Intro.java – Beginner-Friendly Java Overview (Expanded for PDF)

/ This guide is designed to help complete beginners understand Java. It explains Java basics, common data types, fundamental concepts, and prepares learners for practical coding examples in this folder. /

What is Java?

Java is a **high-level, object-oriented programming language**. It allows developers to write programs that are: - Portable ("write once, run anywhere") - Secure - Reliable - Widely used in desktop, web, and mobile applications

Java programs are compiled into **bytecode**, which can run on any machine with a Java Virtual Machine (JVM).

Java Program Structure

A Java program typically contains: 1. **Class declaration** – defines a blueprint for objects 2. **Main method** – the entry point of the program 3. **Methods** – define actions or behavior 4. **Variables/Fields** – store data

Example Structure:

```
public class Example {  // Class declaration

    // Main method - program starts here
    public static void main(String[] args) {
        // Code goes here
    }

    // A sample method
    public void greet() {
        System.out.println("Hello, Java!");
    }
}
```

Basic Data Types in Java

Java has **primitive** and **non-primitive** data types.

Primitive Data Types:

- **int** – whole numbers (e.g., 5, 100)
- **double** – decimal numbers (e.g., 3.14, 19.99)
- **float** – decimal numbers with less precision (e.g., 3.14f)
- **char** – single character (e.g., 'A')
- **boolean** – true or false
- **byte, short, long** – numbers of different sizes

Non-Primitive Data Types:

- **String** – sequence of characters (e.g., "Hello World")
- **Arrays** – collection of similar data types
- **Objects** – instances of classes

Example:

```
int age = 25;                // Integer
double price = 19.99;        // Decimal number
char grade = 'A';             // Single character
boolean isJavaFun = true;    // True/False
String name = "Alex";       // Text
```

Variables and Constants

- **Variables** store data that can change during program execution
- **Constants** store data that does not change (use `final` keyword)

Example:

```
final double PI = 3.14159; // Constant value
int score = 100;           // Variable value
score = 120;               // Updated value
```

Operators

Java supports: - Arithmetic: +, -, *, /, % - Comparison: ==, !=, >, <, >=, <= - Logical: &&, ||, !

Example:

```
int a = 10;
int b = 5;
int sum = a + b;        // 15
boolean isEqual = (a == b); // false
boolean result = (a > b) && (b > 0); // true
```

Control Flow Statements

- **if / else** – decision making
- **switch** – multiple choice
- **for, while, do-while** – loops

Example:

```
int num = 10;
if(num > 5) {
    System.out.println("Number is greater than 5");
} else {
    System.out.println("Number is 5 or less");
}
```

Core Concepts: Class, Object, Public, Main

- **class** – blueprint for objects
- **object** – instance of a class
- **public** – access modifier (accessible from anywhere)
- **main** – starting point of execution

Example:

```
public class Person {
    String name;
    int age;

    // Method
    public void introduce() {
        System.out.println("Hi, I am " + name + ", age " + age);
    }
}
```

```java
public class Main {
    public static void main(String[] args) {
        Person p = new Person();
        p.name = "Alex";
        p.age = 25;
        p.introduce();
    }
}
```

Folder Structure

```
├── calculator Example
│   ├── Calculator - java oop.pdf
│   └── CalculatorMain.java
├── JAVA_OOP_COMPLETE_GUIDE.md
├── JavaOOPGuide.java
└── Java Oopguide.pdf
```

- **Calculator Example**: hands-on calculator program
- **JAVA_OOP_COMPLETE_GUIDE.md**: detailed OOP guide in Markdown
- **JavaOOPGuide.java**: Java file with all OOP examples
- **Java Oopguide.pdf**: PDF version of the OOP guide

After Learning Java

By studying this folder, learners can: - Understand **Java basics and data types** - Use **variables, operators, and control flow statements** - Design **simple classes and objects** - Understand **OOP principles** like encapsulation, inheritance, and polymorphism - Prepare for **intermediate Java projects and frameworks**

This guide is meant to be read as a **PDF introduction**, providing a foundation before diving into coding examples like the calculator and OOP guide.