Prinsip-Prinsip OOP

- Berikut ini merupakan konsep mana dari OOP?

Polymorphism	Encapsulation
Abstraction	Inheritance

1. Membuang detail yang tidak penting
2. Menggabungkan data dan prosedur dalam sebuah objek
3. Menambahkan fungsionalitas dengan membuat subclass baru
4. Suatu objek bisa memiliki banyak bentuk

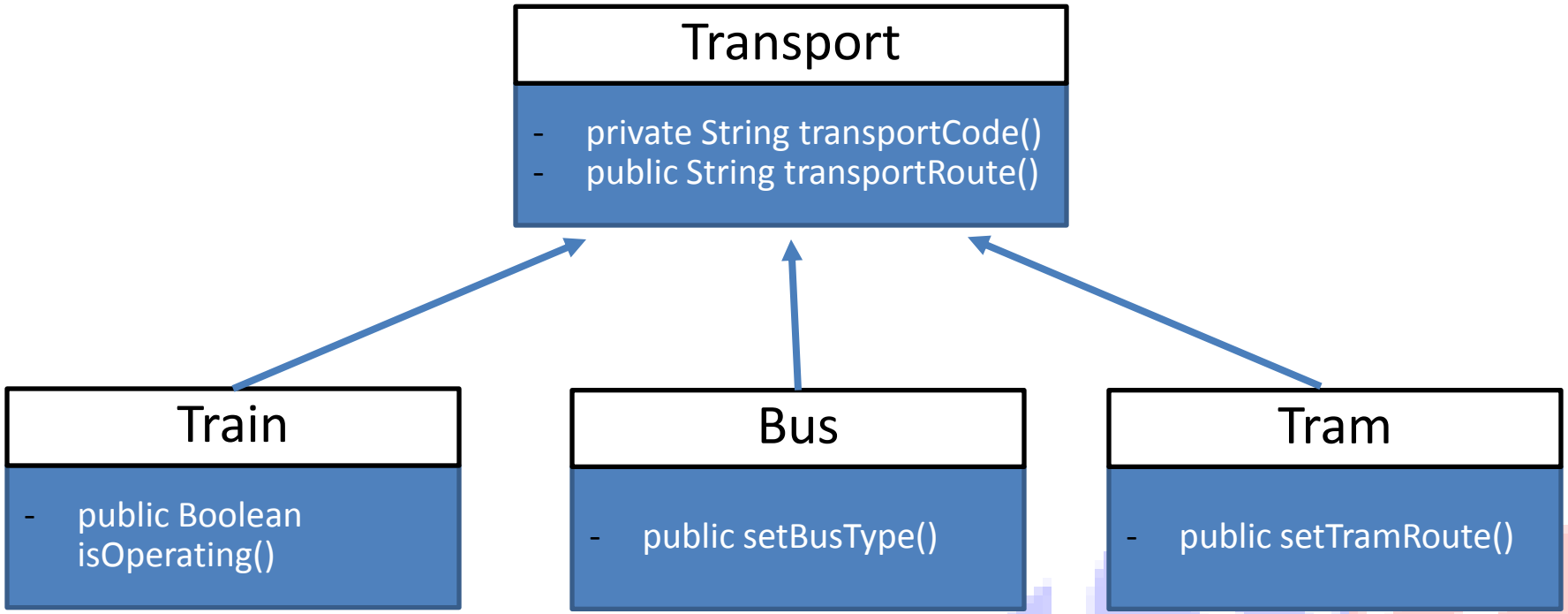
Overloading

- Manakah method-method yang saling overloading?

```
//a
public void calculate(int a0, int b0){...}

//b
private String calculate(int a0, int b0){...}

//c
public void calculate(String a0, int b0){...}

//d
public void calculate(int a1, int b1){...}
```

- Method yang saling overloading memiliki nama yang sama dan **signature** yang berbeda. Apa sajakah **signature method**?
- Bagaimana penanganan overloading methods? Apakah *static binding* atau *dynamic binding*?

Inheritance



- Apakah class Train mewarisi method transportCode()?
- Dapatkah objek dari class Tram mengakses method transportCode()?

Overriding

```
class Transport{

  public String travelRoute(){
        return "Inside the City";
    }
}

Class Train extends Transport{
    public String travelRoute(){
        return "From suburb to City";
    }
}

Class TestTransport{
    public static void main(String[] args){
        Transport transport0 = new Train();

        System.out.println(transport0.travelRoute());
    }
}
```

Dynamic Binding

- Apakah output dari Program tersebut?

Polymorphism

- Dalam mendefinisikan method untuk sebuah subclass, ada tiga kemungkinan:
  - override a method from the superclass (Nama dan signature sama dengan method milik superclass).
  - inherit method from the superclass. Methods yang tidak ditulis ulang otomatis akan diturunkan kepada seluruh subclass.
  - create a new method (Nama dan signature berbeda dengan method milik superclass)

Polymorphism

- Ketika sebuah method dari sebuah object dipanggil (called), class/type dari object tersebut menentukan implementasi method yang mana yang akan dijalankan.
- Sebuah method/field dapat atau tidaknya dipanggil/diakses ditentukan oleh type of the reference variable.
- Kata kunci (keyword) “super” berfungsi sebagai reference terhadap object tapi diperlakukan sebagai instance dari superclass.
- Polymorfism dapat diartikan bahwa sebuah object dapat memiliki banyak ‘bentuk’, yaitu sebagai object dari class-nya sendiri atau sebagai object dari superclass-nya.

