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

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

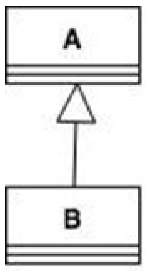


Figure 1. Example of a Single Inheritance

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

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

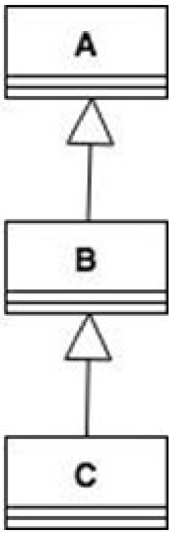


Figure 2. Example of Multilevel Inheritance

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:

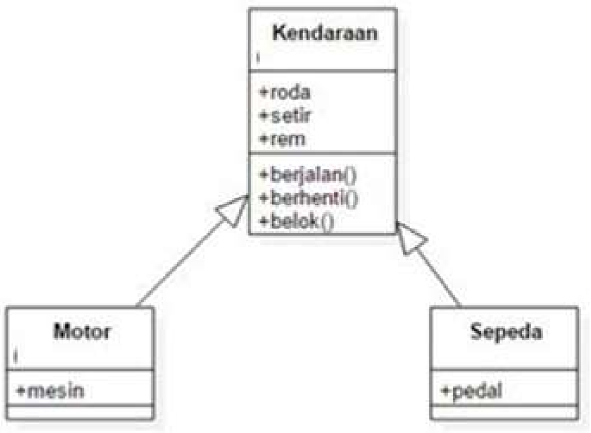
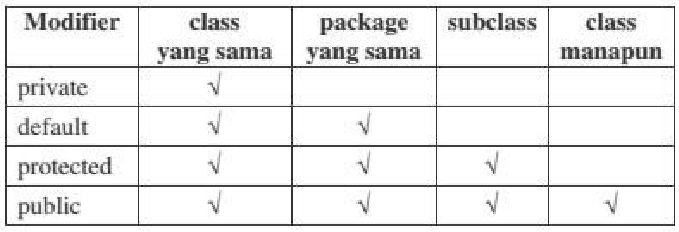


Figure 3 Example of a class diagram in an error

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.)

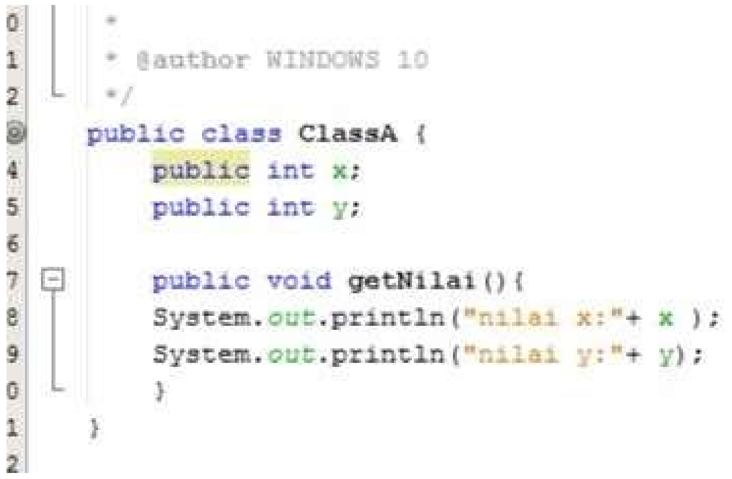
Refers / invokes the parameterized constructor of superclass

