Introduction to Object Oriented Programming (OOP)

1. Competency

After taking this module, students may be able to:

1. Recognize the difference between object oriented and procedural paradigm in programming.
2. Recognize the basic concept of OOP
3. Implement OOP in java programming

2. Introduction

2.1 Procedural Programming vs Object Oriented Programming

In procedural programming, a program is divided into some sub-program called function. It is difference with OOP, the main concept is a program is divided into object where object consists of state and method.

OOP is more flexible and modular, it can be described that each part of program won’t be disturbed if a feature is updated. It is difference with procedural programming where the program may change overall when an update is implemented.

More clearly, this is the sample of procedural programming and object-oriented programming.

Procedural Programming	**Object Oriented Programming**

`const roti = {nama: ‘Roti’, harga: 5000}`
`const susu = {nama: ‘Susu’, harga: 8000)`
 	
`const keranjang = [];`
`keranjang.push(roti);`
`keranjang.push(roti);`
`keranjang.push(susu);`
`keranjang.push(susu);`
`keranjang.push(susu);`
 	
`const total = keranjang`
`  .map(product => produk.harga)`
`  .reduce((a, b) => a + b, 0);`
 	
`console.log('Total bayar: ' + total);`
	`const roti = new Product('Roti', 5000);`
	`const susu = new Product('Susu', 8000)`
	`const keranjang = new Keranjang();`
	`keranjang.tambahProduk(2, roti);`
	`keranjang.tambahProduk(3, susu);`
	`keranjang.printShoppingInfo();`

Based on those code, object-oriented perspective can be identified more clearly as human language concept than procedural programming. From the first line of object-oriented, a new object uses **new** followed by the name of the class. It shows that an object is assigned to a variable with some return value, so it can be identified that object oriented programming is more efficient.

NB:
In this module, we will try to create class/classes, creating object, and calling methods in object. More explanation about class anatomy will be discussed in the next module.

2.2 Basic Concept of OOP

Some aspects are known in OOP, such:

a. Object

Object is a sequence of program which consists of **state** and **behavior**. Object in software is modelled to be similar of object in real world. State in object is attribute of the object itself, in example: object Sepeda has state like merek, kecepatan, gear, etc. Behavior is a something that the object can do, in example: in object Sepeda, the behaviors are: tambah kecepatan, pindah gear, kurangi kecepatan, belok, etc.

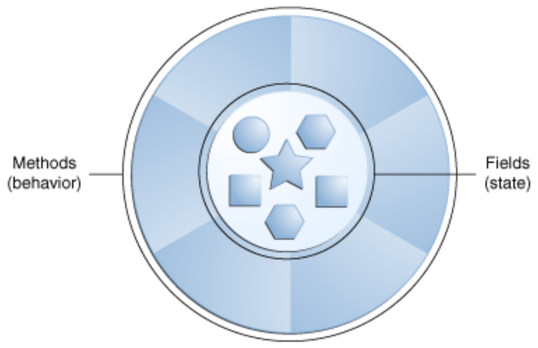


Figure 1. Illustration of object in software

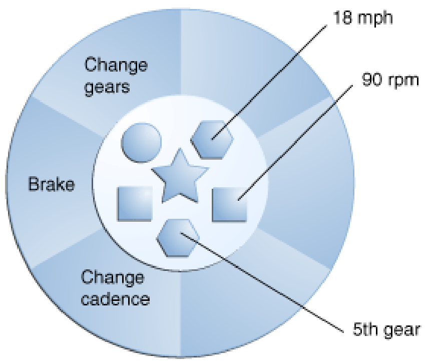


Figure 2. Illustration of state and behavior in object Sepeda

b. Class

Class is a blueprint or prototype from object. Taking example from object Sepeda, although there are many kinds of bicycle based on brand and model, the blueprint of bicycle will be the same such component and characteristic. The one you may belong in your home can be defined as instantiation from class Sepeda.

c. Encapsulation

Also known as **information-hiding**. The complex scheme in program may not be shown necessarily. In bicycle, we can change gear position by pressing gear knob at the handlebar, we may need not to know how the scheme in changing gear works technically.

d. Inheritance

Inheritance allow us to organize the structure of program naturally. We can expand the functionality of program without make a large modification in that program. In example, bicycle can be inherited to another model like mountain bike and road bike where each model has additional component which is not owned by common bike. In another word, mountain bike and road bike inherit bicycle.

