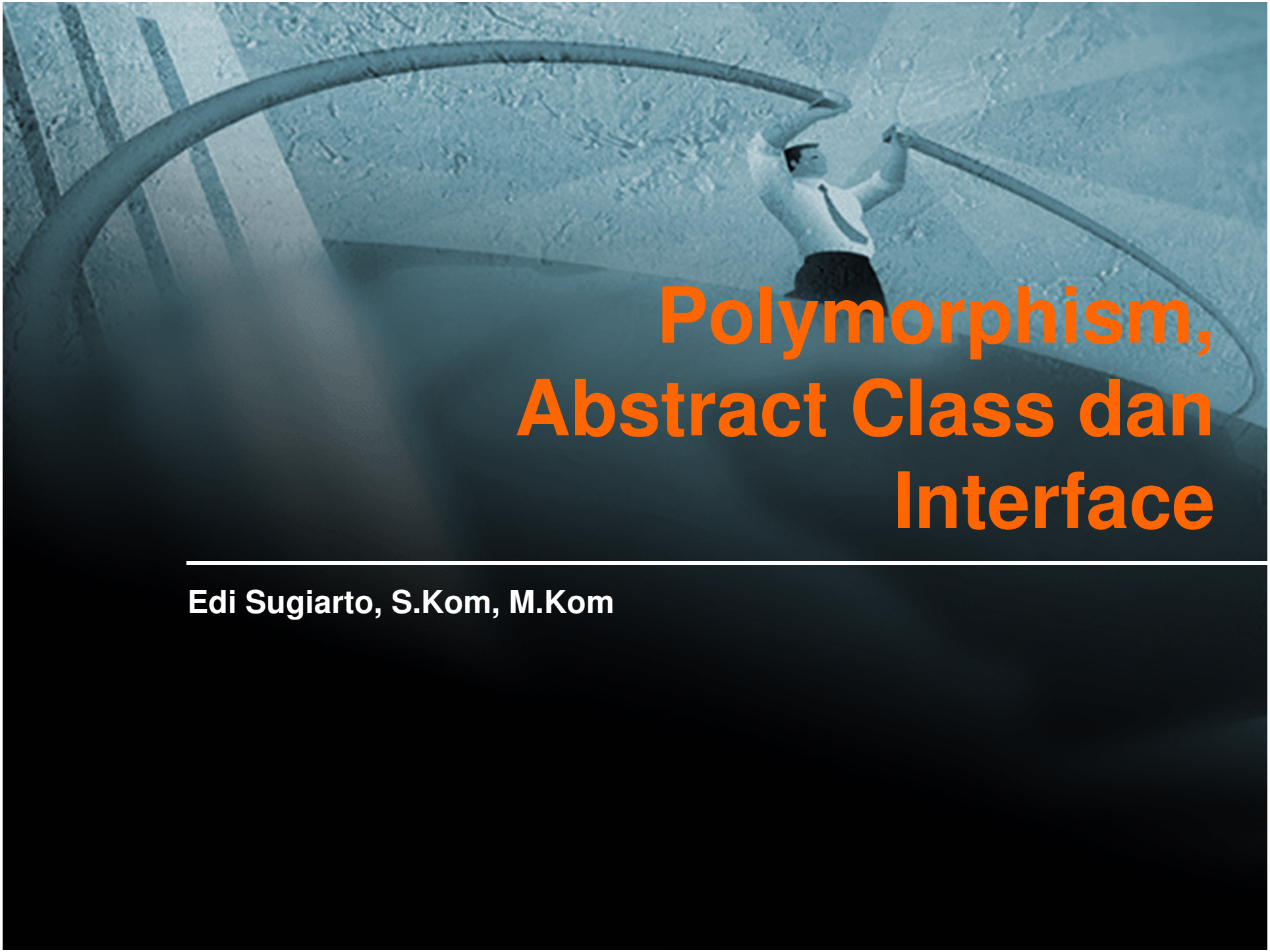


Polymorphism, Abstract Class dan Interface

Edi Sugiarto, S.Kom, M.Kom

Pendahuluan

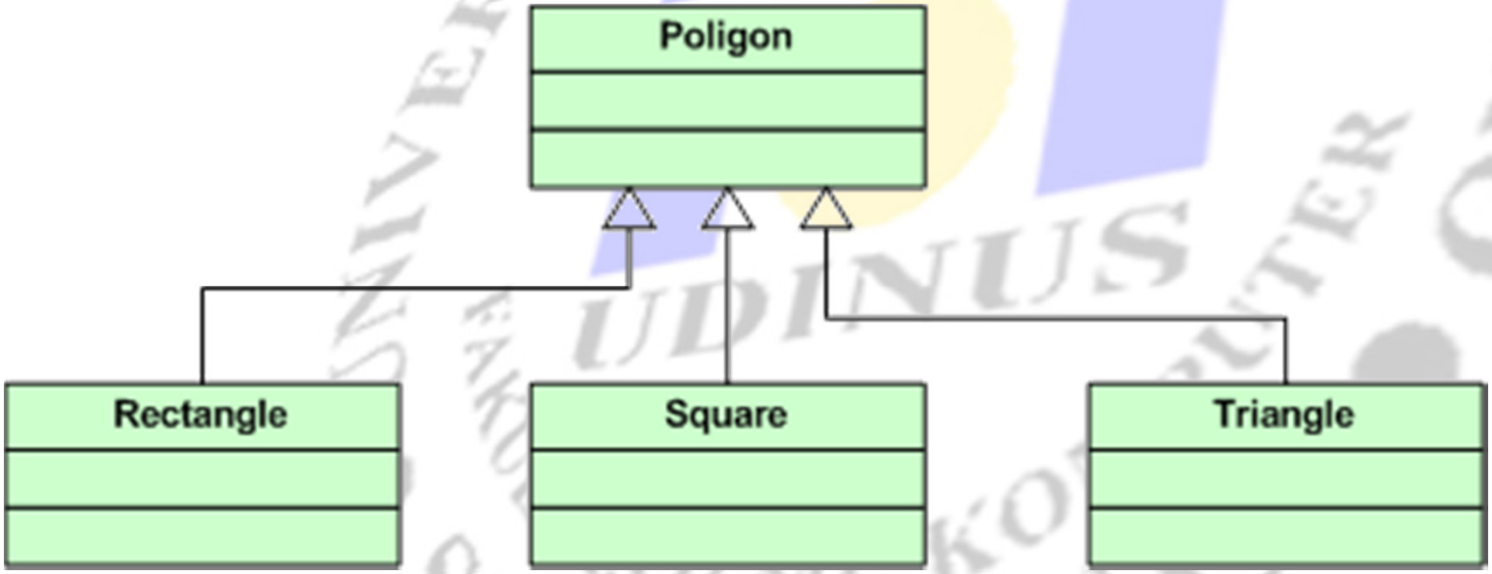
- Dalam pengembangan berorientasi objek yang dimaksud polymorphism atau biasa disebut polimorfisme adalah kemampuan dapat memperoleh berapa bentuk.
- Polimorfisme merupakan fitur penting pada bahasa pemrograman berorientasi objek
- Fitur ini adalah pembeda antara pemrograman berorientasi objek dengan bahasa pemrograman tradisional yg hanya sampai menerapkan tipe data abstrak.