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. STEP BY STEP

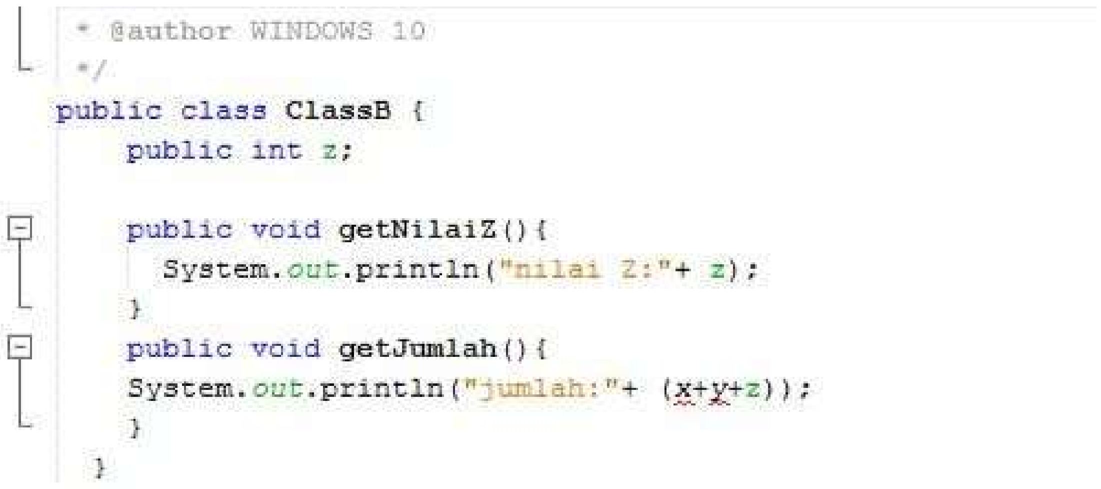
1. Create a parent / superclass class named ClassA.java



```java
 *
 * @author WINDOWS 10
 */
public class ClassA {
    public int x;
    public int y;

    public void getNilai(){
    System.out.println("nilai x:"+ x );
    System.out.println("nilai y:"+ y);
    }
}
```

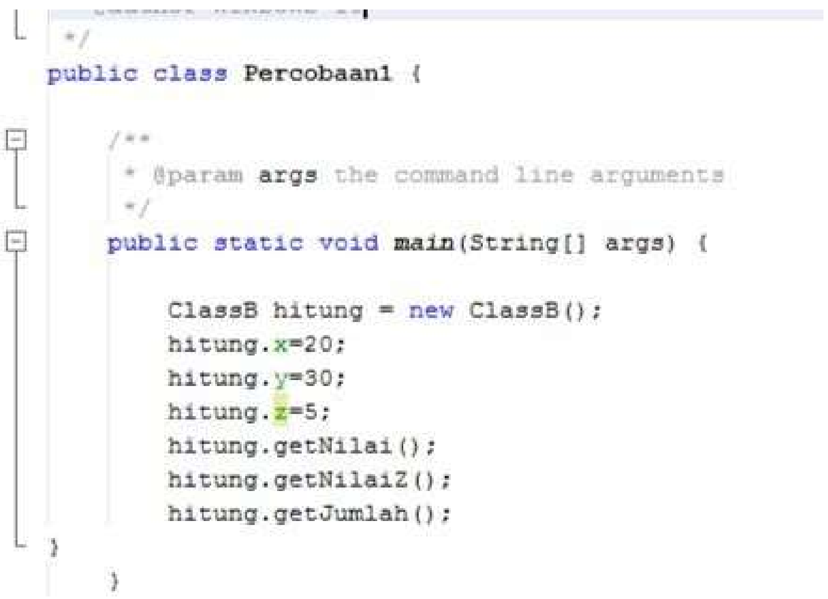
2. Create a child class / subclass with the name ClassB.java



```java
 * @author WINDOWS 10
 */
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));
    }
 }
```

3. Create a class Percobaan1.java to run the above program!



```java
 */
public class Percobaan1 {

    /**
     * @param args the command line arguments
     */
    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();
}
    }
```

4. Run the program above, then observe what happens!

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!

....

....

....

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

....

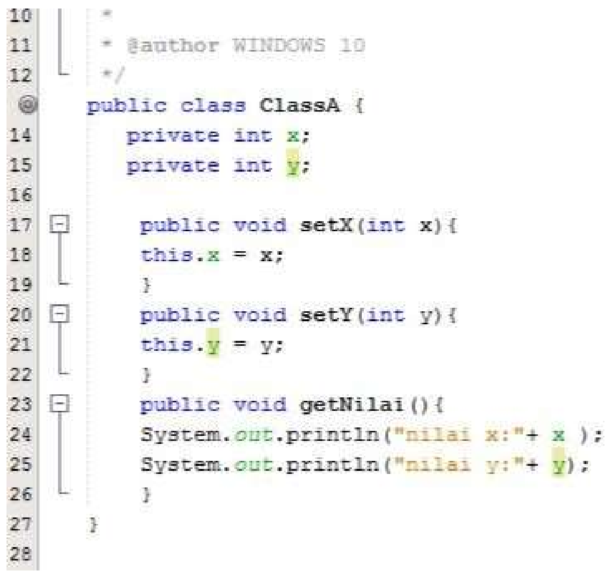
....

