



JAVA PROGRAMMING

Chap 5 : String Handling

- ▶ A string is a sequence of characters surrounded by double quotations.
- ▶ In a java programming language, a string is the object of a built-in class String.
- ▶ In the background, the string values are organized as an array of a character data type.
- ▶ **The string created using a character array** can not be extended.
- ▶ **It does not allow to append more characters after its definition, but it can be modified.**
- ▶ The Java String is immutable by default which means it cannot be changed.
- ▶ Whenever we change any string, a new instance is created.
- ▶ For mutable strings, you can use StringBuffer and StringBuilder classes.
- ▶ Syntax :


```
char[] name = {'J', 'a', 'v', 'a', ' ', 'T', 'u', 't', 'o', 'r', 'i', 'a', 'l', 's'};
```

```
// name[14] = '@'; //ArrayIndexOutOfBoundsException when trying to add more characters to the string
```

```
System.out.println(name);
```

```
name[4] = '-'; // modifying the string
```

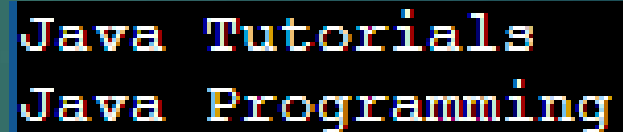
```
System.out.println(name);
```



```
Java Tutorials  
Java-Tutorials
```

- ▶ The String class defined in the package java.lang package.
- ▶ The String class implements Serializable, Comparable, and CharSequence interfaces.
- ▶ **Java String** class provides a lot of methods to perform operations on strings such as compare(), concat(), equals(), split(), length(), replace(), compareTo(), intern(), substring() etc.
- ▶ The string created using the String class can be extended.
- ▶ It allows us to add more characters after its definition, and also it can be modified.
- ▶ Syntax :

```
String name = "Java Tutorials";  
System.out.println(name);  
name = "Java Programming";  
System.out.println(name);
```



```
Java Tutorials  
Java Programming
```

Creating String object in Java

► In java, we can use the following two ways to create a string object.

- Using string literal
- Using String constructor

► Syntax :

```
String title = "Java Tutorials"; // Using literals
```

```
String name = new String("Java Programming"); // Using constructor
```

```
System.out.println(title);
```

```
System.out.println(name);
```

```
char ch[]={'S','t','r','i','n','g','s'};
```

```
String s=new String(ch); //converting char array to string using constructor
```

```
System.out.println(s);
```

```
Java Tutorials
Java Programming
Strings
```

Why are String objects immutable?

- ▶ A String is an unavoidable type of variable while writing any application program.
- ▶ String references are used to store various attributes like username, password, etc.
- ▶ In Java, String objects are immutable. Immutable simply means unmodifiable or unchangeable.
- ▶ Once String object is created its data or state can't be changed but a new String object is created.
- ▶ Let's try to understand the concept of immutability by the example given below:

```
String s="Java";
```

```
s.concat(" Programming"); //concat() method appends the string at the end
```

```
System.out.println(s); //will print Java because strings are immutable objects
```



- ▶ Now it can be understood that Java is not changed but a new object is created with Java Programming and s reference variable still refers to "Java" and not to "Java Programming". That is why String is known as immutable.
- ▶ But if we explicitly assign it to the reference variable, it will refer to "Java Programming" object. Please notice that still "Java" object is not modified, only that the reference variable refers to the new object.

```
String s="Java";
```

```
s=s.concat(" Programming");
```

```
System.out.println(s);
```



String Handling Methods

Method	Description	Return Value
charAt(int)	Finds the character at given index	char
length()	Finds the length of given string	int
compareTo(String)	Compares two strings	int
compareToIgnoreCase(String)	Compares two strings, ignoring case	int
concat(String)	Concatenates the object string with argument string.	String
contains(String)	Checks whether a string contains sub-string	boolean
contentEquals(String)	Checks whether two strings are same	boolean
equals(String)	Checks whether two strings are same	boolean
equalsIgnoreCase(String)	Checks whether two strings are same, ignoring case	boolean
startsWith(String)	Checks whether a string starts with the specified string	boolean
endsWith(String)	Checks whether a string ends with the specified string	boolean
getBytes()	Converts string value to bytes	byte[]

String Handling Methods

hashCode()	Finds the hash code of a string	int
indexOf(String)	Finds the first index of argument string in object string	int
lastIndexOf(String)	Finds the last index of argument string in object string	int
isEmpty()	Checks whether a string is empty or not	boolean
replace(String, String)	Replaces the first string with second string	String
replaceAll(String, String)	Replaces the first string with second string at all occurrences.	String
substring(int, int)	Extracts a sub-string from specified start and end index values	String
toLowerCase()	Converts a string to lower case letters	String
toUpperCase()	Converts a string to upper case letters	String
trim()	Removes whitespace from both ends	String
toString(int)	Converts the value to a String object	String
split(String)	splits the string matching argument string	String[]
intern()	returns string from the pool	String
join(String, String, ...)	Joins all strings, first string as delimiter.	String

