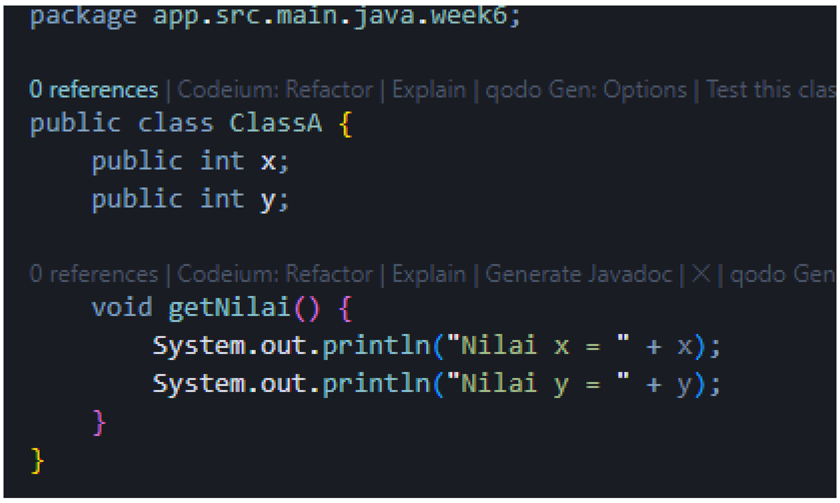
JOBSHEET 6 - INHERITANCE

Kevin Bramasta Arvyto Wardhana

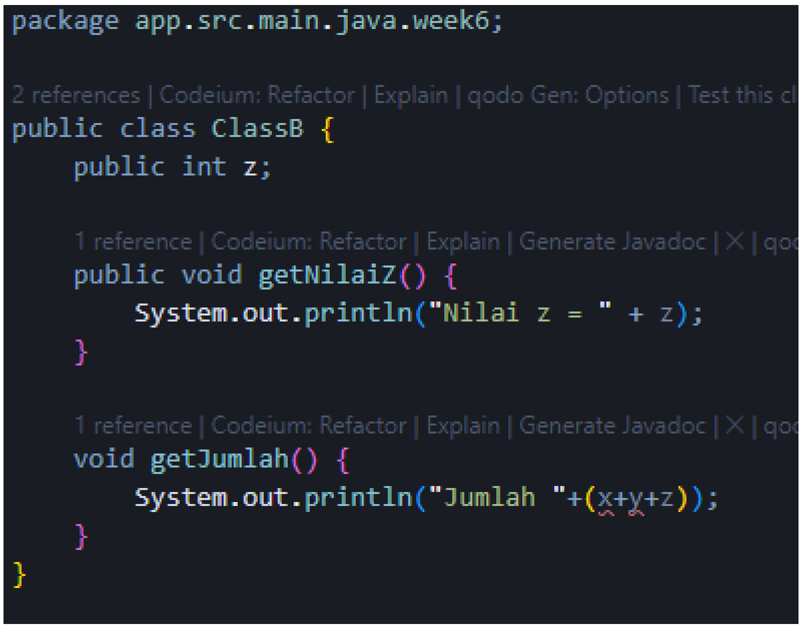
2341720233 / 13

Trial 1

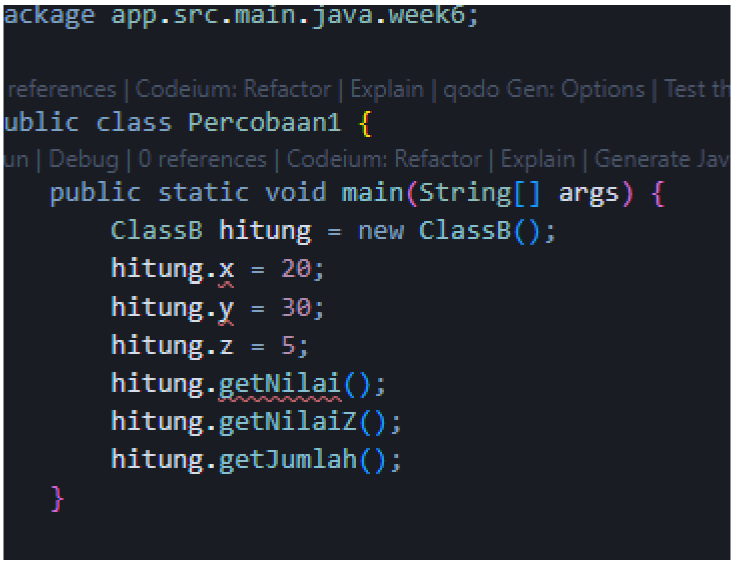
Code:



```java
package app.src.main.java.week6;

0 references | Codeium: Refactor | Explain | qodo Gen: Options | Test this clas
public class ClassA {
    public int x;
    public int y;

    0 references | Codeium: Refactor | Explain | Generate Javadoc | X | qodo Gen
    void getNilai() {
        System.out.println("Nilai x = " + x);
        System.out.println("Nilai y = " + y);
    }
}
```

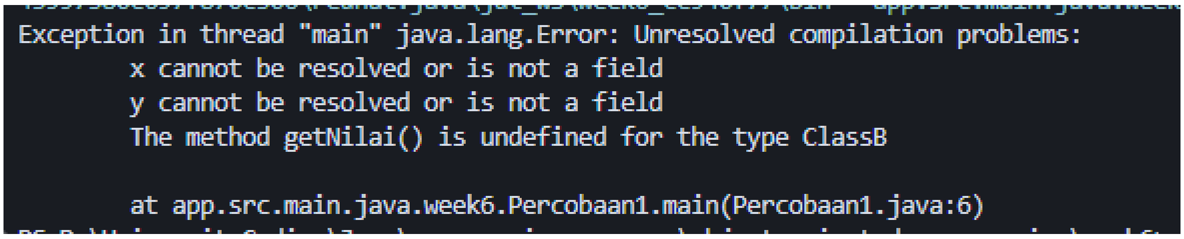


```java
package app.src.main.java.week6;

2 references | Codeium: Refactor | Explain | qodo Gen: Options | Test this cl
public class ClassB {
    public int z;

    1 reference | Codeium: Refactor | Explain | Generate Javadoc | X | qod
    public void getNilaiZ() {
        System.out.println("Nilai z = " + z);
    }

    1 reference | Codeium: Refactor | Explain | Generate Javadoc | X | qod
    void getJumlah() {
        System.out.println("Jumlah "+(x+y+z));
    }
}
```



```
ackage app.src.main.java.week6;

 references | Codeium: Refactor | Explain | qodo Gen: Options | Test th
ublic class Percobaan1 {
un | Debug | 0 references | Codeium: Refactor | Explain | Generate Jav
   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();
   }
```

