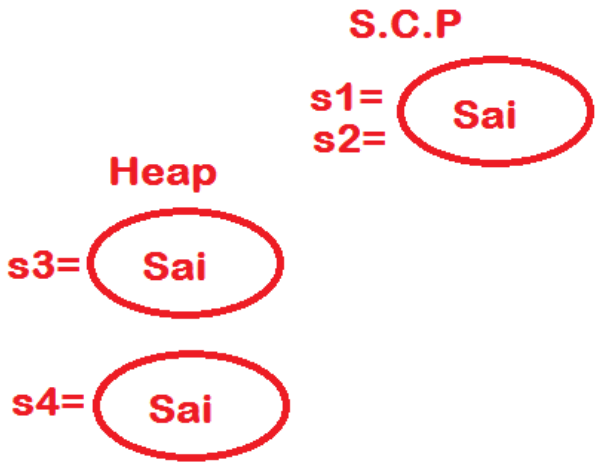
String Manipulation

- **Using String Class**
  - **String is predefined class From java.lang package**
  - **String is an immutable object i.e content string object can't be modified, by chance if you are trying to make the any changes the string object then it will be creating new string object with modified content.**
  - **String is a "Wrapper Class"**
  - **String is implemented by "java.lang.Comparable(i)"**
  - **String is implemented by java.io.Serailizable [Files]**
  - **String not synchronized, thus string is not thread safety [multithreading]**
- **String Object can be created with or without constructor**
  - **Eg: String s="Shashi";**
    - **While creating String Object without using constructor then JVM will ensure does any String object is existed or not in the String constant pool with the specified content. If String object is existed with specified content then it won't create new String object rather will be refer to the existed object**

- **String s=new String("Shashi");**
  - **While creating string object using constructor then JVM will not ensure the existence of String Object in the "Heap" organization rather it will create new String Object for every time.**

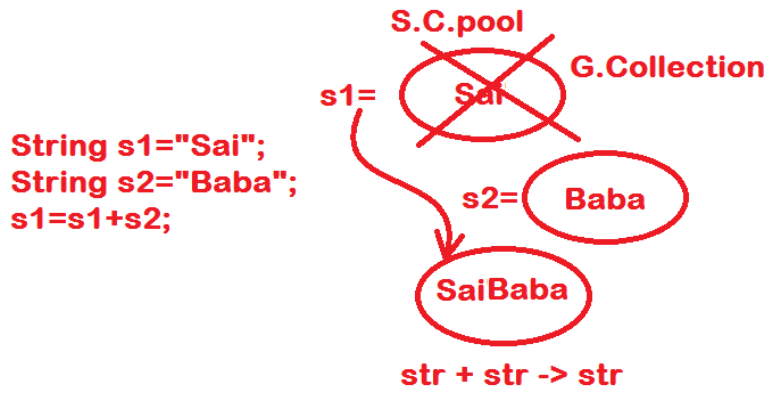
```
String s1="Sai";
String s2="Sai";

String s3=new String("Sai");
String s4=new String("Sai");
```



Example:

```
class Sample
{
	public static void main(String args[ ])
	{
		String s1="Sai";
		String s2="Baba";
		s1=s1+s2;
		System.out.println("Result is : "+s1);
	}
}
```



Exampl 2:

```
//String2.java
class String2
{
	public static void main(String args[ ])
	{
		System.out.println(10+20+30+"Shashi"); //60Shashi
		System.out.println(10+20+"Shashi"+30); //30Shashi30
		System.out.println(10+"Shashi"+20+30); //10Shashi2030
		System.out.println("Shashi"+10+20+30); //Shashi102030
	}
}
```

```
//String + String -> String [concatenation]
// int + int -> int [Addition]
// String + xxx --> String [concatenation]
// xxx + String --> String [Concatenation]
```

Constructor for String:

String()

```
String s=new String();
System.out.println(s);   //nothing
```

String(String)

```
String s1=new String("Ramesh");
System.out.println(s1);  //Ramesh
```

String(char[ ])

```
char[ ] x={'w','e','l','c','o','m','e'};
String s2=new String(x);
System.out.println(s2); //welcome
```

