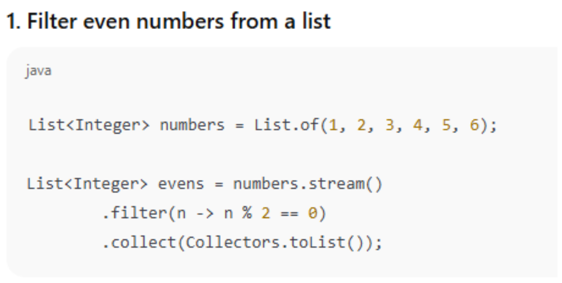
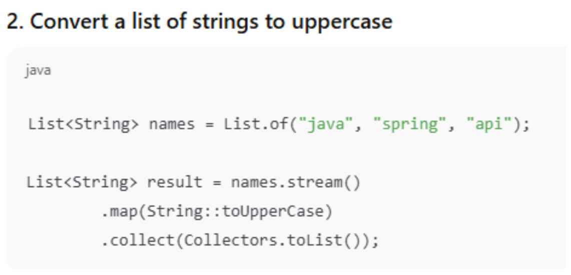
Streams Coding Qns



1. Filter even numbers from a list

java

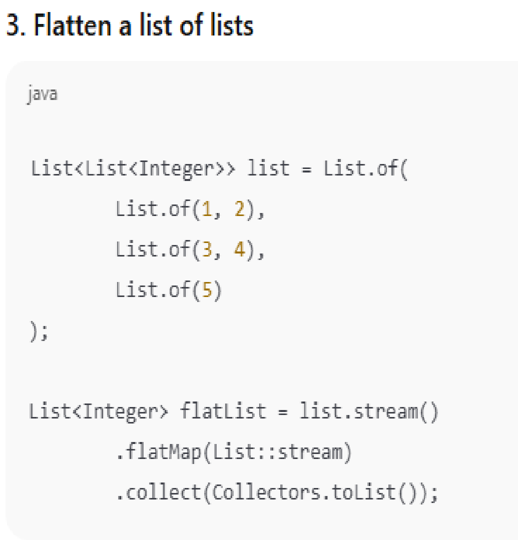
```java
List<Integer> numbers = List.of(1, 2, 3, 4, 5, 6);

List<Integer> evens = numbers.stream()
        .filter(n -> n % 2 == 0)
        .collect(Collectors.toList());
```



2. Convert a list of strings to uppercase

java

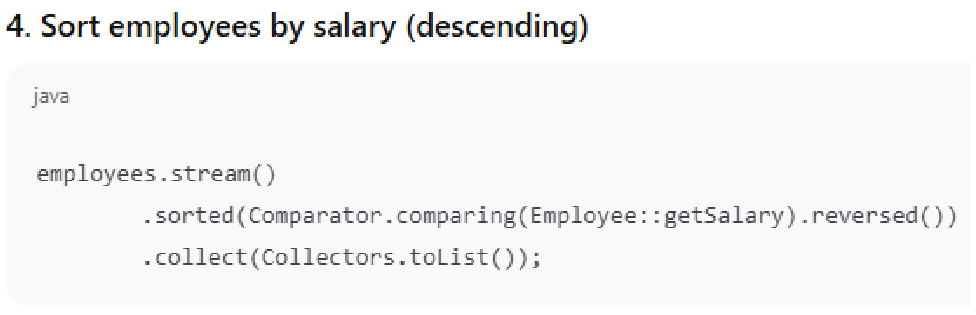
```java
List<String> names = List.of("java", "spring", "api");

List<String> result = names.stream()
        .map(String::toUpperCase)
        .collect(Collectors.toList());
```



3. Flatten a list of lists

java

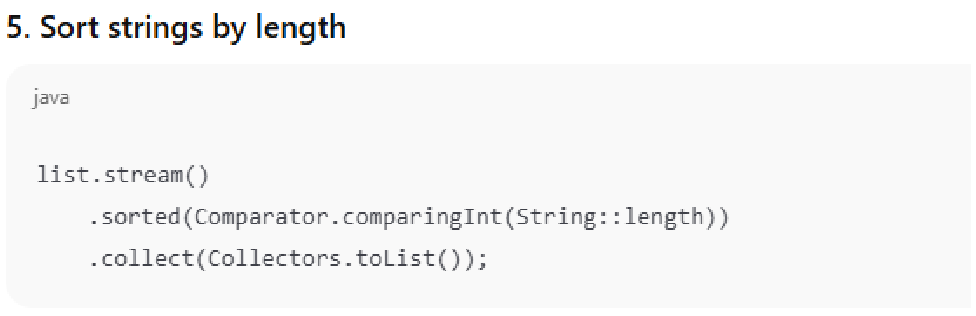
```java
List<List<Integer>> list = List.of(
        List.of(1, 2),
        List.of(3, 4),
        List.of(5)
);

List<Integer> flatList = list.stream()
        .flatMap(List::stream)
        .collect(Collectors.toList());
```



