**Strings Continued**

**Substring() method**

The substring method can be used to extract some characters from the String. These substrings can be obtained using indices.

public String substring(int beginIndex)

Returns a new string that is a substring of this string. The substring begins with the character at the beginIndex and extends to the end of this string.

public String substring(int beginIndex, int endIndex)

This returns a new string that is a substring of this string. The substring begins at the specified beginIndex and extends to the character at index (endIndex - 1).

class Substring  
{  
    public static void main(String arg[])  
    {  
        String s1 = "Welcome to a morning class ";  
        String s2 = s1.substring(11);  // LINE A  
        System.out.println(s2);  
        String s3 = s1.substring(11, 17);  // LINE B  
        System.out.println(s3);      
    }  
}

**replace()**

The replace() is used to replace a character or a sequence of characters of an invoking string with a desired character or set of characters .

String replace(char original, char replacement)

This form is used to replace a character with a desired character in the invoking string. Here original specifies the character to be replaced, replacement specifies the character that replaces original.

The other form of replace is:  
String replace(CharSequence original, CharSequence replacement)  
  
This form is used to replace a charactersequence with a desired charactersequence in the invoking string. Here also original specifies the character sequence to be replaced, replacement specifies the character sequence that replaces original.

class ReplaceDemo  
{  
    public static void main(String arg[])  
    {  
        String sentence = "Moon is bright";  
        System.out.println(sentence); // LINE A  
        String sentence2 = sentence.replace('M', 'N');  
        System.out.println(sentence2); // LINE B  
        sentence = sentence + " Moon";  
        System.out.println(sentence); // LINE C  
        String sentence3 = sentence.replace("Moon", "Sun");  
        System.out.println(sentence3); // LINE D      
    }  
}

**trim()**

trim is used to trim/remove whitespace from the beginning and the end of the string that invoked this method. It returns a NEW (new object) string with leading and trailing whitespace removed or the SAME string if it has no leading or trailing whitespace.

**Valueof()**

The valueOf() method is a static method in String class, it returns the string representation of the specified value. The value can be a boolean, char, char [], double, float, int, long or an Object.

**toLowerCase and toUpperCase()**

public String toLowerCase(Locale locale)

str = "turkish";  
        Locale locale = new Locale("tr"); // LINE C  
        System.out.println("String in Turkish locale is: " + str.toUpperCase(locale)); // LINE D  
        // System.out.println("String in default locale (English) :" + str.toUpperCase()); // LINE E  
        str = "TURKISH";  
        System.out.println("String in Turkish locale is: " + str.toLowerCase(locale)); // LINE F

**contains()**

to check if a string contains some pattern

**String Arrays()**

String Array is a container object that holds a fixed number of String values. The length of an String array is established when the array is created. After creation, its length is fixed.

String names[] = new String[3]; // Declaring a String array with size

String names[] = {"Prem", "Siva", "Mahesh", "Srinath"}; // Declaring a String array with elements

String names[] = new String[]{"Prem", "Siva", "Mahesh", "Srinath"}; // Other way of declaring

**Program to:**

1. **LengthOfEachWordInAnArray**String a[]= {"hello","good mroning"};
2. **Split a sentence and count words**String input = “Good morning friends”;
3. **Print only names start with Ro**String names[] = {“Abhishek”,”Roopa”,”Rohit”,”Rishi”,”Rohan”,”Vinod”,”Ronit”,”Naresh”}
4. **Print name which contain “sh”**
5. **Reversing a string**

**String Buffer**

StringBuffer class enables you to represent and manipulate sequence of characters. Unlike String class, StringBuffer class is mutable.

When you modify a string of StringBuffer class, you are not creating a new String object, but rather operating directly on the original string itself.

The instances of StringBuffer hold the sequence of characters in some internal array of characters. The StringBuffer has a capacity which is the number of characters it can hold without requiring reallocation. When the length of characters in the StringBuffer overflow and exceed the capacity, the capacity is automatically increased.

