**Assignment: (Core Java)**

**5 : Methods in Java**

**Que.1 Defining Methods**

**Ans.1**  A method is a block of code that performs a specific task. Methods help make programs modular and easy to maintain.

**Syntax**

returnType methodName(parameters) {

// method body

}

**Que.2 Method Parameters and Return Types**

**Ans.2** Parameters are inputs passed to a method when it is called. They allow a method to perform tasks using different data. A method can have zero, one, or multiple parameters.

class Calculator {

void add(int a, int b) {

int sum = a + b;

System.out.println("Sum = " + sum);

}

}

public class methodPara {

public static void main(String[] args) {

Calculator calc = new Calculator();

calc.add(10, 20); // Pass value to method

calc.add(5, 15);

}

}

**Return Types:** Return type specifies what type of value the method will return. If a method does not return any value, use void. Otherwise, use a data type like int, double, String, etc.

**Example:**

class Calculator {

int multi(int a, int b) {

return a \* b; // return result

}

}

public class methodReturn {

public static void main(String[] args) {

Calculator calc = new Calculator();

int product = calc.multi(5, 6); // store return value

System.out.println("Product = " + product);

}

**Que.3 Method Overloading**

**Ans.3** Method Overloading means having multiple methods in the same class with the same name but different parameters. The compiler decides which method to call based on the number or type of arguments. Overloading cannot be done by changing only the return type; the parameters must differ.

class Calculator {

// Method 1: two parameters

int add(int a, int b) {

return a + b;

}

// Method 2: three parameters

int add(int a, int b, int c) {

return a + b + c;

}

}

public class methodOverloading {

public static void main(String[] args) {

Calculator calc = new Calculator();

System.out.println("Sum of 2 numbers: " + calc.add(10, 20));

System.out.println("Sum of 3 numbers: " + calc.add(10, 20, 30));

}

}

**Que.4 Static Methods and Variables**

**Ans.4** static members belong to the class, not to any object. You can access static members without creating an object of the class. Both variables and methods can be static.

**Static Variables:** Static variables are shared among all objects of a class. Changes made to a static variable by one object affect all objects.

**Static Methods:** Static methods can be called without creating an object. They can access only static variables directly. They cannot use non-static variables unless an object reference is used.