**30-Day Java & Angular Study Plan with LeetCode Problems**

**WEEK 1: JAVA FUNDAMENTALS & ANGULAR BASICS**

**DAY 1: Java Setup & First Steps**

**~~Java Morning (2 hours)~~**

* ~~Set up Java Development Kit (JDK)~~
* ~~Configure IDE (Eclipse, IntelliJ, or VS Code)~~
* ~~Write first "Hello World" program~~
* ~~Learn about variables, primitive data types, and operators~~
* ~~Practice type conversion and basic arithmetic operations~~

**~~Java Afternoon (2 hours)~~**

* ~~HackerRank (1 hour): Complete "Java Introduction" challenges:~~ 
  + ~~Welcome to Java!~~
  + ~~Java Stdin and Stdout I~~
  + ~~Java If-Else~~
  + ~~Java Stdin and Stdout II~~
  + ~~Java Output Formatting~~
* ~~Practice (1 hour): Create simple programs that use variables and operators~~

**~~Angular (1 hour) - Modules 1-11~~**

* ~~Set up Node.js and npm~~
* ~~Install Angular CLI~~
* ~~Create your first Angular application~~
* ~~Understand Angular project structure~~
* ~~Introduction to TypeScript basics~~

**LeetCode Problems:**

* [1. Two Sum](https://leetcode.com/problems/two-sum/) (Easy)
* [1480. Running Sum of 1d Array](https://leetcode.com/problems/running-sum-of-1d-array/) (Easy)

**DAY 2: Java Control Flow**

**Java Morning (2 hours)**

* ~~Decision making statements (if, if-else, switch)~~
* ~~Looping structures (for, while, do-while)~~
* ~~Jump statements (break, continue, return)~~
* ~~Create programs implementing different control flow statements~~

**Java Afternoon (2 hours)**

* HackerRank (1 hour): Complete challenges:
  + ~~Java Loops I~~
  + ~~Java Loops II~~
  + ~~Java Datatypes~~
  + ~~Java End-of-file~~
* Practice (1 hour): Implement a simple calculator program using control flow structures

**~~Angular (1 hour) - Modules 12-22~~**

* ~~TypeScript variables, types, and functions~~
* ~~TypeScript classes and interfaces~~
* ~~Angular component basics~~

**LeetCode Problems:**

* [9. Palindrome Number](https://leetcode.com/problems/palindrome-number/) (Easy)
* [1342. Number of Steps to Reduce a Number to Zero](https://leetcode.com/problems/number-of-steps-to-reduce-a-number-to-zero/) (Easy)

**DAY 3: Java Classes & Objects I**

**Java Morning (2 hours)**

* Object-oriented programming concepts
* Classes and objects
* Class attributes and methods
* Constructors and this keyword
* Object instantiation and reference variables

**Java Afternoon (2 hours)**

* HackerRank (1 hour): Complete OOP challenges:
  + Java Object Oriented Programming
  + Java Inheritance I
  + Java Method Overriding
* Practice (1 hour): Create a simple Student class with attributes and methods

**Angular (1 hour) - Modules 23-33**

* Creating components
* Component templates
* Understanding component lifecycle

**LeetCode Problems:**

* [1603. Design Parking System](https://leetcode.com/problems/design-parking-system/) (Easy)
* [705. Design HashSet](https://leetcode.com/problems/design-hashset/) (Easy)

**DAY 4: Java Classes & Objects II**

**Java Morning (2 hours)**

* Encapsulation and access modifiers
* Getter and setter methods
* Static variables and methods
* Final variables and methods
* Method overloading

**Java Afternoon (2 hours)**

* HackerRank (1 hour): Complete challenges:
  + Java Inheritance II
  + Java Abstract Class
  + Java Interface
* Practice (1 hour): Extend your Student class with proper encapsulation and static members

**Angular (1 hour) - Modules 34-44**

* Data binding in Angular
* Property binding
* Event binding
* Two-way binding with ngModel

**LeetCode Problems:**

* [155. Min Stack](https://leetcode.com/problems/min-stack/) (Medium)
* [706. Design HashMap](https://leetcode.com/problems/design-hashmap/) (Easy)

**DAY 5: Java Inheritance & Polymorphism**

**Java Morning (2 hours)**

* Inheritance concepts
* Super keyword and constructor chaining
* Method overriding
* Polymorphism concepts
* Dynamic method dispatch

