

```
OCJA 1.8 Java SE 8 Programmer - I (1Z0-808 ) by Durga Sir On 28-03-2018  
1 class Test  
2 {  
3     public static void main(String[] args)  
4     {  
5         Test t = new Test();  
6         System.out.println(t);  
7         System.out.println(t.toString());  
8     }  
9 }  
10
```

Questions

X	Question	Asker
	test of obj	Govindu Rayapur

System.out(t) will call t.toString() method. If the Test Class does not have a toString Method it will call the parent class toString method.

Which will print the “class name” followed by “@” followed by hexadecimal string representation of the HashCode .

```
OCJA 1.8 Java SE 8 Programmer - I (1Z0-808 ) by Durga Sir On 28-03-2018  
1 class Test  
2 {  
3     public static void main(String[] args)  
4     {  
5         Test t = new Test();  
6         System.out.println(t.hashCode());  
7         System.out.println(Integer.toHexString(t.hashCode()));  
8         System.out.println(t);  
9     }  
10 }  
11
```

The hashCode is converted to HexaDecimal below

```
D:\durgaclasses>java Test  
366712642  
15db9742  
Test@15db9742
```

```
OCJA 1.8 Java SE 8 Programmer - I (1Z0-808 ) by Durga Sir On 28-03-2018
1 class Test
2 {
3     public static void main(String[] args)
4     {
5         Integer I = new Integer(10);
6         Test t = new Test();
7         System.out.println(t);
8         System.out.println(I);
9     }
10 }
11
```

Integer class has overriden the `toString` method to return the content

```
D:\durgaclasses>java Test
Test@15db9742
10
```

```
OCJA 1.8 Java SE 8 Programmer - I (1Z0-808 ) by Durga Sir On 28-03-2018
1 java.lang
2 only String and StringBuilder, Wrapper classes
3
4 Wrapper classes
5 Collection classes
6 String,StringBuffer,StringBuilder
```

In all String, Wrapper, collection classes , StringBuffer and String Builder the `toString` method is overridden to return the actual value

## Equals method

```
OCJA 1.8 Java SE 8 Programmer - I (1Z0 - 808 ) by Durga Sir On 28-03-2018
```

```
1 java.lang
2 only String and StringBuilder, Wrapper classes
3
4 Wrapper classes
5 Collection classes
6 String, StringBuffer, StringBuilder
7 -----
8 == operator and equals() method:
9 -----
10 == operator meant for reference comparison
11 equals() meant for content comparison
12
13 r1==r2 true
14
```

Handwritten notes:

- A large circle with a minus sign inside is drawn next to the == operator.
- A checkmark is drawn next to the equals() method definition.
- A checkmark is drawn next to the line "r1==r2 true".
- A small circle with a minus sign inside is drawn next to the == operator in the line "r1==r2 true".

Questions

X	Question	Asker
1	== ref comp	Govindu Rayapur
2	== for ref comp and equals for content comp	Poja Chavan
3	== ref comp	Deepakj Yadav
4	equals for content comparsion	Govindu Rayapur
5	content	Amit Kumar

sir in exp handling multiple catch and catch with multiple exps which one is recommended

Send Privately Send to All

Equals method is used to compare the content

== method to compare the ref

```
OCJA 1.8 Java SE 8 Programmer - I (1Z0 - 808 ) by Durga Sir On 28-03-2018
```

```
1 class Student
2 {
3     String name;
4     int rollno;
5     Student(String name,int rollno)
6     {
7         this.name=name;
8         this.rollno=rollno;
9     }
10    public static void main(String[] args)
11    {
12        Student s1= new Student("Durga",101);
13        Student s2= new Student("Ravi",102);
14        Student s3= new Student("Durga",101);
15        System.out.println(s1==s2);
16        System.out.println(s1==s3);
17
18    }
19 }
```

False

False

Both are diff objects used new operator

A screenshot of a Java IDE showing the code for `Student.java`. The code defines a `Student` class with a constructor and a `main` method. The `main` method creates three `Student` objects (`s1`, `s2`, `s3`) and prints their equality using the `equals` method. The code uses the `==` operator for comparison.

```
4 int rollno;
5 Student(String name,int rollno)
6 {
7     this.name=name;
8     this.rollno=rollno;
9 }
10 public static void main(String[] args)
11 {
12     Student s1= new Student("Durga",101);
13     Student s2= new Student("Ravi",102);
14     Student s3= new Student("Durga",101);
15     //System.out.println(s1==s2);
16     //System.out.println(s1==s3);
17     System.out.println(s1.equals(s2));
18     System.out.println(s1.equals(s3));
19 }
20 }
```

False

True

The class does not contain equals method so parent class “Object class” equals method is used

Overriding equals method now Student class equals method will be called

A screenshot of a Java IDE showing the code for `Student.java`. The code is identical to the previous version, but it includes an overridden `equals` method in the `Student` class. This overridden method checks if the `name` and `rollno` fields of two `Student` objects are equal.

```
4 int rollno;
5 Student(String name,int rollno)
6 {
7     this.name=name;
8     this.rollno=rollno;
9 }
10 public boolean equals(Student s)
11 {
12     if (s.name.equals(this.name) && s.rollno==this.rollno)
13     {
14         return true;
15     }
16     else
17         return false;
18 }
19 public static void main(String[] args)
20 {
21     Student s1= new Student("Durga",101);
22     Student s2= new Student("Ravi",102);
```

```
OCJA 1.8 Java SE 8 Programmer - I (1Z0-808) by Durga Sir On 28-03-2018  
1 class Test  
2 {  
3     public static void main(String[] args)  
4     {  
5         String s1= new String("Durga");  
6         String s2= new String("Durga");  
7         System.out.println(s1==s2);  
8         System.out.println(s1.equals(s2));  
9     }  
10 }  
11
```