String(char[ ],int spos,int no.of.char)

```
String s3=new String(x,3,4);
System.out.println(s3);   //come
```

Methods:

public int length( ); //non static method

IQ: length vs length( )

- **Length is a property of an array. It is always holding the size of an array.**
- **length() of is predefine method in string class it will return length of string object [No.of.character along with spaces]**

```
String s=”welcome”;
int nc=s.length();
System.out.println(“No.of.char are : “+nc); //7
```

.public String toUpperCase( )

- **It will return new String Object by converting them into upper case letters**

.public String toLowerCase()

- **It will return new String Object by converting String into lower case letters**

Example:

```
//String4.java

class String4
{
	public static void main(String args[ ])
	{
		String s="WeLComE";
		String uc=s.toUpperCase();
		String lc=s.toLowerCase();

		System.out.println("Actual is : "+s);
		System.out.println("Ucase is : "+uc);
		System.out.println("Lcase is : "+lc);
	}
}
```

```
String s="WeLComE";
  String uc = s.toUpperCase( );
String lc=s.toLowerCase( );
System.out.println(s);  //WeLComE
System.out.println(uc); //WELCOME
System.out.println(lc); //welcome
```

.public boolean equals(Object)

- It is overloaded of java.lang.Object class
- equals() return "true" if the content of both string Objects are same otherwise it will returns "False"

IQ: == vs equals( )

Ans : == is an operator which is used to compare the hashcode of two string objects.where as equals() is method of String class will compare content of two String Object

Example :

```
//String5.java

class String5
{
    public static void main(String args[ ])
    {
            String s1="Sai";
            String s2="Sai";

            if(s1==s2)
                    System.out.println("Hcode of s1 and s2 are same");
            else
                    System.out.println("Hcode of s1 and s2 are not same");
```

```
        if(s1.equals(s2))
                System.out.println("s1 and s2 content are same");
        else
                System.out.println("s1 and s2 content are not same");

        String s3=new String("Sai");
        String s4=new String("Sai");

        if (s3==s4)
          System.out.println("Hcode of s3 and s4 are same");
        else
                System.out.println("Hcode of s3 and s4 are not same");

        if(s3.equals(s4))
                System.out.println("s3 and s4 content are same");
        else
                System.out.println("s3 and s4 content are not same");
    }
}
```

public boolean equalsIgnoreCase(String):

IQ: equals() vs equalsIgnoreCase()

Ans : both are meant for comparing content of two string objects but equals() compare content and also their cases. Where as equalsIgnoreCase( ) compare only content but not their cases

Example:

//String6.java

```
class String6
{
    public static void main(String args[ ])
    {
            String s1="SHASHI";
            String s2="shashi";

            //if(s1.equals(s2))

            if(s1.equalsIgnoreCase(s2))
                    System.out.println("Both are same");
            else
                    System.out.println("Both are not same");

    }
}
```

public String replace(String old,String new) :

- **it will return new String Object by replacing oldstring with new string**

```
String s=”Have a nice Day”;
String s2=s.replace(“nice”,”good”);
System.out.println(s2); //Have a good Day
```

public String trim( );

- **It will return string object by erasing empty spaces existed either of side of string content**

Example:

```
//Trim.java
class Trim
{
        public static void main(String args[ ])
        {
      String d=" shashi ";  // d is the Data from Database
          String data=args[0];  // data is from user

      d=d.trim();
          if(data.equalsIgnoreCase(d))
                System.out.println("Valid User ");
          else
                System.out.println("Invalid User ");
        }
}
//javac Trim.java
//java Trim Shashi
```

Example:

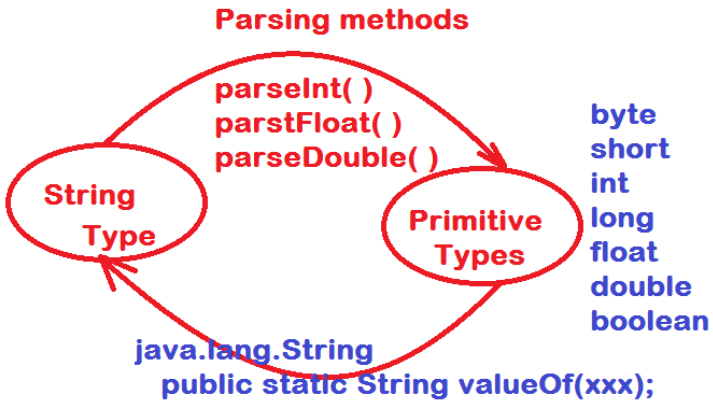
//Replace.java

```
class Replace
{
    public static void main(String args[ ])
    {
            String s=" W EL   C   O   M E      ";
            System.out.println("Data : "+s);

            String s2=s.replace(" ","");
            System.out.println("Result is : "+s2);
    }
}
```

public static String valueOf(xxx):

- It return String object by converting primitive value
- Primitive → String type Conversion



Example:

```
//Demo.java
class Demo
{
	public static void main(String args[ ])
	{
		String s="100";
		int i=Integer.parseInt(s);
		System.out.println("i val is : "+i);

		int a=200;
		//String b=a;
		String b=String.valueOf(a);
		System.out.println("b val is : "+b);
	}
}
```

public String substring(int sp,int ep):

```
String s="W E L C O M E";
String ss=s.substring(1,3);
System.out.println("Result is : "+ss);  //EL
```

Example:

```
class Substr
{
	public static void main(String args[ ])
	{
		String s="WELCOME";
		String ss=s.substring(1,4);   // between 1 and 3
```

```
            System.out.println("Actual String : "+s);
            System.out.println("SubString is : "+ss);
      }
}
```

public String[ ] split(String):

```
   //SplitDemo.java

class SplitDemo
{
      public static void main(String args[ ])
      {
            String s="welcome,to,sssit,kphb,hyd";
            System.out.println("Actual string : "+s);

            String[ ] words=s.split(",");
            for(String word:words)
              {
                  System.out.println(word);
              }
      }
}
```

public char charAt(index):

```
//CharAt.java

class CharAt
{
      public static void main(String args[ ])
      {
        String s="WELCOME";
        char fc=s.charAt(0);
        System.out.println("First Char : "+fc);

        char lc=s.charAt(6);
        System.out.println("Last char : "+lc);
      }
}
```

➢ Using StringBuffer Class

- StringBuffer is the predefined class from java.lang package
- It is not a wrapper class
- It implemented by java.io.Serializable(i) but not by java.lang.Comparable(i)
- It is mutable i.e content of StringBuffer object can be editable
- It is synchronized thus StringBuffer is Thread Safety .

Constructor:

- **StringBuffer()**
  - **StringBuffer sb=new StringBuffer():**
    - **It will create an empty StringBuffer Object with default capacity is 16 bytes**

- **StringBuffer(int):**
  - **StringBuffer sb=new StringBuffer(100);**
    - **It will create an empty StringBuffer object with specified capacity in bytes**

- **StringBuffer(String):**
  - **StringBuffer sb=new StringBuffer("Shashi");**
    - **It will create a StringBuffer object with specified string , But in this case capacity of StringBuffer object will be default capacity + length of string**

Methods:

public int length( ):

it will return no.of.characters are existed in the StringBuffer Object

public int capacity():

It will return capacity of StringBuffer object

Example:

```
//StringBEx1.java
class StringBEx1
{
	public static void main(String args[ ])
	{
		StringBuffer sb=new StringBuffer("shashi");
		System.out.println("Data is : "+sb);
		System.out.println("Capacity is : "+sb.capacity()); //16+6
		System.out.println("Length is : "+sb.length()); //6
	}
}
```

public StringBuffer append(xxx):

It is an overloaded method. It is used append (add at end of StringBuffer object) with specified content

public String insert(int,xxx):

It will return StringBuffer Object by inserting the specified value at the specified index.it is also an overloaded method

public String deleteCharAt(int):

It will return StringBuffer Object by deleting char existed @ specified index

public StringBuffer replace(int,int,String):

It will return StringBuffer Object by replacing no.of. char from the specified position with specified String

public StringBuffer reverse():

It will return StringBuffer Object by reversing the content of StringBuffer Object

