

LAPORAN MODUL PRAKTIKUM 2
PEMROGRAMAN BERORIENTSAI OBJEK DAN PRAKTEK

Laporan Modul Praktikum 2 :
Disusun untuk memenuhi tugas pertemuan ke tujuh matakuliah
Pemrograman Berorientasi Objek dan Praktek A



UIN SUNAN AMPEL
S U R A B A Y A

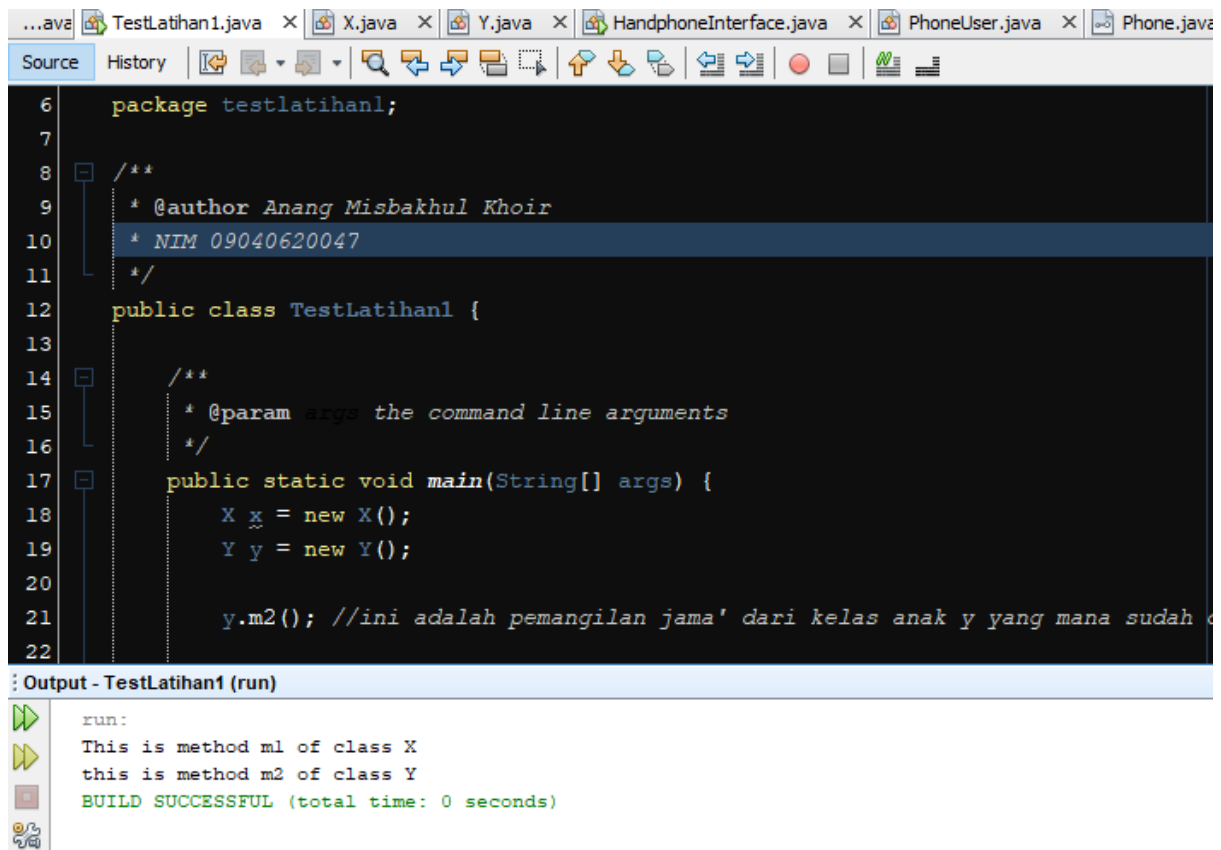
Oleh:
Anang Misbakhul Khoir (09040620047)

Dosen Pengampu:
Dwi Rolliawati, MT

PROGRAM STUDI SISTEM INFORMASI
FAKULTAS SAINS DAN TEKNOLOGI
UIN SUNAN AMPEL SURABAYA

Latihan 1 Polymorfisme

1. Praktikkan source code yang sudah ada dimodul dan terapkan.
 - a. Simpan, Kompilasi & Jalankan kode. Observasi bagaimana keluarannya.



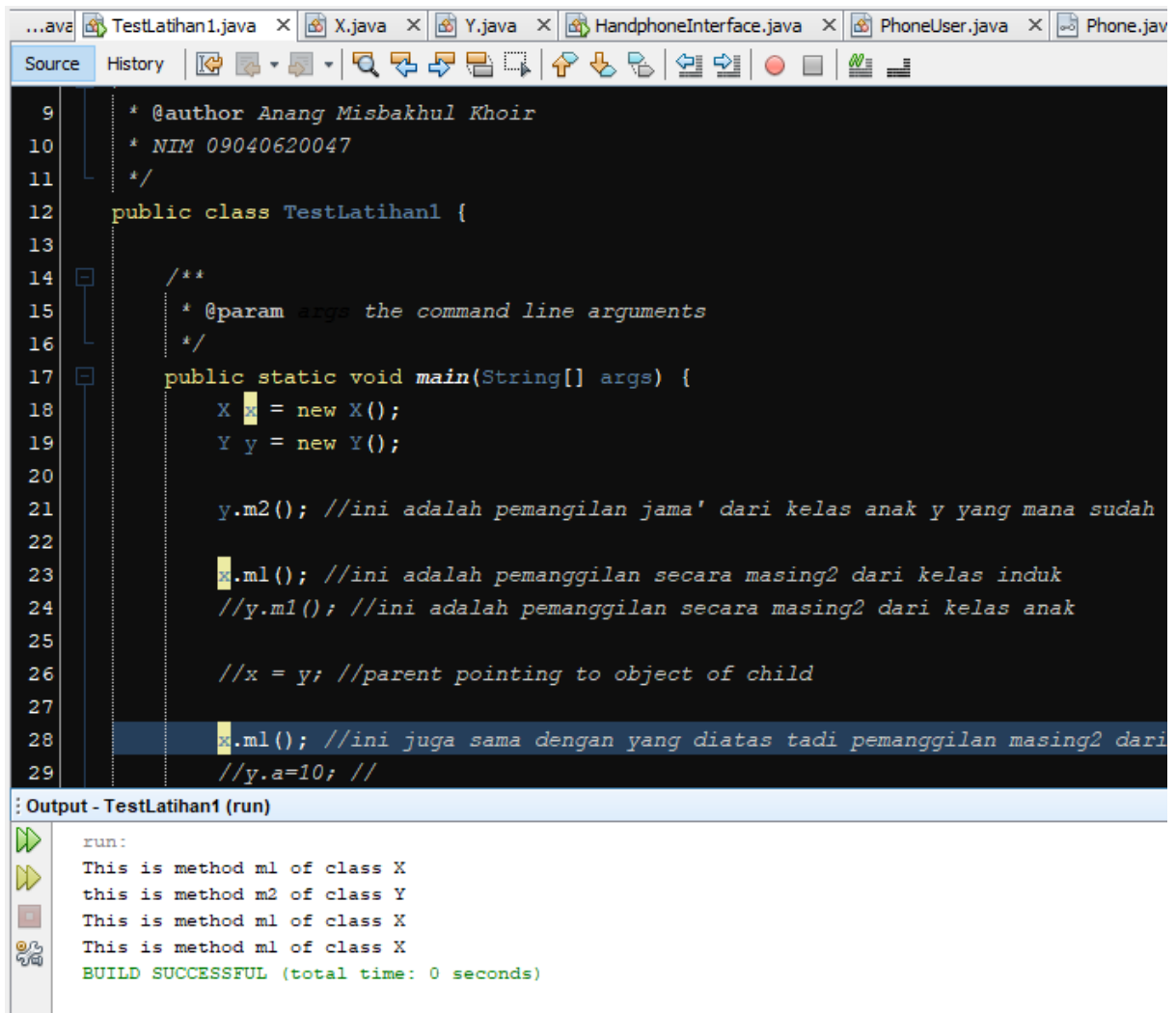
```
6 package testlatihan1;
7
8 /**
9  * @author Anang Misbakhul Khoir
10  * NIM 09040620047
11  */
12 public class TestLatihan1 {
13
14     /**
15      * @param args the command line arguments
16      */
17     public static void main(String[] args) {
18         X x = new X();
19         Y y = new Y();
20
21         y.m2(); //ini adalah pemanggilan jama' dari kelas anak y yang mana sudah
22     }
```

Output - TestLatihan1 (run)

```
run:
This is method m1 of class X
this is method m2 of class Y
BUILD SUCCESSFUL (total time: 0 seconds)
```

Percobaan code yang pertama ini adalah pemanggilan jama' dari kelas anak y yang mana sudah dijadikan stu juga untuk pemangilan dari kelas induk

- b. Hapus komentar pada baris # 6-9. Simpan, Kompilasi & Jalankan kode. Observasi bagaimana keluarannya.



