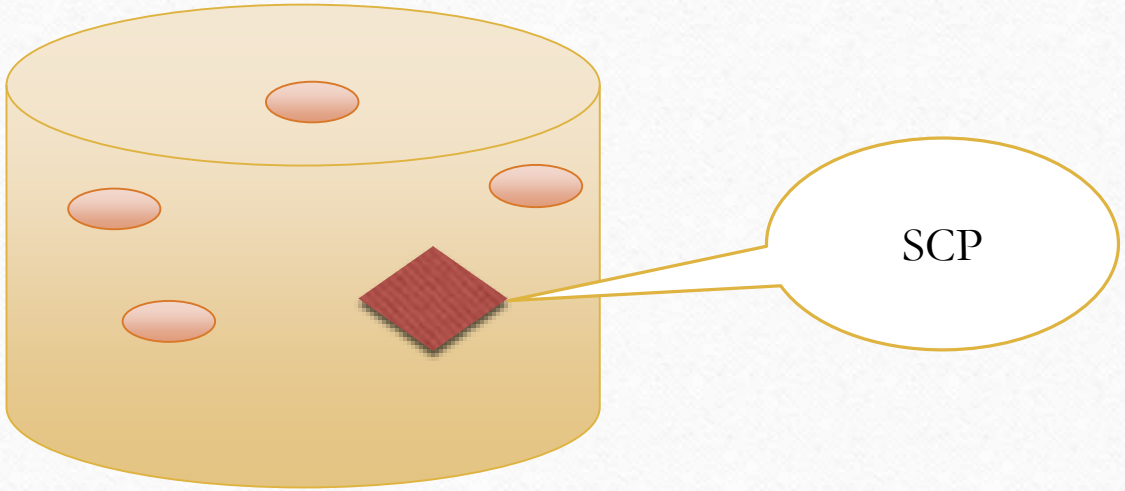
What is String?

- String is non-primitive data type and it is also class which is under java.lang.package.
- String is a collection of characters.
- String is immutable
- Without new keyword we can create object .
- It introduced in JDK1.1.

What is String literals?

- String s="Java";
- SCP is part of Heap memory.

"Java" string literal which is stored in String Constant Pool (SCP)



Why String is immutable?

