```
public record Person(int age,String name,int salary,String address){}
```

1. Group persons by age and collect names into a list

- **Question**: How would you group the list of `Person` objects by age and collect their names into a `List<String>` for each age group?

Code Example:
java
Copy code
```
Map<Integer, List<String>> result = people.stream()
    .collect(Collectors.groupingBy(
        Person::getAge,
        Collectors.mapping(Person::getName, Collectors.toList())
    ));
```

-
- **Explanation**: This groups people by their age and collects their names into a list.

2. Count the number of people in each age group

- **Question**: How would you count the number of persons in each age group using Java 8 Streams?

Code Example:
java
Copy code
```
Map<Integer, Long> result = people.stream()
    .collect(Collectors.groupingBy(
        Person::getAge,
        Collectors.counting()
    ));
```

-
- **Explanation**: This groups the people by their age and counts how many people are in each age group.

3. Collect a list of names into an unmodifiable list

- **Question**: How would you collect the names of all persons into an unmodifiable list?

Code Example:
java

Copy code
```java
List<String> result = people.stream()
    .map(Person::getName)
    .collect(Collectors.collectingAndThen(
        Collectors.toList(),
        Collections::unmodifiableList
    ));
```

-
- **Explanation**: The `collectingAndThen` collector first collects the names into a `List` and then makes it unmodifiable.

4. Find the total salary of all people older than 30

- **Question**: How would you calculate the total salary of all people older than 30?

Code Example:
java
Copy code
```java
int totalSalary = people.stream()
    .filter(person -> person.getAge() > 30)
    .mapToInt(Person::getSalary)
    .sum();
```

-
- **Explanation**: The stream filters out people whose age is 30 or below, maps their salary to an integer stream, and sums it up.

5. Group people by age and calculate the average salary for each age group

- **Question**: How would you group persons by age and calculate the average salary for each age group?

Code Example:
java
Copy code
```java
Map<Integer, Double> result = people.stream()
    .collect(Collectors.groupingBy(
        Person::getAge,
        Collectors.averagingDouble(Person::getSalary)
    ));
```

-
- **Explanation**: This groups the people by their age and calculates the average salary for each group.

6. Filter persons by salary greater than a threshold and collect their names

- **Question**: How would you filter people whose salary is greater than 50,000 and collect their names?

Code Example:
java
Copy code
```
List<String> result = people.stream()
    .filter(person -> person.getSalary() > 50000)
    .map(Person::getName)
    .collect(Collectors.toList());
```

-
- **Explanation**: The stream filters people with a salary greater than 50,000 and collects their names into a list.

7. FlatMap addresses of persons into a list (if they have multiple addresses)

- **Question**: If each person has multiple addresses (stored as a list in each `Person` object), how would you collect all the unique addresses into a list?

Code Example:
java
Copy code
```
List<String> result = people.stream()
    .flatMap(person -> person.getAddresses().stream())
    .distinct()
    .collect(Collectors.toList());
```

-
- **Explanation**: This uses `flatMap` to flatten the list of addresses for each person and collects all unique addresses into a single list.

8. Group people by age and concatenate their names for each age group

- **Question**: How would you group people by age and concatenate their names into a single string for each age group?

Code Example:
java
Copy code
```
Map<Integer, String> result = people.stream()
    .collect(Collectors.groupingBy(
        Person::getAge,
        Collectors.mapping(Person::getName, Collectors.joining(",
"))
    ));
```

-
- **Explanation**: This groups the people by age and concatenates the names of people in each age group into a single string, separated by commas.

9. Partition people into two groups: salary above 60,000 and below 60,000

- **Question**: How would you partition the list of people into two groups: one where the salary is greater than or equal to 60,000 and another where it's less than 60,000?

Code Example:
java
Copy code
```
Map<Boolean, List<Person>> result = people.stream()
    .collect(Collectors.partitioningBy(person -> person.getSalary()
>= 60000));
```

-
- **Explanation**: This partitions the people into two groups, based on whether their salary is above or below 60,000.

10. Find the highest paid person in each age group

- **Question**: How would you find the highest-paid person in each age group?

Code Example:
java
Copy code
```
Map<Integer, Optional<Person>> result = people.stream()
    .collect(Collectors.groupingBy(
        Person::getAge,
        Collectors.maxBy(Comparator.comparingInt(Person::getSalary))
    ));
```

-
- **Explanation**: This groups people by age and finds the person with the highest salary in each age group using `maxBy`.