

JOBSHEET 6 PBO

Name : Muhammad Khoirul Anwarudin

NIM : 244107023007

Class : TI 2I

Trial 1

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:

ClassB inherit from ClassA. This will allow ClassB to access x and y from ClassA

```
1 public class ClassB extends ClassA {
2     public int z;
3
4     public void getNilaiZ() {
5         System.out.println("Nilai z : " + z);
6     }
7
8     public void getJumlah() {
9         System.out.println("Jumlah : " + (x + y + z));
10    }
11 }
```

Output:

```
Nilai x : 20
Nilai y : 30
Nilai z : 5
Jumlah : 55
PS D:\kuliah_2\OOP\jobsheet6>
```

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

Answer:

The original code causes an error because ClassB does not inherit from ClassA, meaning it does not have access to the x and y variables or the getNilai() method that belong to ClassA. Since ClassB needs to access these members (x and y) to perform operations, inheritance (extends ClassA) is required to make ClassB inherit all the properties and methods of ClassA.

Trial 2

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:

```
1 public class ClassB extends ClassA {
2     private int z;
3
4     public void setZ(int z) {
5         this.z = z;
6     }
7
8     public void getNilaiZ() {
9         System.out.println("Nilai z : " + z);
10    }
11
12    public void getJumlah() {
13        System.out.println("Jumlah : " + (getX() + getY() + z));
14    }
15 }
```

```
1 public class ClassA {
2     private int x;
3     private int y;
4
5     public void setX(int x) {
6         this.x = x;
7     }
8
9     public void setY(int y) {
10        this.y = y;
11    }
12
13    public int getX() {
14        return x;
15    }
16
17    public int getY() {
18        return y;
19    }
20
21    public void getNilai() {
22        System.out.println("Nilai x : " + x);
23        System.out.println("Nilai y : " + y);
24    }
25 }
```

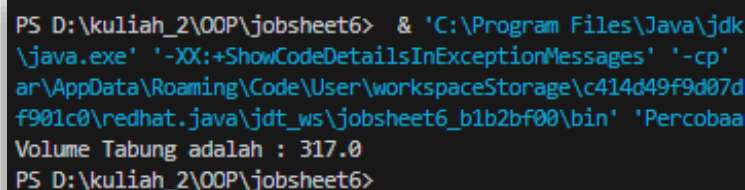
2. Explain what caused the program in experiment 1 when it ran an error!

Answer:

The error occurs because in the original code, x and y in ClassA are marked as private, meaning they can only be accessed within ClassA itself. However, in ClassB, the getJumlah() method tries to directly access x and y, which results in a compilation error since private fields are not accessible outside the class they are declared in.

Trial 3

- Output



```
PS D:\kuliah_2\OOP\jobsheet6> & 'C:\Program Files\Java\jdk\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'ar\AppData\Roaming\Code\User\workspaceStorage\c414d49f9d07df901c0\redhat.java\jdt_ws\jobsheet6_b1b2bf00\bin' 'Percobaa'
Volume Tabung adalah : 317.0
PS D:\kuliah_2\OOP\jobsheet6>
```

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

Answer:

The super keyword refers to the parent class Bangun. It is used in Tabung to access and modify the inherited attributes phi and r from Bangun.

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

Answer:

super.phi and super.r refer to the phi and r attributes from the parent class Bangun. This.t refers to the t attribute from the current instance of the Tabung class. The formula uses super.phi and super.r from Bangun, and this.t from Tabung to calculate the tube's volume.

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

Answer:

Tabung doesn't declare phi and r because it inherits them from Bangun. Since they are protected, Tabung can access them directly.

Trial 4

- Output

```
Konstruktor A dijalankan
Konstruktor B dijalankan
Konstruktor C dijalankan
PS D:\kuliah_2\OOP\jobsheet6>
```

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

Answer:

The Superclass are: ClassA and ClassB

The subclass are: ClassB and ClassC

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

```
1 public class ClasssC extends ClasssB{
2     ClasssC(){
3         super();
4         System.out.println("Konstruktor C dijalankan");
5     }
6 }
```

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!

```
Konstruktor A dijalankan
Konstruktor B dijalankan
Konstruktor C dijalankan
PS D:\kuliah_2\OOP\jobsheet6>
```

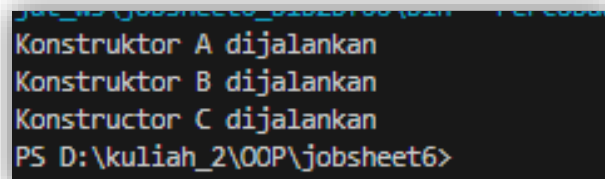
Note: no change

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

```
1 public class ClasssC extends ClasssB{
2     ClasssC(){
3         System.out.println("Konstruktor C dijalankan");
4         super();
5     }
6 }
7
```