**Java Afternoon (2 hours)**

* HackerRank (1 hour): Complete challenges:
  + Java Method Overriding 2 (Super Keyword)
  + Java Instanceof keyword
  + Java Iterator
* Practice (1 hour): Create a simple inheritance hierarchy for a school system

**Angular (1 hour) - Modules 45-55**

* Directives in Angular
* Built-in directives: ngIf, ngFor
* Custom directive creation

**LeetCode Problems:**

* [1114. Print in Order](https://leetcode.com/problems/print-in-order/) (Easy)
* [622. Design Circular Queue](https://leetcode.com/problems/design-circular-queue/) (Medium)

**DAY 6: Java Abstract Classes & Interfaces**

**Java Morning (2 hours)**

* Abstract classes and methods
* Interface definition and implementation
* Default and static methods in interfaces
* Functional interfaces
* Multiple interface implementation

**Java Afternoon (2 hours)**

* HackerRank (1 hour): Complete challenges:
  + Java Interface
  + Java Exception Handling
  + Java Exception Handling (Try-catch)
* Practice (1 hour): Extend your school system with interfaces and abstract classes

**Angular (1 hour) - Modules 56-66**

* Pipes in Angular
* Built-in pipes
* Custom pipe creation
* Pure and impure pipes

**LeetCode Problems:**

* [303. Range Sum Query - Immutable](https://leetcode.com/problems/range-sum-query-immutable/) (Easy)
* [208. Implement Trie (Prefix Tree)](https://leetcode.com/problems/implement-trie-prefix-tree/) (Medium)

**DAY 7: Java Packages & Exception Handling**

**Java Morning (2 hours)**

* Packages in Java
* Import statements
* Access control with packages
* Exception handling (try, catch, finally)
* Checked vs. unchecked exceptions
* Creating custom exceptions

**Java Afternoon (2 hours)**

* HackerRank (1 hour): Complete challenges:
  + Java Factory Pattern
  + Java Hashset
  + Java Generics
* Project (1 hour): Start a mini-project implementing concepts learned this week

**Angular (1 hour) - Modules 67-77**

* Angular styling
* Component-specific styles
* Global styles
* CSS encapsulation modes

**LeetCode Problems:**

* [1046. Last Stone Weight](https://leetcode.com/problems/last-stone-weight/) (Easy)
* [232. Implement Queue using Stacks](https://leetcode.com/problems/implement-queue-using-stacks/) (Easy)

**WEEK 2: JAVA INTERMEDIATE & ANGULAR SERVICES**

**DAY 8: Java Collections Framework I**

**Java Morning (2 hours)**

* Collections Framework overview
* List interface and implementations (ArrayList, LinkedList)
* Set interface and implementations (HashSet, TreeSet)
* Basic operations on collections

**Java Afternoon (2 hours)**

* HackerRank (1 hour): Complete "Data Structures" challenges:
  + Java List
  + Java Map
  + Java Sort
* Practice (1 hour): Implement programs using different collection types

**Angular (1 hour) - Modules 78-88**

* Services in Angular
* Dependency injection
* Creating and using services

**LeetCode Problems:**

* [217. Contains Duplicate](https://leetcode.com/problems/contains-duplicate/) (Easy)
* [1. Two Sum](https://leetcode.com/problems/two-sum/) (Easy, review with HashSet)

**DAY 9: Java Collections Framework II**

**Java Morning (2 hours)**

* Map interface and implementations (HashMap, TreeMap)
* Queue and Deque interfaces
* Collections utility class
* Sorting and searching collections

**Java Afternoon (2 hours)**

* HackerRank (1 hour): Complete challenges:
  + Java Priority Queue
  + Java BitSet
  + Java Comparator
* Practice (1 hour): Implement a contact management system using collections

**Angular (1 hour) - Modules 89-99**

* Communicating between components
* @Input and @Output decorators
* EventEmitter
* Service-based communication

**LeetCode Problems:**

* [347. Top K Frequent Elements](https://leetcode.com/problems/top-k-frequent-elements/) (Medium)
* [1338. Reduce Array Size to The Half](https://leetcode.com/problems/reduce-array-size-to-the-half/) (Medium)

**DAY 10: Java Generics**

**Java Morning (2 hours)**

* Generic types and methods
* Type parameters and wildcards
* Bounded type parameters
* Generic interfaces and classes

**Java Afternoon (2 hours)**

* HackerRank (1 hour): Complete algorithm challenges:
  + Problem Solving: Basic
  + Algorithm implementation challenges
* Practice (1 hour): Refactor your collections programs to use generics

