📝 Java Programming Course by Mosh

Part 1: Introduction to Java

- **Course Overview:** This course is designed for beginners and covers all the essential concepts to start programming in Java.
- **Instructor:** The course is taught by Mosh, a software engineer with 20 years of experience who has taught over 3 million people how to code.
- **Getting Started:** The first step is to install the necessary tools, which are the **Java Development Kit (JDK)** and the **IntelligJ IDEA** integrated development environment.
- **Core Concepts:** The course will cover the basics of Java, including:
    - How Java code is executed.
    - How to build simple algorithms.
    - How to write professional-level code.

Part 2: Understanding the Structure of a Java Program

- **Building Blocks:**
    - **Functions:** The smallest building blocks of a Java program that perform specific tasks.
    - **Classes:** Containers for related functions. When a function is part of a class, it is called a **method**.
- **Access Modifiers:**
    - The public access modifier is used to make a class or method accessible from other parts of the program.
- **Naming Conventions:**
    - **PascalCase:** Used for naming classes (e.g., MyClass).
    - **camelCase:** Used for naming methods (e.g., myMethod).
- **The main Method:**
    - The entry point for every Java program.

Part 3: Your First Java Program - "Hello World"

- **Creating a Project:**
  - Use IntelliJ IDEA to create a new Java project.
  - Set up the **Project SDK** and the **base package**.
- **Printing to the Console:**
  - The System.out.println() method is used to print text to the terminal.
- **Compilation and Execution:**
  - **Java Compiler:** Compiles the Java code into bytecode.
  - **Java Virtual Machine (JVM):** Executes the bytecode.

Part 4: Key Facts About Java

- **History:**
  - Developed by **James Gosling** at **Sun Microsystems** in 1995.
  - Originally named **Oak** and then **Green**.
  - Sun Microsystems was later acquired by **Oracle**.
- **Editions of Java:**
  - **Standard Edition (SE):** For desktop and server applications.
  - **Enterprise Edition (EE):** For large-scale enterprise applications.
  - **Micro Edition (ME):** For mobile and embedded devices.
  - **Card Edition:** For smart cards.
- **Popularity:**
  - Java is a very popular programming language with a large number of developers.
  - Java developers command a high average salary.

Part 5: Course Structure (First Part)

- **Programming Fundamentals:**
  - The course starts with the fundamentals of programming with Java.
  - It covers the Java type system, including how to work with numbers, strings, and arrays.
- **First Project: Mortgage Calculator:**
  - The first project is to build a mortgage calculator.

- **Control Flow:**
  - The course covers control flow statements, including conditional statements and loops.
- **Clean Coding:**
  - The course teaches clean coding techniques to write maintainable code.
- **Error Handling and Deployment:**
  - The course covers how to find and fix errors in your code.
  - It also covers how to package your programs for deployment.

Part 6: Programming Fundamentals in Java

- **Variables and Constants:**
  - **Variables:** Used to store data. They should be declared with meaningful names.
  - **Constants:** Variables whose value cannot be changed. They are declared using the final keyword.
- **Primitive vs. Reference Types:**
  - **Primitive Types:** Store simple values (e.g., int, double, boolean).
  - **Reference Types:** Store complex objects (e.g., String, Array).
- **Strings:**
  - **Concatenation:** Combining strings using the + operator.
  - **Useful Methods:**
    - endsWith(): Checks if a string ends with a specified suffix.
    - length(): Returns the length of a string.
    - indexOf(): Returns the index of a specified character or substring.
    - replace(): Replaces a specified character or substring with another.
    - toLowerCase(): Converts a string to lowercase.
    - toUpperCase(): Converts a string to uppercase.
    - trim(): Removes whitespace from the beginning and end of a string.
  - **Escape Sequences:** Used to represent special characters in a string (e.g., \n for a new line, \t for a tab).
- **Arrays:**
  - **Declaration and Initialization:**
    - Single-dimensional arrays: int[] numbers = new int[5];
    - Multi-dimensional arrays: int[][] matrix = new int[2][3];
  - **Accessing Elements:**
    - Elements are accessed by their index, which starts at 0.
  - **Useful Methods:**
    - Arrays.toString(): Returns a string representation of a single-dimensional array.
    - Arrays.deepToString(): Returns a string representation of a multi-dimensional array.

        - Arrays.sort(): Sorts an array.
- **Arithmetic Expressions:**
    - **Operators:** +, -, *, /, % (modulus).
    - **Increment/Decrement Operators:** ++, --.
    - **Order of Operations:** Parentheses, Exponents, Multiplication and Division (from left to right), Addition and Subtraction (from left to right).
    - **Augmented Assignment Operators:** +=, -=, *=, /=.
- **Casting and Type Conversion:**
    - **Implicit Casting:** Automatic conversion between compatible types (e.g., int to double).
    - **Explicit Casting:** Manual conversion between types (e.g., (int) 1.1).
    - **String to Number Conversion:**
        - Use wrapper classes like Integer.parseInt() and Double.parseDouble().
- **The Math Class:**
    - Provides useful mathematical methods:
        - round(): Rounds a number to the nearest integer.
        - ceil(): Rounds a number up to the nearest integer.
        - floor(): Rounds a number down to the nearest integer.
        - max(): Returns the larger of two numbers.
        - min(): Returns the smaller of two numbers.
        - random(): Returns a random number between 0.0 and 1.0.
- **Formatting Numbers:**
    - Use the NumberFormat class to format numbers as currency or percentages.
    - getCurrencyInstance(): Returns a NumberFormat object for formatting currency.
    - getPercentInstance(): Returns a NumberFormat object for formatting percentages.
- **Reading User Input:**
    - Use the Scanner class to read input from the user.
    - nextByte(): Reads a byte from the user.
    - nextLine(): Reads a line of text from the user.
    - trim(): Removes whitespace from the beginning and end of the input.

Part 7: Control Flow Statements

- **Comparison Operators:** ==, !=, >, >=, <, <=.
- **Logical Operators:**
    - && (and): Returns true if both operands are true.
    - || (or): Returns true if either operand is true.
    - ! (not): Reverses the logical state of an operand.
- **if Statements:**
    - Used to make decisions in a program.

    - if, else if, and else clauses can be used to create complex decision-making structures.
- **Ternary Operator:**
    - A concise way to implement conditional assignments.
    - variable = (condition) ? value_if_true : value_if_false;
- **switch Statements:**
    - Used to execute different blocks of code based on the value of an expression.
- **Loops:**
    - **for Loops:** Used to repeat a block of code a known number of times.
    - **while Loops:** Used to repeat a block of code when the number of iterations is unknown.
    - **do-while Loops:** Similar to while loops, but the loop body is executed at least once.
    - **break and continue:**
        - break: Terminates a loop.
        - continue: Skips to the next iteration of a loop.
    - **for-each Loops:** Used to iterate over arrays or collections.

Part 8: Project: Mortgage Calculator

- **Objective:** Build a mortgage calculator that takes user input and calculates the monthly mortgage payment.
- **Concepts Applied:**
    - Reading user input using the Scanner class.
    - Performing calculations based on a mortgage formula.
    - Formatting the output as currency using the NumberFormat class.
    - Implementing input validation and error handling using loops and conditional statements.

Part 9: Clean Coding Principles

- **Importance:** Writing clean, understandable, and maintainable code is crucial for professional software development.
- **Techniques:** The course demonstrates various techniques to improve code structure and readability, using the mortgage calculator project as an example.