Polimorfisme (Polymorphism)

- Polimorfisme adalah konsep dalam pemrograman berorientasi objek dimana objek dapat memiliki beberapa bentuk.
- Dua objek dikatakan sebagai polymorphic bila objek itu mempunyai antarmuka identik namun memiliki perilaku yang berbeda.
- Polimorfisme memungkinkan kita mengenali dan mengeksploitasi keserupaan-keserupaan diantara kelas-kelas yang berbeda.

edi.sugiarto@yahoo.com

Perhatikan hirarki kelas diagram berikut



edi.sugiarto@yahoo.com

Dalam Bahasa Java

```
public static void main(String[] args)
    {
        Poligon p = new Poligon();
        Square s = new Square();
        Rectangle r = new Rectangle();
        Triangle t = new Triangle();

        p=s;
        p=r;
        p=t;


    }
```

Instruksi ini menugaskan variabel yang dideklarasikan bertipe Polygon memuat objek Rectangle, kemudian Square, dan Triangle.

"p=s" disebut polymorphic assignment, dan "p" adalah polymorphic object.

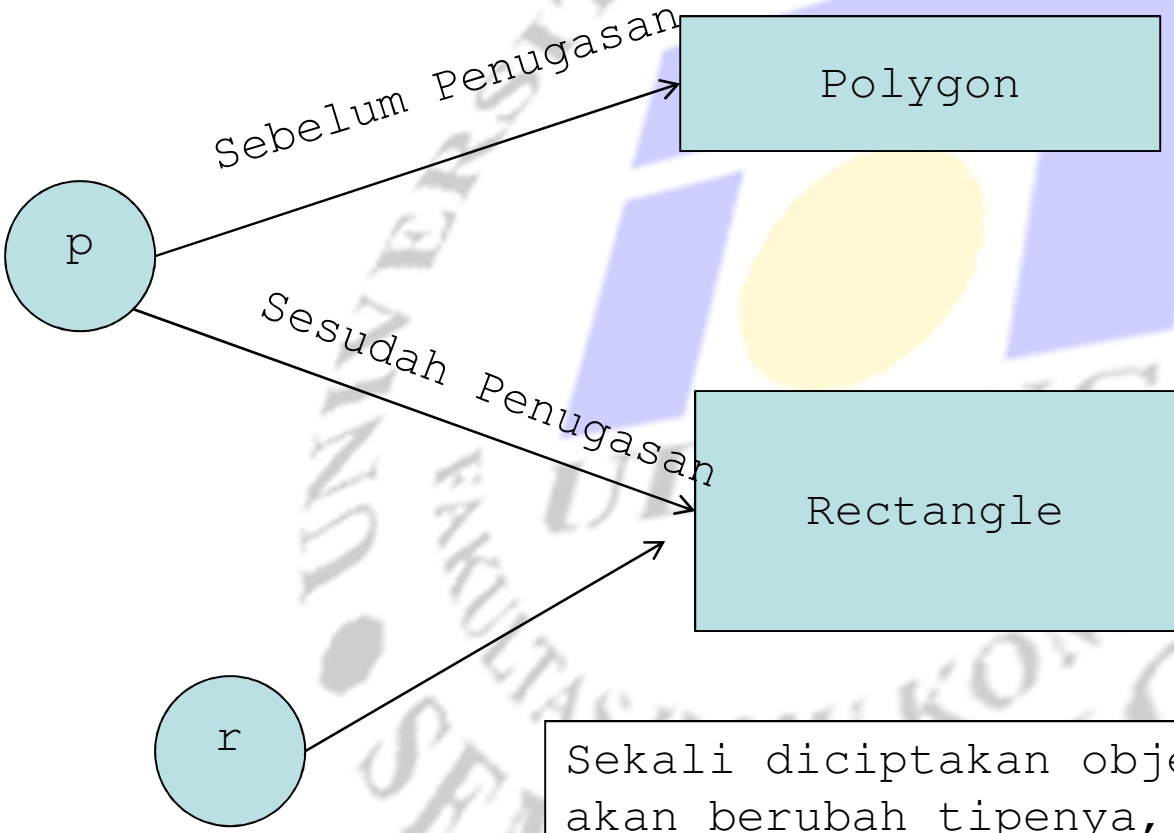
edi.sugiarto@yahoo.com

- Penugasan dimana tipe dari sumber (sisi kanan) berbeda dengan target (sisi kiri) berarti disebut penugasan *polymorphic (polymrphic assignment).*
- Objek p yang dapat muncul dari penugasan polymorphic disebut *objek polymorphic (polymorphic object)*

Mekanisme Polymorphism

- Polimorfisme merupakan fitur yang terjadi karena interaksi konsep pewarisan dan *dynamic binding*.
- Semua objek yang muncul adalah acuan (reference) ke objek, bukan objek itu sendiri. Sehingga penugasan seperti p=r, p=s artinya mengacukan ulang alamat r ke p, alamat s ke p.

