BCA – 401: Java Programming

Rahul Kumar Singh

In today's Class we have discussed on Method Overriding in Java.

Method Overriding in Java:-

If subclass (child class) has the same method as declared in the parent class, it is known as method overriding in Java.

In other words, If a subclass provides the specific implementation of the method that has been declared by one of its parent class, it is known as method overriding.

Usage of Java Method Overriding:-

- Method overriding is used to provide the specific implementation of a method which is already provided by its superclass.
- Method overriding is used for runtime polymorphism.

Rules for Java Method Overriding:-

- The method must have the same name as in the parent class.
- The method must have the same parameter as in the parent class.
- The argument list should be exactly the same as that of the overridden method.
- There must be an IS-A relationship (inheritance). If a

method cannot be inherited, then it cannot be overridden.

- The return type should be the same or a subtype of the return type declared in the original overridden method in the superclass.
- The access level cannot be more restrictive than the overridden method's access level. For example: If the superclass method is declared public then the overridding method in the sub class cannot be either private or protected.
- Instance methods can be overridden only if they are inherited by the subclass.
- A method declared final cannot be overridden.
- A method declared static cannot be overridden but can be re-declared.
- A subclass within the same package as the instance's superclass can override any superclass method that is not declared private or final.
- A subclass in a different package can only override the non-final methods declared public or protected.
- An overriding method can throw any uncheck exceptions, regardless of whether the overridden method throws exceptions or not. However, the overriding method should not throw checked exceptions that are new or broader than the ones

declared by the overridden method. The overriding method can throw narrower or fewer exceptions than the overridden method.

- Constructors cannot be overridden.

Example of method overriding:-

In this example, we have defined the run method in the subclass as defined in the parent class but it has some specific implementation. The name and parameter of the method are the same, and there is IS-A relationship between the classes, so there is method overriding.

//Java Program to illustrate the use of Java Method Overriding.

```
class Vehicle
{
  void run()
  {
      System.out.println("Vehicle is running");
  }
}
class Bike2 extends Vehicle
{
```

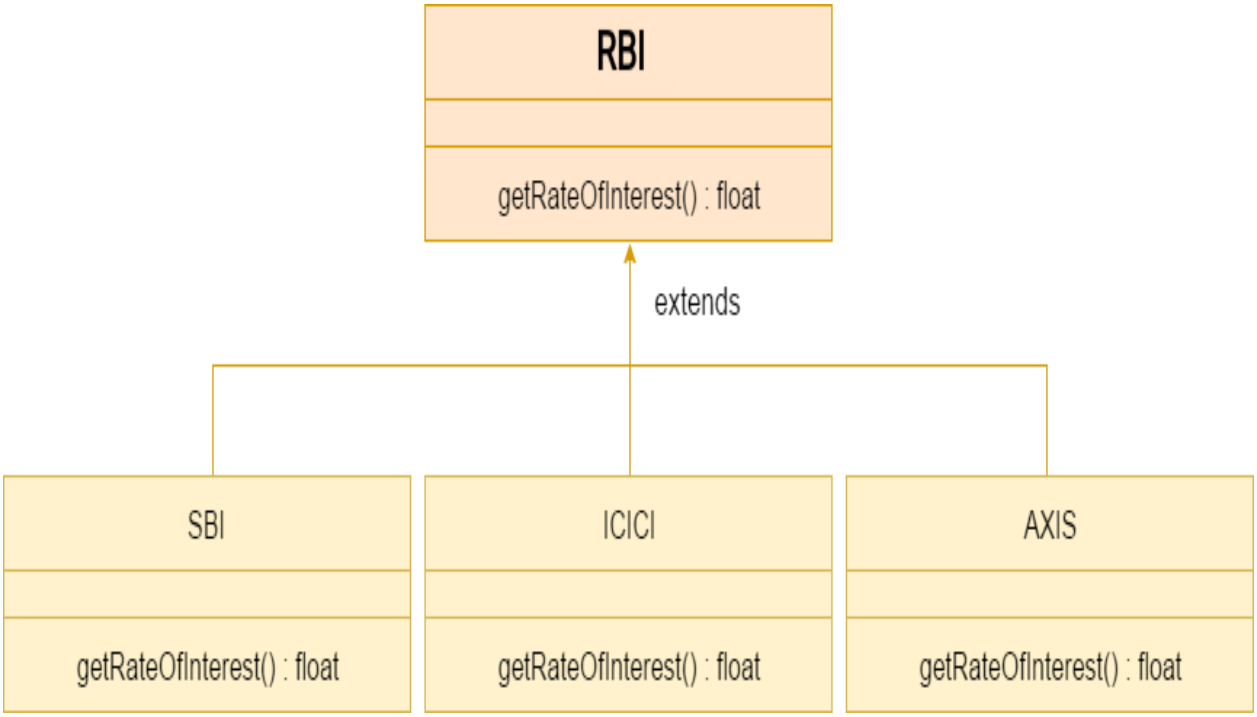
```
  void run()
  {
        System.out.println("Bike is running safely");
   }
  public static void main(String args[])
  {
  Bike2 obj = new Bike2();
  obj.run();
  }
}
```