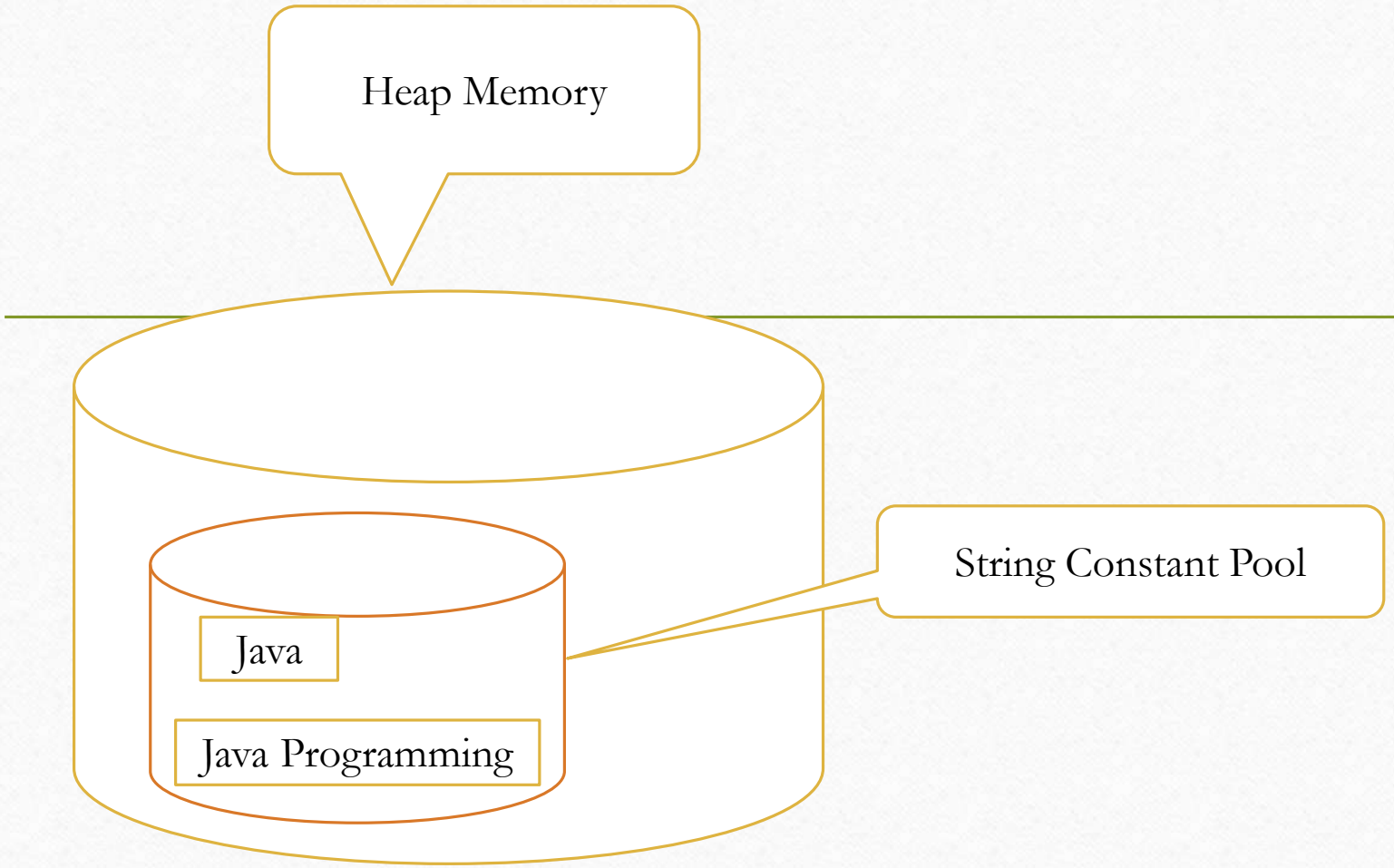
- Strings are constant, values can't be changed after they are created.
- Because java uses the concept of string literal.
- Suppose, if one reference variable changes the value of the object, it will be affected to all the reference variables. That is why string objects are immutable in java.
- So, String is immutable.

Example 1:

```
public class StringDemo8 {

    public static void main(String[] args) {
        String s="Java";
        s.concat(" Programming");
        System.out.println(s);

    }

}
```

```
Java
```

concat() method appends the string at the end so strings are immutable objects.


Heap Memory
String Constant Pool
Java
Java Programming

Example 2:

```
public class StringDemo8 {

    public static void main(String[] args) {
        String s="Java";
        s=s.concat(" Programming");
        System.out.println(s);

    }

}
```

```
Java Programming
```

So, it assign it to the reference variable it prints Java Programming

Methods

- charAt(), contains(), equals(), equalsIgnoreCase(), toUpperCase, toLowerCase
- length(), compareTo, join(), isEmpty(), length(), replaceAll(), replaceFirst()
- trim(), index(), lastIndexOf(), toString(),concat(), replace(), equals()
- replace(), hashCode(), compareToIgnorCase()
- split()
- substring()

Example 3: subString()

```
public class StringDemo {

    public static void main(String[] args) {
        String s=new String();
         s="Java Programming";
         System.out.println(s.substring(5));
         System.out.println(s.substring(0,5));

    }

}
```

```
Programming
Java
```

Overloading

substring(int start/begin index)
substring(int startindex, int endindex)

Returns a new string that is a substring of this string.

Example 4: equals()

```
public class StringDemo1 {

    public static void main(String[] args) {
        String s="Python";
        String s1="python";
        if(s.equals(s1))
        {
            System.out.println("Both are equal");
        }
        else
        {
            System.out.println("Not equal");
        }
    }
```

```
Not equal
```

Return type is boolean

The String equals() method overrides the equals() method of Object class.