Mekanisme Pengacuan



Sekali diciptakan objek tidak akan berubah tipenya, hanya acuan ke objek itu yang di arahkan ke objek-objek dengan tipe berbeda.

Batasan terhadap Poymorphism

- Pewarisan menuntut apakah suatu pencantolan polymorphism ke suatu acuan diijinkan.
- Dari contoh sebelumnya pengacuan seperti p=r, p=s diijinkan karena semuanya memiliki tipe sumber yang merupakan turunan dari kelas target, artinya tipe sumber memenuhi conform thd kelas target.
- Tapi tidak untuk r=s karena s tidak memenuhi (conform) terhadap r.

Overriding dan Overloading

- Overriding
  - Terjadi apabila deklarasi method pada subkelas (subclass) sama (termasuk parameter) dengan method pada superkelas (superclass).
- Overloading
  - Yakni penggunaan satu nama untuk beberapa method yang berbeda (berbeda parameter).

Overriding

- Sebuah method dikatakan meng-override method di kelas induknya jika di dalam subclass didefinisikan method yang memiliki nama, tipe kembalian dan daftar argumen yang persis sama.
- Subclass dapat mengesampingkan method yang didefinisikan dalam superclass dengan menyediakan implementasi baru dari method tersebut

Contoh Overriding

```
import java.util.Date;
public class Employee {
        protected String name;
        protected double salary;
        protected Date birthDate;

        public String getDetails() {
            return "Name: " + name + "\n Salary: " + salary;
        }

}
```

Contoh Overriding

```
public class Manager extends Employee {
        protected String department;

        public String getDetails() {
            return  "Name: " + name + "\n" +
                    "Salary: " + salary + "\n" +
                    "Manager of: " + department;
        }

}
```

Method getDetails pada kelas Manager adalah method overriding dari kelas Employee

- Secara definitif kelas Manager mempunyai method yang bernama getDetails(), karena kelas Manager mewarisinya dari kelas Employee, tetapi kelas Manager mengubah implementasi dari method tersebut,
- dalam hal ini kelas Manager dapat dikatakan meng-Overriding method getDetails() dari kelas Employee.

Aturan Overriding

- Beberapa yang harus dipenuhi dalam overriding method sbb :
  - Daftar argumen pada method harus sama dengan method yang di override
  - Tipe kembalian method harus sama dengan method yang di override
  - Access control method tidak boleh lebih ketat (retrictive) daripada method yang mengoverride
  - Method yang meng-override tidak dapat melempar exception diluar exception yang dideklarasikan pada method yang di override.

Overloading

- Objek memungkinkan memiliki method yang sama namun dengan kegunaan dan fungsi yang berbeda.
- Perbedaan disini termasuk passing argumentnya dan nilai pengembalianya.
- Overloading artinya penggunaan satu nama untuk beberapa method yang berbeda (berbeda parameter).

- Overloading tidak hanya dapat dikenakan pada method, ketika kita membuat definisi kelas dengan lebih dari satu konstruktor sebenarnya kita telah menggunakan prinsip overloading

