String Handling in java

String Class

String is a sequence of characters. for e.g "Hello" is a string of 5 characters. In java, string is an immutable object which means it is constant and can cannot be changed once it has been created.

Java String Methods

Here are the list of the methods available in the Java String class. These methods are explained in the separate tutorials with the help of examples. ~~Links to the tutorials are provided below~~:

1. char charAt(int index): It returns the character at the specified index. Specified index value should be between 0 to length() -1 both inclusive. It throws IndexOutOfBoundsException if index<0 >= length of String.
2. boolean equals(Object obj): Compares the string with the specified string and returns true if both matches else false.
3. boolean equalsIgnoreCase(String string): It works same as equals method but it doesn't consider the case while comparing strings. It does a case insensitive comparison.
4. int compareTo(String string): This method compares the two strings based on the Unicode value of each character in the strings.
5. int compareToIgnoreCase(String string): Same as CompareTo method however it ignores the case during comparison.
6. boolean startsWith(String prefix, int offset): It checks whether the substring (starting from the specified offset index) is having the specified prefix or not.
7. boolean startsWith(String prefix): It tests whether the string is having specified prefix, if yes then it returns true else false.
8. boolean endsWith(String suffix): Checks whether the string ends with the specified suffix.
9. int hashCode(): It returns the hash code of the string.
10. int indexOf(int ch): Returns the index of first occurrence of the specified character ch in the string.
11. int indexOf(int ch, int fromIndex): Same as indexOf method however it starts searching in the string from the specified fromIndex.
12. int lastIndexOf(int ch): It returns the last occurrence of the character ch in the string.
13. int lastIndexOf(int ch, int fromIndex): Same as lastIndexOf(int ch) method, it starts search from fromIndex.
14. int indexOf(String str): This method returns the index of first occurrence of specified substring str.
15. int lastindexOf(String str): Returns the index of last occurrence of string str.
16. String substring(int beginIndex): It returns the substring of the string. The substring starts with the character at the specified index.
17. String substring(int beginIndex, int endIndex): Returns the substring. The substring starts with character at beginIndex and ends with the character at endIndex.

18. String concat(String str): Concatenates the specified string "str" at the end of the string.
19. String replace(char oldChar, char newChar): It returns the new updated string after changing all the occurrences of oldChar with the newChar.
20. boolean contains(CharSequence s): It checks whether the string contains the specified sequence of char values. If yes then it returns true else false. It throws NullPointerException of 's' is null.
21. String toUpperCase(Locale locale): Converts the string to upper case string using the rules defined by specified locale.
22. String toUpperCase(): Equivalent to toUpperCase(Locale.getDefault()).
23. public String intern(): This method searches the specified string in the memory pool and if it is found then it returns the reference of it, else it allocates the memory space to the specified string and assign the reference to it.
24. public boolean isEmpty(): This method returns true if the given string has 0 length. If the length of the specified Java String is non-zero then it returns false.
25. public static String join(): This method joins the given strings using the specified delimiter and returns the concatenated Java String
26. String replaceFirst(String regex, String replacement): It replaces the first occurrence of substring that fits the given regular expression "regex" with the specified replacement string.
27. String replaceAll(String regex, String replacement): It replaces all the occurrences of substrings that fits the regular expression regex with the replacement string.
28. String[] split(String regex, int limit): It splits the string and returns the array of substrings that matches the given regular expression. limit is a result threshold here.
29. String[] split(String regex): Same as split(String regex, int limit) method however it does not have any threshold limit.
30. String toLowerCase(Locale locale): It converts the string to lower case string using the rules defined by given locale.
31. public static String format(): This method returns a formatted java String
32. String toLowerCase(): Equivalent to toLowerCase(Locale. getDefault()).
33. String trim(): Returns the substring after omitting leading and trailing white spaces from the original string.
34. char[] toCharArray(): Converts the string to a character array.
35. static String copyValueOf(char[] data): It returns a string that contains the characters of the specified character array.
36. static String copyValueOf(char[] data, int offset, int count): Same as above method with two extra arguments – initial offset of subarray and length of subarray.
37. void getChars(int srcBegin, int srcEnd, char[] dest, int destBegin): It copies the characters of **src**array to the **dest** array. Only the specified range is being copied(srcBegin to srcEnd) to the dest subarray(starting fromdestBegin).
38. static String valueOf(): This method returns a string representation of passed arguments such as int, long, float, double, char and char array.
39. boolean contentEquals(StringBuffer sb): It compares the string to the specified string buffer.
40. boolean regionMatches(int srcoffset, String dest, int destoffset, int len): It compares the substring of input to the substring of specified string.
41. boolean regionMatches(boolean ignoreCase, int srcoffset, String dest, int destoffset, int len): Another variation of regionMatches method with the extra boolean argument to specify whether the comparison is case sensitive or case insensitive.
42. byte[] getBytes(String charsetName): It converts the String into sequence of bytes using the specified charset encoding and returns the array of resulted bytes.