Example 5: equalsIgnoreCase()

```java
public class StringDemo1 {

    public static void main(String[] args) {
        String s="PYTHON";
        String s1="python";
        if(s.equalsIgnoreCase(s1))
        {
            System.out.println("Both are equal");
        }
        else
        {
            System.out.println("Not equal");
        }
    }

}
```

```
Both are equal
```

Return type is boolean

Example 6: trim()

```
public class StringDemo2 {

    public static void main(String[] args) {
    String s="  Java Python  ";
    System.out.println(s);
    System.out.println(s.trim());

    }

}
```

```
  Java Python
Java Python
```

Example 7: replace()

```
    public static void main(String[] args) {
        String s1="Java is a very good language";
        String replaceString=s1.replace('v','w');
        System.out.println(replaceString);
        String replaceString1=s1.replace("Java", "Python");
        System.out.println(replaceString1);
    }

}
```

```
Jawa is a wery good language
Python is a very good language
```

Return type is char, charSequence

Replace methods introduced in JDK1.5

Example 8: split()

```java
public static void main(String[] args) {
    String s1="Redington Smart Learning Center, I learn Java here. It is also learn in core java.";
    String[] sentence=s1.split(",");
    String[] words=s1.split(" ");
    System.out.println("Sentences");
    System.out.println("~~~~~~~~~~");
    if(sentence!=null)
    {
        for(String s:sentence)
        {
            System.out.println(s);
        }
        System.out.println("No of sentences "+sentence.length);
    }
    System.out.println();
    System.out.println("Words");
    System.out.println("~~~~~");
    if(words!=null)
    {
        for(String ss:words)
        {
        System.out.println(ss);
        }
        System.out.println("No of words "+words.length);
    }
```

```
Sentences
~~~~~~~~~~
Redington Smart Learning Center
 I learn Java here. It is also learn in core java.
No of sentences 2

Words
~~~~~
Redington
Smart
Learning
Center,
I
learn
Java
here.
It
is
also
learn
in
core
java.
No of words 15
```

It returns array of strings computed by splitting this string around matches of the given regular expression. It introduced in JDK 1.4.

Example 9: split()

```
for(String w:s1.split(" ", 3))
{
    System.out.println(w);
}
```

Overloading

```
Redington
Smart
Learning Center, I learn Java here. It is also learn in core java.
```

It returns array of strings computed by splitting this string around matches of the given regular expression. It introduced in JDK 1.4.

Example 10: concat()

```
public class StringDemo8 {

    public static void main(String[] args) {
        String s="Java";
        s.concat(" Programming");
        System.out.println(s);

    }

}
```

```
Java
```

- It returns a string that represents the concatenation of this object's characters followed by the string argument's characters.

Example 11: contains()

```java
import java.util.Scanner;
public class StringContains {
    public static void main(String[] args) {
        Scanner scan=new Scanner(System.in);
        String s = new String();
        System.out.println("Enter a sentences");
        s=scan.nextLine();
        System.out.println("Find a sentences");
        String find=scan.nextLine();
        System.out.println(s.contains(find));
    }

}
```

```
Enter a sentences
Java is good
Find a sentences
good
true
```

Returns a boolean

Example 12: toString()

```
    private int regno;
    private String name;
    public StringCompareTo() {
        super();
    }
    public StringCompareTo(int regno, String name) {
        super();
        this.regno = regno;
        this.name = name;
    }
    public static void main(String[] args) {
        StringCompareTo sc=new StringCompareTo(123,"Dhivya");
        StringCompareTo sc1=new StringCompareTo(122,"Dhivya");
        System.out.println(sc+" "+sc1);
    }
}
```

```
exercise29aug19.StringCompareTo@1db9742 exercise29aug19.StringCompareTo@106d69c
```

Returns a hash code

Example 12:

```
public class StringCompareTo {
    private int regno;
    private String name;
    public StringCompareTo() {
        super();
    }
    public StringCompareTo(int regno, String name) {
        super();
        this.regno = regno;
        this.name = name;
    }
    public String toString()
    {
        return "Name "+name+" Regno "+regno+"\n";
    }
    public static void main(String[] args) {
        StringCompareTo sc=new StringCompareTo(123,"Dhivya");
        StringCompareTo sc1=new StringCompareTo(122,"Dhivya");
        System.out.println(sc+" "+sc1);
    }
```

```
Name Dhivya Regno 123
 Name Dhivya Regno 122
```

