# 30-Day Java & Angular Study Plan

## 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~~

### 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

### 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

### 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

### 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

### 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

### 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

## 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

### 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

### 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

### 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

### 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

### 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

### 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

## 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

### 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

### 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

### 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

### 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

### 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

### 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

## 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

### 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

### 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

### 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

### 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

### 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

### 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

### 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

### 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

## 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

## Post-Study Plan Assessment

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

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