|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 02-01-2025 | Thursday | Tech Stack | Core Java(Java SE 20) |  |  |  |
| 03-01-2025 | Friday | Tech Stack | Core Java(Java SE 20) |  |  |  |
| 06-01-2025 | Monday | Tech Stack | Java Threads and Collections |  |  |  |
| 13-01-2025 | Monday | Tech Stack | **HTML, CSS and JS** |  |  |  |
| 15-01-2025 | Wednesday | Tech Stack | **HTML, CSS and JS** |  |  |  |
| 16-01-2025 | Thursday | Tech Stack | **J2EE, Security** |  |  |  |
| 17-01-2025 | Friday | Tech Stack | **J2EE, Security** |  |  |  |
| 20-01-2025 | Monday | Tech Stack | Spring, Spring Boot, Microservices |  |  |  |
| 21-01-2025 | Tuesday | Tech Stack | Spring, Spring Boot, Microservices |  |  |  |
| 22-01-2025 | Wednesday | Tech Stack | API, API First, **Security** |  |  |  |
| 23-01-2025 | Thursday | Tech Stack | Node JS |  |  |  |
| 24-01-2025 | Friday | Tech Stack | React JS |  |  |  |
| 27-01-2025 | Monday | Tech Stack | Angular JS |  |  |  |
| 28-01-2025 | Tuesday | Tech Stack | Angular JS |  |  |  |
| 29-01-2025 | Wednesday | Tech Stack | Unit Testing & Code Coverage - Junit, Jacoco |  |  |  |
| 30-01-2025 | Thursday | Tech Stack | Testing basics concepts and NFR |  |  |  |
| 31-01-2025 | Friday | Tech Stack | Automation (Selenium & Rest API) |  |  |  |
| 03-02-2025 | Monday | Tech Stack | Automation (Selenium & Rest API) |  |  |  |

Course content

Core java – 2 days

### Introduction to Java

* *Features of Java*
* *JDK, JRE and JVM*
* *OOPs*
* *Class, Object, Attribute, method*
* *Access Modifiers – Private, Public*
* *Constructors*
* ***Default and Args Constructor***
* ***Getters and Setters***
* *Data types and Operators*
* *Arrays*
* *Control Flow statements*
* *Enums*

### Class Design

* *Class and Objects*
* *Instance Variables and Static Variables*
* *Methods and Constructors*

### Polymorphism

* *Method Overloading*
* *Constructor Overloading*
* *Use of “this” keyword*
* *Variable Argument Method (Varargs) in Java*
* *Varargs versus Overloading*

### Inheritance

* *Inheritance Basics*
* *Use of super keyword*
* *Overriding*
* *Rules regarding overriding*
* *Runtime Polymorphism*

### Advanced Class Features

* *Static variables, Static block and methods*
* *Static Import*

Abstraction

* *Abstract classes and methods*
* *Final classes and methods*
* *Interfaces*

Collections and Threads

Threads

* *Introduction to Threads*
* *Thread class and Runnable Interface*
* *Different ways of creating threads*
* *Thread priorities*
* *Methods of thread class*
* *Lifecycle of a thread*
* *Use of sleep(), join() methods*
* *Synchronization*
* *Inter-thread Communication*

### 1. **Introduction to Java Collections Framework (JCF)**

* What is the Collections Framework?
* Core interfaces: Collection, List, Set, Queue, Map
* Benefits of using collections
* Differences between Arrays and Collections

### 2. **Collection Interface**

* Overview of Collection interface
* Basic operations: add(), remove(), size(), clear(), contains(), etc.

### 3. **List Interface**

* Implementation classes: ArrayList, LinkedList, Vector, Stack
* Key methods: add(), get(), set(), remove(), indexOf(), size()
* Differences between ArrayList and LinkedList
* Usage of ListIterator

### 4. **Set Interface**

* Implementation classes: HashSet, LinkedHashSet, TreeSet
* Key methods: add(), remove(), size(), contains()
* Characteristics of Sets (No duplicates, unordered/ordered, etc.)
* Difference between HashSet and TreeSet

### 5. **Queue Interface**

* Implementation classes: LinkedList, PriorityQueue, ArrayDeque
* Key methods: offer(), poll(), peek(), remove()
* Usage of Queue for FIFO (First In First Out) order
* Priority Queue: Comparator and natural ordering

### 6. **Map Interface**

* Implementation classes: HashMap, LinkedHashMap, TreeMap, Hashtable, ConcurrentHashMap
* Key methods: put(), get(), remove(), containsKey(), keySet(), values()
* Differences between HashMap and TreeMap
* LinkedHashMap for maintaining insertion order
* Thread-safe maps: Hashtable and ConcurrentHashMap