Using toString method to display a value

Example 13: toUpperCase(), toLowerCase(), chatAt()

```java
public class CharAtUppLow {

    public static void main(String[] args) {
        String s=new String("JAVA");
        String c="python";
        System.out.println(s.charAt(2));
        System.out.println("To lower case "+s.toLowerCase());
        System.out.println("To upper case "+c.toUpperCase());
    }

}
```

```
V
To lower case java
To upper case PYTHON
```

Example 14: length(), join(), isEmpty()

```
public static void main(String[] args) {
    StringJoinLengthIsEmpty n=new StringJoinLengthIsEmpty();
    String s = new String();
    System.out.println("Given String "+s.isEmpty());
    s="Java Programming";
    s=s.join("-", "welcome","to","java");
    System.out.println("Join a words : "+s);
    System.out.println("The length is "+s.length());
    System.out.println("Given String "+s.isEmpty());
}
```

```
Given String true
Join a words : welcome-to-java
The length is 15
Given String false
```

Example 15: replaceAll(), replaceFirst()

```java
public class StringReplaceAll {

    public static void main(String[] args) {
        String s="Java Programming is super ";
        System.out.println(s.replaceAll("Java", "Python"));
        System.out.println(s.replaceFirst("super", "new"));

    }

}
```

```
Python Programming is super
Jtvt Progrtmming is super
Java Programming is new
```

Example 16: index()

```
public static void main(String[] args) {
    String s = new String("Java is a Object oriented programming");
    System.out.println(s.indexOf("is"));
    System.out.println(s.indexOf('j', 4));
    System.out.println(s.indexOf('j', 16));
    System.out.println(s.indexOf("Object"));
    System.out.println(s.indexOf("Object", 8));
    System.out.println(s.indexOf("Object", 18));
}
```

```
5
12
-1
10
10
-1
```

Overloading

Example 17: lastIndexOf()

```
public class StringLastIndexOf {

    public static void main(String[] args) {
String s ="Java Programming is awesome, wonderful";
System.out.println(s.lastIndexOf("o"));
System.out.println(s.lastIndexOf("is"));
System.out.println(s.lastIndexOf("o", 28));
System.out.println(s.lastIndexOf("is", 20));
    }

}
```

```
<term
30
17
24
17
```

Overloading

Example 18: compareTo()

```
public class StringCompareTo {
    public static void main(String[] args) {
    String[] s={"guru","divya","anju","ice","Divya"};
    int i;
    for(i=0;i<s.length;i++)
    {
        for(int j=i+1;j<s.length;j++)
        {
            if(s[i].compareTo(s[j])>0)
            {
                String temp=s[i];
                s[i]=s[j];
                s[j]=temp;
            }
        }
        System.out.println(s[i]);
    }
    }
}
```

```
anju
divya
Divya
guru
ice
```

Example 19: compareToIgnoreCase()

```
package exercise30aug19;

public class StringCompareTo {
    public static void main(String[] args) {
    String[] s={"guru","divya","anju","ice","Divya"};
    int i;
    for(i=0;i<s.length;i++)
    {
        for(int j=i+1;j<s.length;j++)
        {
            if(s[i].compareToIgnoreCase(s[j])>0)
            {
                String temp=s[i];
                s[i]=s[j];
                s[j]=temp;
            }
        }
        System.out.println(s[i]);
    }
    }
}
```

```
anju
divya
Divya
guru
ice
```

Methods Name	Introduced Version	Description	Return Type	Parameter
charAt()	1.0	The char value at the specified index. An index ranges from 0 to length() – 1(length in interface CharSequence)	The char value at the specified index of this string. The first char value is at index 0.	index - the index of the char value.
IsEmpty()	1.6	If it is true, and only if, length() is 0.	True if length() is 0, otherwise false	No
concat()	1.0	Concatenates the specified string to the end of this string.If the length of the argument string is 0, then this String object is returned	A string that represents the concatenation of this object's characters followed by the string argument's characters.	The String that is concatenated to the end of this String.
compareTo()	1.0	The comparison is based on the Unicode value of each character in the strings.	The value 0 if the argument string is equal to this string	The String to be compared.