A screenshot of an IDE showing Java code. The code defines a class named 'Test' with a main method. Inside the main method, two strings 's1' and 's2' are created with the value 'Durga'. The first line of output is 'false' (the reference equality check), and the second line is 'true' (the content comparison using equals). A tooltip '5 seconds' is visible near the bottom left.

```
D:\durgaclasses>javac Test.java
```

```
D:\durgaclasses>java Test
```

```
false  
true
```

String class – overrides equals method to perform content comparison. Object class .equals method is used for Ref Comparison

```
1 class Test  
2 {  
3     public static void main(String[] args)  
4     {  
5         StringBuffer s1= new StringBuffer("Durga");  
6         StringBuffer s2= new StringBuffer("Durga");  
7         System.out.println(s1==s2);  
8         System.out.println(s1.equals(s2));  
9     }  
10 }  
11
```

A screenshot of an IDE showing Java code. The code defines a class named 'Test' with a main method. Inside the main method, two StringBuffer objects 's1' and 's2' are created with the value 'Durga'. The first line of output is 'false' (reference equality), and the second line is 'true' (content comparison using equals). A cursor is visible at the end of the code.

False

False

In StringBuffer or StringBuilder .equals method are not overridden to perform content comparison

All wrapper classes, collections and String equals method is overridden to perform content comparison

```

1 class Test
2 {
3     public static void main(String[] args)
4     {
5         Integer s1= new Integer(10);
6         Integer s2= new Integer(10);
7         System.out.println(s1==s2);
8         System.out.println(s1.equals(s2));
9     }
10 }
11

```

```

1 class Test
2 {
3     public static void main(String[] args)
4     {
5         Integer s1= new Integer(10);
6         Integer s2= new Integer(10);
7         System.out.println(s1==s2);
8         System.out.println(s1.equals(s2));
9     }
10 }
11

```

String  
Co

SB SB  
ref

wra  
conter

X	Question	Asker
ok		Shubham Gupta
false, true		Himanshu G
my trick to learn too much		Deepankaj Yadav
prakash:D		Govindu Rayapur
ok		Shubham Gupta

sir means in object class both == & equals used for reference?

Send Privately Send to All

StringBuilder and StringBuffer -> calls the equals method of the object class and does ref comparison

CE

```

1 class Test
2 {
3     public static void main(String[] args)
4     {
5         String s= new String("Durga");
6         StringBuffer sb= new StringBuffer("Durga");
7         System.out.println(s==sb);
8         System.out.println(s.equals(sb));
9     }
10 }
11

```

When using == operator

Btw two object types there should be some relationship between arg type either child ->Parent

Or parent -> child

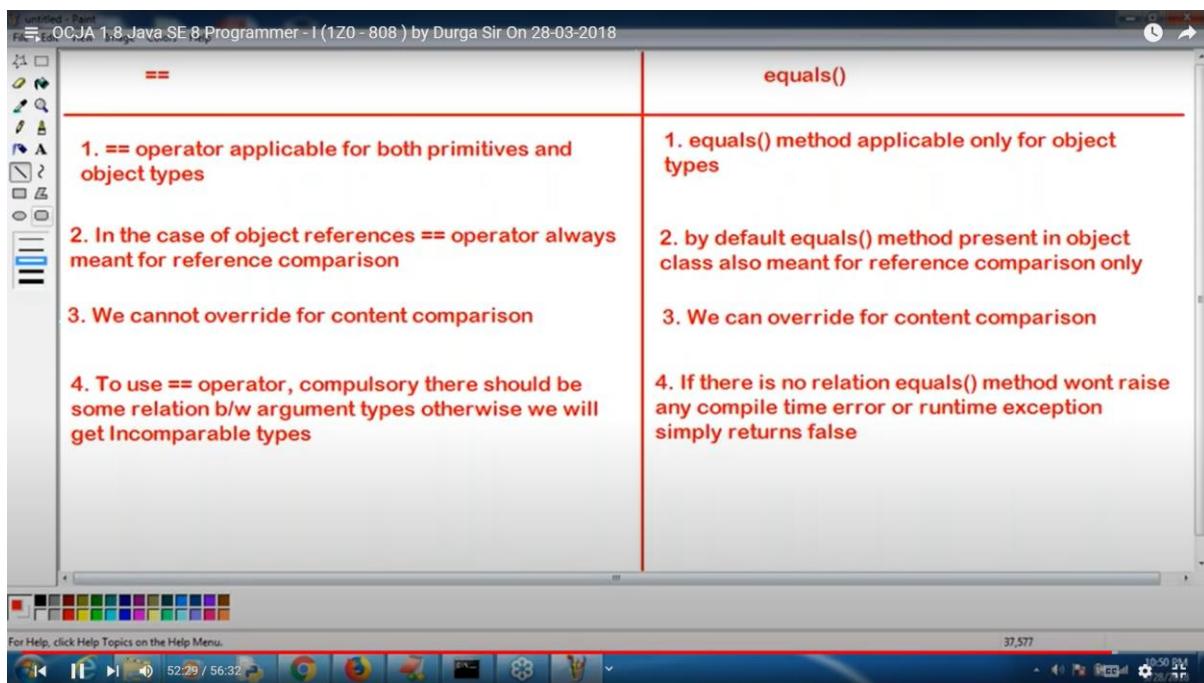
```
1 class Test
2 {
3     public static void main(String[] args)
4     {
5         String s= new String("Durga");
6         StringBuffer sb= new StringBuffer("Durga");
7         System.out.println(s==sb);
8         System.out.println(s.equals(sb));
9     }
10}
11
```

```
C:\Windows\system32\cmd.exe
D:\durgaclasses>javac Test.java
Test.java:7: error: incomparable types: String and StringBuffer
        System.out.println(s==sb);
                           ^
1 error
```