....

4. TRIAL 2 (Access Control)

A. STEP BY STEP

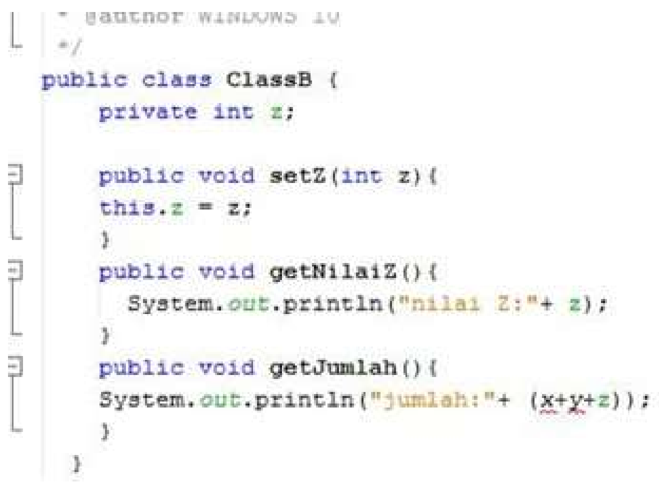
1. Create a parent / superclass class named ClassA.java



```
 *
 * @author WINDOWS 10
 */
public class ClassA {
   private int x;
   private 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);
    }
}
```

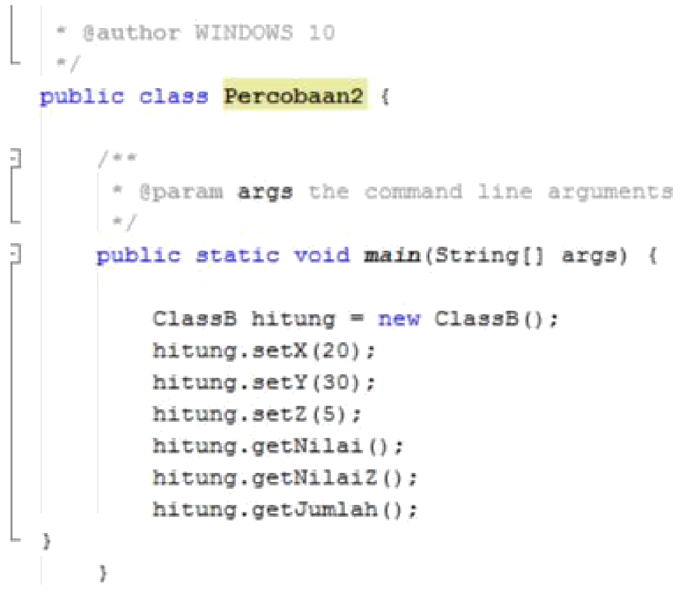
2. Create a child / subclass class named ClassB.java



```
 * @author WINDOWS 10
 */
public class ClassB {
    private 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));
    }
  }
```

3. Create a class Percobaan2.java to run the above program!



```
 * @author WINDOWS 10
 */
public class Percobaan2 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {

        ClassB hitung = new ClassB();
        hitung.setX(20);
        hitung.setY(30);
        hitung.setZ(5);
        hitung.getNilai();
        hitung.getNilaiZ();
        hitung.getJumlah();
}
    }
```

4. Run the program above, then observe what happens!

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!

……………………………………………… ……………………………………………… ………………………………………………

……………………………………………… ……………………………………………… ………………………………………………

……………………………………………… ……………………………………………… ………………………………………………

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

....

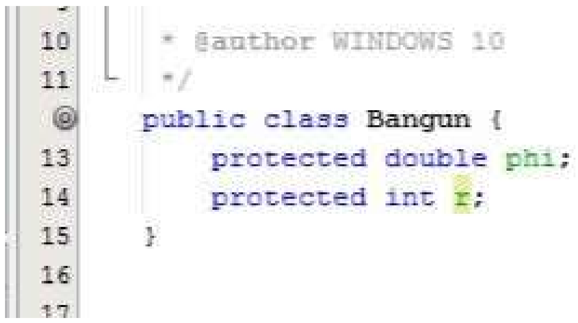
....

....

