1.Describe the Collections Type Hierarchy. What Are the Main Interfaces, and What Are the Differences Between Them?
The hierarchy of the entire collection framework consists of four core interfaces such as Collection, List, Set, Map, and two specialized interfaces named SortedSet and SortedMap for sorting.

Collection framework mainly consists of four interfaces:

1] List: It contains elements arranged in sequential order, and these elements can be accessed by using their index position (starting from 0). It implements ArrayList, Vector, and VectorList. Elements can be inserted in the list at any position but can be retrieved in the same order in which they are added. It can also store duplicate elements.

2] Queue: Queue contains similar elements in which new elements are added from one end, whereas removed from the other end (FIFO – First In First Out). It implements LinkedList and PriorityQueue.

3] Set: Set is used to store the collection of unique elements. Duplicacy is not allowed in the Set. It does not store the elements in sequential order. While accessing the elements, we may not get the elements in the same order in which they are stored. HashSet, LinkedHashSet, and TreeSet implements the Set interface. Elements in the Set can only be iterated using Iterator.

4] Map: Map stores the elements in the form of Key/Value pairs. It is extended by SortedMap and implemented by HashMap and HashTable. It does not contain duplicate values, and each key can, at most, store one value.

2. What are List interface implementations and what are the differences between them and when to use what?

There are two general-purpose List implementations — ArrayList and LinkedList. Most of the time, you'll probably use ArrayList, which offers constant-time positional access and is just plain fast.

If you frequently add elements to the beginning of the List or iterate over the List to delete elements from its interior, you should consider using LinkedList.

3. What are Queue interface implementations and what are the differences and when to use what?

The Queue interface is used to hold the elements about to be processed in FIFO(First In First Out order.

LinkedList
- Head is the first element of the list

PriorityQueue
- Head is the smallest/largest element

4. What are Set interface implementations and what are the differences and when to use what?

three general-purpose Set implementations: HashSet, TreeSet, and LinkedHashSet.

HashSet, which stores its elements in a hash table, is the best-performing implementation; however it makes no guarantees concerning the order of iteration.

TreeSet, which stores its elements in a red-black tree, orders its elements based on their values; it is substantially slower than HashSet.

LinkedHashSet, which is implemented as a hash table with a linked list running through it, orders its elements based on the order in which they were inserted into the set (insertion-order).

5. What is the Hashcode() and equal() function?

Hashing is a process of converting an object into integer form by using the method hashCode().

simply checks if two Object references (say x and y) refer to the same Object.

6. How Is Hashmap Implemented in Java? How Does Its Implementation Use Hashcode and Equals Methods of Objects? What Is the Time Complexity of Putting and Getting an Element from Such Structure?

Hashmap uses the array of Nodes(named as table), where Node has fields like the key, value (and much more). Here the Node is represented by class HashMapEntry. Basically, HashMap has an array where the key-value data is stored. It calculates the index in the array where the Node can be placed and it is placed there. Now while getting the element from HashMap, it again calculates the index of the element to retrieve and goes to the array index and returns the value of the element/Node(if exists).

In HashMap, hashCode() is used to calculate the bucket and therefore calculate the index.
HashMap uses equals() to compare the key to whether they are equal or not. If the equals() method return true, they are equal otherwise not equal.
HashMap has complexity of O(1) for insertion and lookup.

7. What is Comparable and Comparator interface? What are differences between them and how to use them?

Comparable interface is used to sort the objects with natural ordering. Comparator in Java is used to sort attributes of different objects. Comparable interface compares “this” reference with the object specified. Comparator in Java compares two different class objects provided.

8. What is Iterator? Why do we need iterator?

Iterator in Java is an object used to cycle through arguments or elements present in a collection. Generally, an iterator in Java is used to loop through any collection of objects.

9. What is Stream API?

the Stream API is used to process collections of objects. A stream is a sequence of objects that supports various methods which can be pipelined to produce the desired result.

10. What is the difference between the stream API and the collections?

stream API doesn't store data, it operates on the source data structure i.e collection.

the collections stores/holds all the data that the data structure currently has in a particular data structure like Set, List or Map,

11. What are intermediate operations and terminal operations, what is the difference.

The main difference between intermediate and terminal operations is that intermediate operations return a stream as a result and terminal operations return non-stream values like primitive or object or collection or may not return anything.

12. What is the advantages of builder design patter?

Builder pattern aims to Separate the construction of a complex object from its representation • It is used to construct a complex object step by step and the final step will return the object. • Readability • Minimizing the number of parameters

13. What is Optional Class and why we need it?

Optional is a container object used to contain not-null objects. Optional object is used to represent null with absent value. This class has various utility methods to facilitate code to handle values as 'available' or 'not available' instead of checking null values.