**Day-1**

**1. Created a plan today for learning Data Structures and Algorithms (DSA) and full-stack development.**

**2. Committed to consistent hard work and smart efforts to support your preparation.**

**3. Encouraging you to start your journey today!**

**Day-2**

**Today, I covered my development tasks and also began preparing for Core Java. Like**

**What is Class?**

**How to Create Classes ?**

**Static functions in a Class**

**Program on classes & static members**

**1. Private Instance members & static members**

**2. Classes and Objects**

**3. What is Wrapper Classes in Java**

**List of Wrapper Predefined Classes**

**Wrapper classes Predefined functions**

**1. Wrapper class Program in Java**

**2. Command Line Arguments**

**Packages in Java**

**1. How to Create Packages in Java**

**2. Access Specifiers in Java**

**Constructors in Java**

**Types of Constructor in Java**

**Constructor Program in Java**

**1. Why there is no any Destructor in Java**

**2. Inheritance in Java**

**Types of Inheritance in Java**

**Inheritance Explained with Program**

**Constructor using Inheritance & SUPER**

**Day-3**

**Day 3/105, preparing for a 1 crore package. #nevergiveup**

**Today, I covered my development tasks and also began preparing for Core Java. Like**

**1. This Keyword & Super Keyword in Java**

**2. 1 Function Overloading & Function Overriding**

**What is Polymorphism in OOPs**

**Abstract Class in Java**

**Abstract Functions in Java**

**Abstract Class with Program in Java**

**1. Interface in Java | Abstract class vs Interface in Difference in JAVA**

**2. Interface Program in Java**

**Multithreading Explained in Hindi in Java**

**Implementation Types of Multithreading in**

**Java**

**1. Program Multithreading Implementation**

**2. Thread States in Multithreading in Java**

**Thread Priority in Multithreading in Java**

**Scanner class & Synchronization in**

**Multithreading(FULL Program)**

**Exception Handling in JAVA**

**Exception Classes in JAVA**

**What is the use of try, catch and throw block**

**Program**

**Day-4**

**Day 4/105, preparing for a 1 crore package.**

**Today, I have completed my development project. Additionally, I have learned Java topics such as...**

**1. Program Multithreading Implementation**

**2. Thread States in Multithreading in Java**

**Thread Priority in Multithreading in Java**

**Scanner class & Synchronization in**

**Multithreading (FULL Program)**

**Exception Handling in JAVA**

**Exception Classes in JAVA**

**What is the use of try, catch and throw block**

**Program on Exception Handling in Java**

**1. Hierarchy of Exception Classes in Java**

**2. Finally Keyword in Java**

**Checked Exception | Compile Time**

**Exception & Throw Keyword in Java**

**throws keyword in Java**

**Difference Between throw and throws Keyword**

**Day 4.5/105 preparing for 1 CR package.**

**This evening, I deepened my understanding of OOP concepts.**

**1. Classes and Objects**

**2. Encapsulation**

**3. Inheritance**

**4. Polymorphism**

**5. Abstraction**

**6. Interfaces**

**7. Abstract Classes**

**8. Method Overloading**

**9. Method Overriding**

**10. Constructor Overloading**

**11. Constructor Chaining**

**12. Access Modifiers (public, private, protected, default)**

**13. Association**

**14. Aggregation**

**15. Composition**

**Day-5**

**Day 5/105 preparing for 1cr package**

**1. Program Multithreading Implementation**

**2. Thread States in Multithreading in Java**

**Thread Priority in Multithreading in Java**

**Scanner class & Synchronization in**

**Multithreading(FULL Program)**

**Exception Handling in JAVA**

**Exception Classes in JAVA**

**What is the use of try, catch and throw block**

**Program on Exception Handling in Java**

**1. Hierarchy of Exception Classes in Java**

**2. Finally Keyword in Java**

**Checked Exception | Compile Time**

**Exception & Throw Keyword in Java**

**throws keyword in Java**

**Difference Between throw and throws**

**Keyword**

**Day-6**

**Day 6 of preparing for 1cr package**

**1. Reading from a File**

**2. Writing to a File**

**3. Buffered Reading and Writing**

**4. Character Streams vs. Byte Streams**

**5. File Input/Output with Scanner**

**6. File Input/Output with PrintWriter**

**7. File Navigation and Information**

**8. Creating and Deleting Files**

**9. Handling Exceptions in File I/O**

**10. Using try-with-resources for Auto-closing Files**

**Day-7**

**Yesterday, on the 6th day, I completed the entire Java. Now, my learning path includes:**

**Day 7: Preparation for roadmap to become full stack developer**

**1. Java back-end Development:**

**- Spring Framework**

**- Spring Microservice**

**- System Design**

**- Software Architecture**

**- Design Patterns**

**2. DevOps:**

**- Writing YAML files**

**- Docker**

**- Kubernetes**

**- Jenkins (CI/CD)**

**- Git commands**

**- AWS (Amazon Web Services)**

**3. Frontend:**

**- JavaScript**

**- HTML**

**- CSS**

**- React**

**This is my current roadmap for technologies learning. I recommend you start your learning journey as well.