5. TRIAL 3 (Super)

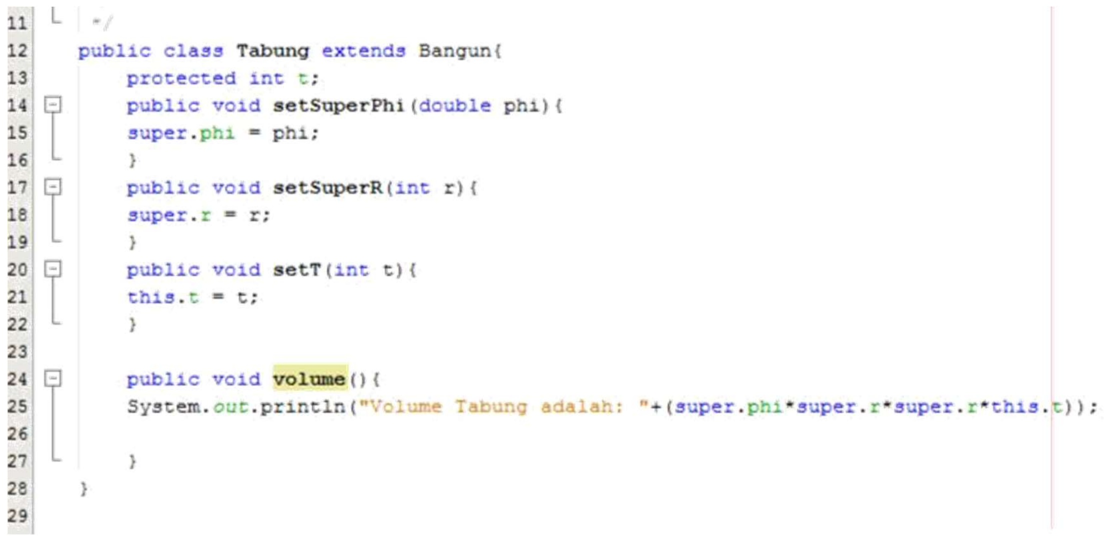
A. STEP BY STEP

1. Create a parent / superclass class named Bangun.java



```java
 * @author WINDOWS 10
 */
public class Bangun {
    protected double phi;
    protected int r;
}
```

2. Create a child / subclass class named Tabung.java

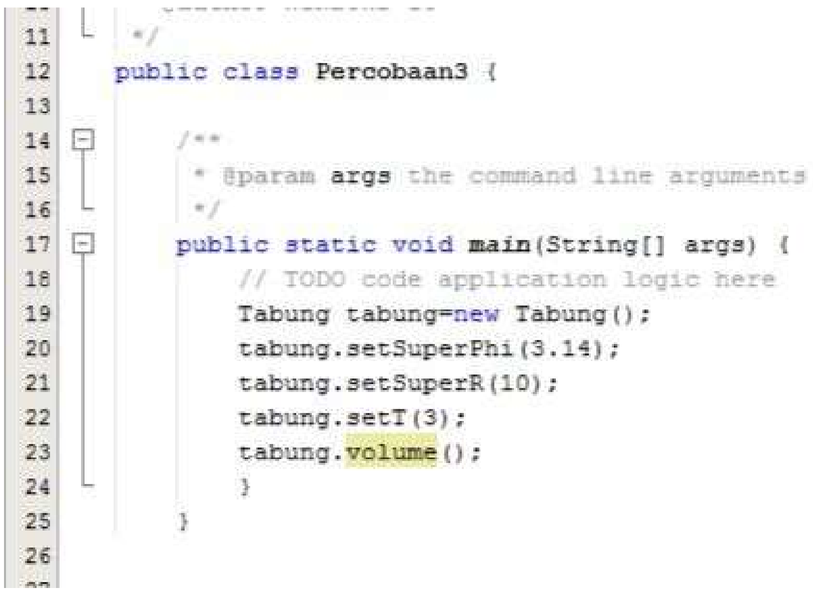


```java
 */
public class Tabung extends Bangun{
    protected int t;
    public void setSuperPhi(double phi){
    super.phi = phi;
    }
    public void setSuperR(int r){
    super.r = r;
    }
    public void setT(int t){
    this.t = t;
    }

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

    }
}
```

