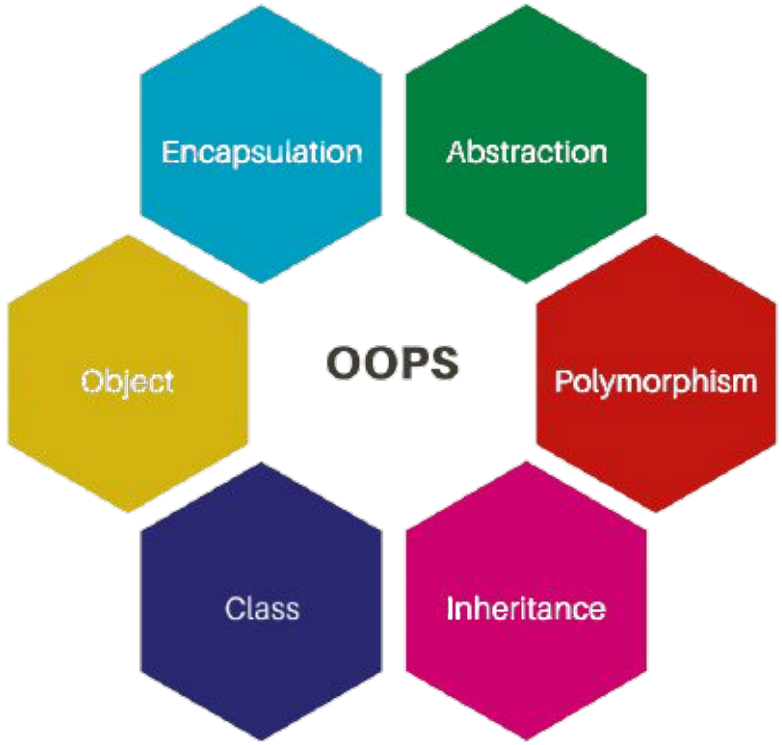
ACHARYA

Dept of AIML

Module 1

Title: Object Oriented Programming with Java

Subtitle: Object-Oriented Concepts, Data Types, Operators, and Control Statements



ACHARYA

Dept of AIML

Overview of Java: Object-Oriented Programming (OOP)

Java is a versatile programming language that follows the Object-Oriented Programming (OOP) paradigm. OOP in Java revolves around the concept of objects and classes, which organize code in a way that promotes reusability, scalability, and efficiency.

Two Paradigms:

1. **Procedural Programming:** Traditional programming where code is organized in a step-by-step manner.
2. **Object-Oriented Programming:** Focuses on objects and their interactions, allowing for more modular and maintainable code.

Dept of AIML

Abstraction:

Abstraction is the process of hiding complex implementation details and showing only the essential features of an object. It allows developers to work with higher-level ideas without needing to understand all the complexities underneath.

ACHARYA

Dept of AIML

The Three OOP Principles:

1. **Encapsulation:** Bundling data (variables) and methods (functions) that operate on the data into a single unit called a class. Encapsulation restricts direct access to some of the object's components, which is a means of preventing unintended interference.
2. **Inheritance:** A mechanism where one class inherits the properties and behaviors (methods) of another class. This promotes code reuse and establishes a natural hierarchy between classes.
3. **Polymorphism:** The ability of different classes to be treated as instances of the same class through a common interface. It allows one interface to be used for a general class of actions.

Lexical Issues:

Dept of AIML

1. **Whitespace:** Java ignores extra spaces, tabs, and newline characters. However, proper use of whitespace improves code readability.
2. **Identifiers:** Names used to identify variables, methods, classes, etc. They must start with a letter, dollar sign ($), or underscore (_) and can be followed by digits.
3. **Literals:** Constants assigned to variables, such as 100, 3.14, 'A', or "Hello".
4. **Comments:** Used for code documentation.
    - Single-line comments: // comment
    - Multi-line comments: /* comment */
    - Documentation comments: /** comment */
5. **Separators:** Characters like (), {}, [], ;, ,, and . are used to define code structure and separate expressions.
6. **The Java Keywords:** Reserved words that have predefined meanings in Java, like class, public, static, void, etc.

