Practical 05: Inheritance & Abstract Classes

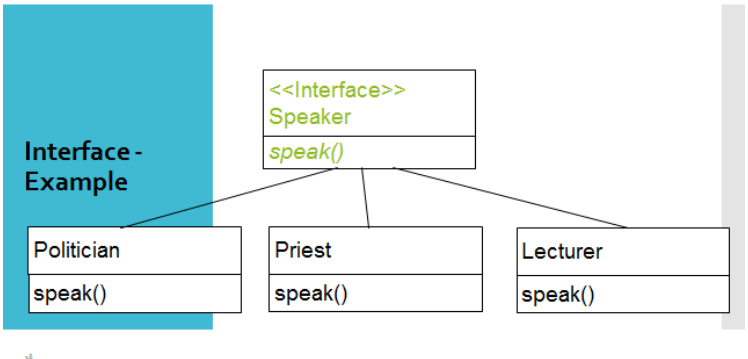
Exercise 01:

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?
    - There’s no difference between declaring the variable with or without the “public static final” keywords with the interface because when you declared a variable within an interface,it is automatically treated as “public ,static,final” and the behaviour remains the same.
2. Declare the abstract method with/without abstract keyword. Is there any difference between these two approaches? Why?
    - All the methods withina an interface are by default abstract and lack an implementation.
3. 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?
    - NO, because “X” variable is implicitly declared as final within the interface, it’s value cannot be changed by any class that implements the interface, including the “InterfaceImplemented” class. Therefore, your will get a compilation error if you try to modify the value of “within the implementation class.

Exercise 02:

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



Practical 05: Inheritance & Abstract Classes

Exercise 03:

Try following code. What is the outcome? Why?

Class 01:

```
final class Student {

    final int marks = 100;

    final void display();

}
```

Class 02:

```
class Undergraduate extends Student{}
```

** In here student class marked as “final” which means it cannot be subclassed. Because of that “Undergraduate” class will result in compilation error.

** To fix the issue we can remove “final” keyword from the “student” class which intended to be subclassed.

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…