# JOBSHEET W06

**INHERITANCE**

# COMPETENCE

* 1. Understand the basic concept of inheritance.
  2. Able to create a subclass of a certain superclass.
  3. Able to implement the concept of hierarchical inheritance
  4. Able to create objects from a subclass and access attributes and methods either own or derived from their superclass.

# INTRODUCTION

**Inheritance** in object oriented programming is the concept of **inheritance** from a more general class to a more specific class. The class that is derived is called the base class ( **base class**/**super class/parent class**), while the class that is derived is called a derived **class** (**sub class**/**child class**). Each **subclass** will "inherit" the attributes and methods of the public *or* protected *superclass*. The benefit of inheritance is *reusability* or reuse of lines of code.

In the Java programming language, inheritance declarations are made by adding the **extends keyword** after the class name declaration, followed by the parent class-‐name. The extends keyword tells the Java compiler that we want to do **an extension/extension** of the class. Here is an example of an inheritance declaration.

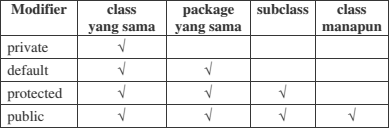
public class B extends A {

...

}

The example above tells the Java compiler that class B is extending class A. This means that class B is a subclass of class A by extension. This extension will be done by adding special attributes and methods that only class B has.

A parent class can limit the attributes and methods that will be inherited to its subclasses . The restriction is carried out through the determination of access level modifiers. In Java, the access level modifier attributes and methods are summarized in the following table:



Attributes and methods that will be inherited from parent class to child class are attributes and methods with a protected or public modifier.

The keyword **this** is used to refer to the current object/class. While the **super** keyword is used to refer to the parent object/class. The writing format is as follows:

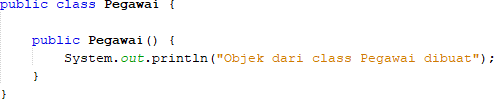
### super.<nameAttributes>

Accessing parent attributes

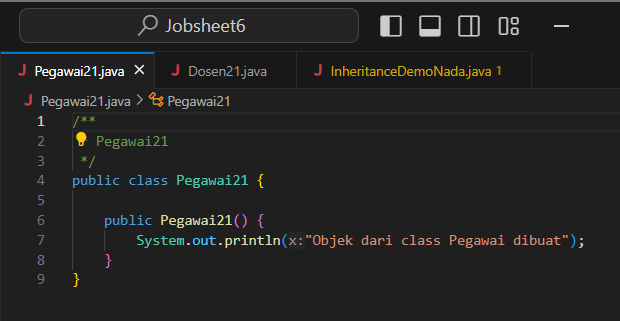
### super.<nameMethod>()

Calling the parent method

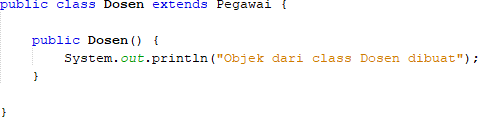
1. **EXPERIMENT 1 (extends)**
   1. **TRIAL STAGES**
      1. Create a parent class with the name of the Pegawai. Then create a parameterless constructor with the following line of code:



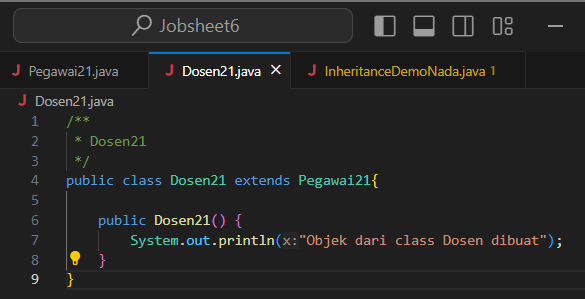
My Answer :



* + 1. Create a subclass of the Pegawai class with the name Dosen, then also create a parameterless constructor with the following line of code:



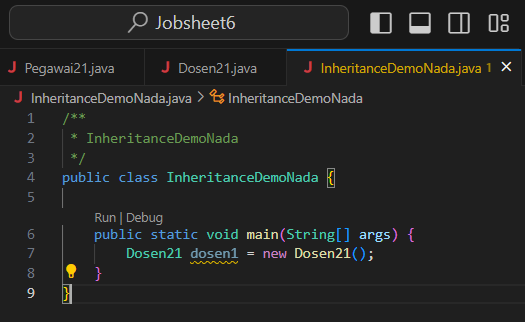
My Answer :



* + 1. Create a main class, for example InheritanceDemo.java, instantiate a new object named dosen1 from the lecturer class as follows:

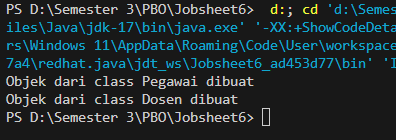


My Answer :



* + 1. Run the program and then observe the results.

My Answer :



* 1. **QUESTION**
     1. In experiment 1 above, determine the child class and parent class!

My Answer :

* Parent class : Pegawai21
* Child class : Dosen21
* Run the progam : InheritanceDemoNada
  + 1. What keywords make the child class and parent class have a relationship?

My Answer :

The keyword used to create a relationship between the two classes is the word “extends Pegawai21” which is in the Lecturer21 class.

* + 1. Based on the results displayed by the program, how many constructors are executed? Which constructor class is executed first?

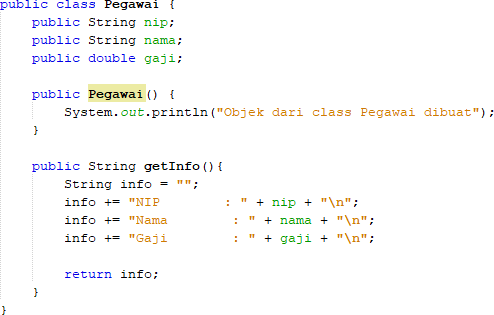
My Answer :

The first constructor to be executed is the constructor of the Employee class, after that the constructor of the lecturer class.

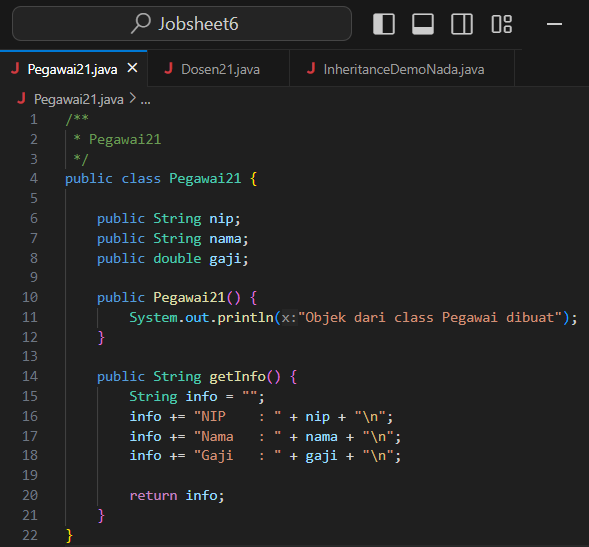
# EXPERIMENT 2 (Inheritance)

## TRIAL STAGES

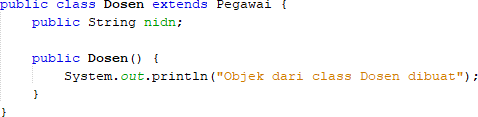
* + - * 1. Add the nip, nama, and gaji attributes and the getInfo() method to the Pegawai class



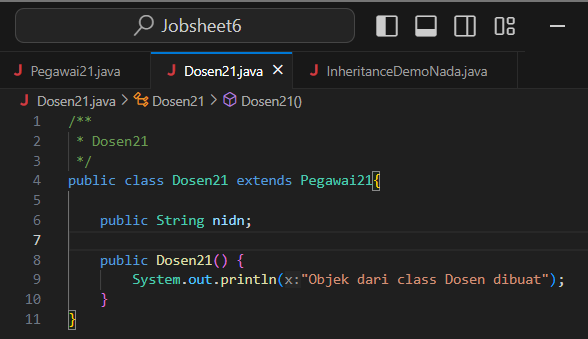
My Answer :



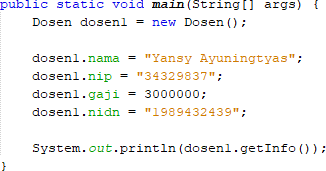
* + - * 1. Also add the NIDN attribute to the Dosen class



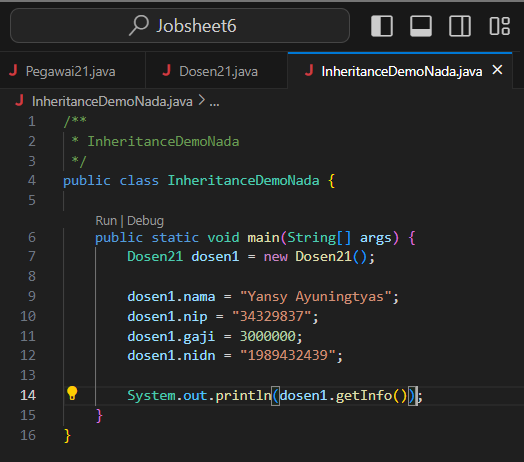
My Answer :



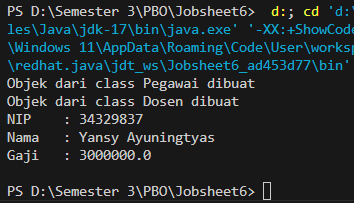
* + - * 1. In class InheritanceDemo.java write the following line of code:



My Answer :



* + - * 1. Run the program then observe the results

My Answer :

## QUESTION

* + - * 1. In experiment 2 above, can the program run successfully or does an error occur?