String Handling Methods

```
public class Main {
public static void main(String[] args) {
String title = "Java Tutorials";
String Name = "www.javaprogramming.com";
System.out.println("Length of title: " + title.length());
System.out.println("Char at index 3: " + title.charAt(3));
System.out.println("Index of 'T': " + title.indexOf('T'));
System.out.println("Last index of 'a': " +
title.lastIndexOf('a'));
System.out.println("Empty: " + title.isEmpty());
System.out.println("Ends with '.com': " +
Name.endsWith(".com"));
System.out.println("Equals: " + Name.equals(title));
System.out.println("Sub-string: " + Name.substring(9, 14));
System.out.println("Upper case: " + Name.toUpperCase());
System.out.println("Upper case: " + title.toLowerCase());
String s=" Java ";
System.out.println("Before Trimming :"+s);
System.out.println("After trimming :"+s.trim());
System.out.println("Replacing a by z in " + title + " we get "
+ title.replace("a","z"));
}
}
```

```
Length of title: 14
Char at index 3: a
Index of 'T': 5
Last index of 'a': 11
Empty: false
Ends with '.com': true
Equals: false
Sub-string: rogra
Upper case: WWW.JAVAPROGRAMMING.COM
Upper case: java tutorials
Before Trimming : Java
After trimming :Java
Replacing a by z in Java Tutorials we get Jzvvz Tutorizls
```


String Compare

- ▶ We can compare String in Java on the basis of content and reference.
- ▶ It is used in authentication (by equals() method), sorting (by compareTo() method), reference matching (by == operator) etc.
- ▶ There are three ways to compare String in Java:
 - ▶ By Using equals() Method
 - ▶ By Using == Operator
 - ▶ By compareTo() Method
- ▶ The String class **equals() method** compares the original **content** of the string. It compares values of string for equality. String class provides the following two methods:
 - ▶ public boolean equals(Object another) compares this string to the specified object.
 - ▶ public boolean equalsIgnoreCase(String another) compares this string to another string, ignoring case.

Eg 1: String s1="Java";
String s2="Java";
String s3=new String("Java");
String s4="Python";
System.out.println(s1.equals(s2)); // o/p will be true
System.out.println(s1.equals(s3)); // o/p will be true
System.out.println(s1.equals(s4)); // o/p will be false

```
Eg 2: String s1="Java";  
String s2="JAVA";  
System.out.println(s1.equals(s2)); //false  
System.out.println(s1.equalsIgnoreCase(s2));  
//true
```

String Compare

- ▶ The **== operator** compares **references** not values.

Eg 1: `String s1="Java";`
`String s2="Java";`
`String s3=new String("Java");`
`System.out.println(s1==s2);` // o/p will be true (because both refer to same instance)
`System.out.println(s1==s3);` // o/p will be false(because s3 refers to another instance created)

- ▶ The String class **compareTo() method** compares values lexicographically and returns an integer value that describes if first string is less than, equal to or greater than second string.
- ▶ Suppose s1 and s2 are two String objects. If:
 - ▶ `s1 == s2` : The method returns 0.
 - ▶ `s1 > s2` : The method returns a positive value.
 - ▶ `s1 < s2` : The method returns a negative value.

Eg : `String s1="Java";`
`String s2="Java";`
`String s3="Jp";`
`System.out.println(s1.compareTo(s2));` //0
`System.out.println(s1.compareTo(s3));` //-15 (because s1<s3 and "a" is 15 times lower than "p")
`System.out.println(s3.compareTo(s1));` //15 (because s3 > s1 and "p" is 15 times greater than "a")

String Concatenation

- ▶ In Java, String concatenation forms a new String that is the combination of multiple strings. There are two ways to concatenate strings in Java:
 - ▶ By + (String concatenation) operator
 - ▶ By concat() method

Eg 1:

```
String s="Java"+" Programming";
```

```
String s1=50+30+" JP "+40+40;
```

```
System.out.println(s); // Java Programming
```

```
System.out.println(s1); //80 JP 4040
```

Note: After a string literal, all the + will be treated as string concatenation operator.

```
String s2=" by Oracle";
```

```
String s3=s.concat(s2);
```

```
System.out.println(s3); // Java Programming by Oracle
```

StringBuffer Class

- ▶ Java StringBuffer class is used to create mutable (modifiable) String objects.
- ▶ A String that can be modified or changed is known as mutable String. StringBuffer and StringBuilder classes are used for creating mutable strings.
- ▶ The StringBuffer class in Java is the same as String class except it is mutable i.e. it can be changed.
- ▶ Important Constructors of StringBuffer Class
 - ▶ `StringBuffer()` : It creates an empty String buffer with the initial capacity of 16. If the number of the character increases from its current capacity, it increases the capacity by $(\text{oldcapacity} * 2) + 2$.
 - ▶ `StringBuffer(String str)` : It creates a String buffer with the specified string.
 - ▶ `StringBuffer(int capacity)` : It creates an empty String buffer with the specified capacity as length.

