time table

Here's a **detailed time table** for learning to become a **Full Stack Java Developer** based on the timeline provided earlier. This time table assumes you dedicate **2-3 hours daily** for learning and practice. You can adjust the schedule based on your availability.

Week 1: Core Java Basics

Day	**Topic**	**Tasks**
Day 1	Introduction to Java	Install JDK, IDE, write "Hello World," learn Java basics.
Day 2	Variables, Data Types, Operators	Practice basic arithmetic, relational, and logical operators.
Day 3	Control Flow (if-else, switch)	Write programs using conditional statements.
Day 4	Loops (for, while, do-while)	Solve problems using loops.
Day 5	Arrays	Practice array manipulation (sorting, searching).
Day 6	Strings	Learn string methods and practice string manipulation.
Day 7	Practice Problems	Solve 5-7 problems on arrays and strings.

Week 2: Object-Oriented Programming (OOP)

Day	**Topic**	**Tasks**
Day 8	Classes and Objects	Create classes, define methods, and instantiate objects.
Day 9	Constructors	Practice creating constructors and using this keyword.
Day 10	Inheritance	Implement inheritance, use super keyword.
Day 11	Polymorphism	Practice method overriding and overloading.
Day 12	Encapsulation	Use access modifiers and create getters/setters.
Day 13	Abstraction (Abstract Classes, Interfaces)	Implement abstract classes and interfaces.
Day 14	Practice Problems	Solve 5-7 OOP-related problems.

Week 3: Advanced Java Concepts

Day	**Topic**	**Tasks**
Day 15	Exception Handling	Write programs with try-catch-finally.
Day 16	Collections Framework	Learn List, Set, and Map interfaces.
Day 17	Generics	Practice creating generic classes and methods.
Day 18	File Handling	Read/write files using FileReader, BufferedReader.
Day 19	Serialization	Serialize and deserialize objects.
Day 20	Practice Problems	Solve 5-7 problems on collections and file handling.

Week 4: Core Java Review and Mini Projects

Day	**Topic**	**Tasks**
Day 21	Review Core Java Concepts	Revise all topics covered in Weeks 1-3.
Day 22	Mini Project: Calculator	Build a simple calculator using Java.
Day 23	Mini Project: Student Management System	Create a basic student management system.
Day 24	Mini Project: Library Management System	Build a library management system.
Day 25	Practice Problems	Solve 5-7 advanced Java problems.

Week 5: Introduction to Back-End Development

Day	**Topic**	**Tasks**
Day 26	Overview of Back-End Development	Learn about RESTful APIs and HTTP methods.
Day 27	Introduction to Java EE	Learn about Servlets and JSP.
Day 28	Servlets Basics	Write a simple servlet to handle requests.
Day 29	JSP Basics	Create dynamic web pages using JSP.
Day 30	Practice Problems	Solve 3-5 problems on Servlets and JSP.

Week 6: Spring Framework

Day	**Topic**	**Tasks**
Day 31	Introduction to Spring Core	Learn Dependency Injection and Inversion of Control.
Day 32	Spring Boot Basics	Create a simple Spring Boot application.
Day 33	Spring MVC	Build a basic web application using Spring MVC.
Day 34	REST APIs with Spring Boot	Create RESTful APIs using Spring Boot.
Day 35	Practice Problems	Solve 3-5 problems on Spring Boot and REST APIs.

Week 7: Database Integration

Day	**Topic**	**Tasks**
Day 36	SQL Basics	Learn basic SQL queries (SELECT, INSERT, UPDATE, DELETE).
Day 37	JDBC	Connect Java applications to databases using JDBC.
Day 38	Hibernate Basics	Learn ORM concepts and implement Hibernate.
Day 39	Spring Data JPA	Use Spring Data JPA for database operations.
Day 40	Practice Problems	Solve 3-5 problems on database integration.

Week 8: Advanced Spring

Day	**Topic**	**Tasks**
Day 41	Spring Security	Implement authentication and authorization.
Day 42	REST API Security	Secure REST APIs using Spring Security.
Day 43	Caching with Redis	Implement caching in Spring applications.
Day 44	Testing with JUnit	Write unit tests for Spring applications.
Day 45	Practice Problems	Solve 3-5 problems on Spring Security and testing.

Week 9: Deployment and DevOps

Day	Topic	Tasks
Day 46	Deployment with Tomcat	Deploy a Spring Boot application on Tomcat.
Day 47	Docker Basics	Learn Docker basics and containerize a Java application.
Day 48	CI/CD Basics	Set up a CI/CD pipeline for a Java project.
Day 49	Practice Problems	Solve 3-5 problems on deployment and Docker.