43. byte[] getBytes(): This method is similar to the above method it just uses the default charset encoding for converting the string into sequence of bytes.
44. int length(): It returns the length of a String.
45. boolean matches(String regex): It checks whether the String is matching with the specified regular expression regex.
46. int codePointAt(int index):It is similar to the charAt method however it returns the Unicode code point value of specified index rather than the character itself.

1. Creating String ,using Constructors

```
public class FirstProgram
{
    public static void main(String[] args)
    {
      char ch[]={'a','b','c','d','e','f','g','h','i'};


        //creating string
        String s1=new String(ch);
        System.out.println(s1);  //ans is : abcdefghi

        String s2=new String(ch,2,4);
        System.out.println(s2);  //ans is:cdef

        byte b[] = {65, 66, 67, 68, 69, 70, 71};
        String s3 = new String(b);
        System.out.println(s3);  //ABCDEFG

        String s4 = new String(b, 2, 5);
        System.out.println(s4);  //CDEFG

        String s5 = new String("RAT");
        System.out.println(s5);  //RAT

        String s6=new String(s1);
        System.out.println(s6);  //abcdefghi
    }
}
```

2. toString() method

The java toString() method is used when we need a string representation of an object. It is defined in Object class. This method can be overridden to customize the String representation of the Object. ... Below is an example shown of the same program by Overriding the default Object toString() method

```
class Demo
{
    @Override
    public String toString()
    {
        return "toString() method called";
    }
}
public class DemoToString
{
   public static void main(String args[])
   {
        Demo d=new Demo();
        System.out.println(d);//it will not print the address,instead
toString() will be called
   }
}
//ans :toString() method called
```

3. length() method

```
public class DemoLength
{
   public static void main(String args[])
   {
        String str="road ahead technologies";
        //one way
        int len=str.length();
        System.out.println("Total length is:"+len);

        //second way
        System.out.println("length is:"+str.length());

        //third way
        System.out.println("road ahead technologies".length());
   }
}
//Ans is 23 in every line
```

including spaces.

used to calculate string length.

length - used to calculate array length.

4. Comparing String

```
public class ComparingString
{
   public static void main(String args[])
   {
        String s1=new String("RAT");
        String s2=new String("RAT");

        //following are two string literals having the same values
        String s3="RAT";  //literal
        String s4="RAT"; //literal

        System.out.println("------------ == operator------------------");
        // == operator will check the memory address
```

equals — content match
= = — Object match

```
        System.out.println(s1==s2);   //false
        System.out.println(s1==s3);   //false
        System.out.println(s3==s4);   //true

        System.out.println("------------equals() method----------------");

        // .equals() method will check the content
        System.out.println(s1.equals(s2));   //true
        System.out.println(s1.equals(s3));   //true
        System.out.println(s3.equals(s4)); //true
        System.out.println("RAT".equals("RAT"));   //true
        System.out.println("RAT".equals("rat"));   //false


        System.out.println("----------equalsIgnoreCase()-------------------");
        System.out.println("RAT".equalsIgnoreCase("rat"));   //true
        System.out.println("RAT".equalsIgnoreCase("ret"));   //false

        System.out.println("----------compareTo()------------------");
//It compares strings on the basis of Unicode value of each character in the
strings.
        String ss1="hello";
        String ss2="hello";
        String ss3="meklo";
        String ss4="hemlo";
        String ss5="flag";
        System.out.println(ss1.compareTo(ss2)); //0 because both are equal
        System.out.println(ss1.compareTo(ss3));
       //-5 because "h" is 5 times lower than "m"
        System.out.println(ss1.compareTo(ss4));
       //-1 because "l" is 1 times lower than "m"
        System.out.println(ss1.compareTo(ss5));
       //2 because "h" is 2 times greater than "f"
    }
}
```