My Answer :

The program code in my experiment 2 can be executed and the output results can appear, no errors.

* + - * 1. If the program is successfully executed, why is there no error in the assignment/filling in the values of the nip, gaji, and NIDN attributes on the lecturer object1 even though there is no declaration of these three attributes in the lecturer class?

My Answer :

In this experiment, the reason is that the filling such as nip, name, salary and nidn in the Dosen21 class is a subclass of the Pegawai21 class, so that when it is run, no errors occur and can show the output from the input that we have entered.

* + - * 1. If the program is successfully executed, why is there no error in the call of the getInfo() method by the lecturer1 object even though there is no getInfo() method declaration in the dosen class?

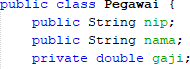
My Asnwer :

The program does not generate an error when it does not call the getInfo() method in the Dosen21 class because the Dosen21 class is an inheritance of the getInfo() method in the Pegawai21 class, so no errors occur when running it.

# EXPERIMENT 3 (Access rights)

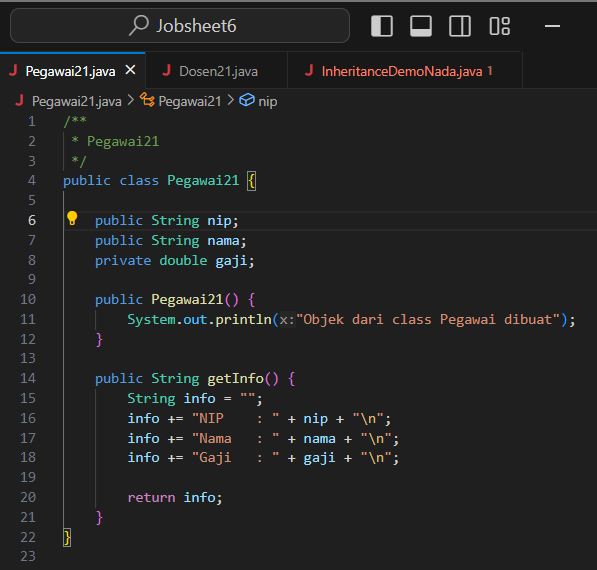
## TRIAL STAGES

* + 1. Modification of access level modifier on gaji attributes to private in class Pegawai.java



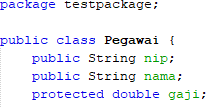
* + 1. Run the program then observe the results.

My Answer :

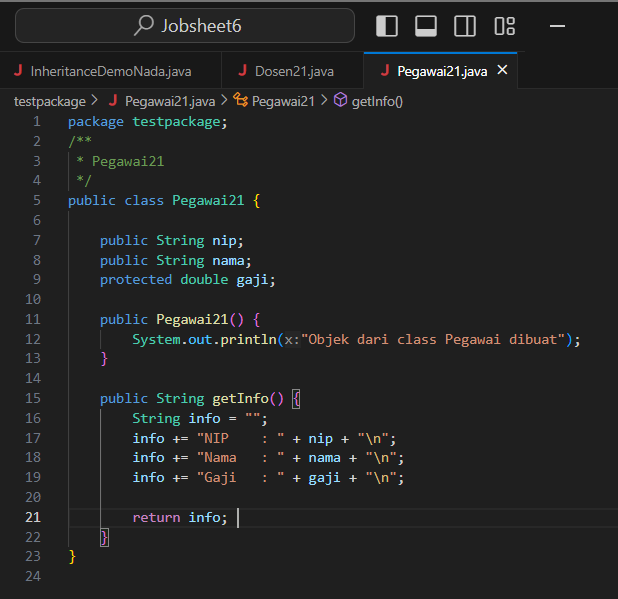


There is an error in the InheritanceDemoNada class which causes the program to not run because it cannot be accessed directly in other classes because it is private. if we want it not to error, we have to add setters and getters for salary in my Pegawai21 class, so that later the program code can be run and not error.

* + 1. Change the access level modifier of the gaji attribute to protected and then move the Pegawai class to a new package, for example "testpackage".



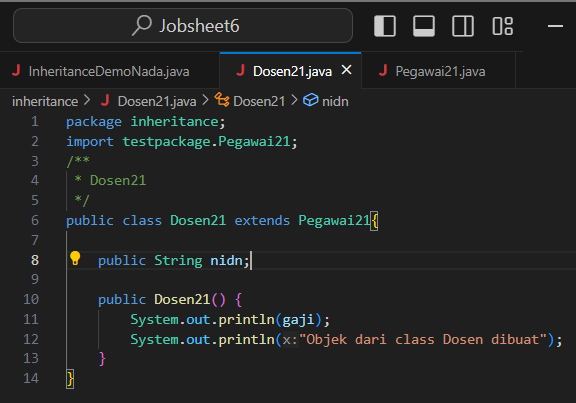
My Answer :



* + 1. Import the Pegawai class from the testpackage in the Dosen class.



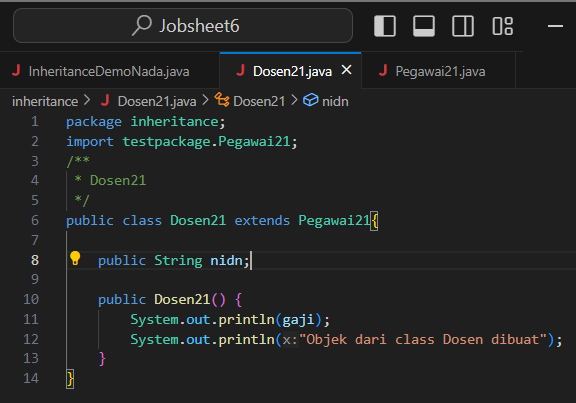
My Answer :



* + 1. Access salary attributes in the Dosen class by trying to print the gaji attributes in the Lecturer constructor

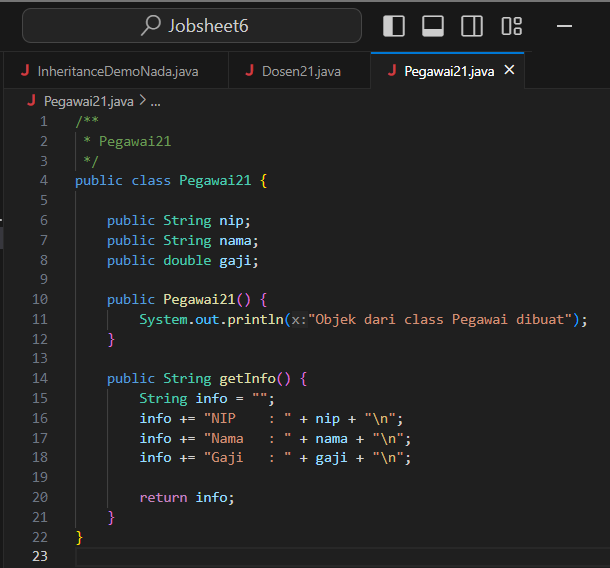


My Answer :



* + 1. Change the access level modifier back to public and revert the Pegawai class to the original package.

My Answer :



## QUESTION

* + 1. In step 1 above, an error occurred because the dosen object1 could not access the gaji attributes. Even though gaji is an attribute of an pegawai who is the parent class of the dosen. Why does this happen?

My Answer :

if the lecturer object cannot access the salary attribute in the Pegawai21 class, then the modifier we give is not appropriate and causes an error. the salary attribute in the Dosen21 class must be declared to be protected first so that it can be accessed by the Dosen21 class. if we declare the salary to be private, then the Dosen21 class cannot access it. and we also have to make sure between the two classes (Pegawai21 and Dosen21) are in the appropriate package.

