

JOBSHEET 6 PBO

INHERITANCE

Muhammad Tegar Hibatulloh

2341720221 / 22 / 2I

Link github: <https://github.com/Garrss/Pemrogramman-Berbasis-Objek-Jobsheet/tree/main/Week6>

3. TRIAL 1 (Extends)

Code:

```
Week6 > Trial1 > J ClassA.java > ClassA > getNilai()
1  package Week6.Trial1;
2
3  public class ClassA {
4      public int x;
5      public int y;
6
7      public void getNilai() {
8          System.out.println("nilai x:" + x);
9          System.out.println("nilai y:" + y);
10     }
11 }
12
```

```
Week6 > Trial1 > J ClassB.java > ClassB > getJumlah()
1  package Week6.Trial1;
2
3  public class ClassB {
4      public int z;
5
6      public void getNilaiZ() {
7          System.out.println("nilai Z:" + z);
8      }
9
10     public void getJumlah() {
11         System.out.println("jumlah:" + (x + y + z));
12     }
13 }
14
```

```
Week6 > Trial1 > J Percobaan1.java > 🐞 Percobaan1 > 📦 main(String[])
1  package Week6.Trial1;
2
3  public class Percobaan1 {
    Run | Debug
4      public static void main(String[] args) {
5          ClassB hitung = new ClassB();
6          hitung.x = 20;
7          hitung.y = 30;
8          hitung.z = 5;
9          hitung.getNilai();
10         hitung.getNilaiZ();
11         hitung.getJumlah();
12     }
13 }
14
```

Output:

```
▼ J ClassB.java Week6\Trial1 2
  ✖ x cannot be resolved to a variable Java(33554515) [Ln 11, Col 41]
  ✖ y cannot be resolved to a variable Java(33554515) [Ln 11, Col 45]
▼ J Percobaan1.java Week6\Trial1 3
  ✖ x cannot be resolved or is not a field Java(33554502) [Ln 6, Col 16]
  ✖ y cannot be resolved or is not a field Java(33554502) [Ln 7, Col 16]
  ✖ The method getNilai() is undefined for the type ClassB Java(67108964) [Ln 9, Col 16]
▼ J AnggotaTest.java Week3 1
  ⚠ Resource leak: 'scan' is never closed Java(536871799) [Ln 7, Col 17]
```

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!

```
Week6 > Trial1 > J ClassB.java > 🐞 ClassB
1  package Week6.Trial1;
2
3  public class ClassB extends ClassA {
4      public int z;
5
6      public void getNilaiZ() {
7          System.out.println("nilai Z:" + z);
8      }
9
10     public void getJumlah() {
11         System.out.println("jumlah:" + (x + y + z));
12     }
13 }
14
```

```
nilai x:20
nilai y:30
nilai Z:5
jumlah:55
PS D:\Pemrograman Berbasis Objek-Jobsheet>
```

2. 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.

4. TRIAL 2 (Access Control)

Code:

```
Week6 > Trial1 > J ClassA.java > ClassA > y
1  package Week6.Trial1;
2
3  public class ClassA {
4      private int x;
5      private int y;
6
7      public void setX(int x) {
8          this.x = x;
9      }
10
11     public void setY(int y) {
12         this.y = y;
13     }
14
15     public void getNilai() {
16         System.out.println("nilai x:" + x);
17         System.out.println("nilai y:" + y);
18     }
19 }
20
```

```
Week6 > Trial1 > J ClassB.java > ClassB > z
1  package Week6.Trial1;
2
3  public class ClassB extends ClassA {
4      private int z;
5
6      public void setZ(int z) {
7          this.z = z;
8      }
9
10     public void getNilaiZ() {
11         System.out.println("nilai Z:" + z);
12     }
13
14     public void getJumlah() {
15         System.out.println("jumlah:" + (x + y + z));
16     }
17 }
18
```

```
Week6 > Trial1 > J Percobaan2.java > Percobaan2 > main(String[])
1  package Week6.Trial1;
2
3  public class Percobaan2 {
4      Run | Debug
      public static void main(String[] args) {
5          ClassB hitung = new ClassB();
6          hitung.setX(x:20);
7          hitung.setY(y:30);
8          hitung.setZ(z:5);
9          hitung.getNilai();
10         hitung.getNilaiZ();
11         hitung.getJumlah();
12     }
13 }
14
```

Output:

```
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 Week6.Trial1.ClassB.getJumlah(ClassB.java:15)
    at Week6.Trial1.Percobaan2.main(Percobaan2.java:11)
PS D:\Pemrograman Berbasis Objek-Jobsheet>
```

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!

