1. length()

syntax:

```
int length();
```

length() method of String class is used to count the number of characters of invoking String object. It returns the length of string.

```
class Example1
{
public static void main (String arg[ ])
{
        String s1 = "KP Solutions";
        int l = s1.length();
        System.out.println("Length="+l);
}
}
```

output :- Length=12

2. **charAt()**

syntax :-

```
char charAt ( int index);
```

charAt ( ) is standard inbuilt method of String class, it is used to find character at a given index number from the string of invoking object.

```
class Example1
{
        public static void main (String arg[ ])
        {
                String s1 = "KP Solutions";
                char   ch = s1.charAt (1);
                System.out.println(ch);
                System.out.println(s1.charAt(0));
        }
}
```

output :-

```
P
K
```

3. **indexOf()**

syntax :-

```
int indexOf(char ch);
```

indexOf() is standard inbuilt method of String class. It used to find the position of given character, if the given character is present in invoking string then it returns index number of the given character otherwise returns -1;

```
class Example
{
    public static void main ( String arg[ ])
    {
    String s1 = "KP Solutions";
    int n =s1.indexOf ('P');
    System.out.println(n);
    System.out.println(s1.indexOf('u') );
    System.out.println(s1.indexOf('b'));
    System.out.println(s1.indexOf('r'));
}
}
```

output:- 1

6

-1

-1

4. lastIndexOf ( )

Syntax

```
int lastIndexOf(char ch);
```

lastIndexOf( ) is standard inbuilt method of String, it is used to search a given character (ch) from the last index number of invoking object if the given character is present then it returns its respective index number otherwise returns – 1;

```
class Example 1
{
        public static void main (String arg[ ] )
        {
        String s1="KP Solutions";
        int n=s1.lastIndexOf('o');
        System.out.println (n);
        System.out.println (s1.lastIndexOf('n'));
        System.out.println (s1.lastIndexOf('b'));
        System.out.println (s1.lastIndexOf('r'));
        }
}
```

output :-

```
9
10
-1
-1
```

5. **startsWith( )**

Syntax

```
boolean startsWith(String str);
```

startsWith ( ) is standard method of String, it is used to compare the given string with the starting characters of invoking object, if the given string is matched with the starting characters of invoking object then it will return true otherwise it will return false it is case sensitive method.

```
class Example
{
        public static void main (String arg[ ])
        {
        String s1 = "KP Solutions";
        boolean b = s1.startsWith("KP");
        System.out.println(b);
        System.out.println(s1.startsWith("tions"));
        System.out.println (s1.startsWith("sita"));
        System.out.println (s1.startsWith("KP Solutions"));
        }
}
```

output :-

```
true
false
false
true
```

6. **endsWith ( )**

Syntax

```
boolean endsWith (String str);
```

endsWith ( ) is standard inbuilt method of String, it is used to compare given string with the last characters of invoking object, if the given string is matched with the last characters of invoking string then it will return true otherwise return false.

```
class Example
{
        public static void main(String arg[ ] )
        {
        String s1 = "KP Solutions";
        boolean b = s1.endsWith("Solutions");
        System.out.println(b);
        System.out.println(s1.endsWith("tions"));
        System.out.println (s1.endsWith("sita"));
        System.out.println (s1.endsWith("KP Solutions"));
        }
}
```

output :-

```
true
true
false
true
```

7. **equals ( )**

Syntax :

```
boolean equals (String str);
```

equals( ) method of String class is used to compare given string with the string of invoking object if both the strings are equal then it will return true otherwise returns false.

```
class Example
{
    public static void main(String arg[ ])
    {
        String s = "KP Solutions";
        boolean b = s.equals ("KP Solutions");
        System.out.println(b);
        System.out.println(s.equals("KP"));
        System.out.println(s.equals("Solutions"));
        System.out.println(s.equals("Problem"));
        System.out.println(s.equals("Answer"));
    }
}
```

Output:

```
true
false
false
false
false
```

8. **equalsIgoreCase ( )**

Syntax

```
boolean equalsIgnoreCase (String str);
```

equalsIgnoreCase ( ) method of String is used to compare given string with the string of invoking object by ignoring case. If given string is matched with the string of invoking object them it will return true otherwise returns false.

```
class Example
{
public static void main (String arg[ ] )
{
        String s = "KP Solutions";
        boolean b = s.equalsIgnoreCase("KP Solutions");
        System.out.println(b);
        System.out.println(s.equalsIgnoreCase("kp solutions"));
        System.out.println(s.equalsIgnoreCase("Solutions"));
        System.out.println(s.equalsIgnoreCase("Problem"));
        System.out.println(s.equalsIgnoreCase("Answer"));
 }
 }
```

output :-

```
true
true
false
false
false
```

9. **toUpperCase( ) and toLowerCase ( )**

Syntax

```
String toUpperCase ( );
String toLowerCase ( );
```

toUpperCase() and toLowerCase() methods of String class is used to convert string of invoking object into upper case and lower case letters respectively and returns the resultant string.

```
class Example 1
{
public static void main ( String arg[ ])
{
String s1 = "KP Solutions";
String s2=s1.toUpperCase( );
System.out.println(s1);
System.out.println(s2);
System.out.println(s1.toLowerCase());
}
}
```

output :-

```
KP Solutions
KP SOLUTIONS
kp solutions
```

10. trim ( )