//  To create an empty StringBuffer with a default initial capacity of 16 characters  
StringBuffer sb = new StringBuffer();  
//  To create a StringBuffer from a String  
StringBuffer sb = new StringBuffer(“Happy new year”);  
//  To create an empty StringBuffer with an initial capacity of 100 characters  
StringBuffer sb = new StringBuffer(100);

class StringBufferBasics  
{  
    public static void main(String arg[])  
    {  
        StringBuffer sb = new StringBuffer("Good Morning, Thursday");  
        sb.setCharAt(5, '\*');  //LINE A  
        System.out.println(sb);  
        StringBuffer sb1 = new StringBuffer("C++,Java,C");  
        sb1.replace(4, 8,"HTML");    //LINE B  
        System.out.println(sb1);  
        StringBuffer sb2 = new StringBuffer("Hapy");  
        sb2.insert(2,'p');  
        System.out.println(sb2);  
      
    }  
}

Following are the important points about StringBuffer:

* A StringBuffer is like a String, but can be modified.
* It contains some particular sequence of characters, but the length and content of the sequence can be changed through certain method calls.
* They are safe for use by multiple threads.
* Every StringBuffer has a capacity.

**Different ways of creating SB**

class StringBufferConstructorDemo  
{  
    public static void main(String arg[])  
    {  
        StringBuffer sb1 = new StringBuffer();  
        System.out.println("Empty StringBuffer : " + sb1);  
        CharSequence cs = "ABC"; // LINE A  
        StringBuffer sb2 = new StringBuffer(cs);  
        System.out.println("StringBuffer using CharSequence : " + sb2);  
        String s = "XYZ"; // LINE B  
        StringBuffer sb3 = new StringBuffer(s);  
        System.out.println("StringBuffer using String : " + sb3);      
    }  
}

**length and capacity methods()**

The length is the character count of the sequence of characters currently represented by StringBuffer.  
The capacity is the amount of storage available for newly inserted characters, beyond which an allocation will occur.

class StringBufferDemo1  
{  
    public static void main(String[] args) {  
      
        StringBuffer buff = new StringBuffer("");  
      
        // returns the current capacity of the String buffer i.e. 16 + 0  
        System.out.println("empty capacity = " + buff.capacity());  
      
        // printing the length of empty StringBuffer  
        System.out.println("empty length = " + buff.length());  
      
        buff = new StringBuffer("Java"); // LINE A  
      
        // returns the current capacity of the String buffer i.e. 16 + 4  
        System.out.println("Java capacity = " + buff.capacity());  
      
        // printing the length of StringBuffer  
        System.out.println("Java length = " + buff.length());  
      
    }  
}

StringBuffer sb = new StringBuffer(6);  
System.out.println("Initial capacity : " + sb.capacity());  
sb.append("Happy Diwali");  
System.out.println("Lenght of sb : " + sb.length());  
System.out.println("StringBuffer capacity : " + sb.capacity());

The initial capacity is 6. But when the capacity is exceeded the capacity will be increased as (initial capacity \* 2 + 2), so the capacity will become 14.

StringBuffer sb = **new** StringBuffer(11);

System.***out***.println("Intial capacity : " + sb.capacity());

sb.append("Happy Diwali");

System.***out***.println("Length of sb : " + sb.length());

System.***out***.println("StringBuffer capacity : " + sb.capacity());

**setLength()**

The setLength() method of StringBufferclass changes the length of the StringBuffer object to a specified length.  
Syntax of setLength method is:  
public void setLength(int newLength)

* The length of the StringBuffer object is reduced to the specified length.
* The new length can be greater than, equal to or less than the current length.
* If the specified length is greater than the current length, null characters are padded to attain the specified length. If the specified length is less than the current length then the StringBufferobject is truncated to the required length
* class SetLengthDemo  
  {  
      public static void main(String arg[])  
      {  
          StringBuffer sb = new StringBuffer("Java Programming");  
          System.out.println(sb + "\nLength = " + sb.length());  
          sb.setLength(18); // LINE A  
          System.out.println(sb + "\nLength = " + sb.length());  
          sb.setCharAt(16, '.');  
          sb.setCharAt(17, '.');  
          System.out.println(sb + "\nLength = " + sb.length());  
          sb.setLength(4); // LINE B  
          System.out.println(sb + "\nLength = " + sb.length());      
      }  
  }

**charAt() and setCharAt()**

**append()**

We know that concat method or the concatenation operator +, can be used to concatenate two strings of String class.

