Java Learning Roadmap for Beginners - Kabirosky

1. Introduction to Java

- **What is Java?**
  Overview of Java as a high-level, object-oriented programming language. Discuss its "write once, run anywhere" philosophy, and its use in web, desktop, and mobile applications.
- **Setting Up the Development Environment**
  How to install the Java Development Kit (JDK) and set up an Integrated Development Environment (IDE) like IntelliJ IDEA, Eclipse, or NetBeans.
- **Understanding Java's Structure**
  Introduction to the structure of a Java program: classes, methods, the `main` method, and the standard Java entry point (`public static void main(String[] args)`).
- **Writing Your First Java Program**
  Create and run a "Hello, World!" program, explaining the syntax and steps involved.

2. Basic Java Syntax

- **Variables and Data Types**
  Overview of Java's strongly typed nature. Learn about primitive data types (`int`, `double`, `char`, `boolean`, etc.) and non-primitive types (Strings, Arrays).
- **Operators in Java**
  - **Arithmetic Operators**: `+`, `-`, `*`, `/`, `%`
  - **Comparison Operators**: `==`, `!=`, `>`, `<`, `>=`, `<=`
  - **Logical Operators**: `&&`, `||`, `!`
  - **Assignment Operators**: `=`, `+=`, `-=`, `*=`, `/=`
- **Input and Output**
  Using `System.out.println()` for output. Introduction to `Scanner` for user input.
- **Comments and Documentation**
  Single-line (`//`) and multi-line (`/*...*/`) comments. Introduction to JavaDoc comments (`/**...*/`) for generating documentation.

3. Control Flow

- **Conditional Statements**
  - **If-Else Statements**: Basic conditional logic.
  - **Switch Statements**: Use cases and syntax for cleaner multi-conditional branches.
- **Loops**
  - **For Loops**: Syntax and use cases for iterating over sequences.
  - **While Loops and Do-While Loops**: Understanding loop conditions and iterations.
- **Loop Control Statements**

    - `break`: Exit the loop.
    - `continue`: Skip the current iteration and continue with the next.

4. Methods

- **Defining Methods**
  Syntax for defining methods in Java using the `returnType methodName(parameters)` structure.
- **Method Overloading**
  Define multiple methods with the same name but different parameters.
- **Recursion**
  Understand recursive functions and when to use them.

5. Object-Oriented Programming (OOP)

- **Classes and Objects**
  Define classes, create objects, and understand the relationship between classes and objects.
- **Attributes and Methods**
  Learn about class fields (attributes) and instance methods.
- **Constructors**
  Define constructors to initialize new objects. Explain the default and parameterized constructors.
- **Inheritance**
  Understanding how to inherit properties from a parent class using the `extends` keyword.
- **Polymorphism**
  Learn method overriding and the concept of dynamic method dispatch.
- **Encapsulation**
  Introduction to access modifiers (`private`, `protected`, `public`) and getter/setter methods.
- **Abstraction**
  Understanding abstract classes and interfaces in Java, and their role in OOP design.

6. Arrays and Strings

- **Arrays**
  Declare, initialize, and use one-dimensional and multi-dimensional arrays.
- **Common Array Operations**
  Iterating over arrays, finding the length, sorting, and searching.
- **Strings**
  Learn about the `String` class, common string methods (`length()`, `substring()`, `indexOf()`, `charAt()`, etc.), and string manipulation.
- **StringBuilder and StringBuffer**
  Understand when and how to use `StringBuilder` and `StringBuffer` for mutable strings.

7. Exception Handling

- **Understanding Exceptions**
  Differentiate between checked and unchecked exceptions.
- **Try, Catch, Finally Blocks**
  Using `try`, `catch`, `finally` to handle exceptions.
- **Throw and Throws Keywords**
  Manually throw exceptions and declare them using `throws`.
- **Custom Exceptions**
  Learn to create user-defined exceptions for more specific error handling.

8. Java Collections Framework

- **Introduction to Collections**
  Overview of the Java Collections Framework: List, Set, Map, Queue.
- **List Interface**
  Learn about `ArrayList`, `LinkedList`, and their use cases.
- **Set Interface**
  Learn about `HashSet`, `LinkedHashSet`, `TreeSet`, and their unique properties.
- **Map Interface**
  Introduction to `HashMap`, `LinkedHashMap`, `TreeMap`, and when to use each.
- **Queue Interface**
  Learn about `PriorityQueue`, `Deque`, `ArrayDeque`.