.equals method will not rise CE even if there are no relationship. Either true or false

False -> Output

```
1 class Test
2 {
3     public static void main(String[] args)
4     {
5         String s= new String("Durga");
6         StringBuffer sb= new StringBuffer("Durga");
7         //System.out.println(s==sb);
8         System.out.println(s.equals(sb));
9     }
10}
11
```



Sb.toString-> return String only.

So comparison of String content -> True

```
OCJA 1.8 Java SE 8 Programmer - I (1Z0-808 ) by Durga Sir On 28-03-2018  
1 class Test  
2 {  
3     public static void main(String[] args)  
4     {  
5         String s= new String("Durga");  
6         StringBuffer sb= new StringBuffer("Durga");  
7         //System.out.println(s==sb);  
8         //System.out.println(s.equals(sb));  
9         System.out.println(s.equals(sb.toString()))  
10    }  
11 }
```

```
1 class Test
2 {
3     public static void main(String[] args)
4     {
5         String s= new String("Durga");
6         StringBuffer sb= new StringBuffer("Durga");
7         String s1=sb.toString();
8         System.out.println(s1);
9         System.out.println(s.equals(s1));
10    }
11 }
12
```

```
6 case-1:  
7 -----  
8 String immutable  
9 StringBuffer ==>mutable
```

The screenshot shows a Java code editor window titled "OCJA 1.8 Java SE 8 Programmer - I (1Z0-808) by Durga Sir On 29-03-2018". The code in the editor is:

```
1 class Test  
2 {  
3     public static void main(String[] args)  
4     {  
5         String s = new String("Durga");  
6         s.concat("Software");  
7         System.out.println(s);  
8     }  
9 }  
10
```

One string object is created , we are not allowed to change its content. If u try to change a new object will be created and the original object is unaffected

:

Line6- creates a new object but is not assigned to any variable

The screenshot shows a Java code editor window titled "OCJA 1.8 Java SE 8 Programmer - I (1Z0-808) by Durga Sir On 29-03-2018". The code in the editor is identical to the previous screenshot:

```
1 class Test  
2 {  
3     public static void main(String[] args)  
4     {  
5         String s = new String("Durga");  
6         s.concat("Software");  
7         System.out.println(s);  
8     }  
9 }  
10
```

The screenshot shows a Windows Command Prompt window titled "OCJA 1.8 Java SE 8 Programmer - I (1Z0-808) by Durga Sir On 29-03-2018". The command prompt shows the following output:

```
D:\durgaclasses>javac Test.java  
D:\durgaclasses>java Test  
Durga
```

A new object is created and assigned to ref variable s.

The diagram illustrates the creation of a String object. A red oval labeled "Durga" is connected by a line to a plus sign (+). Another line connects the plus sign to a second red oval labeled "DurgaSoft". This visualizes the concatenation of the strings "Durga" and "Software" into a single string "DurgaSoft".

```
1 class Test
2 {
3     public static void main(String[] args)
4     {
5         String s = new String("Durga");
6         s.concat("Software");
7         System.out.println(s);
8     }
9 }
10
```

String buffer and Builder same –

Since they are mutable- the content of the string can be modified

The diagram illustrates the modification of a StringBuffer object. A red oval labeled "DurgaSoft" has a line pointing to a plus sign (+). Another line connects the plus sign to a second red oval labeled "DurgaSoft". This visualizes the append operation on the StringBuffer object, resulting in the string "DurgaSoft" being modified to "DurgaSoft" followed by "Software".

```
1 class Test
2 {
3     public static void main(String[] args)
4     {
5         StringBuffer sb = new StringBuffer("Durga");
6         sb.append("Software");
7         System.out.println(sb);
8     }
9 }
10
```

