**====================== STRINGS ========================**

Strings in Java are objects that represent a sequence of characters. The java.lang.Strings class is used to create and manipulate strings. In Java, strings are immutable, which means once a string is created, it cannot be changed. This immutability offers several benefits, such as security, thread-safety, and performance enhancements due to string pooling.

**Creating Strings**

There are two primary ways to create strings in Java:

1. **String Literals**: This is the most common method to create a string. It uses double quotes and can be directly assigned to a string variable. For example: String greet = “Hi”; When using string literals, Java stores them in the string constant pool, which helps in saving memory. If a string already exists in the pool, Java will not create a new object but will return a reference to the same instance.
2. **Using the new Keyword**: This method creates a new string object in the heap memory, regardless of the content of the string. For example: String ex = new String(“Hi”); Using the new keyword guarantees the creation of a new string object, even if an identical string already exists in the string constant pool.

**String Pool and Memory Management**

Java optimises memory usage by storing string literals in a string constant pool. This pool is a cache that helps avoid creating multiple instances of the same string. When a new string literal is created, Java checks the pool first. If the string exists, it returns a reference to the pooled instance; otherwise, it creates a new string instance and adds it to the pool.

**======================= 2D Arrays =======================**

Brute Force:

- Traverse every element in each row and column until the target is found.

- Usually used when there is no certain order for the matrix elements.

- This is the Brute Force approach.

- Time Complexity: O(m X n)

Row/ Column Sorted Matrix:

- If each row or column is sorted, binary search can be applied to each row or column.

- Time Complexity: O(m X log n)

Completely sorted matrix:

- For matrices sorted both row-wise and column-wise:

- Start from the top-right or bottom-left corner.

- Move left if the element is larger than the target.

- Move down if the element is smaller than the target.

- Time Complexity: O(m + n).

Pathfinding and Graph Traversal in 2D Grids

Depth-First Search (DFS)

- When you need to explore all possible paths, find islands, or check connected components.

- Use recursion or a stack.

Breadth-First Search (BFS)

- Shortest path problems in an unweighted grid.

- Use a queue to explore level by level.

Dynamic Programming on 2D Grids:

- Problems involving path counting, minimum cost path, longest increasing path, etc.

- Create a dp table of the same size as the matrix.

- Use the states of neighbouring cells to compute the value for each cell.