**Practical 05**

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 and final keywords. Is there any difference between these two approaches? Why?

* Yes.
* If we declared the variable as static, it can be reinitialized but if we declared the variable as final it cannot be reinitialized. On the other hand, final variables cannot be inherited since the static variables can access only the static members of the class and static methods can only call them.

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

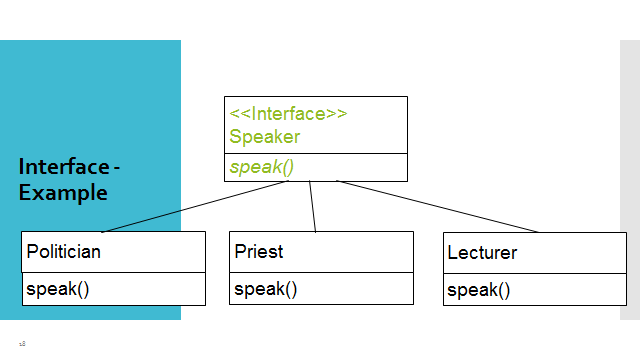
* No.
* We already created an interface which is a set of abstract methods so it is not necessary and there is no difference between using abstract keyword or not.

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?

* No. it cannot be changed.
* Because it is declared as ‘final’ in the interface and so the final keyword makes the variable constant. So the variable ‘x’ cannot be modified. If we try to do so it gives a compilation error.

Exercise 02:

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



**public interface Speaker**

**{**

**public void speak();**

**}**

**public class Politician implements Speaker**

**{**

**public void speak()**

**{**

**System.out.println("Talks politics");**

**}**

**}**

**public class Priest implements Speaker**

**{**

**public void speak()**

**{**

**System.out.println("Talks religious");**

**}**

**}**

**public class Lecturer implements Speaker**

**{**

**public void speak()**

**{**

**System.out.println("Talks OOP programming");**

**}**

**}**

**//main method**

**public class Lab05**

**{**

**public static void main(String[] args)**

**{**

**Lecturer l1=new Lecturer();**

**l1.speak();**

**Politician p1=new Politician();**

**p1.speak();**

**Priest pr1=new Priest();**

**pr1.speak();**

**}**

**}**

Exercise 03:

Try the following code. What is the outcome? Why?

Class 01: Class 02:

final class Student { class Undergraduate extends Student {}

final int marks = 100;

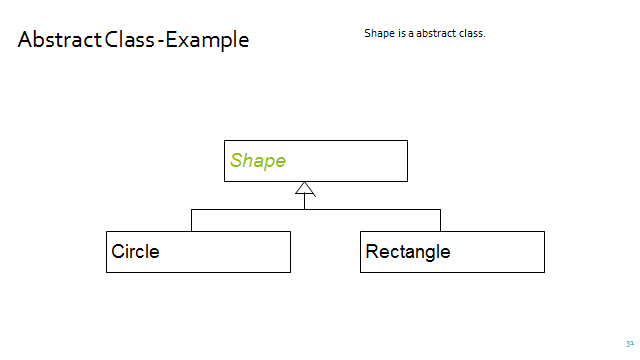
final void display ();

}

* This code gives a compilation error. Because class 01 is declared as final and class 02 is extended class 01. Since the final classes cannot be sub-classed this code will result in a compilation error.

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…



public abstract class Shape

{

public abstract double calculateArea();

public void display()

{

}

}

public class Circle extends Shape

{

private double r;

protected static final double pi=3.14159;

public void circle(double r)

{

r=this.r;

}

public double calculateArea()

{

return pi\*r\*r;

}

}

public class Rectangle extends Shape

{

private double h,w;

public void rect(double h,double w)

{

h=this.h;

w=this.w;

}

public double calculateArea()

{

return w\*h;

}

}