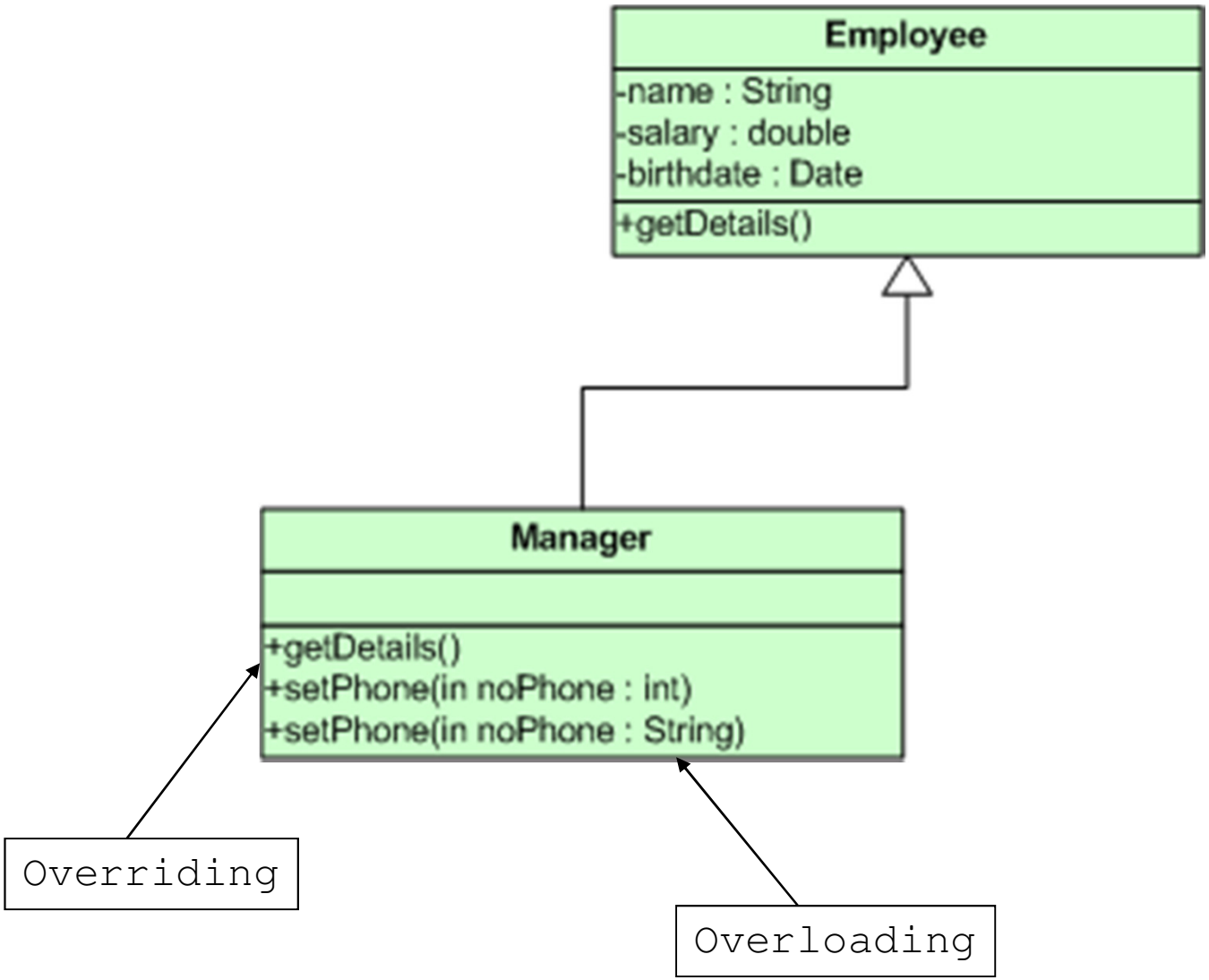
Contoh Overloading

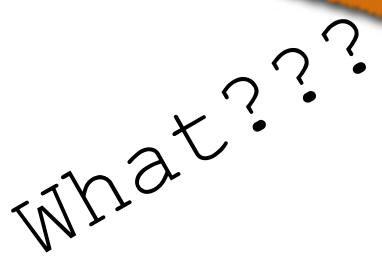
```
import java.util.Date;
public class Employee {
        protected String name;
        protected double salary;
        protected Date birthDate;

        public String getDetails() {
            return "Name: " + name + "\n Salary: " + salary;
        }

}
```

```
public class Manager extends Employee {
        protected String department;

        public String getDetails() {
            return  "Name: " + name + "\n" +
                    "Salary: " + salary + "\n" +
                    "Manager of: " + department;
        }

        void setPhone(int noPhone)
        {
            System.out.println("No Phone :"+noPhone);
        }
        void setPhone(String noPhone)
        {
            System.out.println("No Phone :"+noPhone);
        }

}
```

Method setPhone pada kelas Manager adalah method overloading.

- Dengan overloading maka suatu objek dapat memiliki satu method dengan fungsionalitas yang berbeda.


Employee
-name : String
-salary : double
-birthdate : Date
+getDetails()
Manager
+getDetails()
+setPhone(in noPhone : int)
+setPhone(in noPhone : String)
Overriding
Overloading



Ada Pertanyaan ?

edi.sugiarto@yahoo.com

Kelas Abstrak (Abstract Class)

edi.sugiarto@yahoo.com

Pendahuluan

- Kelas merupakan blue print, atau prototipe, yang mendefinisikan variabel dan metode-metode umum untuk semua objek dari jenis tertentu.
- Pada prinsipnya kelas ini adalah definisi statis yang berhubungan dengan pembuatnya (programmer) selanjutnya kelas akan diinstance menjadi objek dan selanjutnya objeklah yang berperan saat aplikasi runtime.

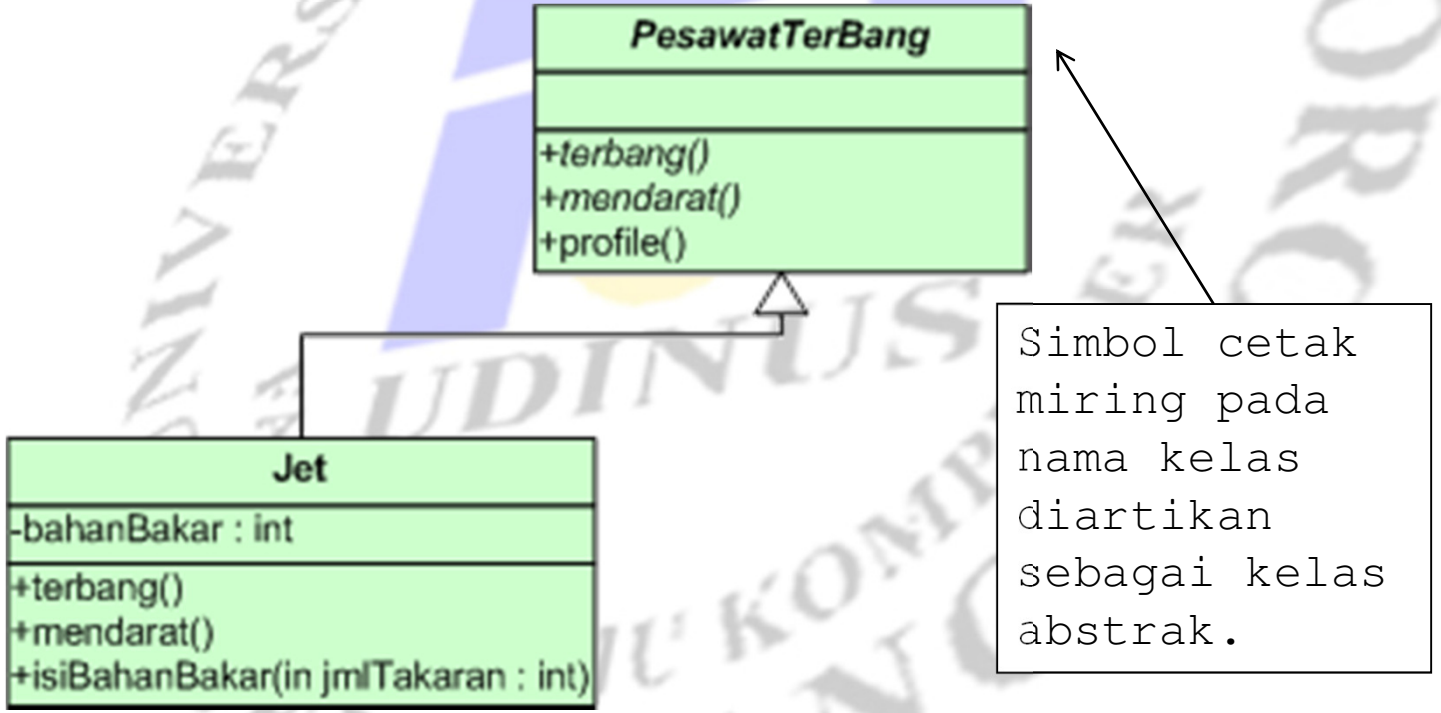
- Dalam beberapa masalah mungkin tidak diinginkan agar kelas tersebut diinstance menjadi objek dengan alasan karna kelas tersebut hanya digunakan untuk menyimpan definisi dan kelas lain lah yang melakukan perluasan terhadap definisi dari kelas tersebut.
- Jika hal tersebut yang akan dilakukan maka kelas tersebut harus terbentuk sebagai kelas abstrak.