4. Sort employees by salary (descending)

java

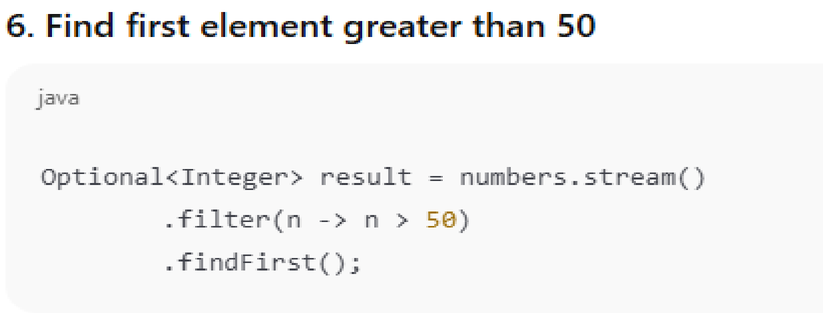
```java
employees.stream()
        .sorted(Comparator.comparing(Employee::getSalary).reversed())
        .collect(Collectors.toList());
```



5. Sort strings by length

java

```java
list.stream()
    .sorted(Comparator.comparingInt(String::length))
    .collect(Collectors.toList());
```



6. Find first element greater than 50

java

```java
Optional<Integer> result = numbers.stream()
        .filter(n -> n > 50)
        .findFirst();
```

7. Check if any number is divisible by 5

java

```java
boolean exists = numbers.stream()
        .anyMatch(n -> n % 5 == 0);
```

🔹 Reduce Operations

8. Find sum of numbers

java

```java
int sum = numbers.stream()
        .reduce(0, Integer::sum);
```

9. Find maximum number

java

```java
Optional<Integer> max = numbers.stream()
        .reduce(Integer::max);
```

🔹 Collectors (Very Important)

10. Group employees by department

java

```java
Map<String, List<Employee>> map =
        employees.stream()
                .collect(Collectors.groupingBy(Employee::getDepartment));
```

11. Count employees in each department

java

```java
Map<String, Long> countMap =
        employees.stream()
                .collect(Collectors.groupingBy(
                        Employee::getDepartment,
                        Collectors.counting()));
```

12. Partition numbers into even and odd

java

```java
Map<Boolean, List<Integer>> partition =
        numbers.stream()
                .collect(Collectors.partitioningBy(n -> n % 2 == 0));
```

🔹 Advanced Stream Questions

13. Find second highest number

java

```java
int secondHighest = numbers.stream()
        .distinct()
        .sorted(Comparator.reverseOrder())
        .skip(1)
        .findFirst()
        .orElseThrow();
```

14. Find duplicate elements

java

```java
Set<Integer> duplicates = numbers.stream()
        .filter(n -> Collections.frequency(numbers, n) > 1)
        .collect(Collectors.toSet());
```

✅ Better (efficient) approach:

java

```java
Set<Integer> seen = new HashSet<>();
Set<Integer> duplicates = numbers.stream()
        .filter(n -> !seen.add(n))
        .collect(Collectors.toSet());
```

15. Convert List to Map (handle duplicates)

java

```java
Map<Integer, String> map =
        employees.stream()
        .collect(Collectors.toMap(
            Employee::getId,
            Employee::getName,
            (oldVal, newVal) -> oldVal
        ));
```

🔹 String-Based Stream Questions

16. Count character frequency

java

```java
String str = "programming";

Map<Character, Long> freq =
        str.chars()
           .mapToObj(c -> (char) c)
           .collect(Collectors.groupingBy(
                   c -> c, Collectors.counting()));
```

17. Find first non-repeated character

java

```java
Character result =
        str.chars()
           .mapToObj(c -> (char) c)
           .collect(Collectors.groupingBy(
                   c -> c,
                   LinkedHashMap::new,
                   Collectors.counting()))
           .entrySet()
           .stream()
           .filter(e -> e.getValue() == 1)
           .map(Map.Entry::getKey)
           .findFirst()
           .orElse(null);
```

2 Find Duplicate Numbers

java

```java
Set<Integer> seen = new HashSet<>();

Set<Integer> duplicates = list.stream()
        .filter(n -> !seen.add(n))
        .collect(Collectors.toSet());
```

4 Longest String in List

java

```java
String longest = list.stream()
        .max(Comparator.comparingInt(String::length))
        .orElse("");
```

