(20) Mutable String and String Programming

↳ Nitin Sir

Today's topic

(a) String Buffer
(b) String Builder
(c) Discussion on programs
(d) github integration with Eclipse (next class).

(1) String Buffer :-

① if the Content will change frequently then it is not recommended to go for String object because for new object will be created.

② To handle this type of requirement we have StringBuffer / String Builder Concept.

(2) Append Method :- append()

→ append Method is overloaded Method, Method name is same but change in the argument.

① Public StringBuffer append(String s)
② " " " (int i)
③ Public StringBuffer append(long l)
④ Public StringBuffer append(boolean b)
⑤ Public StringBuffer append(double d)
⑥ Public String Buffer append(float f)
⑦ Public String Buffer append(int index, object o)

Example①

```
StringBuffer sb = new StringBuffer();

// a. Public StringBuffer append(String)
sb.append("Value of Pie is");
// b. Public StringBuffer append(double d)
sb.append(3.1414);
// c. Public StringBuffer append(boolean true)
sb.append(true);
S.O.P(sb); → output:
```

Value of Pie is: 3.1414true

③ Insert Method :- it is overloade Method. Method name is same but change in the argument.

① Public StringBuffer insert(int index, String s).
② Public StringBuffer insert(int index, int i)
③ Public StringBuffer insert(int index, long l)
④ Public StringBuffer insert(int index, double d)
⑤ Public StringBuffer insert(int index, boolean b)
⑥ Public StringBuffer insert(int index, float f)
⑦ Public StringBuffer insert(int index, Object o)

Example① Public StringBuffer insert(int index, String s);

Char: a b c d e f g h
index: 0 1 2 3 4 5 6 7

```
StringBuffer sb = new StringBuffer("abcdefgh");
sb.insert(4, "xyz");
S.O.P(sb) → o/p: abcdxyzefgh
                 0 1 2 3 4 5 6 7 8 9 10
```

↳ xyz is inserted at 4th index.

② Public StringBuffer insert(int index, int i)
↳ inserting integer

a	b	c	d	e	f
0	1	2	3	4	5

```
StringBuffer sb = new StringBuffer("abcdef");
sb.insert(4, 4);
S.O.P(sb); → o/p: abcd4ef
                  0 1 2 3 4 5 6
```

→ 4 is inserted at index 4.

④ ~~delete~~

④ Method for deleting data :-

(i) Public StringBuffer delete(int begin, int End)

↳ it deletes the character from specified index to End - 1

Eg① :-

```
s a c h i n r a m e s h t e n d u l k a r
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
```

deleting ramesh

```
StringBuffer sb = new StringBuffer("SachinrameshtendulkarSachin");
sb.delet(6,12);
System.out.Println(sb);
```

↳ Sachintendulkar

(ii) Public StringBuffer deleteCharAt(int index)

↳ it deletes the character at specified index.

```
StringBuffer sb = new StringBuffer("SachinTtendulkar");
                                    0 1 2 3 4 5 6
// deleting T in string
sb.deleteCharAt(6);
S.o.P(sb) → Sachintendulkar
```

(ii) Public StringBuffer deleteCharAt(int index)

Eg :-

```
s a c h i n T e n d u l k a r
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
```

→ it deletes the character at specified index.

```
StringBuffer sb = new StringBuffer("SachinTendul");
sb.deleteCharAt(6);
S.o.P(sb); o/p: Sachintendulkar
```

→ char at 6 index got deleted.

⑤ Public String Buffer reverse() :-

↳ Reversing String

```
StringBuffer sb = new StringBuffer("Ashish");
S.o.P(sb) // o/p: Ashish
sb.reverse();
S.o.P(sb) // o/p: hsihsA
```

⑥ Public Void setLength(int length) :-

↳ sachintendulkar
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14

```
StringBuffer sb = new StringBuffer ("sachintendulkar");
S.o.p(sb);  // Sachintendulkar
sb.setLength(6);
S.o.p(sb);  // Sachin.
```

→ it is used to consider only the specified no of characters and remove all the remaining characters.