Kelas Abstrak

- Sebuah kelas abstrak adalah kelas yang tidak dapat diinstansiasi menjadi sebuah objek.
- Dalam kelas abstrak dapat memiliki method abstrak atau bukan method abstrak
- Method abstrak adalah method yang memiliki definisi tanpa memiliki implementasi.
- Kelas abstrak atau method abstrak didefinisikan menggunakan keyword ***abstract*** pada bagian definisi

edi.sugiarto@yahoo.com

Contoh hierarki kelas abstrak



Contoh Kelas Abstrak

```
public abstract class PesawatTerbang {
    abstract void terbang();
    abstract void mendarat();
    void profile()
    {
        //statement
    }
}
```

Kelas abstrak dan method abstrak, method abstrak hanya memiliki definisi namun tidak memiliki implementasi

Kelas abstrak dapat memiliki method abstrak atau bukan method abstrak

```
public class Jet extends PesawatTerbang {

    int bahanbakar;
    void mendarat()
    {
        System.out.println("Mendarat");
    }
    void terbang()
    {
        System.out.println("Terbang");
    }
    void isiBahanBakar(int jmlTakaran)
    {
        this.bahanbakar=jmlTakaran;
    }
}
```

Kelas yang merupakan subkelas dari superkelas yang merupakan kelas abstrak harus meng-Override semua method abstrak dari kelas abstrak tersebut.

- Prinsipnya kelas abstrak tidak dapat diinstansiasi menjadi sebuah objek, oleh karena itu maka kelas abstrak harus melakukan perluasan dengan menurunkan attribut dan method ke kelas lain, dan kelas turunanyalah yang akan diinstansiasikan menjadi objek dengan membawa implementasi dari method yg dimiliki pada kelas abstrak yg merupakan superkelas.
- Oleh karena itu kelas abstrak setidaknya harus memiliki kelas turunan agar dapat digunakan.

Interface

edi.sugiarto@yahoo.com

Interface

- Interface bukanlah sebuah kelas, interface hanya perangkat yang mendefinisikan aturan perilaku (protocol of behaviour) yang dapat diimplementasikan oleh kelas manapun.
- Interface mendefinisikan satu set method tanpa menyediakan implementasinya.
- interface merupakan kumpulan dari method abstrak dan konstanta

Interface

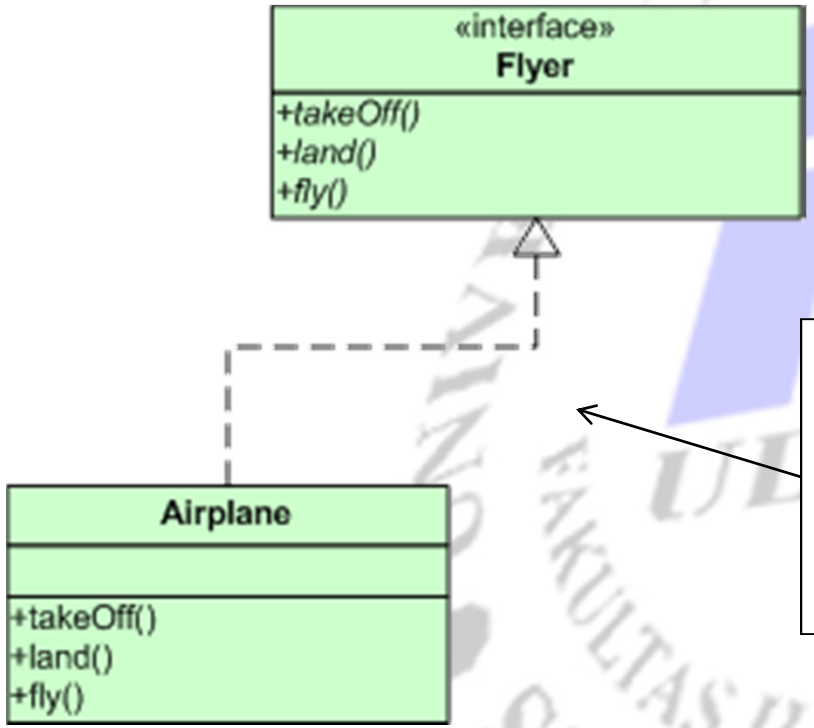
- Setiap kelas yang mengimplementasi sebuah interface terikat kontrak terhadap interface tsb untuk mengimplementasikan semua method yang ada pada interface
- Dengan kata lain, kelas tersebut terikat untuk mengimplementasikan perilaku tertentu yang tertulis dalam *interface*.