Methods Name	Introduced Version	Description	Return Type	Parameter
contains()	1.5	True if and only if this string contains the specified sequence of char values.	True if this string contains s, false otherwise	The sequence to search for.
trim()	1.0	a copy of the string, with leading and trailing whitespace omitted.	A copy of this string with leading and trailing white space removed, or this string if it has no leading or trailing white space.	No
equals IgnoreCase()	1.0	Compares this String to another String, ignoring case considerations	true if the argument is not null and it represents an equivalent String ignoring case; false otherwise.	The String to compare this String against
length()	1.0	The length of this string. The length is equal to the number of Unicode code units in the string. (charAt in interface CharSequence)	the length of the sequence of characters represented by this object.	No

Methods Name	Introduced Version	Description	Return Type	Parameter	Exception
replaceAll()	1.4	Replaces each substring of this string that matches the given regular expression with the given replacement.	The resulting String.	the sequence to search for	PatternSyntaxException
replaceFirst()	1.4	Replaces the first substring of this string that matches the given regular expression with the given replacement.	The resulting String.	the regular expression to which this string is to be matchedreplacement - the string to be substituted for the first match	IndexOutOfBoundsException -
compareToIgnoreCase()	1.2	Compares two strings lexicographically, ignoring case differences.	The specified String is greater than, equal to, or less than this String, ignoring case considerations.	The String to be compared.	No

Methods Name	Introduced Version	Description	Return Type	Parameter	Exception
toUpperCase()	1.0	Converts all of the characters in this String to upper case using the rules of the default locale.	the String, converted to uppercase.	No	No
toLoweCase()	1.0	Converts all of the characters in this String to lower case using the rules of the default locale.	the String, converted to lowercase.	No	No
join()	1.8	A string joined with given delimiter. In string join method, delimiter is copied for each elements.	Joined string with delimiter.	**delimiter** : char value to be added with each element **elements** : char value to be attached with delimiter	**NullPointerException**

Methods Name	Introduced Version	Description(Overrides a class Object)	Return Type	Parameter
equals()	1.0	Compares this string to the specified object. The result is true if and only if the argument is not null and is a String object that represents the same sequence of characters as this object.	True if the given object represents a String equivalent to this string, false otherwise	The object to compare this String against
hashCode()	1.0	A hash code for this string.	A hash code value for this object.	No
toString()	1.0	The toString method for class Object returns a string consisting of the name of the class of which the object is an instance. This object (which is already a string!) is itself returned.	A string representation of the object.	No

Methods Name	Introduced Version	Description(Overloading)	Return Type	Parameter	Exception
substring()	1.0	Returns a new string that is a substring of this string. The substring begins with the character at the specified index and extends to the end of this string.	the specified substring.	beginIndex - the beginning index, inclusive.	IndexOutOfBoundsException -
substring()	1.0	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. Thus the length of the substring is endIndex-beginIndex.	the specified substring.	beginIndex - the beginning index, inclusive. endIndex - the ending index, exclusive.	IndexOutOfBoundsException -

Methods Name	Introduced Version	Description(Overloading)	Return Type	Parameter	Exception
replace()	1.0	Returns a new string resulting from replacing all occurrences of oldChar in this string with newChar.	A string derived from this string by replacing every occurrence of oldChar with newChar.	oldChar - the old character.newChar - the new character.	No
replace()	1.5	Replaces each substring of this string that matches the literal target sequence with the specified literal replacement sequence.	The resulting String.	target - The sequence of char values to be replacedreplacement - The replacement sequence of char values	NullPointerException