* + 1. In step 5, after the Pegawai class moves to a different package, the Dosen class can still access the gaji attributes. Why?

My Answer :

because after the Pegawai21 class goes to a different package, the Dosen21 class can still access the salary attribute in its class because the Pegawai21 class has a protected modifier that can be accessed by subclasses in different packages.

* + 1. Based on the experiment, how to determine the attributes and methods that will be inherited by the parent class to the child class?

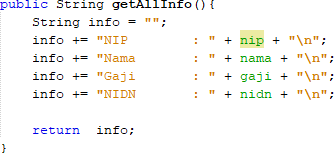
My Answer :

by identifying attributes and methods that are common to both classes, then we can use the appropriate modifiers so that they can be accessed in other classes (such as public and protected). and we can use the extends keyword in the child class.

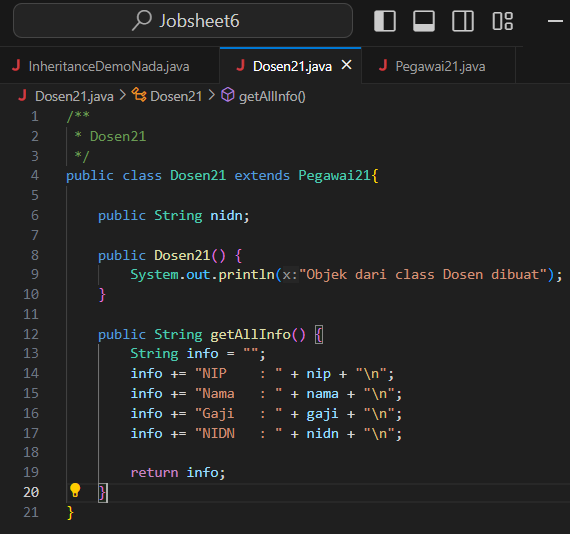
# EXPERIMENT 4 (Super - attributes)

## TRIAL STAGES

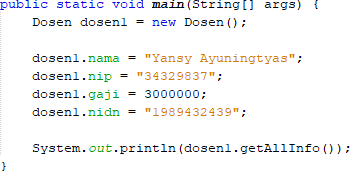
* + 1. Use the getAllInfo() method in the Dosen class



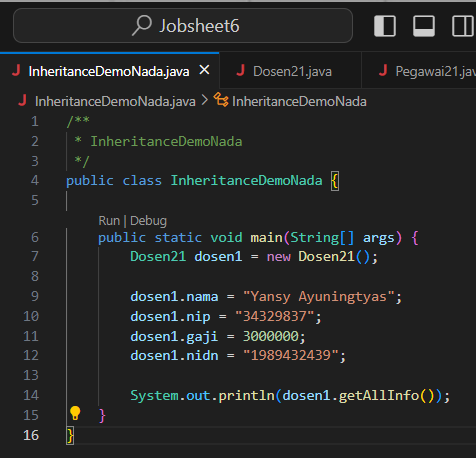
My Answer :



* + 1. Call the getAllInfo() method by the dosen1 object on class InheritanceDemo.java

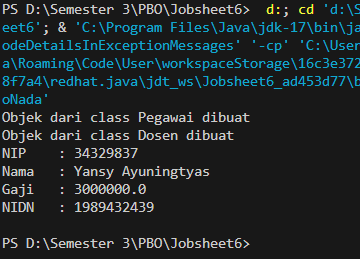


My Answer :

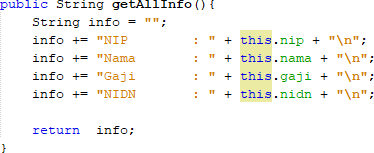


* + 1. Run the program then observe the results

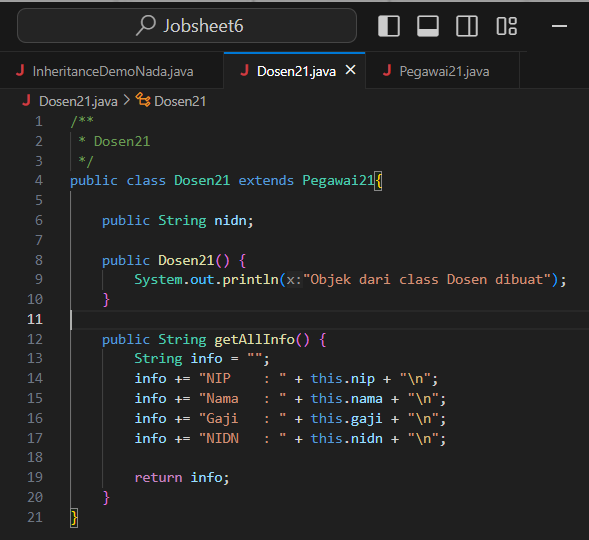
My Answer :



* + 1. Modify the getAllInfo() method in the Dosen class

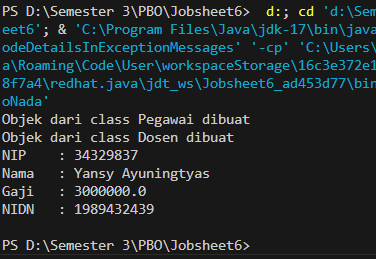


My Answer :



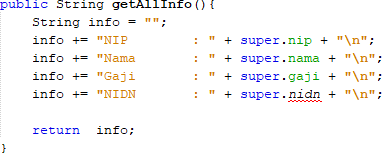
* + 1. Run the program then compare the results with step no 2.

My Answer :

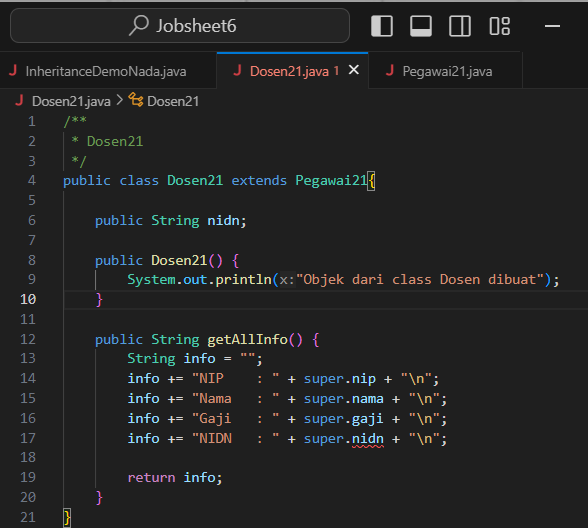
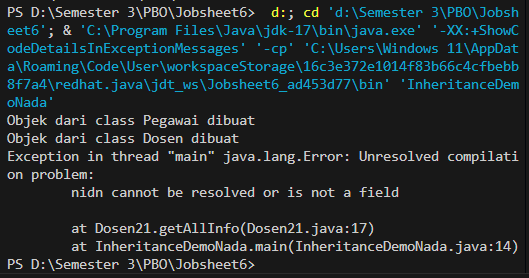


For the difference with setp no. 2, it's just the word “this”, where in the initial experiment there was no word “this” in the getAllInfo() section, while after modification there is the word “this”. Then for the output results are the same.

* + 1. Modify the getAllInfo() method on the Dosen class again



* + 1. Run the program and then compare the results with the program at no 1 and no 4.

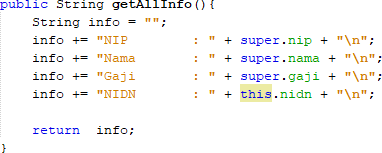
My Answer :

The difference is:

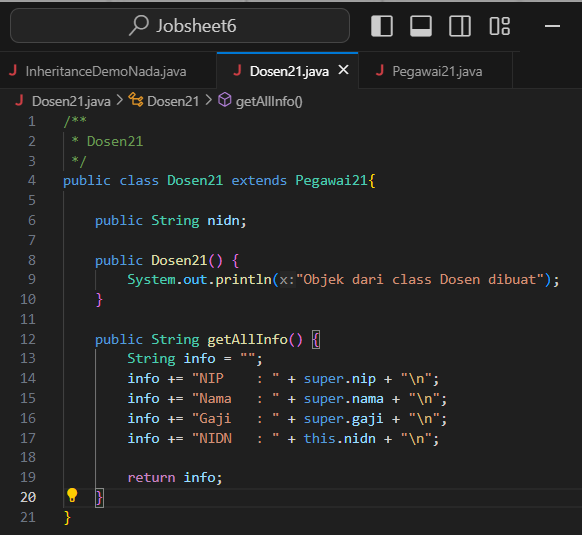
* In experiment no. 1, there is no additional word in the getAllInfo() method, so the program code does not have errors and can be run by producing output.
* Then if in experiment no 4 we add the word “this” to the getAllInfo() method which is used to access the attributes and methods of the object itself.

