

**Government College of Engineering, Jalgaon**  
**Department of Computer Engineering**  
**Experiment No: 04**

**Name:**  
**Subject:**CO310U (Application programming Lab)  
**Class:**T.Y. B.Tech  
**Date of Performance:**

**PRN:**  
**Sem:**V(Odd)  
**Academic Year:**2024-25  
**Date of Completion:**

**Aim:** Write a program for the following

- a. Write a java program for abstract class to find areas of different shape
- b. Write a java program to give examples for “super” keywords.
- c. Write a java program to implement Interface.
- d. Write a program for the implementation of Multiple inheritance using interfaces to calculate the area of a rectangle and triangle

**Required Software:** OpenJDK version "1.8.0\_131"

OpenJDK Runtime Environment (build 1.8.0\_131-8u131-b11-2ubuntu1.16.04.3-b11)

OpenJDK 64-Bit Server VM (build 25.131-b11, mixed mode)

**Java Compiler Version - JAVAC 1.8.0\_131**

**Theory:**

A class which is declared with the abstract keyword is known as an abstract class in Java. It can have abstract and non-abstract methods (method with the body). Abstraction is a process of hiding the implementation details and showing only functionality to the user. Another way, it shows only essential things to the user and hides the internal details, for example, sending SMS where you type the text and send the message. You don't know the internal processing about the message delivery. Abstraction lets you focus on what the object does instead of how it does it.

**Ways to achieve Abstraction**

There are two ways to achieve abstraction in java

1. Abstract class (0 to 100%)
2. Interface (100%)

A class which is declared as abstract is known as an abstract class. It can have abstract and non-abstract methods. It needs to be extended and its method implemented. It cannot be instantiated. An abstract class must be declared with an abstract keyword.

- It can have abstract and non-abstract methods.
- It cannot be instantiated.
- It can have constructors and static methods also.
- It can have final methods which will force the subclass not to change the body of the method

Example of abstract class

### 1. `abstract class A{ }`

A method which is declared as abstract and does not have implementation is known as an abstract method. Example of abstract method

### 1. `abstract void printStatus();` //no method body and abstract

## Super keyword:

The super keyword in Java is a reference variable which is used to refer to an immediate parent class object. Whenever you create the instance of a subclass, an instance of the parent class is created implicitly which is referred to by a super reference variable.

## Usage of Java super Keyword

1. super can be used to refer to the immediate parent class instance variable.
2. super can be used to invoke immediate parent class methods.
3. super() can be used to invoke immediate parent class constructor.

We can use super keywords to access the data member or field of the parent class. It is used if parent class and child class have the same fields. The super keyword can also be used to invoke the parent class method. It should be used if the subclass contains the same method as the parent class. In other words, it is used if the method is overridden. The super keyword can also be used to invoke the parent class constructor.

## Interface in JAVA:

An interface in Java is a blueprint of a class. It has static constants and abstract methods. The interface in Java is *a mechanism to achieve abstraction*. There can be only abstract methods in the Java interface, not method body. It is used to achieve abstraction and multiple inheritance in Java. In other words, you can say that interfaces can have abstract methods and variables. It cannot have a method body. If a class implements multiple interfaces, or an interface extends multiple interfaces, it is known as multiple inheritance. Multiple inheritance is not supported in the case of class because of ambiguity. However, it is supported in case of an interface because there is no ambiguity. It is because its implementation is provided by the implementation class.

- Java Interface also represents the IS-A relationship.
- It cannot be instantiated just like the abstract class.
- Since Java 8, we can have default and static methods in an interface.
- Since Java 9, we can have private methods in an interface.

## Why use Java interface?

There are mainly three reasons to use interfaces. They are given below.

- It is used to achieve abstraction.
- By interface, we can support the functionality of multiple inheritance.
- It can be used to achieve loose coupling.

### How to declare an interface?

An interface is declared by using the interface keyword. It provides total abstraction; means all the methods in an interface are declared with the empty body, and all the fields are public, static and final by default. A class that implements an interface must implement all the methods declared in the interface.

