Q1) What are the differences between Exception and Error in java?

Exception	Error
● Exception can be recovered by using the try-catch block. ● It can be classified into two categories i.e. checked and unchecked. ● It occurs at compile time or run time. ● It belongs to java.lang.Exception package. ● Only checked exceptions are known to the compiler. ● It is mainly caused by the application itself. ● Examples: NullpointerException, ArithmaticException.	● An error cannot be recovered. ● All errors in java are unchecked. ● It occurs at run time. ● It belongs to java.lang.Errors package. ● Errors will not be known to the compiler. ● It is mostly caused by the environment in which the application is running. ● Examples: java.lang.StackoverFlow, java.lang.OutOfMemoryError.

Q2) What is the difference between ArrayList and LinkedList and Explain when do we use ArrayList and LinkedList with proper reasons.

ArrayList	LinkedList
● ArrayList internally uses a dynamic array to store the elements	● LinkedList internally uses a double linked list to store the elements.

• Manipulation with ArrayList is slow because it internally uses an array. If any element is removed from the array, all the other elements are shifted in memory. • An ArrayList class can act as a list only because it implements List only. • ArrayList is better for storing and accessing data. • The memory location for the elements of an ArrayList is contiguous. • Generally, when an ArrayList is initialized, a default capacity of 10 is assigned to the ArrayList. • To be precise, an ArrayList is a resizable array.	• Manipulation with LinkedList is faster than ArrayList because it uses a doubly linked list, so no bit shifting is required in memory. • LinkedList class can act as a list and queue both because it implements List and Deque interfaces. • LinkedList is better for manipulating data. • The location for the elements of a linked list is not contagious. • There is no case of default capacity in a LinkedList. In LinkedList, an empty list is created when a LinkedList is initialized. • LinkedList implements the doubly linked list of the list interface.

<u>Reasons for when do we use ArrayList and LinkedList.</u>

- When the rate of addition or removal rate is more than the read scenarios, then go for the LinkedList. On the other hand, when the frequency of the read scenarios is more than the addition or removal rate, then ArrayList takes precedence over LinkedList.

- Since the elements of an ArrayList are stored more compact as compared to a LinkedList; therefore, the ArrayList is more cache-friendly as compared to the LinkedList. Thus, chances for the

cache miss are less in an ArrayList as compared to a LinkedList. Generally, it is considered that a LinkedList is por in cache-locality.

- Memory overhead in the LinkedList is more as compared to the ArrayList. It is because, in a LinkedList, we have two extra links as it is required to store the address of the previous and the next nodes, and these links consume extra space. Such links are not present in an ArrayList.