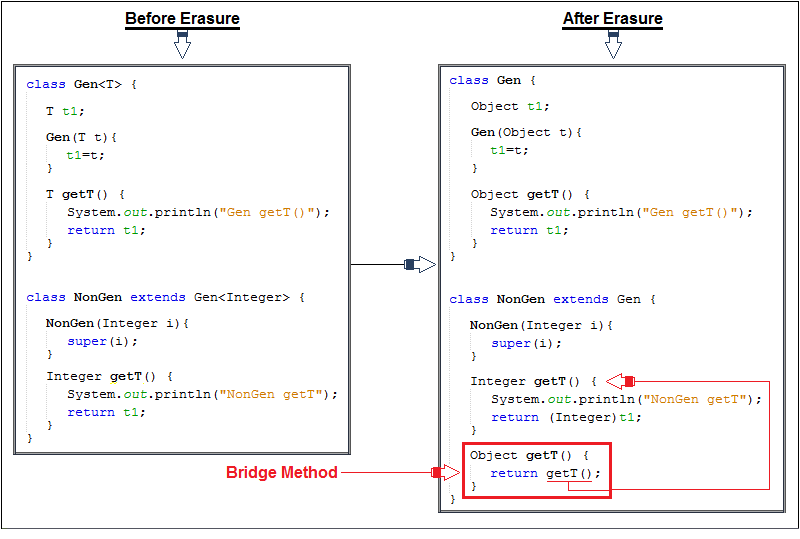


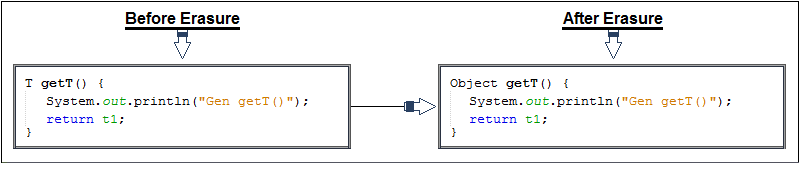
Java-Bridge Method

**Bridge Method**

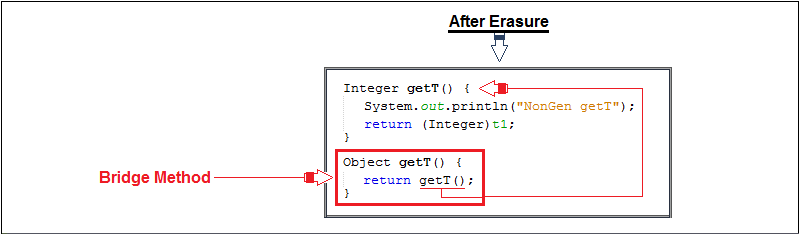
Sometimes, the compiler will need to add a bridge method to a class to handle situations in which the type erasure of an overriding method in a subclass does not produce the same erasure as the method in the superclass. In this case, a method is generated that uses the type erasure of the superclass, and this method calls the method that has the type erasure specified by the subclass. Of course, bridge methods only occur at the bytecode level, are not seen by you, and are not available for your use. Although bridge methods are not something that you will normally need to be concerned with, it is still instructive to see a situation in which one is generated. For example :

[](https://hajsoftutorial.com/?attachment_id=9792#main)

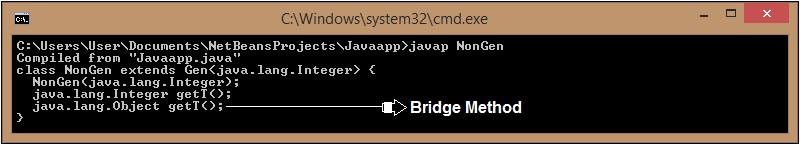
The subclass **NonGen** extends **Gen**, but does so using a **Integer**-specific version of **Gen**,***extends*** **Gen<Integer>.** Inside **NonGen**, **getT( )** is overridden with **Integer** specified as the return type. All of this is perfectly acceptable. The only trouble is that because of type erasure, the expected form of **getT( )** will be :

[](https://hajsoftutorial.com/java-bridge-method/23-6/#main)

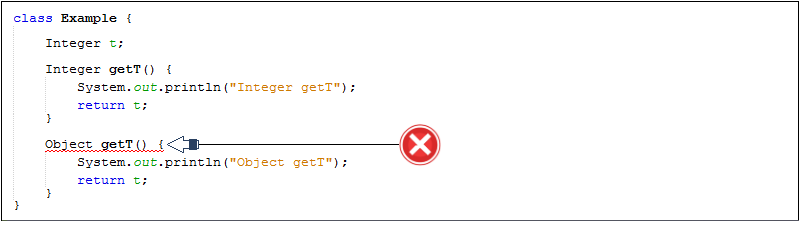
To handle this problem, the compiler generates a bridge method with the preceding signature that calls the **Integer** version:

[](https://hajsoftutorial.com/java-bridge-method/attachment/11211/#main)

The effect of the bridge method is to convert any calls to the inherited version of **getT( )** to a call to your version, thus making your attempt at overriding the base class method effective. This approach adopted by the compiler has implications for you. You must take care not to define methods in your derived class that have the same signature as an inherited method. Because the compiler changes the parameter types and return types involving type variables to their bounds, you must consider the inherited methods in these terms when you are defining your derived class methods. If you examine the class file for **NonGen** by using **javap**, you will see the bridge method:

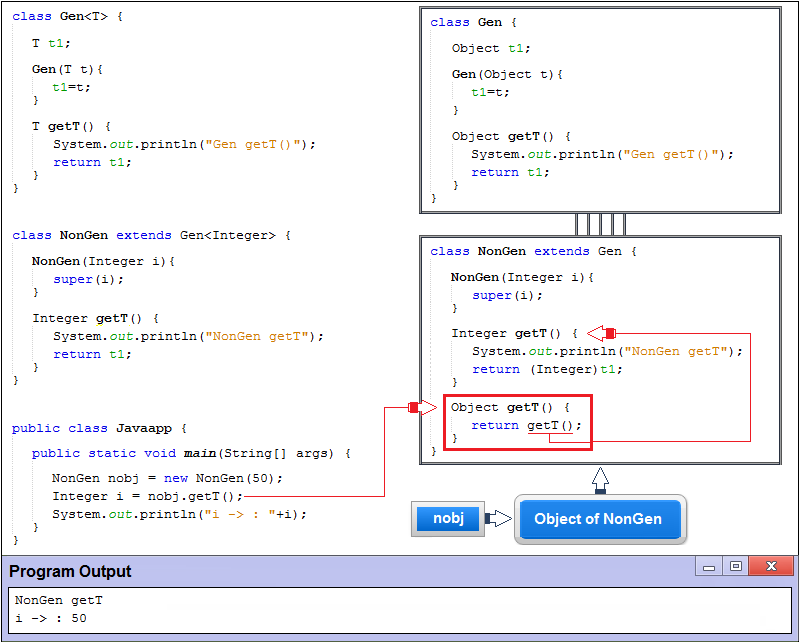
[](https://hajsoftutorial.com/java-bridge-method/23-7/#main)

As you can see, the bridge method has been included. There is one last point to make about this example. Notice that the only difference between the two **getT( )** methods is their return type. Normally, this would cause an error, for example :

[](https://hajsoftutorial.com/java-bridge-method/11211-copy/#main)

But because this does not occur in your source code, it does not cause a problem and is handled correctly by the **JVM**.

**Program**

[](https://hajsoftutorial.com/java-bridge-method/attachment/112/#main)