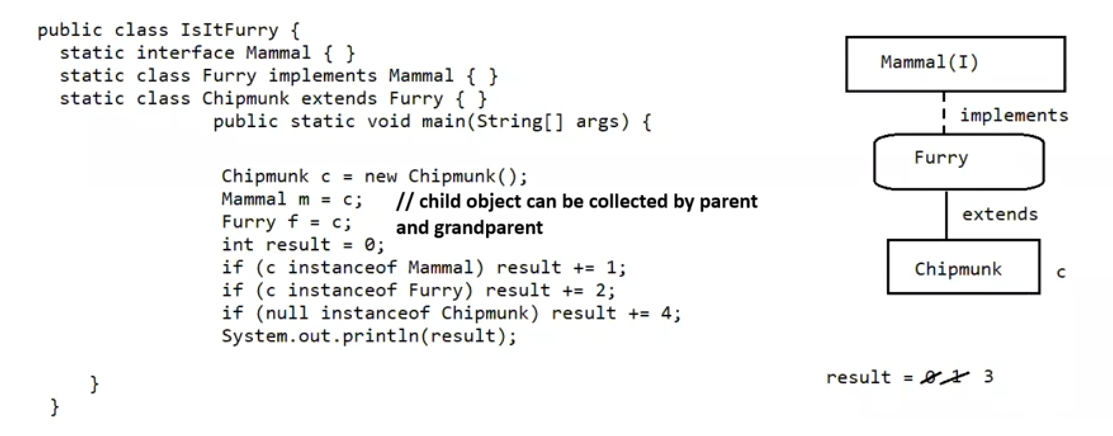


If the class implements interface and it doesn’t give the implementation for interface methods it must be marked as abstract. otherwise it leads to compile time error.

To give the implementation to the interface method, method signature and return type should be same otherwise it leads to compile time error.

Eg: IsItFurry



// Interface can be written inside class also, and interface written inside class can also be static.

Eg: Interface\_Eg3

//from jdk 1.8 we can have concrete methods in interface

Eg: Interface\_Eg4

// go through the code

Note: static methods are inherited in Java, but they are not overridden. Subclasses have access to static methods from their superclass, but redeclaring a static method with the same signature in the subclass creates a new method that hides the superclass method. ( class-level)

Eg: Interface\_Eg5

// go through the code

Eg: Interface\_Eg6

// go through the code

public interface InterfaceX {

public int geek();

}

public interface InterfaceY {

public String geek();

}

Now, Suppose we have a class that implements both those interfaces:

public class Testing implements InterfaceX, InterfaceY {

public String geek() {

return "hello";

}

}

The question is: Can a java class implement Two interfaces with same methods having the same signature but different return types?

No, its an error

If two interfaces contain a method with the same signature but different return types, then it is impossible to implement both the interface simultaneously.

