Java Streams Practice Questions and Solutions

Q1. Given a list of integers, return a list of even numbers.

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

Q2. Convert a list of strings to uppercase.

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

Q3. Remove null, empty, and blank strings from a list.

```
List<String> cleaned = list.stream().filter(s -> s != null && !s.trim().isEmpty()).collect(Collectors.toList());
```

Q4. Find the square of each number from a list of integers.

```
List<Integer> squares = list.stream().map(n -> n * n).collect(Collectors.toList());
```

Q5. Given a list of names, find how many names start with a vowel.

```
long count = list.stream().filter(n -> n.matches("(?i)^[aeiou].*")).count();
```

Q6. Find sum, average, min, and max from a list of integers.

```
IntSummaryStatistics stats = list.stream().mapToInt(Integer::intValue).summaryStatistics();
```

Q7. Sort a list of strings in reverse order and remove duplicates.

```
List<String> sorted = list.stream().distinct().sorted(Comparator.reverseOrder()).collect(Collectors.toList());
```

Q8. Count how many integers are greater than 50.

```
long count = list.stream().filter(n -> n > 50).count();
```

Q9. Find the longest string in a list.

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

Q10. Concatenate all strings in a list with comma separator.

```
String joined = list.stream().collect(Collectors.joining(", "));
```

Q11. Group products by category and count how many per category.

```
Map<String, Long> map = products.stream().collect(Collectors.groupingBy(p -> p.category, Collectors.counting()));
```

Q12. Group products by category and average the price.

```
Map<String, Double> avg = products.stream().collect(Collectors.groupingBy(p -> p.category, Collectors.averagingDouble(p -> p.price)));
```

Q13. Create a Map<String, List<Product>> grouped by category.

```
Map<String, List<Product>> map = products.stream().collect(Collectors.groupingBy(p -> p.category));
```

Q14. Partition products into two lists based on price > 100.

```
Map<Boolean, List<Product>> map = products.stream().collect(Collectors.partitioningBy(p -> p.price > 100));
```

Q15. Find the most expensive product in each category.

```
Map<String, Product> maxMap = products.stream().collect(Collectors.groupingBy(p -> p.category, Collectors.collectingAndThen(Collectors.maxBy(Comparator.comparingDouble(p -> p.price)), Optional::get)));
```

Q16. Find all employees from IT department.

```
List<Employee> itEmps = employees.stream().filter(e -> e.dept.equals("IT")).collect(Collectors.toList());
```

Q17. Find employee with the highest salary.

```
Employee highest = employees.stream().max(Comparator.comparingInt(e -> e.salary)).orElse(null);
```

Q18. Create map of department to total salary paid.

```
Map<String, Integer> totalSal = employees.stream().collect(Collectors.groupingBy(e -> e.dept, Collectors.summingInt(e -> e.salary)));
```

Q19. Sort employees by salary descending and age ascending.

```
List<Employee> sorted = employees.stream().sorted(Comparator.comparingInt(Employee::getSalary).reversed().thenComparingInt(e -> e.age)).collect(Collectors.toList());
```

Q20. Group employees by department and find average salary.

```
Map<String, Double> avgSal = employees.stream().collect(Collectors.groupingBy(e -> e.dept, Collectors.averagingInt(e -> e.salary)));
```

Q21. Flatten a list of list of integers.

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

Q22. Extract all unique words from comma-separated strings.

```
List<String> unique = list.stream().flatMap(s -> Arrays.stream(s.split(","))).distinct().collect(Collectors.toList());
```

Q23. Find the first string that starts with 'A' and has length > 3.

```
Optional<String> result = list.stream().filter(s -> s.startsWith("A") && s.length() > 3).findFirst();
```

Q24. Check if any name starts with Z.

```
boolean anyZ = list.stream().anyMatch(s -> s.startsWith("Z"));
```

Q25. Check if all ages are above 18.

```
boolean allAdult = list.stream().allMatch(age -> age > 18);
```

Q26. Create stream of first 10 even numbers.

```
List<Integer> evens = Stream.iterate(0, n -> n + 2).limit(10).collect(Collectors.toList());
```

Q27. Generate 5 random numbers using Stream.generate.

```
List<Integer> randoms = Stream.generate(() -> new Random().nextInt(100)).limit(5).collect(Collectors.toList());
```

Q28. Print first 10 Fibonacci numbers.

```
Stream.iterate(new int[]{0, 1}, a -> new int[]{a[1], a[0]+a[1]}).limit(10).map(a -> a[0]).forEach(System.out::println);
```

Q29. Find top 3 scorers from student list.

```
List<Employee> top3 = students.stream().sorted(Comparator.comparingInt(e -> -e.salary)).limit(3).collect(Collectors.toList());
```

Java Streams Practice Questions and Solutions

Q30. Group strings by length and count frequency.

```
Map<Integer, Long> map = list.stream().collect(Collectors.groupingBy(String::length, Collectors.counting()));
```

Q31. Convert CSV string like 'A,10,B,20' to Map<String, Integer>.

```
Map<String, Integer> map = IntStream.range(0, tokens.length/2).boxed().collect(Collectors.toMap(i -> tokens[i*2], i -> Integer.parseInt(tokens[i*2+1])));
```

Q32. Remove duplicates, multiply by 3, and sort.

```
List<Integer> result = list.stream().distinct().map(n -> n * 3).sorted().collect(Collectors.toList());
```

Q33. Find most recent date from list of strings.

```
LocalDate max = dates.stream().map(LocalDate::parse).max(LocalDate::compareTo).orElse(null);
```

Q34. Find pairs from a list whose sum is even.

```
List<List<Integer>> pairs = list.stream().flatMap(i -> list.stream().filter(j -> (i + j) % 2 == 0).map(j -> Arrays.asList(i, j))).collect(Collectors.toList());
```