Practical-05

T.A.Sandali Sewmini-27745

Exercise-01

```
public interface MyFirstInterface
 {
   int x = 10; // Integer type variable
   void display(); // Abstract method declaration
}
```

1)Declaring the variable without public static final keywords:

```
public interface MyFirstInterface
{
 int n= 10;
}
```

-) Declaring the variable with public static final keywords:

```
public interface MyFirstInterface
 {
public static final int n = 10;
}
```

There is no practical difference between these two approaches. In both cases, the variable "n" will be treated as a constant value that is accessible to implementing classes, and its value cannot be changed once it is assigned.

2)

1. Declaring the abstract method without the **abstract** keyword:

```
public interface MyFirstInterface
{
 void display();
 }
```

2. Declaring the abstract method with the **abstract** keyword:

```
public interface MyFirstInterface
 {
abstract void display();
}
```

There is no practical difference between these two approaches. In both cases, the method **display()** will be treated as an abstract method, which means that any class implementing the interface must provide an implementation for this method.

3)

```
public interface MyFirstInterface {
   int x = 10; // Integer type variable


   void display(); // Abstract method declaration
}

public class InterfaceImplemented implements MyFirstInterface {
    @Override
    public void display() {
        // Attempt to change the value of x
        // x = 20; // This will cause a compilation error

        // Print the value of x
        System.out.println(x);
    }
}
```

Exercise-02

Main-

```
package com.mycompany.main4;
public class Main4 {

    public static void main(String[] args)
    {
        Politician p1=new Politician();
        p1.speak();
        Priest p2=new Priest();

        p2.speak();

        Lecturer L1=new Lecturer();

        L1.speak();


    }

}
```

-

```
package com.mycompany.main4;

public interface Speaker

{

    public void speak();

}
```

-

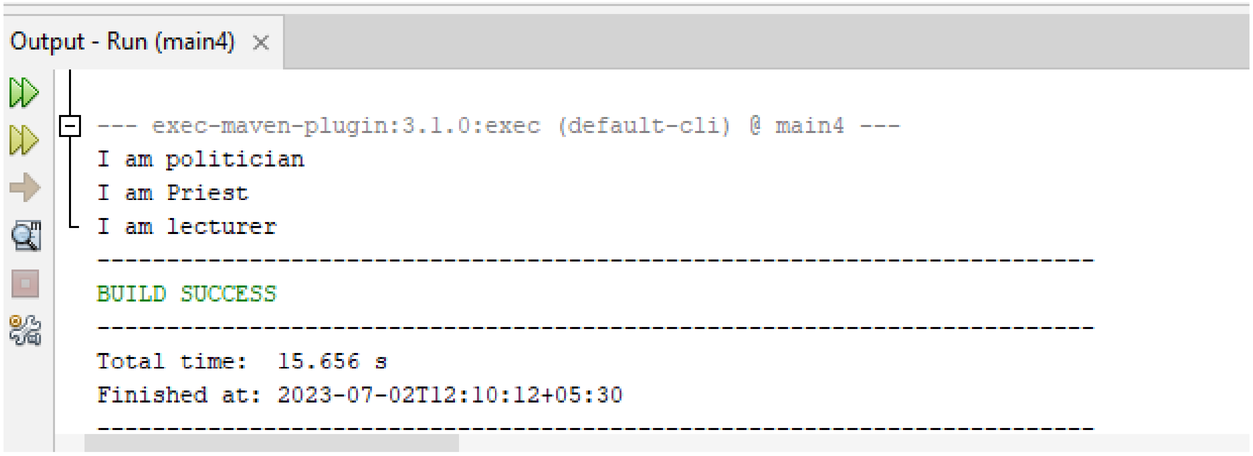
```
package com.mycompany.main4;

public class Politician implements Speaker

{

@Override

public void speak()

{

    System.out.println("I am politician");

}
}
```

-

```
package com.mycompany.main4;
public class Priest implements Speaker
{
  @Override
  public void speak()
  {
      System.out.println("I am Priest");
  }
}
```

-

```
package com.mycompany.main4;
public class Lecturer implements Speaker
{
 @Override
   public void speak()
{
    System.out.println("I am lecturer");
}
}
```



```
--- exec-maven-plugin:3.1.0:exec (default-cli) @ main4 ---
I am politician
I am Priest
I am lecturer
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  15.656 s
Finished at: 2023-07-02T12:10:12+05:30
------------------------------------------------------------------------
```

Exercise-03

Try following code. What is the outcome? Why?

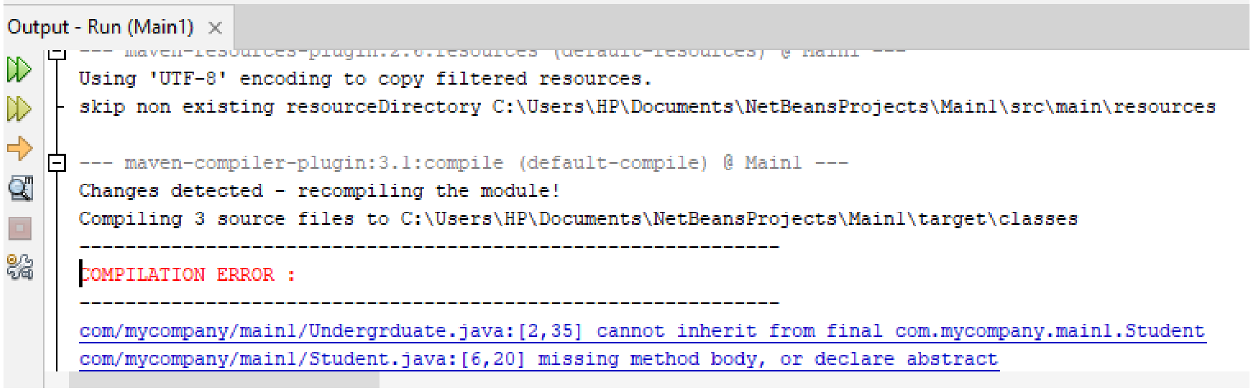
Class 01:

final class Student {

final int marks = 100; final void display(); }

Class 02:

class Undergraduate extends Student{}



```
--- maven-resources-plugin:2.6:resources (default-resources) @ Main1 ---
Using 'UTF-8' encoding to copy filtered resources.
skip non existing resourceDirectory C:\Users\HP\Documents\NetBeansProjects\Main1\src\main\resources

--- maven-compiler-plugin:3.1:compile (default-compile) @ Main1 ---
Changes detected - recompiling the module!
Compiling 3 source files to C:\Users\HP\Documents\NetBeansProjects\Main1\target\classes
-------------------------------------------------------------
COMPILATION ERROR :
-------------------------------------------------------------
com/mycompany/main1/Undergrduate.java:[2,35] cannot inherit from final com.mycompany.main1.Student
com/mycompany/main1/Student.java:[6,20] missing method body, or declare abstract
```

Correct code

```
package com.mycompany.main1;
public class Main1 {

  public static void main(String[] args)
  {
   Student student = new Student();
    student.display();

    Undergraduate undergraduate = new Undergraduate();
    undergraduate.display();
  }
}
```

- package com.mycompany.main1;

```
final class Student
  {

       final int marks = 100;
       final void display()
      {
      System.out.println("Marks is"+marks);
      }
  }
```

-
  ```
  package com.mycompany.main1;
  class Undergraduate extends Student
  {

  }
  ```

Exercise-04

- Main class-
  ```
  package com.mycompany.findarea;
  public class Findarea {

     public static void main(String[] args)
     {
       Circle c1=new Circle(2);
       c1.display();
       Rectangle r1=new Rectangle(5,4);
       r1.display();
     }
  }
  ```
- Shape class(parent class)
  ```
  package com.mycompany.findarea;
  public abstract class Shape
  {
     public abstract double calculateArea();
     public void display()
     {
        System.out.println("Area:"+calculateArea());
     }
  }
  ```

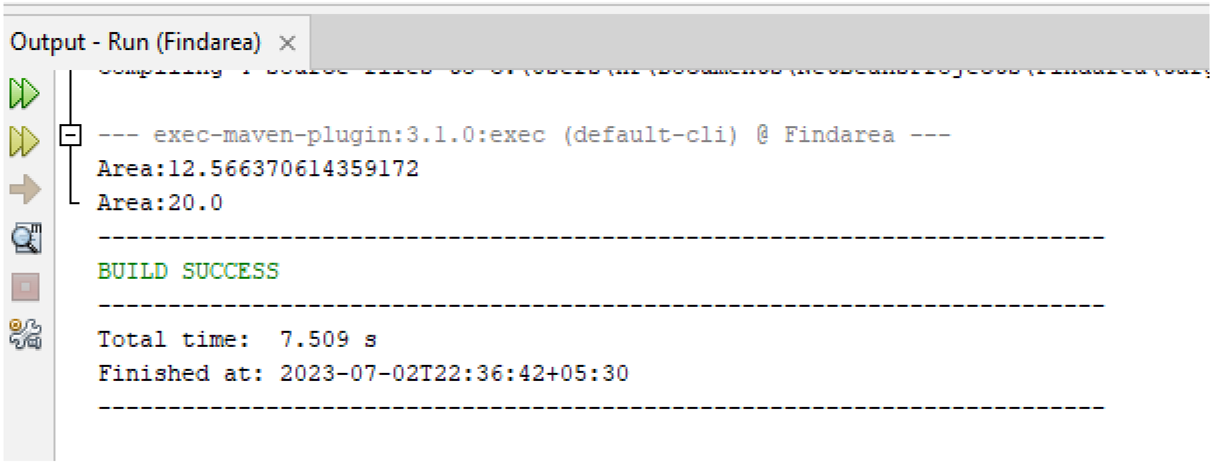
- Circle class(child class)

```
package com.mycompany.findarea;
public class Circle extends Shape
{
  private double rad;
  public Circle(double rad)
  {
     this.rad=rad;
  }
  @Override
  public double calculateArea()
  {
     return Math.PI*rad*rad;
  }
}
```

- Rectangle class(child class)

```
package com.mycompany.findarea;
public class Rectangle extends Shape
{
  private double width,height;
  public Rectangle(double width,double height)
  {
     this.width=width;
     this.height=height;
  }
  @Override
  public double calculateArea()
  {
     return width*height;
  }
```

}



```
--- exec-maven-plugin:3.1.0:exec (default-cli) @ Findarea ---
Area:12.566370614359172
Area:20.0
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  7.509 s
Finished at: 2023-07-02T22:36:42+05:30
------------------------------------------------------------------------
```