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?

Public static final is default method, with or without not different.

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

With or without abstract keyword is not different in 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?

Not possible because inside interface declared x is final one. In get default final

Answer

package com.mycompany.interfaceobj;

public interface MyFirstInterface

{

int x=5;

abstract void display();

}

package com.mycompany.interfaceobj;

public class InterfaceImplemented implements MyFirstInterface

{

public void display()

{

System.out.println("X is:"+x);

}

}

package com.mycompany.interfaceobj;

public class InterfaceObj

{

public static void main(String[] args)

{

InterfaceImplemented i1= new InterfaceImplemented();

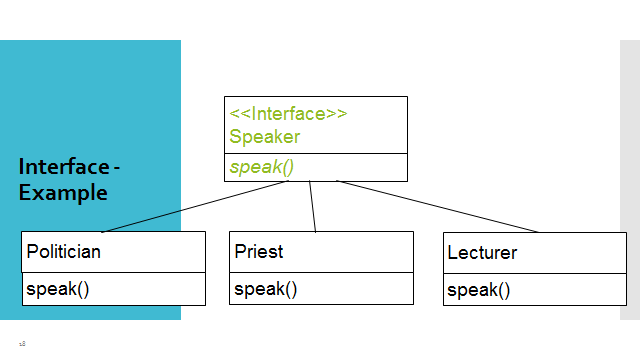
i1.display();

}

}

Exercise 02:

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



Answer:

package com.mycompany.speakerobj;

public class SpeakerObj

{

public static void main(String[] args)

{

Politician p1=new Politician();

p1.speak("hi");

Priest p2=new Priest();

p2.speak("good");

Lecturer p3= new Lecturer();

p3.speak("stand up");

}

}

package com.mycompany.speakerobj;

public interface Speaker

{

public void speak(String phrase);

}

package com.mycompany.speakerobj;

public class Politician implements Speaker

{

@Override

public void speak(String phrase)

{

System.out.println("Politician:"+phrase);

}

}

package com.mycompany.speakerobj;

public class Lecturer implements Speaker

{

@Override

public void speak(String phrase)

{

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

}

}

package com.mycompany.speakerobj;

public class Priest implements Speaker

{

@Override

public void speak(String phrase)

{

System.out.println("Preist:"+phrase);

}

}

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:

Student class:

This class is declared as final, which means it cannot be subclassed or extended. It has a final instance variable marks with a value of 100. It also has a final method display() that prints the value of marks. The final keyword in the method declaration prevents any subclasses from overriding this method.

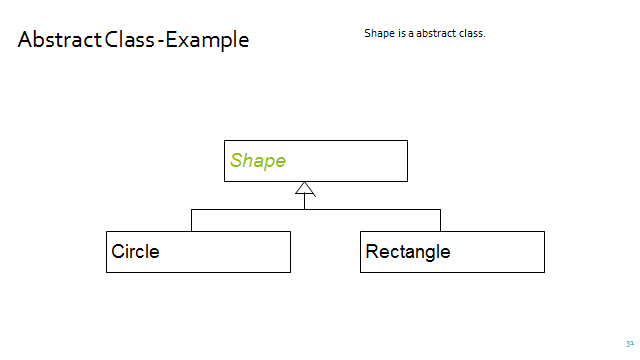
Undergraduate class:

This class extends the Student class. However, since the Student class is declared as final, it cannot be extended. Therefore, the code will result in a compilation error.

The outcome of trying to compile this code will be a compilation error due to the attempt to extend a final class. The error message will indicate that the class Student cannot be subclassed or extended.

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…



Answer:

package com.mycompany.shapeobject;

abstract class Shape

{

public abstract float calculateArea();

public void display()

{

System.out.println("Area:"+calculateArea());

}

}

public class Circle extends Shape

{

final float pi=3.14f;

private float radius;

public Circle(float r)

{

radius =r;

}

@Override

public float calculateArea()

{

return pi\*radius\*radius;

}

}

public class Rectangle extends Shape

{

private float width,length;

public Rectangle(float w,float l)

{

width=w;

length=l;

}

@Override

public float calculateArea()

{

return width\*length;

}

}

public class ShapeObject {

public static void main(String[] args)

{

Circle c=new Circle(7);

c.display();

Rectangle r=new Rectangle(2,5);

r.display();

}

}