**

**Day-8**

**Day 8: Preparing for 1CR package.**

**To become a proficient software developer, it is essential to comprehend software development architecture. This includes understanding:**

**- The purpose of the architecture.**

**- The necessity for employing specific architectures.**

**- The reasons behind choosing particular architectural approaches.**

**- In-depth conceptual understanding.**

**These are the Architecture You should know :**

**1. Monolithic Architecture**

**2. Microservices Architecture**

**3. Client-Server Architecture**

**4. Model-View-Controller (MVC)**

**6. Event-Driven Architecture (EDA)**

**7. Layered Architecture**

**8. P2P (Peer-to-Peer) Architecture**

**9. Event Sourcing**

**10. Serverless Architecture**

**Regardless of whether you are a frontend or backend developer, using frameworks such as React or Spring, it is crucial for all developers to be well-versed in these architectural concepts.**

**Take the time today to learn and understand these architectures, as it is a key factor in achieving success in your endeavors.**

**Day-9**

**Day 9: Preparing for 1CR package.**

**On day 9, I have understood all the design patterns for software development and how to write the code. Learn how to design your code and other related concepts today.**

**1. Creational design pattern**

**a. Factory method**

**b. Abstract factory**

**c. Builder**

**d. Prototype**

**e. Singleton**

**2. Structural design pattern**

**a. Adaptor**

**b. Bridge**

**c. Composite**

**d. Decorator**

**e. Facade**

**f. Flyweight**

**g. Proxy**

**3. Behavioral design pattern**

**a. Chain of responsibility**

**b. Command**

**c. Interpreter**

**d. Iterator**

**e. Mediator**

**f. Memento**

**g. Observer**

**h. State**

**i. Strategy**

**j. Visitor**

**k. Template method.**

**Day-10**

**Day 10/105 of preparing for 1 CR package**

**Monolithic: Single, tightly integrated application.**

**Microservices: Independent, small services that work together through APIs.**

**Monolithic Application Components:**

**- Single codebase**

**- Unified database**

**- Centralized user interface**

**- Shared libraries and resources**

**Microservices Application Components:**

**- Distributed services**

**- Isolated databases per service**

**- Decentralized user interfaces**

**- Independent libraries and resources**

**Additionally Microservices have**

**1. Database per Microservice**

**2. Event Sourcing**

**3. CQRS**

**4. Saga**

**5. BFF**

**6. API Gateway**

**7. Strangler**

**8. Circuit Breaker**

**9. Externalized Configuration**

**10. Consumer-Driven Contract Tracing**

**Day-11**

**Day 11/105: Preparing for 1 CR package**

**Day 1/10: Preparing for Spring Framework (Backend Development)**

**Today, I started learning Spring Framework for backend development. I aim to complete this in the next 10 days.**

**Started with Spring MVC (Model-View-Controller)**

**There are four layers in the Spring MVC.**

**1. Controller Layer:**

**- Use: Receives and processes user requests, interacts with the Model layer, and selects the appropriate view to render the response.**

**2. Service Layer:**

**- Use: Contains business logic and acts as an intermediary between the Controller and the Data Access layer, facilitating reusable and modular code.**

**3. Repository (Data Access) Layer:**

**- Use: Manages data persistence and retrieval, interacts with databases or other data sources, providing an abstraction for the underlying storage.**

**4. Model Layer:**

**- Use: Represents the application’s data and business logic, ensuring a clean separation from the presentation and control layers for enhanced maintainability.**

**Day-12**

**Day 12/105 of preparing for a 1 crore package.**

**Day 2/10 of preparing for the Spring Framework.**

**Today, I have gained understanding in the concepts and implementation of Spring Data and Spring Security within the Spring Framework.**

**- Spring Data simplifies data access in Spring applications, providing a consistent approach to interact with various databases.**