**Angular (1 hour) - Modules 100-110**

* Angular modules
* Feature modules
* Shared modules
* Lazy loading modules

**LeetCode Problems:**

* [20. Valid Parentheses](https://leetcode.com/problems/valid-parentheses/) (Easy)
* [146. LRU Cache](https://leetcode.com/problems/lru-cache/) (Medium)

**DAY 11: Java I/O Fundamentals**

**Java Morning (2 hours)**

* File class
* FileInputStream and FileOutputStream
* FileReader and FileWriter
* BufferedReader and BufferedWriter
* Scanner and PrintWriter

**Java Afternoon (2 hours)**

* HackerRank (1 hour): Complete challenges:
  + Java Regex
  + Java Regex 2 - Duplicate Words
  + Valid Username Regular Expression
* Practice (1 hour): Create a file processing utility

**Angular (1 hour) - Modules 111-121**

* Routing in Angular
* Router module setup
* Route configuration
* Route parameters

**LeetCode Problems:**

* [929. Unique Email Addresses](https://leetcode.com/problems/unique-email-addresses/) (Easy)
* [1472. Design Browser History](https://leetcode.com/problems/design-browser-history/) (Medium)

**DAY 12: Java Serialization & Date/Time API**

**Java Morning (2 hours)**

* Object serialization
* Serializable interface
* transient keyword
* Java Date and Time API
* LocalDate, LocalTime, LocalDateTime

**Java Afternoon (2 hours)**

* HackerRank (1 hour): Complete challenges:
  + Java BigDecimal
  + Java BigInteger
  + Java Primality Test
* Practice (1 hour): Build a scheduler application using Date/Time API

**Angular (1 hour) - Modules 122-132**

* Navigation in Angular
* Router links
* Programmatic navigation
* Route guards

**LeetCode Problems:**

* [1360. Number of Days Between Two Dates](https://leetcode.com/problems/number-of-days-between-two-dates/) (Easy)
* [539. Minimum Time Difference](https://leetcode.com/problems/minimum-time-difference/) (Medium)

**DAY 13: Java Regular Expressions**

**Java Morning (2 hours)**

* Regular expression syntax
* Pattern and Matcher classes
* Character classes and quantifiers
* Grouping and backreferences
* String methods with regex

**Java Afternoon (2 hours)**

* HackerRank (1 hour): Complete challenges:
  + Pattern Syntax Checker
  + Java Regex 3 - Username Checker
  + Tag Content Extractor
* Practice (1 hour): Create a form validation utility using regex

**Angular (1 hour) - Modules 133-143**

* Child routes
* Secondary routes
* Route data and resolvers
* Route animations

**LeetCode Problems:**

* [520. Detect Capital](https://leetcode.com/problems/detect-capital/) (Easy)
* [8. String to Integer (atoi)](https://leetcode.com/problems/string-to-integer-atoi/) (Medium)

**DAY 14: Java Multi-threading Basics**

**Java Morning (2 hours)**

* Thread class and Runnable interface
* Creating and starting threads
* Thread lifecycle
* Thread priorities and scheduling
* Sleep, yield, and join methods

**Java Afternoon (2 hours)**

* HackerRank (1 hour): Take Java coding contest or practice challenges
* Practice (1 hour): Implement a multi-threaded application

**Angular (1 hour) - Modules 144-154**

* Route guards in depth
* CanActivate
* CanDeactivate
* CanLoad guards

**LeetCode Problems:**

* [1115. Print FooBar Alternately](https://leetcode.com/problems/print-foobar-alternately/) (Medium)
* [1226. The Dining Philosophers](https://leetcode.com/problems/the-dining-philosophers/) (Medium)

**WEEK 3: JAVA ADVANCED & ANGULAR FORMS**

**DAY 15: Java Thread Synchronization**

**Java Morning (2 hours)**

* Thread synchronization concepts
* Synchronized methods and blocks
* Volatile keyword
* Object locks and monitor concept
* Thread-safe collections

**Java Afternoon (2 hours)**

* HackerRank (1 hour): Focus on medium algorithm challenges
* Practice (1 hour): Implement producer-consumer problem

**Angular (1 hour) - Modules 155-165**

* Template-driven forms
* Form controls
* ngModel in depth
* Form validation

**LeetCode Problems:**

* [1195. Fizz Buzz Multithreaded](https://leetcode.com/problems/fizz-buzz-multithreaded/) (Medium)
* [1188. Design Bounded Blocking Queue](https://leetcode.com/problems/design-bounded-blocking-queue/) (Medium)

