Logging – Basics

- Logging is **mandatory** for debugging and production support.
- `System.out.println()` is **bad practice**:
  - Hard to remove later
  - Pollutes production logs
  - No control (enable/disable, level, destination)
- Logging can be **turned ON/OFF** and redirected to files, DB, email, etc.

What is Log4j?

- A **logging/tracing framework** mainly for **production environments**.
- Captures:
  - Info, warnings, errors, fatal issues
  - Class name, method, line number, date/time
- Logs can go to:
  - Console (temporary)
  - File (`.log`)
  - Database, Email, Network, etc.

Logging Frameworks

Common options:

- Log4j
- Logback
- Commons Logging
- `java.util.logging`

Best Practice
👉 Use **SLF4J (Simple Logging Facade for Java)**

```
Your Code → SLF4J → Log4j / Logback / JUL
```

Why?

- Decouples code from logging implementation

- Swap logging frameworks without changing code

Logging Levels (Priority Order)

Level	Meaning
ALL	Log everything
DEBUG	Detailed debugging info
INFO	Normal flow messages
WARN	Potential issues
ERROR	Exceptions & failures
FATAL	System-breaking issues
OFF	Disable logging

Log4j Architecture (Very Important)

`Logger → Appender → Layout`

1. Logger

- Created **per class**
- Enables logging for that class
- Methods:
  - `debug()`
  - `info()`
  - `warn()`
  - `error()`
  - `fatal()`

---

2. Appender (Where to log?)

Defines **destination**:

- Console
- File
- Database
- Email
- Network

Common appenders:

- `ConsoleAppender`
- `FileAppender`
- `JdbcAppender`
- `SmtpAppender`

---

3. Layout (How to log?)

Defines **format** of message.

Types:

- SimpleLayout
- HTMLLayout
- XMLLayout
- **PatternLayout (most used)**

Example pattern:

```
%d %p %C %M %m %n
```

Common pattern symbols:

- `%d` → date/time
- `%p` → log level
- `%C` → class name
- `%M` → method name
- `%L` → line number
- `%m` → message
- `%n` → new line

---

log4j.properties

- Central configuration file
- Uses `key=value` format
- Order:
    1. `rootLogger`
    2. Appenders
    3. Layouts

Key rules:

- Can define **multiple appenders**
- Every appender **must have a layout**
- File location:
    - `src/` (normal project)
    - `src/main/resources/` (Maven)
- `rootLogger=OFF` → disables logging **without deleting code**

Log4j with SLF4J

Required dependencies:

- `slf4j-api`
- `slf4j-log4j12`

Usage:

```
private static final Logger logger =
        LoggerFactory.getLogger(MyClass.class);

logger.info("Start processing");
logger.error("Something failed", ex);
```

Key Takeaways (Interview Gold)

- Never use `System.out.println()` in production
- Always use **SLF4J + Log4j**
- Logger = **who logs**
- Appender = **where it logs**
- Layout = **how it looks**
- Logging level controls **verbosity**, not code removal