Kelas Abstrak vs Interface

- Interface memiliki kemiripan dengan kelas abstrak dimana keduanya memiliki method abstrak.
- Yang membedakan adalah :
  - Interface tidak memiliki implementasi method, sedang kelas abstrak boleh memiliki implementasi method
  - Kelas dapat mengimplementasikan lebih dari satu interface namun hanya bisa meng extends satu kelas abstrak

Deklarasi Interface

- Seandainya terdapat sekumpulan objek yang memiliki kemampuan yang sama yakni dapat terbang.
- Maka kita bisa membuat interface public yang memuat method-method yang dibutuhkan untuk membuat implementasi terhadap kemampuan tersebut.
- Contoh interface “flyer” memiliki method abstrak takeOff(), land(), dan fly()/



Panah dengan garis terputus menunjukkan bahwa kelas Airplane mengimplementasi interface Flyer

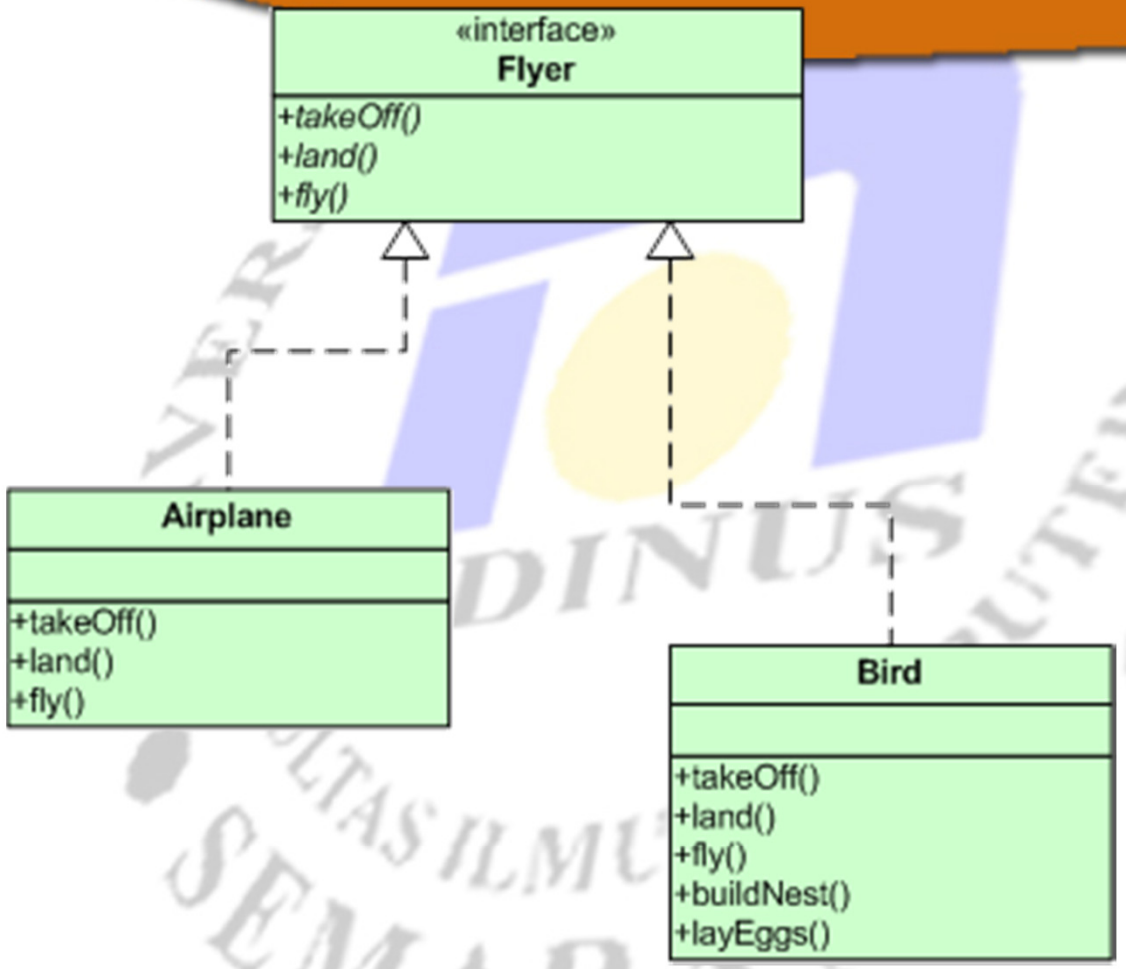
Contoh Source Code

```
public interface Flyer {
  public void takeOff();
  public void land();
  public void fly();
}
```

edi.sugiarto@yahoo.com

```
public class Airplane implements Flyer {
     public void takeOff() {
          //statement
     }
     public void land() {
          //statement
     }
     public void fly() {
          //statement
     }
}
```

- Suatu interface boleh diimplementasikan oleh lebih dari satu kelas
- Dan satu kelas boleh mengimplementasikan lebih dari satu interface.

