

Shape and Overriding

```
import java.util.*;

import java.io.*;

// Create the classes here

class Shape{
    String shapeType;

    public void printMyType(String shapeType)
    {
        System.out.println(shapeType);
    }
}

class Square extends Shape{
    int length;
    int breadth;

    public int calculateArea(int length, int breadth)
    {
        int area = length*breadth;
        return area;
    }

    public void printMyType(String shapeType, int length, int breadth)
    {
        this.length = length;
        this.breadth = breadth;
        System.out.println(shapeType);
        System.out.println(calculateArea(this.length, this.breadth));
    }
}

class Rectangle extends Shape
{
    int length;
    int breadth;

    public int calculateArea(int length, int breadth)
    {

```

```
        int area = length*breadth;

        return area;

    }

    public void printMyType(String shapeType, int length, int breadth)
    {

        this.length = length;
        this.breadth = breadth;

        System.out.println(shapeType);
        System.out.println(calculateArea(this.length, this.breadth));

    }

}

class Solution {

    public static void main(String args[]) {

        // Write your code here

        Square obj1 = new Square();
        obj1.printMyType("square", 5, 5);
        Rectangle obj2 = new Rectangle();
        obj2.printMyType("rectangle", 5, 4);

    }

}
```

How many types of polymorphism in java?

Options:

- ☐ 1
- ☒ 2
- ☐ 4
- ☐ 5

✓ Correct Answer

Is method overloading possible by changing the return type?

Options:

- ☐ Yes
- ☒ No

✓ Correct Answer

Method overriding can occur through only?

Options:

- ☐ Abstract class
- ☐ Interface
- ☒ Inheritance
- ☐ None of the above

✓ Correct Answer


What is the output of the following code?

```
class Test
{
    public void fun()
    {
        System.out.println("Coding Ninjas");
    }
}

public class Derived extends Test
{
    public void fun()
    {
        System.out.println("Ninjas");
    }
    public static void main(String[] args)
    {
        Derived obj = new Test();
        obj.fun();
    }
}
```

Options:

- ☐ Ninjas
- ☒ Compile time error
- ☐ Coding Ninjas
- ☐ Runtime error

 Correct Answer