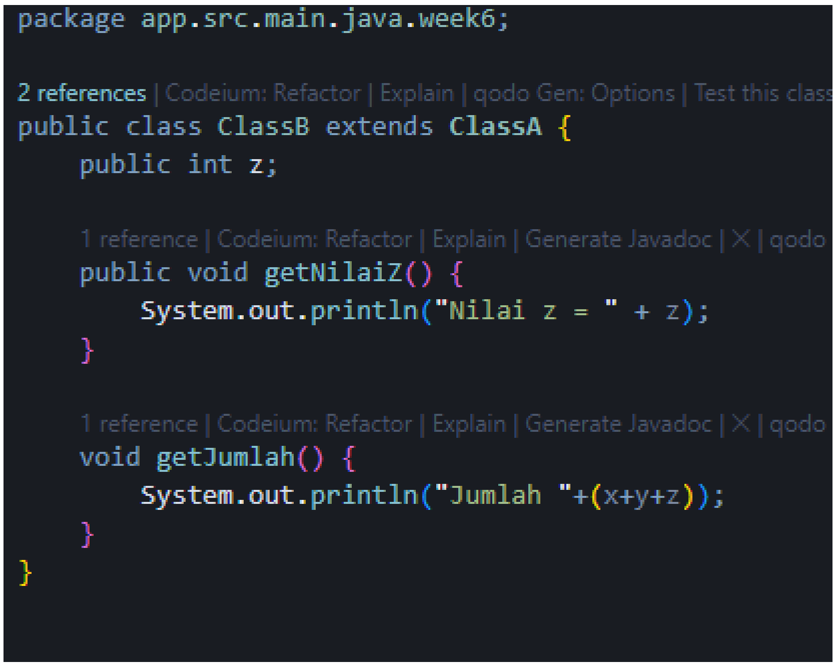
Output:



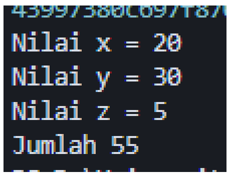
```
Exception in thread "main" java.lang.Error: Unresolved compilation problems:
        x cannot be resolved or is not a field
        y cannot be resolved or is not a field
        The method getNilai() is undefined for the type ClassB

        at app.src.main.java.week6.Percobaan1.main(Percobaan1.java:6)
```

Question

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



```
package app.src.main.java.week6;

2 references | Codeium: Refactor | Explain | qodo Gen: Options | Test this class
public class ClassB extends ClassA {
    public int z;

    1 reference | Codeium: Refactor | Explain | Generate Javadoc | X | qodo
    public void getNilaiZ() {
        System.out.println("Nilai z = " + z);
    }

    1 reference | Codeium: Refactor | Explain | Generate Javadoc | X | qodo
    void getJumlah() {
        System.out.println("Jumlah "+(x+y+z));
    }
}
```



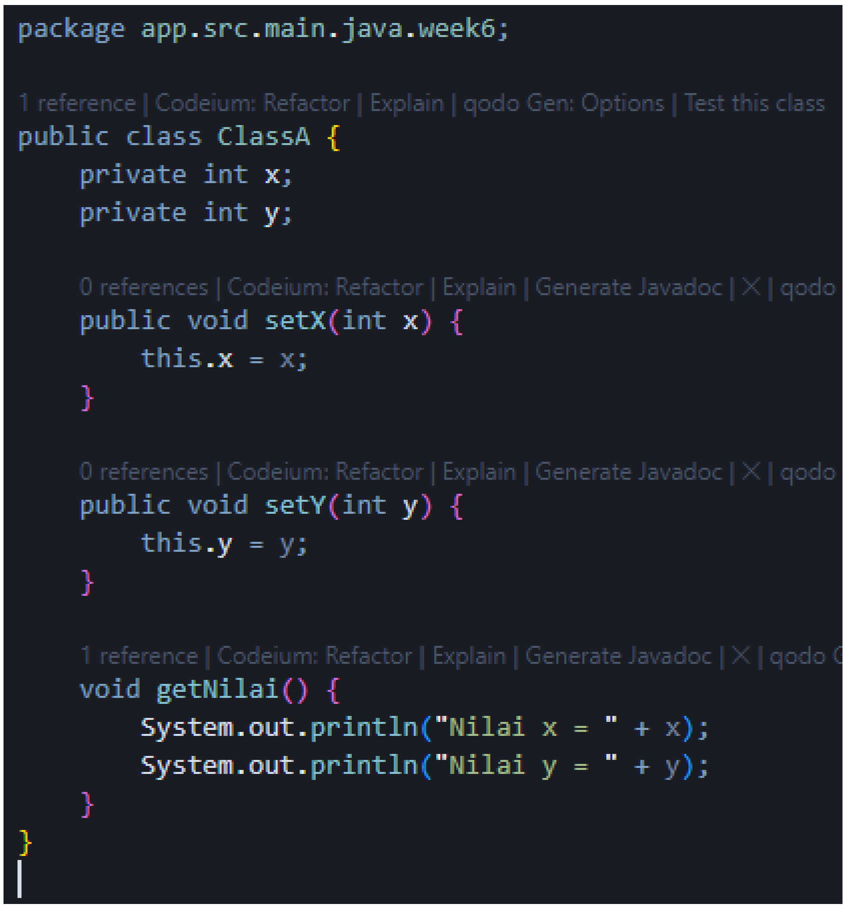
```
Nilai x = 20
Nilai y = 30
Nilai z = 5
Jumlah 55
```

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

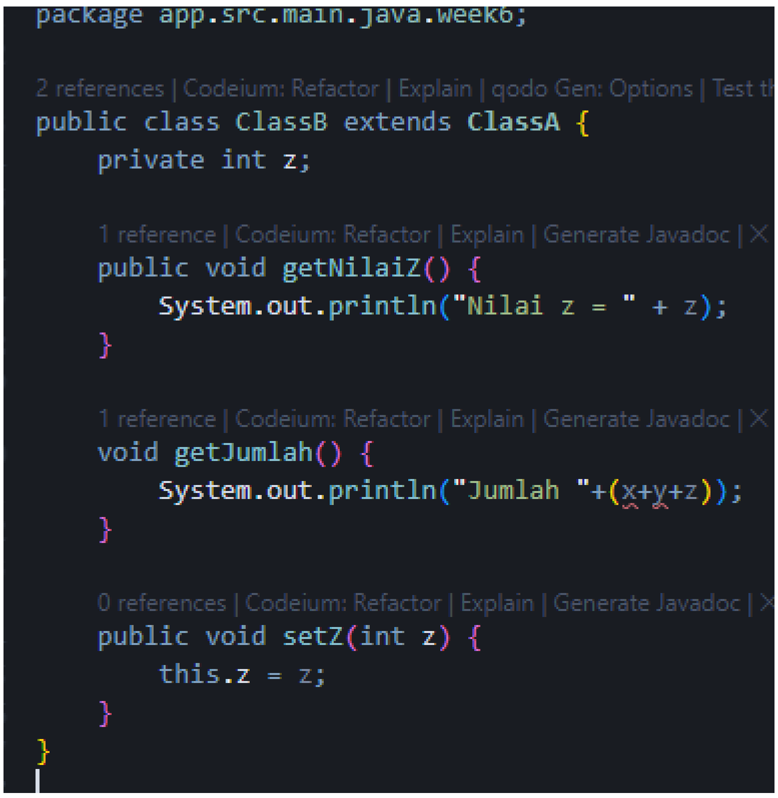
The program encountered an error because the ClassB class did not have direct access to the variables x and y which are defined in ClassA. Since ClassB did not extend ClassA, it did not inherit the variables x and y, leading to compilation errors when trying to access these variables in ClassB.

