

<b>Course Name:</b>	<b>Object Oriented Programming</b>	<b>Semester:</b>	<b>III</b>
<b>Date of Performance:</b>	<b>11/8/2025</b>	<b>Batch No:</b>	<b>Batch A1</b>
<b>Faculty Name:</b>	<b>Prof. Amrita Naiksatam</b>	<b>Roll No:</b>	<b>20</b>
<b>Faculty Sign &amp; Date:</b>		<b>Grade/Marks:</b>	<b>___/15</b>

**Experiment No: 4**  
**Title: Concept of Inheritance and Abstract class**

<b>Aim and Objective of the Experiment:</b>
Learn the concept of Inheritance, method overriding and Abstract class in Java

<b>COs to be achieved:</b>
<b>CO2:</b> Understand concepts of Object Oriented Programming and basic characteristics of Java.

<b>Tools used:</b>
Notepad, Command Prompt

<b>Theory:</b>
(About Types of Inheritance and abstract class)

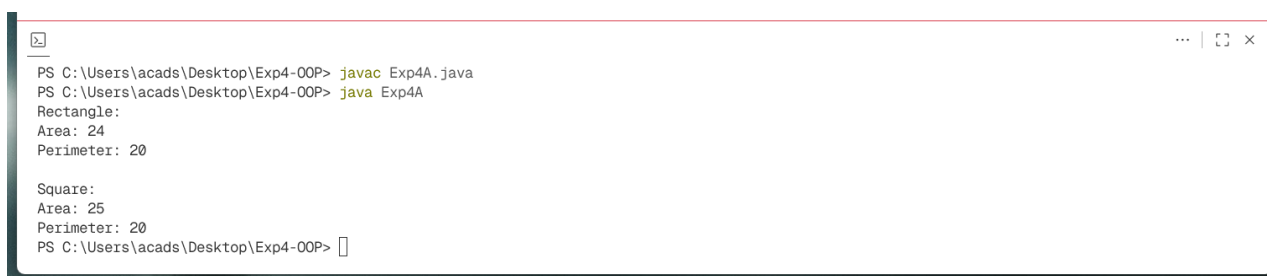
<b>Code:</b>
<ol style="list-style-type: none"> <li>1. <i>dth'</i> and two methods to print the area and perimeter of the rectangle respectively. Its constructor having parameters for length and breadth is Create a class named '<i>Rectangle</i>' with two data members '<i>length</i>' and '<i>brea</i>' used to initialize length and breadth of the rectangle. Let class '<i>Square</i>' inherit the '<i>Rectangle</i>' class with its constructor having a parameter for its side (suppose s) calling the constructor of its parent class as '<i>super(s,s)</i>'. Print the area and perimeter of a rectangle and a square.</li> <li>2. Now repeat the above example to print the area of 10 squares. Hint-Use array of objects</li> <li>3. Write a java program to create an abstract class named Shape that contains two integers and an empty method named print Area (). Provide three classes named Rectangle, Triangle and Circle such that each one of the classes extends the class Shape. Each one of the classes contains only the method print Area () that prints the area of the given shape.</li> </ol>

## Q1.

### Program:

```
class Rectangle
{
    int length, breadth;
    Rectangle(int l, int b)
    {
        length = l;
        breadth = b;
    }
    void area()
    {
        System.out.println("Area: " + (length * breadth));
    }
    void perimeter()
    {
        System.out.println("Perimeter: " + (2 * (length + breadth)));
    }
}
class Square extends Rectangle
{
    Square(int s)
    {
        super(s, s);
    }
}
class Exp4A
{
    public static void main(String[] args)
    {
        System.out.println("Rectangle:");
        Rectangle r = new Rectangle(4, 6);
        r.area();
        r.perimeter();
        System.out.println("\nSquare:");
        Square s = new Square(5);
        s.area();
        s.perimeter();
    }
}
```

