📝 Java Practice Questions for Spring Boot Backend Foundations

---

Q1: ArrayList

Question: Create an `ArrayList` of strings representing fruits.

- Add `"Apple"`, `"Banana"`, `"Orange"`, `"Mango"`.
- Remove `"Banana"`.
- Print all fruits using a `for-each` loop.

Solution:

```
import java.util.ArrayList;

public class ArrayListExample {
    public static void main(String[] args) {
        ArrayList<String> fruits = new ArrayList<>();
        fruits.add("Apple");
        fruits.add("Banana");
        fruits.add("Orange");
        fruits.add("Mango");

        fruits.remove("Banana");

        for (String fruit : fruits) {
            System.out.println(fruit);
        }
    }
}
```

Q2: Map

Question: Create a `HashMap<Integer, String>` for employee IDs and names.

- Add entries: `101 → "Alice"`, `102 → "Bob"`, `103 → "Charlie"`.
- Print all IDs and names using a `for-each` loop.

Solution:

```
import java.util.HashMap;
import java.util.Map;

public class MapExample {
```

```
    public static void main(String[] args) {
        Map<Integer, String> employees = new HashMap<>();
        employees.put(101, "Alice");
        employees.put(102, "Bob");
        employees.put(103, "Charlie");

        for (Map.Entry<Integer, String> entry : employees.entrySet()) {
            System.out.println("ID: " + entry.getKey() + ", Name: " + entry.getValue());
        }
    }
}
```

Q3: Lambda & Functional Interface

Question: Sort a list of integers `[5, 2, 9, 1, 7]` in ascending order using a lambda expression.

Solution:

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

public class LambdaExample {
    public static void main(String[] args) {
        List<Integer> numbers = new ArrayList<>();
        Collections.addAll(numbers, 5, 2, 9, 1, 7);

        numbers.sort((a, b) -> a - b);  // Lambda comparator

        System.out.println(numbers); // [1, 2, 5, 7, 9]
    }
}
```

Q4: Stream API

Question: Given a list of names `["John", "Jane", "Jack", "Jill"]`,

- Convert all names to uppercase,
- Filter those starting with `"J"`,
- Collect them into a new list.

Solution:

```java
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;

public class StreamExample {
    public static void main(String[] args) {
        List<String> names = Arrays.asList("John", "Jane", "Jack", "Jill");

        List<String> result = names.stream()
                .map(String::toUpperCase)
                .filter(name -> name.startsWith("J"))
                .collect(Collectors.toList());

        System.out.println(result); // [JOHN, JANE, JACK, JILL]
    }
}
```

Q5: Utility Class (Optional + LocalDate)

Question: Use `Optional` to handle a missing key in a `Map`.

- Create a `Map<Integer, String>` with a single entry `1 → "Hello"`.
- Try to fetch key 2.
- If not present, return `"Default"`.

Solution:

```java
import java.util.HashMap;
import java.util.Map;
import java.util.Optional;

public class OptionalExample {
    public static void main(String[] args) {
        Map<Integer, String> data = new HashMap<>();
        data.put(1, "Hello");

        Optional<String> value = Optional.ofNullable(data.get(2));
        System.out.println(value.orElse("Default")); // Default
    }
}
```

Use Case Question (Spring Boot-like scenario)

Question: You are building a Spring Boot API that returns all adult users (age □ 18). Given a list of `User` objects (`name`, `age`):

```
List<User> users = Arrays.asList(
    new User("Alice", 22),
    new User("Bob", 17),
    new User("Charlie", 25)
);
```

- Use Stream API to filter adults,
- Collect their names into a list,
- Print the result.

Solution:

```
import java.util.*;
import java.util.stream.Collectors;

class User {
    String name;
    int age;

    User(String name, int age) {
        this.name = name;
        this.age = age;
    }
}

public class UseCaseExample {
    public static void main(String[] args) {
        List<User> users = Arrays.asList(
            new User("Alice", 22),
            new User("Bob", 17),
            new User("Charlie", 25)
        );

        List<String> adultNames = users.stream()
                .filter(user -> user.age >= 18)
                .map(user -> user.name)
                .collect(Collectors.toList());

        System.out.println(adultNames); // [Alice, Charlie]
    }
}
```