Name : Faricha Aulia
Class : 2I / Informatics Engineering
NIM : 2141720155

Jobsheet 6
Inheritance

1. COMPETENCY

- Understand the basic concepts of inheritance.
- Able to create a subclass of a particular superclass.

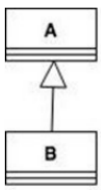
2. INTRODUCTION

Inheritance is a way to reduce a more general class to a more specific class. Inheritance is one of the main features of an object-oriented programming language. The essence of inheritance is the reusable nature of the concept of object oriented. Each subclass will "inherit" the nature of the superclass as long as it is protected or public. In inheritance there are two terms that are often used.
The derived class is called the base class (base class / super class), while the derived class is called a derived class / sub class / child class. In Java to declare a class as a subclass is done by adding the extends keyword after the class name declaration, then followed by the name of the parent class. The extends keyword tells the Java compiler that we want to extend the class. Following is an example of inheritance declaration.

```
public class B extends A {

...
}
```

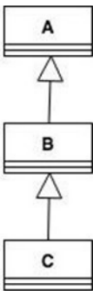
The above example tells the Java compiler that we want to extend class A to class B. In other words, class B is a subclass (class derived) from class A, whereas class A is the parent class of class B. The characteristics of the super class will also be owned by the subclasses. There are 3 forms of inheritance: single inheritance, multilevel inheritance, and multiple inheritance. But what will be discussed in this jobsheet are single inheritance and multilevel inheritance.

1. Single inheritance is a class that only has one parent class. Example:

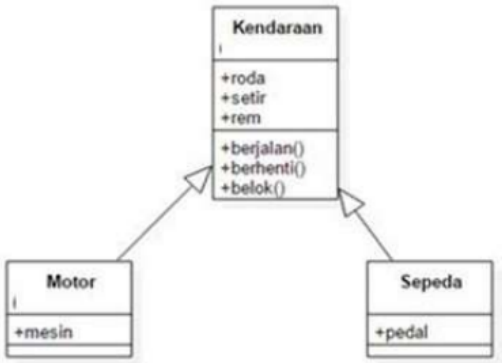


Based on Figure 1, it can be seen that class B is a subclass that has one parent, class A, so it is called single inheritance.

2. Multilevel inheritance is a subclass that can be a superclass for other classes. Example:



Based on Figure 2 above it can be seen that class B is a subclass of class A, so in this case class A is a superclass and class B is a subclass. Then class B, which was originally a subclass, has another subclass, class C, so class B becomes a superclass of class C, and so does class if class C has more subclasses. In class diagrams, inheritance is represented by a solid line, with a triangle at the end. Classes that are close to triangles are superclass, while classes that are far from triangles are subclasses. To form a subclass, the keyword "extends" is used (see examples in the "Inheritance Implementation" section). Following is an example class diagram of inheritance:



A parent class may not pass a portion of its members to its subclass. The extent to which a member can be inherited to another class, or a member can be accessed from another class, is closely related to access control. In java, access controls can be described in the following table:

Modifier	class yang sama	package yang sama	subclass	class manapun
private	√			
default	√	√		
protected	√	√	√	
public	√	√	√	√

The super keyword is used to refer to members of the parent class. As the keyword this is used to refer to members of the class itself. The format of the writing is as follows:

• super.name attribute
refers / access attributes from parent class / superclass
• super.nameMethod ()
refers / invokes methods from the parent class / superclass
• super ()
referring / calling the parent class / superclass constructor Can only be used in the first row in the constructor.
• super (parameter1, parameter2, etc.)
refers / invokes the parameterized constructor of superclass.

When creating an object from a subclass, at that moment the object in the superclass will also be formed. With catalyst, when the subclass constructor is run to create the object, then the superclass constructor will also be running. So in each subclass constructor, in the first row the subclass constructor will be called a superclass constructor. Before the subclass runs its own constructor, the subclass will run the superclass constructor first.