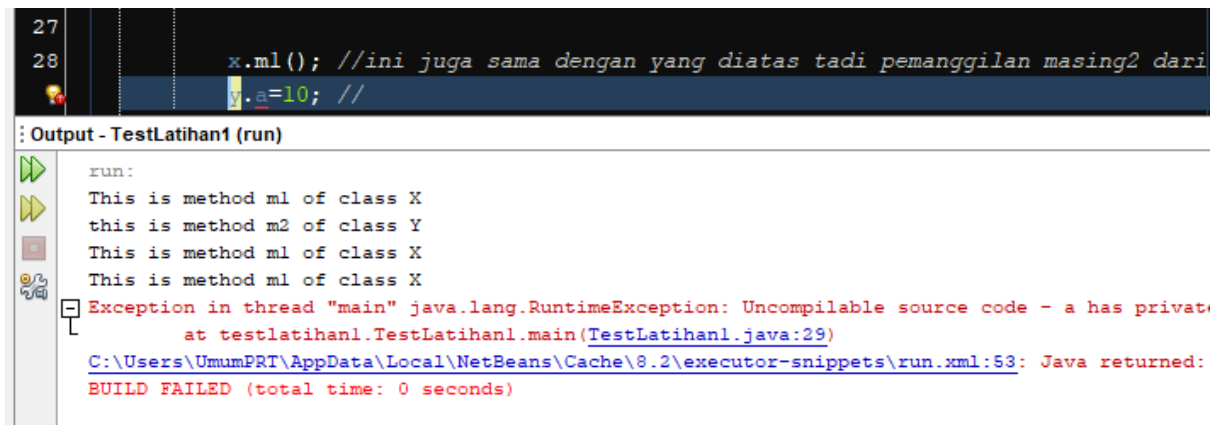
```
9      * @author Anang Misbakhul Khoir
10     * NIM 09040620047
11     */
12     public class TestLatihan1 {
13
14         /**
15          * @param args the command line arguments
16          */
17         public static void main(String[] args) {
18             X x = new X();
19             Y y = new Y();
20
21             y.m2(); //ini adalah pemanggilan jama' dari kelas anak y yang mana sudah
22
23             x.ml(); //ini adalah pemanggilan secara masing2 dari kelas induk
24             //y.ml(); //ini adalah pemanggilan secara masing2 dari kelas anak
25
26             //x = y; //parent pointing to object of child
27
28             x.ml(); //ini juga sama dengan yang diatas tadi pemanggilan masing2 dari
29             //y.a=10; //
```

Output - TestLatihan1 (run)

```
run:
This is method ml of class X
this is method m2 of class Y
This is method ml of class X
This is method ml of class X
BUILD SUCCESSFUL (total time: 0 seconds)
```

Dari percobaan kali ini adalah pemanggilan secara masing2 dari kelas induk

- c. Batalkan komentar pada baris # 10 . Simpan & Kompilasi kode.



```
27
28     x.ml(); //ini juga sama dengan yang diatas tadi pemanggilan masing2 dari
    y.a=10; //
```

Output - TestLatihan1 (run)

```
run:
This is method ml of class X
this is method m2 of class Y
This is method ml of class X
This is method ml of class X
Exception in thread "main" java.lang.RuntimeException: Uncompilable source code - a has private
    at testlatihan1.TestLatihan1.main(TestLatihan1.java:29)
C:\Users\UmmuMPT\AppData\Local\NetBeans\Cache\8.2\executor-snippets\run.xml:53: Java returned:
BUILD FAILED (total time: 0 seconds)
```

Dipercobaan kali ini adanya error karena belum adanya pendeklarasian value di class child

[Type text]

- d. Apakah ada Kesalahan = ? Ini karena sub-kelas tidak dapat mengakses anggota private dari kelas super, perbaiki kesalahan tersebut

Dengan cara merubah deklarasi private yang ada di class induk menjadi public dan maka hasilnya akan bisa dirun.

```
27
28     x.m1(); //ini juga sama dengan yang diatas tadi pemanggilan masing2 dari
29     y.a = 10;
30 }
```

Output - TestLatihan1 (run)

run:

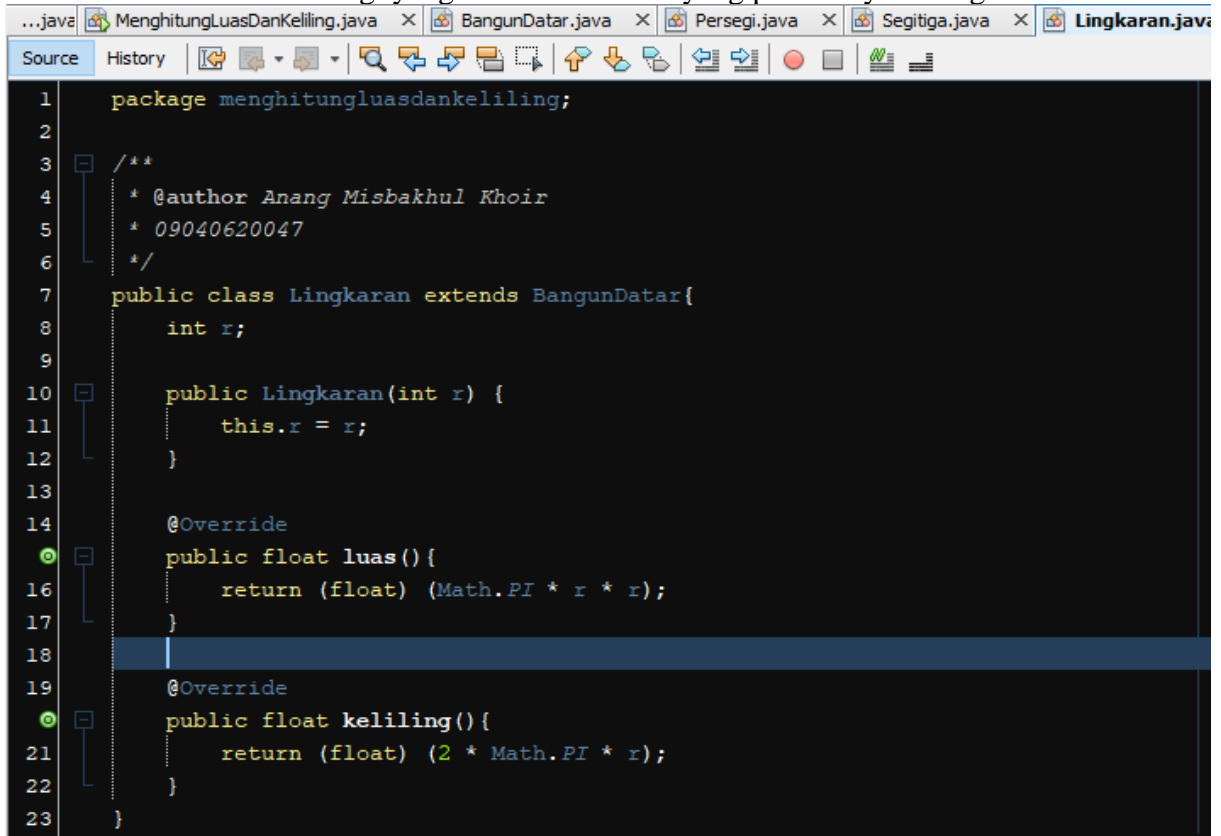
This is method m1 of class X
this is method m2 of class Y
This is method m1 of class X
This is method m1 of class Y
This is method m1 of class Y
BUILD SUCCESSFUL (total time: 0 seconds)

2. Buatlah program menghitung luas dan keliling bangun dengan menerapkan konsep overriding. Sertakan diagram class anda dan hasil keluaran programnya.

Pertama tama kita buat class BangunDatar seperti gambar dibawah ini yang mana tujuannya sebagai class induk nantinya.

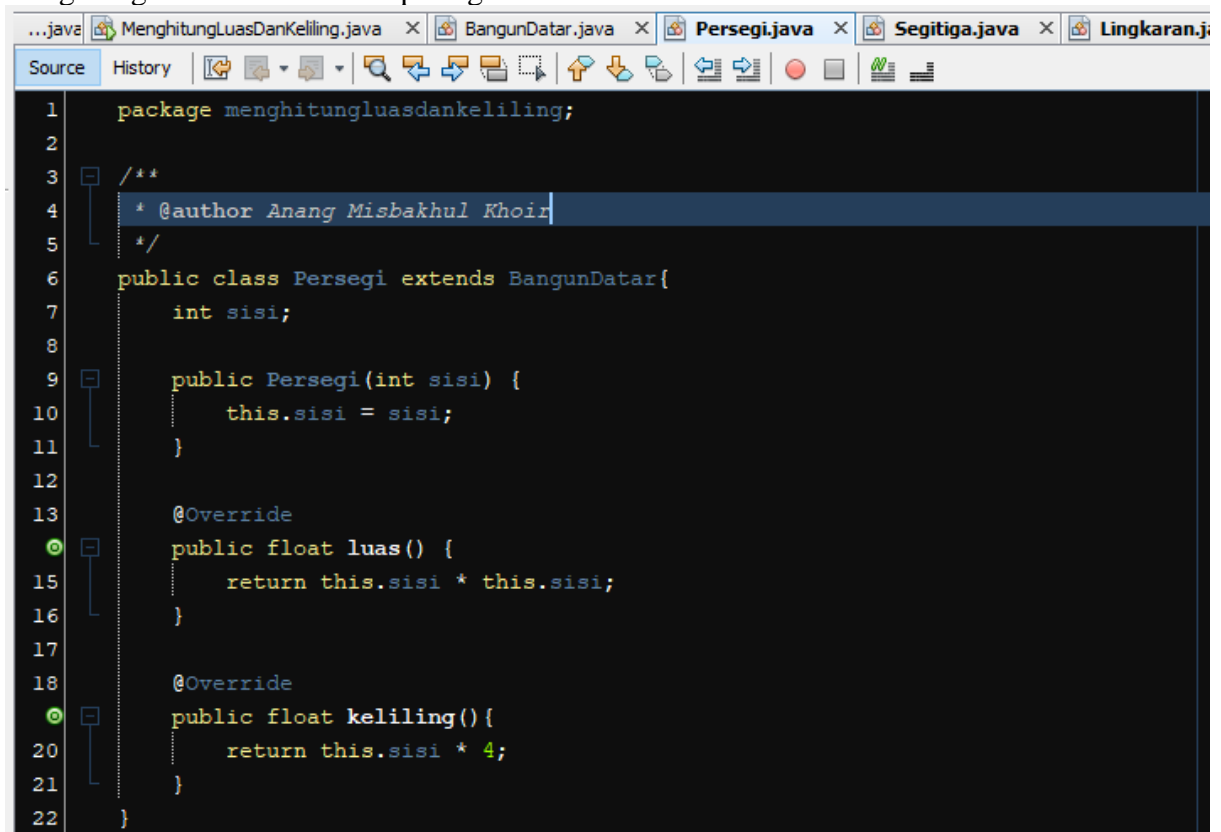
```
...ava MenghitungLuasDanKeliling.java x BangunDatar.java x Persegi.java x Segitiga.java x Lingkaran
Source History
1 package menghitungluasdankeliling;
2
3 /**
4  * @author Anang Misbakhul Khoir
5  */
6 public class BangunDatar {
7     float luas() {
8         System.out.println("Menghitung luas bangun datar");
9         return 0;
10    }
11
12    float keliling() {
13        System.out.println("Menghitung keliling bangun datar");
14        return 0;
15    }
16 }
```

