1. E-commerce Cart System

Collection Choice: Linked hash map

Reason:
A LinkedHashMap maintains insertion order and allows duplicate keys. By using the item name as the key and quantity as the value, you can efficiently track items in the cart while preserving the order in which they were added.

2. Auto-Complete Feature

Collection Choice: TreeSet or TreeMap

Reason:
Both TreeSet and TreeMap store elements in sorted order. For an auto-complete feature, you can store dictionary words in a TreeSet to ensure they are sorted alphabetically. If you need to associate additional data (like word frequency), TreeMap can be used with the word as the key.

3. LRU Cache

Collection Choice: LinkedHashMap

Reason:
LinkedHashMap can be configured to maintain access order. By overriding the removeEldestEntry method, you can implement an LRU cache that evicts the least recently accessed entries when the cache size exceeds a specified limit.

4. Leaderboard System

Collection Choice: TreeMap

Reason:
A TreeMap stores entries in ascending key order. To maintain a leaderboard with scores in descending order, you can use a custom comparator that reverses the natural ordering. This allows efficient retrieval of the highest scores.

5. Real-Time Notification System

Collection Choice: LinkedList or ArrayDeque

Reason:
Both LinkedList and ArrayDeque implement the Queue interface and maintain the order of elements. They allow for efficient insertion and removal of elements, making them suitable for a notification system where events are processed in the order they are received.

6. Eliminating Duplicate Emails

Collection Choice: LinkedHashSet

Reason:
A LinkedHashSet combines the properties of a HashSet (no duplicates) and a LinkedList (maintains insertion order). This ensures that duplicate email addresses are eliminated while preserving the order of their first appearance.

7. Directory Structure Simulation

Collection Choice: Map<String, List<String>>

Reason:
To represent a directory structure, you can use a Map where the key is the folder name and the value is a list of subfolders or files. This allows for a flexible representation of hierarchical data.

8. Priority-Based Task Scheduling

Collection Choice: PriorityQueue

Reason:
A PriorityQueue is a queue that orders elements according to their natural ordering or by a comparator provided at queue construction time. It is ideal for scheduling tasks based on their priority, ensuring that the highest priority task is processed first.

Gunda Phaninder

52161185