⑦ ~~public void ensureCapacity(int capacity);~~

⑦ Public Void trimToSize();

```
StringBuffer sb = new StringBuffer(1000);
S.o.p(sb.Capacity()); // 1000
sb.append("sachin");
S.o.p(sb.Capacity()); // 1000
```

→ 994 locations are wasted.

→ 1000 locations are reserved but we are not using them.

→ to Ensure Capacity Equal to length of data

Char:	s	a	c	h	i	n
index:	0	1	2	3	4	

```
StringBuffer sb = new StringBuffer(1000);
S.o.p(sb.Capacity()) // o/p = 1000
```

→ Appending string data.

```
sb.append("Sachin");
S.o.p(sb.Capacity()) // o/p = 1000
```

→ to Ensure Capacity to length of data

```
sb.trimToSize();
S.o.p(sb.Capacity()); // o/p = 6
```

→ Capacity trimed to 6 (Sachin string has 6 characters).

→ JVM clean the Memory in heap area and allocate the Memory and takeup all data into Memory keep it and it shares address.

→ Any thing doing at Memory level with the help of a Method it is always a costly operation for application.

② Public Void EnsureCapacity(int Capacity)

↳ it is used to increase the capacity dynamically based on our requirement.

Eg:-
```
StringBuffer sb = new StringBuffer();
System.out.println(sb.capacity()); //16
sb.EnsureCapacity(1000);
S.o.P(sb.capacity()); // o/p:-1000
```

→ increasing capacity at runtime.

→ In Java to create mutable object we have two classes.

ⓐ StringBuffer (available JDK 1.0V)

ⓑ StringBuilder (1.5V) → It is given based on demands of developers

↳ gamming changing Version

→ to see StringBuffer API use this Cmd

java.lang.StringBuffer

→ to see StringBuilder API's use this Command

java.lang.StringBuilder

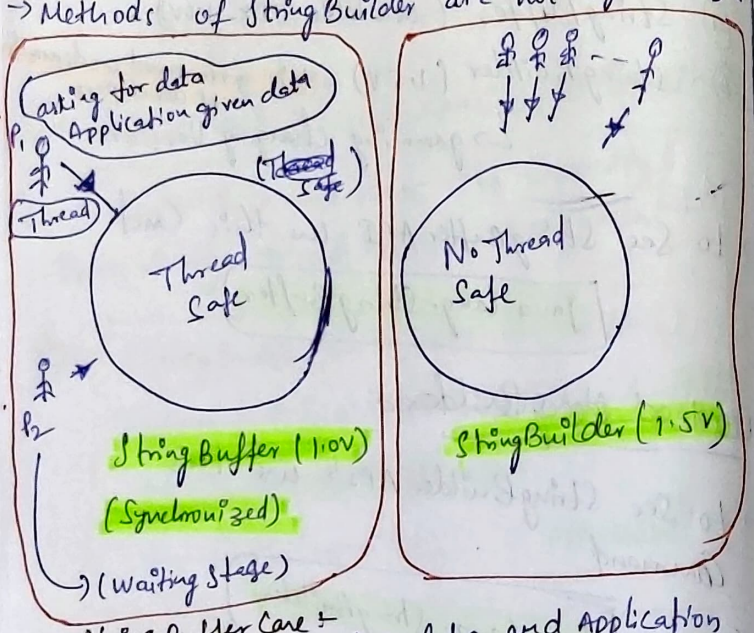
→ In StringBuffer Synchronized Concept is present
↳ it comes under Multi threading.

→ Synchronized Concept not present in StringBuilder.

→ Apart from Synchronized Concept all Methods present in StringBuffer are also there in StringBuilder.

⑫ Difference Between String Buffer and Builder

→ Methods of String Buffer are Synchronized
→ Methods of String Builder are not Synchronized



String Buffer Case :-

(i) Person 1 asked for data and Application given data.

(ii) → ~~Person~~ user we Call as thread

(iii) L→ stringBuffer object is used by user

(iv) Person2 asked for data. Application interacts with stringBuffer.

→ String Buffer says 1 user is using till he tell I am done you have to wait

⑬ String Builder

→ Simultaneously Multiple people Can use this application
→ no need to wait for using application if other user using it

important points :- ① StringBuffer :-

① Every Method present in StringBuffer is Synchronized, So at a time only one thread Can are allowed to operate on String Buffer object.
L→ it would create performance problems, to overcome this problem we should go for String Builder.