Kemudian kita buat class lagi yang mana class child yang pertama yaitu lingkaran



```
1 package menghitungluasdankeliling;
2
3 /**
4  * @author Anang Misbakhul Khoir
5  * 09040620047
6  */
7 public class Lingkaran extends BangunDatar{
8     int r;
9
10    public Lingkaran(int r) {
11        this.r = r;
12    }
13
14    @Override
15    public float luas(){
16        return (float) (Math.PI * r * r);
17    }
18
19    @Override
20    public float keliling(){
21        return (float) (2 * Math.PI * r);
22    }
23 }
```

Yang ketiga ini kita buat class persegi.



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

[Type text]

Class segitiga.

```
...java MenghitungLuasDanKeliling.java x BangunDatar.java x Persegi.java x Segitiga.java x Lingkaran.ja
Source History
1 package menghitungluasdankeliling;
2
3 /**
4  * @author Anang Misbakhul Khoir
5  */
6
7 public class Segitiga extends BangunDatar{
8     int alas;
9     int tinggi;
10
11     public Segitiga(int alas, int tinggi) {
12         this.alas = alas;
13         this.tinggi = tinggi;
14     }
15
16
17 @Override
18 public float luas(){
19     return this.alas * this.tinggi;
20 }
21 }
```

Yang terakhir ini kita buat main class yang mana tujuan nya menjalankan semua class yang kita buat tadi.

```
...ava MenghitungLuasDanKeliling.java x BangunDatar.java x Persegi.java x Segitiga.java x Ling
Source History
1 package menghitungluasdankeliling;
2 /**
3  * @author Anang Misbakhul Khoir
4  */
5 public class MenghitungLuasDanKeliling {
6
7     public static void main(String[] args) {
8         BangunDatar bangunDatar = new BangunDatar();
9         Persegi persegi = new Persegi(4);
10        Segitiga segitiga = new Segitiga(6, 3);
11        Lingkaran lingkaran = new Lingkaran(50);
12
13        // memanggil method luas dan keliling
14        bangunDatar.luas();
15        bangunDatar.keliling();
16
17        System.out.println("Luas persegi: " + persegi.luas());
18        System.out.println("keliling persegi: " + persegi.keliling());
19        System.out.println("Luas segitiga: " + segitiga.luas());
20        System.out.println("Luas lingkaran: " + lingkaran.luas());
21        System.out.println("keliling lingkaran: " + lingkaran.keliling());
22    }
23 }
```

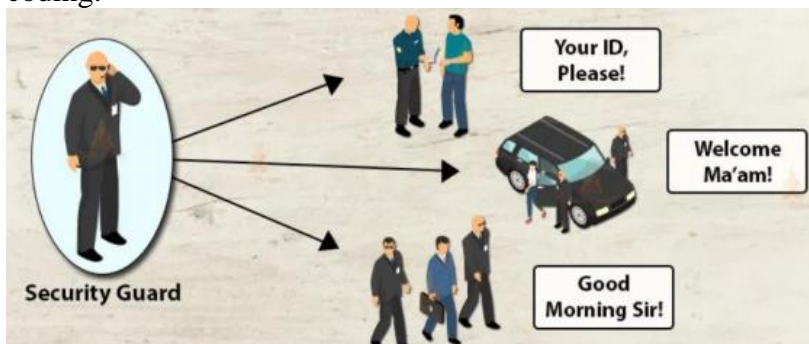
[Type text]

Dengan begitu kita bisa melihat hasil yang dikeluarkan seperti gambar dibawah ini.

: Output - MenghitungLuasDanKeliling (run)

```
run:
Menghitung luas bangun datar
Menghitung keliling bangun datar
Luas persegi: 16.0
keliling persegi: 16.0
Luas segitiga: 18.0
Luas lingkaran: 7853.9814
keliling lingkaran: 314.15927
BUILD SUCCESSFUL (total time: 0 seconds)
```

3. Cermati gambar dibawah ini. Jelaskan intepretasi anda, selanjutnya selesaikan dengan coding.



Hal pertama tama yaitu dengan membuat class parent dengan nama SecurityGuard. Seperti gambar dibawah ini.

```
...ava SecurityBeraksi.java x ScurityGuard.java x Scurity1.java x Security2.java x Security3.java x
Source History
1 package securityberaksi;
2
3 /**
4  * @author Anang Misbakhul Khoir
5  */
6
7 public class ScurityGuard {
8     String nama;
9
10    ScurityGuard(String nama){
11        this.nama = nama;
12    }
13    void display(){
14        System.out.println("\nNama\t : " + this.nama);
15    }
16 }
```

Kemudian membuat anak class pertama dengan nama security1

```
1 package securityberaksi;
2
3 /**
4  * @author Anang Misbakhul Khoir
5  */
6 public class Scurity1 extends ScurityGuard{
7     String type = "Scurity 1";
8
9     Scurity1(String nama){
10         super(nama);
11     }
12
13     @Override
14     void display(){
15         super.display();
16         System.out.println("Type\t : " +this.type);
17     }
18     void showoff(){
19         System.out.println("Saya SCURITY 1, Tunjukkan ID kamu !!!");
20     }
21 }
```

Kemudian selanjutnya juga membuat class child lagi yaitu dengan nama security2

```
1 package securityberaksi;
2
3 /**
4  * @author Anang Misbakhul Khoir
5  */
6 public class Security2 extends ScurityGuard{
7     String type = "Scurity 2";
8
9     Security2(String nama){
10         super(nama);
11     }
12
13     @Override
14     void display(){
15         super.display();
16         System.out.println("Type\t : " +this.type);
17     }
18     void showoff(){
19         System.out.println("Saya SCURITY 2, Selamat Datang Pak/Bu...");
20     }
21 }
```


Dan class security3

```
...ava SecurityBeraksi.java x ScurityGuard.java x Scurity1.java x Security2.java x Security3.java
Source History
1 package securityberaksi;
2
3 /**
4  * @author Anang Misbakhul Khoir
5  */
6 public class Security3 extends ScurityGuard{
7     String type = "Scurity 3";
8
9     Security3(String nama){
10         super(nama);
11     }
12
13     @Override
14     void display(){
15         super.display();
16         System.out.println("Type\t : " +this.type);
17     }
18     void showoff(){
19         System.out.println("Saya SCURITY 3, Selamat Pagi Pak/Bu...");
20     }
21 }
```

Kemudian membuat main class yang bertujuan untuk menjalankan class class yang sudah kita buat tadi, berikut main classnya.

```
...ava SecurityBeraksi.java x ScurityGuard.java x Scurity1.java x Security2.java x Security3.jav
Source History
1 package securityberaksi;
2
3 /**
4  * @author Anang Misbakhul Khoir
5  */
6 public class SecurityBeraksi {
7
8     public static void main(String[] args) {
9
10         ScurityGuard Satpam = new ScurityGuard("Muhammad Ali");
11         Scurity1 Satpaml = new Scurity1("Osama");
12         Security2 Satpam2 = new Security2("Shidiq");
13         Security3 Satpam3 = new Security3("Umar");
14         Satpam.display();
15
16         Satpaml.display();
17         Satpaml.showoff();
18         Satpam2.display();
19         Satpam2.showoff();
20         Satpam3.display();
21         Satpam3.showoff();
22     }
```

[Type text]

```

22
23      //Polymorphic
24      ScurityGuard Penjagal = new Scurity1("Rianto");
25      Penjagal.display();
26
27
28      ScurityGuard Penjaga2 = new Security2("Agus");
29      Penjaga2.display();
30
31      ScurityGuard Penjaga3 = new Security3("Brudin");
32      Penjaga3.display();
33
34  }
35  }

