📝 Java Practice Set for Spring Boot Backend

---

1. ArrayList (5 Questions)

Q1 (Easy): Create an `ArrayList<String>` of cities. Add 3 cities and print them.

Q2 (Medium): Create an `ArrayList<Integer>` with `[1,2,3,4,5]`. Remove all even numbers.

Q3 (Medium+): Find the largest number in an `ArrayList<Integer>`.

Q4 (Hard): Reverse an `ArrayList<String>` without using `Collections.reverse()`.

Q5 (Challenging): Create a `User` class (`id`, `name`). Store multiple users in an `ArrayList`.

- Search for a user with `id=2`.
- Print `"Not Found"` if missing.

---

2. Map (5 Questions)

Q1 (Easy): Create a `HashMap<String, Integer>` of fruits and prices. Add 3 fruits and print them.

Q2 (Medium): Create a `HashMap<Integer, String>` of student IDs and names. Retrieve the student with ID `101`.

Q3 (Medium+): Count the frequency of each character in a string using `HashMap<Character, Integer>`.

Q4 (Hard): Create a `LinkedHashMap<Integer, String>` to maintain insertion order of employees and print them in order.

Q5 (Challenging): Use a `TreeMap<Integer, String>` to store roll numbers and names. Print them in descending order of roll numbers.

---

3. Lambda & Functional Interfaces (5 Questions)

Q1 (Easy): Sort an `ArrayList<Integer> [5,3,8,1]` in ascending order using a lambda comparator.

Q2 (Medium): Use a `Predicate<Integer>` to filter numbers greater than 10 from a list.

Q3 (Medium+): Use a `Function<String, Integer>` to calculate string lengths.

Q4 (Hard): Create a custom functional interface `MathOperation` with a method `operate(int a, int b)`. Implement addition and multiplication using lambdas.

Q5 (Challenging): Given a list of words, use a `Consumer<String>` lambda to print each word with its length.

---

4. Stream API (5 Questions)

Q1 (Easy): Given a list `[2, 4, 6, 8]`, use streams to double each number.

Q2 (Medium): From a list of names, filter only those starting with `"A"`.

Q3 (Medium+): From a list of integers, find the sum of all odd numbers using `reduce`.

Q4 (Hard): Given a list of employees (`name`, `department`), group them by department using `Collectors.groupingBy`.

Q5 (Challenging): Flatten a list of lists `[[1,2], [3,4], [5]]` into a single list `[1,2,3,4,5]` using `flatMap`.

---

5. Utility Classes (Optional, DateTime, Optional, String) (5 Questions)

Q1 (Easy): Use `Optional` to handle a missing value from a `Map`. Return `"Unknown"` if missing.

Q2 (Medium): Get today's date using `LocalDate` and print it in format `dd-MM-yyyy`.

Q3 (Medium+): Parse `"2025-09-24"` into a `LocalDate` object.

Q4 (Hard): Use `StringBuilder` to reverse the string `"SpringBoot"`.

Q5 (Challenging): Use `Optional` with a method `findUserById(int id)` that may return `null`. If present, print user's name; else print `"Not Found"`.

🔥 Use Case (Spring Boot-like)

Scenario: You are building a REST API that returns orders grouped by customers.

Task:

- Define a class `Order` (`orderId`, `customerName`, `amount`).

- Create a `List<Order>` with sample orders.
- Use Stream API and Collectors.groupingBy to group orders by `customerName`.
- For each customer, calculate the total order amount.

Expected Output Example:

```
Alice → 250
Bob → 400
Charlie → 150
```

✅ This set progressively covers basic to advanced questions across all the core Java features most used in Spring Boot.