Then in this experiment, I added the word “super” used to access members of the parent class.

* + 1. Modify the getAllInfo() method on the Dosen class again

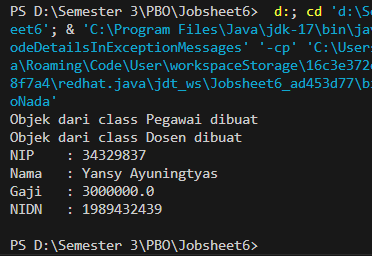


My Answer :



* + 1. Run the program and then compare the results with the program at number 2 and number 4.

My Answer :



## QUESTION

* + 1. Are there any differences in the results of nama, nip, and gaji displayed in programs 1, 4, and 8? Why?

My Answer :

There are differences in the programming code between steps 1, 4, and 8 where the differences are :

* For step 1, it only adds the getAllInfo() method to the Dosen21 class and the addition of NIDN info.
* For step 4, there is the addition of the word “this” in the info attribute section.
* For step 8, there is a change of word to “super” (except the NIDN part still uses this)
* And for the output results, each output (1,4,8) is the same.
  + 1. Why does the error occur in program no 6?

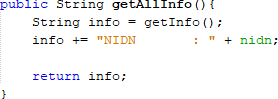
My Answer :

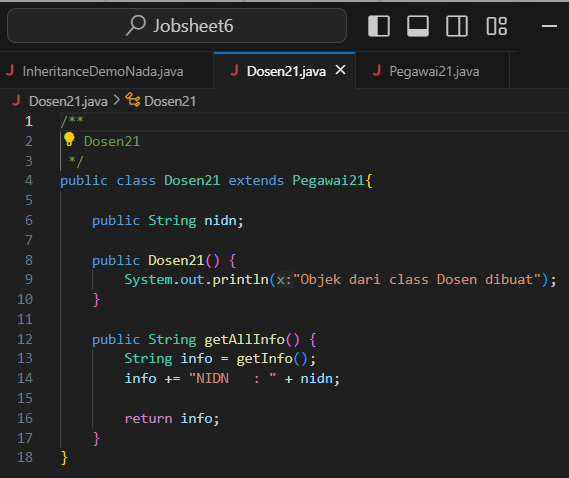
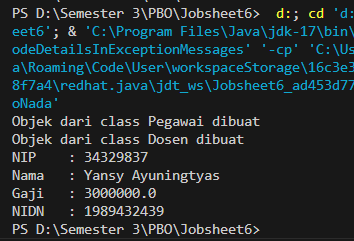
In step 6, there is an error in the nidn section with the word “super” because the use of super is only declared for attributes and methods in the parent class (Pegawai21 class), while nidn is declared in the child class (Dosen21 class).

# EXPERIMENT 5 (super & overriding)

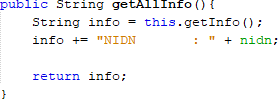
## TRIAL STAGES

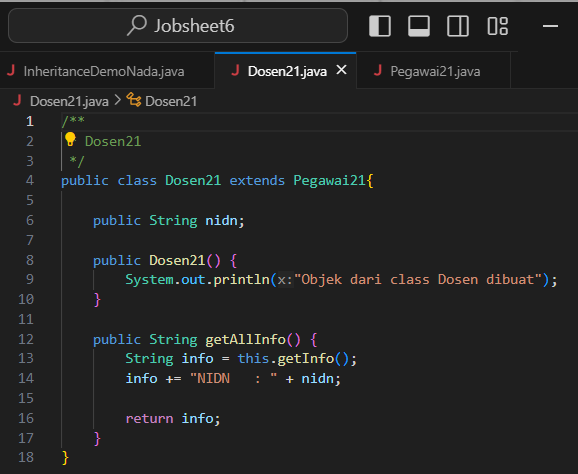
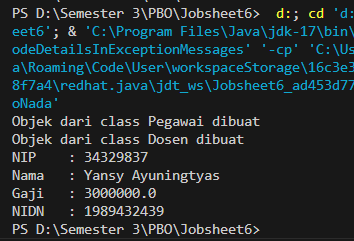
* + 1. Modify the getAllInfo() method again. Run the program then observe the results



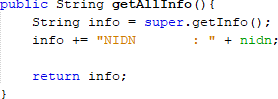
My Answer :

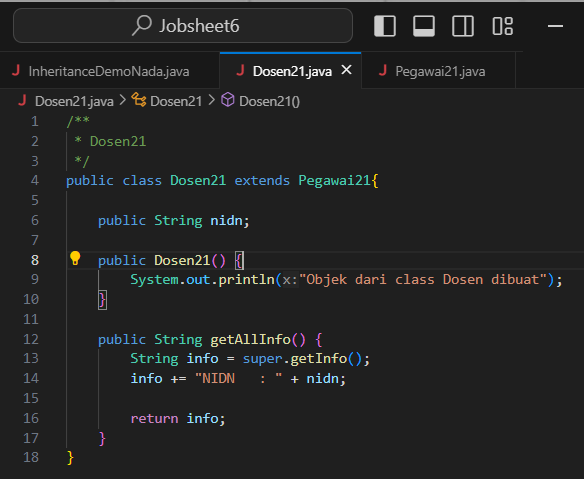
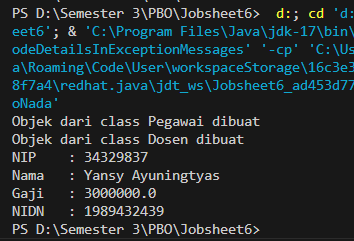
* + 1. Modify the getAllInfo() method again. Run the program then observe the results



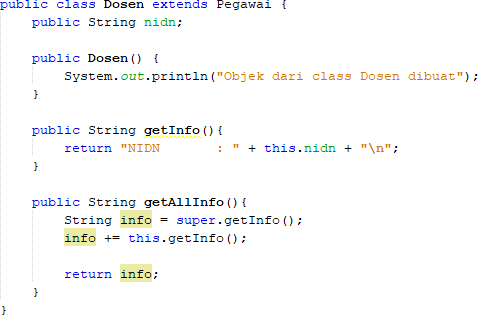
My Answer :

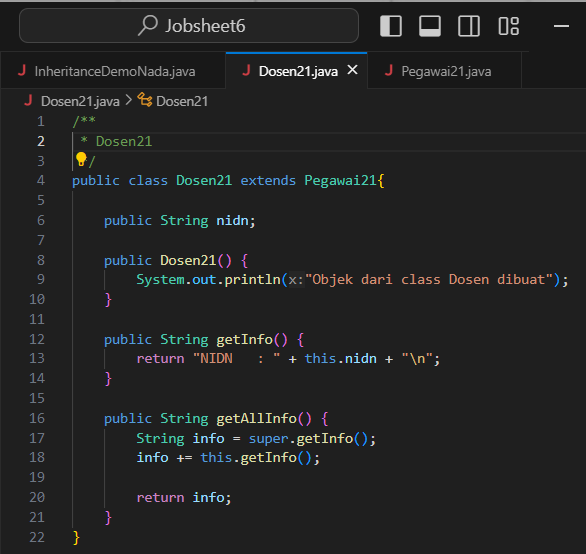
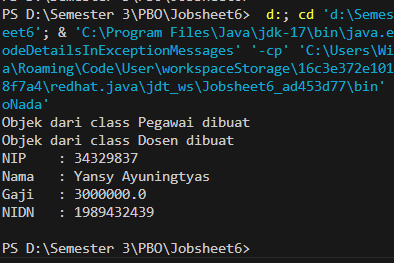
* + 1. Modify the getAllInfo() method again. Run the program then observe the results



My Answer :

* + 1. Add the getInfo() method to the Dosen class and modify the getAllInfo() method as follows



My Answer :

## QUESTION

* + 1. Are there any differences in the getInfo() methods accessed in steps 1, 2, and 3?

My Answer :

There is a difference between steps 1, 2 and 3:

* In the first step, the getAllInfo() method only leaves the nidn attribute (the name, nip, salary attributes are deleted) and the following program code info += “NIDN : “ + nidn;
* As for the second step, in the getAllInfo() method in the info string section we add this, the following program code String info = this.getInfo();
* As for the third step, in the getAllInfo() method in the info string section we change this to super, the following program code String info = super.getInfo();
* The output results are the same for each step.
  + 1. Is there a difference between the super.getInfo() and this.getInfo() methods called in the getAllInfo() method in step 4? Explain!