```

Berikut adalah keluaran yang dihasilkan dari main class yang kita buat.

```

: Output - SecurityBeraksi (run)
run:
Nama      : Muhammad Ali
Nama      : Osama
Type      : Scurity 1
Saya SCURITY 1, Tunjukkan ID kamu !!!

Nama      : Shidiq
Type      : Scurity 2
Saya SCURITY 2, Selamat Datang Pak/Bu...

Nama      : Umar
Type      : Scurity 3
Saya SCURITY 3, Selamat Pagi Pak/Bu...

Nama      : Rianto
Type      : Scurity 1

Nama      : Agus
Type      : Scurity 2

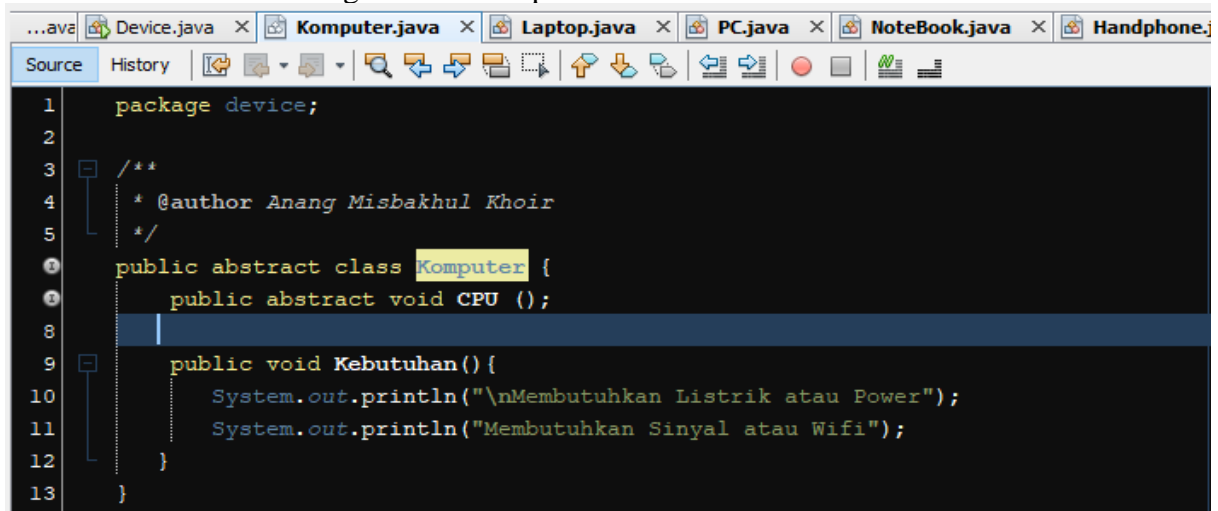
Nama      : Brudin
Type      : Scurity 3
BUILD SUCCESSFUL (total time: 0 seconds)

```

LATIHAN 2 CLASS DAN METHOD ABSTRACT

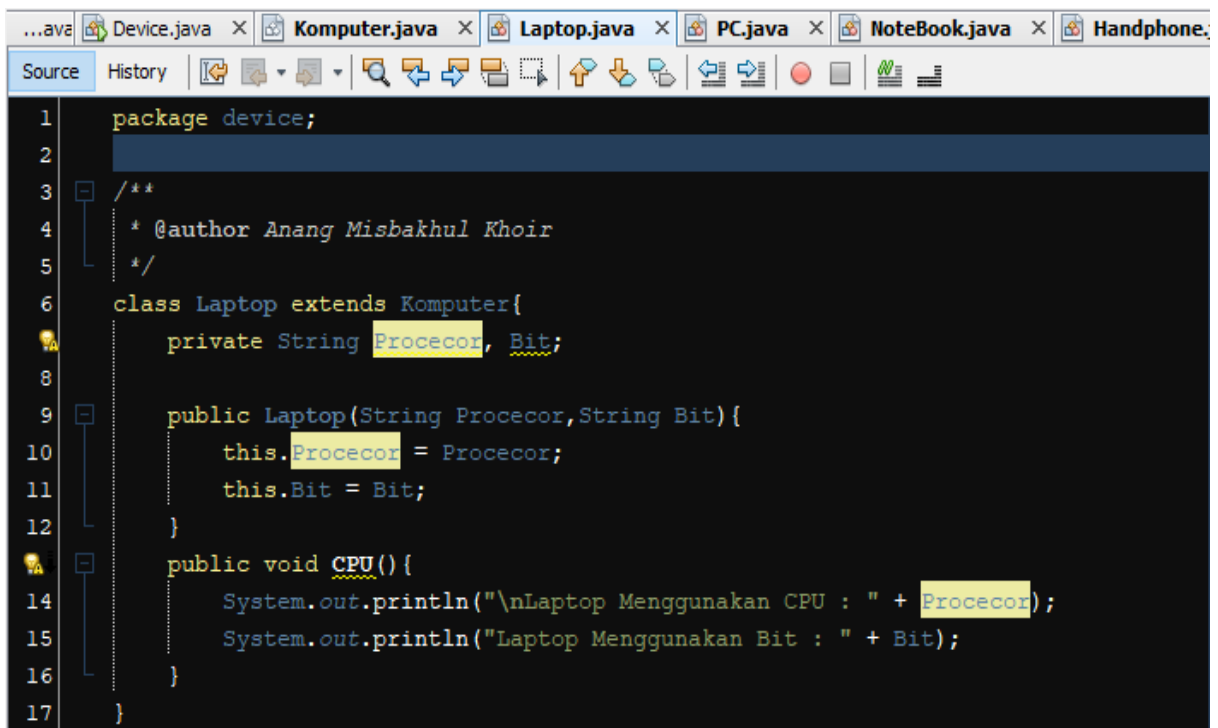
1. Rifa ingin membuat class yang terdiri dari berbagai jenis komputer seperti class laptop, class PC, class netbook, dan lain-lain. Seluruh class ini tentunya memiliki sifat-sifat komputer, seperti memiliki spesifikasi, memiliki processor, dan membutuhkan listrik. Dalam implementasinya, Rifa bisa membuat seluruh class ini diturunkan dari class komputer. Agar lebih seragam, Rifa ingin seluruh class yang diturunkan dari class komputer, memiliki method yang 'pasti' ada dalam setiap class anak. Setiap komputer tentunya memiliki spesifikasi, sehingga Rifa ingin setiap class yang diturunkan dari class komputer memiliki method lihat_spec(). Bagaimana caranya 'memaksa' setiap class agar memiliki method lihat_spec()? Selesaikan kasus Rifa ini dalam bentuk source code lengkap dengan capture output dan diagram class nya.

Hal yang pertama perlu diingat adalah membuat class parent seperti gambar dibawah, class ini kita namakan dengan class Komputer.



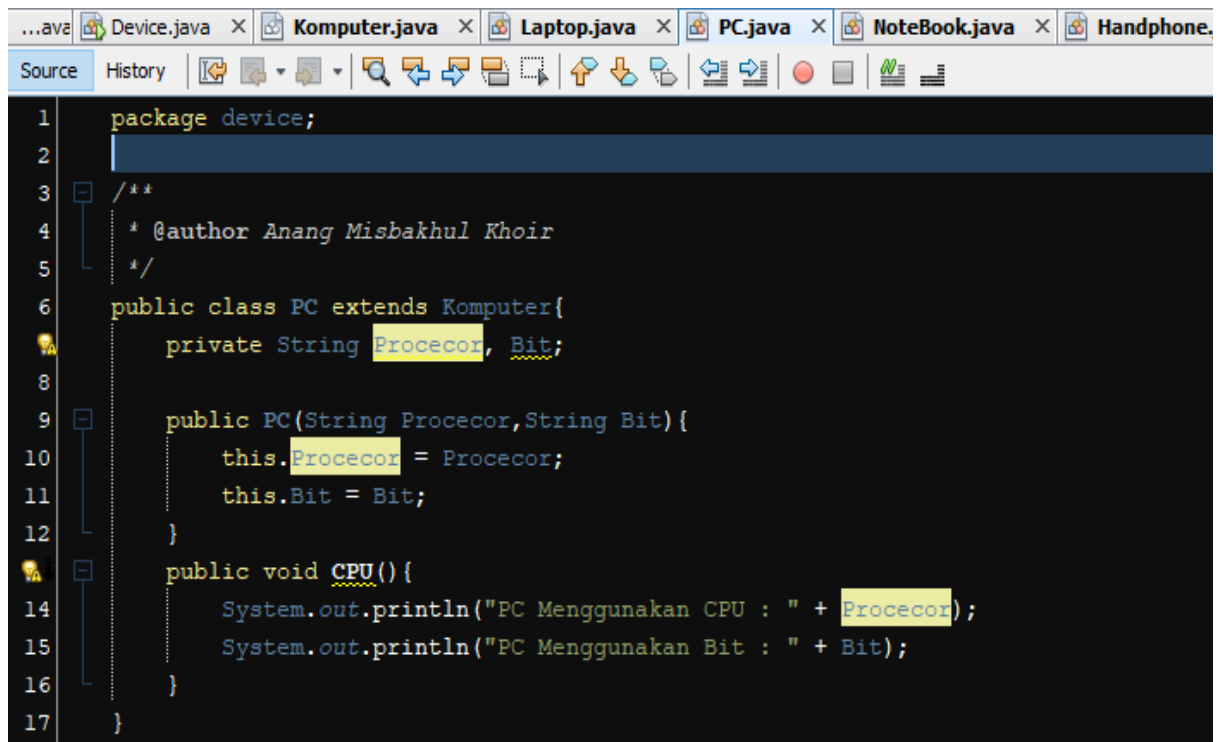
```
1 package device;
2
3 /**
4  * @author Anang Misbakhul Khoir
5  */
6 public abstract class Komputer {
7     public abstract void CPU ();
8
9     public void Kebutuhan() {
10         System.out.println("\nMembutuhkan Listrik atau Power");
11         System.out.println("Membutuhkan Sinyal atau Wifi");
12     }
13 }
```

Kemudian yang kedua selanjutnya adalah dengan membuat class child yaitu dengan nama laptop, seperti gambar dibawah ini.



