

Lab 02

# Course Code: CSE-124

# Course Title: OOP Lab

Submitted To: Golam Daiyan

***Submitted By:***Name: Abdur Rahim

ID: **231001422E**

Department: ECSE

Semester: 3DCSE (EVENING)  
**Submitted Date:** 17/11/2023

**Assignment Name:** Java MethodsJava MethodsJava Method ParametersJava Method OverloadingJava ScopeJava Recursion Java ClassesJava OOPJava Classes/ObjectsJava Class AttributesJava Class MethodsJava ConstructorsJava ModifiersJava EncapsulationJava Packages / APIJava InheritanceJava PolymorphismJava Inner Classes

**Java Methods:**

Definition:

A method in Java is a set of instructions or a block of code that performs a specific task. Methods are used to organize code into reusable units.

Syntax:

returnType methodName(parameterType parameterName) {

// method body

// return statement if the method has a return type

}

**Java Method Parameters:**

Definition:

Method parameters are variables used in a method to receive values from the caller. They allow methods to accept input, making them more flexible and reusable.

Syntax:

void methodName(int parameter1, String parameter2) {

// method body

}

**Java Method Overloading:**

Definition:

Method overloading allows a class to have multiple methods with the same name but different parameter lists. It improves code readability and reusability.

Example:

int add(int a, int b) {

return a + b;

}

double add(double a, double b) {

return a + b;

}

**Java Scope:**

Definition:

Scope refers to the region of code where a variable can be accessed. In Java, variables can have local scope (limited to a block), class scope (accessible throughout the class), or global scope (accessible across classes).

**Java Recursion:**

Definition:

Recursion is a programming technique where a method calls itself to solve a problem. It involves breaking down a problem into smaller subproblems and solving them recursively.

**Java Classes:**

Definition:

A class in Java is a blueprint for creating objects. It encapsulates data (attributes) and behavior (methods) that define the object's characteristics.

Syntax:

public class ClassName {

// class body

}

**Java OOP (Object-Oriented Programming):**

Definition:

Object-Oriented Programming is a programming paradigm that uses objects (instances of classes) to model and solve real-world problems. Key principles include encapsulation, inheritance, and polymorphism.

**Java Class Attributes:**

Definition:

Class attributes are variables declared within a class. They represent the state or characteristics of objects created from the class.

**Java Class Methods:**

Definition:

Class methods are functions defined within a class. They operate on class-level data and are invoked using the class name rather than an instance of the class.

**Java Constructors:**

Definition:

Constructors are special methods used for initializing objects. They have the same name as the class and are called when an object is created.

**Java Modifiers:**

Definition:

Modifiers in Java are keywords that provide additional information about classes, methods, and variables. Examples include public, private, static, final, etc.

**Java Encapsulation:**

Definition:

Encapsulation is the bundling of data (attributes) and methods that operate on that data within a single unit (class). It helps in hiding the internal implementation details.

**Java Packages / API:**

Definition:

Packages are used to organize classes into namespaces. The Java API (Application Programming Interface) is a collection of classes and methods that provide pre-built functionality for common tasks.

**Java Inheritance:**

Definition:

Inheritance is a mechanism where a class (subclass/derived class) inherits properties and behaviors from another class (superclass/base class). It promotes code reuse.

**Java Polymorphism:**

Definition:

Polymorphism allows a single entity (method or object) to take on multiple forms. In Java, it can be achieved through method overloading and method overriding.

**Java Inner Classes:**

Definition:

Inner classes are classes defined inside another class. They have access to the enclosing class's members and can be used for encapsulation and organization.