🗁 Detailed structure and content of JAVA/

```
JAVA/
├─ 01-platform/
├─ 02-language/
├─ 03-oop/
├─ 04-collections-iterators/
├─ 05-functional-streams/
├─ 06-data-access/
├─ 07-testing/
├─ 08-http-apis/
├─ 09-spring/
└─ 10-architecture/
└─ 11-patterns/
...
```

📌 01-platform/ → Ecosystem and tools

- Here goes everything related to **how Java works**, what it is composed of, and **which tools we work with.**
  - JVM, JRE, JDK (differences, roles).
  - Compilation process (javac → bytecode → execution with java).
  - JDK tools (javac, java, jar, javadoc, jshell).
  - Java versions and modules (since Java 9, java.base, java.sql).
  - Dependency management: Maven and Gradle (project structure, pom.xml, lifecycle).
  - IDEs and environment setup (IntelliJ, Eclipse, VS Code).
  - Documentation (javadoc).
- 👉 Example file: platform/jvm-jdk.md, platform/maven.md.

📌 02-language/ → Pure language (Java SE)

- Here goes everything that is **syntax**, **rules**, and **construction** of the language.
  - Basic syntax: primitive types, operators, variables, constants.
  - Control flow: if, switch, for, while, do-while.
  - Methods and overloading.
  - Modifiers (public, private, protected, static, final, abstract).
  - Strings and wrappers (Integer, Double, Boolean).
  - Exception handling: try/catch/finally, throw, throws, checked vs unchecked.
  - Inner and anonymous classes (when not in the context of collections).
- 👉 Example file: language/basic-syntax.md, language/exceptions.md.

📌 03-oop/ → Object-Oriented Programming

- Everything related to modeling with **objects** and **classes**.
  - OOP concepts: class, object, encapsulation.
  - Inheritance and composition.
  - Polymorphism.

    - Interfaces and abstract classes.
    - SOLID principles (introduction).
    - Lombok (as a boilerplate simplifier).
- 👉 Example file: oop/inheritance-polymorphism.md, oop/interfaces.md.

📌 04-collections-iterators/ → Collections and data structures

- Here goes the **Java Collections Framework** and **data manipulation**.
    - Arrays vs Collections.
    - List, Set, Map.
    - Iterators (Iterator, Iterable, for-each).
    - Inner and anonymous classes applied to collections.
    - Comparators and sorting.
    - Implement your own collection by extending Collection or List.
- 👉 Example file: collections/list-set-map.md, collections/iterators.md.

📌 05-functional-streams/ → Functional and Streams

- Here goes everything **new from modern Java** (>= 8).
    - Lambda expressions.
    - Functional interfaces (Predicate, Function, Consumer, Supplier).
    - Streams API: map, filter, reduce, collect.
    - Optional.
    - Functional programming in Java (what to do and what not to do).
    - File processing (e.g., CSV with streams).
- 👉 Example file: functional/streams.md, functional/lambdas.md.

📌 06-data-access/ → Data access

- Everything related to **persistence and database connection.**
    - JDBC (drivers, Connection, Statement, ResultSet).
    - JPA (Java Persistence API).
    - Hibernate (most used ORM).
    - 1-1, 1-N, N-N relationships.
    - Transactions and EntityManager.
- 👉 Example file: data-access/jdbc.md, data-access/jpa.md.

📌 07-testing/ → Testing

- Here goes everything related to **Testing and code quality.**
    - JUnit (basic tests).
    - Test lifecycle (@BeforeEach, @AfterEach).
    - Assertions.
    - Mockito (mocks, stubs).
    - TDD (Test Driven Development).
- 👉 Example file: testing/junit.md, testing/mockito.md.

📌 08-http-apis/ → HTTP and API design

- Everything related to **REST, HTTP and documentation.**
  - HTTP: methods (GET, POST, PUT, DELETE), headers, status codes.
  - REST principles: resources, URIs, versioning.
  - API design (best practices).
  - OpenAPI/Swagger (automatic documentation).
- 👉 Example file: http-apis/http-basics.md, http-apis/rest-design.md.

📌 09-spring/ → Spring and Spring Boot

- Here goes everything related to the **framework**.
  - Spring Core: IoC (Inversion of Control), DI (Dependency Injection).
  - Spring Boot: auto-configuration, starters, application.yml.
  - Spring MVC: controllers, REST endpoints, validation.
  - Spring Data JPA: repositories, queries.
  - Spring Security: authentication/authorization, JWT.
  - Testing with Spring Boot.
- 👉 Example file: spring/spring-core.md, spring/spring-rest.md, spring/spring-data.md.

📌 10-architecture/ → Architecture (ONLY THE NECESSARY because is topic of DSI)(Full microservices here)

- Everything related to **software architecture** and **application structure**.
  - Layered architecture (Controller → Service → Repository).
  - Hexagonal / Ports and Adapters.
  - Microservices vs Monoliths.
  - DTOs, Mappers, Entities.
  - Architectural diagrams and 4+1 views.
  - Quality attributes (availability, scalability, maintainability, security).
- 👉 Example file: architecture/layers.md, architecture/microservices.md, architecture/quality-attributes.md.

📌 11-patterns/ → Design patterns

- Everything related to **design patterns** (reusable solutions at the code level).
  - Creational patterns: Factory, Builder, Singleton, Prototype.
  - Structural patterns: Adapter, Decorator, Composite, Proxy.
  - Behavioral patterns: Strategy, State, Observer, Iterator, Command.
  - Domain-driven patterns: Repository, Aggregate, Value Object, Entity.
- 👉 Example file: patterns/strategy.md, patterns/observer.md, patterns/factory.md.