

STRINGS

import java.io.*
import java.util.*

128 ASCII characters are covered by strings.

Java uses UTF-16 (16 bits) } Unicode format.

Examples

1) `char x = 'a';`
`System.out.println ((int)(x));`

O/P → 97.

2) Array to String

Method 1
`char[] arr = { 'p', 'q', 'r', 's' };`

`String str1 = String.valueOf(arr);`

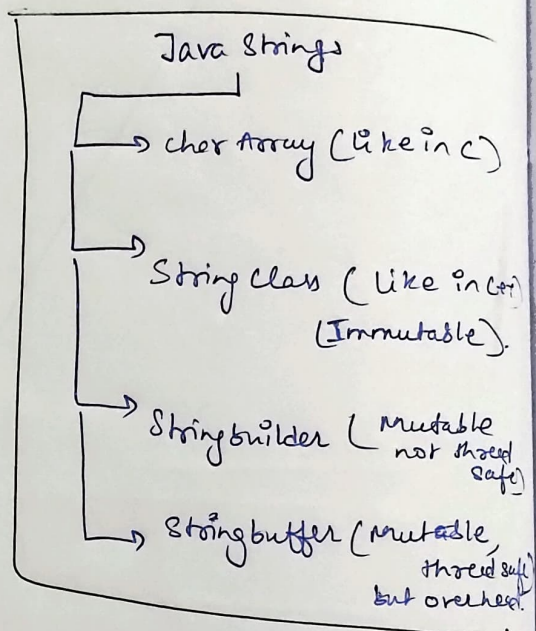
Method 2

Use `Arrays.toString(arr);`

3) String to Array

`x = "abcdef"`

`char[] char1 = x.toCharArray();`



4) for char Array \longrightarrow Use $x[i]$
 for strings \longrightarrow Use $x.charAt[i]$ } to access an element in them.

Q: Print freq of char (in sorted order).

String str = "geeksforgeeks";

int[] count = new int[26];

for (int i=0; i < str.length(); i++)

{ count[str.charAt(i) - 'a']++;

}

for (int i=0; i < 26; i++) \longrightarrow constant time
as only 26 loops
not $O(n)$

{ if (count[i] > 0)

{ System.out.println((char)(i + 'a') + " " + count[i]);

}

}

Note :- charAt cannot be used on left side i.e

str.charAt(3) = 'a'; } not valid
 as strings are
 immutable.

\rightarrow str = "geeks";

str.substring(2,4) \rightarrow ek

\rightarrow not counted,

Imp fn (String)
 contains
 equals \rightarrow for
 compare to
 index of

Q. Palindrome Check.

Main → take rev of str

Compare rev == str & Return true/false.

~~Prog~~ bool isPal (String str)

```
{ StringBuilder str_rev = new StringBuilder(str);
```

```
  str_rev.reverse();
```

```
  System.out.println (str.equals(str_rev.toString()));
```

```
}
```

$O(n)$ & $O(n)$

Efficient → Compare first & last characters.

~~Prog~~

```
{ int begin = 0;
```

```
  int end = str.length() - 1;
```

```
  while (begin < end)
```

```
  { if (str.charAt(begin) != str.charAt(end))
```

```
    return false;
```

```
  }
```

```
    begin++;
```

```
    end--;
```

```
}
```

```
  return true;
```

```
}
```

Q. Check if a string is subseq of other.

Note → Substring is continuous, not subsequence.
→ 2^n subseq possible for n -letter word.

→ Naive → Generate all subseq & compare all.
 $O(2^n * n)$.

Efficient ⇒ 2 pointer approach.

```
if (s1[i] == s2[j]) { i++; j++; }  
else { i++; }
```

return (j == s2.length())

Q. Check for Anagram

2/p → s1 = "listen" s2 = "silent"

O/p → Yes

Naive → Sort both and compare.

In Java sort → $\left[\begin{array}{l} \text{char } a[] = s.toCharArray(); \\ \text{Arrays.sort}(a); \\ s1 = \text{new String}(a); \end{array} \right]$

Efficient → Counting freq like in first question.

count++ for letter in s1
count-- for letter in s2
At end count array for all char must be 0.

Q. Reverse words in a string

I/p \rightarrow "welcome to gfg"

O/p \rightarrow "gfg to welcome"

I/p \rightarrow "abc"

O/p \rightarrow "abc"

Naive \rightarrow Use Stack. , strings separated by space.
(Auxiliary space)

Efficient \rightarrow Reverse individual words
& then reverse the whole string.
(Constant space)

1) \rightarrow abc bda

2) \rightarrow cba adb

3) \rightarrow bda abc