9. File Handling in Java

- **Reading and Writing Files**
  Using classes like `File`, `FileReader`, `FileWriter`, `BufferedReader`, and `BufferedWriter`.
- **Handling File Exceptions**
  Managing exceptions specific to file handling (`FileNotFoundException`, `IOException`).
- **Serialization and Deserialization**
  Introduction to object serialization (`Serializable` interface) and how to save/load objects.

10. Java Generics

- **Introduction to Generics**
  Understanding generic classes and methods for type safety.
- **Bounded Types**
  Use bounded types with generics (`<T extends Number>`).
- **Wildcard Generics**
  Use cases for `?`, `<? extends T>`, `<? super T>`.

11. Java Streams and Lambda Expressions

- **Introduction to Streams**
  Understanding streams for functional-style operations on collections.

- **Stream Operations**
  Learn about intermediate (`filter()`, `map()`, `sorted()`) and terminal operations (`collect()`, `forEach()`, `reduce()`).
- **Lambda Expressions**
  Learn to write concise and readable code using lambda expressions.
- **Functional Interfaces**
  Introduction to functional interfaces like `Predicate`, `Consumer`, `Supplier`, `Function`.

12. Multithreading and Concurrency

- **Introduction to Multithreading**
  Basics of threading and processes.
- **Creating Threads**
  Learn to create threads using `Thread` class and `Runnable` interface.
- **Thread Synchronization**
  Understand synchronized methods, blocks, and inter-thread communication.
- **Concurrency Utilities**
  Introduction to classes in `java.util.concurrent` package (`ExecutorService`, `Callable`, `Future`, `Semaphore`, etc.).

13. Networking in Java

- **Introduction to Networking**
  Understanding basic concepts of networking: IP, TCP, UDP, sockets.
- **Working with Sockets**
  Learn to create client-server applications using `Socket` and `ServerSocket` classes.
- **Handling HTTP Requests**
  Using `HttpURLConnection` to make HTTP requests.

14. Working with Databases

- **Introduction to JDBC**
  Learn how Java interacts with relational databases using JDBC (Java Database Connectivity).
- **CRUD Operations**
  Execute basic SQL operations (`SELECT`, `INSERT`, `UPDATE`, `DELETE`) through JDBC.
- **Connection Pooling**
  Introduction to using connection pools for efficient database management.

15. Java GUI Development

- **Introduction to JavaFX and Swing**
  Basic GUI components (buttons, labels, text fields, etc.) and layouts.
- **Event Handling**
  Learn to handle user interactions (mouse clicks, keyboard input) in GUI applications.

- **Building a Simple GUI Application**
  Step-by-step guide to create a basic JavaFX application.

16. Java Design Patterns

- **Introduction to Design Patterns**
  Learn about design patterns and their importance in software development.
- **Common Design Patterns in Java**
  Understand and implement patterns like Singleton, Factory, Observer, Strategy, and Decorator.

17. Testing and Debugging in Java

- **Unit Testing with JUnit**
  Write and run test cases using JUnit.
- **Test-Driven Development (TDD)**
  Understand the principles of TDD and how to apply them.
- **Debugging Techniques**
  Learn how to debug Java applications using IDE tools (breakpoints, watches, step execution).

18. Version Control with Git

- **Introduction to Git**
  Understanding repositories, commits, branches, merging, and pull requests.
- **Using GitHub**
  Learn to manage and collaborate on Java projects using GitHub.

19. Advanced Java Topics

- **Java Reflection API**
  Understand how to inspect classes, methods, and fields at runtime.
- **Annotations and Metadata**
  Learn about custom annotations and reflection-based processing.
- **Java Memory Management**
  Understanding garbage collection, memory leaks, and optimization techniques.

20. Projects and Practice

- **Building Small Projects**
  Start with projects like a simple calculator, to-do list, file explorer, or chat application.
- **Contribute to Open Source**
  Find beginner-friendly Java projects on GitHub to contribute to.
- **Capstone Project**
  Plan and build a comprehensive project like a web application, game, or enterprise-level software.

Tips for Learning Java Effectively:

- **Understand Concepts Deeply**: Don’t just memorize; understand why things work the way they do.
- **Practice Regularly**: The more you code, the more familiar you become with Java syntax and paradigms.
- **Work on Real Projects**: Build small to medium-sized projects to apply what you've learned.
- **Read Java Documentation**: Oracle’s Java documentation is a valuable resource to understand the core of Java.
- **Engage with the Community**: Join Java communities, forums, and study groups to collaborate and learn from others.