② String Builder (1.5V)
L→ stringBuilder is same as String Buffer (1.0V) with few differences.

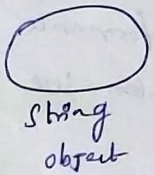
~~② Stage~~
→ No Methods are Synchronized.
→ At a time More than one thread Can operate So it is not thread Safe.
→ Threads are not required to wait So performance is high.
→ introduced in JDK 1.5 version.

→ String can be created in three ways :-

(a) String
(b) StringBuffer
(c) StringBuilder

(3) String Vs StringBuffer Vs StringBuilder

(1) String :- use String when Content is fixed and it wont change frequently.

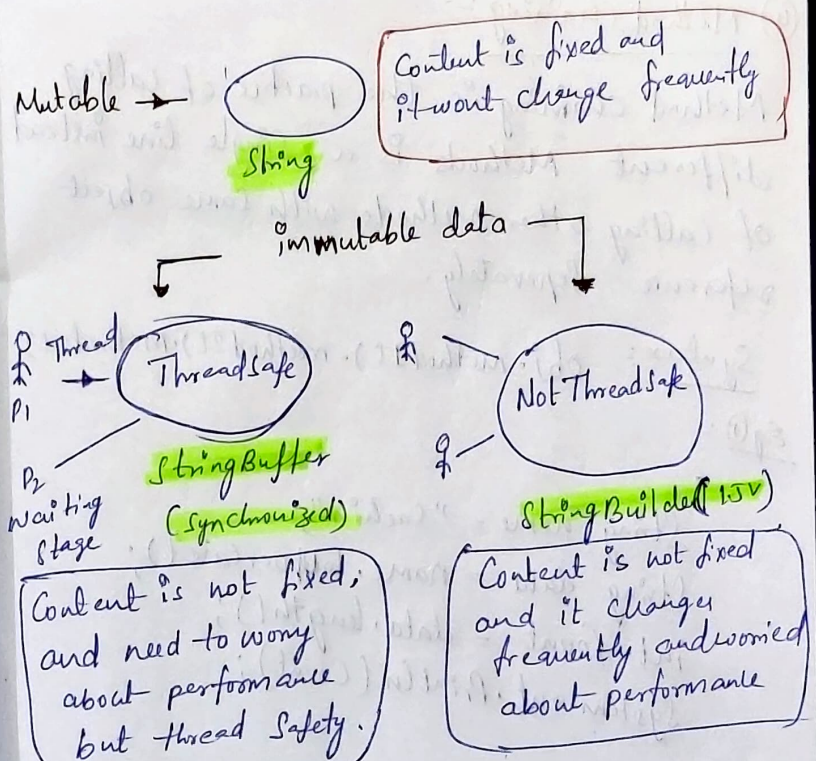


(2) StringBuffer :- we opt it if the Content changes frequently but thread safety is required.

→ need to worry about performance

(3) StringBuilder :- we opt it if the Content changes frequently and but Thread Safety is not required

→ no need to worry about performance.



(4) ~~Method~~ Chaining :-

~~Method chaining~~

④ Method Chaining :-

Method chaining is the practice of calling different Methods is a single line instead of calling other methods with same object reference seperately.

Syntax :- obj.method1().method2().method3()

Eg ① :-

```
String name = "Sachin";
String data = name.toUpperCase();
int Count = data.length();
System.out.println(Count);
```

// Method chaining.

```
System.out.Print(name.toUpperCase().length());
```

Eg ②

```
StringBuffer sb = new StringBuffer("virat");
S.o.P(sb.append("kohli").insert(10, "anushka"));
```

Note:- Method chaing applied on object with same return type.

→ append and insert can be applied on StringBuffer object.

```
sb.append("Kohli").insert(10, "Anushka");
```

Method-1 Method-2

→ Return type of Method-1 and object on Method invoked should be same. this

Eg ②

→ append method is ~~StringBuffer~~ return type is StringBuffer so insert Method can be applied on this object.

