# Java Programs

**String functions**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Function** | **Description** |
| 1 | [**charAt**](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#charAt-int-)(int index) | Returns the char value at the specified index. |
| 2 | [**compareTo**](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#compareTo-java.lang.String-)([**String**](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html) anotherString) | Compares two strings lexicographically. |
| 3 | [**concat**](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#concat-java.lang.String-)([**String**](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html) str) | Concatenates the specified string to the end of this string. |
| 4 | [**contains**](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#contains-java.lang.CharSequence-)(**[CharSequence](https://docs.oracle.com/javase/8/docs/api/java/lang/CharSequence.html" \o "interface in java.lang)** s) | Returns true if and only if this string contains the specified sequence of char values. |
| 5 | [**equals**](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#equals-java.lang.Object-)([**Object**](https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html) anObject) | Compares this string to the specified object. |
| 6 | [**equalsIgnoreCase**](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#equalsIgnoreCase-java.lang.String-)([**String**](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html) anotherString) | Compares this String to another String, ignoring case considerations. |
| 7 | [**indexOf**](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#indexOf-int-)(int ch) | Returns the index within this string of the first occurrence of the specified character. |
| 8 | [**indexOf**](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#indexOf-int-int-)(int ch, int fromIndex) | Returns the index within this string of the first occurrence of the specified character, starting the search at the specified index. |
| 9 | [**indexOf**](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#indexOf-java.lang.String-)([**String**](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html) str) | Returns the index within this string of the first occurrence of the specified substring. |
| 10 | [**indexOf**](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#indexOf-java.lang.String-int-)([**String**](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html) str, int fromIndex) | Returns the index within this string of the first occurrence of the specified substring, starting at the specified index. |
| 11 | [**isEmpty**](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#isEmpty--)() | Returns true if, and only if, [**length()**](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#length--) is 0. |
| 12 | [**lastIndexOf**](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#lastIndexOf-int-)(int ch), [**lastIndexOf**](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#lastIndexOf-java.lang.String-)([**String**](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html) str) | Returns the index within this string of the last occurrence of the specified character. |
| 13 | [**length**](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#length--)() | Returns the length of this string. |
| 14 | [**matches**](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#matches-java.lang.String-)([**String**](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html) regex) | Tells whether or not this string matches the given [**regular expression**](https://docs.oracle.com/javase/8/docs/api/java/util/regex/Pattern.html#sum). |
| 15 | [**replaceAll**](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#replaceAll-java.lang.String-java.lang.String-)([**String**](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html) regex, [**String**](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html) replacement), [**replaceFirst**](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#replaceFirst-java.lang.String-java.lang.String-)([**String**](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html) regex, [**String**](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html) replacement) | Replaces each substring of this string that matches the given [**regular expression**](https://docs.oracle.com/javase/8/docs/api/java/util/regex/Pattern.html#sum) with the given replacement. |
| 16 | [**split**](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#split-java.lang.String-)([**String**](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html) regex) | Splits this string around matches of the given [**regular expression**](https://docs.oracle.com/javase/8/docs/api/java/util/regex/Pattern.html#sum). |
| 17 | [**subSequence**](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#subSequence-int-int-)(int beginIndex, int endIndex) | Returns a character sequence that is a subsequence of this sequence. |
| 18 | [**substring**](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#substring-int-)(int beginIndex), [**substring**](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#substring-int-int-)(int beginIndex, int endIndex) | Returns a string that is a substring of this string. |
| 19 | [**toCharArray**](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#toCharArray--)() | Converts this string to a new character array. |
| 20 | [**toLowerCase**](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#toLowerCase--)(),**[toUpperCase](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html" \l "toUpperCase--)**() |  |
| 21 | [**trim**](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#trim--)() | Returns a string whose value is this string, with any leading and trailing whitespace removed. |

**Reverse a string without using String inbuilt function**

|  |
| --- |
| public static void main(String[] args) { |

|  |  |
| --- | --- |
|  | // TODO Auto-generated method stub |

|  |  |
| --- | --- |
|  | String str = "Automation"; |

|  |  |
| --- | --- |
|  | StringBuilder str2 = new StringBuilder(); |

|  |  |
| --- | --- |
|  | str2.append(str); |

|  |  |
| --- | --- |
|  | str2 = str2.reverse(); // used string builder to reverse |

|  |  |
| --- | --- |
|  | System.out.println(str2); |

|  |  |
| --- | --- |
|  | } |

------------------------------------------------------------------------------------------------------------------------------------------

|  |
| --- |
| String str = "Saket Saurav"; |

|  |  |
| --- | --- |
|  | char chars[] = str.toCharArray(); |

|  |  |
| --- | --- |
|  | for(int i= chars.length-1; i>=0; i--) { |

|  |  |
| --- | --- |
|  | System.out.print(chars[i]); |

|  |  |
| --- | --- |
|  | } |

------------------------------------------------------------------------------------------------------------------------------------------

