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?

No,

When declaring a variable within an interface it’s always public, static and final.

There for it’s no difference between declaring a variable with or without public static final keywords.

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

No,

Declaring abstract methods with or without abstract methods are not any difference, because methods in interface are by default in abstract.

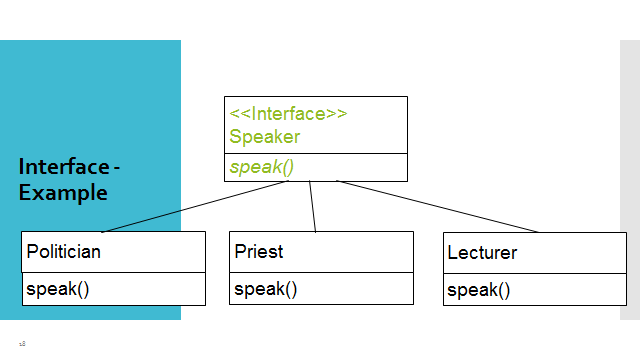
1. Implement this into a class called “IntefaceImplemented” . Override all the abstract methods. Try to change the value of x inside this method and print the value of x. Is it possible for you to change x? why?

No,

In interfaces, variables are always final. So, we can’t assign a value to the final variable.

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{}

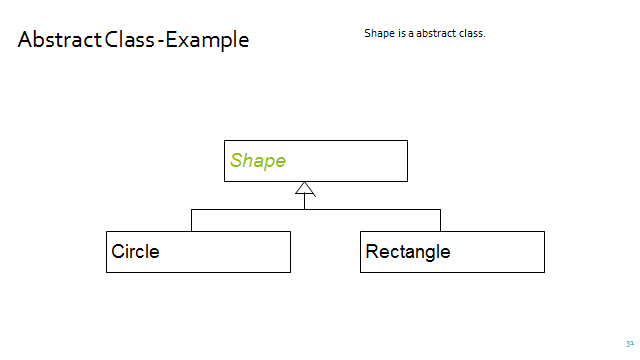
final int marks = 100;

final void display();

}

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 {  
 abstract double calculateArea();  
  
 public void display(){  
 System.*out*.println("Area: "+calculateArea());  
 }  
}

public class Circle extends Shape{  
 private int radius;  
  
 public Circle(int radius){  
 this.radius = radius;  
 }  
 @Override  
 double calculateArea(){  
 return Math.*PI*\*radius\*radius;  
 }  
}

public class Rectangle extends Shape{  
 private int width;  
 private int height;  
  
 public Rectangle(int width, int height){  
 this.height = height;  
 this.width = width;  
 }  
 @Override  
 double calculateArea() {  
 return height\*width;  
 }  
}

public class FindArea {  
 public static void main(String[] args){  
 var circle1 = new Circle(7);  
 circle1.display();  
  
 var rectangle1 = new Rectangle(5,6);  
 rectangle1.display();  
  
 }  
}