1. Strings in Java and Major Operations

In Java, a String is an immutable object that represents a sequence of characters. Strings are part of the `java.lang` package and are widely used in Java programming for text manipulation.

Major Operations with Strings:

- Concatenation: Joining two or more strings using the + operator or concat() method.

```
String s1 = "Hello";
String s2 = "World";
String s3 = s1 + s2; // HelloWorld
```

- Length: Finding the number of characters in a string using the length() method.

```
int len = s1.length(); // 5
```

- Substring: Extracting a part of a string using substring(int beginIndex, int endIndex).

```
String sub = s1.substring(0, 4); // "Hell"
```

- Comparison: Comparing two strings using equals() for exact match and compareTo() for lexicographical comparison.

```
boolean isEqual = s1.equals(s2); // false
```

- Conversion: Converting a string to upper/lower case using toUpperCase() or toLowerCase().

```
String upper = s1.toUpperCase(); // "HELLO"
```

2. Linear and Non-Linear Data Structures in Java

Linear Data Structures:

These structures store data sequentially, and elements are arranged in a linear order.

- **Arrays**: A collection of elements of the same type stored in contiguous memory locations.
- **Linked Lists**: Each element points to the next, allowing dynamic memory allocation.
- **Stacks**: A Last In First Out (LIFO) structure where insertion and removal occur at the top.
- **Queues**: A First In First Out (FIFO) structure where elements are added at the rear and removed from the front.

Non-Linear Data Structures:

These structures do not store data sequentially. Elements are connected hierarchically.

- **Trees**: A collection of nodes organized in a hierarchical structure (e.g., Binary Trees, AVL Trees).
- **Graphs**: A set of vertices connected by edges, used to represent networks.

3. Arrays and Matrices in Java

- Arrays:

An array in Java is a collection of elements of the same type stored in contiguous memory locations. Arrays can have fixed sizes and can be *one*-dimensional or *multi*-dimensional.

- Declaration:

```
int[] arr = new int[5];
int[] arr2 = {1, 2, 3, 4, 5};
```

- Matrices:

A matrix is essentially a two-dimensional array where each element is accessed using two indices (rows and columns).

```
int[][] matrix = new int[3][3]; // 3x3 matrix
```

4. ArrayList in Java and Difference from Array

- **ArrayList**:

ArrayList is part of java.util package and is a resizable array, meaning it can grow or shrink dynamically. Unlike arrays, ArrayList does not have a fixed size, and it allows adding or removing elements easily.

```
ArrayList<Integer> list = new ArrayList<>();
list.add(10);
list.add(20);
```

- **Differences between ArrayList and Array:**

Size: Arrays are fixed in size, while ArrayList is dynamic.

Type: Arrays can hold both primitive types and objects, but ArrayList only holds objects.

Performance: Arrays are faster for indexed access, but ArrayList provides flexibility and additional methods.

5. LinkedList in Java

LinkedList in Java is a doubly-linked list, where each element (node) contains a reference to the previous and next element. It allows efficient insertion and removal operations.

```
LinkedList<String> list = new LinkedList<>();
list.add("First");
list.add("Second");
```

Key features:

- Better performance for insertion/deletion compared to ArrayList.
- Can be used as a stack, queue, or deque.

6. HashSet and HashMap in Java

HashSet:

A HashSet is a collection that contains no duplicate elements. It uses hashing for storage, ensuring constant-time performance for add, remove, and contains operations.

```
HashSet<Integer> set = new HashSet<>();
set.add(1);
set.add(2);
```

HashMap:

A HashMap is a map-based collection that stores key-value pairs. It allows fast retrieval based on keys.

```
HashMap<String, Integer> map = new HashMap<>();
map.put("One", 1);
map.put("Two", 2);
```

Difference:

- HashSet stores only unique elements, while HashMap stores key-value pairs.
- HashSet is part of the Set interface, while HashMap is part of the Map interface.

7. Stacks and Their Hierarchy in Java

A Stack is a Last In First Out (LIFO) data structure, meaning elements are added and removed from the top. Java provides a Stack class as part of java.util.

Hierarchy:

- The Stack class extends Vector, which means it inherits methods for capacity management.
- It's part of the Collection framework and implements interfaces like List, Serializable, and Cloneable.

```
Stack<Integer> stack = new Stack<>();
stack.push(1);
stack.push(2);
int top = stack.pop(); // Removes and returns top element
```

8. Queues in Java

A Queue is a First In First Out (FIFO) data structure. Elements are added at the rear and removed from the front. Java provides several implementations of Queue under the java.util package, such as LinkedList (which implements Queue) and PriorityQueue.

```
Queue<Integer> queue = new LinkedList<>();
queue.add(10);
queue.add(20);
int front = queue.remove(); // Removes and returns front element
```

Key methods:

- *add()* / *offer()* to insert elements.
- *remove()* / *poll()* to retrieve and remove the front element.
- *peek()* to retrieve the front element without removing it.