False

True

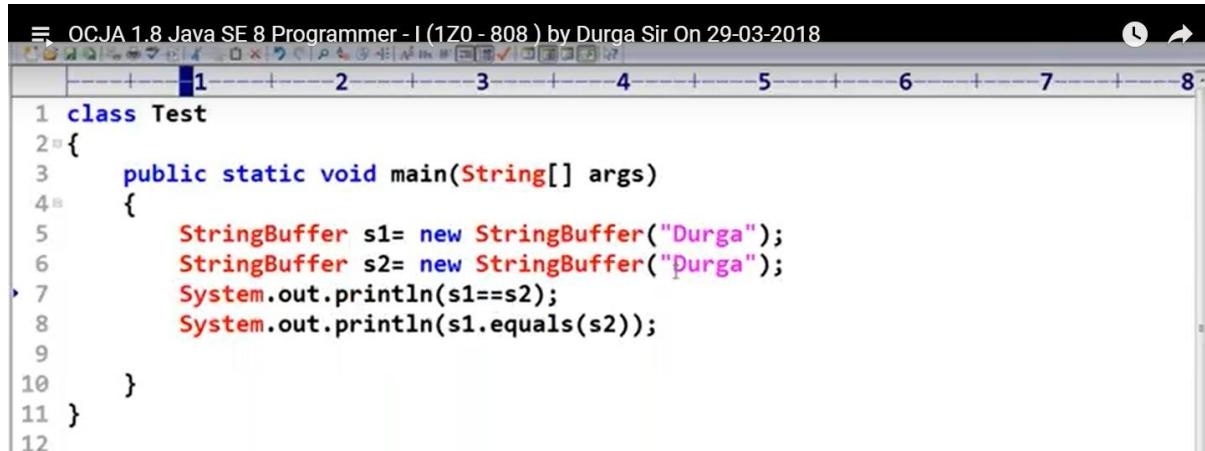
The diagram compares two String objects. A blue box highlights the line "String s1= new String("Durga");". A red box highlights the line "String s2= new String("Durga");". A line connects the "new String("Durga")" part of both lines to a red oval labeled "Durga". Another line connects the "Durga" oval to a plus sign (+). A third line connects the plus sign to a second red oval labeled "Durga". This visualizes that two separate new objects are created, even though they have the same value, and therefore s1.equals(s2) will return false.

```
1 class Test
2 {
3     public static void main(String[] args)
4     {
5         String s1= new String("Durga");
6         String s2= new String("Durga");
7         System.out.println(s1==s2);
8         System.out.println(s1.equals(s2));
9     }
10 }
11 }
```

False

False

StringBuffer and Builder == and equals method both are not overridden hence object class equals method will be called performing ref comparison



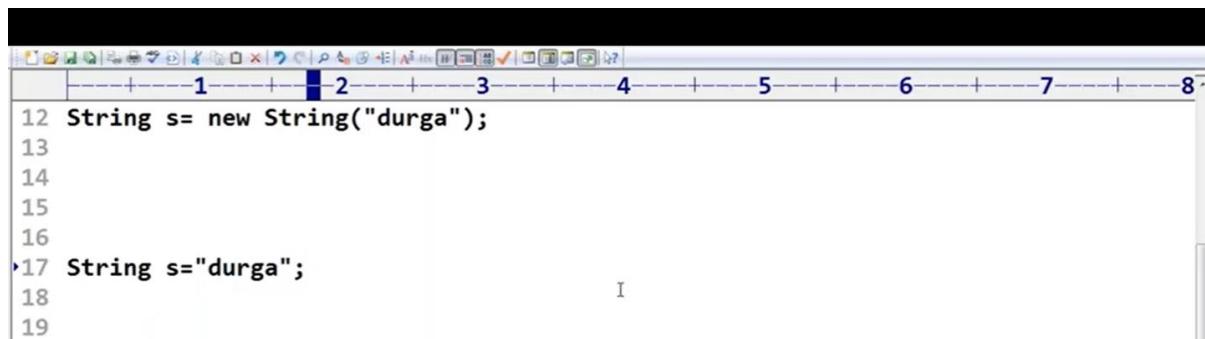
```
OCJA 1.8 Java SE 8 Programmer - I (1Z0-808) by Durga Sir On 29-03-2018
1 class Test
2 {
3     public static void main(String[] args)
4     {
5         StringBuffer s1= new StringBuffer("Durga");
6         StringBuffer s2= new StringBuffer("Durga");
7         System.out.println(s1==s2);
8         System.out.println(s1.equals(s2));
9
10    }
11 }
12
```

In this scenario, two objects will be created

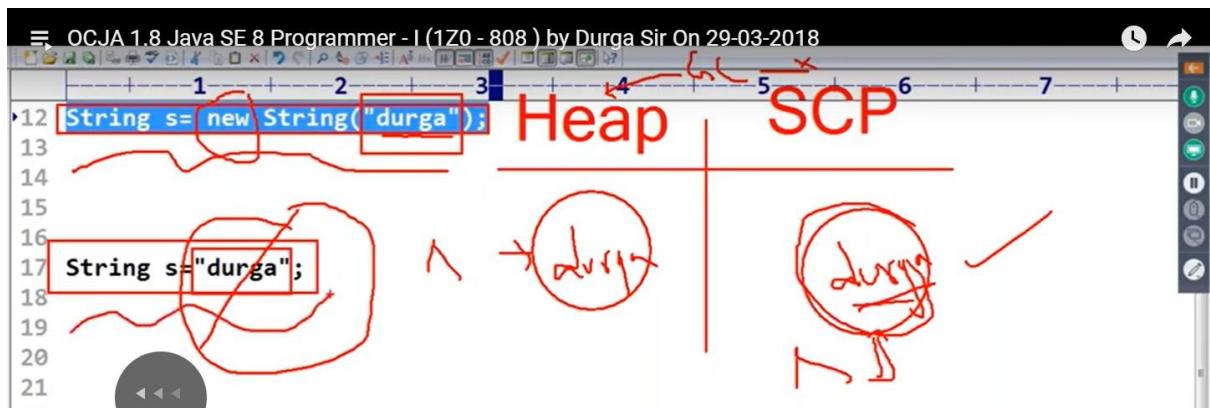
12- Heap area new object will be created. Its constant , and another object gets created in SCP area.