My Answer :

Yes, there is a difference in the call. super.getInfo() is used to access the implementation of the parent class (Pegawai21 class) and provides details about the attributes of the Pegawai21 class while this.getInfo() accesses the implementation of the child class (Dosen21 class), which focuses on the specific attributes of the class.

* + 1. In what method does overriding occur? Explain!

My Answer :

Overriding is when a method in a subclass has the same name and parameters as a method in the parent class, so there is overriding in the getInfo() method in the Dosen21 class where the Dosen21 class is a subclass of Pegawai21, and both have methods with the same name, namely getInfo().

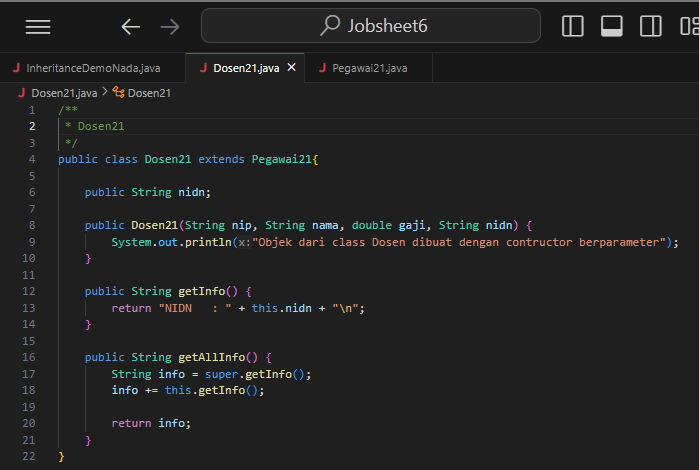
# EXPERIMENT 6 (overloading)

## TRIAL STAGES

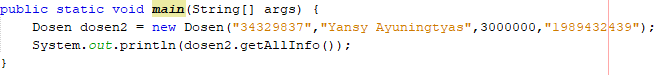
* + 1. Add a new constructor for the Dosen class as follows



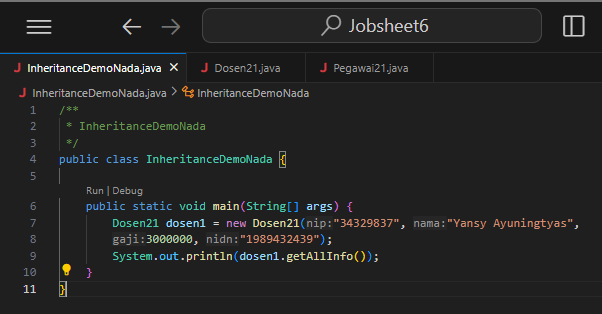
My Answer :

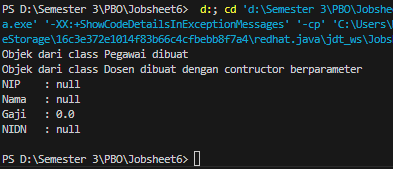


* + 1. Modify the InheritanceDemo class to instantiate a new object with the name lecturer2 with a parameterized constructor. Run the program then observe the results.



My Answer :





## QUESTION

* + 1. What are the results of the nip, nama, gaji, and nidn values displayed in step 2? Why is that?

My Answer :

The result of Step 2 can be seen in the output image that I worked on, why is that? Because the constructor in the Dosen21 class does not initialize the attributes of the Pegawai21 class. So the output that appears is “NULL”.

* + 1. Explain whether the parameterless constructor and the Dosen class constructor created in step 1 have the same signature?

My Answer :

Constructors without parameters and constructors with parameters in the Dosen21 class do not have the same signature. The signature consists of the constructor name and a list of parameters. The parameterless constructor (Dosen21()) has no parameters, while the parameterized constructor (Dosen21 (String nip, String name, double salary, String nidn)) has four parameters, so they are considered different and are constructor overloading.

* + 1. What is the concept in OOP that allows a class to have a constructor or method with the same name and a different signature on a class?

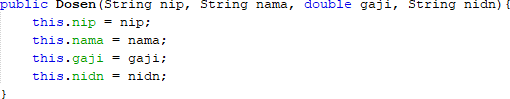
My Answer :

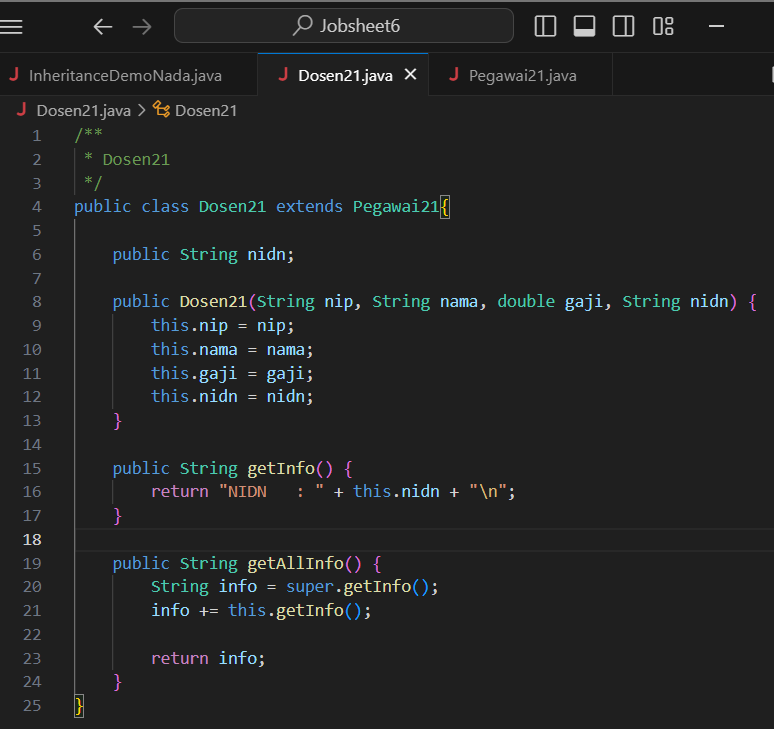
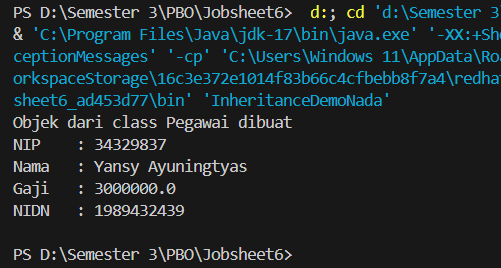
Method overloading. Because in method overloading, we can define multiple methods or constructors with the same name in one class, as long as they have different parameters.

# EXPERIMENT 7 (super-constructor)

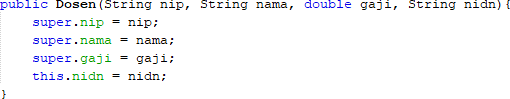
## TRIAL STAGES

* + 1. Constructor modifications in the Dosen class are as follows. Run the program then observe the results.

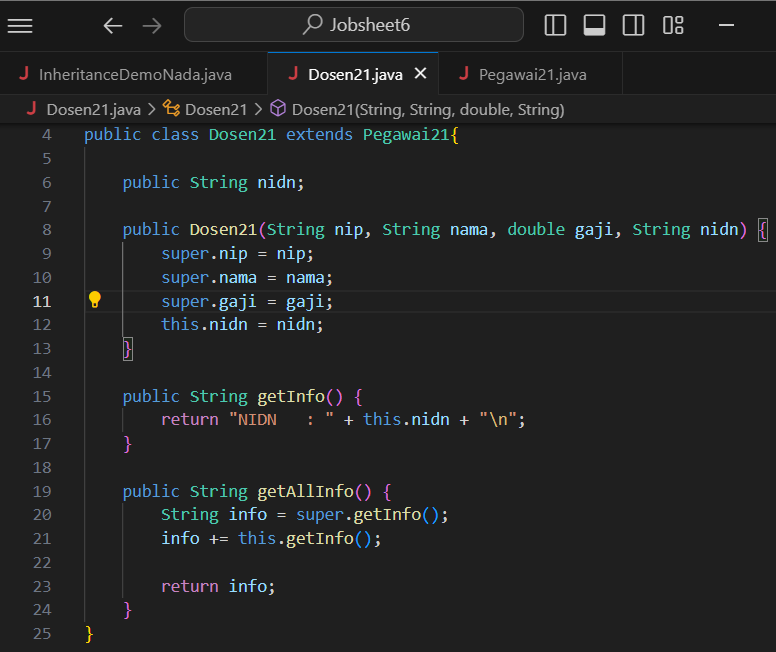


CMy Answer :

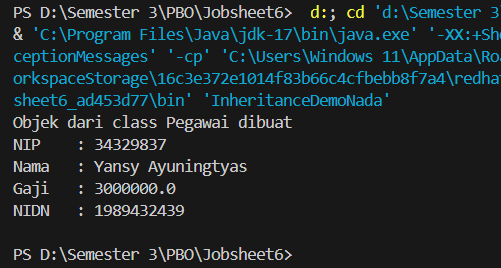
* + 1. Constructor modifications in the Dosen class are as follows. Run the program then observe the results.