Trial 2

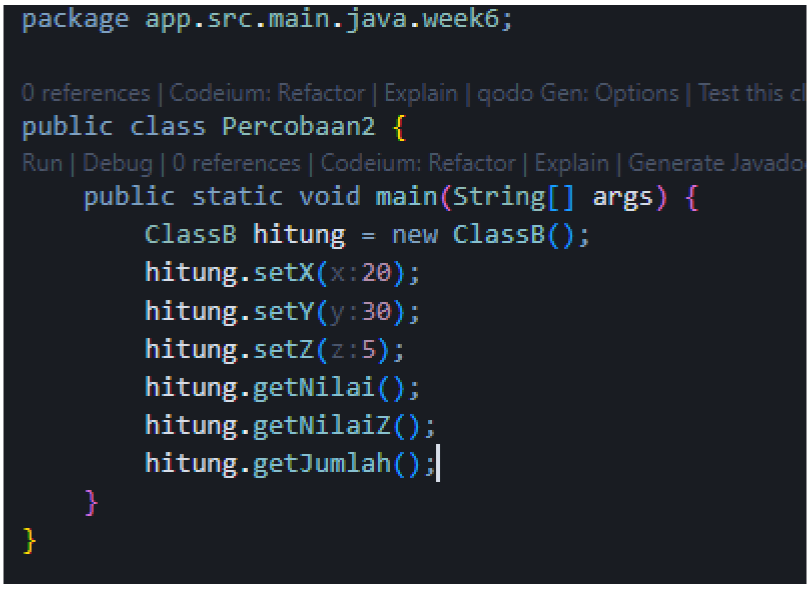
Code:



```java
package app.src.main.java.week6;

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;
    }

    void getNilai() {
        System.out.println("Nilai x = " + x);
        System.out.println("Nilai y = " + y);
    }
}
```

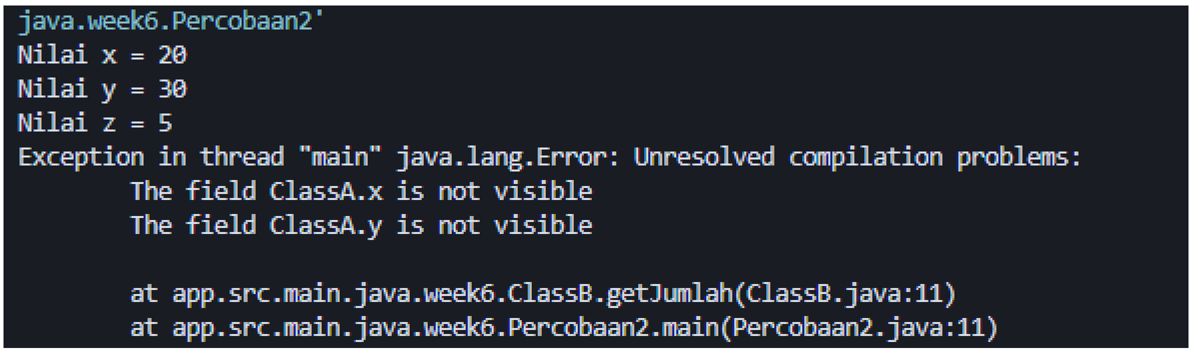


```
package app.src.main.java.week6;

2 references | Codeium: Refactor | Explain | qodo Gen: Options | Test t
public class ClassB extends ClassA {
    private int z;

    1 reference | Codeium: Refactor | Explain | Generate Javadoc | X
    public void getNilaiZ() {
        System.out.println("Nilai z = " + z);
    }

    1 reference | Codeium: Refactor | Explain | Generate Javadoc | X
    void getJumlah() {
        System.out.println("Jumlah "+(x+y+z));
    }

    0 references | Codeium: Refactor | Explain | Generate Javadoc | X
    public void setZ(int z) {
        this.z = z;
    }
}
```



```
package app.src.main.java.week6;

0 references | Codeium: Refactor | Explain | qodo Gen: Options | Test this cl
public class Percobaan2 {
Run | Debug | 0 references | Codeium: Refactor | Explain | Generate Javado
    public static void main(String[] args) {
        ClassB hitung = new ClassB();
        hitung.setX(x:20);
        hitung.setY(y:30);
        hitung.setZ(z:5);
        hitung.getNilai();
        hitung.getNilaiZ();
        hitung.getJumlah();
    }
}
```

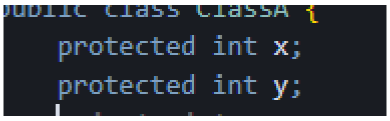
Output:



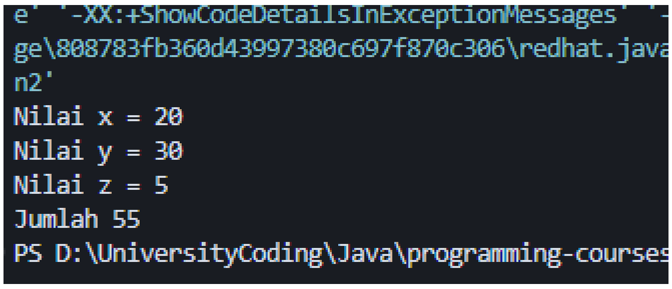
```
java.week6.Percobaan2'
Nilai x = 20
Nilai y = 30
Nilai z = 5
Exception in thread "main" java.lang.Error: Unresolved compilation problems:
        The field ClassA.x is not visible
        The field ClassA.y is not visible

        at app.src.main.java.week6.ClassB.getJumlah(ClassB.java:11)
        at app.src.main.java.week6.Percobaan2.main(Percobaan2.java:11)
```

Question

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



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

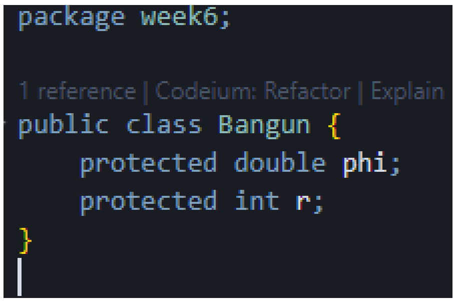


```
e' '-XX:+ShowCodeDetailsInExceptionMessages' '-
ge\808783fb360d43997380c697f870c306\redhat.java
n2'
Nilai x = 20
Nilai y = 30
Nilai z = 5
Jumlah 55
PS D:\UniversityCoding\Java\programming-courses
```