S will be pointed to Heap Area only. The object in SCP has no ref but still will not be eligible for garbage collection and will be used in future

17-> when no new operator is used. JVM checks if there is an existing object with value "durga", if its there then will create a variable with ref to the existing object. Else a new object will be created



```
12 String s= new String("durga");
13
14
15
16
17 String s="durga";
18
19
```



Restarting JVM will be refreshed and SCP area will be refreshed. If you think this would cause memory problem

Any run time operation on String at run time like method call a new object will be created in Heap Area only and not in SCP area. SCP contains only String Constants

```

24
25 String s = new String("Durga");
26 s1=s.concat("Software");
27 s2=s.concat("Software");
28 s3=s.toLowerCase();
29 s4=s.toUpperCase();
30

```

Line 37-> Durga will be created in SCP

Line 38-> Since Durga is already there s2 will point to Durga ref, hence both will have the same ref

True

```

34 String s1= new String("Durga");
35 String s2= new String("Durga");
36 System.out.println(s1==s2); I
37 String s1= "Durga";
38 String s2= "Durga";
39 System.out.println(s1==s2);
40

```

Line 35 and 36 are run time method call will create a new object in HEAP so the answer is FALSE

OCJA 1.8 Java SE 8 Programmer - I (1Z0 - 808 ) by Durga Sir On 29-03-2018

```
30
31 like method call...compulsory a new object will be created in the heap
32
33 -----
34 String s1= new String("Durga");
35 String s2=s1.toLowerCase();
36 String s3=s1.toLowerCase();
37 System.out.println(s2==s3);■
38
```

Unicode character of 100,101 -> d e f g

97-A 98-B 99-C 100-D

OCJA 1.8 Java SE 8 Programmer - I (1Z0 - 808 ) by Durga Sir On 29-03-2018

```
1 class Test
2 {
3     public static void main(String[] args)
4     {
5         byte[] b = {100,101,102,103};
6         String s = new String(b);
7         System.out.println(s);
8     }
9 }
10 }
```

Create string using the below ways

OCJA 1.8 Java SE 8 Programmer - I (1Z0 - 808 ) by Durga Sir On 29-03-2018

```
45
46
47 String s = new String();
48 String s = new String(String s1);
49 String s = new String("Durga");
50 String s = new String(StringBuffer sb);
51
52 String s = new String(char[] ch)
53 String s = new String(byte[] b)
54
55
```

OCJA 1.8 Java SE 8 Programmer - I (1Z0-808 ) by Durga Sir On 29-03-2018

```
1 2 3 4 5 6 7
64
65 String s = new String();
66 String s = new String(String literal);
67 String s = new String(StringBuffer sb);
68 String s = new String(char[] ch);
69
70     char[] ch = {'a', 'b', 'c', 'd'};
71     String s = new String(ch);
72     System.out.println(s); //abcd
73
74
75 String s = new String(byte[] b);
76
77 byte[] b = {100, 101, 102, 103};
78 String s = new String(b);
79 System.out.println(s); //defg
80
81
82
```

Untitled1 String\_Methods.java Test.java

For Help, press F1 39:57 / 1:01:50 In 72 col 34 1027 00 PC ANSI

```
1 2 3 4 5 6 7
97
98
99
100 public char charAt(int index);
101
102 String s = "durga";
103 System.out.println(s.charAt(3)); g
104 System.out.println(s.charAt(30));
105             RE: StringIndexOutOfBoundsException
106
107 -----
```

```
111 public String concat(String s)
112
```

+ operator for concatenation.

```
58
59 String s="Durga"+ "Software";
60
61
62
63 String s="Durga";
64 s += "Software"
65
```

X	Question	Asker	R...
yes		Amit Kumar	...
s		Govindu Rayapur	...
s		Pooja Chavan	...
s		Amit Kumar	...

GC not allowed to SCP then there may be chance of OutOfMemoryError??

```
124  
125  
126  
127  
128 The overloaded + and += operators also meant for concatenation purpose only  
129  
130 String s = "durga";  
131 s = s.concat("software");  
132 //s = s+"software";  
133 //s += "software";  
134 System.out.println(s); //durgasoftware  
135 =====
```

