CodInClub

Powered By BridgeLabz

List Interface

1. **Reverse a List**
   Write a program to reverse the elements of a given List without using built-in reverse methods. Implement it for both ArrayList and LinkedList.
   Example:
   Input: [1, 2, 3, 4, 5] → Output: [5, 4, 3, 2, 1].

2. **Find Frequency of Elements**
   Given a list of strings, count the frequency of each element and return the results in a Map<String, Integer>.
   Example:
   Input: ["apple", "banana", "apple", "orange"] → Output: {apple=2, banana=1, orange=1}.

3. **Rotate Elements in a List**
   Rotate the elements of a list by a given number of positions.
   Example:
   Input: [10, 20, 30, 40, 50], rotate by 2 → Output: [30, 40, 50, 10, 20].

4. **Remove Duplicates While Preserving Order**
   Remove duplicate elements from a list while maintaining the original order of elements.
   Example:
   Input: [3, 1, 2, 2, 3, 4] → Output: [3, 1, 2, 4].

5. **Find the Nth Element from the End**
   Given a singly linked list (use LinkedList), find the Nth element from the end without calculating its size.
   Example:
   Input: [A, B, C, D, E], N=2 → Output: D.

Set Interface

1. **Check if Two Sets Are Equal**
   Compare two sets and determine if they contain the same elements, regardless of order.
   Example:
   Set1: {1, 2, 3}, Set2: {3, 2, 1} → Output: true.

2. **Union and Intersection of Two Sets**
   Given two sets, compute their union and intersection.
   Example:
   Set1: {1, 2, 3}, Set2: {3, 4, 5} → Union: {1, 2, 3, 4, 5}, Intersection: {3}.

3. **Symmetric Difference**
   Find the symmetric difference (elements present in either set but not in both) of two sets.
   Example:
   Set1: {1, 2, 3}, Set2: {3, 4, 5} → Output: {1, 2, 4, 5}.

4. **Convert a Set to a Sorted List**
   Convert a HashSet of integers into a sorted list in ascending order.
   Example:
   Input: {5, 3, 9, 1} → Output: [1, 3, 5, 9].

5. **Find Subsets**
   Check if one set is a subset of another.
   Example:
   Set1: {2, 3}, Set2: {1, 2, 3, 4} → Output: true.

Insurance Policy Management System

Each policy has the following attributes:
● Policy Number (unique identifier)
● Policyholder Name
● Expiry Date
● Coverage Type (e.g., Health, Auto, Home)
● Premium Amount

Requirements:
1. Store Unique Policies: Implement methods to store policies using different types of sets (HashSet, LinkedHashSet, TreeSet), each serving different purposes:

- HashSet for quick lookups.
- LinkedHashSet to maintain the order of insertion.
- TreeSet to maintain policies sorted by expiry date.

2. Retrieve Policies: Implement methods to retrieve and display policies based on certain criteria:

- All unique policies.
- Policies expiring soon (within the next 30 days
- Policies with a specific coverage type.
- Duplicate policies based on policy numbers.

3. Performance Comparison: Compare the performance of HashSet, LinkedHashSet, and TreeSet in terms of adding, removing, and searching for policies.

Queue Interface

1. **Reverse a Queue**
   Reverse the elements of a queue using only queue operations (e.g., add, remove, isEmpty).
   Example:
   Input: [10, 20, 30] → Output: [30, 20, 10].

2. **Generate Binary Numbers Using a Queue**
   Generate the first N binary numbers (as strings) using a queue.
   Example:
   N=5 → Output: ["1", "10", "11", "100", "101"].

3. **Hospital Triage System**
   Simulate a hospital triage system using a PriorityQueue where patients with higher severity are treated first.
   Example:
   Patients: [("John", 3), ("Alice", 5), ("Bob", 2)] → Order: Alice, John, Bob.

4. **Implement a Stack Using Queues**
   Implement a stack data structure using two queues and support push, pop, and top operations.
   Example:
   Push 1, 2, 3 → Pop → Output: 3.

5. **Circular Buffer Simulation**
   Implement a circular buffer (fixed-size queue) using an array-based queue. When full, overwrite the oldest element.
   Example:
   Buffer size=3: Insert 1, 2, 3 → Insert 4 → Buffer: [2, 3, 4].

Map Interface

1. **Word Frequency Counter**
   Read a text file and count the frequency of each word using a HashMap. Ignore case and punctuation.
   Example:
   Input: "Hello world, hello Java!" → Output: {hello=2, world=1, java=1}

2. **Invert a Map**
   Invert a Map<K, V> to produce a Map<V, K>. Handle duplicate values by storing them in a list.
   Example:
   Input: {A=1, B=2, C=1} → Output: {1=[A, C], 2=[B]}.

3. **Find the Key with the Highest Value**
   Given a Map<String, Integer>, find the key with the maximum value.
   Example:
   Input: {A=10, B=20, C=15} → Output: B.

4. **Merge Two Maps**
   Merge two maps such that if a key exists in both, sum their values.
   Example:
   Map1: {A=1, B=2}, Map2: {B=3, C=4} → Output: {A=1, B=5, C=4}.

5. **Group Objects by Property**
   Given a list of Employee objects, group them by their department using a Map<Department, List<Employee>>.
   Example:
   Employees: [Alice (HR), Bob (IT), Carol (HR)] → Output: HR: [Alice, Carol], IT: [Bob].

Insurance Policy Management System

Build a system for managing insurance policies where you have to:

- Store and manage policies with unique identifiers.
- Retrieve and manipulate policies based on different criteria.
- Track policies by various attributes such as policyholder name and expiry date.

Requirements:

1. Store Policies in a Map:
    - Use HashMap to store policies with policy numbers as keys and policy
    - details as values.
    - Use LinkedHashMap to maintain the insertion order of policies.
    - Use TreeMap to store policies sorted by expiry date.

2. Retrieve and Manipulate Policies:
    1) Implement methods to:
        - Retrieve a policy by its number.
        - List all policies expiring within the next 30 days.
        - List all policies for a specific policyholder.
        - Remove policies that are expired.

Design a Voting System

Description: Design a system where:

- Votes are stored in a HashMap (Candidate -> Votes).i
- TreeMap is used to display the results in sorted order.
- LinkedHashMap is used to maintain the order of votes.

Implement a Shopping Cart

Description:

- Use HashMap to store product prices.
- Use LinkedHashMap to maintain the order of items added.
- Use TreeMap to display items sorted by price.

Implement a Banking System

Description:

- HashMap stores customer accounts (AccountNumber -> Balance).
- TreeMap sorts customers by balance.
- Queue processes withdrawal requests.