**Swap two numbers without using the third variable**

|  |
| --- |
| int x, y; |

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  | x = x + y; |

|  |  |
| --- | --- |
|  | y = x - y; |

|  |  |
| --- | --- |
|  | x = x - y; |

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  |  |

**Swap two numbers without using the third variable**

|  |
| --- |
| int x, y; |

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  | x = x + y; |

|  |  |
| --- | --- |
|  | y = x - y; |

|  |  |
| --- | --- |
|  | x = x - y; |

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  |  |

------------------------------------------------------------------------------------------------------------------------------------------

**find whether a number is prime or not**

|  |
| --- |
| for (int i = 2; i<= num/2; i++) { |

|  |  |
| --- | --- |
|  | temp = num%i; |

|  |  |
| --- | --- |
|  | if (temp == 0) { |

|  |  |
| --- | --- |
|  | isPrime = false; |

|  |  |
| --- | --- |
|  | break; |

|  |  |
| --- | --- |
|  | } |

|  |  |
| --- | --- |
|  | } |

------------------------------------------------------------------------------------------------------------------------------------------

**find whether a string or number is palindrome or not**

|  |
| --- |
| for (int i =length -1; i>=0; i--) { |

|  |  |
| --- | --- |
|  | reverse = reverse + original.charAt(i); |

|  |  |
| --- | --- |
|  | } |

|  |  |
| --- | --- |
|  | System.out.println("reverse is:" +reverse); |

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  | if(original.equals(reverse)) |

|  |  |
| --- | --- |
|  | System.out.println("The number is palindrome"); |

|  |  |
| --- | --- |
|  | else |

|  |  |
| --- | --- |
|  | System.out.println("The number is not a palindrome"); |

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  | } |

------------------------------------------------------------------------------------------------------------------------------------------

**Java Program for Fibonacci series**

**For Example** 0,1,1,2,3,5,8,13,21………

int num, a = 0,b=0, c =1;

|  |
| --- |
| for (int i=0; i<=num; i++) { |

|  |  |  |
| --- | --- | --- |
|  |  | a = b; |

|  |  |
| --- | --- |
|  | b = c; |

|  |  |
| --- | --- |
|  | c = a+b; |

|  |  |
| --- | --- |
|  | System.out.println(a + ""); |
| } |  |

------------------------------------------------------------------------------------------------------------------------------------------

**find the second highest number in an array**

|  |
| --- |
| int arr[] = { 14, 46, 47, 45, 92, 52, 48, 36, 66, 85 }; |

|  |  |
| --- | --- |
|  | int largest = arr[0]; |

|  |  |
| --- | --- |
|  | int secondLargest = arr[0]; |

|  |  |
| --- | --- |
|  | for (int i = 0; i < arr.length; i++) { |

|  |  |
| --- | --- |
|  | if (arr[i] > largest) { |

|  |  |
| --- | --- |
|  | secondLargest = largest; |

|  |  |
| --- | --- |
|  | largest = arr[i]; |

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  | } else if (arr[i] > secondLargest) { |

|  |  |
| --- | --- |
|  | secondLargest = arr[i]; |

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  | } |

|  |  |
| --- | --- |
|  | } |

------------------------------------------------------------------------------------------------------------------------------------------

**Java Program to check Armstrong number**

153 = 1\*1\*1 + 5\*5\*5 + 3\*3\*3 = 1 + 125 + 27 = 153

|  |
| --- |
| int c=0,a,temp; |

|  |  |
| --- | --- |
|  | int n=153;//It is the number to check Armstrong |

|  |  |
| --- | --- |
|  | temp=n; |

|  |  |
| --- | --- |
|  | while(n>0) |

|  |  |
| --- | --- |
|  | { |

|  |  |
| --- | --- |
|  | a=n%10; |

|  |  |
| --- | --- |
|  | n=n/10; |

|  |  |
| --- | --- |
|  | c=c+(a\*a\*a); |

|  |  |
| --- | --- |
|  | } |

|  |  |
| --- | --- |
|  | if(temp==c) |

|  |  |
| --- | --- |
|  | System.out.println("armstrong number"); |

|  |  |
| --- | --- |
|  | else |

|  |  |
| --- | --- |
|  | System.out.println("Not armstrong number"); |

|  |  |
| --- | --- |
|  | } |

------------------------------------------------------------------------------------------------------------------------------------------