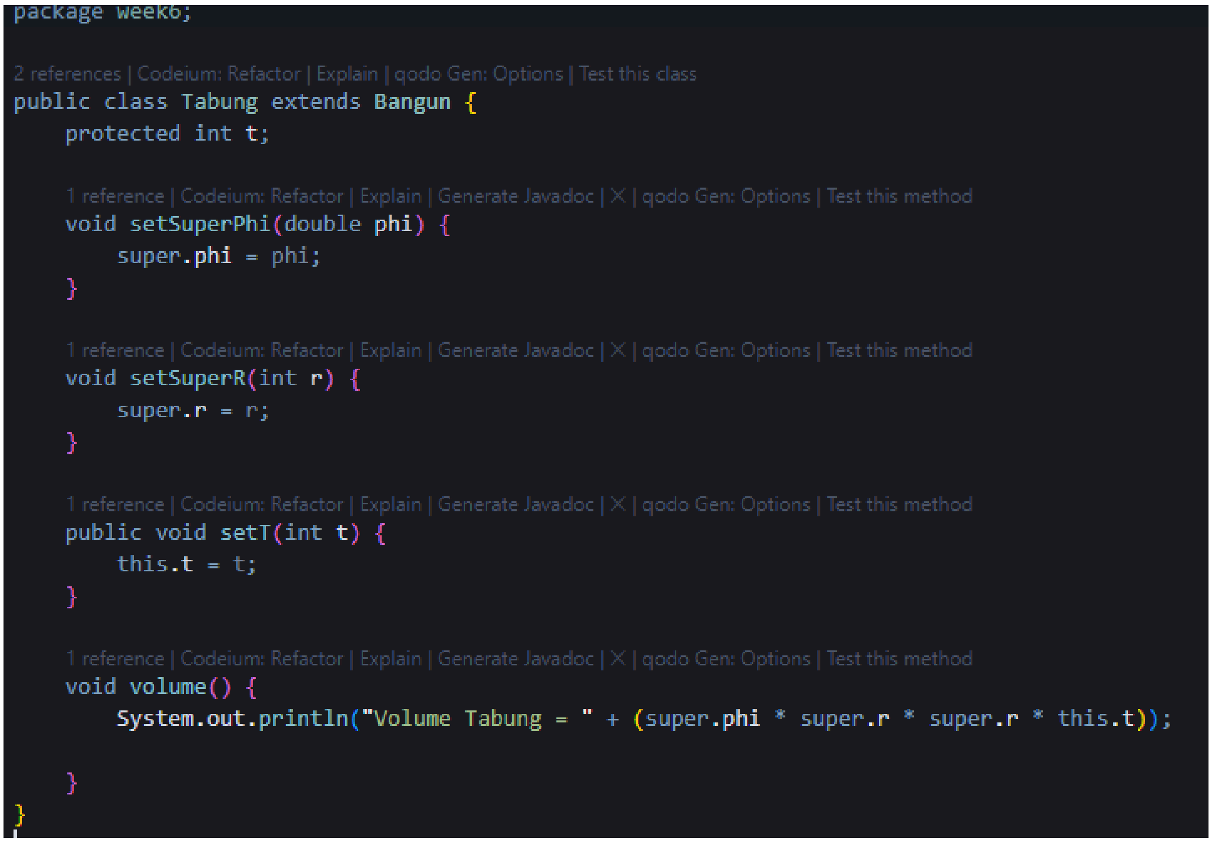
Explain what caused the program in experiment 1 when it ran an error!
Because the variables x and y are not visible in classB class. You need to have a getter or change the modifier into a protected instead of private.

Trial 3

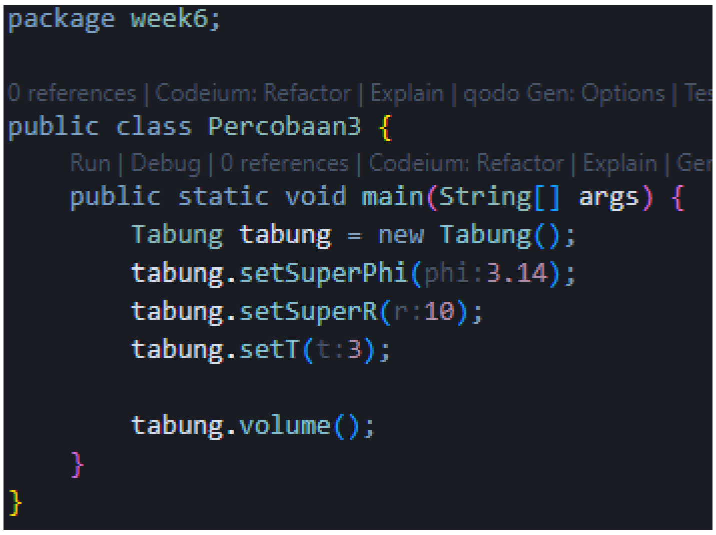
Code:



```
package week6;

1 reference | Codeium: Refactor | Explain
public class Bangun {
    protected double phi;
    protected int r;
}
```

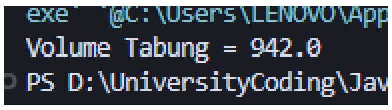


```
package week6;

2 references | Codeium: Refactor | Explain | qodo Gen: Options | Test this class
public class Tabung extends Bangun {
    protected int t;

    1 reference | Codeium: Refactor | Explain | Generate Javadoc | X | qodo Gen: Options | Test this method
    void setSuperPhi(double phi) {
        super.phi = phi;
    }

    1 reference | Codeium: Refactor | Explain | Generate Javadoc | X | qodo Gen: Options | Test this method
    void setSuperR(int r) {
        super.r = r;
    }

    1 reference | Codeium: Refactor | Explain | Generate Javadoc | X | qodo Gen: Options | Test this method
    public void setT(int t) {
        this.t = t;
    }

    1 reference | Codeium: Refactor | Explain | Generate Javadoc | X | qodo Gen: Options | Test this method
    void volume() {
        System.out.println("Volume Tabung = " + (super.phi * super.r * super.r * this.t));

    }
}
```



```
package week6;

0 references | Codeium: Refactor | Explain | qodo Gen: Options | Tes
public class Percobaan3 {
    Run | Debug | 0 references | Codeium: Refactor | Explain | Ger
    public static void main(String[] args) {
        Tabung tabung = new Tabung();
        tabung.setSuperPhi(phi:3.14);
        tabung.setSuperR(r:10);
        tabung.setT(t:3);

        tabung.volume();
    }
}
```

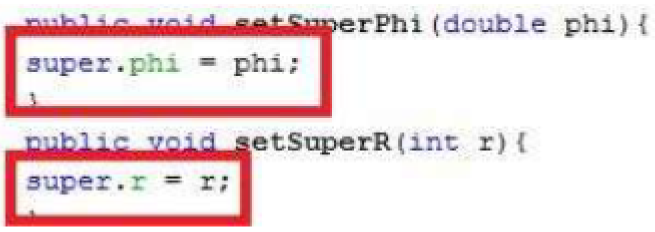
Output:



```
 exe' '@C:\Users\LENOVO\App
 Volume Tabung = 942.0
 PS D:\UniversityCoding\Jav
```

Question

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;
}
```

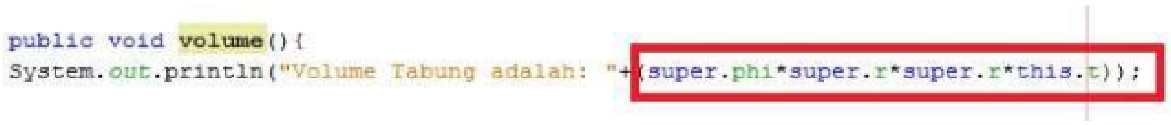
the super keyword is used to refer to the superclass of the current class. This allows access to the members (fields or methods) of the superclass.

In this context:

- super.phi = phi; assigns the value of the phi parameter to the phi field of the superclass.
- super.r = r; assigns the value of the r parameter to the r field of the superclass.

By using super, you can access and modify the superclass's fields directly from the subclass. This is particularly useful when the superclass fields are declared as protected and need to be accessed or modified in the subclass.

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));
```

In summary, in the volume() method of the Tube class:
- super.phi, super.r, and super.t access variables from the superclass.
- this.t accesses the instance variable t of the current Tube class.