Example:

//StringBE2.java

```
class StringBE2
{
	public static void main(String args[ ])
	{
		StringBuffer sb=new StringBuffer("Shashi");
		System.out.println("Data is : "+sb); //Shashi

		sb.append("Chinni"); //ShashiChinni
		System.out.println("After Append : "+sb);

		sb.insert(6,'-');
		System.out.println("After Insert : "+sb);

		sb.deleteCharAt(6);
		System.out.println("After DelChar : "+sb);

		sb.replace(0,6,"Khanna");
		System.out.println("After Replace : "+sb);

		sb.reverse();
		System.out.println("Revers : "+sb);

	}
}
```

java.lang.String
public boolean equals(Object):

java.lang.StringBuffer
public boolean equals(Object):

Imp: equals() of String class is overridden method of java.lang.Object it will compare the content of two string object . where as equals() of StringBuffer is not overridden method of Object class it is used to compare hashcode of two StringBuffer Object.

//StringBE3.java

```
class StringBE3 {
	public static void main(String args[ ])
	{
		StringBuffer sb1=new StringBuffer("shashi");
		StringBuffer sb2=new StringBuffer("shashi");

		if(sb1.equals(sb2))
			System.out.println("Hcode of Both are same");
		else
			System.out.println("Hcode of Both are not same");
	}
}
```

```
//Stringpol.java
class Stringpol
{
        public static void main(String args[ ])
        {
                java.util.Scanner s=
                        new java.util.Scanner(System.in);
                System.out.println("Enter any String ");
                String data=s.next();   // data=chinni

                //reverse the String
                StringBuffer sb=new StringBuffer(data);  //sb="chinni"
                                sb.reverse(); //sb="innihc"

                //to convert StringBuffer Object to String()
                  //public String toString()
                        String rs=sb.toString();

          //compare original string with reversed string
          if(data.equals(rs))
                  System.out.println("Pol_String");
          else
                  System.out.println("Not Pol_String ");
        }
}
```

➢ Using StringBuilder Class

- StringBuiler is the predefined class from java.lang package
- It is not a wrapper class
- It is implemented by java.io.Serializable(i) but not by java.lang.Comparable(i)
- It is mutable
- It is not synchronized thus StringBuilder is not Thread Safety .

Constructor:

StringBuilder()

```
StingBuilder sb=new StringBuilder();
```

StringBuilder(int)

```
StringBuilder sb=new StringBuilder(100);
```

StringBuilder(String)

```
StringBuilder sb=new StringBuider("chinni");
```

Note: All the methods of StringBuffer are the methods of StringBuilder but difference is StringBuffer methods are synchronized where as StringBuilder methods are not synchronized .

➢ Using StringTokenzier Class

- StringTokenizer is the predefined class from java.util package
- It is used to divide the String Object into tokens [small piece] at the specified delimiter
- Default delimiter is space

Constructor:

StringTokenizer(String):

Eg:StringTokenizer st=

new StringTokenizer("welcome to sssit");

- **It will divide the given string into tokens at the default delimiter and those token can manipulated from StringTokenizer**

StringTokenizer(String data,String delimiter):

StringTokenizer st=new StringTokenizer("welcome,to,sssit.kphb.hyd","," ,");

Methods:

java.util.StringTokenizer

- **public int countTokens( ):**
  - **It will return no.of.Tokens are existed in the StringTokenizer Collection**

- **public boolean hasMoreTokens():**
  - **It will returns true if the StringTokenizer collection has tokens otherwise it will return false**

- **public String nextToken():**
  - **It will return next Token from the StringTokenizer Collection**

Example:

//StringTK1.java

```
import java.util.StringTokenizer;

class StringTK1
{
        public static void main(String args[ ])
        {
                StringTokenizer st=
                        new StringTokenizer("welcome to sssit kphb hyd");

                int nt=st.countTokens();
                System.out.println("No.of.Tokens are : "+nt);

                while( st.hasMoreTokens() )
                {
                String token=st.nextToken();
         System.out.println
                ( token.toUpperCase()+" .... "+token.length()) ;
                }
        }
}
```