Nested Interface in Java

2.8

We can declare interfaces as member of a class or another interface. Such an interface is called as member interface or nested interface.

Interface in a class

Interfaces (or classes) can have only public and default access specifiers when declared outside any other class (Refer this for details). This interface declared in a class can either be default, public, private, protected. While implementing the interface, we mention the interface as **c_name.i_name** where **c_name** is the name of the class in which it is nested and **i_name** is the name of the interface itself.
Let us have a look at the following code:-

```
// Java program to demonstrate working of
// interface inside a class.
import java.util.*;
class Test
{
    interface Yes
    {
        void show();
    }
}

class Testing implements Test.Yes
{
    public void show()
    {
        System.out.println("show method of interface");
    }
}

class A
{
    public static void main(String[] args)
    {
        Test.Yes obj;
        Testing t = new Testing();
        obj=t;
        obj.show();
    }
}
```

Run on IDE

```
show method of interface
```

The access specifier in above example is default. We can assign public, protected or private also. Below is an example of protected. In this particular example, if we change access specifier to private, we get compiler error because a derived class tries to access it.

```
// Java program to demonstrate protected
// specifier for nested interface.
import java.util.*;
class Test
{
    protected interface Yes
    {
        void show();
    }
}

class Testing implements Test.Yes
{
    public void show()
    {
        System.out.println("show method of interface");
    }
}

class A
{
    public static void main(String[] args)
    {
        Test.Yes obj;
        Testing t = new Testing();
        obj=t;
        obj.show();
    }
}
```

Run on IDE

```
show method of interface
```

Interface in another Interface

An interface can be declared inside another interface also. We mention the interface as **i_name1.i_name2** where **i_name1** is the name of the interface in which it is nested and **i_name2** is the name of the interface to be implemented.

```
// Java program to demonstrate working of
// interface inside another interface.
import java.util.*;
interface Test
{
   interface Yes
   {
      void show();
   }
}

class Testing implements Test.Yes
{
   public void show()
   {
      System.out.println("show method of interface");
   }
}

class A
{
   public static void main(String[] args)
   {
     Test.Yes obj;
     Testing t = new Testing();
     obj = t;
     obj.show();
   }
}
```

Run on IDE

```
show method of interface
```

Note: In the above example, access specifier is public even if we have not written public. If we try to change access specifier of interface to anything other than public, we get compiler error. Remember, interface members can only be public..

```
// Java program to demonstrate an interface cannot
// have non-public member interface.
import java.util.*;
interface Test
{
    protected interface Yes
    {
        void show();
    }
}

class Testing implements Test.Yes
{
    public void show()
    {
        System.out.println("show method of interface");
    }
}

class A
{
    public static void main(String[] args)
    {
        Test.Yes obj;
        Testing t = new Testing();
        obj = t;
        obj.show();
    }
}
```

Run on IDE

```
illegal combination of modifiers: public and protected
   protected interface Yes
```