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

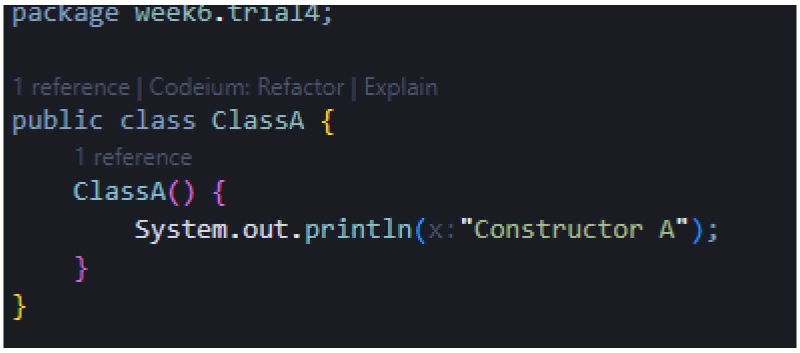
he Tabung class does not declare the attributes phi and r, but it can still access these attributes. This is because the Tabung class extends the Bangun class using the extends keyword.

When a class extends another class, it inherits all the non-private attributes and methods of the parent class. In this case, Tabung is a subclass of Bangun, so it inherits the attributes phi and r from the Bangun class. Since phi and r are declared as protected in the Bangun class, they are accessible to subclasses like Tabung.

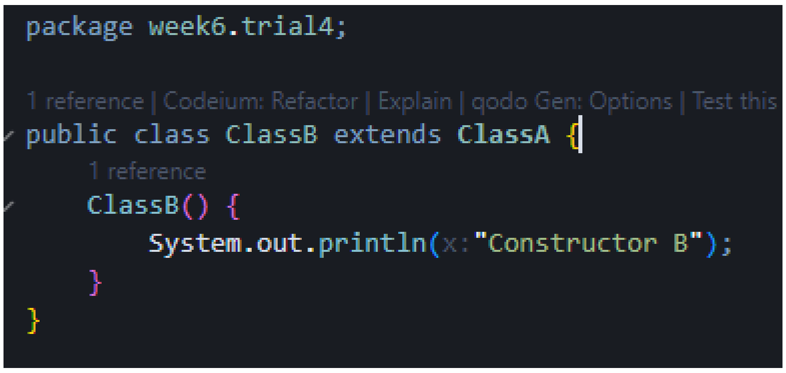
By calling super.phi and super.r in the Tabung class, it accesses the phi and r attributes of the superclass Bangun. This allows the Tabung class to utilize these attributes without explicitly declaring them in its own class definition.

Trial 4

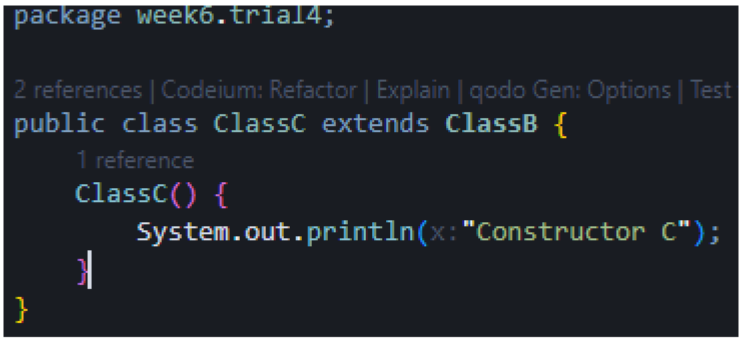
Code:



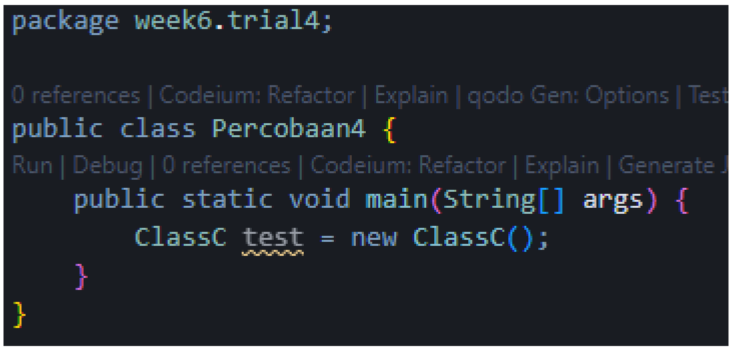
```
package week6.trial4;

1 reference | Codeium: Refactor | Explain
public class ClassA {
    1 reference
    ClassA() {
        System.out.println(x:"Constructor A");
    }
}
```



```
package week6.trial4;

1 reference | Codeium: Refactor | Explain | qodo Gen: Options | Test this
public class ClassB extends ClassA {
    1 reference
    ClassB() {
        System.out.println(x:"Constructor B");
    }
}
```

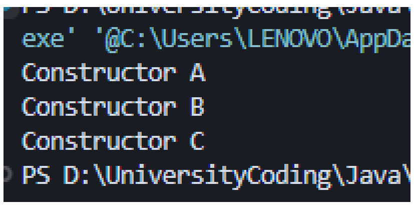


```
package week6.trial4;

2 references | Codeium: Refactor | Explain | qodo Gen: Options | Test
public class ClassC extends ClassB {
    1 reference
    ClassC() {
        System.out.println(x:"Constructor C");
    }
}
```



```
package week6.trial4;

0 references | Codeium: Refactor | Explain | qodo Gen: Options | Test
public class Percobaan4 {
    Run | Debug | 0 references | Codeium: Refactor | Explain | Generate
    public static void main(String[] args) {
        ClassC test = new ClassC();
    }
}
```

Output:



```
exe' '@C:\Users\LENOVO\AppDa
Constructor A
Constructor B
Constructor C
PS D:\UniversityCoding\Java\
```

Question

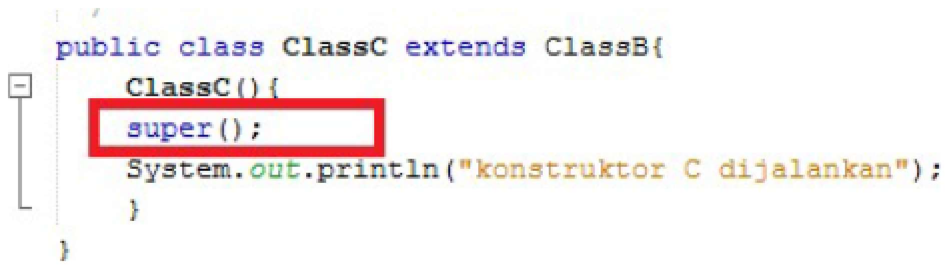
In experiment 4 state which class includes the superclass and subclass, then explain the reason!

