SNS COLLEGE OF ENGINEERING

Kurumbapalayam(Po), Coimbatore – 641 107
Accredited by NAAC-UGC with ‘A’ Grade
Approved by AICTE, Recognized by UGC & Affiliated to Anna University, Chennai

Department of Information Technology

19CS204 OBJECT ORIENTED PROGRAMMING

I YEAR /II SEMESTER

Topic – Method Overriding

Method Overriding

- In a class hierarchy, when a method in a subclass has the same name and type signature as a method in its superclass, then the method in the subclass is said to override the method in the superclass.

- When an overridden method is called from within its subclass, it will always refer to the version of that method defined by the subclass.

- Method overriding occurs only when the names and the type signatures of the two methods are identical. If they are not, then the two methods are simply overloaded

Method Overriding

Rules for Overriding

- Both the superclass and the subclass must have the same method name, the same return type and the same parameter list.
- We cannot override the method declared as final, static and private
- Access Modifier of the overriding method (method of subclass) cannot be more restrictive than the overridden method of parent class.
- For e.g. if the Access Modifier of parent class method is public then the overriding method (child class method) cannot have private, protected and default Access modifier.
- All the abstract methods in the parent class should be overridden in the c
- Method overriding performs only if two classes have is-a relationship. It mean class must have inheritance. In other words, It is performed between two classes using inheritance relation

Method Overriding

Example

```
class Bank{
int getRateOfInterest(){return 0;}
}
//Creating child classes.
class SBI extends Bank{
int getRateOfInterest(){return 8;}
}

class ICICI extends Bank{
int getRateOfInterest(){return 7;}
}
class AXIS extends Bank{
int getRateOfInterest(){return 9;}
}
```

```
//Test class to create objects and call the methods
public class Test2{
public static void main(String args[]){
SBI s=new SBI();
ICICI i=new ICICI();
AXIS a=new AXIS();
System.out.println("SBI Rate of Interest: "+s.getRateOfInterest());
System.out.println("ICICI Rate of Interest: "+i.getRateOfInterest());
System.out.println("AXIS Rate of Interest: "+a.getRateOfInterest());
}
}
```

Method Overriding

Difference between Method Overloading and Overriding

Method Overloading	Method Overriding
Method overloading is used to increase the readability of the program	Method overriding is used to provide the specific implementation of the method that is already provided by its super class
Method overloading is performed within class.	Method overriding occurs in two classes that have IS-A (inheritance) relationship.
In case of method overloading, parameter must be different	In case of method overriding, parameter must be same
Method overloading is the example of compile time polymorphism	Method overriding is the example of run time polymorphism.
It should have methods with the same name but a different signature	It should have methods with same name and signature.

Super

What if I want to access Superclass methods or variables ?

- Whenever a subclass needs to refer to its immediate superclass, it can do so by use of the keyword super.

- The super keyword in Java is a reference to the object of the parent/superclass. Using it, you can refer/call a field, a method or, a constructor of the immediate superclass.

Super

Uses of super keyword

- To call methods of the superclass that is overridden in the subclass.

- To access attributes (fields) of the superclass if both superclass and subclass have attributes with the same name.

- To explicitly call superclass no-arg (default) or parameterized constructor from the subclass constructor.

Super

Uses of super keyword

- To call methods of the superclass that is overridden in the subclass.

```
class Parentclass
{
   //Overridden method
   void display(){
           System.out.println("Parent class method");
   }
}
class Subclass extends Parentclass
{
   //Overriding method
   void display(){
           System.out.println("Child class method");
   }
   void printMsg(){
           //This would call Overriding method
           display();
           //This would call Overridden method
           super.display();
   }
}
```

```
public class example
{
public static void main(String args[]){
          Subclass obj= new Subclass();
          obj.printMsg();
 }

}
```

Super

Uses of super keyword

- To access attributes (fields) of the superclass if both superclass and subclass have attributes with the same name

```
class Superclass
{
   int num = 100;
}
class Subclass extends Superclass
{
   int num = 110;
   void printNumber(){
               /* Note that instead of writing num we are
                * writing super.num in the print statement
                * this refers to the num variable of Superclass
                */
               System.out.println(super.num);
   }
}
   public class example
   {
   public static void main(String args[]){
               Subclass obj= new Subclass();
               obj.printNumber();
   }
}
```

Super

Uses of super keyword

- To explicitly call superclass no-arg (default) or parameterized constructor from the subclass constructor.

```
class Person{
  int id;
 String name;
Person(int id,String name){
 this.id=id;
this.name=name;
 }   }
 class Emp extends Person{
float salary;
Emp(int id,String name,float salary){
super(id,name);//reusing parent constructor
 this.salary=salary;    }
 void display(){
System.out.println(id+" "+name+" "+salary);
}   }
class TestSuper11{
public static void main(String[] args){
 Emp e1=new Emp(1,"ankit",411000f);   e1.display();
  }}
```

THANK YOU