Exercise 01:

Declare an interface called “MyFirstInterface”. Decalre integer type variable called “x”. Declare an abstract method called “display()”.

1. Try to declare the variable with/without public static final keywords. Is there any difference between these two approaches? Why?

You can declare the variable with/without public static final keywords. The difference between these two approaches is that if you declare the variable with public static final keywords, it becomes a constant and its value cannot be changed. If you declare the variable without these keywords, it becomes a normal variable, and its value can be changed.

1. Declare the abstract method with/without abstract keyword. Is there any difference between these two approaches? Why?

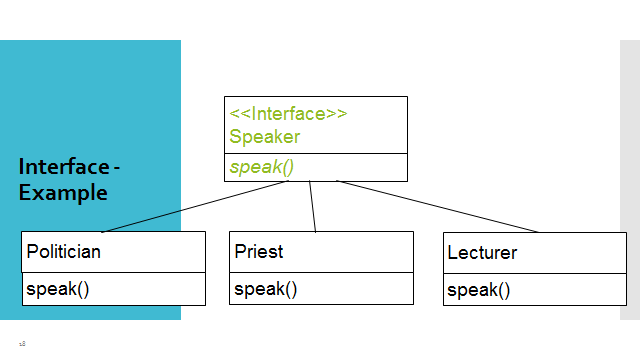
You must declare the abstract method with the abstract keyword. If you don’t use the abstract keyword, it becomes a normal method, and you must provide an implementation for it. An abstract method is a method that has no implementation and must be implemented by any class that implements the interface.

1. Implement this into a class called “IntefaceImplemented” . Override all the abstract methods. Try to change the value of x inside this method and print the value of x. Is it possible for you to change x? why?

You can change the value of x inside this method but it will not change the value of x in the interface because variables in an interface are static and final by default. Therefore, they cannot be changed once they are initialized.

Exercise 02:

Develop a code base for the following scenario. Recall what we have done at the lecture…



Exercise 03:

Try following code. What is the outcome? Why?

Class 01: Class 02:

final class Student { class Undergraduate extends Student{}

final int marks = 100;

final void display();

}

Output: The code will not compile because

Exercise 04:

Develop a code base for the following scenario. Shape class contains an abstract method called “calculateArea” and non-abstract method called “display”. Try to pass required values at the instantiation. Recall what we have done at the lecture…