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

nilai x:20
nilai y:30
nilai Z:5
jumlah:55
PS D:\Pemrograman Berbasis Objek-Jobsheet>
```

2. 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 (Super)

Code:

```
Week6 > Trial3 > J Bangun.java > Bangun > r
1  package Week6.Trial3;
2
3  public class Bangun {
4      protected double phi;
5      protected int r;
6  }
7
```

```
Week6 > Trial3 > J Tabung.java > Tabung > volume()
1  package Week6.Trial3;
2
3  public class Tabung extends Bangun {
4      protected int t;
5
6      public void setSuperPhi(double phi) {
7          super.phi = phi;
8      }
9
10     public void setSuperR(int r) {
11         super.r = r;
12     }
13
14     public void setT(int t) {
15         this.t = t;
16     }
17
18     public void volume() {
19         System.out.println("Volume Tabung adalah: " + (super.phi * super.r * super.r * this.t));
20     }
21 }
22
```

```
Week6 > Trial3 > J Percobaan3.java > Percobaan3 > main(String[])
1  package Week6.Trial3;
2
3  public class Percobaan3 {
4      Run | Debug
5      public static void main(String[] args) {
6          Tabung tabung = new Tabung();
7          tabung.setSuperPhi(phi:3.14);
8          tabung.setSuperR(r:10);
9          tabung.setT(t:3);
10         tabung.volume();
11     }
12 }
```

Output:

```
Volume Tabung adalah: 942.0
PS D:\Pemrograman Berbasis Objek-Jobsheet>
```

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;  
}
```

The super keyword in Java refers to the parent class (or superclass) from which the current class is derived. In this case, the Tabung class is a subclass of Bangun, and the super keyword is used to access variables in the parent class.

- super.phi = phi: This assigns the value of the parameter phi to the phi attribute of the superclass Bangun.
- super.r = r: Similarly, this assigns the value of the parameter r to the r attribute of the superclass Bangun.

Without super, it would be unclear whether the code is referring to the local variables in the Tabung class (if they existed) or the inherited attributes from Bangun. The super keyword ensures that the superclass attributes phi and r are explicitly accessed.

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));  
}
```

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

The 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.

6. TRIAL 4 (super constructor)

Code:

```
Week6 > Trial4 > J ClassA.java > ClassA > ClassA()
1  package Week6.Trial4;
2
3  public class ClassA {
4      public ClassA() {
5          System.out.println(x:"konstruktor A dijalankan");
6      }
7  }
8
```

```
Week6 > Trial4 > J ClassB.java > ClassB > ClassB()
1  package Week6.Trial4;
2
3  public class ClassB extends ClassA {
4      public ClassB() {
5          System.out.println(x:"konstruktor B dijalankan");
6      }
7  }
8
```

```
Week6 > Trial4 > J ClassC.java > ClassC > ClassC()
1  package Week6.Trial4;
2
3  public class ClassC extends ClassB {
4      public ClassC() {
5          System.out.println(x:"konstruktor C dijalankan");
6      }
7  }
8
```

Output:

