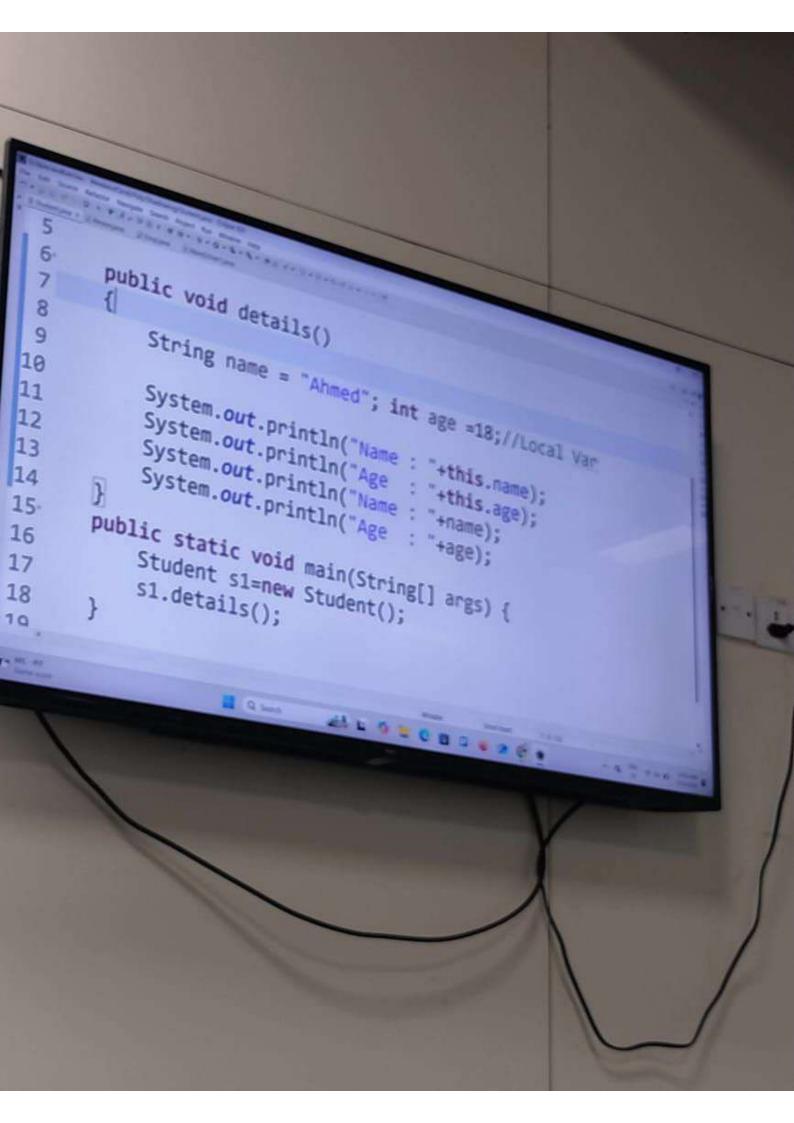```java
1 package Shadowing;
2
3 public class Student {
4   String name="Krishna";int age =21;//Global Var
5
6   public void details()
7   {
8
9   String name = "Ahmed"; int age =18;//Local Var
10
11   System.out.println("Name : "+this.name);
12   System.out.println("Age   : "+this.age);
13   System.out.println("Name : "+name);
14   System.out.println("Age   : "+age);
15   }
    public static void main(String[] angc) {
```

```java
    public void details()
    {
        String name = "Ahmed"; int age =18;//Local Var
        System.out.println("Name  :  "+this.name);
        System.out.println("Age   :  "+this.age);
        System.out.println("Name  :  "+name);
        System.out.println("Age   :  "+age);
    }
    public static void main(String[] args) {
        Student s1=new Student();
        s1.details();
    }
}
```

```java
public class Emp extends Person {
    int eid=122;
    String desg="Developer";

    public void data()
    {
        String name="Abi",desg="Tester";
        int age=21,eid=125;
        System.out.println("Name : "+super.name);
        System.out.println("Age   : "+super.age);
        System.out.println("EmpId: "+this.eid);
        System.out.println("Emp Desg : "+this.desg);
    }
}
```

```java
1 package Overloading;
2
3 class A {
4     static void m1(int i)
5     {
6         System.out.println("int i = "+i);
7     }
8     static void m1(double d)
9     {
10        System.out.println("double d = "+d);
11    }
12 }
13
```

```java
package Shadowing;

public class Person {
    String name="Jana";
    int age=23;
}
```

```java
1 package Shadowing;
2
3 public class MainDriver1 {
4
5     public static void main(String[] args) {
6         Emp e1=new Emp();
7         e1.data();
8     }
9 }
10
```

```java
package Overloading;
class B extends A {
    void m2(String s)
    {
        System.out.println("String s = "+s);
    }
    void m2(long l)
    {
        System.out.println("long l = "+l);

    public static void main(String[] args) {
        B b1=new B();
        b1.m2(9008675651);
```

```java
     }

void m2(long l)
{
     System.out.println("long l = "+l);
}

public static void main(String[] args) {
     B b1=new B();
     b1.m2(90867565L);
     b1.m2("Good");
     B.m1(52);//class by static way ClassName
     B.m1(90.09);
}
}
```

```
7
8   }
9   void m2(long l)
10  {
11      System.out.println("long l = "+l);
12  }
13
14  public static void main(String[] args) {
15      B b1=new B();
16      //call by Object ref of the ClassName
17      b1.m2(90867565L);
18      b1.m2("Good");
19      //call by static way ClassName
20      B.m1(52);
        B.m1(90.09);
```

```java
7     }
8
9     void m2(long l)
10    {
11        System.out.println("long l = "+l);
12    }
13
14    public static void main(String[] args) {
15        B b1=new B();
16        //call by Reference o|
17        b1.m2(90867565L);
18        b1.m2("Good");
19        //call by static way ClassName
20        B.m1(52);
       B.m1(90.09);
```

Run-time / Dynamic Polymorphism

Based on the type of instance calling the behavior, during the execution process, the behavior can exhibit different task. This is called as Run-Time polymorphism.

i. **Overriding**

Here, a parent class and a child class can have methods, with same name, same return type, same arguments and broader access modifier (public in child).

Rule to achieve method overriding :

- Is-A relationship is mandatory.
- It is applicable only for nonstatic methods.
- The signature of the subclass method and superclass method should be the same.

VARIABLE SHADOWING :

If the superclass and subclass have variables with the same name then it is known as variable shadowing.

Which variable is used, depending on what?

In variable shadowing binding is done at compile-time, hence it is a compile-time polymorphism. The Variable used depends on the reference type and does not depend on the type of object created.

NOTE :

- It is applicable for both static and non-static variables.
- It is a compile-time polymorphism.
- Variable usage depends on the type of reference and does not depend on the type of object created.

Bank
+ UserName
+ Mob No

+ Saving() + CurrentAc()

Extends

extends

Acc1
+ Accno
+ bal
⊕ Saving () ⊕ Current Acc()
+ Create() + withdraw()
+ deposit() + LoanAmt()

Acc2
+ Accno
+ bal
+ Saving() + CurrentAcc()
+ Create() + withdrawl()
+ deposit() + LoanAmt()

→ Inheritance

→ Typecasting.

→ Polymor

⤷ override