Group Assignment

1. Introduction and Overview
    - Person A: Explain the general idea of the Java Collections Framework, its purpose, and the core interfaces (`Collection`, `List`, `Set`, `Queue`, and `Map`).
2. Lists
    - Person B: Discuss the `List` interface and its common implementations (`ArrayList`, `LinkedList`). Explain their characteristics, use cases, and differences.
        - Key Points:
            - `ArrayList`: Resizable array, fast random access, slower for insertions/removals in the middle.
            - `LinkedList`: Doubly-linked list, efficient for insertions/removals anywhere, slower random access.
3. Sets
    - Person C: Discuss the `Set` interface and its common implementations (`HashSet`, `LinkedHashSet`, `TreeSet`). Explain their characteristics, use cases, and differences.
        - Key Points:
            - `HashSet`: No duplicates, unordered.
            - `LinkedHashSet`: No duplicates, maintains insertion order.
            - `TreeSet`: No duplicates, ordered.
4. Queues
    - Person D: Discuss the `Queue` interface and its common implementations (`PriorityQueue`, `LinkedList`). Explain their characteristics, use cases, and differences.
        - Key Points:
            - `PriorityQueue`: Orders elements based on priority.
            - `LinkedList` as a Queue: FIFO behavior, efficient insertions/removals at both ends.
5. Maps
    - Person E: Discuss the `Map` interface and its common implementations (`HashMap`, `LinkedHashMap`, `TreeMap`). Explain their characteristics, use cases, and differences.
        - Key Points:
            - `HashMap`: Key-value pairs, no order, fast access.

- `LinkedHashMap`: Key-value pairs, maintains insertion/access order.
- `TreeMap`: Key-value pairs, ordered.

Person A: Introduction and Overview

Introduction and Overview of the Java Collections Framework

- Introduction: The Java Collections Framework provides a set of interfaces and classes for storing and manipulating groups of data as a single unit.
- Core Interfaces: Overview of the five core interfaces:
    - Collection: The root of the collection hierarchy.
    - List: Ordered collection that allows duplicates.
    - Set: Unordered collection that does not allow duplicates.
    - Queue: Collection for holding elements prior to processing.
    - Map: Object that maps keys to values, no duplicate keys allowed.

Person B: Lists

List Interface and Implementations

- List Interface: An ordered collection, also known as a sequence, that allows duplicate elements.
- ArrayList:
    - Characteristics: Backed by a dynamic array.
    - Use Cases: Fast random access to elements, best for read-heavy applications.
    - Differences: Slower insertions and deletions in the middle compared to `LinkedList`.
- LinkedList:
    - Characteristics: Doubly-linked list implementation.
    - Use Cases: Efficient for frequent insertions and deletions.
    - Differences: Slower random access compared to `ArrayList`.

Person C: Sets

Set Interface and Implementations

- Set Interface: A collection that does not allow duplicate elements.
- HashSet:
    - Characteristics: Backed by a hash table.
    - Use Cases: Best for fast access and lookups.
    - Differences: No guaranteed order.
- LinkedHashSet:
    - Characteristics: Hash table and linked list implementation.
    - Use Cases: Maintains insertion order.
    - Differences: Slightly slower than `HashSet` due to additional linked list.
- TreeSet:
    - Characteristics: Backed by a tree.
    - Use Cases: Sorted order of elements.
    - Differences: Guarantees log(n) time cost for basic operations.

Person D: Queues

Queue Interface and Implementations

- Queue Interface: Used to hold elements prior to processing, typically in a FIFO manner.
- PriorityQueue:
    - Characteristics: Elements are ordered according to their natural ordering or by a comparator.
    - Use Cases: Tasks needing processing in a specific order, like job scheduling.
    - Differences: Not thread-safe, requires `PriorityBlockingQueue` for concurrent use.
- LinkedList as Queue:
    - Characteristics: Implements `Deque` interface, allowing it to be used as a queue.
    - Use Cases: General-purpose queue, FIFO operations.
    - Differences: Also allows deque operations like adding/removing from both ends.

Person E: Maps

Map Interface and Implementations

- Map Interface: Maps unique keys to values.
- HashMap:
    - Characteristics: Backed by a hash table.
    - Use Cases: Fast access to key-value pairs.
    - Differences: No guaranteed order of keys.
- LinkedHashMap:
    - Characteristics: Hash table and linked list implementation.
    - Use Cases: Maintains insertion order or access order.
    - Differences: Slightly slower than `HashMap` due to additional linked list.
- TreeMap:
    - Characteristics: Backed by a Red-Black tree.
    - Use Cases: Sorted order of keys.
    - Differences: Guarantees log(n) time cost for get, put, and remove operations.