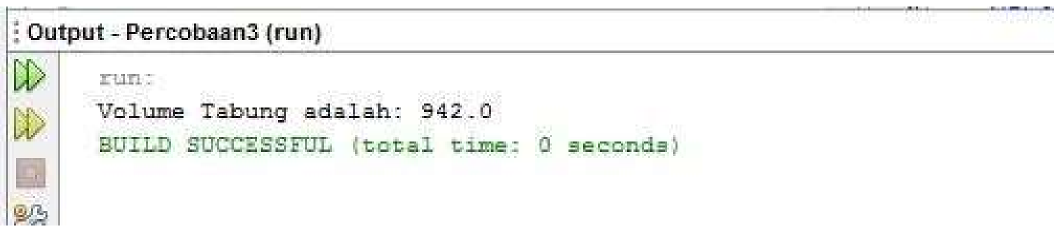
3. Create a class Percobaan3.java to run the program above!



```java
 */
public class Percobaan3 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        Tabung tabung=new Tabung();
        tabung.setSuperPhi(3.14);
        tabung.setSuperR(10);
        tabung.setT(3);
        tabung.volume();
        }
    }
```

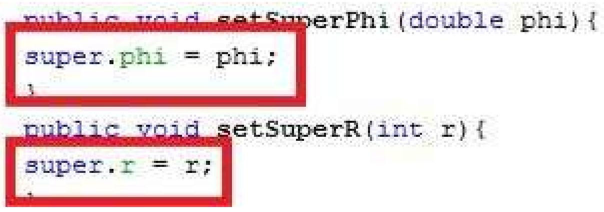
4. Run the program above!



```
Output - Percobaan3 (run)
run:
Volume Tabung adalah: 942.0
BUILD SUCCESSFUL (total time: 0 seconds)
```

B. QUESTIONS

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



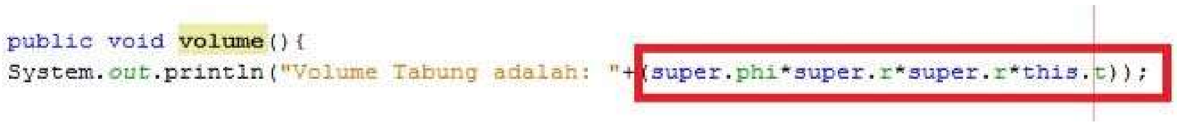
```
public void setSuperPhi(double phi){
super.phi = phi;
}
public void setSuperR(int r){
super.r = r;
}
```

....

....

....

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));
```

3. Explain why the Tube class does not declare the "phi" and "r" attributes, but the class can access these attributes!

....

....

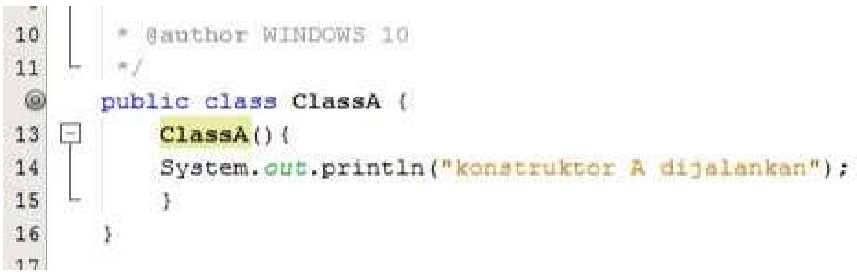
....

6. TRIAL 4 (super contructor)

A. STEP BY STEP

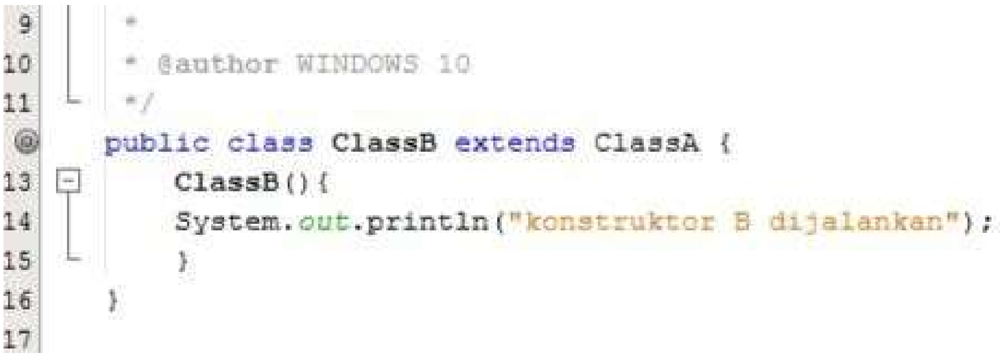
1. Make three files with the name ClassA.java, ClassB.java, and ClassC.java, as in the program code below!

ClassA.java