**Factorial**

public static int factorial(int number){         
        //base case  
        if(number == 0){  
            return 1;  
        }  
        return number\*factorial(number -1); //is this tail-recursion?  
    }

------------------------------------------------------------------------------------------------------------------------------------------

**Binary Search**

int binary\_search(int A[], int key, int imin, int imax) {  
  
 /\* test if array is empty \*/  
 if (imax < imin)  
  
   /\* set is empty, so return value showing not found \*/  
   return KEY\_NOT\_FOUND;  
 else {  
  
     /\* calculate midpoint to cut set in half \*/  
     int imid = midpoint(imin, imax);  
  
     /\* three-way comparison \*/  
     if (A[imid] > key)  
  
       /\* key is in lower subset \*/  
       return binary\_search(A, key, imin, imid-1);  
     else if (A[imid] < key)  
  
       /\* key is in upper subset \*/  
       return binary\_search(A, key, imid+1, imax);  
     else  
  
       /\* key has been found \*/  
       return imid;  
   }  
}

------------------------------------------------------------------------------------------------------------------------------------------

**Bubble Sort**

 void bubbleSort(int arr[])

    {

        int n = arr.length;

        for (int i = 0; i < n-1; i++)

            for (int j = 0; j < n-i-1; j++)

                if (arr[j] > arr[j+1])

                {

                    // swap temp and arr[i]

                    int temp = arr[j];

                    arr[j] = arr[j+1];

                    arr[j+1] = temp;

                }

    }

------------------------------------------------------------------------------------------------------------------------------------------

**Heap Sort**

**static** **void** libraryHeapSort(**int**[] array){

PriorityQueue<Integer> priorityQueue = **new** PriorityQueue<>();

**int** i;

**for**(i=0; i<array.length; i++){

priorityQueue.add(array[i]);

}

i=0;

**while**(!priorityQueue.isEmpty()){

array[i++] = priorityQueue.poll();

}

}

------------------------------------------------------------------------------------------------------------------------------------------

**Insertion Sort**

void sort(int arr[])

    {

        int n = arr.length;

        for (int i = 1; i < n; ++i) {

            int key = arr[i];

            int j = i - 1;

            /\* Move elements of arr[0..i-1], that are

               greater than key, to one position ahead

               of their current position \*/

            while (j >= 0 && arr[j] > key) {

                arr[j + 1] = arr[j];

                j = j - 1;

            }

            arr[j + 1] = key;

        }

    }

------------------------------------------------------------------------------------------------------------------------------------------

**Merge Sort**

  void merge(int arr[], int l, int m, int r)

    {

        // Find sizes of two subarrays to be merged

        int n1 = m - l + 1;

        int n2 = r - m;

        /\* Create temp arrays \*/

        int L[] = new int [n1];

        int R[] = new int [n2];

        /\*Copy data to temp arrays\*/

        for (int i=0; i<n1; ++i)

            L[i] = arr[l + i];

        for (int j=0; j<n2; ++j)

            R[j] = arr[m + 1+ j];

        /\* Merge the temp arrays \*/

        // Initial indexes of first and second subarrays

        int i = 0, j = 0;

        // Initial index of merged subarry array

        int k = l;

        while (i < n1 && j < n2)

        {

            if (L[i] <= R[j])

            {

                arr[k] = L[i];

                i++;

            }

            else

            {

                arr[k] = R[j];

                j++;

            }

            k++;

        }

        /\* Copy remaining elements of L[] if any \*/

        while (i < n1)

        {

            arr[k] = L[i];

            i++;

            k++;

        }

        /\* Copy remaining elements of R[] if any \*/

        while (j < n2)

        {

            arr[k] = R[j];

            j++;

            k++;

        }

    }

    // Main function that sorts arr[l..r] using

    // merge()

    void sort(int arr[], int l, int r)

    {

        if (l < r)

        {

            // Find the middle point

            int m = (l+r)/2;

            // Sort first and second halves

            sort(arr, l, m);

            sort(arr , m+1, r);

            // Merge the sorted halves

            merge(arr, l, m, r);

        }

    }

**Selection Sort**

void sort(int arr[])

    {

        int n = arr.length;

        // One by one move boundary of unsorted subarray

        for (int i = 0; i < n-1; i++)

        {

            // Find the minimum element in unsorted array

            int min\_idx = i;

            for (int j = i+1; j < n; j++)

                if (arr[j] < arr[min\_idx])

                    min\_idx = j;

            // Swap the found minimum element with the first

            // element

            int temp = arr[min\_idx];

            arr[min\_idx] = arr[i];

            arr[i] = temp;

        }

    }

------------------------------------------------------------------------------------------------------------------------------------------