### 7. **SortedSet and SortedMap**

* SortedSet interface and TreeSet
* SortedMap interface and TreeMap
* Key methods: first(), last(), headSet(), tailSet(), subSet()
* NavigableSet and NavigableMap: lower(), higher(), ceiling(), floor()

### 8. **Iterator and ListIterator**

* Introduction to Iterator interface
* Methods of Iterator: hasNext(), next(), remove()
* ListIterator methods: add(), set(), previous(), hasPrevious()
* Fail-fast vs Fail-safe iterators

### 9. **Comparable and Comparator Interfaces**

* Comparable interface: Natural ordering
* Comparator interface: Custom sorting logic
* Sorting a collection using Collections.sort()
* Usage of Comparator for sorting complex objects

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 13-01-2025 | Monday | Tech Stack | HTML, CSS and JS |  |  |  |
| 15-01-2025 | Wednesday | Tech Stack | HTML, CSS and JS |  |  |  |

1. **Basic Structure of HTML**
   * <html>, <head>, <body>
   * HTML document structure
   * DOCTYPE declaration
   * Character encoding (<meta charset="UTF-8">)
2. **HTML Elements and Tags**
   * Block-level vs inline elements
   * Common tags: <div>, <span>, <p>, <h1>, <h2>, <ul>, <ol>, <li>, <a>, <img>, <form>
   * Self-closing tags (e.g., <img>, <br>, <input>)
   * Attributes: id, class, src, href, **alt**, style, target
3. **HTML Forms**
   * Form elements: <form>, <input>, <select>, <textarea>, <button>
   * Form attributes: action, method, enctype
   * Input types: text, password, email, checkbox, radio, submit
   * Form validation: required, pattern, min, max, step
4. **HTML Links and Navigation**
   * Hyperlinks: <a> tag, href attribute
   * Navigation bar (navbar) construction
   * Links to other pages, email, and external sites
   * Anchor links and # anchor navigation
5. **HTML Media Elements**
   * Images: <img>, src, alt, width, height
   * Audio and Video: <audio>, <video>, controls, autoplay, loop, muted
   * Embedding media: <iframe>, <embed>, <object>
6. **HTML Tables**
   * Table structure: <table>, <tr>, <td>, <th>, <caption>
   * Table attributes: border, cellspacing, cellpadding
   * Merging cells: colspan, rowspan
7. **HTML5 Features**
   * Semantic elements: <header>, <footer>, <article>, <section>, <nav>, <aside>, <main>
   * Local Storage and Session Storage

### **CSS Topics**

1. **CSS Syntax and Selectors**
   * Basic syntax: selector { property: value; }
   * Types of selectors: ID, class, type, attribute, descendant, child, adjacent
   * Pseudo-classes (:hover, :focus, :nth-child(), :first-child)
   * Pseudo-elements (::before, ::after, ::first-letter)
2. **CSS Box Model**
   * Components of the box model: content, padding, border, margin
   * box-sizing property: content-box vs border-box
   * Adjusting width/height with width, height, padding, border, and margin
3. **Positioning in CSS**
   * Static, relative, absolute, fixed, and sticky positioning
   * top, right, bottom, left, z-index
   * position: absolute vs position: fixed
4. **CSS Layout Techniques**
   * Flexbox: display: flex, justify-content, align-items, flex-direction, flex-wrap
   * CSS Grid: display: grid, grid-template-columns, grid-template-rows, grid-gap
   * Float-based layouts (legacy method)
   * Responsive design with @media queries
5. **CSS Styling**
   * Colors: color, background-color, rgba(), hsl(), gradients
   * Fonts: font-family, font-size, font-weight, font-style
   * Text styles: text-align, line-height, text-transform, text-decoration, letter-spacing
   * Border styles: border, border-radius, border-width, border-color
   * Shadows: box-shadow, text-shadow

### **JavaScript Topics**

1. **Basic JavaScript Syntax**
   * Variables: var, let, const
   * Data types: String, Number, Boolean, Object, Array, Null, Undefined
   * Operators: +, -, \*, /, %, comparison operators, logical operators
   * Control flow: if, else, else if, switch, ternary operator
   * Loops: for, while, do...while, forEach
2. **Functions**
   * Function declaration: function myFunction() {}
   * Function expression
   * Arrow functions: const myFunc = () => {}
   * Parameters, arguments, and return values
   * IIFE (Immediately Invoked Function Expression)
3. **Objects and Arrays**
   * Object creation and manipulation: {}, object methods, properties
   * Arrays: [], accessing elements, push(), pop(), shift(), unshift(), splice(), slice()
   * Array iteration: forEach(), map(), filter(), reduce()
   * Destructuring assignment for objects and arrays
4. **DOM Manipulation**
   * Accessing DOM elements: getElementById(), getElementsByClassName(), querySelector(), querySelectorAll()
   * Modifying DOM elements: innerHTML, textContent, setAttribute(), classList
   * Event handling: addEventListener(), click, submit, keydown, etc.
   * Creating and deleting DOM elements dynamically: createElement(), appendChild(), removeChild()