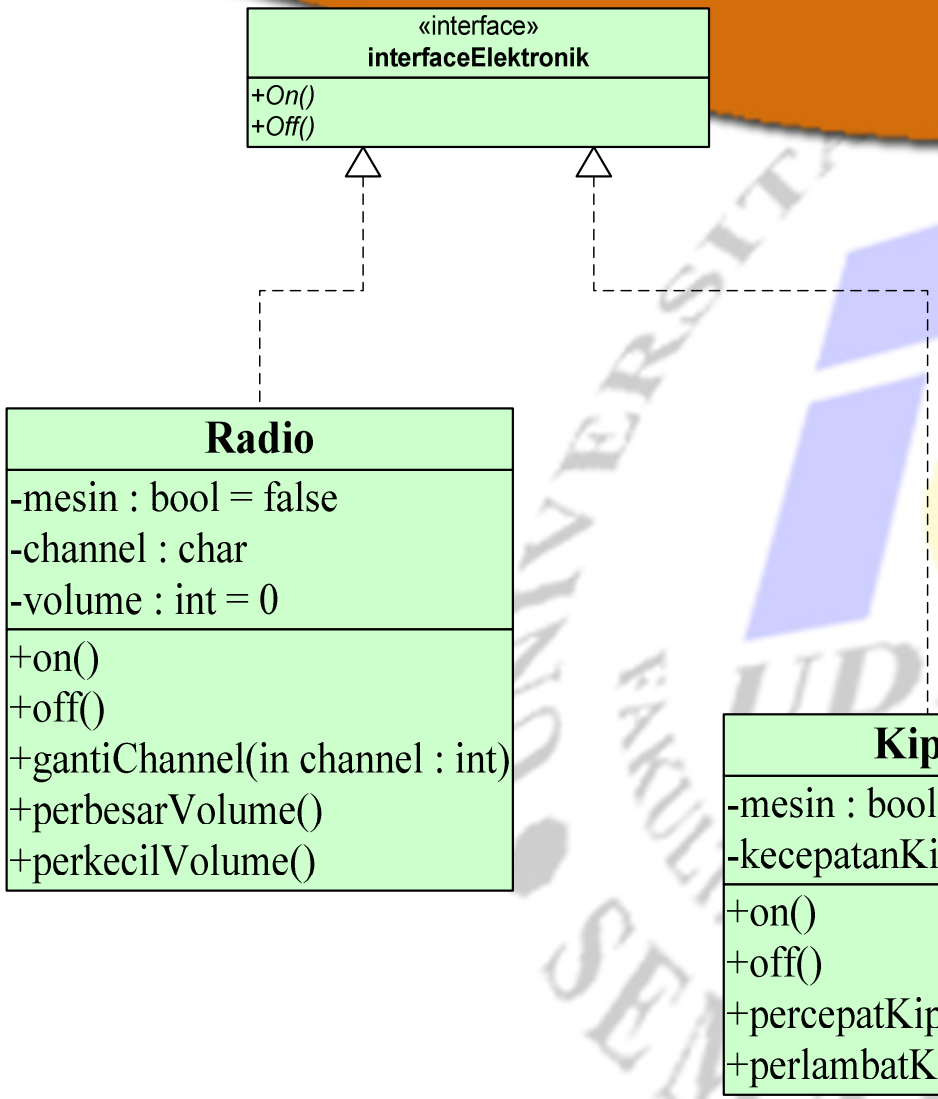

«interface»
Flyer
+takeOff()
+land()
+fly()
Airplane
+takeOff()
+land()
+fly()
Bird
+takeOff()
+land()
+fly()
+buildNest()
+layEggs()

Perbedaan Interface dan Kelas Abstrak

Interface	Kelas Abstrak
Tidak dapat membuat implementasi method	Dapat membuat implementasi method
Sebuah kelas dapat mengimplementasikan beberapa interface	Sebuah kelas hanya dapat meng-Extends satu superclass.

Mengapa menggunakan interface?

- Mendeklarasikan *method* yang akan diimplementasikan oleh satu atau beberapa kelas
- Menangkap kesamaan di antara beberapa kelas tanpa perlu memasukkannya dalam hirarki kelas.
- Mensimulasikan konsep pewarisan banyak kelas dengan mendeklarasikan kelas yang mengimplementasikan beberapa interface sekaligus



Kelas yang meng-Implements suatu interface maka kelas tersebut harus mengOverride method yang ada pada interface.

Dari contoh disamping, method on dan off merupakan method overriding dari interface interfaceElektronik

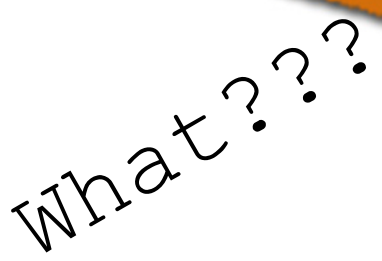
```
public interface interfaceElektronik
{
  public void on();
  public void off();
}
```

```
public class Radio implements
  interfaceElektronik
{
    boolean mesin=false;
    String[] channel={"Gajah Mada FM","Smart
  FM","Buana FM","DINUS FM"};
    int volume=0;

    public void on()
    {
        mesin=true;
    }
    public void off()
    {
        mesin=false;
    }
```

… Continue

```
public class Kipas implements
  interfaceElektronik {
    boolean mesin=false;
    int kecepatanKipas=0;

    public void on()
    {
        mesin=true;
    }
    public void off()
    {
        mesin=false;
    }
```