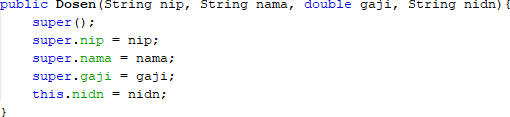


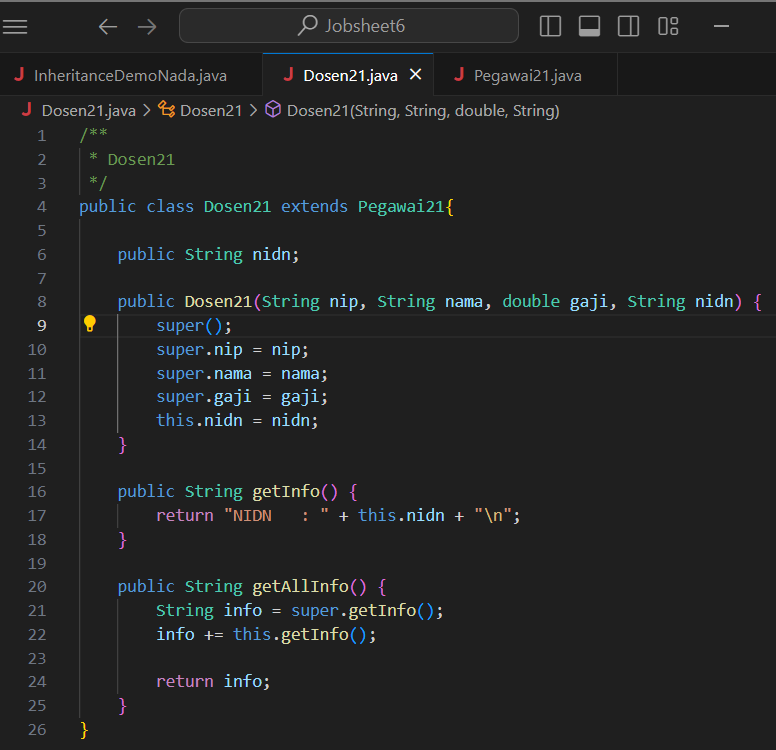
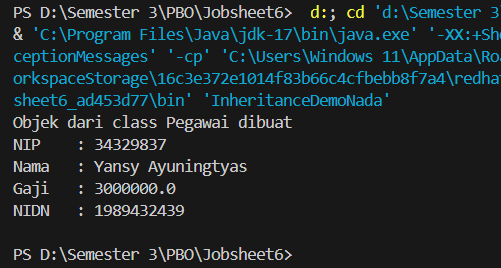
My Answer :



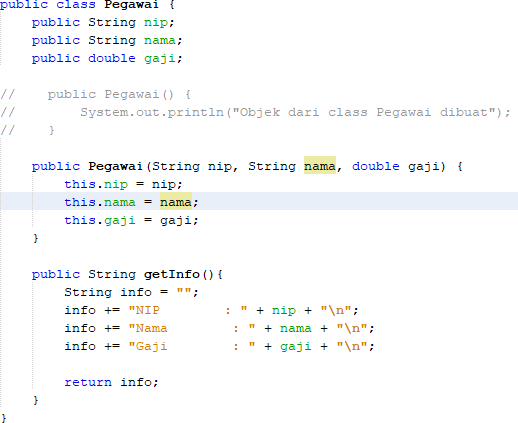
* + 1. Constructor modifications in the Dosen class are as follows. Run the program then observe the results.



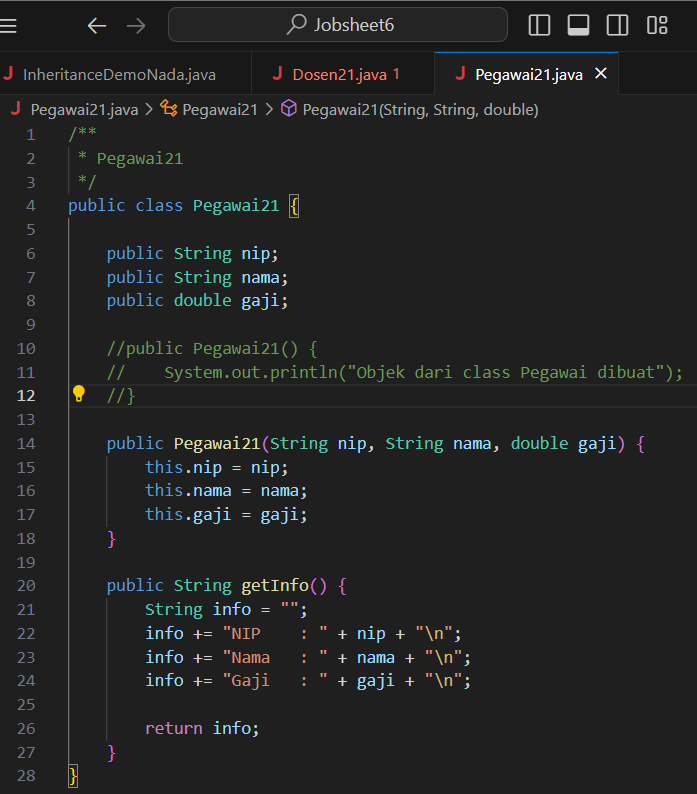


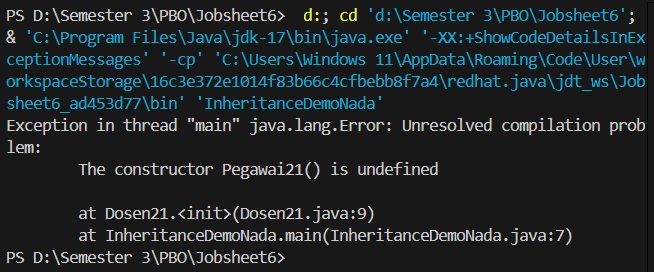
My Answer :

* + 1. Remove/comment constructor without parameters from the Pegawai class. Add a new constructor for the Pegawai class as follows. Run the program then observe the results.

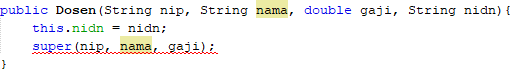


My Answer :

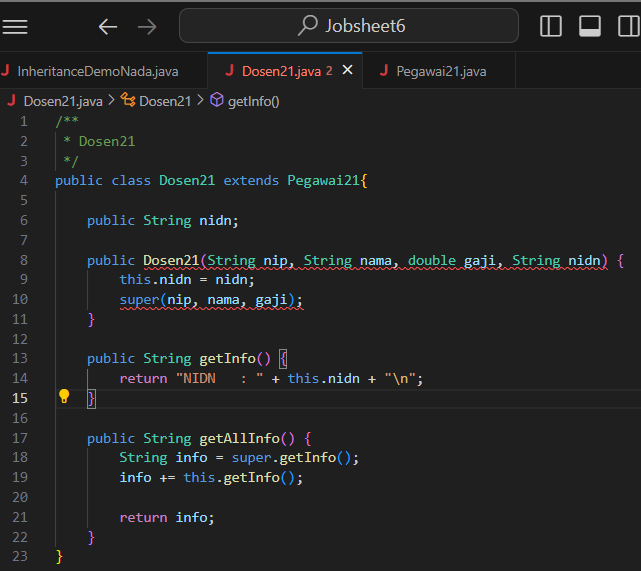


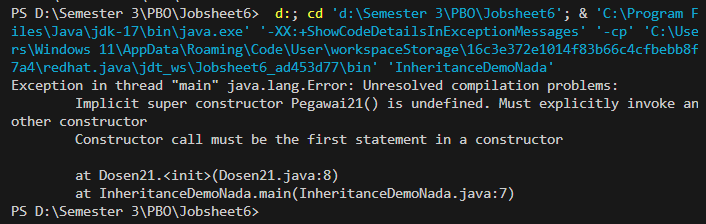


* + 1. Constructor modifications in the Dosen class are as follows. Run the program then observe the results.