e. Polymorphism

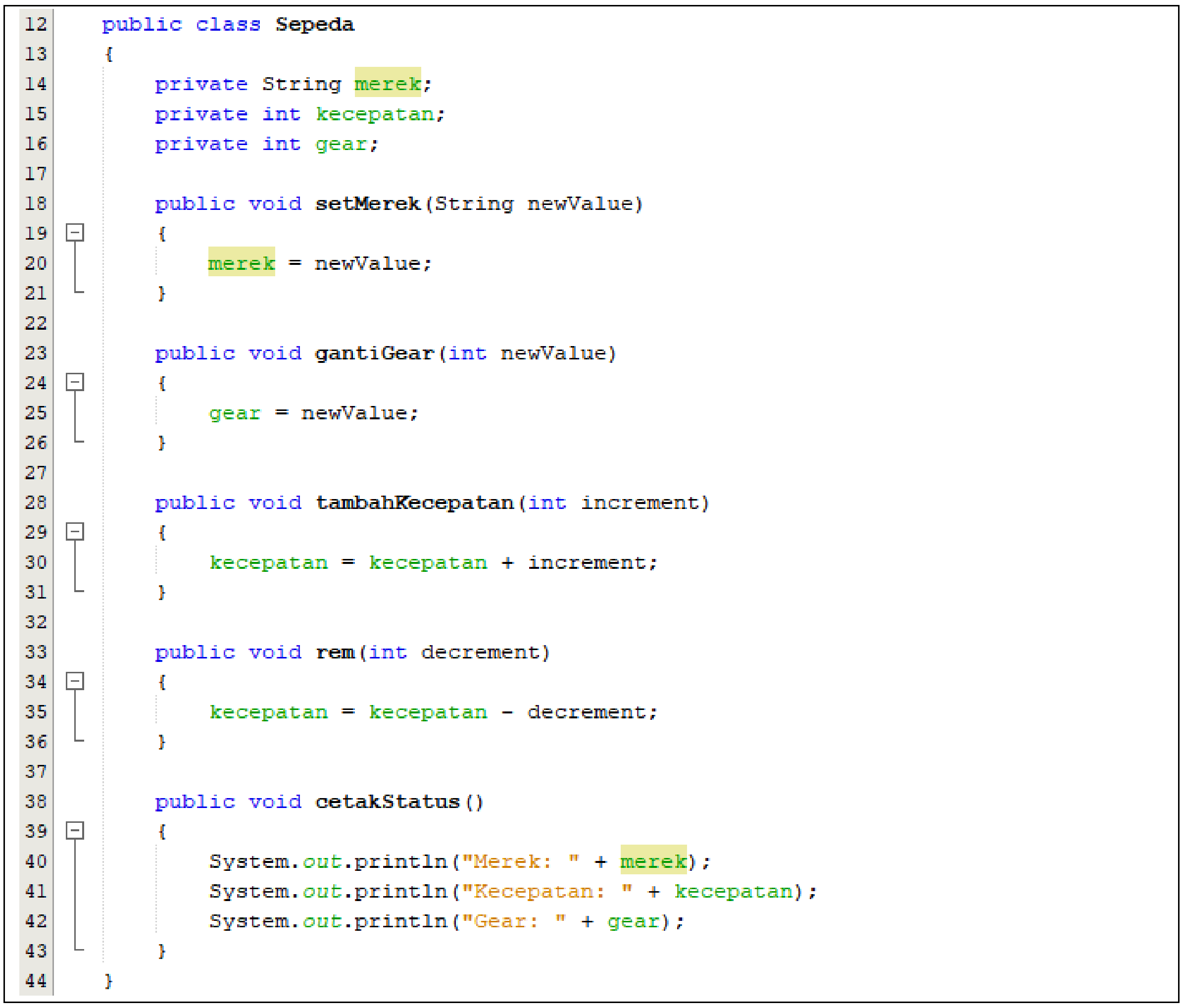
Polymorphism imitates real world object characteristics where an object may has different form. In example, plane object can be inherited to be jet plane and helicopter which have ability to increasing the speed. But, the method to increase the speed is different between those models because of different engine used technically.