… Continue

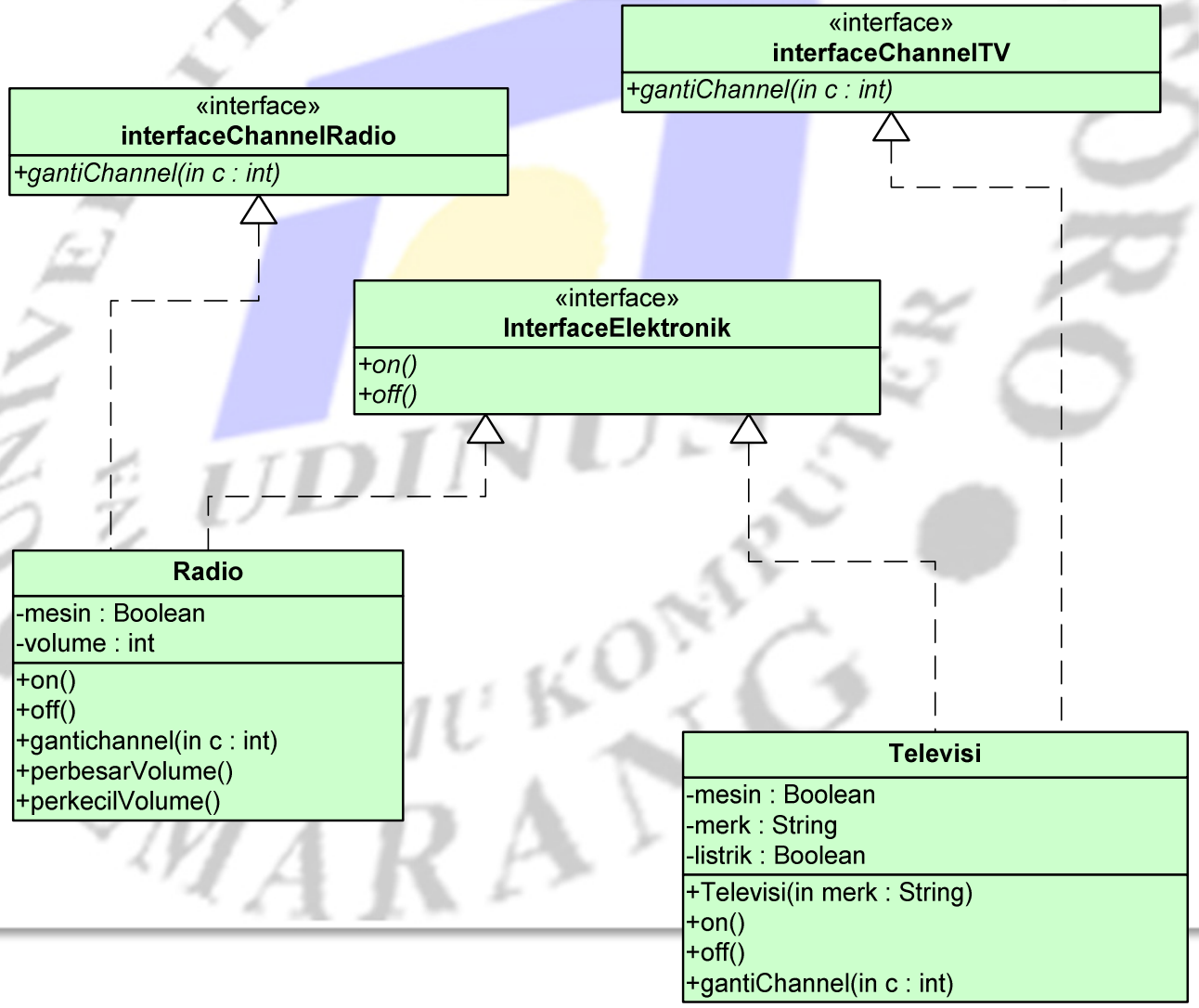


Ada Pertanyaan ?

edi.sugiarto@yahoo.com

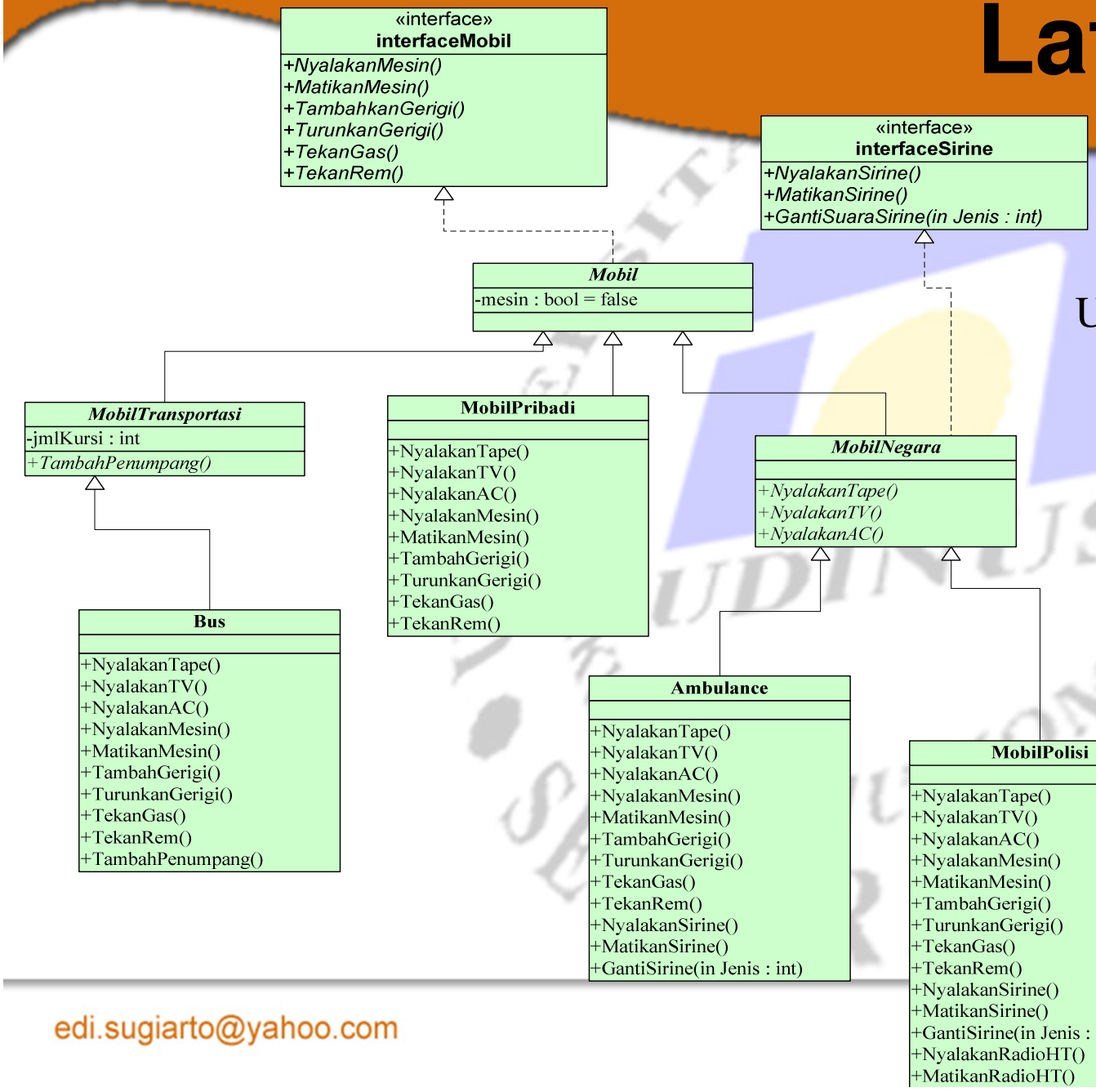
Latihan 7.1

Ubahlah struktur Kelas Diagram disamping ke dalam bahasa pemrograman



edi.sugiarto@yahoo.com

Latihan 7.2



Untuk latihan, ubahlah struktur kelas diagram disamping ke dalam bahasa pemrograman

edi.sugiarto@yahoo.com