### Equals Ignore Case

```
OCJA 1.8 Java SE 8 Programmer - I (1Z0 - 808 ) by Durga Sir On 29-03-2018  
1 2 3 4 5 6 7 8  
69  
70 equals()=>Content comp  
71  
72 s1="Durga";  
73 s2="durga";  
74 sop(s1.equals(s2))false  
75 sop(s1.equalsIgnoreCase(s2))true
```

```
OCJA 1.8 Java SE 8 Programmer - I (1Z0 - 808 ) by Durga Sir On 29-03-2018  
1 2 3 4 5 6 7  
127  
128 public boolean equals(Object o)  
129  
130     To perform content comparison where case is important.  
131     This is overriding version of Object class equals() method  
132  
133 public boolean equalsIgnoreCase(String s)  
134     To perform content comparison where case is not important.  
135  
136 String s = "java";  
137 System.out.println(s.equals("JAVA")); //false  
138 System.out.println(s.equalsIgnoreCase("JAVA")); //true  
139  
140  
141  
142  
143 ======
```

OCJA 1.8 Java SE 8 Programmer - I (1Z0-808 ) by Durga Sir On 29-03-2018

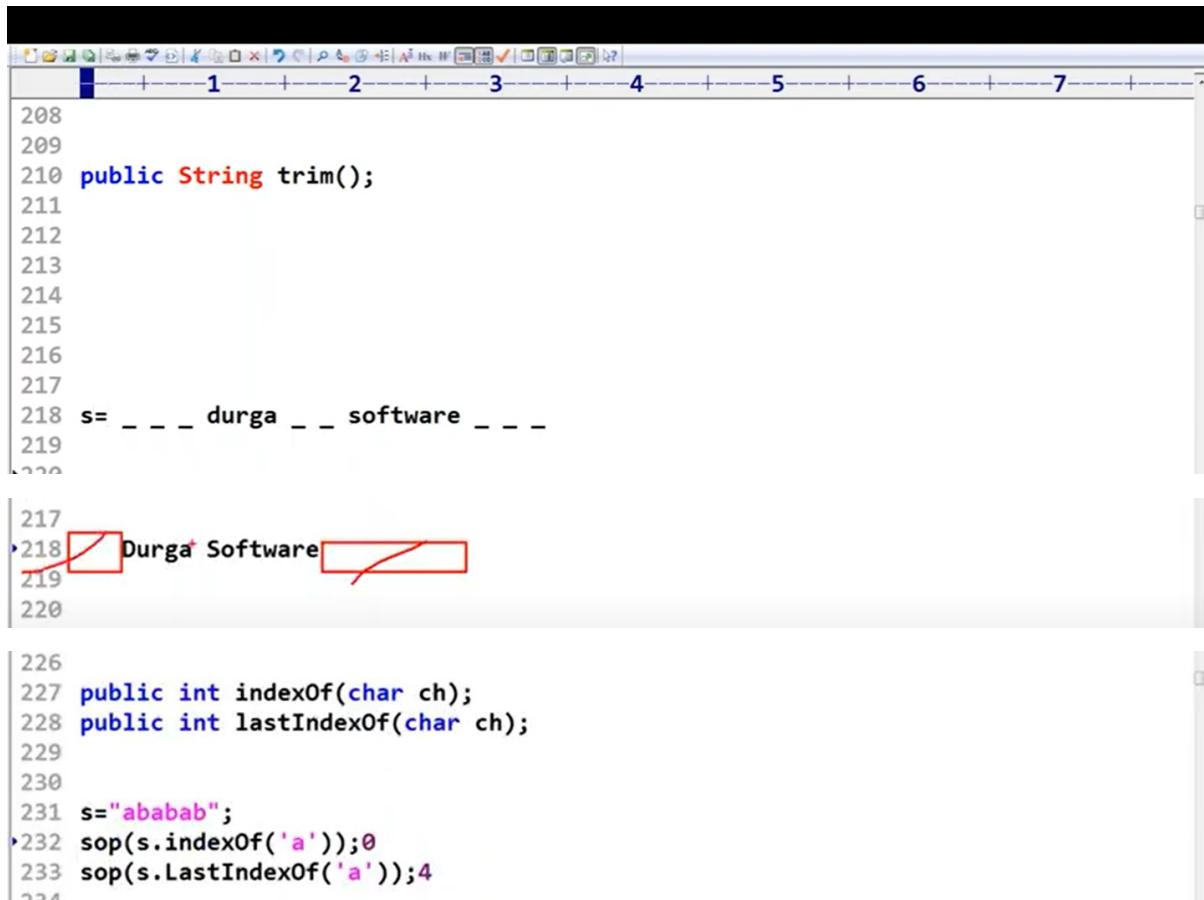
```
151 public String substring(int begin);
152     return substring from begin index to end of the String
153
154 public String substring(int begin,int end);
155     return substring from begin index to end-1 index
156
157 String s = "abcdefg";
158 System.out.println(s.substring(3));//defg
159 System.out.println(s.substring(2,5));//cde
160
```

s.length – is a variable in array wheras in String we need to call length method

```
172
173
174 =====
175 public int length()
176
177 String s = "java";
178 System.out.println(s.length());
179     CE: cannot find symbol
180             symbol: variable length
181             location:java.lang.String
182 System.out.println(s.length());4
183 -----
184
185 -----  
186
187 public String replace(char old, char new)
188
189 String s = "ababa";
190 System.out.println(s.replace('a', 'b'));  
191 //bbbbbb
192 -----
```

All white spaces will be removed. All beginning and trailing.