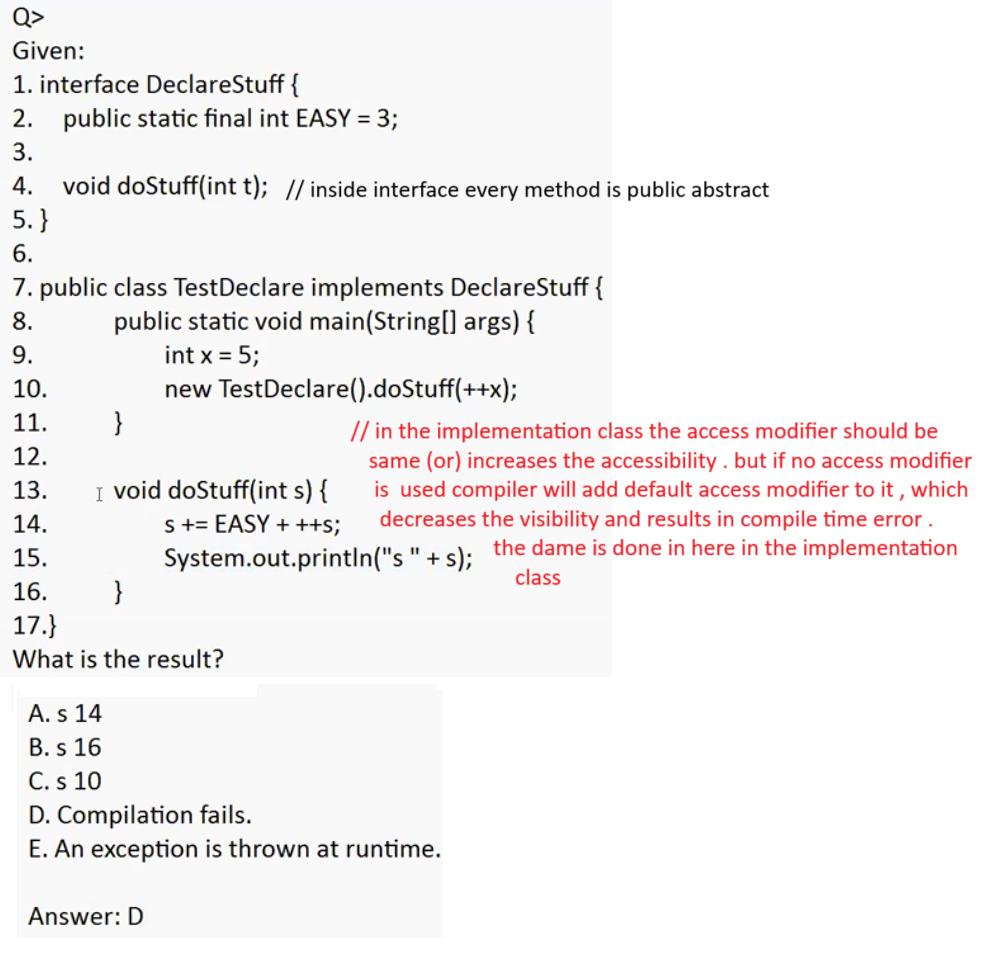
What if the methods have same signature and return type ?

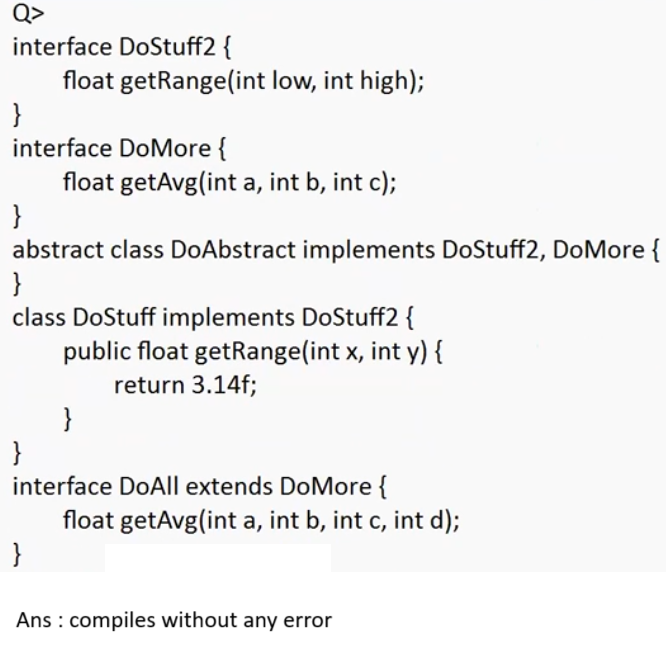
No problem absolutely works fine

If two interfaces having same method name return types, but with different argument list then they can be overridden separately in the implementation class

Eg: Interface\_Methods\_With\_Same\_Name\_Different\_Parameterd

// go through the code





Eg: Interface\_Concrete\_Static\_Private\_Methods

// go through the code

We can write the method body in the interface if the method is declared with default access modifier in the interface

The concrete method can be inherited in the implementation class.

Overriding the concrete method is optional in the implementation class. But if it needs to be overridden the overriding method should be given public modifier.

Implementation class cannot have default methods

why default methods in interface are allowed?

if there is requirement that which ever classes are going to provide the implementation for your methods and you want some body to be common all the implementation classes they we should use default methods in interface

from java 8 inside interface we can have static methods. it will not participate in the inheritance in the inherited clasess.

To invoke them we can have call with Interface.methodName

to write a static method in the interface only static keyword is sufficient, default keyword in not required (cannot be used).

in normal inheritance concept static methods will get inherited but cannot be overridden, method hiding concept.

Final modifier for method is not allowed in interface

In java 9 we can have private methods, they can’t be inherited, can’t be called outside interface

We can use it within interface other default methods, we can have private static methods also in interface

Static private method cannot be used in the private method of an interface

Because default method is a non-static context.

Private static methods can be used in the static methods of an interface