3. TRIAL 1 (Extends)

A. Code

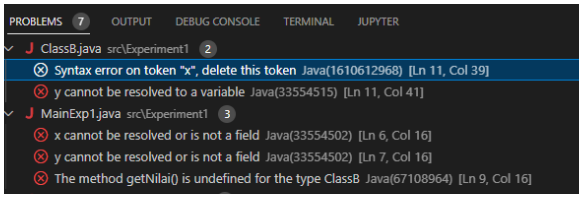
```
public class ClassA {
    public int x;
    public int y;
    public void getNilai(){
        System.out.println("Nilai x: " +x);
        System.out.println("Nilai y: " +y);
    }
}

public class ClassB {
    public int z;

    public void getNilaiZ(){
        System.out.println("Nilai z: " +z);
    }

    public void getJumlah(){
        System.out.println("Jumlah: " x+y+z);
    }
}

public class MainExp1 {
    public static void main(String[] args) {
        ClassB hitung = new ClassB();
        hitung.x = 20;
        hitung.y = 30;
        hitung.z = 5;
        hitung.getNilai();
        hitung.getNilaiZ();
        hitung.getJumlah();
    }
}
```

B. Questions

1. In Experiment 1 above the program that was running error occurred, then fix so that the program can be run and not error!

   Answer:

   - The error:

   - Fix program:
     Public class Class B extends ClassA {
     }
   - Information:

The class that needs to be fixed is ClassB, by adding extends ClassA and add super to the x and y attributes. Because the attribute is taken from class A via extends.

- Run Program:

```
Nilai x: 20
Nilai y: 30
Nilai z: 5
Jumlah: 20305
```

2. Explain what caused the program in experiment 1 when it ran an error!
   Answer:
   Experimental program 1 error because the x and y attributes could not be found. It happens that classB does not extend to classA which has the x and y attributes.

4. TRIAL 2 (Access Control)

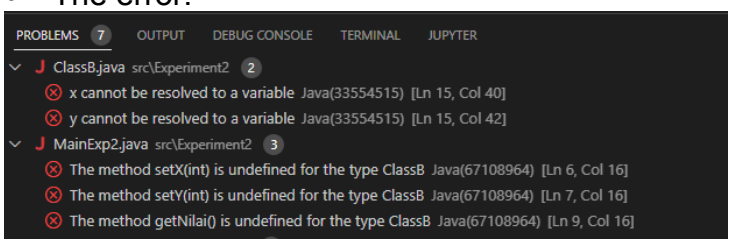
A. Code

```
public class ClassA {
    public int x;
    public int y;
    public void setX(int x) {
        this.x = x;
    }
    public void setY(int y) {
        this.y = y;
    }
    public void getNilai(){
        System.out.println("Nilai x: " +x);
        System.out.println("Nilai y: " +y);
    }
}
public class ClassB {
    public int z;

    public void setZ(int z) {
        this.z = z;
    }
    public void getNilaiZ(){
        System.out.println("Nilai z: " +z);
    }
    public void getJumlah(){
        System.out.println("Jumlah: " +x+y+z);
    }
}
```

B. Questions

1. In Experiment 2 above, the program that runs an error occurs, then fix it so that the program can be run and not error!
   Answer:

- The error:



- Fix Program:

```
public class ClassA {
    public int x;
    public int y;

    public void setX(int x) {
        this.x = x;
    }
    public void setY(int y) {
        this.y = y;
    }
    public int getX() {
        return x;
    }
    public int getY() {
        return y;
    }
    public void getNilai(){
        System.out.println("Nilai x: " +x);
        System.out.println("Nilai y: " +y);
    }
}
public class ClassB extends ClassA{
    public int z;
    public void setZ(int z) {
        this.z = z;
    }
    public void getNilaiZ(){
        System.out.println("Nilai z: " +z);
    )
    public void getJumlah(){
        System.out.println("Jumlah: " +x+y+z);
    }
}
```

- Run Program:

```
Nilai x: 20
Nilai y: 30
Nilai z: 5
Jumlah: 20305
```

Information:

- In ClassB must be added extends ClassA, in order to take or using the attributes and methods of ClassA
- In Class A, it is necessary to add a Getter method to retrieve the value from attributes that are in ClassA. Because of the attribute modifier in the ClassA using private which means that the attribute can only be accessed by ClassA itself. Therefore we need an additional method that functions to retrieve data value of the attribute which later the method is called in another class.
- In the getAmount() method in ClassB, the x attribute is changed to super.getX() and y attribute changed to super.getY()

2. Explain what caused the program in experiment 1 when it ran an error!

   Answer:

   There are several reasons for the 2nd trial program, including:

   - No added extends ClassA on ClassB
   - There is an access block for the x and y attributes in ClassB because of the two attributes it uses the private modifier in ClassA.

