Lab 23

Classroom Activity: Java Methods, Constructors, Recursion

Let's explore different types of methods, constructors, and recursion in Java with complete code examples.

1. Types of Methods:

In Java, there are several types of methods, including instance methods, static methods, and methods with or without parameters. Here are examples of each:

Instance Method:

Instance methods are associated with objects of a class. They can access instance variables and other instance methods.

public class MyClass {

int x;

// Instance method

void setX(int value) {

x = value;

}

// Another instance method

int getX() {

return x;

}

public static void main(String[] args) {

MyClass obj = new MyClass();

obj.setX(42);

System.out.println("Value of x: " + obj.getX()); // Output: Value of x: 42

}

}

Static Method:

Static methods belong to the class itself, rather than instances of the class. They can only access static members of the class.

public class MathUtils {

// Static method

static int add(int a, int b) {

return a + b;

}

public static void main(String[] args) {

int sum = MathUtils.add(5, 3);

System.out.println("Sum: " + sum); // Output: Sum: 8

}

}

2. Constructors:

Constructors are special methods used to initialize objects when they are created. Constructors have the same name as the class and can be overloaded (i.e., multiple constructors with different parameters).

Default Constructor:

A default constructor is automatically provided by Java if you don't define any constructors in your class.

public class MyClass {

int x;

public static void main(String[] args) {

MyClass obj = new MyClass();

System.out.println("Default value of x: " + obj.x); // Output: Default value of x: 0

}

}

Parameterized Constructor:

You can define custom constructors with parameters to initialize object properties.

public class Person {

String name;

int age;

// Parameterized constructor

public Person(String n, int a) {

name = n;

age = a;

}

public static void main(String[] args) {

Person person = new Person("Alice", 30);

System.out.println("Name: " + person.name + ", Age: " + person.age);

// Output: Name: Alice, Age: 30

}

}

3. Recursion:

Recursion is a technique in which a method calls itself to solve a problem. It's often used in scenarios like tree traversal and factorial calculation.

Factorial Calculation using Recursion:

public class FactorialCalculator {

// Recursive method to calculate factorial

static int factorial(int n) {

if (n == 0 || n == 1) {

return 1;

} else {

return n \* factorial(n - 1);

}

}

public static void main(String[] args) {

int num = 5;

int result = factorial(num);

System.out.println("Factorial of " + num + " is " + result); // Output: Factorial of 5 is 120

}

}

In this example, the `factorial` method is defined recursively to calculate the factorial of a number. The base case (when `n` is 0 or 1) terminates the recursion, and the method gradually computes the factorial by calling itself with a smaller value of `n`.