**Spring Data:**

**1. Spring Data JPA**

**2. Spring Data MongoDB**

**3. Spring Data Redis**

**4. Spring Data JDBC**

**5. Spring Data Elasticsearch**

**6. Spring Data REST**

**7. Spring Data for Apache Cassandra**

**8. Spring Data for Apache Solr**

**9. Query methods and custom queries**

**10. Auditing and automatic timestamps**

**- Spring Security is a powerful authentication and access control framework for securing Spring-based applications.**

**Spring Security:**

**1. Core Security**

**2. Authentication and Authorization**

**3. Form-based and OAuth-based authentication**

**4. Role-based access control**

**5. CSRF protection**

**6. Session management**

**7. Method-level security**

**8. LDAP integration**

**9. Remember-Me functionality**

**Day-13**

**Day 13 of 105: Preparing for 1 CR package.**

**Day 3 of 10: Preparing for Spring framework (Back-End development).**

**Yesterday, I completed my Spring Boot.**

**Today, I learned about microservices architecture, exploring components such as Database per Microservice, Event Sourcing, CQRS, Saga, BFF, API Gateway, Strangler, Circuit Breaker, Externalized Configuration, and Consumer-Driven Contract Tracing.**

**Note: from day 15, will start all the components of architecture like what is that, working and implementing**

**Tomorrow, I’ll implement and create a fully-fledged backend application using all these components.**

**Day-14**

**Day 14 out of 105: Preparing for a 1 crore package.**

**Day 4 out of 10: Preparing for the Spring Framework.**

**3. Discussing microservices components in previous video.**

**4. Check the previous reel to see the microservices component.**

**“Now, let me briefly explain every component of microservices.”**

**1. Database per Microservice: Each microservice has its own dedicated database, enhancing independence and scalability.**

**2. Event Driven: System design where components communicate through events, enabling real-time updates and responsiveness.**

**3. API Gateway: Centralized entry point for handling API requests, streamlining communication and managing security.**

**4. Strangler: Gradual migration strategy, replacing parts of an old system with new features incrementally.**

**5. Circuit Breaker: Mechanism preventing system failures by temporarily blocking requests to failing services, ensuring stability.**

**6. Externalized Configuration: Storing configuration settings outside the code, promoting flexibility and easy adjustments.**

**7. Consumer-Driven Contract Tracing: Collaborative approach ensuring services meet expectations, enhancing communication and reliability.**

**day-15**

**Day 15/105 of preparing for a 1 crore package.**

**Day 5/10 of preparing for Spring Framework (backend development).**

**Today, I am discussing a crucial topic in Spring Boot called Spring Cloud.**

**1. API Gateway:**

**- Think of it as the bouncer at a club’s entrance.**

**- It manages who gets in, controls traffic, and handles various checks before letting requests enter the microservices world.**

**2. Ribbon:**

**- Imagine Ribbon as a friendly traffic cop.**

**- It helps your applications find the right microservice by directing them to available instances, ensuring a smooth flow of requests without overwhelming one service.**

**Day-16**

**Day 16/105 of preparing for 1 CR package.**

**Day 6/10 of preparing for Spring Framework (backend development).**

**Today, let’s talk about Resilience4J in the Spring Framework.**

**Resilience4j is like a superhero for your Spring app:**

**1. Circuit Breaker: Stops calling a broken part to avoid making things worse.**

**2. Retry: Keeps trying, so if something fails, it doesn’t give up right away.**

**3. Rate Limiter: Controls how often your app does things to prevent overload.**

**4. Bulkhead: Keeps problems in one part from spreading to the whole app.**

**5. TimeLimiter: Puts a time limit on tasks to avoid long delays.**

**Why use it?**

**- Avoids crashes: It helps your app stay up, even if some parts act up.**

**- Fixes issues quietly: Stops troubled areas from causing more trouble.**

**- Works smoothly with Spring: It’s like a sidekick that fits perfectly into your Spring app.**

**Day-17**

**Day 16/105 - Preparation for a 1 CR Package**

**Today, let’s discuss microservice communication.**

**First, let’s delve into Monolithic and microservices applications.**

**Regarding microservice communication, the Spring framework offers various modules.**

**1. RestTemplate:**

**- What it is: A traditional way to make HTTP requests in Spring.**

**- For beginners: It’s like a tool that helps your program talk to other programs on the web by sending and receiving data.**

**2. Feign Client:**

**- What it is: A friendlier way to talk to other services. You write an interface, and it does a lot of the communication work for you.**

**- For beginners: It’s like having a conversation using a script; you define how you want to talk, and it handles the details.**