**DAY 16: Java Concurrency Utilities I**

**Java Morning (2 hours)**

* java.util.concurrent package
* ExecutorService and thread pools
* Callable and Future
* CountDownLatch and CyclicBarrier
* Semaphore

**Java Afternoon (2 hours)**

* HackerRank (1 hour): Medium difficulty challenges
* Practice (1 hour): Refactor multi-threaded applications to use concurrency utilities

**Angular (1 hour) - Modules 166-176**

* Template form validation
* Built-in validators
* Custom validators
* Validation messages

**LeetCode Problems:**

* [1242. Web Crawler Multithreaded](https://leetcode.com/problems/web-crawler-multithreaded/) (Medium)
* [1116. Print Zero Even Odd](https://leetcode.com/problems/print-zero-even-odd/) (Medium)

**DAY 17: Java Concurrency Utilities II**

**Java Morning (2 hours)**

* Atomic variables
* Locks and conditions
* BlockingQueue implementations
* ConcurrentMap implementations
* CompletableFuture

**Java Afternoon (2 hours)**

* HackerRank (1 hour): Focus on challenges that benefit from concurrency
* Practice (1 hour): Build a concurrent web crawler

**Angular (1 hour) - Modules 177-187**

* Reactive forms
* FormGroup and FormControl
* FormBuilder
* Dynamic form controls

**LeetCode Problems:**

* [1117. Building H2O](https://leetcode.com/problems/building-h2o/) (Medium)
* [1279. Traffic Light Controlled Intersection](https://leetcode.com/problems/traffic-light-controlled-intersection/) (Easy)

**DAY 18: Java Lambda Expressions**

**Java Morning (2 hours)**

* Lambda expression syntax
* Functional interfaces
* Method references
* Built-in functional interfaces (Consumer, Supplier, Function, Predicate)
* Using lambdas with collections

**Java Afternoon (2 hours)**

* HackerRank (1 hour): Functional programming challenges
* Practice (1 hour): Refactor code to use lambda expressions

**Angular (1 hour) - Modules 188-198**

* Reactive form validation
* FormArray
* Nested form groups
* Custom validators

**LeetCode Problems:**

* [1051. Height Checker](https://leetcode.com/problems/height-checker/) (Easy)
* [881. Boats to Save People](https://leetcode.com/problems/boats-to-save-people/) (Medium)

**DAY 19: Java Stream API I**

**Java Morning (2 hours)**

* Stream API introduction
* Creating streams
* Intermediate operations (filter, map, sorted)
* Terminal operations (collect, reduce, forEach)
* Optional class

**Java Afternoon (2 hours)**

* HackerRank (1 hour): Solve challenges using streams
* Practice (1 hour): Process collections data using streams

**Angular (1 hour) - Modules 199-209**

* HTTP client
* Making GET requests
* Headers and parameters
* Error handling

**LeetCode Problems:**

* [1431. Kids With the Greatest Number of Candies](https://leetcode.com/problems/kids-with-the-greatest-number-of-candies/) (Easy)
* [2149. Rearrange Array Elements by Sign](https://leetcode.com/problems/rearrange-array-elements-by-sign/) (Medium)

**DAY 20: Java Stream API II**

**Java Morning (2 hours)**

* Advanced stream operations
* Parallel streams
* Collectors class
* Grouping and partitioning
* Custom collectors

**Java Afternoon (2 hours)**

* HackerRank (1 hour): Data analysis challenges
* Practice (1 hour): Implement a data processing application with streams

**Angular (1 hour) - Modules 210-220**

* POST, PUT, DELETE requests
* Request interceptors
* Progress events
* Testing HTTP requests

**LeetCode Problems:**

* [2006. Count Number of Pairs With Absolute Difference K](https://leetcode.com/problems/count-number-of-pairs-with-absolute-difference-k/) (Easy)
* [791. Custom Sort String](https://leetcode.com/problems/custom-sort-string/) (Medium)

**DAY 21: Java Modules & New Features**

**Java Morning (2 hours)**

* Java Platform Module System
* Creating and using modules
* Java 9+ features (var keyword, private methods in interfaces)
* Text blocks (Java 15+)
* Records (Java 16+)

**Java Afternoon (2 hours)**

* HackerRank (1 hour): Practice newer Java features in challenges
* Practice (1 hour): Refactor existing code to use modern Java features

