
```

- D. Berikutnya kita akan membuat class implementasi dari interface Phone. Kita coba buat satu saja dulu. Silahkan buat class baru dengan nama Xiaomi, kemudian isi kodenya seperti ini:

```
1  /**
2   *
3   * @author ARIF FAHMI
4   */
5  public class Xiaomi implements Phone {
6
7      private int volume;
8      private boolean isPowerOn;
9
10     public Xiaomi() {
11         // set volume awal
12         this.volume = 50;
13     }
14 }
```

```

15  @Override
16  public void powerOn() {
17      isPowerOn = true;
18      System.out.println("Handphone menyala...");
19      System.out.println("Selamat datang di XIAOMI");
20      System.out.println("Android version 29");
21  }
22
23  @Override
24  public void powerOff() {
25      isPowerOn = false;
26      System.out.println("Handphone dimatikan");
27  }
28
29  @Override
30  public void volumeUp() {
31      if (isPowerOn) {
32          if (this.volume == MAX_VOLUME) {
33              System.out.println("Volume FULL!!");
34              System.out.println("sudah " + this.getVolume() + "%");
35          } else {
36              this.volume += 10;
37              System.out.println("Volume sekarang: " + this.getVolume());
38          }
39      } else {
40          System.out.println("Nyalakan dulu donk HP-nya!!");
41      }
42  }
43
44  @Override
45  public void volumeDown() {
46      if (isPowerOn) {
47          if (this.volume == MIN_VOLUME) {
48              System.out.println("Volume = 0%");
49          } else {
50              this.volume -= 10;
51              System.out.println("Volume sekarang: " + this.getVolume());
52          }
53      } else {
54          System.out.println("Nyalakan dulu donk HP-nya!!");
55      }
56  }
57
58  public int getVolume() {
59      return this.volume;
60  }
61
62  }

```

E. Terakhir, buatlah class Main dengan isi sebagai berikut:

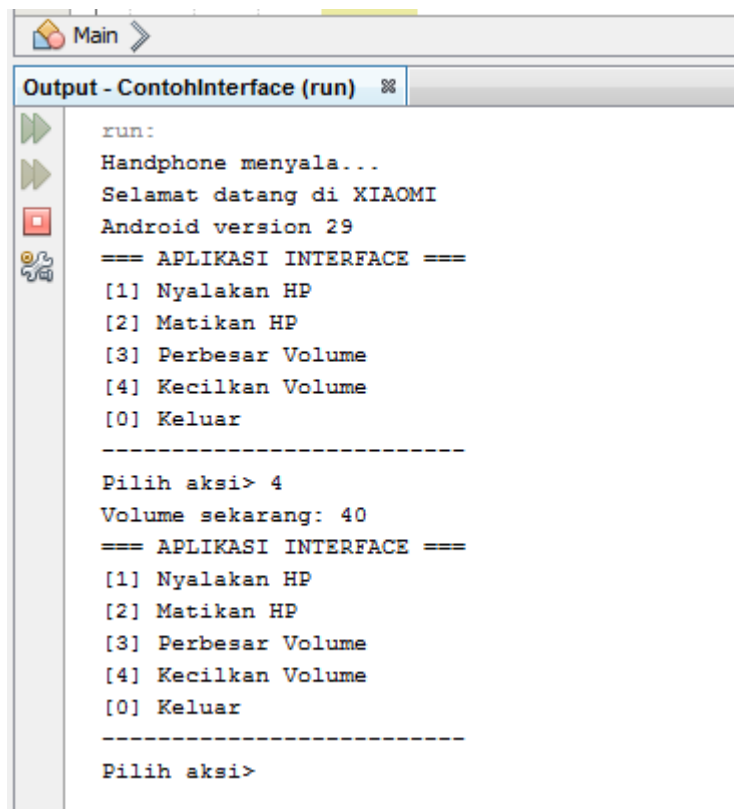
```

1  /**
2   *
3   * @author ARIF FAHMI
4   */
5  import java.util.Scanner;
6
7
8  public class Main {
9
10     public static void main(String[] args) {
11
12         // membuat objek HP
13         Phone redmiNote8 = new Xiaomi();
14
15         // membuat objek user
16         PhoneUser dian = new PhoneUser(redmiNote8);
17
18         // kita coba nyalakan HP-nya
19         dian.turnOnThePhone();
20
21         // biar enak, kita buat dalam program
22         Scanner input = new Scanner(System.in);
23         String aksi;

```