5. startsWith() & endsWith()

Case sensitive

```
public class DemoStart_EndWith
{
   public static void main(String args[])
   {
        String str="road ahead technologies";
        System.out.println(str.startsWith("road")); //true
        System.out.println(str.startsWith("Road")); //false
        System.out.println(str.startsWith("technologies")); //false
        System.out.println(str.endsWith("technologies")); //true
        System.out.println(str.endsWith("gies")); //true
   }}
```

Extracting means single character nikalna

Extracting Character ① charAt ( ) ④ toCharArray ( )
② substring ( ) ⑤ getChars ( )
③ getBytes ( )

6. Extracting Characters and substrings

```
public class ExtractingChar
{
    public static void main(String args[])
    {
        String str="road ahead technologies";
        //Extracting a single char
        char ch=str.charAt(5); //extrating char at 5th positing
            System.out.println("extracted char is:"+ch);//ans is:a

        //Extracting substring from 5th index to last index
        String s1=str.substring(5);
        System.out.println(s1);  //ans is:ahead technologies


        //Extracting substring withing 5-15
        s1=str.substring(5,15);
        System.out.println(s1); //ans is:ahead tech

        //converting String into char array
        char ch1[]=str.toCharArray();
        for(char c1:ch1)
        {
            System.out.print(c1+",");
        }

        System.out.println();   //new line

        //converting String into bytes array
        byte b[]=str.getBytes();
        for(byte b1:b)
        {
            System.out.print(b1+",");
        }

        System.out.println("------------------------------");

        //extracting char array
        char[] target=new char[10];
        str.getChars(5,15,target,0);
        for(char t:target)
        {
            System.out.print(t+",");
        }
    }
}
```

7. Searching and replacing characters/strings

If a char does not found in a string it gives -1.

Case sensitive

```
public class Searching_Replacing
{
    public static void main(String args[])
    {
        String str="road ahead technologies";

        //getting the first occurrence of the char
        int i=str.indexOf('a'); //first occurrence of a in the string
        System.out.println("first occurrence is:"+i);  //2

        i=str.lastIndexOf('a');//last occurrence of a in the string
        System.out.println(("last occurrence is:"+i));//8

        System.out.println("-------------'replacing String'--------------");
        String s1="i cannot do it";
        System.out.println(s1);
        s1=s1.replace("cannot","can");
        System.out.println(s1);


        //replacing char
        System.out.println("-------------'replacing char'--------------");
        s1="poor";
        System.out.println(s1);
        s1=s1.replace('o','e');
        System.out.println(s1);
        s1=s1.replace("ee","rope");
        System.out.println(s1);


        System.out.println("-------------'trim method'--------------");
        String str1="  road  ahead   ";
        int len=str1.length();
        System.out.println("before:"+len+"\n"+str1+"tech");//without trim()

        str1=str1.trim();//trim will remove leading and trailing white spaces
        len=str1.length();
        System.out.println("after trim :"+len+ "\n" + str1 +"tech");
    }
}
```

remove unwanted space

trim function

```
Ans:
first occurrence is:2
last occurrence is:8
-------------'replacing String'--------------
i cannot do it
i can do it
-------------'replacing char'--------------
poor
peer
proper
-------------'trim method'--------------
```

String.join()
String.copyValueOf()
String.valueOf()

```
without trim length is:16
  road  ahead   tech
after trim length is:11
road  aheadtech
```

8. toUpperCase() & toLowerCase()

```
public class DemoUpperLower
{
   public static void main(String args[])
   {
       String str="If We Fail,Let Us Try again and again until we succeed";
       System.out.println(str.toLowerCase());
       System.out.println(str.toUpperCase());
   }
}
```

```
if we fail,let us try again and again until we succeed
IF WE FAIL,LET US TRY AGAIN AND AGAIN UNTIL WE SUCCEED
```