```
konstruktor A dijalankan
konstruktor B dijalankan
konstruktor C dijalankan
PS D:\Pemrograman Berbasis Objek-Jobsheet>
```

B. QUESTIONS

1. In experiment 4 state which class includes the superclass and subclass, then explain the reason!
 - 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 "konstruktor A dijalankan".
 - ClassB is a subclass of ClassA. It extends ClassA, meaning it inherits the properties and methods of ClassA. ClassB has its constructor that prints "konstruktor B dijalankan".
 - 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 "konstruktor C dijalankan".

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:

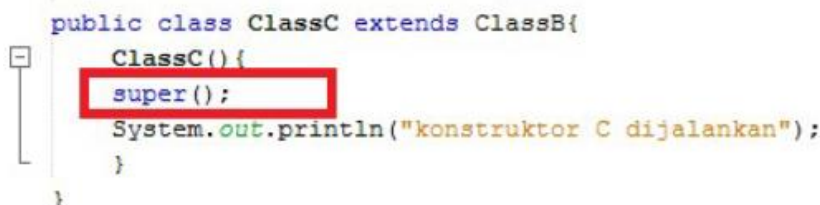
Konstruktor A dijalankan

Konstruktor B dijalankan

Konstruktor C dijalankan

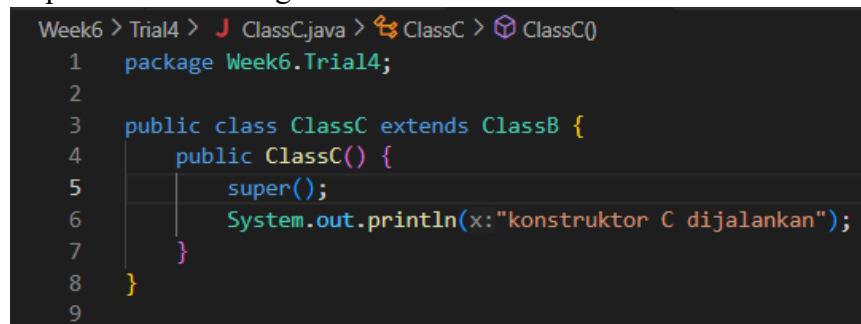
This order ensures that all necessary initializations in the superclass are completed before the subclass's constructor is executed.

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



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

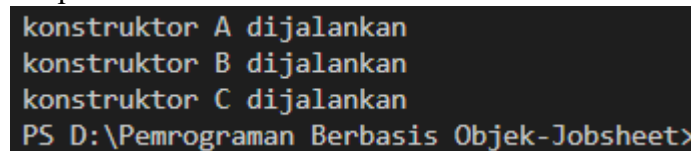
Add the word `super()` in the First row in the default constructor. Try running the Experiment 4 class again and it looks like there is no difference from the output!



```
Week6 > Trial4 > J ClassC.java > ClassC > ClassC()  
1 package Week6.Trial4;  
2  
3 public class ClassC extends ClassB {  
4     public ClassC() {  
5         super();  
6         System.out.println(x:"konstruktor C dijalankan");  
7     }  
8 }  
9
```

Even after adding `super()` in the first line of the constructor, the output remains the same because Java automatically inserts a call to the parent class constructor (`super()`) in the first line if it is not explicitly written. Hence, adding `super()` manually to the first line doesn't change the behavior.

Output:



```
konstruktor A dijalankan  
konstruktor B dijalankan  
konstruktor C dijalankan  
PS D:\Pemrograman Berbasis Objek-Jobsheet>
```

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


```

12 public class ClassC extends ClassB{
13     ClassC() {
14         System.out.println("konstruktor C dijalankan");
15         super();
16     }
17 }

```

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!

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.

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");
    }
}

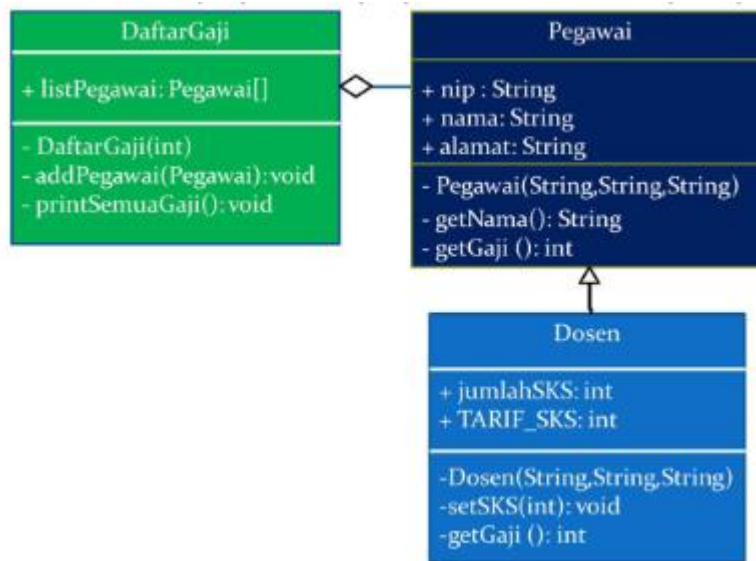
```

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

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.



