**Assisted Practice: 2.1 Methods**

* + 1. Writing a program in Java to verify implementations of methods and ways of calling a method
    2. Executing the program and verifying working of methods
    3. Pushing the code to your GitHub repositories

**Step 2.1.1:** Writing a program in Java to verify implementations of methods and ways of calling a method

public class methodExecution {

public int multipynumbers(int a,int b) {

int z=a\*b;

return z;

}

public static void main(String[] args) {

methodExecution b=new methodExecution();

int ans= b.multipynumbers(10,3);

System.*out*.println("Multipilcation is :"+ans);

}

**//Call by value**

public class callMethod

{

int val=50;

int operation(int val)

{

val =val\*20/100;

return(val);

}

public static void main(String args[])

{

callMethod d = new callMethod();

System.*out*.println("Before operation value of data is "+d.val);

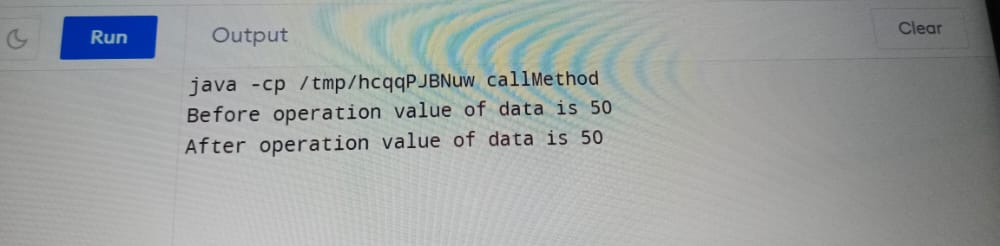
d.operation(100);

System.*out*.println("After operation value of data is "+d.val);

}

}

**Output:**

****

**//Method overloading**

// Java program to demonstrate working of method

// overloading in Java.

public class Sum {

// Overloaded sum(). This sum takes two int parameters

public int sum(int x, int y)

{

return (x + y);

}

// Overloaded sum(). This sum takes three int parameters

public int sum(int x, int y, int z)

{

return (x + y + z);

}

// Overloaded sum(). This sum takes two double parameters

public double sum(double x, double y)

{

return (x + y);

}

// Driver code

public static void main(String args[])

{

Sum s = new Sum();

System.out.println(s.sum(10, 20));

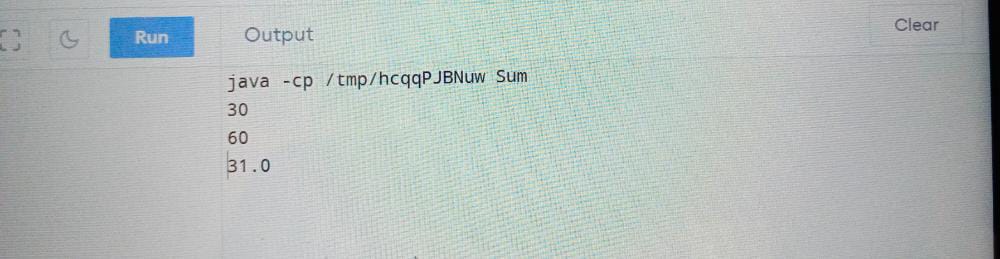
System.out.println(s.sum(10, 20, 30));

System.out.println(s.sum(10.5, 20.5));

}

}

**Output:**

****