3. Experiment Activity

3.1 Experiment Activity 1

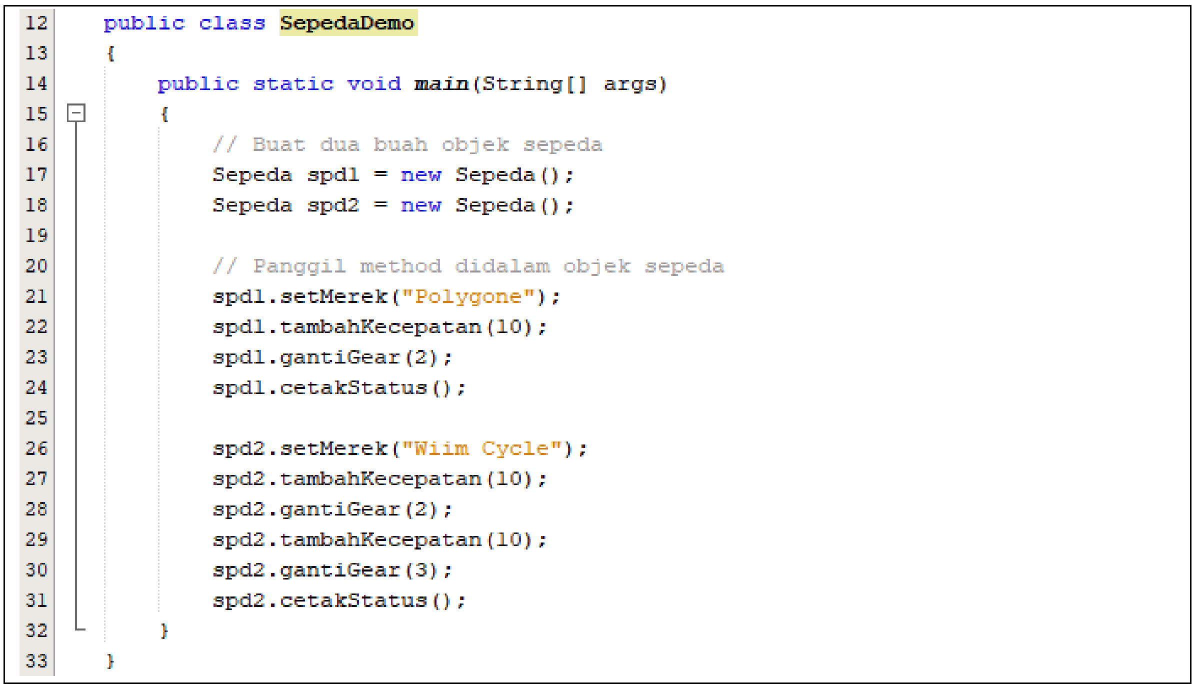
In this activity, we would demonstrate about how creating class, creating object, then accessing method in the class.

1. Open NetBeans, create project **SepedaDemo**.
2. Create class Sepeda. Right click on the package **sepedademo** – New – Java Class.
3. Type this code in class Sepeda.



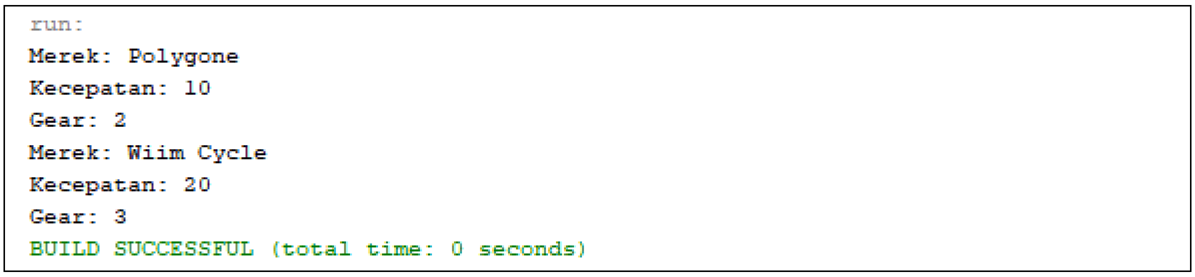
```java
public class Sepeda
{
    private String merek;
    private int kecepatan;
    private int gear;

    public void setMerek(String newValue)
    {
        merek = newValue;
    }

    public void gantiGear(int newValue)
    {
        gear = newValue;
    }

    public void tambahKecepatan(int increment)
    {
        kecepatan = kecepatan + increment;
    }

    public void rem(int decrement)
    {
        kecepatan = kecepatan - decrement;
    }

    public void cetakStatus()
    {
        System.out.println("Merek: " + merek);
        System.out.println("Kecepatan: " + kecepatan);
        System.out.println("Gear: " + gear);
    }
}
```