ACHARYA

Dept of AIML

Data Types, Variables, and Arrays:

Primitive Types:

1. **Integers:** byte, short, int, long—for whole numbers.
2. **Floating-Point Types:** float, double—for numbers with decimal points.
3. **Characters:** char—for single 16-bit Unicode characters.
4. **Booleans:** boolean—for true/false values.

Variables:

Containers for storing data values. In Java, variables must be declared before use.

Type Conversion and Casting:

- **Type Conversion:** Automatically converting a smaller type to a larger type size (e.g., int to double).
- **Casting:** Explicitly converting a larger type to a smaller type (e.g., double to int).

Automatic Type Promotion in Expressions:

In Java, smaller data types are automatically promoted to a larger data type when they are used in expressions.

Arrays:

A container that holds a fixed number of values of a single type. Arrays are declared with a specific type and size.

Type Inference with Local Variables:

Java 10 introduced the var keyword, allowing the compiler to infer the type of local variables based on the assigned value.

Dept of AIML

Operators:

Arithmetic Operators:

Used for basic mathematical operations (+, -, *, /, %).

Relational Operators:

Used to compare two values (==, !=, >, <, >=, <=).

Boolean Logical Operators:

Used to perform logical operations (&&, ||, !).

The Assignment Operator:

Used to assign values to variables (=).

The Ternary (? :) Operator:

A shorthand for the if-else statement. It evaluates a boolean expression and returns one of two values.

ACHARYA

Dept of AIML

Control Statements:

Java's Selection Statements:

- **if:** Executes a block of code if a condition is true.
- **switch:** Evaluates an expression and executes code based on matching case labels.

Iteration Statements:

- **while:** Repeats a block of code as long as a condition is true.
- **do-while:** Similar to while, but guarantees that the block of code is executed at least once.
- **for:** Loops through a block of code a specific number of times.
- **For-Each Version of the for Loop:** Simplifies iteration over arrays or collections.
- **Local Variable Type Inference in a for Loop:** Allows the use of var in loop declarations.

Dept of AIML

Nested Loops:

Loops within loops, enabling complex iterations.

Jump Statements:

- **break:** Exits the current loop or switch statement.
- **continue:** Skips the current iteration of a loop and moves to the next iteration.
- **return:** Exits from the current method and optionally returns a value.

Dept of AIML

Simple Java Program: "Hello, World!“

```
public class HelloWorld
{
        public static void main(String[] args)
        {
                System.out.println("Hello, World!");
        }
}
```

Dept of AIML

Explanation:

1.public class HelloWorld {

- **public**: An access modifier that allows the class to be accessible from anywhere.
- **class**: This keyword is used to declare a class in Java. A class is a blueprint for objects, and in this case, it's used to define a new class called HelloWorld.
- **HelloWorld**: The name of the class. The class name should match the filename (HelloWorld.java).

2.public static void main(String[] args) {

- **public**: This keyword makes the main method accessible from anywhere.
- **static**: This means that the main method belongs to the class, not to any specific object, so it can be called without creating an instance of the class.
- **void**: The return type of the method. void means that this method does not return any value.
- **main**: The name of the method. main is the entry point of any Java application. When you run the program, the main method is executed.
- **String[] args**: This parameter is an array of String objects. It allows you to pass command-line arguments to the program.

3.System.out.println("Hello, World!");

- **System**: A built-in class in Java that provides access to system resources.
- **out**: A static member of the System class, which is an instance of PrintStream. It provides methods to print output to the console.
- **println**: A method of PrintStream that prints the argument passed to it (in this case, "Hello, World!") to the console, followed by a newline.
- **"Hello, World!"**: A string literal that is the argument passed to println. This is what gets printed to the console.