► Important methods of StringBuffer class

Method	Description
append(String s)	It is used to append the specified string with this string. The append() method is overloaded like append(char), append(boolean), append(int), append(float), append(double) etc.
insert(int offset, String s)	It is used to insert the specified string with this string at the specified position. The insert() method is overloaded like insert(int, char), insert(int, boolean), insert(int, int), insert(int, float), insert(int, double) etc.
replace(int startIndex, int endIndex, String str)	It is used to replace the string from specified startIndex and endIndex.
delete(int startIndex, int endIndex)	It is used to delete the string from specified startIndex and endIndex.
reverse()	is used to reverse the string.
capacity()	It is used to return the current capacity.
ensureCapacity(int minimumCapacity)	It is used to ensure the capacity at least equal to the given minimum.
charAt(int index)	It is used to return the character at the specified position.
length()	It is used to return the length of the string i.e. total number of characters.
substring(int beginIndex)	It is used to return the substring from the specified beginIndex.
substring(int beginIndex, int endIndex)	It is used to return the substring from the specified beginIndex and endIndex.

StringBuffer Class Example

```
public class Main {  
    public static void main(String[] args) {  
        StringBuffer sb=new StringBuffer("Java");  
        System.out.println(sb.append(" Tutorials"));    //original string Java is changed to Java Tutorials  
        System.out.println(sb.insert(5, "Programming ")); // inserts the given String in original string at the given position  
        System.out.println(sb.replace(0,4, "HTML"));    // replaces the given String from the specified beginIndex and  
        endIndex.  
        System.out.println(sb.delete(17,26));    //deletes the String from the specified beginIndex to endIndex.  
        System.out.println(sb.reverse());    // reverses the String  
    }  
}
```

```
Java Tutorials  
Java Programming Tutorials  
HTML Programming Tutorials  
HTML Programming  
gnimmargorP LMTH
```

- WAP to convert a string into an array of characters and display each character with its index

```
import java.util.*;
public class Main{
    public static void main(String[] args) {
        String str;
        int i,n;
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the string");
        str=sc.nextLine();
        n=str.length();
        char c[]=new char[n];
        c=str.toCharArray();
        for(i=0;i<=n-1;i++)
        {
            System.out.println(i+"\t"+c[i]);
        }
    }
}
```

```
Enter the string
Hello world !!!
0          H
1          e
2          l
3          l
4          o
5
6          w
7          o
8          r
9          l
10         d
11
12         !
13         !
14         !
```


► WAP to count number of vowels, consonants, digits and blank spaces in a string

```
import java.util.*;
public class Main{
public static void main(String[] args) {
String str;
int i,n,v=0,co=0,d=0,bs=0;
Scanner sc=new Scanner(System.in);
System.out.println("Enter the string");
str=sc.nextLine();
n=str.length();
char c[]=new char[n];
c=str.toCharArray();
for(i=0;i<=n-1;i++)
{
if(c[i]=='a' || c[i]=='e' || c[i]=='i' || c[i]=='o' || c[i]=='u' || c[i]=='A' ||
c[i]=='E' || c[i]=='I' || c[i]=='O' || c[i]=='U')
v++;
else if(c[i]==' ')
bs++;
else if(c[i]>='0' && c[i]<='9')
d++;
else if((c[i]>='a' && c[i]<='z') || (c[i]>='A' && c[i]<='Z'))
co++;
}
System.out.println("No. of vowels : " + v + "\nNo. of Consonants : " +
co + "\nNo. of digits : " + d + "\nNo of blank spaces : " + bs);
}}
```

```
Enter the string
Hello how r u 123
No. of vowels : 4
No. of Consonants : 6
No. of digits : 3
No of blank spaces : 4
```

Programming Practice Questions

- ▶ WAP in Java to perform the following String Operations.
 1. Create a string instance using String and String Buffer class each.
 2. Check the length and capacity of String and String Buffer objects
 3. Reverse the contents of a string and convert the resultant string in Upper Case.
 4. Append the second string to above resultant string.
 5. Extract a substring from resultant string.
- ▶ WAP in Java to implement run-length encoding.
- ▶ Write a Java method to extract all digits from a given string and return them as a new string.