Unit 1: Java Fundamentals, Data Types, Operators and Control Statements

1.1 History and Philosophy of Java

Definition

Java is a **high-level, object-oriented programming language** developed by **James Gosling in 1995** at Sun Microsystems.

Features / Philosophy

- **Platform independent** (Write Once, Run Anywhere)
- **Secure and robust**
- **Object oriented**
- **Simple and portable**

Java programs run using **JVM (Java Virtual Machine)**.

1.2 Object Oriented Programming (OOP)

Definition

Object Oriented Programming is a programming approach based on **objects and classes**.

Main Concepts

- Class
- Object
- Inheritance
- Polymorphism
- Encapsulation
- Abstraction

Example Program

```
class Student {
    int id;
```

```
    String name;

    void display() {
        System.out.println(id + " " + name);
    }
}

public class Main {
    public static void main(String[] args) {
        Student s = new Student();
        s.id = 1;
        s.name = "Amit";
        s.display();
    }
}
```

Output

```
1 Amit
```

1.3 Java Development Kit (JDK)

Definition

JDK is a **software package** used to develop Java programs.

Components

- JDK = JRE + Development Tools
- JRE = JVM + Libraries

Uses

- Compile Java programs
- Run Java programs

1.4 A First Simple Java Program

Definition

This program prints a message on the screen.

Program

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

Output

```
Hello Java
```

1.5 Packages in Java

Definition

A package is a **group of related classes and interfaces**.

Advantages

- Code organization
- Reusability
- Security

Syntax

```
package packagename;
```

Example Program

```
package mypack;

public class Demo {
    public static void main(String[] args) {
        System.out.println("My Package Program");
    }
}
```

Output

```
My Package Program
```

1.6 Java's Data Types

Definition

Data types specify the **type of data a variable can store**.

Java has:

- Primitive data types
- Non-primitive data types

1.6.1 Integers

Definition

Used to store **whole numbers**.

Types

- byte
- short
- int
- long

Example

```
class IntegerDemo {
    public static void main(String[] args) {
        int a = 10;
        System.out.println(a);
    }
}
```

Output

```
10
```

1.6.2 Characters

Definition

Used to store **single characters**.

Syntax

```
char ch = 'A';
```

Example

```
class CharDemo {
    public static void main(String[] args) {
        char ch = 'J';
        System.out.println(ch);
    }
}
```

Output

```
J
```

1.6.3 Floating Point Types

Definition

Used to store **decimal values**.

Types

- float
- double

Example

```
class FloatDemo {
    public static void main(String[] args) {
        double x = 10.5;
        System.out.println(x);
    }
}
```

Output

```
10.5
```

1.6.4 Strings

Definition

String is a **sequence of characters**.

Example

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

Output

```
Java
```

1.6.5 Arrays

Definition

Array stores **multiple values of same data type**.

Syntax

```
int a[] = {1,2,3};
```

Example

```
class ArrayDemo {
    public static void main(String[] args) {
        int a[] = {10, 20, 30};
        System.out.println(a[1]);
    }
}
```

Output

```
20
```

1.6.6 The Boolean Types

Definition

Boolean stores **true or false** values.

Example

```
class BooleanDemo {
    public static void main(String[] args) {
        boolean flag = true;
        System.out.println(flag);
    }
}
```

Output

```
true
```

1.7 Literals

Definition

Literals are **fixed constant values** used in programs.

1.7.1 Hex, Octal and Binary

```
class LiteralDemo {
    public static void main(String[] args) {
        int a = 0x10;   // Hex
        int b = 010;    // Octal
        int c = 0b101;  // Binary
        System.out.println(a);
        System.out.println(b);
        System.out.println(c);
    }
```

```
}
```

Output

```
16
8
5
```

1.7.2 Character Escape Sequences

Examples

- \n → new line
- \t → tab

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

Output

```
Hello
Java
```

1.7.3 String Literals

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

Output

```
Welcome to Java
```

1.8 Variables and Constants

Definition

- Variable: Value can change
- Constant: Value cannot change (using `final`)

Example

```
class VariableDemo {
    public static void main(String[] args) {
        final int x = 10;
        int y = 5;
        System.out.println(x + y);
    }
}
```

Output

```
15
```

1.9 Operators

Definition

Operators perform **operations on variables**.

Types

- Arithmetic
- Relational
- Logical
- Assignment

Example

```
class OperatorDemo {
    public static void main(String[] args) {
        int a = 10, b = 5;
        System.out.println(a + b);
    }
}
```

Output

```
15
```

1.10 Type Casting

Definition

Type casting converts **one data type into another**.

Types

- Implicit
- Explicit

Example

```
class CastingDemo {
    public static void main(String[] args) {
        int a = 10;
        double b = a;
        System.out.println(b);
    }
}
```

Output

```
10.0
```

1.11 Control Statements

Definition

Control statements control the **flow of execution** in a program.

1.11.1 if Statement

```
class IfDemo {
    public static void main(String[] args) {
        int a = 10;
```

```
        if (a > 5) {
            System.out.println("A is greater");
        }
    }
}
```

Output

```
A is greater
```

1.11.2 switch Statement

```
class SwitchDemo {
    public static void main(String[] args) {
        int n = 2;
        switch (n) {
            case 1: System.out.println("One"); break;
            case 2: System.out.println("Two"); break;
        }
    }
}
```

Output

```
Two
```

1.11.3 Loop Statement

```
class LoopDemo {
    public static void main(String[] args) {
        for (int i = 1; i <= 3; i++) {
            System.out.println(i);
        }
    }
}
```

Output

```
1
2
```

```
3
```

1.11.4 continue Statement

```
class ContinueDemo {
    public static void main(String[] args) {
        for (int i = 1; i <= 5; i++) {
            if (i == 3) continue;
            System.out.println(i);
        }
    }
}
```

Output

```
1
2
4
5
```

1.11.5 break Statement

```
class BreakDemo {
    public static void main(String[] args) {
        for (int i = 1; i <= 5; i++) {
            if (i == 3) break;
            System.out.println(i);
        }
    }
}
```

Output

```
1
2
```