Strings in Java – ICSE Class X Notes

This document is prepared keeping in mind a ready reckoner on important String methods in Java for the students of class X as a handout to refer to while doing programs on Strings

What is a String?

- A **String** is a sequence of **characters** enclosed within double quotes.
- In Java, Strings are **objects** of the **String class** in the `java.lang` package.

How to Declare Strings:

```
String s1 = "Hello"; //here "Hello" is a String literal
                     //and s1 is a String variable
```

0	1	2	3	4	Index value
H	**e**	**l**	**l**	**o**	**Data**

```
String s2 = new String("World"); // Using 'new' keyword
```

0	1	2	3	4	Index value
W	**o**	**r**	**l**	**d**	**Data**

Here, s2 is an object of the String class

Declare a string variable to store your name –

```
String name = "ABHAY KUMAR DUBEY";
```

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
A	**B**	**H**	**A**	**Y**		**K**	**U**	**M**	**A**	**R**		**D**	**U**	**B**	**E**	**Y**

In the variable name, there are a total of 17 characters, out of which 15 are alphabets and 2 are spaces. The starting index is 0 and the ending index is 16.

Note: Strings are immutable in Java, which means their value cannot be changed. Every time we assign a value to a string variable or string object, a new memory allocation takes place.

Sprinden Ganguli

Working with String Methods

String str = "Hello everyone!";
String str2 = "World";

1. **Finding the length of String – length()**

Function prototype : <int> String object.length( )

```
int len=str.length(); //Returns the number of characters present
                      //in the string, including all spaces
System.out.println(len); // 15
```

2. **Extraction of character at a given Index Position – charAt()**

Function prototype : <char> String object.charAt(int)

```
char ch = str.charAt(3); //extract character at index 3
System.out.println(ch);  // l

char ch = str.charAt(0); // extract the character 'h'
char ch = str.charAt(-1); // error
char ch = str.charAt(15); // error
```

Note: If the index position is invalid or beyond the range, the compiler will throw an exception –

'StringIndexOutOfBoundsException'

Example to count all alphabets, digits, special characters and spaces in a string:

```
String str = "Welcome";
int alp=0, dgt=0, spc=0, sl=0;
for(int i = 0; i < str.length(); i++)
{
      char ch = str.charAt(i);
      if(ch>='A'&&ch<='Z' || ch>='a'&&ch<='z')
            alp++;
      else if(ch>='0'&&ch<='9')
            dgt++;
      else if(ch==' ')
            spc++;
      else
            sl++;
}
```

3. **Substring from the given String – substring()**

Function prototype :

<String> String object.substring(int) //one parameter

<String> String object.substring(int, int) //two parameters

String example - H e l l o
Respective index pos. 0 1 2 3 4

The substring method with one parameter will extract the characters from that index position to the end of the string, whereas the substring method with two parameters (a, b) will extract characters from index position a to index position (b-1).

```
String s = str.substring(3);// extract the substring
                            // from index position 3
System.out.println(s); // lo

String t = str.substring(0,3);//extract the substring
                              //from index 0 to index 2
System.out.println(t); // hel

So, we can say – substring(int a) will extract a substring from
the given index position (a) to the end of the string and
substring(int a, int b) will extract a substring from index (a)
to index (b-1).
```

4. **Changing Case of String – converting uppercase to lowercase and v.v.**

Function prototype :

<String> String object.toUpperCase( ) or toLowerCase( )

String str1 = "Hello!", str2 = "World 123";

```
String s = str1.toUpperCase(); // All capital letters – HELLO!

String t = str2.toLowerCase(); // All small letters – world 123
```

Both methods only work on letters and all other characters remain unchanged.

5. **Comparing two Strings**

Function prototype :

```
<boolean> String object.equals(String)
<boolean> String object.equalsIgnoreCase(String)
```

These two methods compare two strings for equality and return true or false.

```
<int> String object.compareTo(String)
<int> String object.compareToIgnoreCase(String)
```

These two methods compare two strings lexicographically, i.e. in alphabetical order. Here, the comparison takes place by the ASCII difference of two characters.