My Answer :

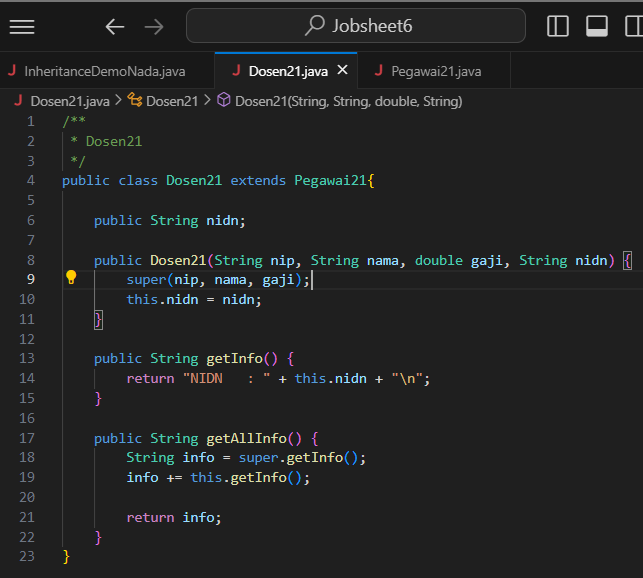


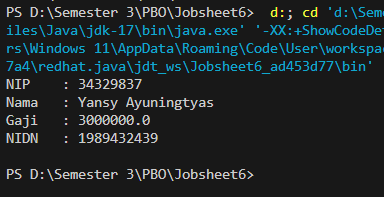


* + 1. Constructor modifications in the Dosen class are as follows. Run the program then observe the results.



My Answer :



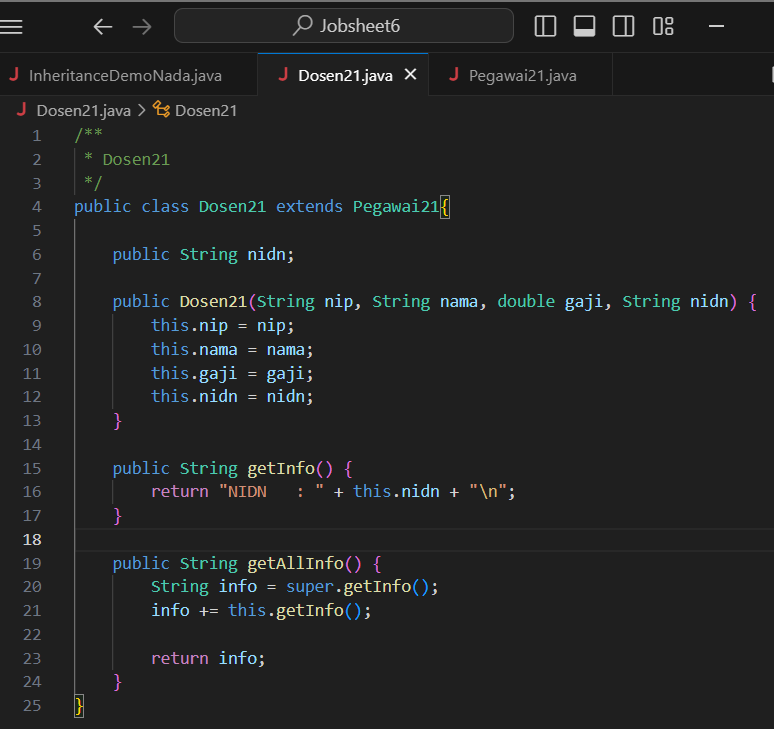
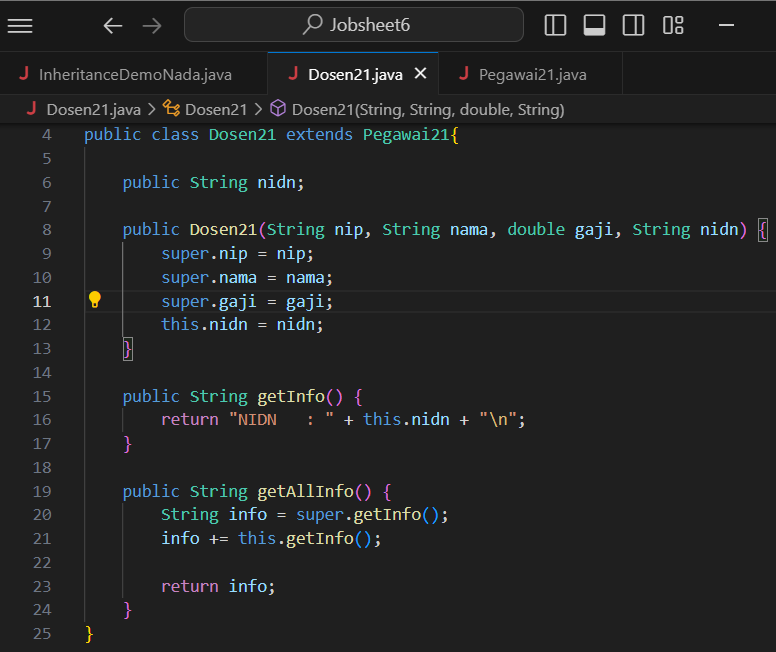
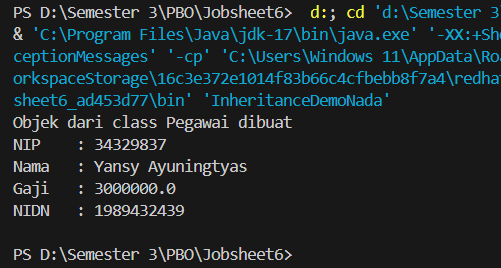


## QUESTION

* + 1. Is there a difference in the results in steps 1 and 2? Explain!

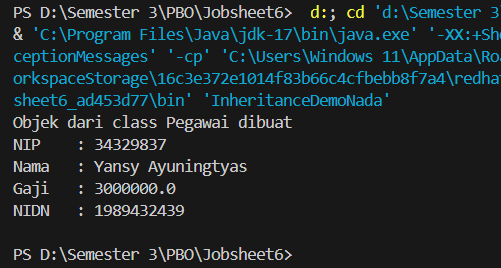
My Answer :

In the experiments in step 1 and 2, there are differences in the input but the output results are the same, here for the explanation:

* In step 1, we modify the parameter constructor part, namely “public Dosen21()” in the Dosen21 class by adding the attributes nip, name, salary, and nidn by adding this in front of it.
*  Then in step 2, we modify again in the parameter constructor section, namely “public Dosen21” in the Dosen21 class with the same attributes as in step 1 but the word this is changed to super (except for the nidn attribute which still uses this).
* from the 2 steps above (step 1 and setp 2) the run results on the output are the same, here are the results of the two steps.
  + 1. Is there a difference in the results in steps 2 and 3? Explain!

My Answer :

From the experiment between step 2 and step 3 there is a difference, here is a brief explanation :

In the experiment between step 2 and step 3 there is not much to modify, for step 2 we only modify the parameter constructor by changing the word this to super (except with the nidn attribute still using this). Whereas in the 3rd step experiment we only add the code “super();” in the parameter constructor in the Dosen21 class, meanwhile for the run output results are the same between the 2nd and 3rd step experiments.

* + 1. Why did the error occur in step 4?

My Answer :

In step 4, the error is because we should call the Employee21 constructor that has parameters, and we cannot call the constructor without parameters that do not exist, so the programming errors and cannot be executed as shown in the figure.

* + 1. What is the difference between super() called in steps 3 and 6?

My Answer :

* The use of (super(nip, name, salary);) : is used to call the constructor of the parent class which initializes the attributes directly and is recommended.
* Usage (super.nip = nip;) : used when we want to change the attribute value after calling the default constructor of the parent class, but less recommended for a cleaner and more structured initialization.
  + 1. Why did the error occur in step 5?

My Answer :

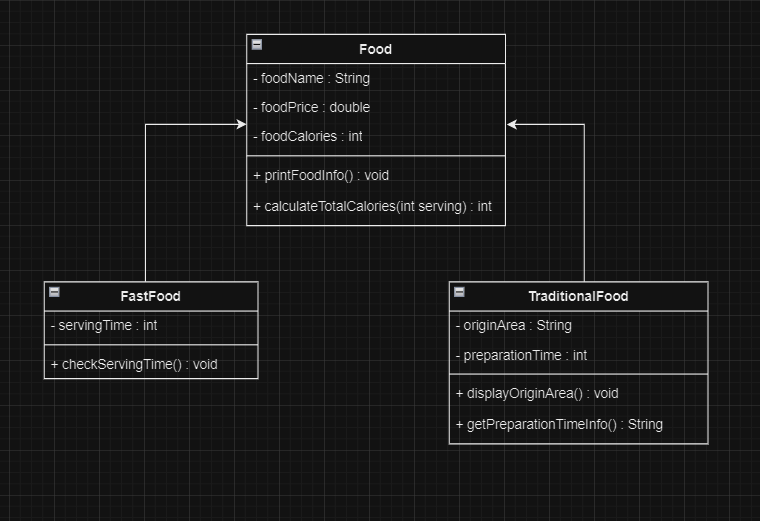
In this 5th step experiment, we modify the parameter constructor part by removing the nip, name, and salary attributes (written with this) but leaving nidn by adding the input super(nip, name, salary);. But the result is an error, why is it an error? Because the constructor of the parent class (Pegawai21) does not exist and the call to super(); is not put as the first statement in the constructor of the child class (Dosen21). So that we should put super()in the section before this.nidn = nidn; so that no errors occur and can be run.

