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:

Note: If you are beginner to java, skip this topic and return to it after OOPs concepts.

Simple example of Covariant Return Type

```
1. class A{
2. A get(){returnthis;}
3. }
4. class B1 extends A{
5. B1 get(){returnthis;}
6. void message(){System.out.println("welcome to covariant return type");}
7. publicstaticvoid main(String args[]){
8. new B1().get().message();
9. }
10. }
```

```
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.

How is Covariant return types implemented?

Java doesn't allow the return type based overloading but JVM always allows return type based overloading. JVM uses full signature of a method for lookup/resolution. Full signature means it includes return type in addition to argument types. i.e., a class can have two or more methods differing only by return type. javac uses this fact to implement covariant return types.