1: EVOLUTION OF JAVA & BASICS OF JAVA PROGRAMMING

Title: Evolution of Java and Overview of Java Language

Introduction: Java is a high-level, object-oriented programming language developed to overcome the limitations of C and C++. It is platform-independent, secure, and widely used for web, mobile, and enterprise applications.

Learning Objectives: After completing this module, learners will be able to:

- Understand the evolution of Java
- Explain how Java differs from C and C++
- Write and execute a simple Java program
- Understand Java program structure and JVM

Main Content

Java Evolution

- Developed by Sun Microsystems
- Designed for portability and security
- “Write Once, Run Anywhere” concept

Java vs C/C++

- Platform independent
- Automatic memory management
- No pointers
- Strong security features

Java and Internet / WWW

- Used in web applications
- Supports networking and distributed systems

Java Program Structure

- Class definition
- main() method
- Statements and tokens

Java Virtual Machine (JVM)

- Converts bytecode into machine code
- Enables platform independence

Command Line Arguments

- Inputs passed during program execution

Example

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

Activity: Write a Java program to print your name.

Assessment

1. Why is the Java platform independent?
2. What is the role of the JVM?

Summary: Java provides portability, security, and object-oriented features for modern software development.

2: DATA TYPES, VARIABLES, OPERATORS & EXPRESSIONS

Title: Constants, Variables, Data Types and Operators in Java

Introduction: Data handling is essential in programming. Java provides various data types, operators, and expressions to perform computations efficiently.

Learning Objectives: Learners will,

- Understand Java data types
- Declare and initialise variables
- Use operators and expressions
- Perform type casting

Main Content

Constants and Variables

- Constants: fixed values
- Variables: store data

Data Types

- Primitive: int, float, double, char, boolean
- Default values of variables

Variable Scope

- Local, instance, static variables

Type Casting

- Implicit and explicit conversion

Operators:

Arithmetic	Relational	Logical	Assignment
Conditional (?:)	Bitwise	Special operators	Increment / Decrement

Expressions

- Operator precedence
- Associativity
- Mathematical functions

Example

```
int a = 10;
```

```
int b = 20;
int sum = a + b;
```

Activity: Identify operators used in a given expression.

Assessment: What is type casting?

Summary: Operators and expressions allow efficient data manipulation and decision-making in Java.

3: DECISION MAKING & LOOPING STATEMENTS

Title: Decision Making and Loop Control in Java

Introduction: Decision making and looping statements help control the flow of a program based on conditions and repetition.

Learning Objectives: Students will be able to,

- Use conditional statements
- Apply looping constructs
- Understand jump statements

Main Content

Decision-Making Statements

- if
- if-else
- nested if-else
- else-if ladder
- switch statement
- conditional (?:) operator

Looping Statements

- while loop
- do-while loop
- for loop

Jump Statements

- break
- continue
- labeled loops

Example

```
for(int i=1; i<=5; i++) {
    System.out.println(i);
}
```

Activity: Write a program to print even numbers using a loop.

Assessment: Difference between while and do-while loop.

Summary: Control structures help manage logic and repetition effectively in Java programs.

4: CLASSES, OBJECTS, INHERITANCE & INTERFACES

Title: Object-Oriented Programming Concepts in Java

Introduction: Java follows Object Oriented Programming (OOP) principles that help in building modular, reusable, and maintainable programs.

Learning Objectives: Learners will,

- Create classes and objects
- Use constructors and method overloading
- Apply inheritance and interfaces
- Understand visibility control

Main Content

Classes and Objects

- Defining a class
- Creating objects
- Accessing members

Constructors

- Default and parameterised
- Method overloading

Static Members

- Shared across objects

Inheritance

- Extending a class
- Method overriding
- final variables, methods, classes
- Abstract methods and classes

Interfaces

- Multiple inheritance using interfaces
- Defining and implementing interfaces

Visibility Control

- public, private, protected, default

Example

```
class A {
    void show() {
        System.out.println("Class A");
    }
}
```

Activity: Create a class and demonstrate inheritance.

Assessment: What is the use of interfaces?

Summary: OOP concepts improve code reusability, flexibility, and clarity.

5: ARRAYS, PACKAGES, MULTITHREADING & EXCEPTIONS

Title: Advanced Java Programming Concepts

Introduction: Advanced Java features help in handling complex programs involving data structures, concurrency, and error handling.

Learning Objectives: Students will,

- Work with arrays and strings
- Create and use packages
- Understand multithreading
- Handle exceptions

Main Content

Arrays and Strings

- One-dimensional and two-dimensional arrays
- String class
- Vectors
- Wrapper classes

Packages

- Java API packages
- Creating and accessing packages
- Naming conventions
- Hiding classes

Multithreading

- Creating threads
- Thread lifecycle
- Thread priority
- Synchronization

Exception Handling

- Types of errors
- try, catch, finally
- Multiple catch blocks
- User-defined exceptions
- Debugging using exceptions

Example

```
try {
    int a = 10/0;
```

```
} catch(Exception e) {
    System.out.println("Error");
}
```

Activity: Create a simple program using try-catch.

Assessment: Why is synchronisation required?

Summary: Advanced features improve the performance, reliability, and robustness of Java programs.

6: APPLET & FILE HANDLING IN JAVA

Title: Applet Programming and File Input/Output in Java

Introduction: Java supports graphical and file-based applications through applets and I/O streams.

Learning Objectives: Learners will,

- Understand applet programming
- Handle files using Java I/O
- Use stream classes

Main Content

Applet Programming

- Difference between an applet and an application
- Applet life cycle
- Applet tag
- Passing parameters
- Running applets

File Handling

- Concept of streams
- Byte streams
- Character streams
- File class

- Creating and managing files
- Input/output exceptions

Example

```
File file = new File("data.txt");
```

Activity: Write steps to create and run an applet.

Assessment: Difference between byte and character streams.

Summary: Applet and file handling features support interactive and persistent Java applications.