**Advanced JAVA through Online**

**Duration: 3 hours per day.**

**Please find the below course outline:**

**Web Services**

**         Introducing Web Services**

**         Types of Web Services**

**         SOAP (Consuming and Creating**

**         WSDL, XML, end points, UDDI**

**         RESTful web services (HTTP methods)**

**         CRUD operations using REST**

**         CRUD operations using REST**

**Angular JS**

**         Introduction to Angular JS, Features RIA, SPA, MVC, Two way Data binding**

**         Controllers in Angular JS**

**         Services using service and factory methods**

**         Form Handling in Angular**

**         Routing with Route module**

**         Angular to back end service communication (servlet, RESTful web services)**

**         What is software design principles**

**         Design vs Patterns**

**         Understanding the principle of “Single Responsibility” and its applications.**

**         Understanding “Open/Closed Principle” along with its applications.**

**         Understanding “Liskov Substitution Principle” along with its applications.**

**         Understanding “Interface Segregation Principle” along with its applications.**

**         Understanding “Dependency Inversion Principle” along with its applications.**

**         Understanding the “GRASP” principles along with its guidelines.**

**         Understanding General Responsibility Assignment Software Patterns**

**         Introduction to Design patterns**

**         GOD Patterns basics - Creational, Structural and Behavioural**

**Widely used patterns**

**         Builder**

**         Prototype (cloning)**

**         Adapter (asList, toString)**

**         COR (Chain of Responsibility) - Logging**

**         Singleton (Runtime)**

**         Factory (Action mapping)**

**         Proxy**

**         Observer (Event Listener)**

**         MVC (Web Frameworks)**

**         Façade Pattern**

**         Template pattern**

**         Understanding a fundamental architectural style [Model View Controller] which is the basic property of any well designed system across different streams of engineering including software engineering based systems.**

**         Understanding different types of software quality attributes (functional and non-functional requirements)**

**         Understanding OOPS principles**

**         Polymorphism**

**         Understanding “Polymorphism” and its importance to Object Oriented Systems.**

**         Inheritance**

**         Understanding “Parent Child” relationship and its importance to Object Oriented Systems.**

**         Encapsulation**

**         Understanding the concept of “Encapsulation” and its importance to Object Oriented Systems.**

**         Understanding the concept of “Abstraction” and its importance to Object Oriented Systems.**

**         Relationships among the classes**

**         IS – A relationship**

**         HAS – A relationship Association**

**         Understanding whole part relationship and its importance to system designing.**

**         Types of Associations**

**         Aggregation**

**         Understanding “Aggregation” relationship and  its importance to system designing.**

**         Implementation of “Aggregation” in Java**

**         Composition**

**         Understanding “Composition” relationship and  its importance to system designing.**

**         Implementation of “Composition” in Java**

**         Overview of UML**

**         UML diagrams**

**         Use Case diagram**

**         Class diagram**

**         Sequence diagram**

**         Collaboration diagram**

**         state chart diagram**

**         Activity diagram**

**         Deployment Diagram**

**         What is build management - What is build - Introduction to Maven - Advantanges of Maven**

**         Casual comparision between ANT and Maven**

**         Installing Maven - Maven Project - Maven Project Structure**

**         Creating Maven project from command line (mvn)**

**         Creating Maven project in Eclipse**

**         About POM (Project Object Model) and its elements (group id, artifact id, version, packaging, name, dependencies)**

**         How to build a project?**

**         Phase in Maven (validate, compile, test, package, integration-test, verify, install, deploy)**

**         Special phases like clean, site**

**         Packaging (jar, war, ear)**

**         Unit testing with Maven project**

**         Creating web application and .war file**

**         What is SCM? - Introduction**

**         SCM components - (requirement analysis, document management, deployment, defect tracking, source code versioning, development tools, designing tools)**

**         What is Version Control and CVS?**

**         Introduction to CVS**

**         Repository**

**         Terminologies (check-out, check-in, branch, merge, code review, approval, version history, version difference)**

**         Why version control is important**

**         Examples of CVS: Win cvs, ClearCase, Subversion, Git**

**         What is Git? And why it is popular?**

**         Git repositories**

**         Installing Git**

**         Getting help from Git**

**         Getting a Git repository (clone)**

**         Creating new repository**

**         Tracking files in Git**

**         Checking the status of your files**

**         Push the files from local repository into Github repository**

**         Important commands like: add, commit, push, status, diff, diff-staged, rm, log**

**         Undoing things**

**         Staging a file, upstaging a file**

**         Git process (working area, staging area, Git repository)**

**         What is code analysis?**

**         Good coding practices**

**         Identifying issues with the source code**

**         Static code analysis with PMD**

**         Eclipse - PMD integration**

**         What is performance management?**

**         How to check whether our code is performing well?**

**         Java Performance tuning**

**         Performance analysis tool**

**         Profiling tools - GC, Method calls, Object creation profiling, Monitoring Gross memory usage, Performance check list**

**         Object creation statistics**

**         String performance**

**         Loops and switches**

**         I/O, Logging and Console output**

**         Appropriate data structure and algorithms**

**         Performance on the DB side (usage of functions, stored process)**

**Kindly share the below details for the same:**