Output:-

Bike is running safely

A real example of Java Method Overriding:-

Consider a scenario where RBI is a class that provides functionality to get the rate of interest. However, the rate of interest varies according to banks. For example, SBI, ICICI and AXIS banks could provide 8%, 7%, and 9% rate of interest.



//Java Program to demonstrate the real scenario of Java Method Overriding where three classes are overriding the method of a parent class.

```
//Creating a parent class.

class RBI

{

int getRateOfInterest()

{

return 0;

}

}

//Creating child classes.
```

```
class SBI extends RBI
{
int getRateOfInterest()
{
return 8;
}
}
class ICICI extends RBI
{
int getRateOfInterest()
{
return 7;
}
}
class AXIS extends RBI
{
int getRateOfInterest()
{
return 9;
}
```

```
}
//Test class to create objects and call the methods
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());
}
}
```

Output:-

```
SBI Rate of Interest: 8
ICICI Rate of Interest: 7
AXIS Rate of Interest: 9
```

Can we override static method?

No, a static method cannot be overridden. We can not override java main method because the main is a static method.

Why can we not override static method?

It is because the static method is bound with class whereas instance method is bound with an object. Static belongs to the class area, and an instance belongs to the heap area.

Difference between method overloading and method overriding in java.

There are many differences between method overloading and method overriding in java. A list of differences between method overloading and method overriding are given below:

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.
In java, method overloading can't be performed by changing return type of the method only. Return type can be same or different in method overloading. But you must have to change the parameter.	Return type must be same or covariant in method overriding.

Java Method Overloading example:-
class OverloadExample
{
static int add(int a,int b)
{
return a+b;
}
static int add(int a,int b,int c)
	Java Method Overriding example:-
	class Animal
	{
	void eat()
	{
	System.out.println("eating...");
	}
	}
	class Dog extends Animal

{
return a+b+c;
}
}
	{
	void eat()
	{
	System.out.println("eating bread...");
	}
	}

Using the super Keyword:-

When invoking a superclass version of an overridden method the super keyword is used.

Example:-

```
class Animal
{
  public void move()
  {
    System.out.println("Animals can move");
  }
}
class Dog extends Animal
{
```

```
  public void move()
{
     super.move();   // invokes the super class method
     System.out.println("Dogs can walk and run");
  }
}
public class TestDog
{
  public static void main(String args[])
  {
    Dog d = new Dog();
    d.move();
//Animal b = new Dog();// Animal reference but Dog object
 // b.move();   // runs the method in Dog class
  }
}
```

Output:-

```
Animals can move
Dogs can walk and run
```