Here \_ is blanks just for ref

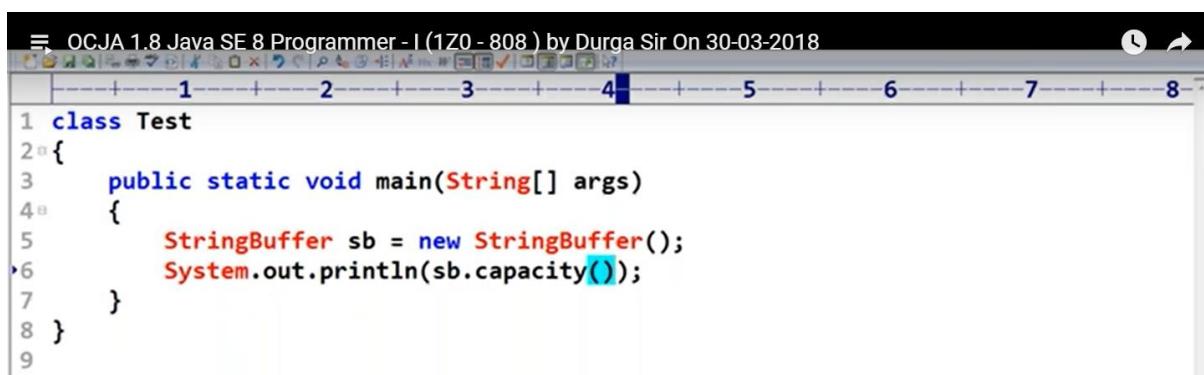


```
208
209
210 public String trim();
211
212
213
214
215
216
217
218 s=    durga   software
219
220
221
222
223
224
225
226
227 public int indexOf(char ch);
228 public int lastIndexOf(char ch);
229
230
231 s="ababab";
232 sop(s.indexOf('a'));0
233 sop(s.lastIndexOf('a'));4
234
```

If the String content keeps on changing then its gonna create a new object everytime you change it.

Then go with String Buffer or Builder. No new object will be created in SB.

String Buffer by default a capacity of 16 will be allocated . Length and Capacity are diff



```
1 class Test
2 {
3     public static void main(String[] args)
4     {
5         StringBuffer sb = new StringBuffer();
6         System.out.println(sb.capacity());
7     }
8 }
9
```

Which means u can add up to 16 chars

```
D:\durgaclasses>javac Test.java  
D:\durgaclasses>java Test  
16
```

Add 16 character and check

The screenshot shows an IDE interface with a code editor and a terminal window. The code editor displays the following Java code:

```
1 class Test
2 {
3     public static void main(String[] args)
4     {
5         StringBuffer sb = new StringBuffer();
6         System.out.println(sb.capacity());
7         sb.append("abcdefghijklmnp");
8         System.out.println(sb.capacity());
9     }
10 }
```

The terminal window below shows the output of the program:

```
D:\durgaclasses>javac Test.java  
D:\durgaclasses>java Test  
16
```

If you add 17 character.. When it reaches the max capacity, a new StringBuffer object will be created with capacity = (CurrentCapacity+1)\*2 -> 34

The screenshot shows an IDE interface with a code editor and a terminal window. The code editor displays the same Java code as before, but with an additional append operation:

```
1 class Test
2 {
3     public static void main(String[] args)
4     {
5         StringBuffer sb = new StringBuffer();
6         System.out.println(sb.capacity());
7         sb.append("abcdefghijklmnp");
8         sb.append("q");
9         System.out.println(sb.capacity());
10    }
11 }
```

The screenshot shows a terminal window titled "OCJA 1.8 Java SE 8 Programmer - I (1Z0 - 808 ) by Durga Sir On 30-03-2018". It displays the command line and the output of the Java program:

```
C:\Windows\system32\cmd.exe  
D:\durgaclasses>javac Test.java  
D:\durgaclasses>java Test  
16  
34
```

Add 34 chars-> no impact will be 34 capacity only

The screenshot shows a Java code editor with the following code:

```
1 class Test
2 {
3     public static void main(String[] args)
4     {
5         StringBuffer sb = new StringBuffer();
6         System.out.println(sb.capacity());
7         sb.append("abcdefghijklmnp");
8         sb.append("q");
9         sb.append("abcdefghijklmnp");
10        sb.append("q");
11        System.out.println(sb.capacity());
12    }
13 }
14
```

The code uses a `StringBuffer` object to append strings and then prints its capacity. The lines from 7 to 10 are highlighted in blue.

Add 35 char

34+1-> 35\*2-> 70

The terminal window shows the following output:

```
D:\durgaclasses>javac Test.java
D:\durgaclasses>java Test
16
70
```

Everytime new object will be created with new capacity and the ref var will be repointed and the old object will be eligible for garbage collection. Rather than doing this , can we specify the capacity

```
22 1. SB sb = new SB();
23     16,34,70....
24
25 2. SB sb = new SB(int capacity);
26   SB sb = new SB(1000);
27 (cc+1)*2
28 I
29 3. SB sb = new SB(String s)
30 |
```

21

If you pass a String , creates a capacity = 16+Length of the String -> 21

The screenshot shows a Java IDE interface with two panes. The top pane displays the code for `Test.java`:

```
1 class Test
2 {
3     public static void main(String[] args)
4     {
5         StringBuffer sb = new StringBuffer("durga");
6         System.out.println(sb.capacity());
7     }
8 }
```

