**30-Day Java 8+ Project-Based Plan**

*(For Industry-Ready Developers)*

**Week 1: Functional Programming & Lambdas**

| **Day** | **Topic** | **Project Task** | **Outcome** |
| --- | --- | --- | --- |
| 1 | Lambda Basics | Convert 5 anonymous classes (Runnable, Comparator) → lambdas | Understand lambda syntax |
| 2 | Functional Interfaces | Build a StringProcessor (Predicate: filter, Function: map, Consumer: print) | Use java.util.function interfaces |
| 3 | Method References | Refactor lambdas → method refs (e.g., System.out::println) | Improve code readability |
| 4 | Comparator Chains | Sort a list of employees by salary, then name (Comparator.comparing) | Chain comparators fluently |
| 5 | Project: Task Manager | Implement task CRUD with lambdas (add, sort, filter, execute in threads) | Apply Week 1 concepts in one project |

**Week 2: Streams API & Data Processing**

| **Day** | **Topic** | **Project Task** | **Outcome** |
| --- | --- | --- | --- |
| 6 | Stream Basics | Filter/transform a list of products (e.g., price > $100 → uppercase names) | Master filter() + map() |
| 7 | Terminal Operations | Calculate stats (sum, avg, max) of sales data using reduce() & collect() | Aggregate data effectively |
| 8 | Collectors | Group employees by department + calculate avg salary per dept | SQL-like operations in Java |
| 9 | Parallel Streams | Process 10K records (compare stream() vs parallelStream() speed) | Understand parallelism tradeoffs |
| 10 | Project: Employee Analytics | Build a dashboard (filter, group, aggregate employee data) | End-to-end data pipeline |

**Week 3: Advanced Java 8 Features**

| **Day** | **Topic** | **Project Task** | **Outcome** |
| --- | --- | --- | --- |
| 11 | Optional | Refactor a null-prone user lookup system → use Optional | Eliminate NullPointerException |
| 12 | Date/Time API | Build a "Days Until Birthday" calculator (LocalDate, Period) | Modern date handling |
| 13 | File Processing | Read a log file → count errors, find top 5 frequent messages (Streams + NIO) | Process I/O functionally |
| 14 | Project: Birthday Reminder | CLI app to track birthdays + send reminders (Optional for missing data) | Combine Week 3 concepts |

**Week 4: Real-World Integrations**

| **Day** | **Topic** | **Project Task** | **Outcome** |
| --- | --- | --- | --- |
| 15 | CompletableFuture | Fetch weather + stock prices from mock APIs → combine results async | Async programming |
| 16 | Database + Streams | Query a DB (JDBC/Hibernate) → process results with Streams | Bridge SQL + Java 8 |
| 17 | Project: Stock Analyzer | Fetch stock prices → compute trends (Streams + CompletableFuture) | Real-world async data pipeline |

**Key Principles:**

1. **Daily Projects**: Each day ends with a mini-task applying the concept.
2. **Weekly Projects**: Consolidate learning into a larger project (e.g., Employee Analytics).
3. **Progressive Complexity**: Start with lambdas → Streams → async (CompletableFuture).
4. **Industry-Aligned**: Focus on data processing, async ops, and null safety (Optional).

**Want the detailed project breakdown for any day?** Let me know! 🚀

New chat