```
24
25     while (true) {
26         System.out.println("=== APLIKASI INTERFACE ===");
27         System.out.println("[1] Nyalakan HP");
28         System.out.println("[2] Matikan HP");
29         System.out.println("[3] Perbesar Volume");
30         System.out.println("[4] Kecilkan Volume");
31         System.out.println("[0] Keluar");
32         System.out.println("-----");
33         System.out.print("Pilih aksi> ");
34         aksi = input.nextLine();
35
36         if(aksi.equalsIgnoreCase("1")){
37             dian.turnOnThePhone();
38         } else if (aksi.equalsIgnoreCase("2")){
39             dian.turnOffThePhone();
40         } else if (aksi.equalsIgnoreCase("3")){
41             dian.makePhoneLouder();
42         } else if (aksi.equalsIgnoreCase("4")){
43             dian.makePhoneSilent();
44         } else if (aksi.equalsIgnoreCase("0")){
45             System.exit(0);
46         } else {
47             System.out.println("Kamu memilih aksi yang salah!");
48         }
49     }
50
51 }
```

## F. Output



```
run:
Handphone menyala...
Selamat datang di XIAOMI
Android version 29
=== APLIKASI INTERFACE ===
[1] Nyalakan HP
[2] Matikan HP
[3] Perbesar Volume
[4] Kecilkan Volume
[0] Keluar
-----
Pilih aksi> 4
Volume sekarang: 40
=== APLIKASI INTERFACE ===
[1] Nyalakan HP
[2] Matikan HP
[3] Perbesar Volume
[4] Kecilkan Volume
[0] Keluar
-----
Pilih aksi>
```



Kalau kita perhatikan di dalam method `main()`, kita hanya menggunakan objek dari class “PhoneUser” saja.

Coba lihat kode-kode ini:

```
dian.turnOnThePhone();
dian.turnOnThePhone();
dian.turnOffThePhone();
dian.makePhoneLouder();
dian.makePhoneSilent();
```

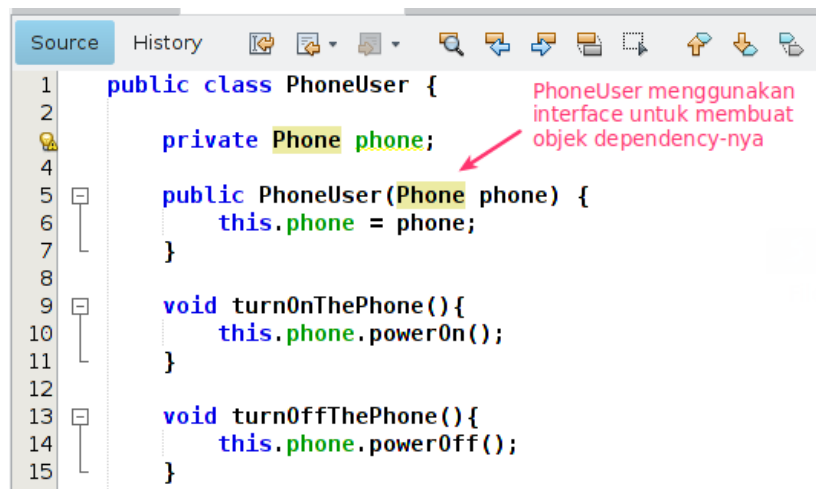
Saat kita memanggil method-method ini, maka method yang ada di dalam objek “redmiNote8” yang akan dipanggil. Ini karena mereka terhubung dengan interface.

Lalu pertanyaanya: **Mengapa Harus Pakai Interface?**

Sebenarnya kita tidak harus selalu menggunakan interface. Tapi dalam kasus tertentu, interface akan sangat berguna.

**Misalnya,**

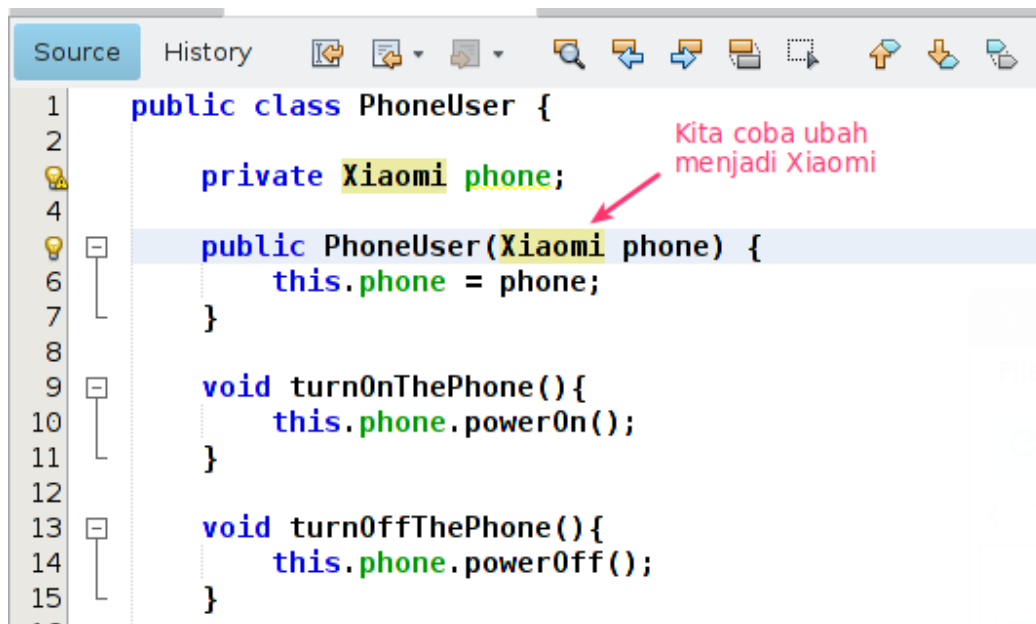
Kita tahu objek “PhoneUser” membutuhkan objek dari handphone. Tanpa objek handphone, si objek user nggak akan bisa kita buat. Ini karena kita menetapkan pada constructornya. ini juga disebut hubungan *dependency*.



