Covariant Return Type

The covariant return type specifies that the return type may vary in the same direction as the subclass.

Before Java5, it was not possible to override any method by changing the return type. But now, since Java5, it is possible to override method by changing the return type if subclass overrides any method whose return type is Non-Primitive but it changes its return type to subclass type. Let's take a simple example:

**Simple example of Covariant Return Type**

class A{

A get(){return this;}

}

class B1 extends A{

B1 get()

{return this;}

void message(){System.out.println("welcome to covariant return type");

}

public static void main(String args[]){

new B1().get().message();

}

}

**Output**:welcome to covariant return type

As you can see in the above example, the return type of the get() method of A class is A but the return type of the get() method of B class is B. Both methods have different return type but it is method overriding. This is known as covariant return type.

**Imp:**

**Object oriented programming (OOP) has a principle named substitutability. In this tutorial, let us learn about substitutability and support for covariant return type in Java. Covariant return type uses the substitutability principle.**

**Liskov Substitution Principle**

**Substitutability was introduced by eminent Barbara Liskov and Jeannette Wing. It is also called as Liskov substitution principle.**

**Let T be a super type and S be its subtype (parent and child class). Then, instances (objects) of T can be substituted with instances of S. Parent’s instances can be replaced with the child’s instances without change in behavior of the program.**

**Covariant, Contravariant and Invariant**

**The subtyping principle which we discussed above as Liskov principle is called covariant. The reverse of it (instead of child replacing the parent, the reverse of it as parent replacing the child) is called contravariant. If no subtyping is allowed then, it is called invariant.**

**Example:**

class WildAnimal {

public String willYouBite(){

return "Yes";

}}

class Lion extends WildAnimal {

public String whoAreYou() {

return "Lion";

}}

class BengalTiger extends WildAnimal {

public String whoAreYou() {

return "Bengal Tiger";

}}

class Zoo {

WildAnimal getWildAnimal() {

return new WildAnimal();

} }

class AfricaZoo extends Zoo {

// @Override

Lion getWildAnimal() {

return new Lion();

}}

class IndiaZoo extends Zoo {

// @Override

BengalTiger getWildAnimal() {

return new BengalTiger();

}}

public class Covariant {

public static void main(String args[]){

AfricaZoo afZoo = new AfricaZoo();

System.out.println(afZoo.getWildAnimal().whoAreYou());

IndiaZoo inZoo = new IndiaZoo();

System.out.println(inZoo.getWildAnimal().whoAreYou());

}}