**3. WebClient:**

**- What it is: A modern way to make web requests that’s good for handling many things at once.**

**- For beginners: Imagine sending messages to multiple friends simultaneously without waiting for each response; it’s good for doing many things at the same time.**

**4. Retrofit:**

**- What it is: Not specific to Spring, but widely used in Java. It helps you create a special type of friend for talking to web services, making sure everything fits together nicely.**

**- For beginners: Like having a special friend who speaks the language of web services fluently; you just tell them what you need, and they take care of the rest.**

**5. OkHttp:**

**- What it is: Another general-purpose tool for making web requests in Java, often used with Spring.**

**- For beginners: Think of it as a reliable messenger that ensures your messages (HTTP requests) reach their destination safely.**

**Each of these tools simplifies the process of communication between different parts of your application or between different applications in a microservices setup.**

**Day 18/105 of preparing for a 1 CR package.**

**Today, let’s discuss message queues like Kafka.**

**Why do we use these KAFKA in development?**

**Use Kafka when you want different parts of your system to talk in real-time and share information smoothly.**

**- Think of Kafka for fast, ongoing updates**

**- Module like RestTemple, web client etc for straightforward, point-to-point talks.**

**Let’s take a real-life scenario where Kafka is in action.**

**Imagine you order food online. The restaurant uses Kafka to notify the kitchen about your order instantly.**

**Each step (order received, cooking, ready for delivery) is like a message in Kafka, ensuring real-time updates.**

**This way, everyone involved is on the same page without constant manual checks, making the process smooth and efficient.**

**Day-19**

**Day 19/105 of preparing for 1Cr package**

**Security is very important role for our application**

**So Today, lets discuss about spring security**

**There are two types of security**

**1. Authentication: Spring Security checks if users are who they say they are.**

**2. Authorization: It decides what specific things a user can do based on their roles, making sure they don’t do anything they’re not allowed to.**

**- There are many types of Spring Security. I will tell you about the most important and commonly used security measures.**

**1. Basic Authentication: Simple username-password validation for secure access.**

**2. JWT (JSON Web Tokens): Uses compact, self-contained tokens to authenticate and authorize users securely.**

**3. OAuth 2.0: Standardized framework for secure third-party access to user resources without exposing credentials.**

**4. HTTPS (HTTP Secure): Encrypts data exchanged between client and server, ensuring a secure communication channel.**

**5. CSRF Protection: Guards against unauthorized requests by validating their origin, enhancing web security.**

**6. Session Management: Controls user sessions, handling aspects like creation and expiration to improve security.**

**7. Role-Based Access Control (RBAC): Assigns permissions to users based on roles, ensuring proper access control within an application.**

**day-20**

**Day 20/105 of Preparing for 1CR Package**

**Today, let’s discuss a very important topic in the Spring Framework: Eureka Server or Eureka Discovery.**

**What is it? Why do we use it?**

**Eureka service registers all the information about other services so that one service can find another service easily.**

**Now Read this - you will understand now**

**Imagine service discovery as a game of hide-and-seek.**

**Each player (service) hides in a different room (server), and Eureka is like the referee who keeps track of where everyone is hiding.**

**So, when one player needs to find another, they just ask the referee, and they point them in the right direction, making it easy to find each other and continue playing together.**

**Eureka server provides:**

**1. Service Registration: Services register themselves with Eureka server.**

**2. Service Discovery: Services find and locate each other through Eureka.**

**3. Health Monitoring: Eureka monitors the health of registered services.**

**4. Load Balancing: Eureka directs client requests to available service instances.**

**5. Failover and Redundancy: Eureka redirects traffic to backup instances if primary ones fail.**

**6. Metadata Management: Eureka supports attaching metadata to services.**

**7. Dynamic Configuration: Eureka enables dynamic updates to service configurations.**

**Note: will explain you each topic in further videos**

**Day 21/105 of Preparing for a 1 CR Package:**

**Today, let’s discuss balancers and API gateways, External Configuration.**

**1. Load Balancer in Spring: It evenly distributes website or app visitors across multiple servers to keep things running smoothly and fast.**

**Dependency - Use Spring Cloud LoadBalancer**

**2. API Gateway in Spring: It’s like a main entrance to your system, managing and securing communication between different services so clients can access functionalities easily.**

**Dependency - Use Spring Cloud Gateway**

**3. External Configuration in Spring:It storing settings outside your code using Spring Cloud Config. It helps manage and update settings without code changes.**

**Dependency - Spring Cloud Config**

**4. Distributed tracing: In Spring lets you track how requests move through your system using tools like Spring Cloud Sleuth and Zipkin or Jaeger. It helps debug issues and monitor performance across multiple services.”**