### Output:

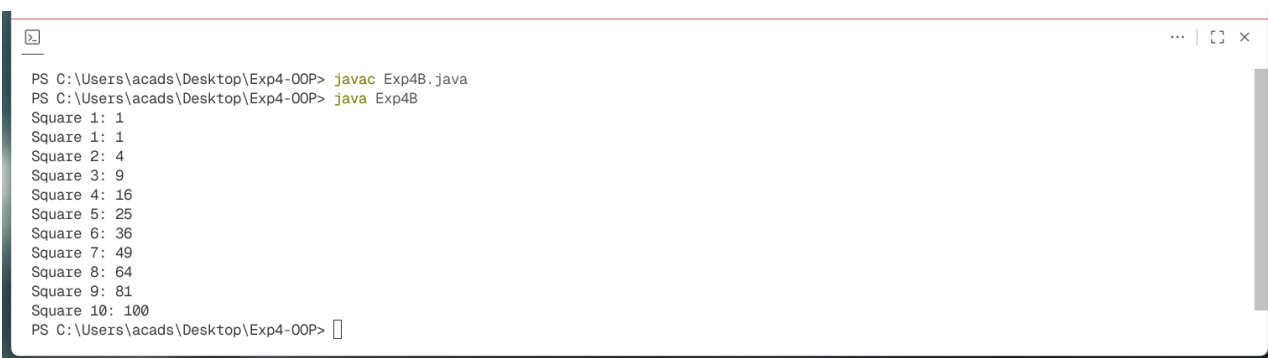


```
PS C:\Users\acads\Desktop\Exp4-00P> javac Exp4A.java
PS C:\Users\acads\Desktop\Exp4-00P> java Exp4A
Rectangle:
Area: 24
Perimeter: 20

Square:
Area: 25
Perimeter: 20
PS C:\Users\acads\Desktop\Exp4-00P>
```

**Q2.****Program:**

```
class Rectangle2
{
    int length, breadth;
    Rectangle2(int l, int b)
    {
        length = l;
        breadth = b;
    }
    void area()
    {
        System.out.println(length * breadth);
    }
}
class Square2 extends Rectangle2
{
    Square2(int s)
    {
        super(s, s);
    }
}
public class Exp4B
{
    public static void main(String[] args)
    {
        Square2[] squares = new Square2[10];
        for (int i = 0; i < 10; i++)
        {
            System.out.print("Square " + (i + 1) + ": ");
            squares[i] = new Square2(i + 1);
            squares[i].area();
        }
    }
}
```

**Output:**

```
PS C:\Users\acads\Desktop\Exp4-00P> javac Exp4B.java
PS C:\Users\acads\Desktop\Exp4-00P> java Exp4B
Square 1: 1
Square 1: 1
Square 2: 4
Square 3: 9
Square 4: 16
Square 5: 25
Square 6: 36
Square 7: 49
Square 8: 64
Square 9: 81
Square 10: 100
PS C:\Users\acads\Desktop\Exp4-00P>
```

### Q3.

#### Program:

```
abstract class Shape
{
    int a, b;
    abstract void printArea();
}
class Rectangle3 extends Shape
{
    Rectangle3(int x, int y)
    {
        a = x;
        b = y;
    }
    void printArea()
    {
        System.out.println("Rectangle Area: " + (a * b));
    }
}
class Triangle extends Shape
{
    Triangle(int x, int y)
    {
        a = x;
        b = y;
    }
    void printArea()
    {
        System.out.println("Triangle Area: " + (0.5 * a * b));
    }
}
class Circle extends Shape
{
    Circle(int x)
    {
        a = x;
    }
    void printArea()
    {
        System.out.println("Circle Area: " + (3.14 * a * a));
    }
}
class Exp4C
{
    public static void main(String[] args)
    {
        Shape r = new Rectangle3(4, 5);
        Shape t = new Triangle(4, 5);
        Shape c = new Circle(3);
        r.printArea();
        t.printArea();
        c.printArea();
    }
}
```

**Output:**

```
PS C:\Users\acads\Desktop\Exp4-OOP> javac Exp4C.java
PS C:\Users\acads\Desktop\Exp4-OOP> java Exp4C
Rectangle Area: 20
Triangle Area: 10.0
Circle Area: 28.259999999999998
PS C:\Users\acads\Desktop\Exp4-OOP> 
```

**Post Lab Subjective/Objective type Questions:**

1. Explain *super* keyword with all its usages. Support explanation with a program.

⇒

1. Call parent constructor → `super(...)`
2. Call parent method → `super.method()`
3. Access parent variable → `super.variable`

Program:

```
class Parent
{
    String name = "Parent";
    void show() { System.out.println("Parent method"); }
}
class Child extends Parent
{
    String name = "Child";
    void display()
    {
        super.show();
        System.out.println(super.name);
    }
}
class Main
{
    public static void main(String[] args)
    {
        Child c = new Child();
        c.display();
    }
}
```

2. Perform following virtual lab experiments.

- a. <https://java-iitd.vlabs.ac.in/exp/inheritance/>
- b. <https://java-iitd.vlabs.ac.in/exp/method-overloading/>
- c. <https://java-iitd.vlabs.ac.in/exp/method-overriding/>

**Conclusion:**

The experiment demonstrated the concepts of inheritance, method overriding, and abstract classes in Java. It showed how child classes can reuse and extend parent class functionality, how the super keyword is used to access parent class members, and how abstract classes define a common structure for different subclasses to implement specific behavior.

**Signature of faculty in-charge with Date:**