- Code error in retrieving values from x and y attributes, because x attributes and y is private which cannot be directly called or used in class other

5. TRIAL 3 (Super)

A. Code

```
public class Bangun {
    protected double phi;
    protected int r;
}
public class MainExp3 {
    public static void main(String[] args) {
        Tabung tbg = new Tabung();
        tbg.setSuperPhi(3.14);
        tbg.setSuperR(10);
        tbg.setT(3);
        tbg.volume();
    }
}
public class Tabung extends Bangun{
    protected int t;
    public void setSuperPhi (double phi){
        super.phi = phi;
    }
    public void setSuperR (int r){
        super.r = r;
    }
    public void setT (int t){
        this.t = t;
    }
    public void volume(){
        System.out.println("Volume tabung adalah: " +(super.phi*super.r*super.r*this.t));
    }
}
```

Run Program:

```
Volume tabung adalah: 942.0
```

B. Questions

1. Explain the "super" function in the following program snippet in the Tube class!

```
public void setSuperPhi(double phi){
super.phi = phi;
}
public void setSuperR(int r){
super.r = r;
}
```

Answer: To call the extended parent class which later the value of the attribute will be used in the Tube class.

2. Explain the "super" and "this" functions in the following program snippet in the Tube class

```
public void volume(){
System.out.println("Volume Tabung adalah: "+(super.phi*super.r*super.r*this.t));
```

Answer:

- The "super" function is to call the parent class that has been extended which later the value of the attribute will be used in the Tube class.

- The function "this" is a guide or info for the user which aims to inform that any code that uses "this" means accessing an attribute or method that is in the class itself.

3. Explain why the Tube class does not declare the "phi" and "r" attributes, but the class can access these attributes!
   Answer: In the Tube class, the attributes "phi" and "r" are not declared because the Tube class has been inherited from its parent class, the Build class. Therefore, the attributes in the Build class can be accessed by the Tube class

6. TRIAL 4 (Super Constructor)

A. Code

```
public class ClassA {
    ClassA(){
        System.out.println("Konstruktor A dijalankan");
    }
}
public class ClassB extends ClassA{
    ClassB(){
        System.out.println("Konstruktor B dijalankan");
    }
}
public class ClassC extends ClassB{
    ClassC(){
        System.out.println("Konstruktor C dijalankan");
    }
}
public class MainExp4 {
    public static void main(String[] args) {
        ClassC test = new ClassC();
    }
}
```

Run Program:

```
Konstruktor A dijalankan
Konstruktor B dijalankan
Konstruktor C dijalankan
```

B. Questions

1. In experiment 4 state which class includes the superclass and subclass, then explain the reason!
   Answer:
   - ClassA becomes a superclass of ClassB: Marked by the existence of extends ClassA in ClassB
   - ClassB becomes a superclass of ClassC: Marked by the existence of extends ClassB in ClassC
   - ClassB becomes a subclass of ClassA : Marked by the existence of extends ClassA in ClassB
   - ClassC becomes a subclass of ClassB : Marked by the existence of extends ClassB in ClassC
2. Change the contents of the ClassC default constructor as follows:

```
public class ClassC extends ClassB{
    ClassC(){
    super();
    System.out.println("konstruktor C dijalankan");
    }
}
```

Add the word super() in the First row in the default constructor. Try running the Experiment 4 class again and it looks like there is no difference from the output!

Answer: There is no difference in the output

```
Konstruktor A dijalankan
Konstruktor B dijalankan
Konstruktor C dijalankan
```

3. Define the contents of the ClassC default constructor as follows:

```
public class ClassC extends ClassB{
    ClassC(){
    System.out.println("konstruktor C dijalankan");
    super();
    }
}
```

When changing the super() position in the second line in the default constructor and there is an error. Then return super () to the first line as before, then the error will disappear.

```
run:
konstruktor A dijalankan
konstruktor B dijalankan
konstruktor C dijalankan
BUILD SUCCESSFUL (total time: 0 seconds)
```

Pay attention to the output when the Test 4 class is run. Why can the output appear as follows when instantiating the test object from the ClassC. Explain how the order of the constructor goes when the test object is created!