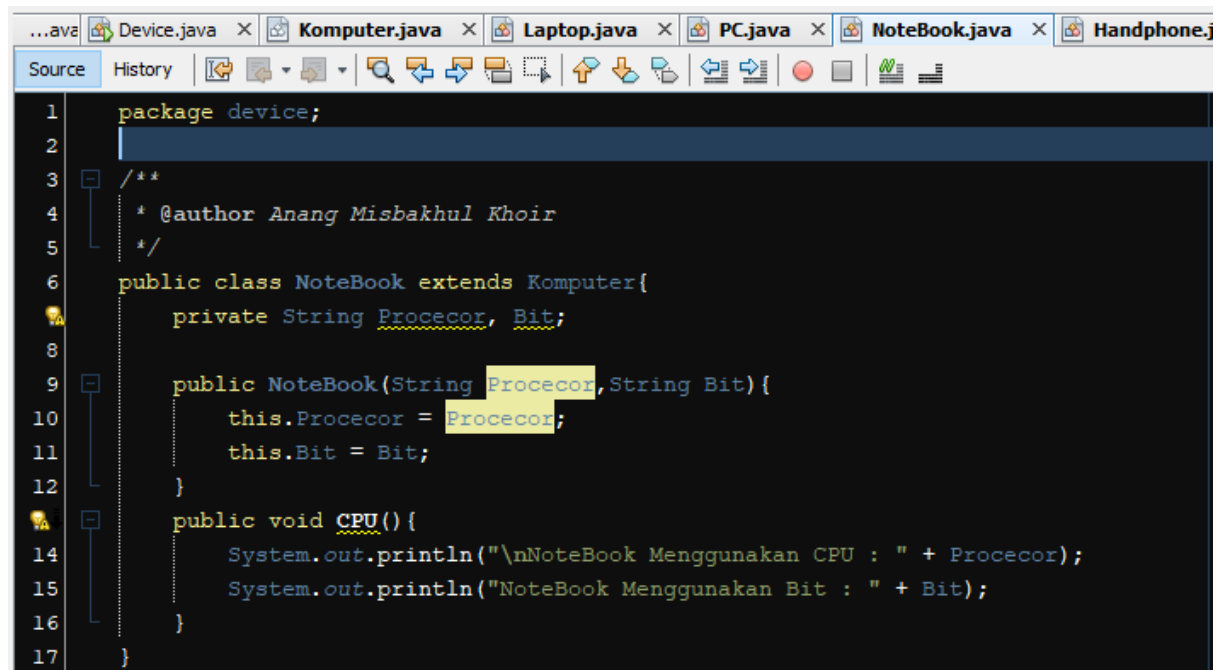
```
1 package device;
2
3 /**
4  * @author Anang Misbakhul Khoir
5  */
6 class Laptop extends Komputer{
7     private String Procecor, Bit;
8
9     public Laptop(String Procecor, String Bit) {
10         this.Procecor = Procecor;
11         this.Bit = Bit;
12     }
13
14     public void CPU() {
15         System.out.println("\nLaptop Menggunakan CPU : " + Procecor);
16         System.out.println("Laptop Menggunakan Bit : " + Bit);
17     }
18 }
```

Membuat class PC



```
1 package device;
2
3 /**
4  * @author Anang Misbakhul Khoir
5  */
6 public class PC extends Komputer{
7     private String Procetor, Bit;
8
9     public PC(String Procetor,String Bit){
10         this.Procetor = Procetor;
11         this.Bit = Bit;
12     }
13
14     public void CPU(){
15         System.out.println("PC Menggunakan CPU : " + Procetor);
16         System.out.println("PC Menggunakan Bit : " + Bit);
17     }
18 }
```

Kemudian membuat class notebook .



```
1 package device;
2
3 /**
4  * @author Anang Misbakhul Khoir
5  */
6 public class Notebook extends Komputer{
7     private String Procetor, Bit;
8
9     public Notebook(String Procetor,String Bit){
10         this.Procetor = Procetor;
11         this.Bit = Bit;
12     }
13
14     public void CPU(){
15         System.out.println("\nNotebook Menggunakan CPU : " + Procetor);
16         System.out.println("Notebook Menggunakan Bit : " + Bit);
17     }
18 }
```

[Type text]

Dan class child yang terakhir yaitu class handphone.

```
...ava Device.java x Komputer.java x Laptop.java x PC.java x Notebook.java x Handphone.java
Source History
1 package device;
2
3 /**
4  * @author Anang Misbakhul Khoir
5  */
6 public class Handphone extends Komputer{
7     private String Procecor, Bit;
8
9     public Handphone(String Procecor, String Bit){
10         this.Procecor = Procecor;
11         this.Bit = Bit;
12     }
13     public void CPU(){
14         System.out.println("PC Menggunakan CPU : " + Procecor);
15         System.out.println("PC Menggunakan Bit : " + Bit);
16     }
17 }
```

Jika sudah jangan lupa untuk membuat main class yang mana tujuannya untuk menjalankan class yang sudah kita buat tadi.

```
...ava Device.java x Komputer.java x Laptop.java x PC.java x Notebook.java x Handphone.java
Source History
1 package device;
2 /**
3  * @author Anang Misbakhul Khoir
4  */
5 public class Device {
6     public void cekKomputer(Komputer DevKom){
7         DevKom.CPU();
8         DevKom.Kebutuhan();
9     }
10
11     public static void main(String[] args) {
12         Device tDevice = new Device();
13
14         tDevice.cekKomputer(new Laptop("Intel CORE i9", "64 Bit"));
15
16         System.out.println("=====");
17         tDevice.cekKomputer(new PC("AMD Radeon", "86 Bit"));
18
19         System.out.println("=====");
20         tDevice.cekKomputer(new Notebook("CORE i7", "64 Bit"));
21
22         System.out.println("=====");
23         tDevice.cekKomputer(new Handphone("SnapDragon", "64 Bit"));
24     }
25 }
```

[Type text]

Berikut adalah hasil keluarannya.

```
Output - Device (run)

run:

Laptop Menggunakan CPU : Intel CORE i9
Laptop Menggunakan Bit : 64 Bit

Membutuhkan Listrik atau Power
Membutuhkan Sinyal atau Wifi
=====
PC Menggunakan CPU : AMD Radeon
PC Menggunakan Bit : 86 Bit

Membutuhkan Listrik atau Power
Membutuhkan Sinyal atau Wifi
=====

NoteBook Menggunakan CPU : CORE i7
NoteBook Menggunakan Bit : 64 Bit

Membutuhkan Listrik atau Power
Membutuhkan Sinyal atau Wifi
=====
PC Menggunakan CPU : SnapDragon
PC Menggunakan Bit : 64 Bit

Membutuhkan Listrik atau Power
Membutuhkan Sinyal atau Wifi
BUILD SUCCESSFUL (total time: 0 seconds)
```

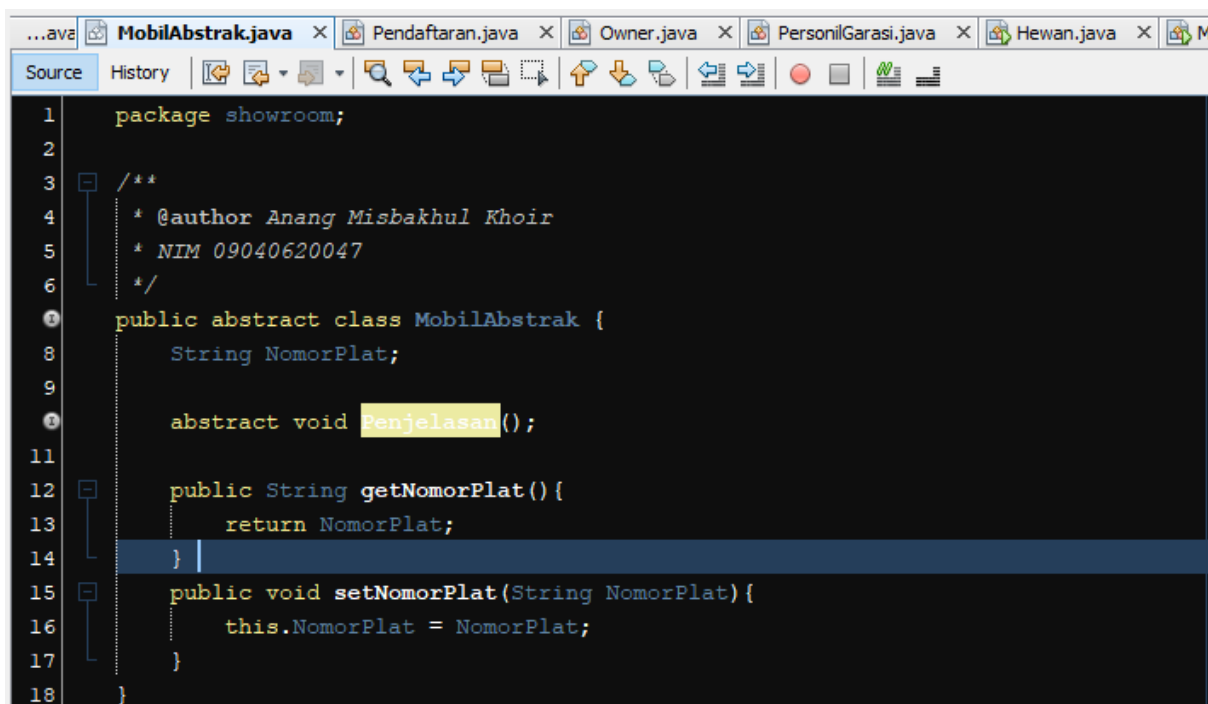
2. Cermati gambar berikut.



Di sini, Anda dapat melihat bahwa Pemilik tertarik dengan detail seperti deskripsi Mobil, riwayat servis, dll; Personil Garasi tertarik dengan detail seperti Lisensi, deskripsi kerja, tagihan, pemilik, dll; dan Kantor Pendaftaran tertarik pada detail seperti nomor identifikasi kendaraan, pemilik saat ini, plat nomor, dll. Ini berarti setiap aplikasi mengidentifikasi detail yang penting untuknya. Abstraksi dapat dilihat sebagai teknik menyaring detail yang tidak perlu dari suatu objek sehingga hanya ada karakteristik berguna yang mendefinisikannya. Abstraksi berfokus pada perilaku yang dirasakan entitas. Gunakan imajinasi anda untuk melengkapi gambar diatas sehingga teknik abstraksi bisa diterapkan. Selesaikan dalam bentuk diagram class, source code dan outputnya.