```
 * @author WINDOWS 10
 */
public class ClassA {
    ClassA(){
    System.out.println("konstruktor A dijalankan");
    }
}
```

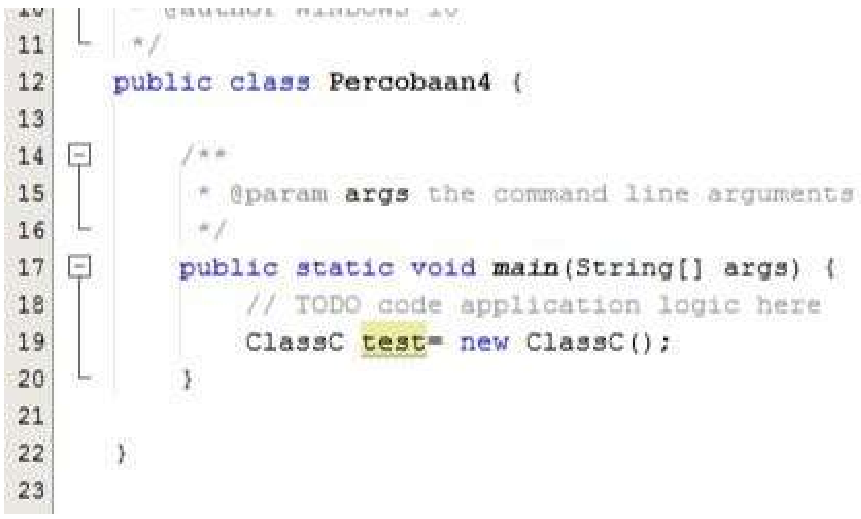
ClassB.java



```
 *
 * @author WINDOWS 10
 */
public class ClassB extends ClassA {
    ClassB(){
    System.out.println("konstruktor B dijalankan");
    }
}
```

ClassC.java

2. Make a class Percobaan4.java to run the program above!

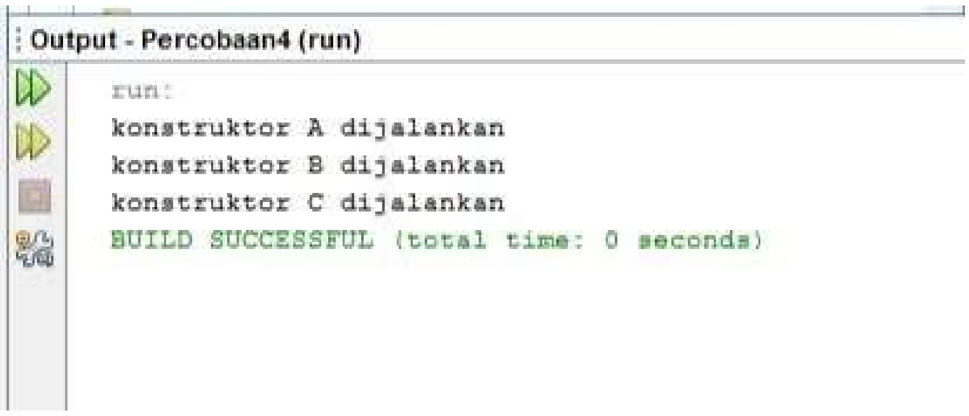


```
 * @author WINDOWS 10
 */
public class Percobaan4 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        ClassC test= new ClassC();
    }

}
```

3. Run the program then observe what happens!



Output - Percobaan4 (run)

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

B. QUESTIONS

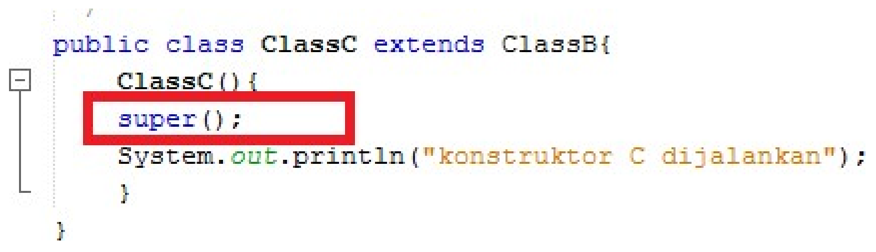
1. In experiment 4 state which class includes the superclass and subclass, then explain the reason!

……………………………………… ……………………………………… ………………………………………

……………………………………… ……………………………………… ………………………………………

....

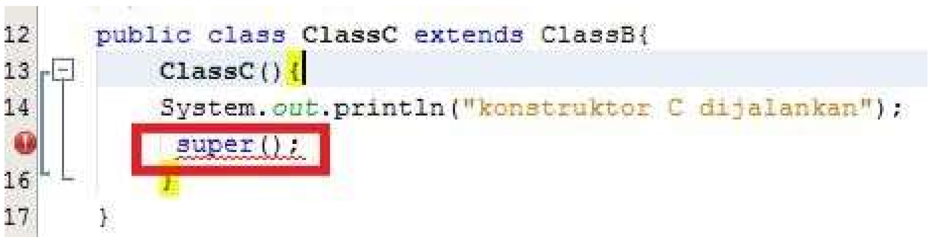
2. Change the contents of the ClassC default constructor as follows:



```
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!

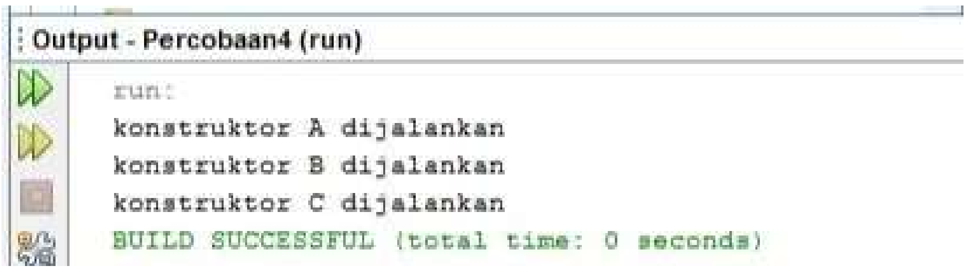
3. Define the contents of the ClassC default constructor as follows:



```
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.

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 class



Output - Percobaan4 (run)

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

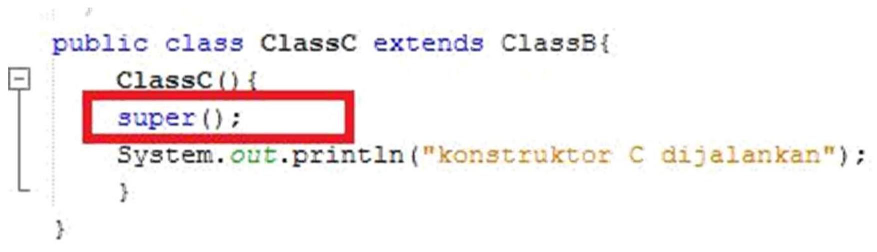
Explain how the order of the constructor goes when the test object is created!

....

....

....

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");
    }
}
```

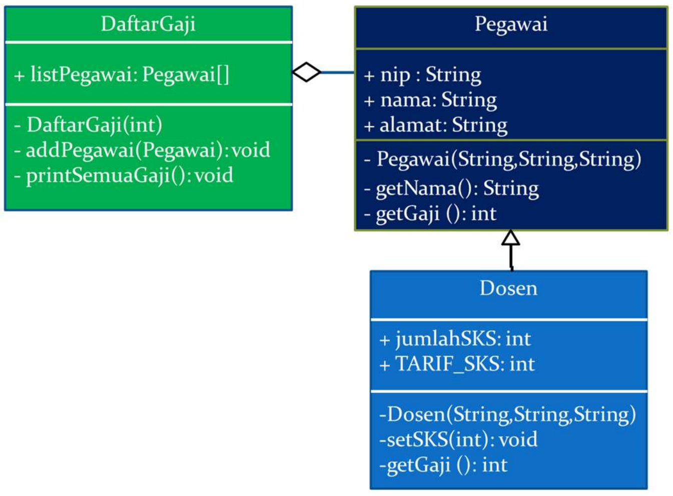
....

....

....

10. ASSIGNMENTS

1. 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.



DaftarGaji
+ listPegawai: Pegawai[]
- DaftarGaji(int)
- addPegawai(Pegawai): void
- printSemuaGaji(): void
Pegawai
+ nip : String
+ nama: String
+ alamat: String
- Pegawai(String,String,String)
- getNama(): String
- getGaji (): int
Dosen
+ jumlahSKS: int
+ TARIF_SKS: int
-Dosen(String,String,String)
-setSKS(int): void
-getGaji (): int