Concatenation of strings of StringBuffer class is done using the append method of StringBuffer class. This method is overloaded to take all types of arguments. append method exists in the following forms:  
public StringBuffer append(Object obj)  
public StringBuffer append(String str)  
public StringBuffer append(StringBuffer sb)  
public StringBuffer append(CharSequence s)  
public StringBuffer append(CharSequence s, int start, int end)  
public StringBuffer append(char[] str)  
public StringBuffer append(char[] str, int offset, int len)  
public StringBuffer append(boolean b)  
public StringBuffer append(char c)  
public StringBuffer append(int i)  
public StringBuffer append(long lng)  
public StringBuffer append(float f)  
public StringBuffer append(double d)

class StringBufferAppend  
{  
    public static void main(String arg[])  
    {  
        StringBuffer a1 = new StringBuffer("abc");  
        StringBuffer a2 = new StringBuffer("def");  
        System.out.println("Appending another StringBuffer- "+ a1.append(a2));  
        String a3 = "ghi";  
        System.out.println("Appending a String- "+ a1.append(a3));  
        char c1[] = {'j','k','l'};  
        System.out.println("Appending a  character array- "+ a1.append(c1));  
        System.out.println("Appending part of a character array- "+ a1.append(c1,0,2)); //LINE A  
        System.out.println("Appending a boolean- "+ a1.append(true));  
        System.out.println("Appending a character- "+ a1.append('Z'));  
        System.out.println("Appending an integer- "+ a1.append(1));  
        System.out.println("Appending a double- "+ a1.append(2.35));  
        StringBufferAppend sba = new StringBufferAppend();  //LINE B  
        System.out.println("Appending an object- "+ a1.append(sba));  //LINE C      
    }  
}

**insert()**

This method can be used to insert characters at a given position in a StringBuffer. It is overloaded to accept all the vlaues of the primitive types, String Object, character array and for Object.

class StringBufferDemo5  
{  
    public static void main(String args[])  
    {  
      
        StringBuffer sb = new StringBuffer("Happy Diwali "); // LINE A  
        sb.insert(sb.length(), " to everyone!!"); // LINE B  
        System.out.println(sb);  
    }  
}

**reverse()**

We can reverse the characters in the StringBuffer using reverse.

The reverse method reverses the StringBuffer object on which it is called.  
StringBuffer reverse()

class StringReverseExample  
{  
    public static void main(String arg[])  
    {  
        StringBuffer firstWord = new StringBuffer("Hello World");  
        System.out.println(firstWord);  
        firstWord.reverse();  
        System.out.println(firstWord);      
    }  
}

**delete and deleteCharAt()**

The methods delete and deleteCharAt are used for deleting a character or a sequence of characters with in a StringBuffer.

class StringBufferDeleteExample  
{  
    public static void main(String arg[])  
    {  
        StringBuffer firstWord = new StringBuffer("Moring");  
        System.out.println(firstWord);  
        firstWord.delete(1, 4);  
        System.out.println(firstWord);  
        firstWord.deleteCharAt(0);  
        System.out.println(firstWord);      
    }  
}

**replace()**

This method replaces the characters in a substring of this sequence with characters in the specified String.

class StringBufferDemo6  
{  
    public static void main(String[] args) {  
      
        StringBuffer buff = new StringBuffer("Java Util Package"); // LINE A  
        System.out.println("Stringbuffer = " + buff);  
      
        // replace substring from index 5 to index 9  
        buff.replace(5, 9, "Lang"); // LINE B  
      
        // prints the stringbuffer after replacing  
        System.out.println("After replacing: " + buff);  
    }  
  
}

**substring()**

class PrintSubstring  
{  
    public static void main(String arg[])  
    {  
        StringBuffer name = new StringBuffer("Wednesday");  
        String firstPart = name.substring(0, 5);  
        String secondPart = name.substring(5);  
        System.out.println(firstPart);  
        System.out.println(secondPart);  
        // String thirdPart = name.substring(0, 10); // Won't work // LINE A      
    }  
}

**StringBuilder**

StringBuilder is a mutable sequence of characters and this class is compatible with StringBuffer, but with no guarantee of synchronization.

Instances of StringBuilder are not safe for use by multiple threads. If such synchronization is required then it is recommended that StringBuffer be used.

The methods and usage of them in StringBuilder class is same as in the String Buffer class

**Conclusion**

Strings are a sequence of characters and are widely used in Java programming. In the Java programming language, strings are objects. The String class has over 60 methods and 13 constructors.