1. Write the difference between checked and unchecked exception with example code

-  Checked: The exceptions that are checked at compile time. If some code within a method throws a checked exception, then the method must either handle the exception or it must specify the exception using throws keyword.

**Unchecked**: The exceptions that are not checked at compiled time. In Java exceptions under Error and RuntimeException classes are unchecked exceptions, everything else under throwable is checked.

1. Write the difference between throw and throws with example code

* **Throws clause** is used to declare an exception, which means it works similar to the try-catch block. On the other hand **throw** keyword is used to throw an exception explicitly.
* If we see syntax wise than **throw** is followed by an instance of Exception class and **throws** is followed by exception class names.
* Throw keyword is used in the method body to throw an exception, while throws is used in method signature to declare the exceptions that can occur in the statements present in the method.
* You can throw one exception at a time but you can handle multiple exceptions by declaring them using throws keyword.

1. Write a note on nested try…catch block with example code

* When a try catch block is present in another try block then it is called the nested try catch block. Each time a try block does not have a catch handler for a particular exception, then the catch blocks of parent try block are inspected for that exception, if match is found that that catch block executes. If neither catch block nor parent catch block handles exception then the system generated message would be shown for the exception, similar to what we see when we don’t handle exception.

1. Write a note on MultiThreading and MultiTasking

* Multithreading is a Java feature that allows concurrent execution of two or more parts of a program for maximum utilization of CPU. Each part of such program is called a thread. So, threads are light-weight processes within a process.  
  Threads can be created by using two mechanisms:   
  a. extending the Thread class  
  b. Implementing the Runnable Interface
* Multitasking is ability to execute more than one task at the same time is known as multitasking. It is possible because multithreading.

1. Write a short note on Deque and give example code.

* The java.util.Deque interface is a subtype of the java.util.Queue interface. The Deque is related to the double-ended queue that supports addition or removal of elements from either end of the data structure, it can be used as a queue (first-in-first-out/FIFO) or as a stack (last-in-first-out/LIFO).
* Methods of deque:
* add(element): Adds an element to the tail.
* addFirst(element): Adds an element to the head.
* addLast(element): Adds an element to the tail.
* offer(element): Adds an element to the tail and returns a boolean to explain if the insertion was successful.
* offerFirst(element): Adds an element to the head and returns a boolean to explain if the insertion was successful.
* offerLast(element): Adds an element to the tail and returns a boolean to explain if the insertion was successful.
* iterator(): Returns an iterator for this deque.
* descendingIterator(): Returns an iterator that has the reverse order for this deque.
* push(element): Adds an element to the head.
* pop(element): Removes an element from the head and returns it.
* removeFirst(): Removes the element at the head.
* removeLast(): Removes the element at the tail.

1. Write a short note on Generics an all types of Parameters used in Generics with example code.

* Before Generics, java collection was used for all data types in single object. That was not a scientific approach because, in every element separate type casting is needed. In order to solve this issue, the pre-defined classes of collection package were created with Generic type <T> . Thus while using any of these classes in our program, we can replace <T> will any wrapper class type. Once we associate a type in place of <T>, that particular object will be able to store only that TYPE of data.

1. Write a short note on Map Interface.

* It is a separate interface from collection package. However, it falls under the framework of collections because it stores series or collection of data. But since it stores data in KEY + VALUE pain which is a type of mapping of data, thus it is termed as MAP interface.
* It is implemented by 2 main classes:

1. HashMap
2. LinkedHashMap
3. Write the difference between LinkedList and ArrayList.

* ArrayList and LinkedList both implements List interface and their methods and results are almost identical. However there are few differences between them which make one better over another depending on the requirement.
* Search: ArrayList search operation is pretty fast compared to the LinkedList search operation. get(int index) in ArrayList gives the performance of O(1) while LinkedList performance is O(n). ArrayList maintains index based system for its elements as it uses array data structure implicitly which makes it faster for searching an element in the list. On the other side LinkedList implements doubly linked list which requires the traversal through all the elements for searching an element.
* Deletion: LinkedList remove operation gives O(1) performance while ArrayList gives variable performance: O(n) in worst case (while removing first element) and O(1) in best case (While removing last element). LinkedList element deletion is faster compared to ArrayList. LinkedList’s each element maintains two pointers (addresses) which points to the both neighbour elements in the list. Hence removal only requires change in the pointer location in the two neighbour nodes (elements) of the node which is going to be removed. While In ArrayList all the elements need to be shifted to fill out the space created by removed element.
* Inserts Performance: LinkedList add method gives O(1) performance while ArrayList gives O(n) in worst case. Reason is same as explained for remove.
* Memory Overhead: ArrayList maintains indexes and element data while LinkedList maintains element data and two pointers for neighbor nodes hence the memory consumption is high in LinkedList comparatively.

1. Write a note on Dynamic array in java.​​

* In Java, the size of an array is fixed when it is created. Elements are not allowed to be inserted or removed. It is possible to implement a dynamic array by allocating a new array and copying the contents from the old array to the new one. A dynamic array has variable size and allows elements to be added or removed. For this, we can allocate a fixed-size array and divide it into two parts:
* The first part stores the elements of the dynamic array and the second part is reserved, but not used.
* Then we can add or remove elements at the end of the array by using the reserved space, until this space is completely consumed. After that, we create a bigger array and copy the contents of the old array to the new one.
* Logical size (size): the number of elements in the dynamic array
* Capacity: the physical size of the internal array (the maximum possible size without relocating storage)
* We now design a class Dynamic Array represents dynamic arrays of integers. It has two attributes:
* int[] data: an integer array, and
* int size: the logical size, the number of elements used
* The capacity of this dynamic array is simply data.length.
* An important method we need is to add elements to the end of the dynamic array. This method should provide automatic extension if the capacity is not large enough to hold the added element.