Methods Name	Introduced Version	Description(Overloading)	Return Type	Parameter	Exception
split()	1.4	Splits this string around matches of the given regular expression	The array of strings computed by splitting this string around matches of the given regular expression	The delimiting regular expression.	PatternSyntaxException
split()	1.4	Splits this string around matches of the given regular expression.	the array of strings computed by splitting this string around matches of the given regular expression.	regex - the delimiting regular expressionlimit - the result threshold, as described above	PatternSyntaxException -

Methods Name	Introduced Version	Description(Overloading)	Return Type	Parameter	Exception
index()	1.0	The index within this string of the first occurrence of the specified character.	the index of the first occurrence of the character in the character sequence represented by this object	a character (Unicode code point).	No
index()	1.0	"	"	fromIndex - the index to start the search from.	No
index()	1.0	"	The index of the first occurrence of the specified substring	the substring to search for.	No
index()	1.0	"	"	fromIndex - the index from which to start the search.	No

Methods Name	Introduced Version	Description(Overloading)	Return Type	Parameter	Exception
lastIndex()	1.0	The index within this string of the last occurrence of the specified character.	the index of the last occurrence of the character in the character sequence represented by this object	a character (Unicode code point).	No
lastIndex()	1.0	"	"	fromIndex - the index to start the search from.	No
lastIndex()	1.0	"	The index of the first occurrence of the specified substring	the substring to search for.	No
lastIndex()	1.0	"	"	fromIndex - the index from which to start the search.	No

What is String Buffer?

- StringBuffer is mutable String.
- Java StringBuffer class is (synchronized)thread-safe i.e. multiple threads cannot access it simultaneously.
- So it is safe and will result in an order.

Why String Buffer is mutable?

```
public class StringDemo4 {

    public static void main(String[] args) {
    StringBuffer sb=new StringBuffer("Ajith ");
    sb.append("Kumar");
    System.out.println(sb);
    }

}
```

```
Ajith Kumar
```

Overloading

append() it return many data types

Methods

- append(), replace(), setCharAt()

- insert()
- delete()
- reverse()
- length()
- charAt()
- deleteChatAt()
- setLength()

Methods Name	Introduced Version	Description	Return Type	Parameter	Exception
delete()	1.2	Removes the characters in a substring of this sequence.	This object.	start - The beginning index, inclusive.end - The ending index, exclusive.	StringIndexOutOfBoundsException
deleteCharAt()	1.2	Removes the char at the specified position in this sequence. This sequence is shortened by one char.	“	Index of char to remove	“
replace()	1.2	Replaces the characters in a substring of this sequence with characters in the specified String.	“	start - The beginning index, inclusive.end - The ending index, exclusive.str - String that will replace previous contents.	“

Methods Name	Introduced Version	Description	Return Type	Parameter	Exception
reverse()	1.0.2	Causes this character sequence to be replaced by the reverse of the sequence	A reference to this object.	No	No
setLength()	1.0	Sets the length of the character sequence. The sequence is changed to a new character sequence whose length is specified by the argument	No	The new length	IndexOutOfBoundsException
charAt()	1.0	Returns the char value in this sequence at the specified index.	the char value at the specified index	index - the index of the desired char value.	IndexOutOfBoundsException
setCharAt()	1.0	The character at the specified index is set.	No	index - the index of the character to modify.ch - the new character.	IndexOutOfBoundsException

Methods Name	Introduced Version	Description(Overloading)	Return Type	Parameter	Exception
append()	1.0	Appends the specified string to this character sequence.	A reference to this object.	A string.	No
insert()	1.0	Inserts the string into this character sequence	A reference to this object.	offset - the offset.str - a string.	StringIndexOutOfBoundsException

Append() and insert() methods has many types in StringBuffer.

