JAVA

DAY 3

C-DAC MUMBAI
AUGUST 2025

LOOPS IN JAVA

- if
- if-else
- for
- while
- do-while
- switch case

- continue
- break

ARRAY

An array is an important linear data structure that allows us to store multiple values of the same type.

- Arrays in Java are objects, like all other objects in Java, arrays implicitly inherit from the java.lang.Object class.
- Arrays have a built-in length property, which provides the number of elements in the array.

KEY FEATURES OF ARRAYS

- **Store Primitives and Objects:** Java arrays can hold both primitive types (like int, char, boolean, etc.) and objects (like String, Integer, etc.)
- **Contiguous Memory Allocation** When we use arrays of primitive types, the elements are stored in contiguous locations. For non primitive types, references of items are stored at contiguous locations.
- **Zero-based Indexing:** The first element of the array is at index 0.
- **Fixed Length:** After creating an array, its size is fixed; we can not change it.

ARRAY DECLARATION

- Array declaration:-
  - int[] a;
  - int []a;
  - int a[];
- declaration & instantiation & initialization :-
  - approach 1:- int a[]={10,20,30,40};
  - approach 2:- int[] a=new int[100];
    - a[0]=10;
    - a[1]=20;
    - a[2]=30;
    - a[4]=40;

WHY ARRAY INDEX SRARTS WITH 0?

①

```
#include <stdio.h>
int main()
{
    int data[6] = {1, 2, 3, 4, 5, 6};
    int i = 0;
    printf("Array address: %p\n", data);
    do {
        printf("Array[%u] = %p\n", i, (void *)(&data[i]));
        i++;
    } while(i < 6);
}
```

```
Array address: 0x7ffe9472bad0
Array[0] = 0x7ffe9472bad0
Array[1] = 0x7ffe9472bad4
Array[2] = 0x7ffe9472bad8
Array[3] = 0x7ffe9472badc
Array[4] = 0x7ffe9472bae0
Array[5] = 0x7ffe9472bae4
```

② Simplified Arithmetic

- If the first element has an index of 0, then its memory address is simply the base address plus 0 times the size of each element.
- The second element (index 1) is at the base address plus 1 times the size of each element, and so on.
- This makes the calculation

  base_address + (index * element_size)

- straightforward and efficient, as it avoids an extra subtraction operation (e.g., index - 1).

ARRAY METHODS

Method	Description
compare()	Compares two arrays
copyOf()	Creates a copy of an array with a new length
fill()	Fills an array with a specified value
mismatch()	Returns the index position of the first mismatch/conflict between two arrays
sort()	Sorts an array in ascending order

STRING

In Java, a String is the type of object that can store a sequence of characters enclosed by double quotes and every character is stored in 16 bits (2 bytes). A string acts the same as an array of characters.