1. **ClassA** is the superclass. It does not extend to any other class, making it the top of the hierarchy in this context. It has a constructor that prints "Constructor A".
2. **ClassB** is a subclass of ClassA. It extends ClassA, meaning it inherits the properties and methods of ClassA. ClassB has its constructor that prints "Constructor B".
3. **ClassC** is a subclass of ClassB. It extends ClassB, which means it inherits from both ClassB and ClassA, due to the inheritance chain. ClassC has its constructor that prints "Constructor C".

The reason for this hierarchy is to demonstrate inheritance in Java, where a subclass inherits the properties and behaviors of its superclass. This allows for code reuse and the creating of more complex objects by building upon simpler ones. When an instance of ClassC is created, the constructors are called in the order of inheritance: first ClassA, then ClassB, and finally ClassC. This is why you will see the output:
Constructor A
Constructor B
Constructor C

This order ensures that all necessary initializations in the superclass are completed before the subclass's constructor is executed.

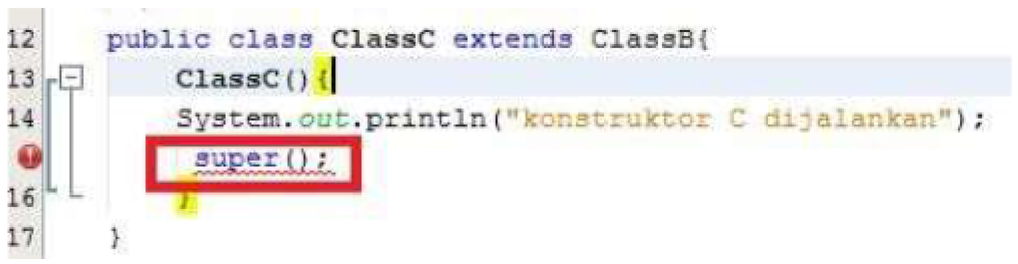
Change the contents of the ClassC default constructor as follows:



```
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!

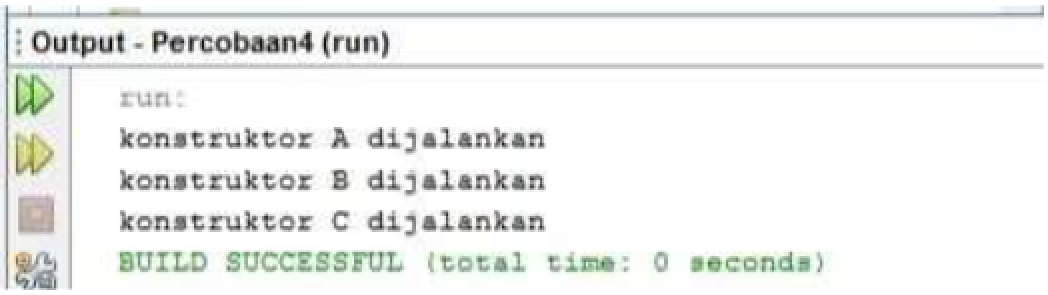
Define the contents of the ClassC default constructor as follows:



```
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

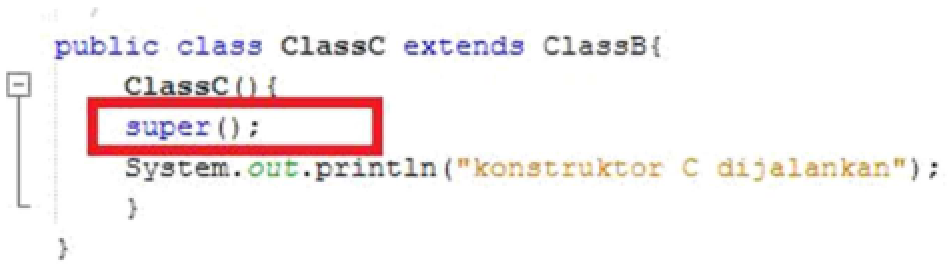


```
run:
konstruktor A dijalankan
konstruktor B dijalankan
konstruktor C dijalankan
BUILD SUCCESSFUL (total time: 0 seconds)
```

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

When the test object of type ClassC is created in the Percobaan4 class, the constructors are called in a specific order due to the inheritance hierarchy.

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



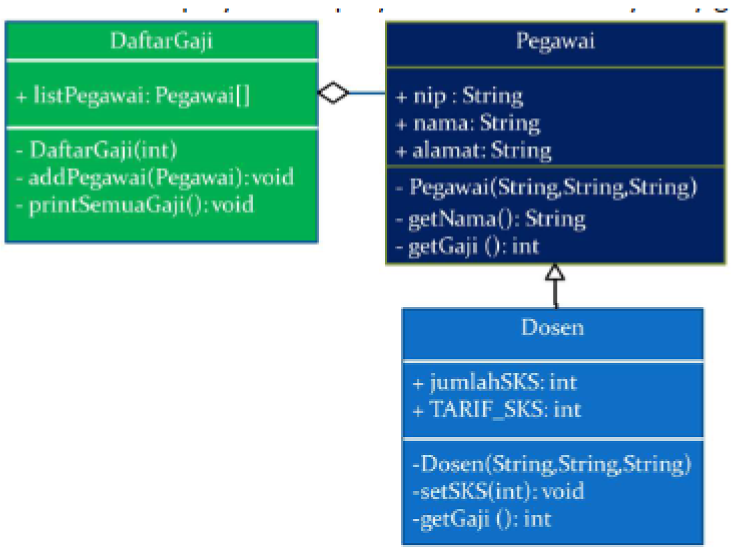
```
public class ClassC extends ClassB{
    ClassC(){
    super();
    System.out.println("konstruktor C dijalankan");
    }
}
```