Example 1: delete(), deleteCharAt()

```
public class StringBufferDelete {

    public static void main(String[] args) {
StringBuffer sb=new StringBuffer("Java Programming is awesome");
System.out.println(sb.delete(5,10));
StringBuffer sb1=new StringBuffer("Python is super");
System.out.println(sb1.deleteCharAt(10));
    }

}
```

```
Java amming is awesome
Python is uper
```

Example 2: reverse(), replace()

```
public class StringBufferReplace {

    public static void main(String[] args) {
        StringBuffer sb=new StringBuffer("Java Programming is awesome");
        System.out.println(sb.replace(5, 15, "Python"));
        System.out.println(sb.reverse());
    }

}
```

```
Java Pythong is awesome
emosewa si gnohtyP avaJ
```

Example 3: setLength()

```
public class StringBufferTrim {

    public static void main(String[] args) {
        StringBuffer sb=new StringBuffer("Java Programming is awesome");
        sb.setLength(8);
        System.out.println(sb);
    }

}
```

```
Java Pro
```

Example 4:charAt(), setCharAt()

```
public class StringBufferChar {

    public static void main(String[] args) {
        StringBuffer sb=new StringBuffer("Java Programming is awesome");
        System.out.println(sb.charAt(5));
        sb.setCharAt(10, 'p');
        System.out.println(sb);
    }
}
```

```
P
Java Progrpmming is awesome
```

Example 5: insert()

```
public class StringBufferInsert {

    public static void main(String[] args) {
        StringBuffer sb=new StringBuffer("Java Programming is awesome");
        sb.insert(17, "nice");
        System.out.println(sb);
    }
```

Overloading

```
Java Programming niceis awesome
```

What is String Builder?

- StringBuilder is mutable String.
- The Java StringBuilder class is same as StringBuffer class except that it is non-synchronized (not thread-safe).
- It is available since JDK 1.5.

Methods

- append(), replace(), subsequence(), substring(), charAt() , trimToSize()
- insert() delete(), capacity(), ensureCapacity(), reverse(), length()
- StringBuffer methods is similar to StringBuilder

Methods Name	Introduced Version	Description(Overloading)	Return Type	Parameter	Exception
subsequence ()	1.5	the new character sequence from given start index to exclusive end index value of this sequence.	The subSequence(int start, int end) method returns the specified subsequence.	The start and end index of a subsequence.	**IndexOutOfBoundsException**
trimToSize( )	1.5	To reduce the storage used for the character sequence.	No	No	No
reverse()	1.5	Causes this character sequence to be replaced by the reverse of the sequence	A reference to this object.	No	No

Example 1:reverse()

```
public class StringDemo6 {

    public static void main(String[] args) {
        StringBuilder sb=new StringBuilder("Hello");
        sb.reverse();
        System.out.println(sb);
    }

}
```

```
olleH
```

Overrides in a String Buffer

Example 2: subsequence()

```
public class StringBuilderSubSeq {

    public static void main(String[] args) {
        StringBuilder sb=new StringBuilder("Java is high level language");
        CharSequence cs=sb.subSequence(2, 10);
        System.out.println(cs);
```

```
va is hi
```

Example 3: trimToSize()

```
public class StringBuilderTrim {
    public static void main(String[] args) {
        StringBuilder sb = new StringBuilder("programming");
        System.out.println("String = "+sb);
        int length = sb.length();
        int capacity = sb.capacity();
        System.out.println("Length = "+length);
        System.out.println("Capacity = "+capacity);
        sb.trimToSize();
        length = sb.length();
        capacity = sb.capacity();
        System.out.println("Length after trimtosize = "+length);
        System.out.println("Capacity after trimtosize= "+capacity);
    }
}
```

```
String = programming
Length = 11
Capacity = 27
Length after trimtosize = 11
Capacity after trimtosize= 11
```

Difference

String	String Buffer
It is immutable	It is mutable.
String is slow and consumes more memory	String is fast and consumes less memory
When you concat too many strings because every time it creates new instance.	When you cancat strings.
1.0	1.0
Java.lang.package	Java.lang.pakage

Difference

StringBuffer	String Builder
It is immutable	It is immutable.
It is synchronized i.e. thread safe.	It is non-synchronized i.e. not thread safe.
It means two threads can't call the methods of StringBuffer simultaneously.	It means two threads can call the methods of StringBuilder simultaneously.
1.0	1.5
It is less efficient.	It is more efficient.