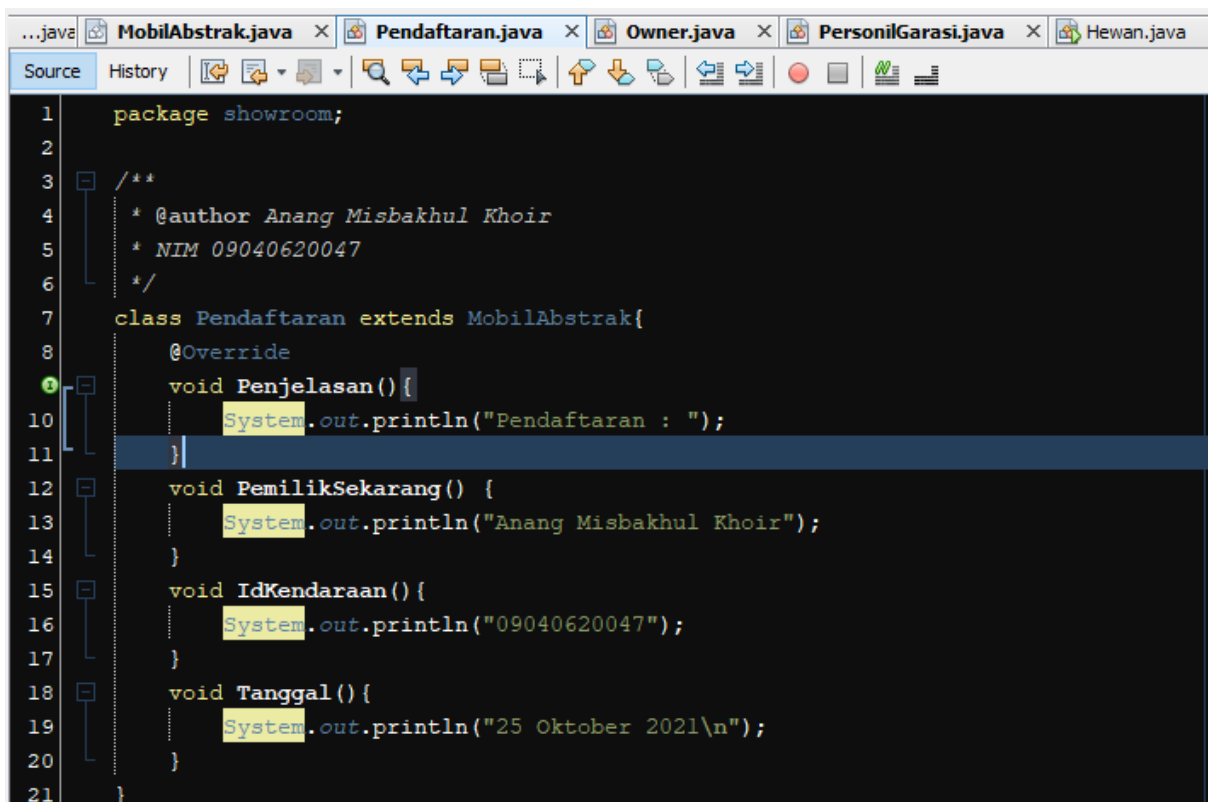
[Type text]

Yang pertama kita buat dulu class parent abstract dengan nama MobilAbstrak.



```
1 package showroom;
2
3 /**
4  * @author Anang Misbakhul Khoir
5  * NIM 09040620047
6  */
7 public abstract class MobilAbstrak {
8     String NomorPlat;
9
10    abstract void Penjelasan();
11
12    public String getNomorPlat() {
13        return NomorPlat;
14    }
15
16    public void setNomorPlat(String NomorPlat) {
17        this.NomorPlat = NomorPlat;
18    }
19 }
```

Kemudian membuat class child pendaftaran.



```
1 package showroom;
2
3 /**
4  * @author Anang Misbakhul Khoir
5  * NIM 09040620047
6  */
7 class Pendaftaran extends MobilAbstrak {
8     @Override
9     void Penjelasan() {
10        System.out.println("Pendaftaran : ");
11    }
12
13    void PemilikSekarang() {
14        System.out.println("Anang Misbakhul Khoir");
15    }
16
17    void IdKendaraan() {
18        System.out.println("09040620047");
19    }
20
21    void Tanggal() {
22        System.out.println("25 Oktober 2021\n");
23    }
24 }
```

[Type text]

Class Owner atau pemilik pertama

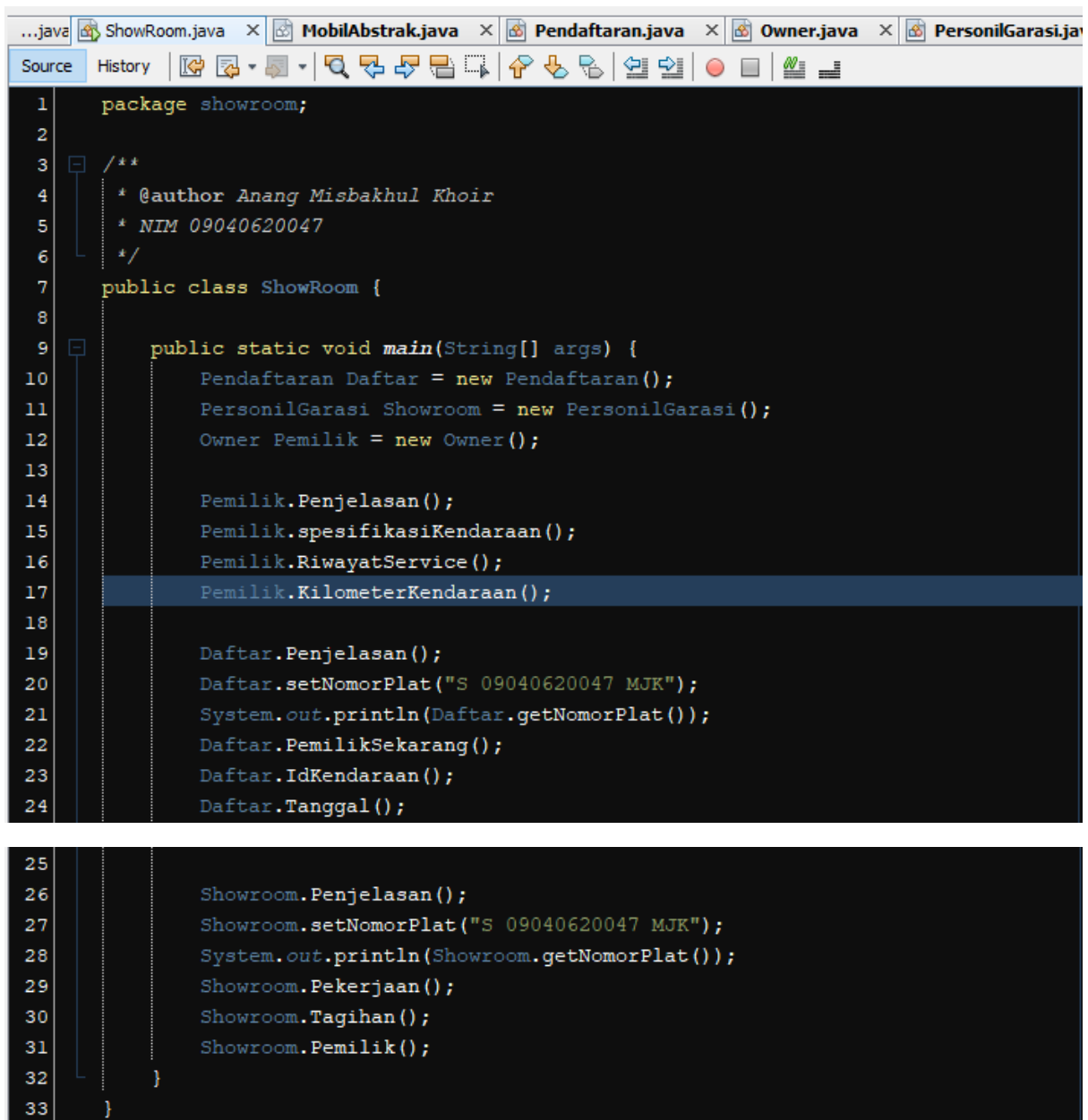
```
...java MobilAbstrak.java x Pendaftaran.java x Owner.java x PersonilGarasi.java x Hewan.java
Source History
1 package showroom;
2
3 /**
4  * @author Anang Misbakhul Khoir
5  * NIM 09040620047
6  */
7 class Owner extends MobilAbstrak {
8     @Override
9     public void Penjelasan(){
10         System.out.println("Pemilik Pertama : ");
11     }
12     void spesifikasiKendaraan(){
13         System.out.println("Mobil ini tahun 2010");
14     }
15     void RiwayatService(){
16         System.out.println("Pernah Turun mesin dan sudah pembaruan");
17     }
18     void KilometerKendaraan(){
19         System.out.println("90.000Km\n");
20     }
21 }
```

Class personil garasi

```
...java MobilAbstrak.java x Pendaftaran.java x Owner.java x PersonilGarasi.java x Hewan.java
Source History
1 package showroom;
2
3 /**
4  * @author Anang Misbakhul Khoir
5  * NIM 09040620047
6  */
7 class PersonilGarasi extends MobilAbstrak {
8     @Override
9     void Penjelasan() {
10         System.out.println("Garage : ");
11     }
12     void Pekerjaan(){
13         System.out.println("Tune UP");
14     }
15     void Tagihan(){
16         System.out.println("Rp.20.000.000");
17     }
18     void Pemilik(){
19         System.out.println("firts Owner\n");
20     }
21 }
```

[Type text]