Syntax

```
String trim ( );
```

trim() method of String class is used to remove the prefix and postfix spaces of invoking string and returns the resultant string.

```
class Example 1
{
        public static void main (String arg[ ])
        {
        String s1 = " KP Solutions ";
        System.out.println(s1);
        String s2 = s1.trim();
        System.out.println(s2);
        }
}
```

output

```
KP Solutions
KP Solutions
```

11. **substring( )**

Syntax :-

```
String substring (int index);
String substring(int startIndex,int endIndex);
```

Substring() is an overloaded method, available in two version, first version of substring ( ) method of String class is used to retrieve part of string from the given index number to last index number and returns the resultant string. Second version of substring() method is used to retrieve part of string from the invoking object.

```
class Example 1
{
        public static void main (String arg[ ] )
        {
        String s1 ="KP Solutions";
        String s2 =s1.substring(1);
        String s3=s1.substring(1,4);
        System.out.println(s1);
        System.out.println(s2);
        System.out.println(s3);
        }
}
```

output :-

```
KP Solutions
P Solutions
P S
```

String Constructors

```
class MakeString
{
  public static void main(String args[])
  {
     char c[] = { 'J', 'a', 'v', 'a' };
     String s1 = new String(c);
     String s2 = new String(s1);
    System.out.println(s1);
    System.out.println(s2);
  }
}
```

o/p:- Java
Java

Example 2

```
class Example
{
 public static void main(String args[])
 {
    byte ascii[] = {65, 66, 67, 68, 69, 70};
    String s1 = new String(ascii);
    System.out.println(s1);
    Syst
     String s2 = new String(ascii, 2, 3);
     System.out.println(s2);
   }
 }
```

o/p:- ABCDEF
CDE.

getChars()

syntax:-

```
void getchars(int sourcestart, int sourceEnd,
              char target[], int targetstart);
```

ex.

```
class Example
{
public static void main(String args[])
 {
   String S="This is a Demo of getchars Method";
    int starts=10;
    int end=14;
    char buf[]=new char[end-start];
     S.getchars(start, end, buf, 0);
    System.out.println(buf);
 }
}
```

o/p:- Demo

toCharArray()

syntax:-

```
char[] toCharArray();
```

It is used to converts all the character in a string object into character array.
It returns an array of characters for entire string. This can be achieved by getChars() method.

concat()

syntax:

```
String concat (String str);
```

ex:

```
class Example
{
public static void main (String args[])
{
  String s1= "one";
  String s2 =s1.concat("two");  // s2 = s1 + "two"
  System.out.println(s2);
 }
}
```

o/p:- onetwo

StringBuffer contructors

```
StringBuffer();
StringBuffer.(int size);
StringBuffer (String str);
```

length() and capacity()

```
class Example
{
public static void main (String args[])
{
  StringBuffer sb =new StringBuffer("Hello");
  System.out.println("buffer =" +sb);
  System.out.println("length =+sb.length());
  System.out.println("capacity =" +sb.capacity());
 }
}
```

o/p:- buffer = Hello
length = 5
capacity = 21 ; Because room for 16 additional characters are automatically added.

ensureCapacity()

Syntax:-

```
void ensureCapacity(int capacity);
```

Here capacity specifies the size of buffer

setLength()

Syntax:-

```
void setLength(int length);
```

charAt() and setCharAt()

Syntax:-

```
char charAt(int where);
void setcharAt(int where, char ch);
```

```
class Example
{
  public static void main(String args[])
  {
    StringBuffer sb = new StringBuffer("Hello");
    System.out.println("Buffer Before =" +sb);
    System.out.println("charAt(1) before =" +
                        sb.charAt(1));
    sb.setcharAt(1,'i');
    sb.setLength(2);
    System.out.println("Buffer after =" +sb);
    System.out.println("charAt(1) after =" +
                        sb.charAt(1));
  }
}
```

o/p:- buffer before = Hello
charAt(1) before = e
buffer after = Hi
charAt(1) after = i

append()

StringBuffer append (String str)
StringBuffer append (int num)
StringBuffer append (object obj)

```
class Example
{
public static void main (String args[])
{
    String s;
    int a = 42;
    StringBuffer sb = new StringBuffer (40);
    s = sb.append ("a = ").append (a).append ("!").
                                    toString();
    System.out.println (s);
}
}
```

O/P:-
a = 42!

insert()

StringBuffer insert (int index, String str)
StringBuffer insert (int index, char ch)
StringBuffer insert (int index, Object obj)

```
class Example
{
  p.s.v.m. (  )
  {
    StringBuffer sb = new StringBuffer ("I Java!");
    sb.insert (2, "like");
    System.out.println(sb);
  }
}
```

O/P:- I like java!

reverse()

StringBuffer reverse();

```
class Example
{
  p.s.v. m(      )
  {
    StringBuffer s=neco StringBuffer("abcdef");
    System.out.println(s);
     s.reverse();
    System.out:println(s);
  }
}
```

o/p:- abcdef
fedcba

delete() and deleteCharAt()

StringBuffer delete(int startIndex, int endIndex);
StringBuffer deleteCharAt(int loc);

```
class Ex
{
  p.s.v.m(    )
  {
  StringBuffer sb=neco StringBuffer("this is a test.");
    sb.delete(4,7);
    S.o.p.("After delete:"+sb);
     sb.deleteCharAt(0);
     S.o.p("After deleteCharAt:"+sb);
  }
}
```

o/p: After delete: this a test.
After deleteCharAt: his a test.

replace()

```
StringBuffer replace (int startIndex, int endIndex,
                      String str);
```

```
class Example
{
  p.s.v.m(  )
  {
   StringBuffer sb = new StringBuffer ("This is a test.");
    sb.replace (5,7, "was");
    S.o.p ("After replace: " +sb);
  }
}
```

o/p:-

After replace: this was a test.