5 Sum of Squares of Even Numbers

java

```java
int sum = list.stream()
        .filter(n -> n % 2 == 0)
        .map(n -> n * n)
        .mapToInt(Integer::intValue)
        .sum();
```

6 Group Anagrams

java

```java
Map<String, List<String>> anagrams =
        words.stream()
        .collect(Collectors.groupingBy(
            w -> w.chars()
                  .sorted()
                  .collect(StringBuilder::new,
                           StringBuilder::appendCodePoint,
                           StringBuilder::append)
                  .toString()
        ));
```

◆ PART 2: Real-Time Employee-Based Stream Problems

Sample Employee Class

java

```java
class Employee {
    int id;
    String name;
    String department;
    double salary;
    int age;

    // getters
}
```

9 Count Employees in Each Department

java

```java
Map<String, Long> countMap =
        employees.stream()
                .collect(Collectors.groupingBy(
                        Employee::getDepartment,
                        Collectors.counting()));
```

🔟 Highest Paid Employee

```java
Employee maxSalaryEmp =
        employees.stream()
                 .max(Comparator.comparing(Employee::getSalary))
                 .orElse(null);
```

1️⃣1️⃣ Highest Paid Employee in Each Department

```java
Map<String, Employee> highestPaid =
        employees.stream()
        .collect(Collectors.groupingBy(
            Employee::getDepartment,
            Collectors.collectingAndThen(
                Collectors.maxBy(Comparator.comparing(Employee::getSalary)),
                Optional::get
            )
        ));
```

1️⃣2️⃣ Average Salary by Department

```java
Map<String, Double> avgSalary =
        employees.stream()
        .collect(Collectors.groupingBy(
            Employee::getDepartment,
            Collectors.averagingDouble(Employee::getSalary)
        ));
```

1️⃣3️⃣ Employees Whose Salary > Average Salary

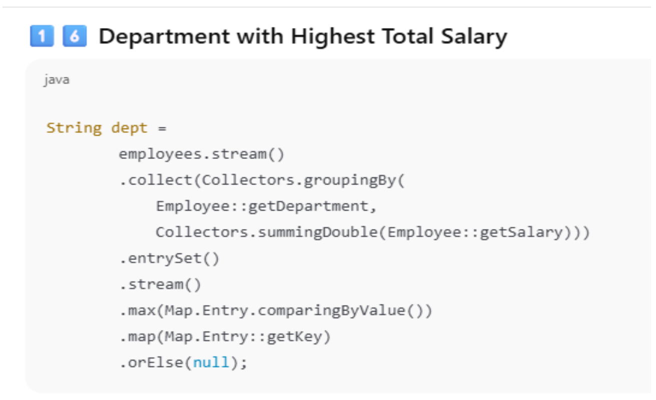
```java
double avgSalary =
        employees.stream()
                 .mapToDouble(Employee::getSalary)
                 .average()
                 .orElse(0);

List<Employee> result =
        employees.stream()
                 .filter(e -> e.getSalary() > avgSalary)
                 .toList();
```

1️⃣4️⃣ Sort Employees by Salary then Name

```java
employees.stream()
          .sorted(Comparator.comparing(Employee::getSalary)
                   .thenComparing(Employee::getName))
          .toList();
```

1️⃣5️⃣ Partition Employees by Age (>30)

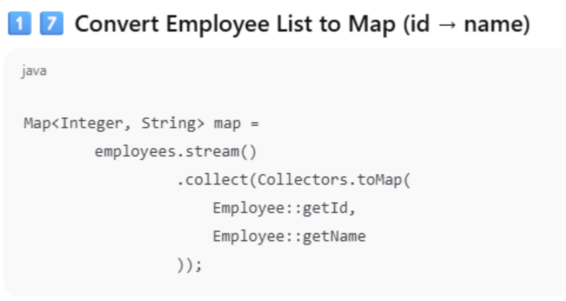
```java
Map<Boolean, List<Employee>> partition =
        employees.stream()
                 .collect(Collectors.partitioningBy(e -> e.getAge() > 30));
```



1️⃣6️⃣ Department with Highest Total Salary

java

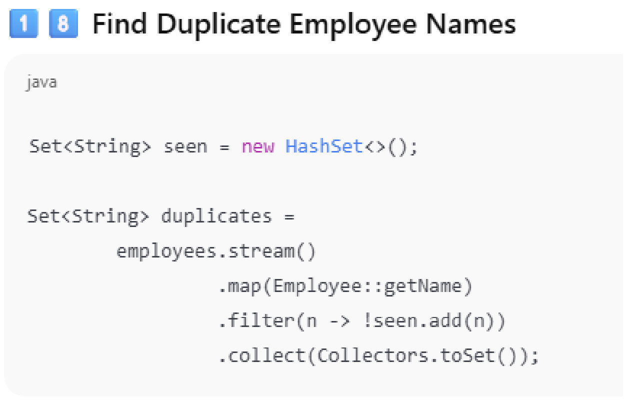
```java
String dept =
        employees.stream()
        .collect(Collectors.groupingBy(
            Employee::getDepartment,
            Collectors.summingDouble(Employee::getSalary)))
        .entrySet()
        .stream()
        .max(Map.Entry.comparingByValue())
        .map(Map.Entry::getKey)
        .orElse(null);
```



