

Java

1. Implement Abstract class with overloading and overriding?

Answer:-

```
package Daily;
public abstract class Shape {

    // Abstract method
    abstract void draw();
    public static void main(String[] args) {
        Shape shape1 = new Circle();
        Shape shape2 = new Rectangle();

        shape1.draw(); // Overridden method
        shape1.calculateArea(5); // Overloaded method
        shape2.draw(); // Overridden method
        shape2.calculateArea(5, 10); // Overloaded method
    }
    // Overloaded method (different parameters)
    void calculateArea(int radius) {
        System.out.println("Area of circle: " + (3.14 * radius * radius));
    }
    void calculateArea(int length, int breadth) {
        System.out.println("Area of rectangle: " + (length * breadth));
    }
}
class Circle extends Shape {
    // Overriding abstract method
    @Override
    void draw() {
        System.out.println("Drawing Circle");
    }

    // Overriding calculateArea (Optional)
    @Override
    void calculateArea(int radius) {
        System.out.println("Circle's specific area calculation: " + (3.14 * radius * radius));
    }
}
class Rectangle extends Shape {
    // Overriding abstract method
    @Override
    void draw() {
        System.out.println("Drawing Rectangle");
    }
}
```

```
}
}
```

Output:-

```

1 package Daily;
2
3 abstract class Shape {
4     // Abstract method
5     abstract void draw();
6     public static void main(String [] args) {
7         Shape shape1 = new Circle();
8         Shape shape2 = new Rectangle();
9
10        shape1.draw(); // Overridden method
11        shape1.calculateArea(5); // Overloaded method
12
13        shape2.draw(); // Overridden method
14        shape2.calculateArea(5, 10); // Overloaded method
15    }
16
17
18
19    // Overloaded method (different parameters)
20    void calculateArea(int radius) {
21        System.out.println("Area of circle: " + (3.14 * radius * radius));
22    }
23
24    void calculateArea(int length, int breadth) {
25        System.out.println("Area of rectangle: " + (length * breadth));
26    }
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

```

<terminated> Shape [Java Application] C:\Users\tapan.k\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_22.0.1.v20240426-1149\jre\bin\javaw.exe (1
Drawing Circle
Circle's specific area calculation: 78.5
Drawing Rectangle
Area of rectangle: 50

```

2. Implement Multiple inheritance with Interface?

Answer:-

```

package Daily;
interface Printable {
    void print();
}
interface Showable {
    void show();
}
class Document implements Printable, Showable {
    // Implementing both interface methods
    @Override
    public void print() {
        System.out.println("Printing Document");
    }
    @Override
    public void show() {
        System.out.println("Showing Document");
    }
}

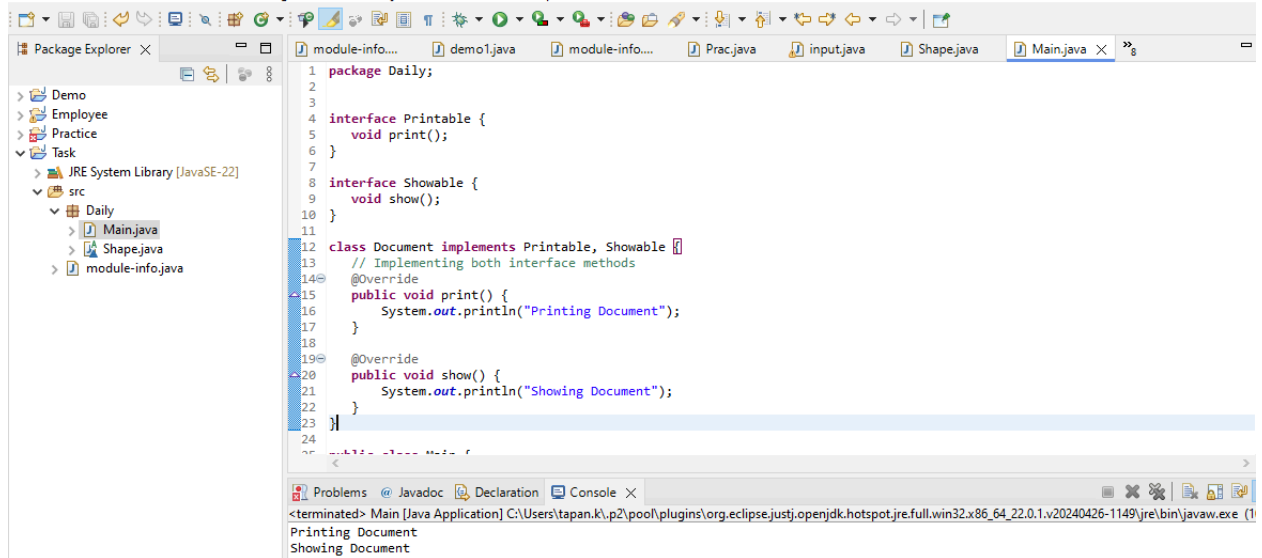
```

```

public class Main {
    public static void main(String[] args) {
        Document doc = new Document();
        doc.print();
        doc.show();
    }
}

```

Output:-



3. Show final methods in the class that can't be overridden?

Answer:-

```

package Daily;
class Base {
    // Final method
    public final void display() {
        System.out.println("Display from Base class");
    }

    public void show() {
        System.out.println("Show from Base class");
    }
}

class Derived extends Base {
    // Trying to override display() would cause a compilation error
    // @Override

```

```

// public void display() {
//     System.out.println("Display from Derived class"); // Not allowed
// }
@Override
public void show() {
    System.out.println("Show from Derived class");
}
}
}
public class Mains {
    public static void main(String[] args) {
        Base base = new Base();
        base.display(); // Final method
        base.show();    // Non-final method
        Derived derived = new Derived();
        derived.display(); // Final method from Base class
        derived.show();   // Overridden method from Derived class
    }
}

```

Output:-

The screenshot shows the Eclipse IDE with the following components:

- Package Explorer (Left):** Shows a project named 'Demo' with a package 'Task' containing a sub-package 'Daily'. The 'Daily' package contains files: 'Main.java', 'Mains.java' (selected), 'Shape.java', and 'module-info.java'.
- Editor (Center):** Displays the code for 'Mains.java'. The code is as follows:


```

1 package Daily;
2
3 class Base {
4     // Final method
5     public final void display() {
6         System.out.println("Display from Base class");
7     }
8
9     public void show() {
10        System.out.println("Show from Base class");
11    }
12 }
13
14 class Derived extends Base {
15     // Trying to override display() would cause a compilation error
16     // @Override
17     // public void display() {
18     //     System.out.println("Display from Derived class"); // Not allowed
19     // }
20
21     @Override
22     public void show() {
23         System.out.println("Show from Derived class");
24     }
25 }

```
- Console (Bottom):** Shows the output of the program:


```

<terminated> Mains [Java Application] C:\Users\tapan.k.p2\poo\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_22.0.1.v20240426-1149\jre\bin\javaw.exe (11
Display from Base class
Show from Base class
Display from Base class
Show from Derived class

```