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 16-01-2025 | Thursday | Tech Stack | J2EE, Security |  |  |  |
| 17-01-2025 | Friday | Tech Stack | J2EE, Security |  |  |  |

### **J2EE (Java EE) Topics**

#### 1. **Overview of Java EE (Enterprise Edition)**

* Difference between Java SE (Standard Edition) and Java EE
* Components of Java EE architecture: Web tier, Business tier, and Enterprise Information Systems tier
* Java EE containers: Web container, EJB container, and Application server
* Overview of J2EE technologies: Servlets, JSP, EJB, JPA, JMS, and more

#### 2. **Servlets**

* Introduction to Servlets
* Servlet lifecycle: init(), service(), destroy()
* Creating and deploying a Servlet
* ServletConfig and ServletContext
* Request and Response objects: HttpServletRequest, HttpServletResponse
* Handling HTTP methods (GET, POST, PUT, DELETE)
* Session management: Cookies, HttpSession
* Servlet Filters: Purpose and implementation

#### 3. **JavaServer Pages (JSP)**

* JSP lifecycle: Translation, compilation, execution
* JSP tags: Standard tags, Custom tags, JSTL (JavaServer Pages Standard Tag Library)
* Expression Language (EL) in JSP
* JSP vs Servlets: Differences and use cases
* Tag libraries in JSP: jsp:useBean, jsp:setProperty, jsp:getProperty

### **Java Security Topics**

#### 1. **Overview of Java Security**

* Java security architecture
* Security manager and access control
* The java.security package and security policies

#### 2. **Authentication in Java**

* Java Authentication and Authorization Service (JAAS)
* Managing user authentication with LoginModules
* Single Sign-On (SSO) in Java-based systems
* OAuth and OpenID for third-party authentication

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 20-01-2025 | Monday | Tech Stack | Spring, Spring Boot, Microservices |  |  |  |
| 21-01-2025 | Tuesday | Tech Stack | Spring, Spring Boot, Microservices |  |  |  |

Spring Introduction

* *Shortcomings of Java EE and the Need for Loose Coupling*
* *Managing Beans, The Spring Container, Inversion of Control*
* *The Factory Pattern*
* *Configuration Metadata - XML, @Component, Auto-Detecting Beans*
* *Dependencies and Dependency Injection (DI) with the BeanFactory*
* *Setter Injection*

 Dependency Injection

* *Using the Application Context*
* *Constructor Injection*
* *PropertyEditors*
* *Factory Methods*
* *Crucial Namespaces ‘p’ and ’c’*
* *Configuring Collections*
* *Bean Definition Inheritance and Collection Merging*
* *Expression Languages, SpEL*

**Spring Boot**

Spring Boot Framework – High Level Overview

* *Spring Architecture*
* *Spring Containers*
* *Spring Bean Lifecycle*
* *Spring DI*
* *Spring Autowiring*

Spring Boot – Database Integration

* *Spring Boot – JDBC*
* *Spring Boot – JPA*
* *Spring Boot – Data*

Introduction to Actuator

* *Microservices*
* *Introduction to Microservices*
* *What are Microservices?*
* *Decentralized Governance, Scalability, Fault Tolerance*
* *Cloud Computing*
* *Spring Cloud*
* *Service and Client Discovery*
* *Netflix OSS*
* *Netflix OSS*
* *Service Discovery*
* *Eureka Servers and Clients*
* *Load Balancing with Ribbon using Eureka*
* *Circuit Breakers*
* *When services fail*
* *Circuit Breakers – Hystrix*

|  |  |  |  |
| --- | --- | --- | --- |
| 22-01-2025 | Wednesday | Tech Stack | API, API First |

### **API (Application Programming Interface) Topics**

#### 1. **Introduction to APIs**

* What is an API?
* Types of APIs:
  + **Web APIs** (REST, SOAP, GraphQL, gRPC)
  + **Library/Framework APIs**
  + **Operating System APIs**
  + **Database APIs**
* API request and response structure
* JSON and XML formats for API communication
* API versioning: URI versioning, header versioning, query parameter versioning
* Public vs Private APIs

#### 2. **RESTful APIs (Representational State Transfer)**

* Principles of REST architecture
* RESTful constraints: Statelessness, Client-Server architecture, Uniform Interface, Cacheable responses, Layered system, Code on Demand
* HTTP methods and RESTful operations:
  + GET (retrieve data)
  + POST (create data)
  + PUT (update data)
  + DELETE (delete data)
  + PATCH (partial update)
* Resource identification (URLs) in REST APIs
* Status Codes: 2xx (success), 4xx (client errors), 5xx (server errors)
* Statelessness in REST: Each request must contain all information for the server to process it.