The bottom pane shows the terminal output:

```
D:\durgaclasses>javac Test.java
D:\durgaclasses>java Test
21
```

The screenshot shows a Java IDE interface with two panes. The top pane displays the code for `String_Methods.java`:

```
304
305
306 StringBuffer sb = new StringBuffer(String s);
307 capacity = s.length()+16
```

The bottom pane shows the terminal output:

```
316
317
318
319 public int length();
320 public int capacity();
321 public char charAt(int index);
322
323 StringBuffer sb = new StringBuffer("durga");
324 System.out.println(sb.charAt(3));g
325 System.out.println(sb.charAt(30));
326     RE:StringIndexOutOfBoundsException
327
328
329
330 public void setCharAt(int index,char ch);
331     To replace the character locating at specified index with provided character
332
333
334
```

The status bar at the bottom indicates: In 322 col 1 990 09 PC ANSI.

```
OCJA 1.8 Java SE 8 Programmer - I (1Z0-808 ) by Durga Sir On 30-03-2018
1 class Test
2 {
3     public static void main(String[] args)
4     {
5         StringBuffer sb = new StringBuffer("durga");
6         sb.setCharAt(2, 'Z');
7         System.out.println(sb);
8     }
9 }
10
```

```
OCJA 1.8 Java SE 8 Programmer - I (1Z0-808 ) by Durga Sir On 30-03-2018
1 class Test
2 {
3     public static void main(String[] args)
4     {
5         StringBuffer sb = new StringBuffer("durga");
6         sb.append("software");
7         System.out.println(sb);
8     }
9 }
10
```

Append can take any arg

```
1 class Test
2 {
3     public static void main(String[] args)
4     {
5         StringBuffer sb = new StringBuffer("durga");
6         sb.append("software");
7         sb.append(10);
8         sb.append(10.5);
9         sb.append(true);
10        sb.append(20);
11        System.out.println(sb);
12    }
13 }
14
```

```
D:\durgaclasses>javac Test.java
D:\durgaclasses>java Test
durgasoftware1010.5true20
.....
```

```
OCJA 1.8 Java SE 8 Programmer - I (1Z0 - 808 ) by Durga Sir On 30-03-2018
```

```
1 class Test
2 {
3     public static void main(String[] args)
4     {
5         StringBuffer sb = new StringBuffer();
6         sb.append("PI Value is:");
7         sb.append(3.14);
8         sb.append(" It is exactly: ");
9         sb.append(true);
10        System.out.println(sb);   I
11    }
12 }
13
```

```
D:\durgaclasses>java Test
PI Value is:3.14 It is exactly: true

D:\durgaclasses>
```

Append adds at the end , insert add at the specified index and performs a shift

```
OCJA 1.8 Java SE 8 Programmer - I (1Z0 - 808 ) by Durga Sir On 30-03-2018
```

```
1 class Test
2 {
3     public static void main(String[] args)
4     {
5         StringBuffer sb = new StringBuffer();
6         sb.append("durgasoftwaresolutions");
7         sb.insert(2,"aaaaaaaa");
8         System.out.println(sb);   I
9     }
10 }
```

```
D:\durgaclasses>java Test
duaaaaaaaaargasoftwaresolutions
```

```

403
404 public StringBuffer delete(int begin,int end)
405     To delete characters locating from begin index to end-1 index
406 public StringBuffer deleteCharAt(int index)
407
408
409
410
411
412
413
414
415
416
417
418 public StringBuffer reverse();
419

```

```

35
36 durgasoftwaresolutions
37
38 8 characters
39 sb.setLength(8)
40

```

```

OCJA 1.8 Java SE 8 Programmer - I (1Z0 - 808) by Durga Sir On 30-03-2018
1 class Test
2 {
3     public static void main(String[] args)
4     {
5         StringBuffer sb = new StringBuffer("aiswaryaabhi");
6         sb.setLength(8);
7         System.out.println(sb); //aiswarya
8     }
9 }
10

```

To delete extra allocated free memory

```

42
43 SB sb = new SB(1000);
44 sb.append("abc");
45 sb.trimToSize();
46
47

```

```

OCJA 1.8 Java SE 8 Programmer - I (1Z0 - 808) by Durga Sir On 30-03-2018
1 class Test
2 {
3     public static void main(String[] args)
4     {
5         StringBuffer sb = new StringBuffer(1000);
6         sb.append("abc");
7         sb.trimToSize();
8         System.out.println(sb.capacity()); //3
9     }
10 }

```

The screenshot shows the Oracle Java Development Kit (JDK) interface. The top window is a code editor with the following Java code:

```
1 class Test
2 {
3     public static void main(String[] args)
4     {
5         StringBuffer sb = new StringBuffer(1000);
6         sb.append("abc");
7         sb.trimToSize();
8         System.out.println(sb.capacity()); //3
9     }
10}
11
12
```

The code uses a `StringBuffer` object with a capacity of 1000, appends the string "abc", trims it to size, and then prints its capacity, which is expected to be 3. The code editor has a horizontal ruler at the top with numbered markers from 1 to 8.

Below the code editor is a terminal window showing the command line interface:

```
Untitled65 > Test.java > String_Methods.java
```

The terminal window also displays the current time and date: 40:25 / 51:14.