Using `forEach` for Logging

Write a program that logs each item in a list of names using `forEach` and a lambda expression.

Instructions:

1. Create a `List<String>` containing some names (e.g., "Alice", "Bob", "Charlie").
2. Use `forEach` with a lambda to print each name in the format: `"Name: <name>"`.

Example Input:

```
List<String> names = Arrays.asList("Alice", "Bob", "Charlie");
```

Expected Output:

```
Name: Alice
Name: Bob
Name: Charlie
```

Remove Names Based on Length

Filter out names that are shorter than 4 characters using `removeIf`.

Instructions:

1. Create a `List<String>` of names with varying lengths.
2. Use `removeIf` with a lambda expression to remove all names shorter than 4 characters.

Example Input:

```
List<String> names = new ArrayList<>(Arrays.asList("Tom",
↪  "Lisa", "Eve", "Robert", "Kate"));
```

Expected Output:

```
[Lisa, Robert, Kate]
```

Modify Names Using `replaceAll`

Convert all names in a list to uppercase using `replaceAll`.

Instructions:

1. Create a `List<String>` of names.
2. Use `replaceAll` with a lambda to transform each name to uppercase.

Example Input:

```
List<String> names = new ArrayList<>(Arrays.asList("Alice",
↪  "Bob", "Charlie"));
```

Expected Output:

```
[ALICE, BOB, CHARLIE]
```

Find and Remove Even Numbers

Remove all even numbers from a list of integers using `removeIf`.

Instructions:

1. Create a `List<Integer>` with both even and odd numbers.
2. Use `removeIf` with a lambda to remove all even numbers.

Example Input:

```
List<Integer> numbers = new ArrayList<>(Arrays.asList(1, 2, 3,
↪  4, 5, 6, 7, 8));
```

Expected Output:

```
[1, 3, 5, 7]
```

Append a Suffix

Append a specific suffix (e.g., "_done") to all strings in a list using `replaceAll`.

Instructions:

1. Create a `List<String>` of task names.
2. Use `replaceAll` with a lambda to append "_done" to each task.

Example Input:

```
List<String> tasks = new ArrayList<>(Arrays.asList("task1",
↪  "task2", "task3"));
```

Expected Output:

```
[task1_done, task2_done, task3_done]
```

Reverse a List

Sort a list of integers in descending order using `sort`.

Instructions:

1. Create a `List<Integer>` with unsorted numbers.
2. Use `List.sort` with a lambda expression to sort the numbers in descending order.

Example Input:

```
List<Integer> numbers = new ArrayList<>(Arrays.asList(5, 2, 8,
↪  1, 3));
```

Expected Output:

```
[8, 5, 3, 2, 1]
```

Remove Strings Containing Specific Characters

Remove all strings that contain the character 'e' using `removeIf`.

Instructions:

1. Create a `List<String>` with several words.
2. Use `removeIf` with a lambda to filter out words containing 'e'.

Example Input:

```
List<String> words = new ArrayList<>(Arrays.asList("apple",
↪  "banana", "cherry", "date", "fig"));
```

Expected Output:

```
[banana, fig]
```

Square Each Number

Multiply each number in a list by itself using `replaceAll`.

Instructions:

1. Create a `List<Integer>` of numbers.
2. Use `replaceAll` with a lambda to replace each number with its square.

Example Input:

```
List<Integer> numbers = new ArrayList<>(Arrays.asList(1, 2, 3,
↪  4));
```

Expected Output:

```
[1, 4, 9, 16]
```

Sorting by Last Character

Sort a list of names by their last character using `sort`.

Instructions:

1. Create a `List<String>` of names.
2. Use `Collections.sort` or `List.sort` with a lambda expression to sort the list by the last character of each name.

Example Input:

```
List<String> names = new ArrayList<>(Arrays.asList("Eve",
↪  "Alice", "Charlie", "Bob"));
```

Expected Output:

```
[Bob, Eve, Alice, Charlie]
```

Count Specific Items

Use `forEach` with a lambda to count how many times a specific item appears in a list.

Instructions:

1. Create a `List<String>` with duplicate items (e.g., “apple”, “orange”, “apple”, “banana”).
2. Use `forEach` with a lambda to count the occurrences of “apple”.

Example Input:

```
List<String> fruits = Arrays.asList("apple", "orange", "apple",
 ↪  "banana", "apple");
```

Expected Output:

```
The word 'apple' appears 3 times.
```