Eg ③

```
StringBuffer sb = new StringBuffer("Virat");
sb.append("Kohli")          → ViratKohli
  .insert(10, "Anushka");   → ViratKohliAnushka
  .reverse()                → akhsunAilhoktariV
  .append("IND")            → akhsunAilhoktariVIND
  .insert(sb.length(), "RCB")
                            ↳ akhsunAilhoktariVINDRCB
  .reverse().delete(0,6);   → ViratKohliAnushka
S.o.P(sb);                  BCRDNIViratkohliAnushka
```

→ output : ViratKohliAnushka

Example ③ :-

```
StringBuffer sb1 = new StringBuffer("dhoni");
sb1.length().append("csk");
```

Return type of length is int

append Method is not applicable on integer type data.
it gives Error

output :- Cannot invoke append(String) on the Primitive type int.

Problems

① Copy one string to another String

```
String s1 = "ineuron";
String s2 = "";
for(int i=0; i<s1.length(); i++)
{
    s2 = s2 + s1.charAt(i);
}
System.out.println("first string is" + s1);
System.out.println("Copy of first string" + s2);
```

↳ one by one characters are copying of s1 in s2

↳ output : ineuron

② Lower Case to upperCase & upper Case to Lower Case

① i/p: ineuron o/p: INEURON
② i/p: INEURON o/p: ineuron.

logic :-

ASCII Value
a ⇒ 97
A ⇒ 97 - 32

```
char ch = 'a'; // 97
S.O.P(ch) ⟶ // output : a
ch = (char)(ch - 32);
S.O.P(ch); // output : A
```

(A)

(i) lower Case to upperCase

```
Scanner Scan = new Scanner(System.in);
S.o.p("Enter String");
String S1 = Scan.nextline();
String S2 = "  ";
for(int i=0; i<=S1.length(); i++);
{
    S2 = S2 + (char)(S1.charAt(i)-32);
}
S.o.P(S2);
```

↳ int to char Conversion.

input: ineuron
output: INEURON

(ii) upperCase to lower Case :-

```
String S1 = "INEURON";
String S2 = "  ";
for(int i=0; i<=S1.length(); i++){
    S2 = S2 + (char)(S1.charAt(i)+32);
}
S.o.p(S2); // output: ineuron
```

(3) To Convert Small to Capital and Vice Versa

```
String S1 = "iNeuRON";
String S2 = "  ";
for(int i=0; i<S1.length(); i++){
    if(S1.charAt>='a' && S1.charAt<='z')
    {
        S2 = S2 + (char)(S1.charAt(i)-32);
    }
    Else if(S1.charAt>='A' && S1.charAt<='Z')
    {
        S2 = S2 + (char)(S1.charAt(i)+32);
    }
}
S.o.p(S2); // output: InEUrOn
```

(4) Possible ways to reversing a string

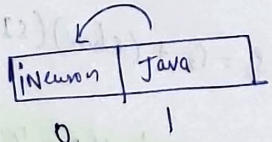
(i) iNeuron --> norueNi

```
String s1 = "iNeuron";
String s2 = "";
for(int i = s1.length()-1; i>=0; i--)
{
    s2 = s2 + s1.charAt(i);
}
s.o.p("original string" + s1);
s.o.p("After Reversing" + s2); // output : norueNi
```

(ii) iNeuron Java --> norueNi avaJ

↳ it is done by using Split Method.

↳ it Split two words and Creates an array.



(5) iNeuron java --> norueNi avaJ

```
String s1 = "iNeuron java";
String s2 = "";
String sarr[] = s1.Split(" ");
for(String elem : sarr)
{
    for(int i = elem.length()-1; i>=0; i--)
    {
        s2 = s2 + elem.charAt(i);
    }
    s2 = s2 + " ";
}
s.o.p(s2); // output : norueNi avaJ
```

③ Ineuron java --> java Ineuron

```
String s1 = "Ineuron java";
String s2= " ";
String sarr[] = s1.split(" ");
for(int i = sarr.length-1; i>=0; i--){
        s2 = s2 + sarr[i] + " ";
}
s.o.p(" ");
s.o.p(s2); // output : java Ineuron.
```