4. Then in the class main, type this code.



```java
public class SepedaDemo
{
    public static void main(String[] args)
    {
        // Buat dua buah objek sepeda
        Sepeda spd1 = new Sepeda();
        Sepeda spd2 = new Sepeda();

        // Panggil method didalam objek sepeda
        spd1.setMerek("Polygone");
        spd1.tambahKecepatan(10);
        spd1.gantiGear(2);
        spd1.cetakStatus();

        spd2.setMerek("Wiim Cycle");
        spd2.tambahKecepatan(10);
        spd2.gantiGear(2);
        spd2.tambahKecepatan(10);
        spd2.gantiGear(3);
        spd2.cetakStatus();
    }
}
```

5. Check the result, it should be like this:



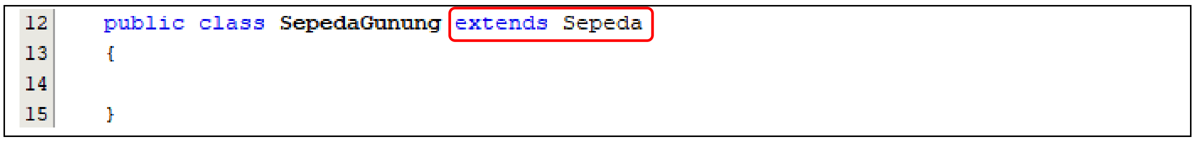
```
run:
Merek: Polygone
Kecepatan: 10
Gear: 2
Merek: Wiim Cycle
Kecepatan: 20
Gear: 3
BUILD SUCCESSFUL (total time: 0 seconds)
```

3.2 Percobaan 2

In this activity, we would demonstrate inheritance. We perform this inheritance technique by creating class **SepedaGunung** which is inherited from class Sepeda. Basically, class SepedaGunung I similar with class Sepeda but it has tipe suspense state. By doing this, we may not creating class SepedaGunung from the scratch, it just inherits from class Sepeda to class SepedaGunung.

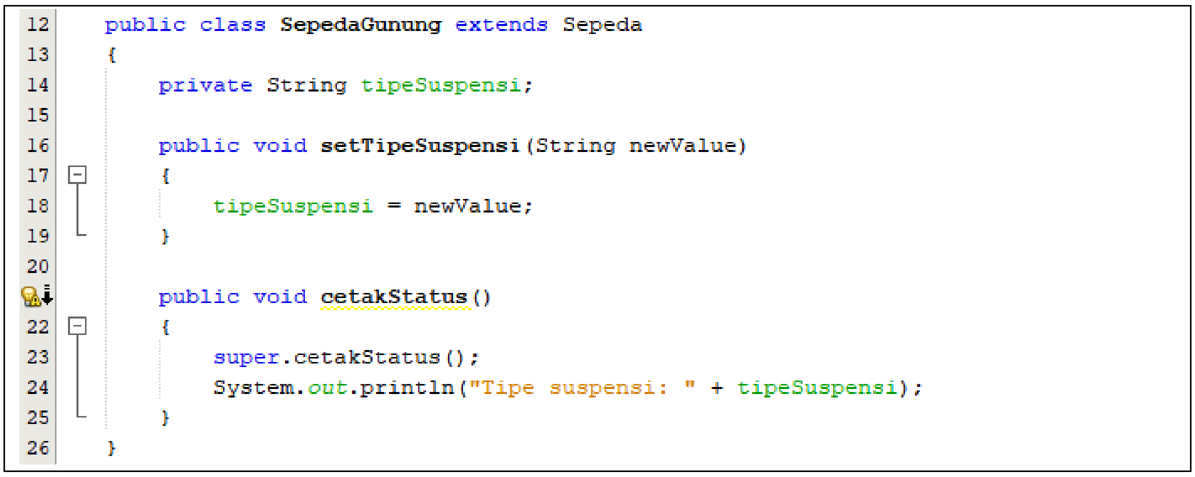
More detail explanation about inheritance will be discussed in the next module.