### Conclusion:

---

---

---

---

---

**Name & Sign of Course Teacher**



**Program :A**

```

abstract class Shape {
    abstract double area();
}

class Rectangle extends Shape {
    double length = 12.5, breadth = 2.5;

    double area() {
        return length * breadth;
    }
}

class Triangle extends Shape {
    double base = 4.2, height = 6.5;

    double area() {
        return 0.5 * base * height;
    }
}

class Square extends Shape {
    double side = 6.5;

    double area() {
        return side * side; // Area of a square is side^2
    }
}

class p4{
    public static void main(String[] args) {
        Rectangle r1 = new Rectangle();
        Triangle t1 = new Triangle();
        Square s1 = new Square();

        System.out.println("The area of the rectangle is: " + r1.area());
        System.out.println("The area of the triangle is: " + t1.area());
        System.out.println("The area of the square is: " + s1.area());
    }
}

```

**Output:**

```

koliv@J4RVIS MINGW64 /d/Codes/APL
$ javac p4.java

koliv@J4RVIS MINGW64 /d/Codes/APL
$ java p4.java
The area of the rectangle is: 31.25
The area of the triangle is: 13.65
The area of the square is: 42.25

```

**Program :B**

```
class A {
    int length, breadth;

    A() {
        length = 10;
        breadth = 20;
    }
}

class B extends A {
    int height;

    B() {
        super(); // Calls the constructor of class A
        height = 30;
    }

    int volume() {
        return length * breadth * height;
    }
}

public class p4 {
    public static void main(String[] args) {
        B b1 = new B();
        int r = b1.volume();
        System.out.println("The volume is: " + r);
    }
}
```

**Output:**

```
koliv@J4RVIS MINGW64 /d/Codes/APL
$ javac p4.java

koliv@J4RVIS MINGW64 /d/Codes/APL
$ java p4.java
The volume is: 6000
```

**Program :C**


```

class B implements A {
    public void display() {
        System.out.println("B's method");
    }
}

class C extends B {
    public void callMe() {
        System.out.println("C's method");
    }
}

public class p4 {
    public static void main(String[] args) {
        C c1 = new C();
        c1.display(); // Calls the display method from class B
        c1.callMe();  // Calls the callMe method from class C
    }
}

```

**Output:**


```

koliv@J4RVIS MINGW64 /d/Codes/APL
$ javac p4.java

koliv@J4RVIS MINGW64 /d/Codes/APL
$ java p4
B's method
C's method

```

**Program :D**

```

import java.io.*;

interface Area {
    float compute(float x, float y);
}

class Rectangle {
    public float compute(float x, float y) {
        return (x * y);
    }
}

class Triangle {
    public float compute(float x, float y) {
        return (x * y / 2);
    }
}

```

```

class Result extends Rectangle implements Area {
    public float compute(float x, float y) {
        return super.compute(x, y); // Calls the compute method from Rectangle
    }
}

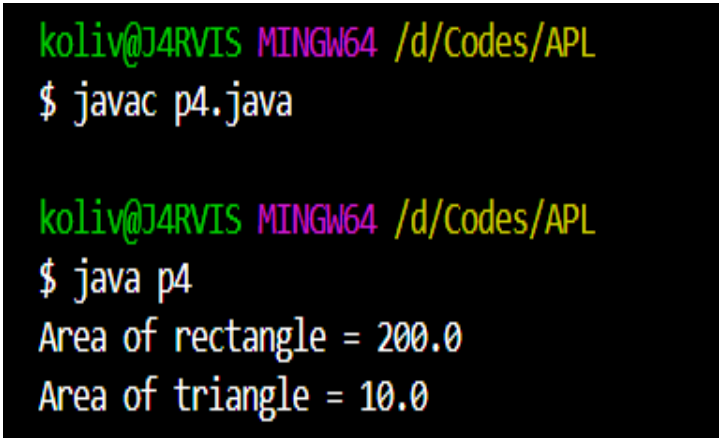
class Result1 extends Triangle implements Area {
    public float compute(float x, float y) {
        return super.compute(x, y); // Calls the compute method from Triangle
    }
}

public class p4 {
    public static void main(String[] args) {
        Result rect = new Result();
        Result1 tri = new Result1();
        Area a;
        a = rect;
        System.out.println("Area of rectangle = " + a.compute(10, 20));

        a = tri;
        System.out.println("Area of triangle = " + a.compute(10, 2));
    }
}

```

Output:



```

koliv@J4RVIS MINGW64 /d/Codes/APL
$ javac p4.java

koliv@J4RVIS MINGW64 /d/Codes/APL
$ java p4
Area of rectangle = 200.0
Area of triangle = 10.0

```