**Angular (1 hour) - Modules 221-231**

* RxJS basics
* Observables
* Subjects
* Operators

**LeetCode Problems:**

* [1365. How Many Numbers Are Smaller Than the Current Number](https://leetcode.com/problems/how-many-numbers-are-smaller-than-the-current-number/) (Easy)
* [2011. Final Value of Variable After Performing Operations](https://leetcode.com/problems/final-value-of-variable-after-performing-operations/) (Easy)

**WEEK 4: JAVA PROFESSIONAL SKILLS & ANGULAR ADVANCED**

**DAY 22: Java Design Patterns I**

**Java Morning (2 hours)**

* Design pattern categories
* Singleton pattern
* Factory pattern
* Builder pattern
* Prototype pattern

**Java Afternoon (2 hours)**

* HackerRank (1 hour): Medium-hard level problems
* Practice (1 hour): Implement selected design patterns

**Angular (1 hour) - Modules 232-242**

* State management principles
* Services as state stores
* Component state
* Local storage

**LeetCode Problems:**

* [705. Design HashSet](https://leetcode.com/problems/design-hashset/) (Easy, implement with Singleton)
* [355. Design Twitter](https://leetcode.com/problems/design-twitter/) (Medium)

**DAY 23: Java Design Patterns II**

**Java Morning (2 hours)**

* Observer pattern
* Strategy pattern
* Decorator pattern
* Adapter pattern
* Composite pattern

**Java Afternoon (2 hours)**

* HackerRank (1 hour): Medium-hard level problems implementing patterns
* Practice (1 hour): Refactor applications using appropriate design patterns

**Angular (1 hour) - Modules 243-253**

* NgRx introduction
* Store
* Actions
* Reducers

**LeetCode Problems:**

* [455. Assign Cookies](https://leetcode.com/problems/assign-cookies/) (Easy)
* [1244. Design A Leaderboard](https://leetcode.com/problems/design-a-leaderboard/) (Medium)

**DAY 24: SOLID Principles**

**Java Morning (2 hours)**

* Single Responsibility Principle
* Open/Closed Principle
* Liskov Substitution Principle
* Interface Segregation Principle
* Dependency Inversion Principle

**Java Afternoon (2 hours)**

* HackerRank (1 hour): Practice design challenges
* Practice (1 hour): Refactor code to follow SOLID principles

**Angular (1 hour) - Modules 254-265**

* Effects
* Selectors
* Entity state
* NgRx DevTools

**LeetCode Problems:**

* [706. Design HashMap](https://leetcode.com/problems/design-hashmap/) (Easy, apply SOLID)
* [1396. Design Underground System](https://leetcode.com/problems/design-underground-system/) (Medium)

**DAY 25: Java Testing with JUnit**

**Java Morning (2 hours)**

* JUnit framework
* Writing test cases
* Assertions
* Test suites
* Test lifecycle annotations

**Java Afternoon (2 hours)**

* HackerRank (1 hour): Timed coding contests
* Practice (1 hour): Write tests for previously created applications

**Angular (1 hour) - Modules 266-276**

* Angular testing introduction
* TestBed
* Component testing
* Service testing

**LeetCode Problems:**

* [2235. Add Two Integers](https://leetcode.com/problems/add-two-integers/) (Easy, focus on test cases)
* [67. Add Binary](https://leetcode.com/problems/add-binary/) (Easy, focus on edge cases and testing)

**DAY 26: Java Mocking with Mockito**

**Java Morning (2 hours)**

* Mockito framework
* Creating mocks
* Stubbing method calls
* Verifying behaviors
* Argument matchers and captor

**Java Afternoon (2 hours)**

* HackerRank (1 hour): Timed coding contests
* Practice (1 hour): Add mocking to your test suites

**Angular (1 hour) - Modules 277-287**

* Testing HTTP requests
* Testing with stubs
* Isolated vs. integrated tests
* E2E testing with Protractor

**LeetCode Problems:**

* [1108. Defanging an IP Address](https://leetcode.com/problems/defanging-an-ip-address/) (Easy, practice mocking external services)
* [641. Design Circular Deque](https://leetcode.com/problems/design-circular-deque/) (Medium, with test cases)

**DAY 27: Java Build Tools & CI/CD**

**Java Morning (2 hours)**

* Maven basics
* Gradle basics
* Project configuration
* Dependency management
* Build lifecycle

**Java Afternoon (2 hours)**

* HackerRank (1 hour): Timed coding contests
* Practice (1 hour): Set up build processes for your projects

**Angular (1 hour) - Modules 288-299**

* Angular optimization
* AOT compilation
* Lazy loading
* Preloading strategies
* Build optimization

