Java Method OverLoading and Method OverRiding :

Certainly! In Java, method overloading and method overriding are two important concepts related to polymorphism and inheritance. Let's start with method overloading.

Method Overloading:

Method overloading allows you to define multiple methods in the same class with the same name but with different parameter lists (number or type of parameters). Java determines which method to call based on the arguments provided during the method invocation.

Example 1: Method Overloading

```java
class Calculator {
    // Method to add two integers
    public int add(int a, int b) {
        return a + b;
    }

    // Method to add three integers (overloaded version)
    public int add(int a, int b, int c) {
        return a + b + c;
    }

    // Method to add two doubles (overloaded version)
    public double add(double a, double b) {
        return a + b;
    }
}

public class Main {
    public static void main(String[] args) {
        Calculator calculator = new Calculator();

        System.out.println("Sum of two integers: " + calculator.add(5, 10));
        System.out.println("Sum of three integers: " + calculator.add(5, 10, 15));
        System.out.println("Sum of two doubles: " + calculator.add(3.5, 2.7));
    }
}
```

In this example, we have a `Calculator` class with three overloaded `add` methods, each taking a different number and type of parameters.

Method Overriding:

Method overriding is used in inheritance when a subclass provides a specific implementation for a method that is already defined in its superclass. The overriding method in the subclass should have the same method signature (name, return type, and parameters) as the method in the superclass.

Example 2: Method Overriding

```java
class Shape {
    public void draw() {
        System.out.println("Drawing a shape");
    }
}

class Circle extends Shape {
    @Override
    public void draw() {
        System.out.println("Drawing a circle");
    }
}

class Square extends Shape {
    @Override
    public void draw() {
        System.out.println("Drawing a square");
    }
}

public class Main {
    public static void main(String[] args) {
        Shape shape1 = new Circle();
        Shape shape2 = new Square();

        shape1.draw();  // Calls the overridden method in Circle
        shape2.draw();  // Calls the overridden method in Square
    }
}
```

In this example, we have a `Shape` class with a `draw` method, and two subclasses `Circle` and `Square` that override the `draw` method to provide their own implementations. When we create objects of the subclasses and call the `draw` method, the overridden version of the method in the respective subclass is executed.