**JAVA TUTORIAL**

**Java Basics**

**1. Java Intro**

* **Definition:** Java is a high-level, object-oriented programming language developed by Sun Microsystems (now owned by Oracle). It is platform-independent, meaning compiled Java code can run on any device that has a Java Virtual Machine (JVM).

**2. Java Get Started**

* To start with Java:
  1. Install the Java Development Kit (JDK).
  2. Use an Integrated Development Environment (IDE) like IntelliJ IDEA, Eclipse, or VS Code.
  3. Write your first Java program:
  4. public class Main {
  5. public static void main(String[] args) {
  6. System.out.println("Hello, World!");
  7. }
  8. }

**3. Java Syntax**

* **Definition:** Java syntax refers to the set of rules that define the structure of valid Java programs. Key points:
  + Case-sensitive.
  + Uses classes to define code structure.
  + Statements end with a semicolon ;.

**4. Java Output**

* **Definition:** Java output refers to displaying information to the user, typically using the System.out.println() method.
* System.out.println("Hello, World!"); // Outputs: Hello, World!

**5. Java Comments**

* **Definition:** Comments are notes in the code that the compiler ignores. Used to improve readability.
  + **Single-line comment:** // This is a comment.
  + **Multi-line comment:**
  + /\*
  + This is
  + a multi-line comment.
  + \*/

**6. Java Variables**

* **Definition:** Variables store data values. Each variable has a type, name, and value.
* int age = 25; // Integer variable

**7. Java Data Types**

* **Definition:** Data types specify the kind of data a variable can hold.
  + **Primitive types:** int, float, char, boolean, etc.
  + **Non-primitive types:** Arrays, Strings, Classes, etc.

**8. Java Type Casting**

* **Definition:** Converting one data type into another.
  + **Implicit Casting (Widening):** Automatic conversion (e.g., int to double).
  + **Explicit Casting (Narrowing):** Manual conversion (e.g., double to int).
* int x = (int) 3.14; // Explicit casting

**9. Java Operators**

* **Definition:** Symbols used to perform operations on variables.
  + Arithmetic: +, -, \*, /, %
  + Relational: ==, !=, <, >
  + Logical: &&, ||, !

**10. Java Strings**

* **Definition:** A sequence of characters, represented by the String class.
* String name = "Alice";

**11. Java Math**

* **Definition:** Java provides the Math class with methods for mathematical operations.
* double sqrt = Math.sqrt(16); // Returns 4.0

**12. Java Booleans**

* **Definition:** Represents two values: true or false.
* boolean isJavaFun = true;

**13. Java If...Else**

* **Definition:** Conditional statements for decision-making.
* if (age > 18) {
* System.out.println("Adult");
* } else {
* System.out.println("Minor");
* }

**14. Java Switch**

* **Definition:** A control statement that tests for equality against multiple values.
* switch (day) {
* case 1: System.out.println("Monday"); break;
* default: System.out.println("Other day");
* }

**15. Java While Loop**

* **Definition:** Executes a block of code as long as a condition is true.
* while (x < 5) {
* System.out.println(x);
* x++;
* }

**16. Java For Loop**

* **Definition:** Iterates a block of code a fixed number of times.
* for (int i = 0; i < 5; i++) {
* System.out.println(i);
* }

**17. Java Break/Continue**

* **Break:** Exits a loop prematurely.
* **Continue:** Skips the current iteration and proceeds with the next one.

**Java Methods**

**18. Java Methods**

* **Definition:** Blocks of code that perform a specific task and can be reused.
* void greet() {
* System.out.println("Hello!");
* }

**19. Java Method Parameters**

* **Definition:** Methods can accept input arguments.
* void greet(String name) {
* System.out.println("Hello, " + name);
* }

**20. Java Method Overloading**

* **Definition:** Multiple methods with the same name but different parameter lists.
* void display(int x) {}
* void display(String y) {}

**21. Java Scope**

* **Definition:** The area where a variable is accessible (local, class-level, global).

**22. Java Recursion**

* **Definition:** A method calls itself to solve smaller subproblems.
* int factorial(int n) {
* return (n == 1) ? 1 : n \* factorial(n - 1);
* }

**Java Classes and OOP**

**23. Java Classes/Objects**

* **Class:** A blueprint for objects.
* **Object:** An instance of a class.

**24. Java Class Attributes**

* **Definition:** Variables within a class.
* class Car {
* String color;
* }

**25. Java Class Methods**

* **Definition:** Functions within a class.

**26. Java Constructors**

* **Definition:** Special methods to initialize objects.

**27. Java Modifiers**

* **Definition:** Keywords like public, private, final, etc., to define scope or behavior.

**28. Java Encapsulation**

* **Definition:** Hiding data using private attributes and providing access through getters/setters.

**29. Java Inheritance**

* **Definition:** One class inherits another’s properties.

**30. Java Polymorphism**

* **Definition:** One interface, many implementations.

**31. Java Abstraction**

* **Definition:** Hiding implementation details and showing only necessary details.

**32. Java Interface**

* **Definition:** A contract that classes must adhere to by implementing its methods.

**33. Java Enums**

* **Definition:** A fixed set of constants.

**34. Java User Input**

* **Definition:** Reading input using Scanner.

**35. Java Date**

* **Definition:** Handling dates using the java.time package.

**36. Java Collections (ArrayList, LinkedList, HashMap, etc.)**

* **ArrayList:** Dynamic array.
* **HashMap:** Key-value pairs.

**Java File Handling**

**37. Java Files**

* **Definition:** Working with files (create, write, read, delete).