Java Collections

Introduction to Java Collections

- Java Collections Framework is a set of classes and interfaces that implement commonly reusable collection data structures.
- Allows for the storage, retrieval, manipulation, and aggregation of data.
- Includes interfaces such as List, Set, and Map.

List Interface

- An ordered collection, also known as a sequence.
- Allows duplicate elements.
- Common implementations: ArrayList, LinkedList.
- Example:
- List<String> list = new ArrayList<>();
- list.add("Apple");
- list.add("Banana");

Set Interface

- A collection that cannot contain duplicate elements.
- Models mathematical set abstraction.
- Common implementations: HashSet, LinkedHashSet, TreeSet.
- Example:
- Set<String> set = new HashSet<>();
- set.add("Apple");
- set.add("Banana");

Map Interface

- An object that maps keys to values.
- Cannot contain duplicate keys; each key can map to at most one value.
- Common implementations: HashMap, TreeMap, LinkedHashMap.
- Example:
- Map<String, Integer> map = new HashMap<>();
- map.put("Apple", 1);

ArrayList

- Resizable array implementation of the List interface.
- Allows for fast random access.
- Slower for adding/removing elements in the middle of the list.
- Example:
- ArrayList<String> arrayList = new ArrayList<>();
- arrayList.add("Apple");
- arrayList.add("Banana");

LinkedList

- Doubly-linked list implementation of the List interface.
- Allows for constant-time insertions or removals.
- Slower for accessing elements by index.
- Example:
- LinkedList<String> linkedList = new LinkedList<>();
- linkedList.add("Apple");
- linkedList.add("Banana");

HashSet

- Uses a hash table for storage.
- Best for searching, adding, and deleting elements.
- Does not maintain any order.
- Example:
- HashSet<String> hashSet = new HashSet<>();
- hashSet.add("Apple");
- hashSet.add("Banana");

TreeSet

- Implements the Set interface that uses a tree for storage.
- Elements are ordered using their natural ordering.
- Example:
- TreeSet<String> treeSet = new TreeSet<>();
- treeSet.add("Apple");
- treeSet.add("Banana");

HashMap

- Uses a hash table for storage.
- Allows null values and the null key.
- Provides constant-time performance for basic operations.
- Example:
- HashMap<String, Integer> hashMap = new HashMap<>();
- hashMap.put("Apple", 1);
- hashMap.put("Banana", 2);

TreeMap

- Implements the Map interface that uses a tree for storage.
- Orders keys using their natural ordering or by a specified comparator.
- Example:
- TreeMap<String, Integer> treeMap = new TreeMap<>();
- treeMap.put("Apple", 1);
- treeMap.put("Banana", 2);