### **API-First Development Topics**

#### 1. **What is API-First Development?**

* API-First philosophy: Design API before implementation
* Benefits of API-First approach:
  + Clear contract between front-end and back-end teams
  + Easier integration with third-party services
  + Faster development cycles
* Differences between traditional development and API-First development

#### 2. **Designing APIs First**

* API design principles and best practices
* Designing RESTful APIs vs GraphQL APIs vs SOAP APIs
* Using OpenAPI Specification (OAS) for API design
* Creating a clear and consistent API contract
* Documentation-first approach: Why it's important to have good API documentation before development

#### 3. **API Design Tools**

* Postman for designing and testing APIs

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 23-01-2025 | Thursday | Tech Stack | Node JS |  |  |  |
| 24-01-2025 | Friday | Tech Stack | React JS |  |  |  |

### **Core Node.js Topics**

#### 1. **Introduction to Node.js**

* What is Node.js? Overview and architecture
* Node.js event-driven, non-blocking I/O model
* Single-threaded vs multi-threaded processing
* Benefits and use cases of Node.js
* Installing Node.js and npm (Node Package Manager)
* Running a simple "Hello, World!" application in Node.js

#### 2. **Node.js Architecture**

* Understanding the Node.js runtime
* Node.js Event Loop and its phases
* Non-blocking I/O and asynchronous operations
* Callbacks and Promises
* The Node.js process object: process.env, process.argv, etc.
* Understanding setImmediate(), setTimeout(), and setInterval()

#### 3. **Node.js Modules**

* Built-in modules in Node.js (e.g., http, fs, path, url, querystring, events, etc.)
* Creating custom modules with module.exports and require()
* Module caching and the require.cache object
* Node.js module system: CommonJS and ES6 import/export
* Using third-party modules from npm

#### 4. **HTTP Module**

* Creating a simple HTTP server with the http module
* Handling GET and POST requests
* Request and response objects in HTTP
* HTTP status codes and headers
* Routing requests in HTTP servers
* Handling JSON and form data in requests

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 27-01-2025 | Monday | Tech Stack | Angular JS |  |  |  |
| 28-01-2025 | Tuesday | Tech Stack | Angular JS |  |  |  |

#### 1. **Introduction to Angular**

* What is Angular? Overview and architecture
* Benefits of Angular as a front-end framework
* Angular CLI (Command Line Interface) for project scaffolding
* Single Page Applications (SPA) concept
* Angular vs other front-end frameworks (React, Vue.js)
* Setting up an Angular project using ng new
* Angular project structure

#### 2. **Components**

* What are Angular Components?
* Creating and using components
* Component lifecycle hooks (ngOnInit, ngOnChanges, ngOnDestroy, etc.)
* Templates, styles, and class binding in components
* Data binding in Angular (One-way binding, Two-way binding)
* Component communication: @Input, @Output, EventEmitter
* Nested components and component hierarchy

#### 3. **Directives**

* What are Directives in Angular?
* Structural directives: \*ngIf, \*ngFor, \*ngSwitch
* Attribute directives: ngClass, ngStyle, ngModel
* Custom directives: Creating and using custom directives
* Directive lifecycle hooks (ngOnInit, ngOnDestroy)

#### 4. **Services and Dependency Injection (DI)**

* What are Angular services?
* Creating and using services for logic, data handling, and HTTP requests
* Dependency Injection (DI) and the provider system in Angular
* Injecting services into components and other services
* Singleton pattern in Angular services
* Using Angular’s @Injectable() decorator

#### 5. **Routing**

* Introduction to Angular routing
* Setting up routing in Angular with RouterModule
* Configuring routes in app-routing.module.ts
* Route parameters (query and path parameters)
* Route guards: CanActivate, CanLoad, CanDeactivate
* Lazy loading modules for efficient routing
* Route resolving and pre-fetching data
* Navigating programmatically using Router service