1. Still in the project **SepedaDemo**. Create a class named **SepedaGunung**.
2. Add a code **extends Sepeda** at the class SepedaGunung declaration. This extends points that class SepedaGunung inherits class Sepeda.



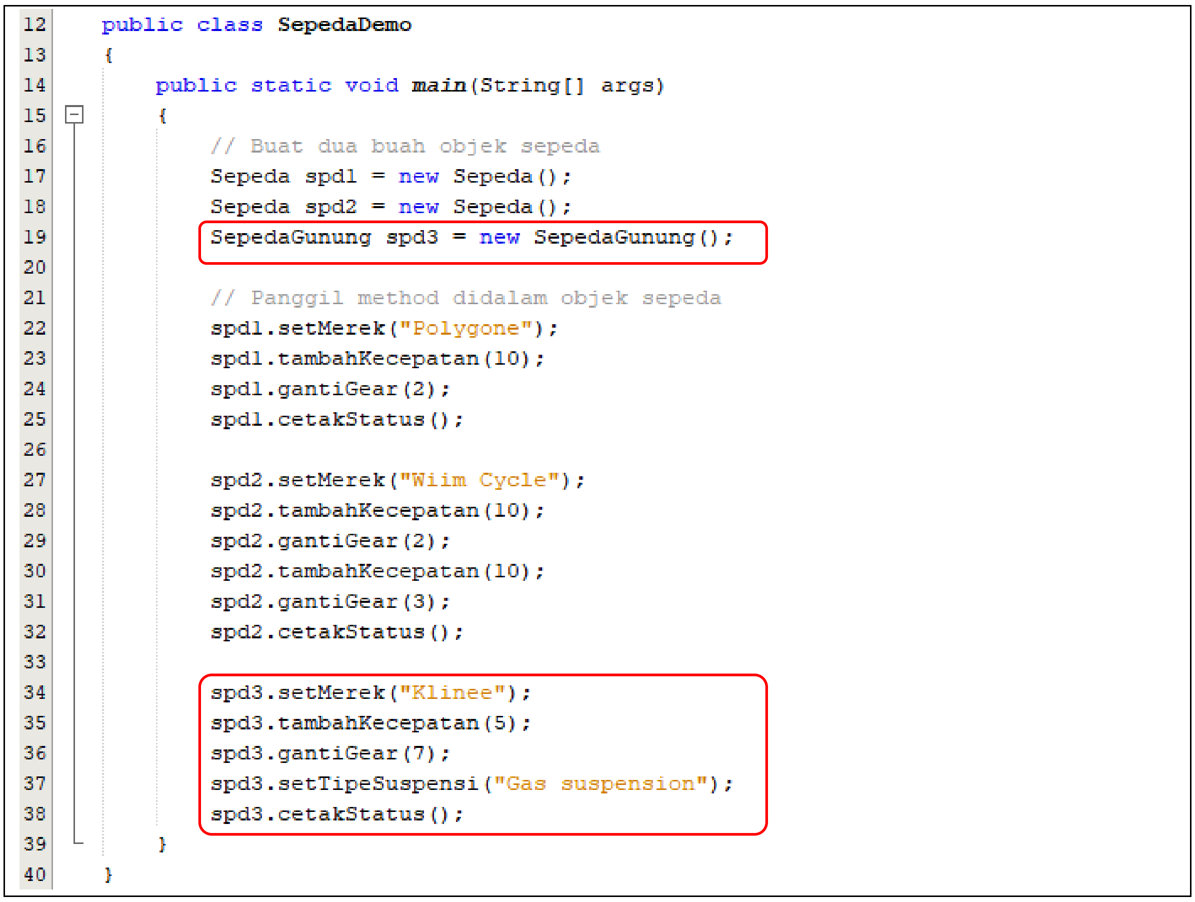
```java
public class SepedaGunung extends Sepeda
{

}
```