String str1 = "India";

String str2 = "INDIA";

```
boolean a = str1.equals(str2);// false as both strings are equal, but their cases are different

boolean b = str1.equalsIgnoreCase(str2);//it ignores the case and thus returns true as both strings have "India".

int c = str1.compareTo(str2);// 0 if both are equal,
                             // <0 if str1<str2,and
                             // >0 if str1>str2
```

The result of the above method: ASCII(n)–ASCII(N) = 32

```
int d=str1.compareToIgnoreCase(str2);//ignores the case
```

This function will return 0**.**

String s="ABC", t="abc";

int a = s.compareTo(t); //(ASCII diff. of 65-97) = -32

int b = t.compareTo(s); //(ASCII diff. of 97-65) = 32

Difference between ASCII ('A') and ASCII ('a')

String x="ALPHA", y="ALTO";

int c = x.compareTo(t); //-4

Difference between ASCII ('P') and ASCII ('T')

6. Finding a Character or Word in a String

Function prototype :

```
<int> String object.indexOf(char/String)
<int> String object.lastIndexOf(char/String)
<int> String object.indexOf(char/String, int)
```

- indexOf() method searches the character or string in the given string object and returns the index position of the **first occurrence** of that character or string.
- lastIndexOf() method too searches the character or string in the given string object but returns the index position of the **last occurrence** of that character or string.
- indexOf() method with two parameters, one of which is integer will begin the search from that index position passed as integer in the string.

Here one thing to notice, if character is present in the string object only once, both the methods (indexOf() and lastIndexOf()) will return the same index position.

Note: if the specified character or string is not found in the given string object, both the methods will return -1.

Let us see the given examples-

String str = ”JAVA PROGRAMMING”;

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
J	A	V	A		P	R	O	G	R	A	M	M	I	N	G

```
int p = str.indexOf('A'); //Returns first index of 'a' i.e. 1
int q = str.lastIndexOf('A');//Returns last index of 'a' i.e.10
int r = str.indexOf("JAVA"); //Returns index of word "Java"
int s = str.indexOf(‘A’, 5); //Returns index position 10
int t = str.indexOf(“JAVAC”); //Returns -1
```

Note: both the methods are case-sensitive.

7. **Replacing Characters in a String**

Function prototype :

```
<String> String object.replace(char, char)

<String> String object.replace(String, String)
```

```
String t = ”aeroplane”;
String p = ”Monday will be the test”;
String s = t.replace('a','A');//Replaces all 'a' with 'A'
           // AeroplAne
String s = p.replace(“Monday”, “Tuesday”);
           // Tuesday will be the test”
```

8. **String Concatenation (Joining Strings)**

Function prototype :

```
<String> String object.concat(String)
String s1=”Wel”, s2=”come”;
String s3 = s1.concat(s2); //Welcome
String s4 = s1 + s2 + s3;  //WelcomeWelcome
```

9. **Some other methods**
   - **startsWith( )**
   - **endsWith( )**
   - **toCharArray( )**

Function prototype :

```
<boolean> String object.startsWith(char/String)
<boolean> String object.endsWith(char/String)
<char array> String object.toCharArray()
String str = “Happy Birthday”;
```

str.startsWith(‘H’)
str.startsWith(“Happy”)
str.endsWith(‘y’);
str.endsWith(“day”) - all above methods will return true.

char arr[]=str.toCharArray( ); - it will transfer all the characters to a character array where each character of the string will be stored in separate cells.

arr[] = {'H','a','p','p','y',' ','B','i','r','t','h','d','a','y'}

Examples for Practice

Example 1: Count vowels in a string

```
String str = ”Welcome”;
int count = 0;
for(int i = 0; i < str.length(); i++)
{
    char ch = str.charAt(i);
    if("AEIOUaeiou".indexOf(ch) != -1)
    {
        count++;
    }
}
System.out.println(“Total no. of vowels=”+count);
```

Example 2: Check if a string is a palindrome

```
String str=”Nitin”, rev = "";
for(int i = str.length()-1; i >= 0; i--)
{
    rev += str.charAt(i);
}
if(str.equalsIgnoreCase(rev))
     System.out.println("Palindrome");
else
     System.out.println("Not a Palindrome");
```

Example 3: Add the ASCII values of all the letters of a string

```
String str = ”COMPUTER”;
int sum=0;
for(int i = 0; i<str.length(); i++)
{
     char ch = str.charAt(i);
     sum += (int) ch;
}
System.out.println(“Sum of ASCII values=”+sum);
```

XXXXX