#### 6. **Forms in Angular**

* Template-driven forms vs Reactive forms
* Creating template-driven forms using ngModel
* Validations in template-driven forms
* Reactive forms: FormControl, FormGroup, FormBuilder
* Form validation and custom validators in reactive forms
* Managing form control states (valid, invalid, touched, pristine, etc.)
* Dynamic form creation in Angular
* Form arrays in reactive forms

#### 7. **Pipes**

* What are pipes in Angular?
* Built-in pipes: date, currency, uppercase, lowercase, json, async
* Creating custom pipes
* Pure vs Impure pipes
* Using pipes in templates for transforming data

#### 8. **HTTP and Observables**

* Introduction to HTTP in Angular with HttpClientModule
* Making HTTP requests with HttpClient service (GET, POST, PUT, DELETE)
* Handling response and error with RxJS operators (map, catchError, retry)
* Understanding Observables and RxJS in Angular
* Using RxJS operators (switchMap, mergeMap, forkJoin, etc.)
* Intercepting HTTP requests with HttpInterceptor
* Authentication and Authorization with HTTP headers and tokens (JWT)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 29-01-2025 | Wednesday | Tech Stack | Unit Testing & Code Coverage - Junit, Jacoco |  |  |  |
| 30-01-2025 | Thursday | Tech Stack | Testing basics concepts and NFR |  |  |  |

**Introduction to Testing**

* *Why Testing*
* *Overview of JUnit’s role in Testing*
* *Unit Testing with JUnit*
* *Installation and setup*

### Working with JUnit 5

* *Use of setUp() and tearDown()*
* *JUnit Assertions*
* *Static imports in Eclipse*
* *Executing test cases*

### Working with JUnit 4 continued...

* *Using @Test in JUnit4*
* *@BeforeClass and @AfterClass annotations*
* *TestFixtures with @Before and @After*
* *Ignoring Test Cases using @Ignore*
* *Testing Exceptions*
* *Testing Isolations*
* *Combining Test Cases as TestSuite*
* *Running Test Cases from main method via runner class*

Working with Jacoco

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 31-01-2025 | Friday | Tech Stack | Automation (Selenium & Rest API) |  |  |  |
| 03-02-2025 | Monday | Tech Stack | Automation (Selenium & Rest API) |  |  |  |

#### 1. **Introduction to Selenium**

* What is Selenium? Overview of Selenium suite
* History of Selenium and its evolution
* Different components of Selenium: Selenium WebDriver, Selenium IDE, Selenium Grid, and Selenium RC
* Benefits of using Selenium for automation
* Setting up Selenium WebDriver with different programming languages (Java, Python, C#, JavaScript, Ruby)
* Selenium architecture and its working principles

#### 2. **Selenium WebDriver Basics**

* Understanding WebDriver and its advantages over Selenium RC
* WebDriver vs Selenium IDE and Selenium Grid
* How WebDriver interacts with browsers
* Setting up WebDriver for different browsers: Chrome, Firefox, Safari, Edge, Internet Explorer
* Basic WebDriver commands: get(), getTitle(), getCurrentUrl(), getPageSource(), close(), quit()

#### 3. **Locators in Selenium**

* Introduction to WebDriver Locators
* Different types of locators:
  + ID, Name, Class Name, Tag Name, Link Text, Partial Link Text, XPath, CSS Selector
* Choosing the best locator for an element
* XPath vs CSS Selectors: Differences and use cases
* Relative vs Absolute XPath
* Handling dynamic elements with XPath and CSS Selectors

#### 4. **WebDriver Commands**

* Interacting with elements: click(), sendKeys(), clear()
* Handling checkboxes, radio buttons, and dropdowns
* Working with buttons, text fields, links, and other input elements
* Mouse actions: click(), doubleClick(), mouseMove(), dragAndDrop()
* Keyboard actions: sendKeys(), KeyEvents, handling special keys like Enter, Tab, Shift, Ctrl, etc.

#### 5. **Synchronization in Selenium**

* Implicit Wait: driver.manage().timeouts().implicitlyWait()
* Explicit Wait: WebDriverWait, ExpectedConditions

#### 6. **Handling Web Elements**

* Finding elements: findElement(), findElements()
* Handling dynamic elements and waiting for visibility
* Extracting data from web elements (e.g., getText(), getAttribute())
* Handling different types of web elements: buttons, links, forms, checkboxes, dropdowns, etc.

#### 7. **Page Object Model (POM)**

* Introduction to Page Object Model (POM) design pattern
* Benefits of POM in test automation (maintainability, reusability, readability)
* Best practices for Page Object Model in Selenium

1. ***Rest API Testing***