ANSWER

Pegawai Class:

```

Week6 > Assignment > J Pegawai.java > 🐞 Pegawai
1  package Week6.Assignment;
2
3  public class Pegawai {
4      protected String nip;
5      protected String nama;
6      protected String alamat;
7
8      public Pegawai(String nip, String nama, String alamat) {
9          this.nip = nip;
10         this.nama = nama;
11         this.alamat = alamat;
12     }
13
14     public String getNama() {
15         return nama;
16     }
17
18     public int getGaji() {
19         return 0; // Base salary for Pegawai (default is 0)
20     }
21 }
  
```

Dosen Class:

```

Week6 > Assignment > J Dosen.java > Dosen
1  package Week6.Assignment;
2
3  public class Dosen extends Pegawai{
4      private int jumlahSKS;
5      private final int TARIF_SKS = 120000; // Assuming a rate for each SKS
6
7      public Dosen(String nip, String nama, String alamat) {
8          super(nip, nama, alamat);
9      }
10
11     public void setSKS(int jumlahSKS) {
12         this.jumlahSKS = jumlahSKS;
13     }
14
15     @Override
16     public int getGaji() {
17         return jumlahSKS * TARIF_SKS;
18     }
19 }

```

Daftar Gaji Class:

```

Week6 > Assignment > J DaftarGaji.java > DaftarGaji > printSemuaGaji()
1  package Week6.Assignment;
2
3  public class DaftarGaji {
4      private Pegawai[] listPegawai;
5      private int index;
6
7      public DaftarGaji(int size) {
8          listPegawai = new Pegawai[size];
9          index = 0;
10     }
11
12     public void addPegawai(Pegawai pegawai) {
13         if (index < listPegawai.length) {
14             listPegawai[index] = pegawai;
15             index++;
16         } else {
17             System.out.println(x:"List is full, cannot add more Pegawai");
18         }
19     }
20
21     public void printSemuaGaji() {
22         for (int i = 0; i < index; i++) {
23             Pegawai pegawai = listPegawai[i];
24             System.out.println("Nama: " + pegawai.getNama() + ", Gaji: " + pegawai.getGaji());
25         }
26     }
27 }
28

```

Main:

```

Week6 > Assignment > J Main.java > Main > main(String[])
1  package Week6.Assignment;
2
3  public class Main {
4      Run | Debug
5      public static void main(String[] args) {
6          // Create DaftarGaji instance
7          DaftarGaji daftarGaji = new DaftarGaji(size:2);
8
9          // Create a Dosen object and set its SKS
10         Dosen dosen1 = new Dosen(nip:"001", nama:"Dr. Ahmad Yanto", alamat:"Jl. Kendalsari No. 1");
11         dosen1.setSKS(jumlahSKS:10); // Setting the SKS
12
13         // Add Dosen to DaftarGaji
14         daftarGaji.addPegawai(dosen1);
15
16         // Create another Dosen
17         Dosen dosen2 = new Dosen(nip:"002", nama:"Dr. Isabella Arie", alamat:"Jl. Blimbing No. 2");
18         dosen2.setSKS(jumlahSKS:8); // Setting the SKS
19
20         // Add second Dosen to DaftarGaji
21         daftarGaji.addPegawai(dosen2);
22
23         // Print all employees' names and their salaries
24         daftarGaji.printSemuaGaji();
25     }
26 }

```

Explanation:

1. **Pegawai Class:** This is the base class representing employees with basic attributes like nip, nama, and alamat. It also contains methods getName() and getGaji(). By default, getGaji() returns 0.
2. **Dosen Class:** This class extends Pegawai and adds specific attributes for lecturers like jumlahSKS and a constant TARIF_SKS. The getGaji() method is overridden to calculate the salary based on the number of SKS and the rate per SKS.
3. **DaftarGaji Class:** This class maintains a list of Pegawai objects and provides functionality to add employees (addPegawai()) and print the name and salary of all employees (printSemuaGaji()).
4. **Main Class:** In the main() method, we create a list of Dosen objects, set their SKS, and add them to DaftarGaji. Finally, we print the names and salaries of all employees in the list.

OUTPUT:

```

Nama: Dr. Ahmad Yanto, Gaji: 1200000
Nama: Dr. Isabella Arie, Gaji: 960000
PS D:\Pemrograman Berbasis Objek-Jobsheet>

```