1️⃣7️⃣ Convert Employee List to Map (id → name)

java

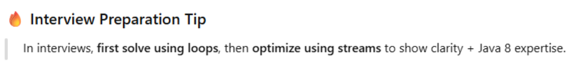
```java
Map<Integer, String> map =
        employees.stream()
                 .collect(Collectors.toMap(
                     Employee::getId,
                     Employee::getName
                 ));
```



1️⃣8️⃣ Find Duplicate Employee Names

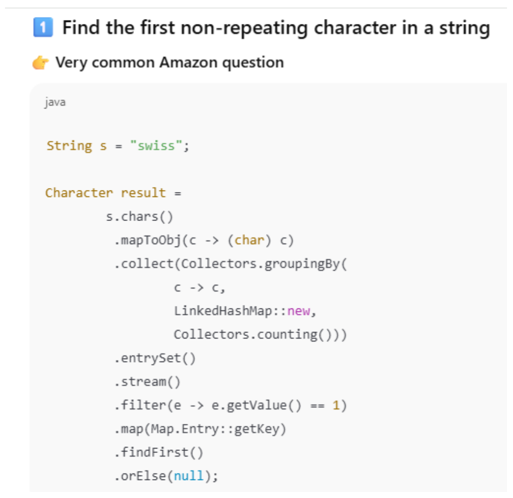
java

```java
Set<String> seen = new HashSet<>();

Set<String> duplicates =
        employees.stream()
                 .map(Employee::getName)
                 .filter(n -> !seen.add(n))
                 .collect(Collectors.toSet());
```



🔥 **Interview Preparation Tip**

> In interviews, **first solve using loops**, then **optimize using streams** to show clarity + Java 8 expertise.



1️⃣ Find the first non-repeating character in a string

👉 **Very common Amazon question**

java

```java
String s = "swiss";

Character result =
        s.chars()
         .mapToObj(c -> (char) c)
         .collect(Collectors.groupingBy(
                 c -> c,
                 LinkedHashMap::new,
                 Collectors.counting()))
         .entrySet()
         .stream()
         .filter(e -> e.getValue() == 1)
         .map(Map.Entry::getKey)
         .findFirst()
         .orElse(null);
```

3️⃣ Group employees by department and find highest paid in each

java

```java
Map<String, Employee> result =
        employees.stream()
        .collect(Collectors.groupingBy(
            Employee::getDepartment,
            Collectors.collectingAndThen(
                Collectors.maxBy(Comparator.comparing(Employee::getSalary)),
                Optional::get
            )
        ));
```

6️⃣ Count occurrences of each character

java

```java
Map<Character, Long> map =
        str.chars()
            .mapToObj(c -> (char) c)
            .collect(Collectors.groupingBy(
                    c -> c,
                    Collectors.counting()));
```

7️⃣ Convert list to map (handle duplicate keys)

java

```java
Map<Integer, String> map =
        employees.stream()
        .collect(Collectors.toMap(
            Employee::getId,
            Employee::getName,
            (oldVal, newVal) -> oldVal
        ));
```

8️⃣ Average salary of employees

java

```java
double avgSalary =
        employees.stream()
                 .mapToDouble(Employee::getSalary)
                 .average()
                 .orElse(0);
```

9️⃣ Sort employees by name and salary

java

```java
employees.stream()
         .sorted(Comparator.comparing(Employee::getName)
                 .thenComparing(Employee::getSalary))
         .toList();
```

1️⃣1️⃣ Find maximum number

java

```java
int max =
        list.stream()
            .max(Integer::compare)
            .orElseThrow();
```

1️⃣2️⃣ Convert string to uppercase

java

```java
list.stream()
    .map(String::toUpperCase)
    .toList();
```

1️⃣3️⃣ Sum of all even numbers

java

```java
int sum =
        list.stream()
            .filter(n -> n % 2 == 0)
            .mapToInt(Integer::intValue)
            .sum();
```

1️⃣4️⃣ Find employee with minimum salary

java

```java
Employee emp =
        employees.stream()
                 .min(Comparator.comparing(Employee::getSalary))
                 .orElse(null);
```