| Experiment No. 5 |
| --- |
| Method Overloading. |
| Date of Performance:02/08/24 |
| Date of Submission:09/08/24 |

**Aim :-**Implementation of method overloading in object-oriented programming

**Objective :-** Implement a program to calculate the area and volume of different geometric figures using method overloading. The program should be able to calculate the area and volume of a circle, semicircle, rectangle, triangle and sphere etc.

**Theory :-**

Method Overloading is a feature in Java that allows a class to have more than one method with the same name, provided that their parameter lists are different. Method overloading is a form of polymorphism and enhances the readability and usability of code.

**Key Concepts**

Method Signature:

The method signature in Java consists of the method name and the parameter list. The return type and exceptions thrown by the method are not considered part of the method signature.

Parameter List:

To overload methods, the parameter list must differ in terms of:

Number of parameters

Type of parameters

Order of parameters (if they are of different types)

Return Type: The return type of the method does not contribute to method overloading. Two methods with the same name and parameter list but different return types will cause a compilation error.

Access Modifiers: Overloaded methods can have different access modifiers (public, private, protected, or default).

Static Methods: Method overloading can also be applied to static methods.

Syntax of Method Overloading

class ClassName {

// Method with one parameter

returnType methodName(dataType1 param1) {

// Method body

}

// Overloaded method with different number of parameters

returnType methodName(dataType1 param1, dataType2 param2) {

// Method body

}

// Overloaded method with different type of parameters

returnType methodName(dataType1 param1, dataType3 param3) {

// Method body

}

// Overloaded method with different order of parameters

returnType methodName(dataType3 param3, dataType1 param1) {

// Method body

}

}

**Code :-**

class Methodoverloading {

public void sum(int x, int y) {

System.out.println("Sum of integers: " + (x + y));

}

public void sum(double x, double y) {

System.out.println("Sum of doubles: " + (x + y));

}

public static void main(String[] args) {

Methodoverloading mo = new Methodoverloading();

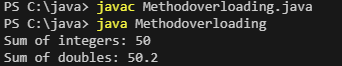
mo.sum(20, 30);

mo.sum(20.1, 30.1);

}

}

**Output:-**

****

**Conclusion :-** Method overloading in Java improves code clarity and maintainability by enabling methods with the same name to perform different tasks based on parameter types or counts. This avoids method name conflicts and simplifies method usage, enhancing code organization and flexibility.