**Java8 features**

1. Lambda Expressions -> A new syntax for providing the implementation of functional interfaces
2. Functional Interfaces -> Interfaces that contain only one abstract method
3. Stream API -> Enables functional-style operations on streams of elements(collections)
4. New Date and Time API -> New API for handling date and time (java.time package)
5. Method References -> Simplified way to refer to methods using :: operator

**Stream API**

The Stream API is used for processing sequence of elements, such as collections.

It supports operations like

1. **filter()** -> filters elements based on predicate
2. **map()** -> Transforms elements
3. **reduce()** -> Aggregates elements
4. **collect()** -> Converts a stream into a collection or another form

**Collection Vs Stream**

**Collection:** A data structure that holds data and allows random access to elements.

**Stream:** A sequence of elements that supports aggregate operations like filtering and mapping. It doesn't store elements, but computes them on demand.

Collections are used to store data, while streams are used to process data.

**List Vs Set vs Map**

**List:** A list is an ordered collection, allowing duplicate elements. Elements can be accessed by their position(index) ---> 1)ArrayList, 2)LinkedList

**Set:** A Set is an unordered collection that does not allow duplicate elements. It can only store unique elements. -----> 1)HashSet, 2)TreeSet

**Map:** A map is a collection that stores key-value pairs. It does not allow duplicate keys, but allows duplicate values. ----> 1)HashMap, 2)TreeMap

**ArrayList Vs LinkedList**

**ArrayList:** It uses a dynamic array to store elements. It provides fast access with O(1) time complexity for get() but slow add() and remove() operations due to potential resizing and shifting.

**LinkedList:** It uses a doubly linked list. Insertion and deletion operations are faster compared to ArrayList but accessing time is slower

**HashSet Vs TreeSet**

**HashSet:** It is unoredered collection that does not maintain any specific order of elements. It allows null values and provides constant-time performance for basic operations(O(1))

**TreeSet:** It is a sorted set that maintains elements in natural order or according to a custom Comparator. It is not allow null values and has O(log n) time complexity for most operations

**HashMap Vs TreeMap**

**HashMap:** It is a non-synchronized, unordered map that stores elements in key-value pairs. It allows null keys and values, and its operations(like get() and put()) have a time complexity of O(1).

**TreeMap:** It is a sorted map based on Red-Black tree structure. It doesn't allow null keys and sorts entries by their keys. The operations have a time complexity of O(log n)

**Iterator Vs ListIterator**

**Iterator:** Can be used to traverse elements of a collection in one direction(forward). It is applicable to all collection types.

**ListIterator:** It is specific to List implementations and allows traversal in both directions(forward and backward). It is also provides methods to modify the list(set(), add()).

**Intermediate Vs Terminal Operation in Stream**

**Intermediate Operations:** These returns a new stream and lazy, meaning they don't execute until a terminal operation is invoked.

filter(), map(),sorted()

**Terminal Operations:** These trigger the execution of the stream pipeline and produce a result or a side-effect.

collect(), forEach(), reduce()