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



```
Konstruktor A dijalankan
Konstruktor B dijalankan
Konstruktor C dijalankan
PS D:\kuliah_2\OOP\jobsheet6>
```

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

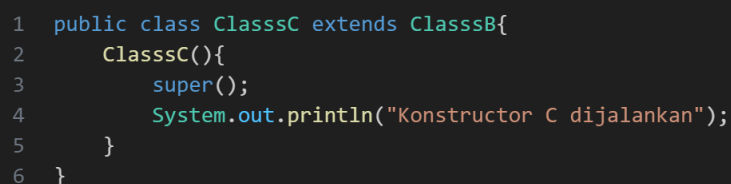
Answer:

The super() syntax can only be used at the beginning of a line in a constructor that serves to access the constructor of the superclass Class itself.

The sequence of programme processing to produce such output is:

1. When called in main (initialises the ClassC object), the program sees that it has a superclass ClassB (extends ClassB)
2. Then looking again at ClassB, it turns out that ClassB has a superclass ClassA (ClassB extends ClassA).
3. Then seeing that ClassA does not have a superclass, the programme runs ClassA's constructor,
4. After completion. Return to ClassB to run ClassB's constructor.
5. After that, return to classC to run the constructor of ClassC.

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



```
1 public class ClassssC extends ClassssB{
2     ClassssC(){
3         super();
4         System.out.println("Konstruktor C dijalankan");
5     }
6 }
```

Answer:

The super() syntax can only be used at the beginning of a line in a constructor that serves to access the constructor of the superclass Class itself. So in that class super() serves to call the constructor in ClassB.

Assignment:

Class Pegawai.java

```
1 public class Pegawai {
2     private String nip;
3     private String nama;
4     private String alamat;
5
6     Pegawai(String nip, String nama, String alamat) {
7         this.nip = nip;
8         this.nama = nama;
9         this.alamat = alamat;
10    }
11
12    public String getNama() {
13        return nama;
14    }
15
16    public String getNip() {
17        return nip;
18    }
19
20    public String getAlamat() {
21        return alamat;
22    }
23
24    public int getGaji() {
25        return 100000;
26    }
27 }
```

Class Dosen.java

```
1 public class Dosen extends Pegawai{
2     private int jumlahSKS;
3     private int tarifSKS = 100000;
4
5     Dosen(String nip, String nama, String alamat) {
6         super(nip, nama, alamat);
7     }
8
9     public void setSKS(int sks) {
10         this.jumlahSKS = sks;
11     }
12
13     public int getGaji() {
14         return tarifSKS * jumlahSKS;
15     }
16 }
```

Class DaftarGaji.java

```

1 public class DaftarGaji {
2     private Pegawai[] listPegawai;
3
4     DaftarGaji(int jmlPegawai) {
5         listPegawai = new Pegawai[jmlPegawai];
6     }
7
8     public void addPegawai(Pegawai pegawai) {
9         for (int i = 0; i < listPegawai.length; i++) {
10             if (listPegawai[i] == null) {
11                 this.listPegawai[i] = pegawai;
12                 break;
13             }
14         }
15     }
16
17     public void printSemuaGaji() {
18         int i = 0;
19         while (listPegawai[i] != null && i < listPegawai.length) {
20             System.out.println("NIP\t: " + listPegawai[i].getNip());
21             System.out.println("Nama\t: " + listPegawai[i].getNama());
22             System.out.println("Alamat\t: " + listPegawai[i].getAlamat());
23             System.out.println("Gaji\t: " + listPegawai[i].getGaji());
24             System.out.println();
25             i++;
26         }
27     }
28 }

```

Class App.java

```

1 public class App {
2     public static void main(String[] args) throws Exception {
3         DaftarGaji data = new DaftarGaji(4);
4         Dosen halim = new Dosen("001", "Halim", "Malang");
5         halim.setSKS(5);
6
7         Pegawai halim1 = new Pegawai("001", "Halim", "Malang");
8         data.addPegawai(halim1);
9
10        Dosen teguh = new Dosen("002", "Teguh", "Surabaya");
11        teguh.setSKS(3);
12        data.addPegawai(teguh);
13
14        Dosen saputro = new Dosen("003", "Saputro", "Bandung");
15        saputro.setSKS(8);
16        data.addPegawai(saputro);
17
18        data.printSemuaGaji();
19    }
20 }

```

Output:

```
NIP      : 001
Nama      : Halim
Alamat    : Malang
Gaji      : 100000
```

```
NIP      : 002
Nama      : Teguh
Alamat    : Surabaya
Gaji      : 300000
```

```
NIP      : 003
Nama      : Saputro
Alamat    : Bandung
Gaji      : 800000
```