# ASSIGNMENT

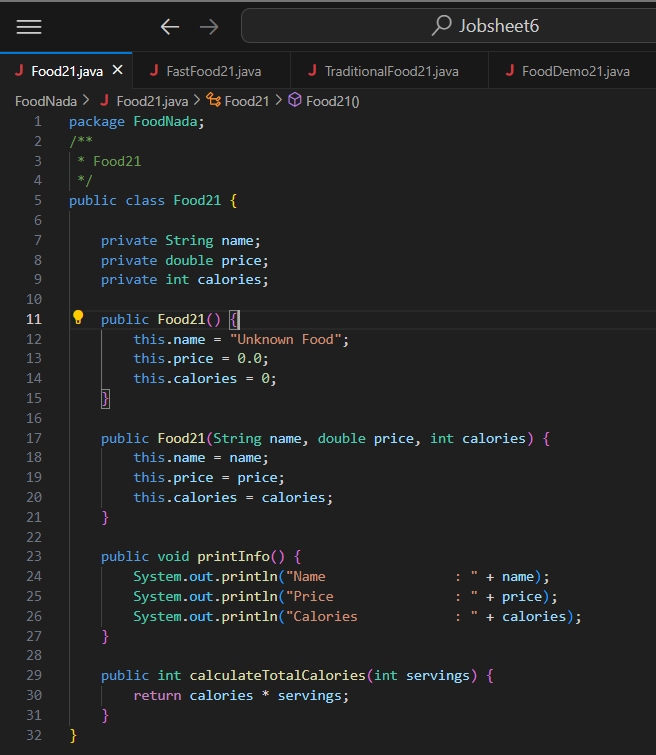
* 1. Define a class that is a derivative of another class.
  2. Create 3 attributes in the parent class then add at least 1 attribute in the child class.
  3. Perform the overloading method by creating 2 constructors, namely a parameterless constructor and a parameterized constructor for each class. Call the parameterized super() constructor to create an object from the parent class on the child class constructor.
  4. Implement the class diagram made in the theoretical PBO course
  5. Create a Demo class then instantiate the child class object in the main function
  6. Try modifying the attribute values (both those declared in the child class and those inherited from the print info.

My Answer :

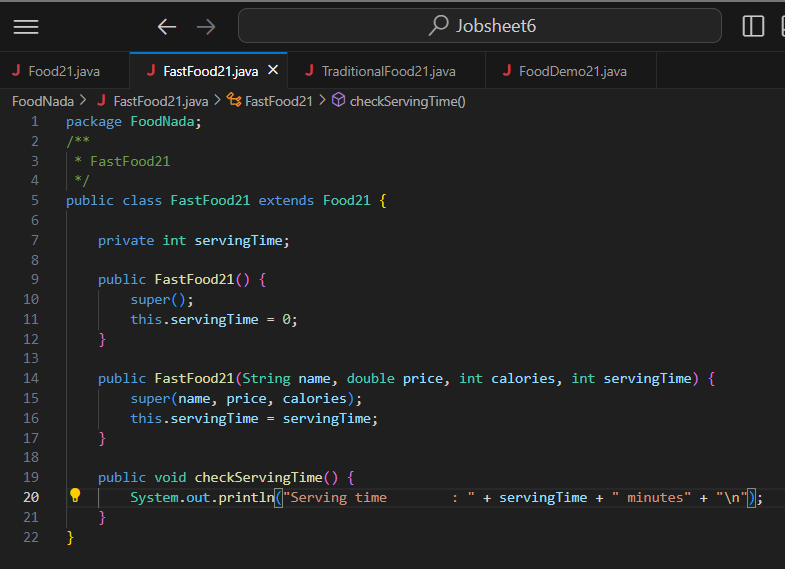
Class Diagram



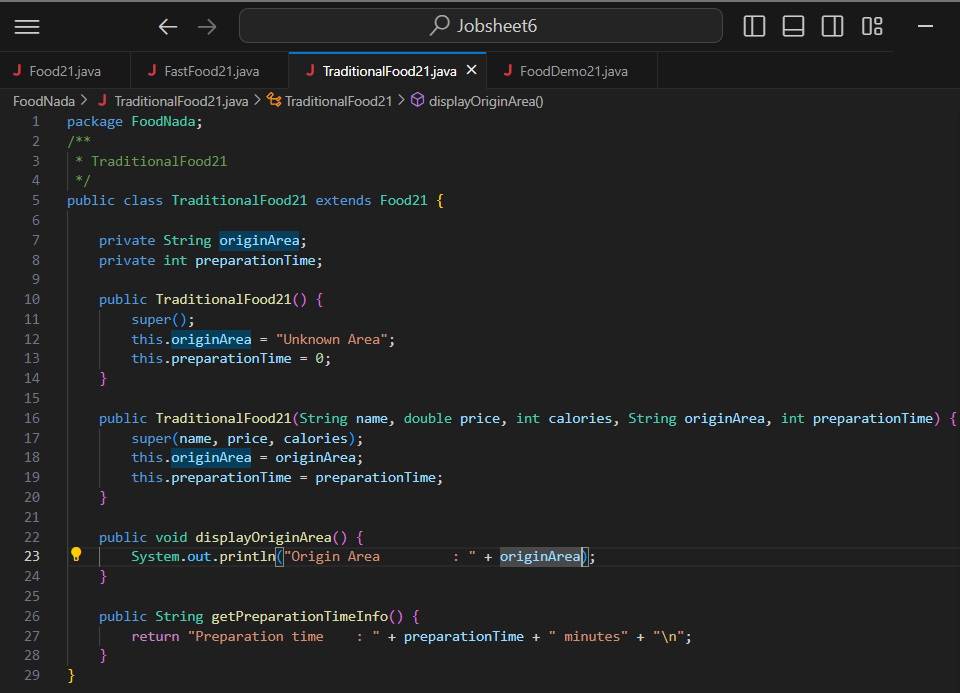
Class Food21



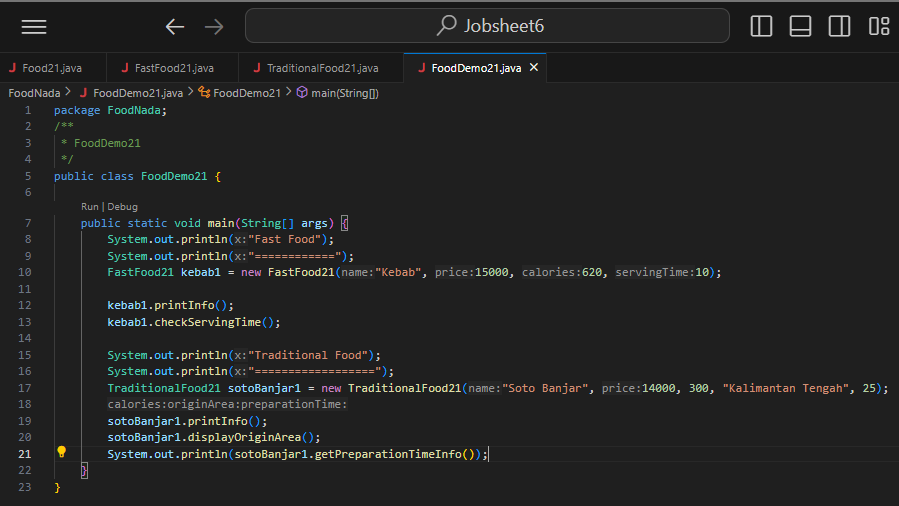
Class FastFood21



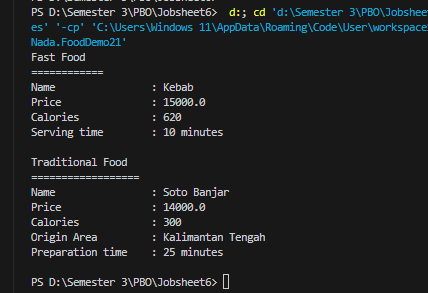
Class TraditionalFood21



Class FoodDemo21



Output



### --- happy working----