Dan yang terakhir yaitu main classnya jangan lupa untuk menjalankan semuanya tadi.



```
1 package showroom;
2
3 /**
4  * @author Anang Misbakhul Khoir
5  * NIM 09040620047
6  */
7 public class ShowRoom {
8
9     public static void main(String[] args) {
10         Pendaftaran Daftar = new Pendaftaran();
11         PersonilGarasi Showroom = new PersonilGarasi();
12         Owner Pemilik = new Owner();
13
14         Pemilik.Penjelasan();
15         Pemilik.spesifikasiKendaraan();
16         Pemilik.RiwayatService();
17         Pemilik.KilometerKendaraan();
18
19         Daftar.Penjelasan();
20         Daftar.setNomorPlat("S 09040620047 MJK");
21         System.out.println(Daftar.getNomorPlat());
22         Daftar.PemilikSekarang();
23         Daftar.IdKendaraan();
24         Daftar.Tanggal();
25
26         Showroom.Penjelasan();
27         Showroom.setNomorPlat("S 09040620047 MJK");
28         System.out.println(Showroom.getNomorPlat());
29         Showroom.Pekerjaan();
30         Showroom.Tagihan();
31         Showroom.Pemilik();
32     }
33 }
```

[Type text]

Dan hasil outputnya bisa kita lihat capture dibawah ini.

```
Output - ShowRoom (run)

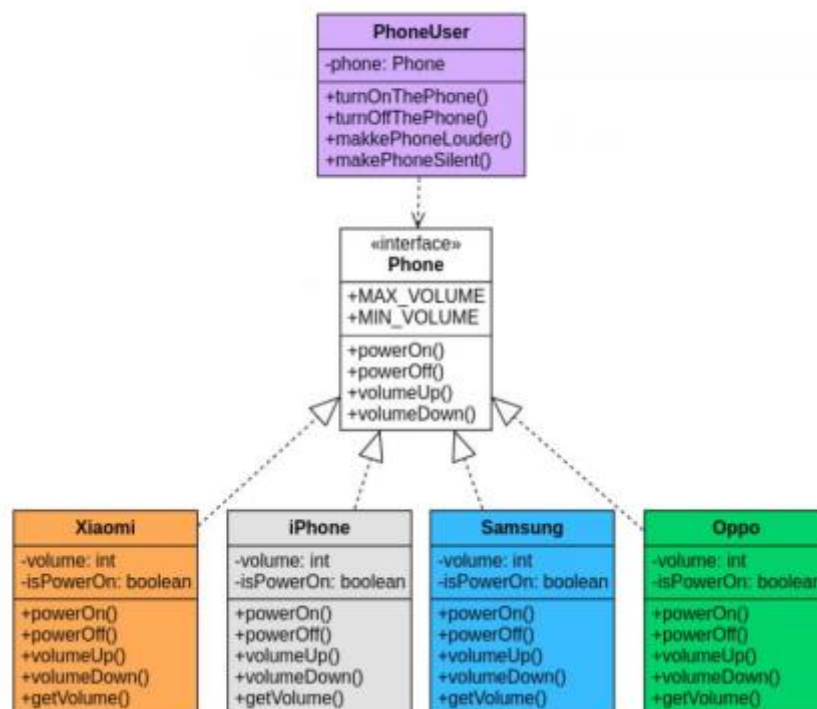
run:
Pemilik Pertama :
Mobil ini tahun 2010
Pernah Turun mesin dan sudah pembaruan
90.000Km

Pendaftaran :
S 09040620047 MJK
Anang Misbakhul Khoir
09040620047
25 Oktober 2021

Garage :
S 09040620047 MJK
Tune UP
Rp.20.000.000
firts Owner

BUILD SUCCESSFUL (total time: 0 seconds)
```

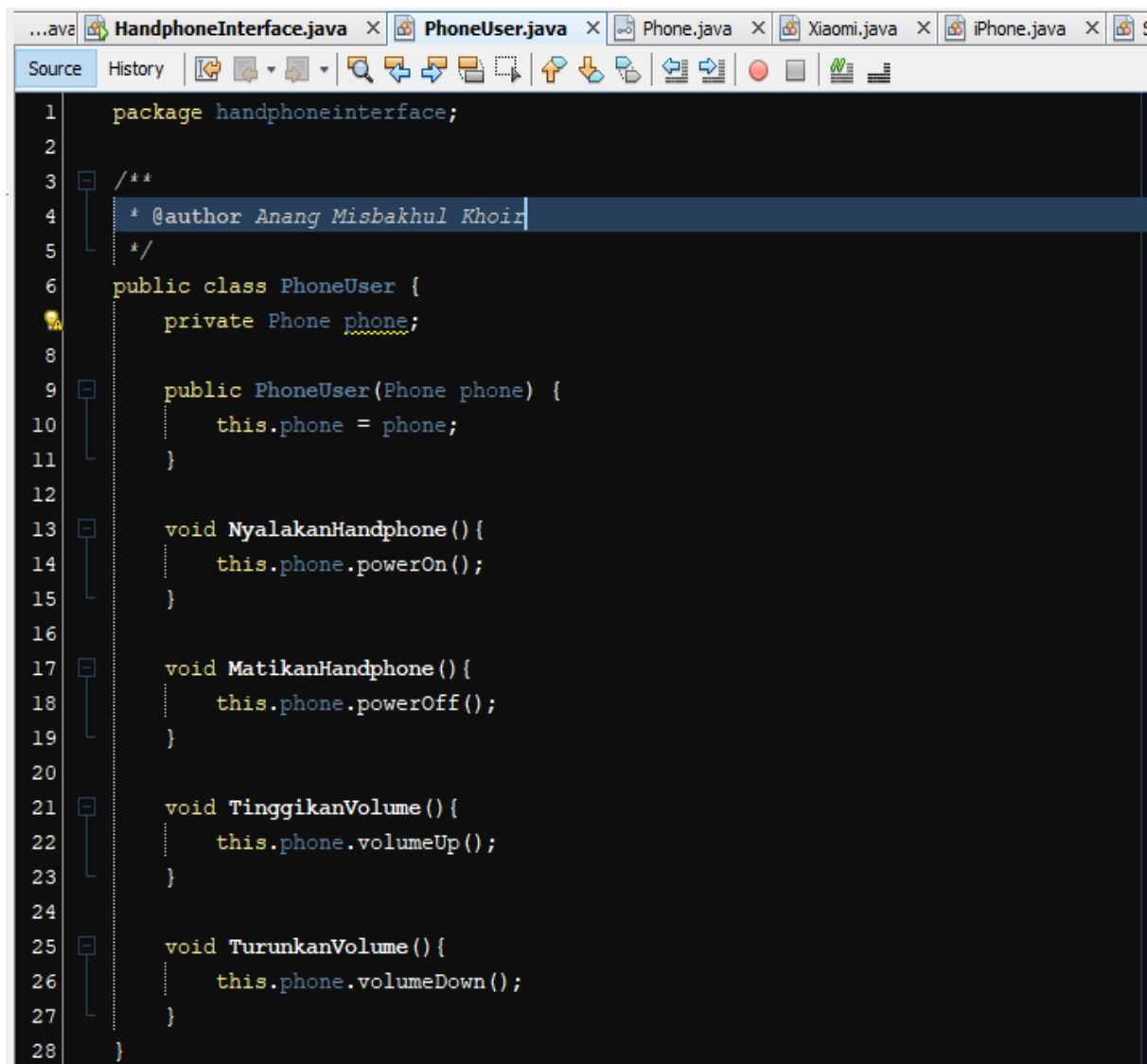
LATIHAN 3 INTERFACE



1. Selesaikan diagram diatas menjadi program yang utuh dan bisa running well
2. Apakah interface bisa memiliki sub interface? Jika bisa bagaimana implementasinya. Jika tidak bisa bagaimana argumentasinya

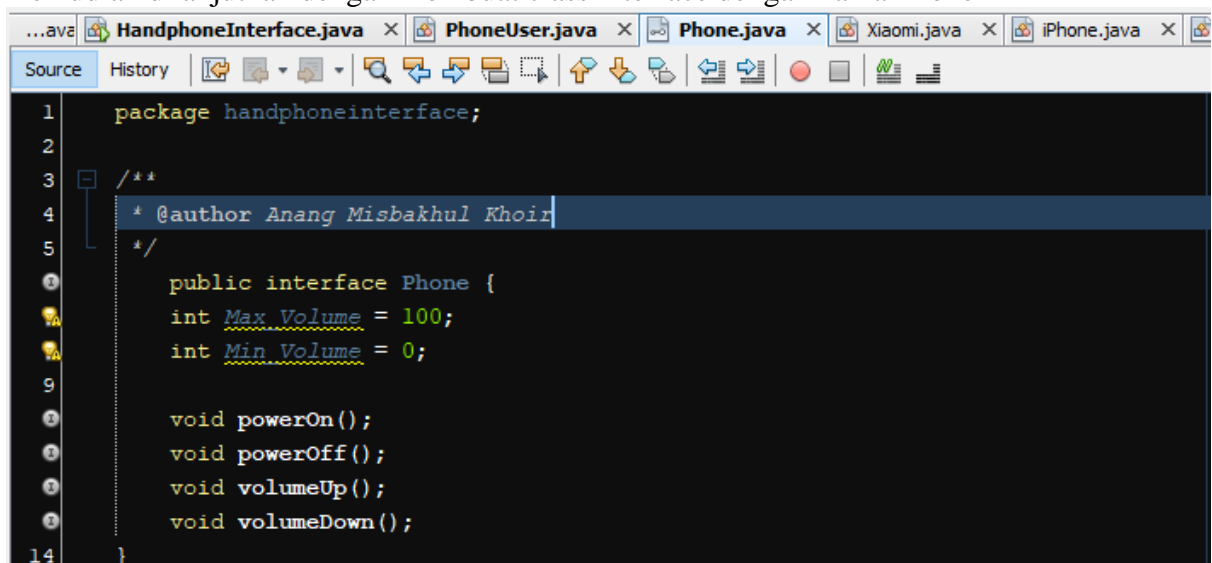
[Type text]

a. Pertama kita buat terlebih dahulu yaitu class PhoneUser.



```
1 package handphoneinterface;
2
3 /**
4  * @author Anang Misbakhul Khoir
5  */
6 public class PhoneUser {
7     private Phone phone;
8
9     public PhoneUser(Phone phone) {
10         this.phone = phone;
11     }
12
13     void NyalakanHandphone() {
14         this.phone.powerOn();
15     }
16
17     void MatikanHandphone() {
18         this.phone.powerOff();
19     }
20
21     void TinggikanVolume() {
22         this.phone.volumeUp();
23     }
24
25     void TurunkanVolume() {
26         this.phone.volumeDown();
27     }
28 }
```