**Dependency - spring cloud sleuth or Zipkin or Jaeger**

**Note: Sleuth and zipkin are widely used**

**Day 22/105 of Preparing for 1 CR Package:**

**Note: I Will give you the implementation in future #due to some problems i will not …**

**Today, let’s discuss Spring MVC And AOP.**

**———Spring MVC———**

**1. Model:**

**- Represents the data of the application.**

**- It’s typically a POJO (Plain Old Java Object) that holds the application’s business logic and data.**

**- Examples include classes that represent entities in your application, such as User, Product, etc.**

**2. View:**

**- Represents the presentation layer of the application.**

**- It’s responsible for rendering the user interface.**

**- In Spring MVC, views are typically JSP (JavaServer Pages), Thymeleaf templates, or any other presentation technology supported by Spring.**

**3. Controller:**

**- Acts as an intermediary between the Model and the View.**

**- Handles user requests, processes them, and returns an appropriate response.**

**- In Spring MVC, controllers are typically Java classes annotated with `@Controller`**

**—————————————————————————————-**

**These are the basic components we need to create an application.**

**If you require additional functionality, you can incorporate elements such as events, configuration, etc.**

**- Entity: Represents a real-world object or concept in the application, often mapped to a database table.**

**- DTO (Data Transfer Object): A simple container used to transfer data between different layers of an application.**

**- Repository: Manages the data access logic and acts as a bridge between the application logic and the database.**

**- Service: Contains business logic and performs operations that are not tied to any specific UI or data access concerns.**

**- Controller: Handles incoming requests, orchestrates the flow of data, and invokes appropriate actions to fulfill client requests in a web application.**

**——- AOP——-**

**AOP in Spring lets you add extra functionality, like logging or security checks, to your code without mixing it with your main logic. It helps keep your code clean and organized by separating these concerns into modules called aspects.**

**Day-23**

**Day 22/105 of preparing for 1 CR package:**

**Today, I have studied three design patterns that are applied in every project, whether you are working at a service-based or product-based company.**

**Note: To all freshers and less experienced individuals, remember these patterns.**

**1. Enrich Pattern: It’s like adding new features to a smartphone app without messing up the existing ones.**

**2. Chain of Responsibility:Imagine a customer service hotline. If the first person can’t help, they transfer you to someone else until your issue gets resolved.**

**3. Breaking a Project into Modules: Think of it as organizing a messy closet. You separate your clothes, shoes, and accessories into different bins, so you can find things easily and keep everything tidy.**

**4. Singleton Pattern: Ensures there’s only one instance of a class, like having a single key to access a room in a building, making it easy to manage resources across the project.**

**day-24**

[endless\_success\_\_\_](https://www.instagram.com/endless_success___/)

Coding Point :  
———————  
  
Full Java ( Advance + Core ): https://youtu.be/32DLasxoOiM?si=LdoAWv9kASowiGyI  
  
Full Java ( Advance + Core ): https://youtube.com/playlist?list=PL0zysOflRCekkNEe4FDmdUN3FATodg3Yf&si=zpqEp-\_TGxGsmQXd  
  
DSA -> https://youtu.be/h3uDCJ5mvgw?si=m4kC3gIpPsx6\_s6L  
  
DSA topic like Dynamic Programming , Sliding Window : https://www.youtube.com/@TheAdityaVerma/playlists  
  
Development  
———————  
  
MYSql DataBase: https://youtu.be/5OdVJbNCSso?si=STTWJf1mDepPlfaq  
  
SpringBoot Backend: https://youtu.be/Uh-N\_6Lccr4?si=IHrLQycjKwxKTtRi  
  
Spring micro services: https://youtube.com/playlist?list=PLSVW22jAG8pBnhAdq9S8BpLnZ0\_jVBj0c&si=ZkezZ3gHP968dCRN  
  
Spring doc: Better more prefer spring official documents: https://spring.io  
  
Integration & Cloud :  
——————————-  
  
Docker: https://youtu.be/e3YERpG2rMs?si=vxr\_YJ2bFzkIUX2o  
  
kubernetes: https://youtu.be/gMmcRbd8L5Y?si=E130afRntf69tQTj  
  
Git: https://youtu.be/CqSnEiKnDpA?si=rkzdMrLg0oz5yBts  
  
Aws: https://youtube.com/playlist?list=PL6XT0grm\_TfgtwtwUit305qS-HhDvb4du&si=oalMIUM6ZKqYWsev