Week 10: Back-End Review and Mini Projects

Day	Topic	Tasks
Day 50	Review Back-End Concepts	Revise all topics covered in Weeks 5-9.
Day 51	Mini Project: Task Management System	Build a task management system with Spring Boot.
Day 52	Mini Project: E-Commerce Back-End	Create a back-end for an e-commerce application.
Day 53	Mini Project: Chat Application	Build a real-time chat application.
Day 54	Practice Problems	Solve 5-7 advanced back-end problems.

Week 11: HTML and CSS

Day	Topic	Tasks
Day 55	HTML Basics	Learn HTML tags, forms, and tables.
Day 56	CSS Basics	Learn CSS selectors, layouts, and responsive design.
Day 57	Practice Problems	Build a simple static website.

Week 12: JavaScript

Day	**Topic**	**Tasks**
Day 58	JavaScript Basics	Learn variables, functions, and loops.
Day 59	DOM Manipulation	Practice DOM manipulation and event handling.
Day 60	AJAX Basics	Make asynchronous requests using AJAX.
Day 61	Practice Problems	Solve 3-5 problems on JavaScript and AJAX.

Week 13: Front-End Frameworks

Day	**Topic**	**Tasks**
Day 62	Introduction to React.js	Learn React basics (components, state, props).
Day 63	React Components	Build reusable components.
Day 64	React State and Props	Manage state and pass props between components.
Day 65	React Routing	Implement routing in a React application.
Day 66	Practice Problems	Solve 3-5 problems on React.

Week 14: Full Stack Integration

Day	**Topic**	**Tasks**
Day 67	Connecting Front-End and Back-End	Integrate React with Spring Boot APIs.
Day 68	Handling CORS	Configure CORS in Spring Boot.
Day 69	Debugging Full Stack Applications	Debug and test full-stack applications.
Day 70	Practice Problems	Solve 3-5 full-stack integration problems.

Week 15-24: Projects and Portfolio Building

Day	**Topic**	**Tasks**
Day 71-80	Mini Projects	Build small projects (e.g., e-commerce, blog, task manager).
Day 81-90	Intermediate Projects	Build intermediate projects (e.g., real-time chat, social media).
Day 91-100	Advanced Projects	Build a full-stack application (e.g., banking system, quiz app).
Day 101-120	Portfolio Building	Document and deploy projects, create a GitHub portfolio.

Additional Tips

- **Consistency**: Stick to the schedule and practice daily.
- **Hands-On Practice**: Write code for every concept you learn.
- **Use Online Resources**: Leverage platforms like YouTube, Udemy, and freeCodeCamp.
- **Join Communities**: Participate in forums like Stack Overflow or Reddit.

By following this time table, you'll be well-prepared to work as a Full Stack Java Developer within **6 months**.

practice questions

Here's a comprehensive list of **practice questions** for each phase of learning to become a **Full Stack Java Developer**. These questions are categorized by topic and difficulty level to help you reinforce your understanding and build practical skills.

Core Java Practice Questions

Week 1: Basics

1. Write a program to print "Hello, World!" in Java.
2. Create a program to calculate the area of a rectangle using user input.
3. Write a program to check if a number is even or odd.
4. Create a program to find the largest of three numbers.
5. Write a program to convert Celsius to Fahrenheit.

Week 2: Control Flow and Loops

1. Write a program to print numbers from 1 to 10 using a for loop.
2. Create a program to print the multiplication table of a given number.
3. Write a program to find the factorial of a number using a while loop.
4. Create a program to check if a number is prime.
5. Write a program to reverse a number.

Week 3: Arrays and Strings

1. Write a program to find the largest element in an array.
2. Create a program to sort an array of integers in ascending order.
3. Write a program to count the number of vowels in a string.
4. Create a program to check if a string is a palindrome.
5. Write a program to find the second largest element in an array.

Week 4: OOP

1. Create a class Employee with attributes name, id, and salary. Implement methods to set and get these attributes.
2. Write a program to demonstrate method overloading by creating a class Calculator with methods to add two numbers and three numbers.
3. Create a class Vehicle and extend it to create a class Car. Override a method to print the type of vehicle.
4. Write a program to demonstrate encapsulation by creating a class BankAccount with private attributes and public getter/setter methods.
5. Create an abstract class Shape with an abstract method calculateArea(). Implement it in subclasses Circle and Rectangle.

Week 5: Advanced Java

1. Write a program to handle ArithmeticException when dividing two numbers.
2. Create a program to demonstrate the use of ArrayList to store and display a list of names.
3. Write a program to serialize and deserialize an object of a class Person.
4. Create a program to read a text file and count the number of words in it.
5. Write a program to demonstrate the use of HashMap to store and retrieve key-value pairs.