Answer: An error occurred because the constructor call of the parent class must be declared at the beginning of a statement

```
ClassC.java src\Expriment4 1
Constructor call must be the first statement in a constructor Java(1207959691) [Ln 6, Col 9]
```

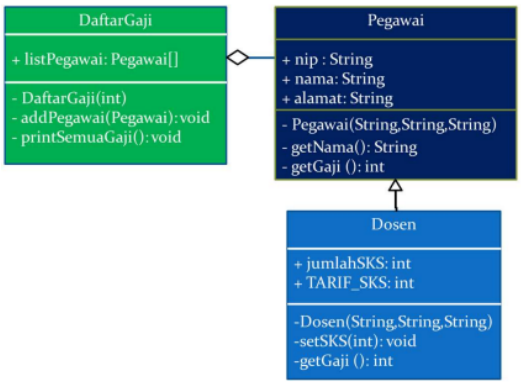
4. What is the super() function in the following program snippet in ClassC!

```
public class ClassC extends ClassB{
    ClassC(){
    super();
    System.out.println("konstruktor C dijalankan");
    }
}
```

Answer: The super() function in the program is to call the default constructor of the parent class. The default super() and extends class functions are almost the same, namely calling the constructor method on the parent class when the program is first run.

7. ASSIGMENT

Make a program with the concept of inheritance as in the following class diagram. Then create an object instantiation to display the employee name and salary they get.



Program code:

```
public class Pegawai {
    private String nip;
    private String nama;
    private String alamat;
    public String getNip() {
        return this.nip;
    }
    public void setNip(String nip) {
        this.nip = nip;
    }
    public String getNama() {
        return this.nama;
    }
    public void setNama(String nama) {
        this.nama = nama;
    }
    public String getAlamat() {
        return this.alamat;
    }
    public void setAlamat(String alamat) {
        this.alamat = alamat;
    }
    public Pegawai(String nip, String nama, String alamat) {
        this.nip = nip;
        this.nama = nama;
        this.alamat = alamat;
    }
    public int getGaji() {
        return 0;
    }
}
```

```
public class Dosen extends Pegawai{
    private int jumlahSKS;
    private int tarifSKS;
    public Dosen(String nip, String nama, String alamat, int jumlahSKS, int tarifSKS) {
        super(nip, nama, alamat);
        this.jumlahSKS = jumlahSKS;
        this.tarifSKS = tarifSKS;
    }
    public int getJumlahSKS() {
        return this.jumlahSKS;
    }
    public void setJumlahSKS(int jumlahSKS) {
        this.jumlahSKS = jumlahSKS;
```

```
    }
    public int getTarifSKS() {
        return this.tarifSKS;
    }
    public void setTarifSKS(int tarifSKS) {
        this.tarifSKS = tarifSKS;
    }
    @Override
    public int getGaji() {
        return this.jumlahSKS * this.tarifSKS;
    }
}
```

```
public class DaftarGaji {
    Pegawai[] listPegawai;
    public DaftarGaji(int jumlah) {
        listPegawai = new Pegawai[jumlah];
    }
    public void printSemuaGaji() {
        for (int i = 0; i < listPegawai.length; i++) {
        if (listPegawai[i] == null) break;
            System.out.println("==============================");
            System.out.println("Pegawai ke-" + (i + 1) + ": ");
            System.out.println("Nama    : " + listPegawai[i].getNama());
            System.out.println("NIP     : " + listPegawai[i].getNip());
            System.out.println("Alamat  : " + listPegawai[i].getAlamat());
            System.out.println("Gaji    : " + listPegawai[i].getGaji());
            System.out.println();
        }
    }
    public void addPegawai(Pegawai p) {
        for (int i = 0; i < listPegawai.length; i++) {
            if (listPegawai[i] == null) {
            listPegawai[i] = p;
            System.out.println("Pegawai berhasil ditambahkan");
            break;
            }
        }
    }
}
```

```
public class Main {
    public static void main(String[] args) {
        DaftarGaji daftarGaji = new DaftarGaji(5);
        daftarGaji.addPegawai(new Dosen("54286", "Spongebob", "Pantai Bikini", 6, 600000));
        daftarGaji.addPegawai(new Dosen("55209", "Sandy", "Texas", 6, 750000));
        daftarGaji.addPegawai(new Dosen("54299", "Squidward", "Bikini Bottom", 5, 500000));
        daftarGaji.printSemuaGaji();
        }
}
```

Output:

```
Pegawai berhasil ditambahkan
Pegawai berhasil ditambahkan
Pegawai berhasil ditambahkan
==============================
Pegawai ke-1:
Nama    : Spongebob
NIP     : 54286
Alamat  : Pantai Bikini
Gaji    : 3600000

==============================
Pegawai ke-2:
Nama    : Sandy
NIP     : 55209
Alamat  : Texas
Gaji    : 4500000

==============================
Pegawai ke-3:
Nama    : Squidward
NIP     : 54299
Alamat  : Bikini Bottom
Gaji    : 2500000
```