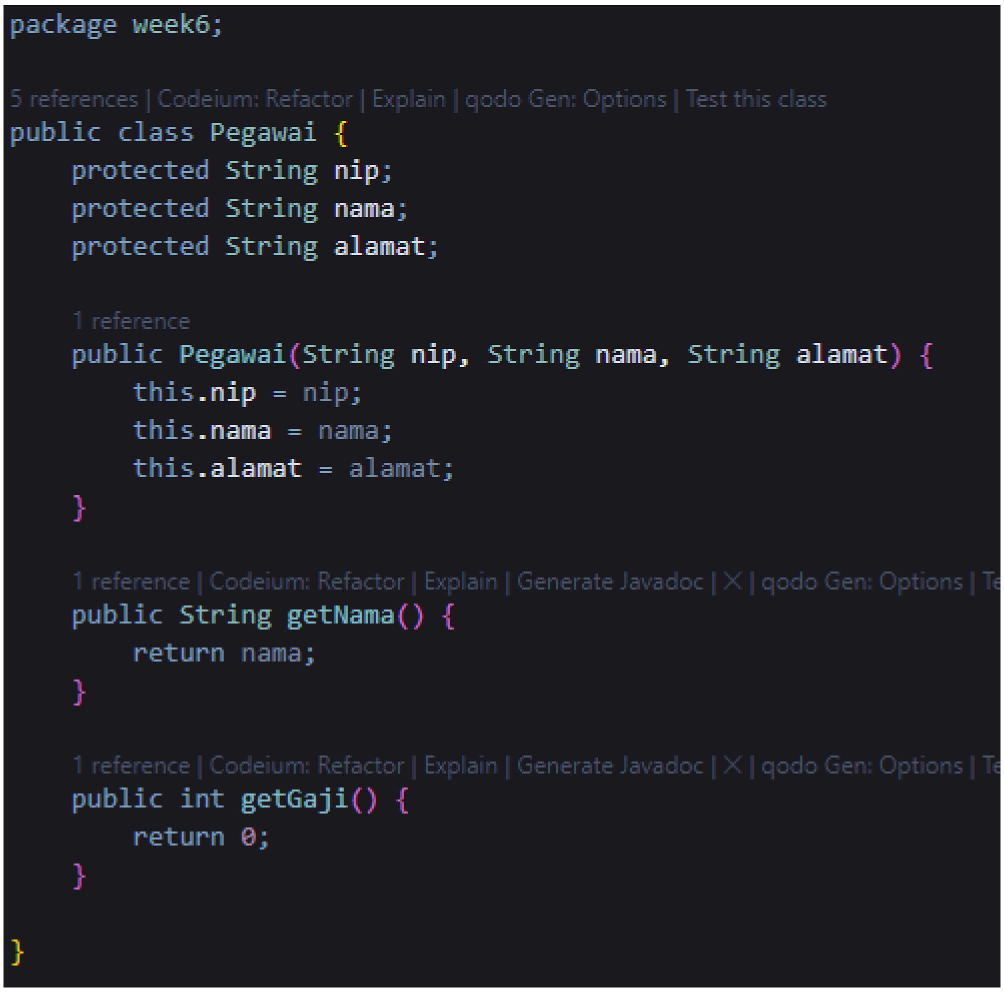
the super() function is used within the constructor of ClassC. This function call is used to invoke the constructor of its immediate superclass, which in this case is ClassB.

The super() function is essential for ensuring that the initialization defined in the superclass constructors is performed before the subclass constructor's own initialization logic.

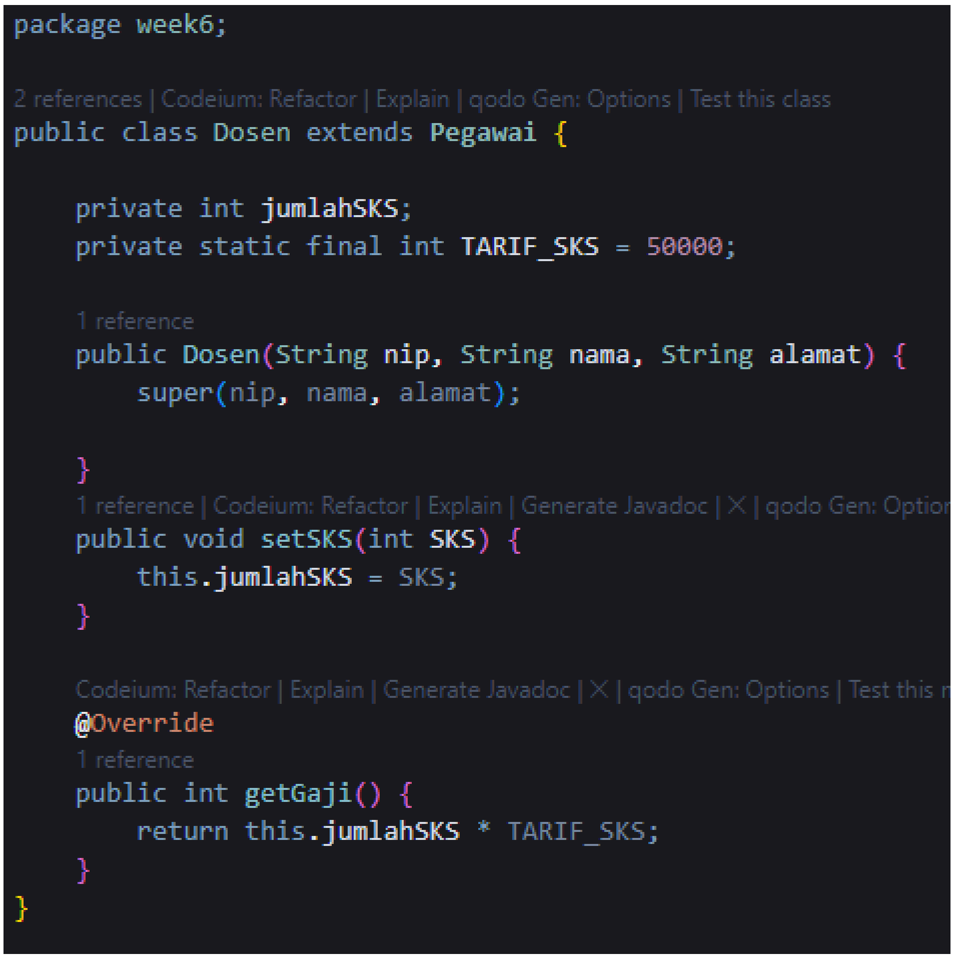
Assignment

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.