Back-End Practice Questions

Week 6: Java EE (Servlets and JSP)

1. Create a servlet to handle a GET request and display a welcome message.
2. Write a JSP page to display the current date and time.
3. Create a servlet to handle form data submitted via POST.
4. Write a program to demonstrate session management using cookies.
5. Create a JSP page to display a list of items fetched from a servlet.

Week 7: Spring Framework

1. Create a Spring Boot application to expose a REST API that returns a list of users.
2. Write a program to demonstrate dependency injection using Spring.
3. Create a Spring MVC application to display a form and handle form submission.
4. Write a program to demonstrate the use of @RestController in Spring Boot.
5. Create a Spring Boot application to connect to a MySQL database and fetch data.

Week 8: Database Integration

1. Write a program to insert, update, and delete records in a MySQL database using JDBC.
2. Create a Spring Boot application to perform CRUD operations on a User entity using JPA.
3. Write a program to demonstrate the use of Hibernate to map a Java class to a database table.
4. Create a Spring Data JPA repository to fetch data from a database.
5. Write a program to implement pagination and sorting in a Spring Boot application.

Week 9: Advanced Spring

1. Create a Spring Security configuration to secure a REST API.
2. Write a program to implement JWT-based authentication in a Spring Boot application.
3. Create a Spring Boot application to implement caching using Redis.
4. Write a program to test a Spring Boot application using JUnit.
5. Create a Spring Boot application to handle file uploads and downloads.

Week 10: Deployment and DevOps

1. Write a program to deploy a Spring Boot application on Tomcat.
2. Create a Docker container for a Spring Boot application.
3. Write a program to set up a CI/CD pipeline for a Java project using Jenkins.
4. Create a Spring Boot application and deploy it on AWS Elastic Beanstalk.
5. Write a program to demonstrate the use of Kubernetes to deploy a Spring Boot application.

Front-End Practice Questions

Week 11: HTML and CSS

1. Create a simple HTML page with a header, footer, and a form.
2. Write a CSS file to style a navigation bar with hover effects.
3. Create a responsive layout using CSS Flexbox.
4. Write a program to create a grid layout using CSS Grid.
5. Create a webpage with a background image and text overlay.

Week 12: JavaScript

1. Write a JavaScript function to calculate the sum of two numbers.
2. Create a program to validate a form using JavaScript.
3. Write a program to display the current date and time on a webpage.
4. Create a program to create a slideshow using JavaScript.
5. Write a program to fetch data from a REST API and display it on a webpage.

Week 13: Front-End Frameworks

1. Create a React component to display a list of items.
2. Write a program to implement routing in a React application.
3. Create a React application to manage state using useState.
4. Write a program to fetch data from a REST API and display it in a React component.
5. Create a React application to implement form validation.

Week 14: Full Stack Integration

1. Write a program to connect a React front-end with a Spring Boot back-end using REST APIs.
2. Create a full-stack application to manage a list of tasks.
3. Write a program to implement user authentication in a full-stack application.
4. Create a full-stack application to upload and display images.
5. Write a program to implement pagination in a full-stack application.

Projects

Mini Projects

1. Build a simple calculator using Java and Swing.
2. Create a student management system with basic CRUD operations.
3. Build a library management system using Java and MySQL.
4. Create a basic e-commerce back-end using Spring Boot.
5. Build a task management system using React and Spring Boot.

Intermediate Projects

1. Create a real-time chat application using WebSocket and Spring Boot.
2. Build a social media platform with user authentication and post management.
3. Create a blogging platform with CRUD operations for posts and comments.
4. Build a quiz application with multiple-choice questions and scoring.
5. Create a file-sharing application using Spring Boot and React.

Advanced Projects

1. Build a full-stack banking application with user authentication, account management, and transaction history.
2. Create a full-stack e-commerce platform with product listings, shopping cart, and payment integration.
3. Build a job portal with user profiles, job listings, and application tracking.
4. Create a real-time collaborative document editor using React, Spring Boot, and WebSocket.
5. Build a full-stack healthcare management system with patient records, appointment scheduling, and doctor profiles.

Additional Tips

- **Practice Platforms**: Use platforms like LeetCode, HackerRank, and CodeChef for coding challenges.
- **Version Control**: Use Git and GitHub to manage your projects.
- **Portfolio**: Document your projects and showcase them on GitHub and LinkedIn.

By solving these practice questions and building projects, you'll gain the skills and confidence needed to become a proficient Full Stack Java Developer.