Java8 - Case Study

1. Lambda Expressions – Case Study: Sorting and Filtering Employees

Scenario:
You are building a human resource management module. You need to:

- Sort employees by name or salary.
- Filter employees with a salary above a certain threshold.

Use Case:
Instead of creating multiple comparator classes or anonymous classes, you use Lambda expressions to sort and filter employee records in a concise and readable manner.

2. Stream API & Operators – Case Study: Order Processing System

Scenario:
In an e-commerce application, you must:

- Filter orders above a certain value.
- Count total orders per customer.
- Sort and group orders by product category.

Use Case:
Streams help to process collections like orders using operators like `filter`, `map`, `collect`, `sorted`, and `groupingBy` to build readable pipelines for data processing.

3. Functional Interfaces – Case Study: Custom Logger

Scenario:
You want to create a logging utility that allows:

- Logging messages conditionally.
- Reusing common log filtering logic.

Use Case:
You define a custom `LogFilter` functional interface and allow users to pass behavior using lambdas. You also utilize built-in interfaces like `Predicate` and `Consumer`.

4. Default Methods in Interfaces – Case Study: Payment Gateway Integration

Scenario:
You're integrating multiple payment methods (PayPal, UPI, Cards) using interfaces.

Use Case:
You use default methods in interfaces to provide shared logic (like transaction logging or currency conversion) without forcing each implementation to re-define them.

5. Method References – Case Study: Notification System

Scenario:
You're sending different types of notifications (Email, SMS, Push). The methods for sending are already defined in separate classes.

Use Case:
You use method references (e.g., `NotificationService::sendEmail`) to refer to existing static or instance methods, making your event dispatcher concise and readable.

6. Optional Class – Case Study: User Profile Management

Scenario:
User details like email or phone number may be optional during registration.

Use Case:
To avoid `NullPointerException`, you wrap potentially null fields in `Optional`. This forces developers to handle absence explicitly using methods like `orElse`, `ifPresent`, or `map`.

7. Date and Time API (java.time) – Case Study: Booking System

Scenario:
A hotel or travel booking system that:

- Calculates stay duration.
- Validates check-in/check-out dates.
- Schedules recurring events.

Use Case:
You use the new `LocalDate`, `LocalDateTime`, `Period`, and `Duration` classes to perform safe and readable date/time calculations.

8. Executor Service – Case Study: File Upload Service

Scenario:
You allow users to upload multiple files simultaneously and want to manage the processing efficiently.

Use Case:
You use `ExecutorService` to handle concurrent uploads by creating a thread pool, managing background tasks without blocking the UI or main thread.