Polymorphism

```
Student s = new Student();
Employee e = new Employee();

Person p = null;

if(getTodaysDay().equals("Tuesday"))
    p = s;
else
    p = e;

System.out.println("Person is " + p.toString());
```

- Method toString() mana sajakah yang akan dipanggil?
  a) toString() di class Person
  b) toString() di class Student
  c) toString() di class Employee

REVIEW DDP

Soal No. 1

```
public class  Misteri1 {
    public static boolean method(Comparable[] a) {
        boolean apaItu = false;
        for (int i = 0; i < a.length; i++) {
            for (int j = i + 1; j < a.length; j++) {
                if (a[i].compareTo(a[j]) == 0) {
                    return true;
                }
            }
        }
        return apaItu;
    }
    public static void main(String[] args) {
        String[] test = {"upin", "ipin", "dora", "donald", "mickey", "unyil",
"spongebob"};
        boolean apa = method(test);
        System.out.println(apa);
    }
}
```

- Apakah output dari program tersebut?

Soal No. 2

```
class Misteri2{
    public static Comparable [] test( Comparable [] a )    {
        Comparable [] obj1 = new Comparable [2];
        int index0 = 0;
        int index1 = 0;
        if( a[0].compareTo(a[1]) > 0)
            index0 = 1;
        else
            index1 = 1;
        for( int i = 1; i < a.length; i++ ){
            if( a[index0].compareTo(a[i]) > 0 ){
                index1 = index0;
                index0 = i;
            }
        }
        obj1[0] = a[index0];
        obj1[1] = a[index1];

        return obj1;
    }
    public static void main( String [] args ){
        String [] st1 = { "B", "C", "A", "F" };
        Comparable [] st2 = new Comparable[2];
        st2 = test(st1);
        System.out.println(st2[0] + " " + st2[1]);
    }
}
```

- Apakah output dari program tersebut?

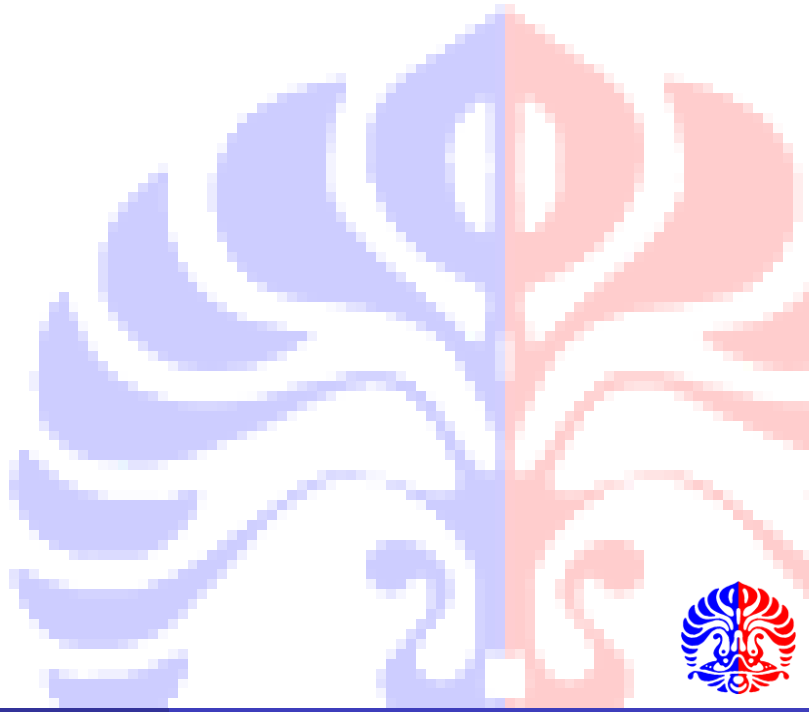
Soal No. 3

```
public static int misteri3(String s, String[] a){
    for(int i = 0; i < a.length; i++){
        if(a[i].compareTo(s) == 0){
            return i;
        }
    }
    return -1;
}
```

- Apakah output dari potongan program tersebut?

Soal No. 4

- Buatlah program untuk menghitung jumlah kemunculan bilangan genap pada suatu *array of integer*!



Soal No. 5

- Diberikan definisi *interface* sebagai berikut:

```
public interface BentukDuaDimensi{
    public double getLuas();
    public double getKeliling();
}
```

Implementasikan kelas **PersegiPanjang** dengan spesifikasi sebagai berikut:

- Mengimplements *interface* **BentukDuaDimensi**
- Mengimplementasikan *constructor* kelas
- Mengimplements *interface* **Comparable** untuk membandingkan dua buah objek **PersegiPanjang** dengan urutan sebagai berikut:
  - Urutkan persegi panjang berdasarkan panjangnya
  - Apabila ada dua objek yang memiliki panjang yang sama, urutkan berdasarkan lebarnya
  - Dua objek dikatakan sama apabila panjang dan lebarnya sama