WAYS OF CREATING A JAVA STRING

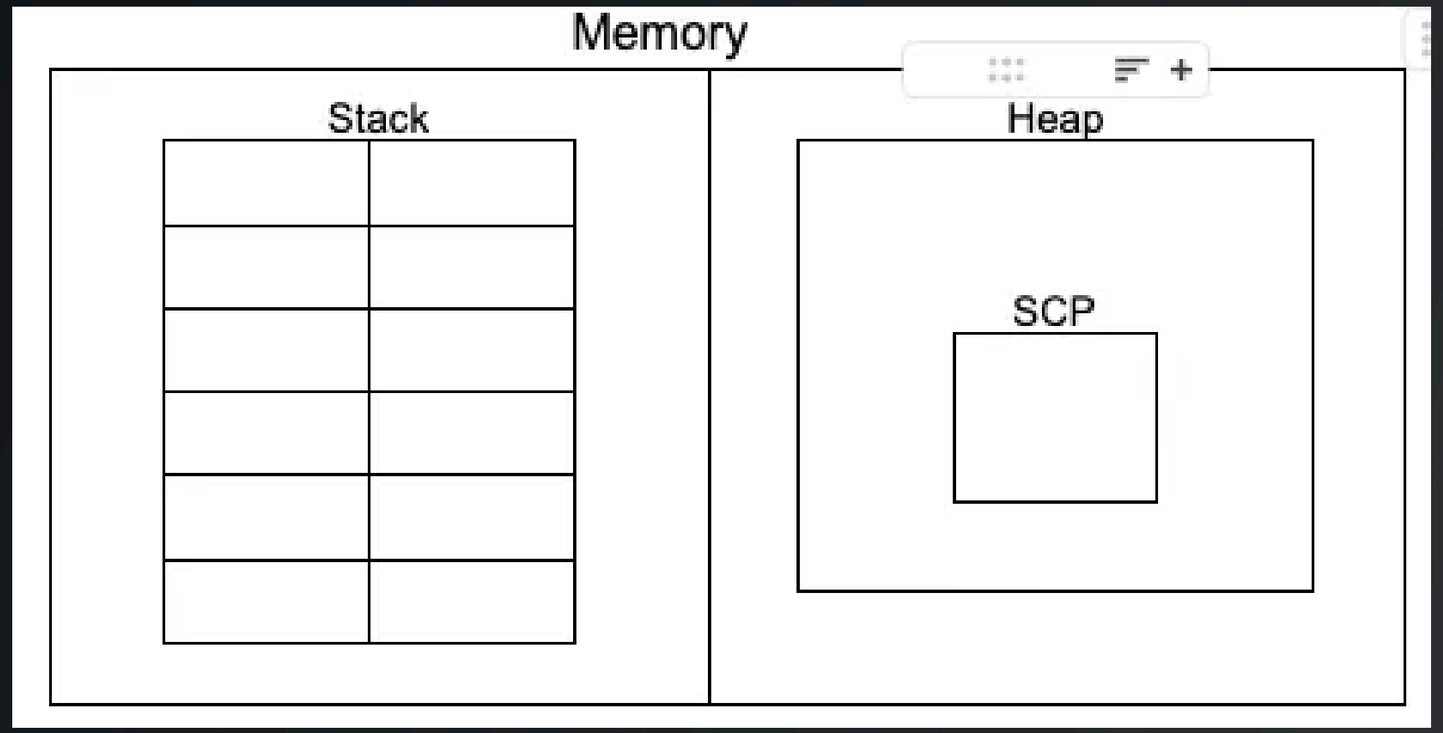
There are two ways to create a string in Java:

- **String Literal**
- **Using new Keyword**

STRINGS ARE STORED IN HEAP MEMORY

STRING

- In Stack memory, variables or variable references or references to the objects are stored.
- In Heap memory, all the objects that are dynamically allocated are stored. To allocate memory to an object we use a new keyword.



STRING CLASS METHODS

Method	Description	Return Type
charAt()	Returns the character at the specified index (position)	char
codePointAt()	Returns the Unicode of the character at the specified index	int
codePointBefore()	Returns the Unicode of the character before the specified index	int
codePointCount()	Returns the number of Unicode values found in a string.	int
compareTo()	Compares two strings lexicographically	int
compareToIgnoreCase()	Compares two strings lexicographically, ignoring case differences	int
concat()	Appends a string to the end of another string	String
contains()	Checks whether a string contains a sequence of characters	boolean

BUFFER AND BUILDER

StringBuffer	StringBuilder
StringBuffer is present since early versions of Java.	StringBuilder was introduced in Java 5 (JDK 1.5).
StringBuffer is synchronized. This means that multiple threads cannot call the methods of StringBuffer simultaneously.	StringBuilder is asynchronized. This means that multiple threads can call the methods of StringBuilder simultaneously.
Due to synchronization, StringBuffer is called a thread safe class.	Due to its asynchronous nature, StringBuilder is not a thread safe class.
Due to synchronization, StringBuffer is lot slower than StringBuilder.	Since there is no preliminary check for multiple threads, StringBuilder is a lot faster than StringBuffer.
Use in multi-threaded environments.	Use in single-threaded or non-concurrent environments.