```
1 public class PhoneUser {
2
3     private Phone phone;
4
5     public PhoneUser(Phone phone) {
6         this.phone = phone;
7     }
8
9     void turnOnThePhone(){
10         this.phone.powerOn();
11     }
12
13     void turnOffThePhone(){
14         this.phone.powerOff();
15     }
16 }
```

Nah, di sana kita menggunakan interface Phone untuk membaut objek dependency-nya. Bila tidak pakai interface apa yang terjadi?

Untuk menjawab pertanyaan ini, mari kita coba ubah Phone menjadi Xiaomi.



```
1 public class PhoneUser {
2
3     private Xiaomi phone;
4
5     public PhoneUser(Xiaomi phone) {
6         this.phone = phone;
7     }
8
9     void turnOnThePhone(){
10         this.phone.powerOn();
11     }
12
13     void turnOffThePhone(){
14         this.phone.powerOff();
15     }
16 }
```



## Java Programming

### Apa yang akan terjadi?

Untuk saat ini tidak akan terjadi apa-apa. Tapi sekarang bagaimana kalau kita punya objek baru dari class yang mengimplementasikan interface Phone juga.

Kita sebut saja objek baru tersebut adalah iPhoneX yang merupakan instance dari class iPhone.

```
1 import java.util.Scanner;
2
3
4
5 public class Main {
6
7     public static void main(String[] args) {
8
9         // membuat objek HP
10        Xiaomi redmiNote8 = new Xiaomi();
11
12        // buat objek HP baru dari class iPhone
13        iPhone iPhoneX = new iPhone();
14
15        // membuat objek user
16        PhoneUser dian = new PhoneUser(iPhoneX);
17
18        // kita coba nyalakan HP-nya
19        dian.turnOnThePhone();
20
21        // biar enak, kita buat dalam program
22        Scanner input = new Scanner(System.in);
23        String aksi;
```

Tapi si User tidak akan bisa pakai HP ini, karena dia butuhnya cuma Xiaomi Saja

Maka objek iPhoneX tidak akan bisa diberikan kepada objek PhoneUser. Karena di konstruktornya dia hanya bisa pakai Xiaomi saja.

Jadi, mengapa kita harus pakai interface?

Interface akan membuat si objek PhoneUser bisa menggunakan objek handphone apapun, dengan syarat objek handphone tersebut harus mengimplementasikan method dari interface Phone.

### Hal-Hal yang tidak boleh dilakukan di Interface

Ada beberapa hal yang tidak boleh kamu lakukan saat membuat interface:

- Jangan buat variabel di dalam interface, tapi membuat konstanta boleh;
- Jangan mengisi method-nya, cukup tuliskan nama method, tipe data, dan parameter saja. Tapi untuk default metod boleh punya isi.

Contoh;

```
public interface Phone{
    void powerOn();
    void powerOff();
    void volumeUp();
    void volumeDown();

    default void charging(){
        System.out.println("Phone charging...")
    }
}
```

- Jangan berikan modifier *private* ataupun *protected* pada method dan konstanta yang ada di dalam interface.
- Interface tidak bisa dibuat objek instance-nya dengan kata kunci new.

### Kesimpulannya:

Interface merupakan penghubung antar objek. Interface bersifat abstrak, sehingga objek yang menggunakannya tidak akan peduli bagaimana ia diimplementasikan.

Karena bersifat abstrak, interface tidak bisa dibuat objek *instance* dengan kata kunci new.

Interface sebenarnya mengamalkan prinsip Abstraksi dan Enkapsulasi (pembungkusan).



# Tugas

## Catatan.

- a. Tugas dikumpulkan maks pada hari Jum'at tgl 02-05-2020 pukul 16.00
- b. Tugas boleh dikumpulkan dalam bentuk (PDF,PPT,WORD)
- c. File dikirim ke email : [fahmi03031995@gmail.com](mailto:fahmi03031995@gmail.com)
- d. Tugas program dapat dikerjakan di aplikasi (Dcoder, java ide for android, maupun software Netbeans,atau Notepad)
- e. Format file dengan tata cara: NamaMhs\_NIM\_PBO

## Soal

1. Buat resume tentang materi Polimorfisme dan interface pada java
2. Buat listring program yang menerapkan konsep Polimorfisme dan interface serta deskripsikan hasil program anda.