**LeetCode Problems:**

* [412. Fizz Buzz](https://leetcode.com/problems/fizz-buzz/) (Easy, implement with CI pipeline)
* [735. Asteroid Collision](https://leetcode.com/problems/asteroid-collision/) (Medium)

**DAY 28: Java Data Structures Implementation**

**Java Morning (2 hours)**

* Implementing linked lists
* Implementing stack and queue
* Implementing binary tree
* Implementing hash table
* Common algorithms (sorting, searching)

**Java Afternoon (2 hours)**

* HackerRank (2 hours): Full mock interview session (timed challenges)
* Review: Analyze solutions and optimize

**Angular (1 hour) - Modules 300-310**

* Deployment preparations
* Environment configuration
* Build for production
* Server configuration basics

**LeetCode Problems:**

* [707. Design Linked List](https://leetcode.com/problems/design-linked-list/) (Medium)
* [703. Kth Largest Element in a Stream](https://leetcode.com/problems/kth-largest-element-in-a-stream/) (Easy)

**DAY 29: Algorithm Practice**

**Java Morning (2 hours)**

* String manipulation algorithms
* Dynamic programming basics
* Greedy algorithms
* Backtracking algorithms
* Graph algorithms

**Java Afternoon (2 hours)**

* HackerRank (2 hours): Full mock interview session (timed challenges)
* Review: Analyze solutions and optimize

**Angular (1 hour) - Modules 311-321**

* Angular Universal (SSR)
* Progressive Web Apps with Angular
* Service workers
* Push notifications

**LeetCode Problems:**

* [70. Climbing Stairs](https://leetcode.com/problems/climbing-stairs/) (Easy, dynamic programming)
* [322. Coin Change](https://leetcode.com/problems/coin-change/) (Medium, dynamic programming)

**DAY 30: Live Coding Simulation & Interview Preparation**

**Java Morning (2 hours)**

* System design basics
* Object modeling
* Performance considerations
* Best practices review
* Mock interview with another developer (if possible)

**Java Afternoon (2 hours)**

* HackerRank (2 hours): Final mock interview session
* Review: Final assessment of strengths and weaknesses

**Angular (1 hour) - Modules 322-327**

* Advanced component patterns
* Content projection
* Dynamic components
* Angular libraries
* Final review

**LeetCode Problems:**

* [53. Maximum Subarray](https://leetcode.com/problems/maximum-subarray/) (Medium)
* [121. Best Time to Buy and Sell Stock](https://leetcode.com/problems/best-time-to-buy-and-sell-stock/) (Easy)

**Study Resources**

**Java Resources**

* Oracle Java Documentation
* "Effective Java" by Joshua Bloch
* Java Specification (JLS)
* Baeldung tutorials
* HackerRank Java track
* LeetCode problems

**Angular Resources**

* Angular.io official documentation
* Angular University courses
* Your 327-module Angular course

**Additional Preparation**

1. Code repository: Maintain a GitHub repository with all your practice code
2. Flashcards: Create flashcards for quick review of key concepts
3. Peer review: If possible, have peers review your code
4. Mock interviews: Practice with friends or online platforms
5. Problem journal: Document challenges faced and solutions found

**LeetCode Study Tips**

1. Start with easy problems that match your daily topics
2. Focus on understanding the algorithm rather than memorizing solutions
3. Review and optimize your solutions after solving each problem
4. Practice explaining your solution approach (useful for interviews)
5. If stuck, look at the discussion section but try to understand the solution rather than copy it
6. Time yourself to simulate interview conditions
7. For hard problems, try to break them down into smaller, manageable steps

**Post-Study Plan Assessment**

After completing the 30-day plan, evaluate your progress:

* Review all completed HackerRank and LeetCode challenges
* Take a final comprehensive assessment test
* Identify any remaining knowledge gaps
* Create a maintenance plan for continued learning

**Beyond the 30-Day Plan: Reaching Interview Mastery**

To reach 90%+ interview readiness, consider these essential next steps:

* **Extending your practice with medium/hard LeetCode problems for another 2-4 weeks**
  + Focus on problems that test advanced data structures
  + Practice dynamic programming, graph algorithms, and complex optimization
  + Set time limits to simulate interview conditions
* **Adding system design practice for senior roles**
  + Study distributed systems concepts
  + Practice designing scalable applications
  + Learn about microservices architecture and API design
  + Review real-world system designs from major companies
* Building at least one full-stack project that demonstrates both Java and Angular skills
* Conducting more mock interviews with experienced developers