9. copyValueOf()

```
public class DemoCopyValueOf
{
    public static void main(String[] args)
    {
        char ch[] = {'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i'};
        //one way
        String s1 = new String(ch);
        System.out.println(s1);  //ans is : abcdefghi

        //another way
        String s2=String.copyValueOf(ch);
        System.out.println(s2);  //ans is: abcdefghi
    }
}
```

10. valueOf()

```
public class DemoValueof
{
    public static void main(String args[])
    {
        int value=9;
        String s1=String.valueOf(value); //it will convert the integer value
into string
        System.out.println(s1+10);//concatenating string with 10
    }}
```

11. concat()

```
public class DemoConcat
{
    public static void main(String args[])
    {
        String s1="beauty lies in the eye";
        s1=s1.concat(" of the beholder!!");
        System.out.println(s1);
    }
}
```

join two strings → split two strings

12. join() and split()

```
public class DemoJoin_Split
{
    public static void main(String args[])
    {
        //------------splitting--------------------------
        String str="Act as if it were impossible to fail";
        String item[]=str.split("\\s");  //splitting the string based on 
whitespace
        for(String s1:item)
        {
            System.out.println(s1);
        }

        System.out.println("--------------------  ------------");
        //-----------joining---------------------------
        String str2=String.join("-","live","the","coding","at","RAT");//joining
        System.out.println(str2);

    }
}
```

for white space

Escape Character \ \s - means white space

extra space.

StringBuffer Class - mutable

The **StringBuffer class** is used to create mutable string. It is same as String class except it is mutable and thread-safe.

13. StringBuffer's methods

```
public class DemoStringBuffer
{
    public static void main(String args[])
    {
        StringBuffer sb=new StringBuffer("core");
        System.out.println(sb);

        //length and capacity
```

String Builder Class is similar to StringBuffer Class
asynchronized speed fast not guaranteed result
Synchronized guaranteed result slow speed.

~~Sb = Sb.append~~ no need

no need to store anywhere i.e it proves that all the modification or changes will reflect in same Object

```
        System.out.println("length is:"+sb.length());   //length is:4
        System.out.println("capacity is:" + sb.capacity());//capacity is:20
                                        → length and capacity are different
        //append
        sb.append("java");//now original string is changed
        System.out.println(sb); //corejava
        //insert
        sb.insert(4,'-'); // index number, insert character.
        System.out.println(sb); //core-java → In this stage capacity will
                          remain same 20, it cannot be exceed until the
        //reverse         length is not more than capacity.
        sb.reverse();
        System.out.println(sb); //avaj-eroc
        System.out.println(sb.reverse()); //core-java

        //deleting a single char
        sb.deleteCharAt(4);
        System.out.println(sb);//corejava

        //delete
        sb.delete(4,8);
        System.out.println(sb); //core

        //setCharAt
        sb.setCharAt(2,'-'); //  '-' will replace the 2nd index character
        System.out.println(sb); //co-e

        //setLength()
        System.out.println("---------------setLength-------------------");
        StringBuffer sb1=new StringBuffer("road ahead technologies");
        System.out.println(sb1+":"+sb1.length());//road ahead technologies:23
        sb1.setLength(10);
        System.out.println(sb1+":"+sb1.length());//road ahead:10

        //ensure capacity
        System.out.println("---------ensureCapacity()--------------------");
        StringBuffer sbb=new StringBuffer("RAT");
        System.out.println("before capacity is:"+sbb.capacity()); //before
capacity is:19
        sbb.ensureCapacity(11);

    System.out.println("new capacity is:"+sbb.capacity()); //new capacity is:19
        sbb.ensureCapacity(50);

   System.out.println("new  capacity is:"+sbb.capacity()); //new capacity is:50
        sbb.ensureCapacity(51);

   System.out.println("new capacity is:"+sbb.capacity()); //new capacity is:102
    }}
```

capacity formula -

(older capacity * 2) + 2