**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?

**In Java interfaces ,all variables are implicity public,static and final.Therefore ,can declare the variable “x” directly within the interface without using the public static final keywords.**

**Both approches are essentially the same in terms of the resulting behaviour.In java interfaces, variables are by default public,static and final.So including the keywords or omitting them doesn’t change the nature of the variable.**

**package com.mycompany.labobj6;**

**public interface MyFirstInterface {**

**int x=10;**

//final public static int x=10;

//All the variables declared inside an interface,by default final public static whether it is implicitly stated or not.

**void display();**

**}**

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

**The Abstract method “display()” is declared within the interface without using the “abstract” keyword explicity.In Java,when you declare a method inside an interface, it is implicity considered as an abstract method .Therefore,the abstract keyword is not required when declaring methods inside an interface.**

**In java,when declaring methods inside an interface:**

* **The methods are implicity considered abstract,even if the “abstract” keyword is not explicity used.**
* **The “abstract” keyword is not required when declaring methods inside an interface.**
* **Interface methods are by default abstract and must be implemented by the classes that implement the interface.**

**Therefore,there is no difference in functionality between declaring an abstract method with or without the “abstract” keyword inside an interface.Both approches indicate that the method is abstract and must be implemented by the implementing classes.**

**package com.mycompany.labobj6;**

**public interface MyFirstInterface {**

**int x=10;**

**//final public static int x=10;**

**//All the variables declared inside an interface,by default final public static whether it is implicitly stated or not.**

**void display();**

//abstract void display();

//All the methods declares inside an interface, all are abstract whether it is implicitly 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 possibssle for you to change x? why?

**package com.mycompany.labobj6;**

**public class InterfaceImplemented implements MyFirstInterface{**

**@Override**

**public void display(){**

**x++;**

//The value x cannot be changed,because it is declared in an interface.

//The variable which initialize inside the interface ,it's value cannot be changed.

**System.out.println("Value: "+x);**

**}**

**}**

Object

**package com.mycompany.labobj6;**

**public class Labobj6 {**

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

**InterfaceImplemented Im=new InterfaceImplemented();**

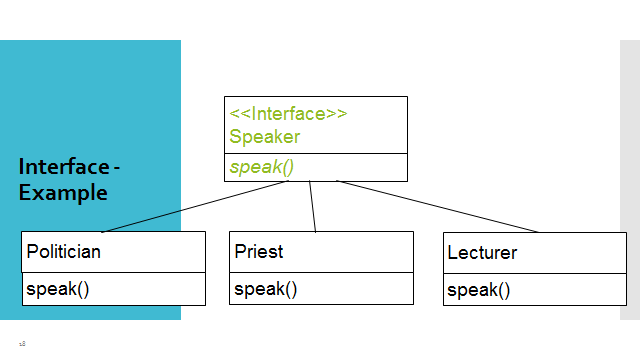
**Im.display();**

**}**

**}**

**Exercise 02:**

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



Interface

**package com.mycompany.practical6;**

**public interface Speaker {**

**int i =100;**

**void speak(String line);**

**}**

Politioian Class

**package com.mycompany.practical6;**

**public class Politioian implements Speaker {**

**@Override**

**public void speak(String phrase){**

**System.out.println(i+"Politioian: "+phrase);**

**}**

**}**

Priest Class

**package com.mycompany.practical6;**

**public class Priest implements Speaker{**

**@Override**

**public void speak(String phrase)**

**{**

**//i=200**

**System.out.println(i+"Priest: "+phrase);**

**}**

**}**

Lecturer Class

**package com.mycompany.practical6;**

**public class Lecturer implements Speaker {**

**@Override**

**public void speak(String phrase)**

**{**

**System.out.println(i+"Lecturer: "+phrase);**

**}**

**}**

Object

**package com.mycompany.practical6;**

**public class Practical6 {**

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

**Speaker speaker1= new Priest();**

**speaker1.speak("Bless You!");**

**Speaker speaker2= new Politioian ();**

**speaker2.speak("Vote me!");**

**Speaker speaker3= new Lecturer ();**

**speaker3.speak("Today we are going to learn Java");**

**}**

**}**

**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();

}

Answer

**package com.mycompany.execise3;**

**final class Student {**

**final int marks = 100;**

**final void display();**

//The “final” keyword is used to restrict the modification of a class,method,or variable.

//In the code “Student” class is declared as ‘final’ ,which means it cannot be inherited by any other class.

}

**package com.mycompany.execise3;**

**class Undergaduate extends Student**

**{**

//’Undergraduate’ class trying to extend the ‘Student’ class.But it not allowed.

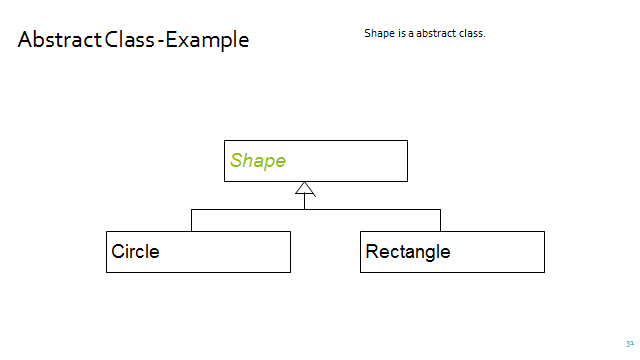
//Because ‘Student’ is marked as ‘final’.

//As a result,a compilation error will occur.

**}**

**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…



**package com.mycompany.shapemain;**

**abstract class Shape {**

**abstract double calculateArea();**

**void display()**

**{**

**System.out.println("Shape");**

**}**

**}**

**package com.mycompany.shapemain;**

**public class Circle extends Shape{**

**private double radius;**

**public Circle(double r)**

**{**

**radius=r;**

**}**

**public double calculateArea()**

**{**

**return Math.PI\*radius\*radius; //you can add manually value also(3.14)**

**}**

**}**

**package com.mycompany.shapemain;**

**public class Rectangle extends Shape{**

**private double length,width;**

**public Rectangle (double h,double w)**

**{**

**length=h;**

**width=w;**

**}**

**public double calculateArea()**

**{**

**return length\*width;**

**}**

**}**

**package com.mycompany.shapemain;**

**public class Shapemain {**

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

**Circle c=new Circle(7.0);**

**double circleArea=c.calculateArea();**

**c.display();**

**System.out.println("Circle area is: "+c.calculateArea());**

**Rectangle r=new Rectangle(5.0,5.0);**

**double rectangleArea=r.calculateArea();**

**r.display();**

**System.out.println("Reactangle area is: "+r.calculateArea());**

**}**

**}**