Java Collections and Generics – Notes

Prepared for GitHub – Beginner to Interview Level

1. What is Collection in Java?

A Collection in Java is a framework that provides an architecture to store, manage, and manipulate a group of objects dynamically. Collections overcome the limitations of arrays such as fixed size and lack of built-in methods.

Why Collections are needed:

- • Arrays are fixed in size
- • No ready-made methods in arrays
- • Collections provide dynamic size
- • Easy insertion, deletion, searching

Collection Framework Hierarchy:

Collection → List (ArrayList, LinkedList), Set (HashSet, TreeSet), Queue (PriorityQueue). Map (HashMap, TreeMap) stores data in key-value pairs and is not a child of Collection.

ArrayList vs LinkedList

- ArrayList uses dynamic array internally
- LinkedList uses doubly linked list internally
- ArrayList is fast for accessing elements
- LinkedList is fast for insertion and deletion

HashSet vs TreeSet

- HashSet stores unique elements without order
- TreeSet stores unique elements in sorted order
- HashSet is faster than TreeSet
- TreeSet uses Red-Black Tree internally

2. What is Generics in Java?

Generics allow us to specify the type of data that a collection or class can work with. Generics provide type safety and reduce runtime errors.

Why Generics are needed:

- • Avoid ClassCastException
- • Type checking at compile time
- • No manual type casting required

- • Cleaner and safer code

Example of Generics with Collection:

ArrayList<Integer> list = new ArrayList<>(); This ensures only Integer values can be stored in the list.

Generic Class Example:

class Box<T> { T value; } Here T represents a type parameter that can be replaced with Integer, String, etc.

Interview Key Points:

- Collection is used to store a group of objects dynamically
- Generics provide type safety
- ArrayList is best for search operations
- LinkedList is best for frequent insert/delete
- HashSet is faster, TreeSet is sorted

End of Notes