3. Complete the class SepedaGunung by typing this code:



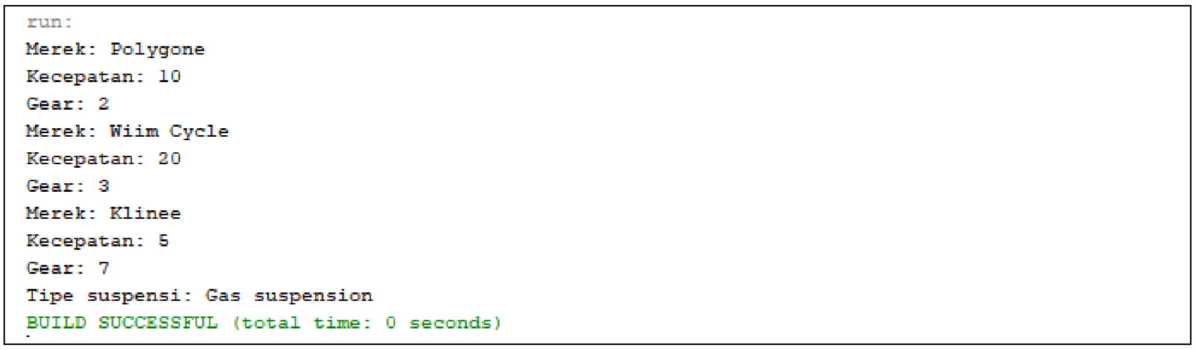
```java
public class SepedaGunung extends Sepeda
{
    private String tipeSuspensi;

    public void setTipeSuspensi(String newValue)
    {
        tipeSuspensi = newValue;
    }

    public void cetakStatus()
    {
        super.cetakStatus();
        System.out.println("Tipe suspensi: " + tipeSuspensi);
    }
}
```

4. Kemudian pada class main, tambahkan kode berikut ini:



```java
public class SepedaDemo
{
    public static void main(String[] args)
    {
        // Buat dua buah objek sepeda
        Sepeda spd1 = new Sepeda();
        Sepeda spd2 = new Sepeda();
        SepedaGunung spd3 = new SepedaGunung();

        // Panggil method didalam objek sepeda
        spd1.setMerek("Polygone");
        spd1.tambahKecepatan(10);
        spd1.gantiGear(2);
        spd1.cetakStatus();

        spd2.setMerek("Wiim Cycle");
        spd2.tambahKecepatan(10);
        spd2.gantiGear(2);
        spd2.tambahKecepatan(10);
        spd2.gantiGear(3);
        spd2.cetakStatus();

        spd3.setMerek("Klinee");
        spd3.tambahKecepatan(5);
        spd3.gantiGear(7);
        spd3.setTipeSuspensi("Gas suspension");
        spd3.cetakStatus();
    }
}
```

5. Check the result:



```
run:
Merek: Polygone
Kecepatan: 10
Gear: 2
Merek: Wiim Cycle
Kecepatan: 20
Gear: 3
Merek: Klinee
Kecepatan: 5
Gear: 7
Tipe suspensi: Gas suspension
BUILD SUCCESSFUL (total time: 0 seconds)
```

4. Conclusion

Based on the activity, we have demonstrated about what is paradigm of OOP and the implementation into a simple program. Inheritance also one of the importance features in OOP, we have demonstrated it by implementing class **SepedaGunung**.

We also know that class SepedaGunung is similar with class **Sepeda** (having gear and kecepatan state, also some behaviors like increasing speed, braking, changing gear, dsb) but with additional state named **tipe suspensi**. So we may not necessarily to create class SepedaGunung from the scratch, we use **extends** or just inherits from class Sepeda, then we add a new feature which is not owned by class Sepeda before. This is the characteristic of OOP which is not having in procedural programming.

5. Question Test

1. Explain the difference between object and class!
2. State your reason why color and engine type can be classified as attribute for car object!
3. State one of OOP better point than procedural programming!
4. Is it allowed to define two attributes in one line code such **“public String nama,alamat;”?**
5. In SepedaGunung class, state your reason why merk, kecepatan, and gear attributes are not written again in this class!

6. Assignment

1. Follow these instructions to make your practical assignment is performed systematically:
    a. Take 4 photographs of objects around you, 2 objects must be implementation of inheritance concept, example: refrigerator, chair, living room table, desk! As we know that living room table and desk are inherited by table class.
    b. Observe those objects to define the attribute and method!
    c. Convert those objects into four classes in Java programming!
    d. Add one additional class as a class which inherits its attribute and method to living room table class and desk class!
    e. Add two attributes for each class!
    f. Add three methods for each class including a method for showing the information!
    g. Add one class named Demo for main class!
    h. Instance an object for each class!
    i. Apply each method for each object in main class!
    j. The example which is mentioned in point 1.a should not be included in your task!