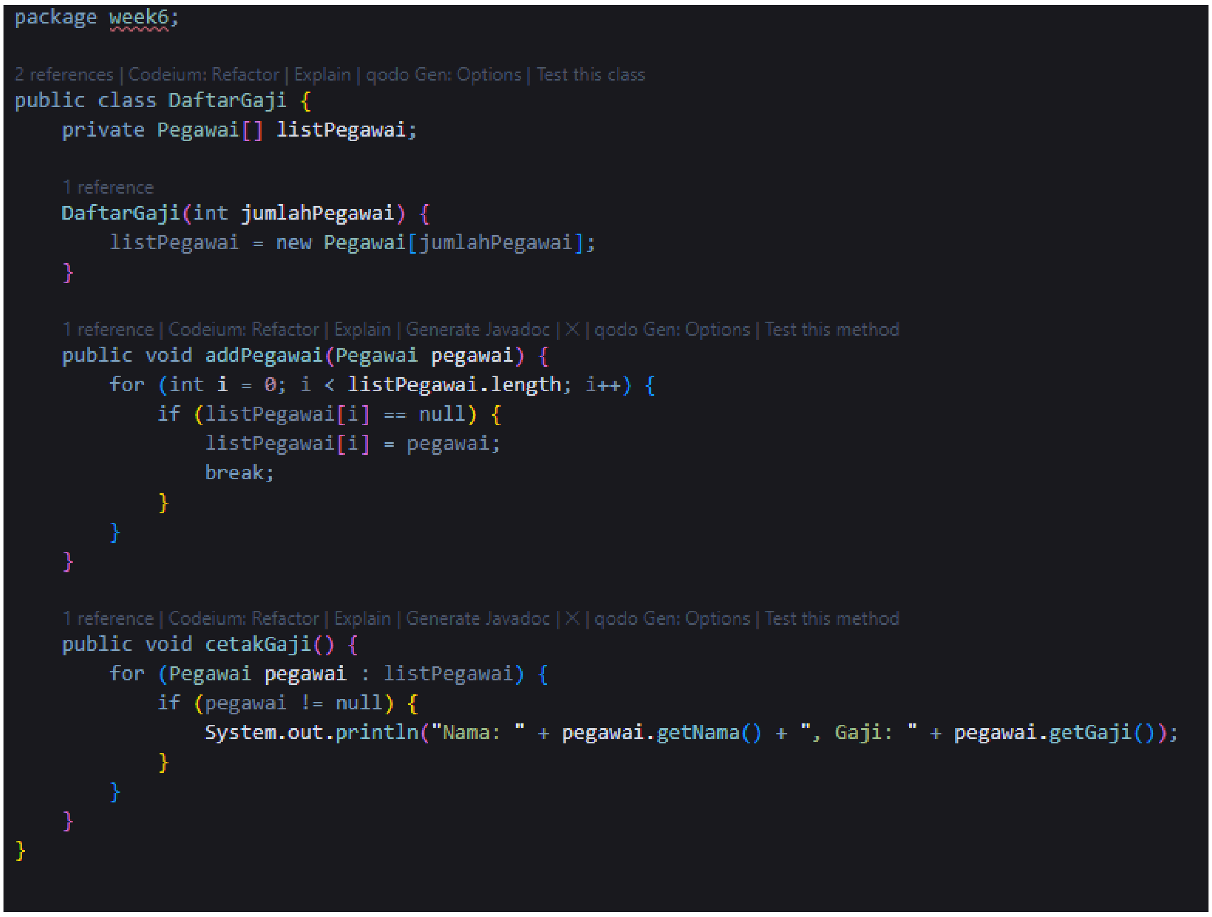




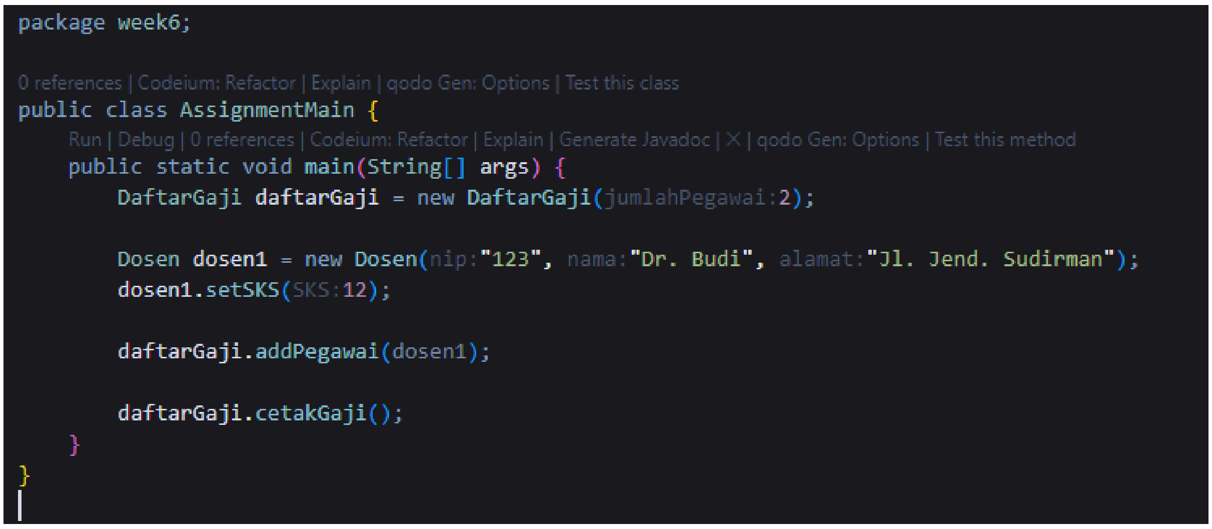
```java
package week6;

5 references | Codeium: Refactor | Explain | qodo Gen: Options | Test this class
public class Pegawai {
    protected String nip;
    protected String nama;
    protected String alamat;

    1 reference
    public Pegawai(String nip, String nama, String alamat) {
        this.nip = nip;
        this.nama = nama;
        this.alamat = alamat;
    }

    1 reference | Codeium: Refactor | Explain | Generate Javadoc | X | qodo Gen: Options | Te
    public String getNama() {
        return nama;
    }

    1 reference | Codeium: Refactor | Explain | Generate Javadoc | X | qodo Gen: Options | Te
    public int getGaji() {
        return 0;
    }

}
```



```
package week6;

2 references | Codeium: Refactor | Explain | qodo Gen: Options | Test this class
public class Dosen extends Pegawai {

    private int jumlahSKS;
    private static final int TARIF_SKS = 50000;

    1 reference
    public Dosen(String nip, String nama, String alamat) {
        super(nip, nama, alamat);

    }
    1 reference | Codeium: Refactor | Explain | Generate Javadoc | X | qodo Gen: Optior
    public void setSKS(int SKS) {
        this.jumlahSKS = SKS;
    }

    Codeium: Refactor | Explain | Generate Javadoc | X | qodo Gen: Options | Test this r
    @Override
    1 reference
    public int getGaji() {
        return this.jumlahSKS * TARIF_SKS;
    }
}
```

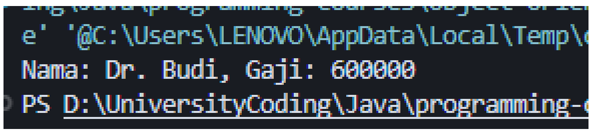


```
package week6;

2 references | Codeium: Refactor | Explain | qodo Gen: Options | Test this class
public class DaftarGaji {
    private Pegawai[] listPegawai;

    1 reference
    DaftarGaji(int jumlahPegawai) {
        listPegawai = new Pegawai[jumlahPegawai];
    }

    1 reference | Codeium: Refactor | Explain | Generate Javadoc | X | qodo Gen: Options | Test this method
    public void addPegawai(Pegawai pegawai) {
        for (int i = 0; i < listPegawai.length; i++) {
            if (listPegawai[i] == null) {
                listPegawai[i] = pegawai;
                break;
            }
        }
    }

    1 reference | Codeium: Refactor | Explain | Generate Javadoc | X | qodo Gen: Options | Test this method
    public void cetakGaji() {
        for (Pegawai pegawai : listPegawai) {
            if (pegawai != null) {
                System.out.println("Nama: " + pegawai.getNama() + ", Gaji: " + pegawai.getGaji());
            }
        }
    }
}
```



```
package week6;

0 references | Codeium: Refactor | Explain | qodo Gen: Options | Test this class
public class AssignmentMain {
    Run | Debug | 0 references | Codeium: Refactor | Explain | Generate Javadoc | X | qodo Gen: Options | Test this method
    public static void main(String[] args) {
        DaftarGaji daftarGaji = new DaftarGaji(jumlahPegawai:2);

        Dosen dosen1 = new Dosen(nip:"123", nama:"Dr. Budi", alamat:"Jl. Jend. Sudirman");
        dosen1.setSKS(SKS:12);

        daftarGaji.addPegawai(dosen1);

        daftarGaji.cetakGaji();
    }
}
```

Output:



```
e' '@C:\Users\LENOVO\AppData\Local\Temp\
Nama: Dr. Budi, Gaji: 600000
PS D:\UniversityCoding\Java\programming-
```