Kemudian dilanjutkan dengan membuat class interface dengan nama Phone



```
1 package handphoneinterface;
2
3 /**
4  * @author Anang Misbakhul Khoir
5  */
6 public interface Phone {
7     int Max_Volume = 100;
8     int Min_Volume = 0;
9
10     void powerOn();
11     void powerOff();
12     void volumeUp();
13     void volumeDown();
14 }
```

[Type text]

Selanjutnya kita buat yaitu class dari Xiaomi

```
HandphoneInterface.java x PhoneUser.java x Phone.java x Xiaomi.java x iPhone.java x
ce History

package handphoneinterface;

/**
 * @author Anang Misbakhul Khoir
 */
public class Xiaomi implements Phone {

    private int volume;
    private boolean isPowerOn;

    public Xiaomi() {
        // set volume awal
        this.volume = 70;
    }

    @Override
    public void powerOn() {
        isPowerOn = true;
        System.out.println("\nHandphone Menyala...");
        System.out.println("WELCOME IN XIAOMI");
        System.out.println("Android version 11\n");
    }

    @Override
    public void powerOff() {
        isPowerOn = false;
        System.out.println("XIAOMI SHUTDOWN");
    }

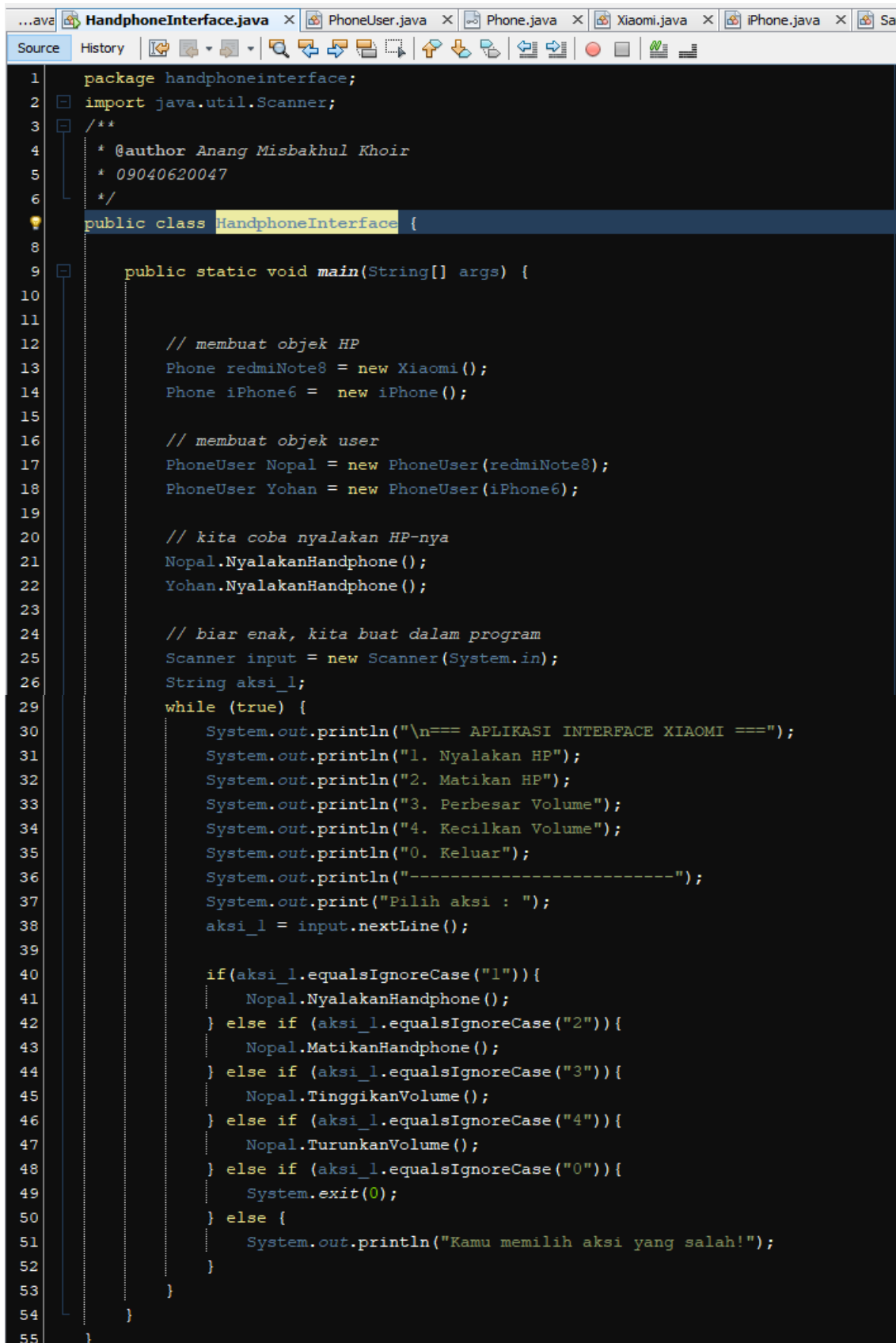
    @Override
    public void volumeUp() {
        if (isPowerOn) {
            if (this.volume == Max_Volume) {
                System.out.println("Volume FULL!!!");
                System.out.println("sudah " + this.getVolume() + "%");
            } else {
                this.volume += 10;
                System.out.println("Volume sekarang: " + this.getVolume());
            }
        } else {
            System.out.println("Nyalakan Dulu HP-nya!!!");
        }
    }

    @Override
    public void volumeDown() {
        if (isPowerOn) {
            if (this.volume == Min_Volume) {
                System.out.println("Volume = 0%");
            } else {
                this.volume -= 10;
                System.out.println("Volume sekarang: " + this.getVolume());
            }
        } else {
            System.out.println("Nyalakan Dulu HP-nya!!!");
        }
    }

    public int getVolume() {
        return this.volume;
    }
}
```

[Type text]

Selanjutnya yaitu membuat main class yang mana bertujuan untuk menjalankan class yang sudah kita buat diatas, berikut main classnya.



```
1 package handphoneinterface;
2 import java.util.Scanner;
3 /**
4  * @author Anang Misbakhul Khoir
5  * 09040620047
6  */
7 public class HandphoneInterface {
8
9     public static void main(String[] args) {
10
11         // membuat objek HP
12         Phone redmiNote8 = new Xiaomi();
13         Phone iPhone6 = new iPhone();
14
15         // membuat objek user
16         PhoneUser Nopal = new PhoneUser(redmiNote8);
17         PhoneUser Yohan = new PhoneUser(iPhone6);
18
19         // kita coba nyalakan HP-nya
20         Nopal.NyalakanHandphone();
21         Yohan.NyalakanHandphone();
22
23         // biar enak, kita buat dalam program
24         Scanner input = new Scanner(System.in);
25         String aksi_l;
26
27         while (true) {
28             System.out.println("\n=== APLIKASI INTERFACE XIAOMI ===");
29             System.out.println("1. Nyalakan HP");
30             System.out.println("2. Matikan HP");
31             System.out.println("3. Perbesar Volume");
32             System.out.println("4. Kecilkan Volume");
33             System.out.println("0. Keluar");
34             System.out.println("-----");
35             System.out.print("Pilih aksi : ");
36             aksi_l = input.nextLine();
37
38             if(aksi_l.equalsIgnoreCase("1")){
39                 Nopal.NyalakanHandphone();
40             } else if (aksi_l.equalsIgnoreCase("2")){
41                 Nopal.MatikanHandphone();
42             } else if (aksi_l.equalsIgnoreCase("3")){
43                 Nopal.TinggikanVolume();
44             } else if (aksi_l.equalsIgnoreCase("4")){
45                 Nopal.TurunkanVolume();
46             } else if (aksi_l.equalsIgnoreCase("0")){
47                 System.exit(0);
48             } else {
49                 System.out.println("Kamu memilih aksi yang salah!");
50             }
51         }
52     }
53 }
54
55 }
```

[Type text]

Berikut adalah output yang dikeluarkan yang mana kita tinggal memilih aksi yang nomor berapa.

Output - HandphoneInterface (run)

```
run:
Handphone Menyala...
WELCOME IN XIAOMI
Android version 11

Handphone menyala...
--WELCOME IPHONE--
Version Software 14.5

=== APLIKASI INTERFACE XIAOMI ===
1. Nyalakan HP
2. Matikan HP
3. Perbesar Volume
4. Kecilkan Volume
0. Keluar
-----
Pilih aksi :
```

- b. Untuk pembuatan sub interface apakah bisa memiliki sebuah sub interface dari argument saya kemungkinan tidak bisa karena class parent hanya bisa memiliki satu kecuali pembuatannya class parent lagi baru bisa.

KESIMPULAN

Dari kesimpulan diatas semuanya adalah Tujuan dari method overloading yaitu memudahkan penggunaan atau pemanggilan method dengan fungsionalitas yang mirip. Abstract class digunakan di dalam inheritance (pewarisan class) untuk 'memaksakan' implementasi method yang sama bagi seluruh class yang diturunkan dari abstract class. Abstract class digunakan untuk membuat struktur logika penurunan di dalam pemrograman objek. Sedangkan Interface untuk memfasilitasi multiple inheritance dalam java, memungkinkan untuk memisahkan definisi metode dari hirarki pewarisan.