Session:10

Type of Variables

String, StringBuffer & StringBuilder

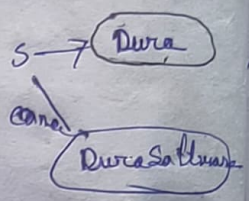
String:

- In project, if there are 1000 object in there, then >200 objects are string-object and <100 object are non-string object.

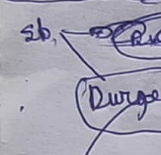
String:

case 1:

```
String s = new String("Durga");
s.concat("software");
S.O.P(s); // Durga
```




```
StringBuffer sb = new StringBuffer("Durga");
sb.append("Software");
S.O.P(sb); // Durga Software
```



case 2

==operator Vs equals() method

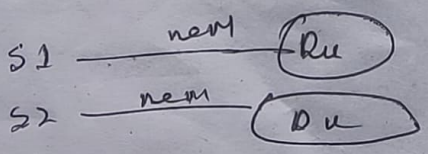
```
String s1 = new String("Durga");
```

```
String s1 = new String("Du");
String s2 = new String("Du");
```

==always use for reference comparison

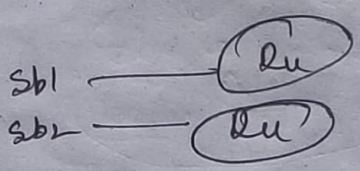
```
S.O.P(s1==s2); // False
S.O.P(s1.equal(s2)); // True
```

(content) use for content comparison




```
StringBuffer sb1 = new StringBuffer("Du");
StringBuffer sb2 = new StringBuffer("Du");
```

```
S.O.P(sb1==sb2); // False
S.O.P(sb1.equal(sb2)); // False
```

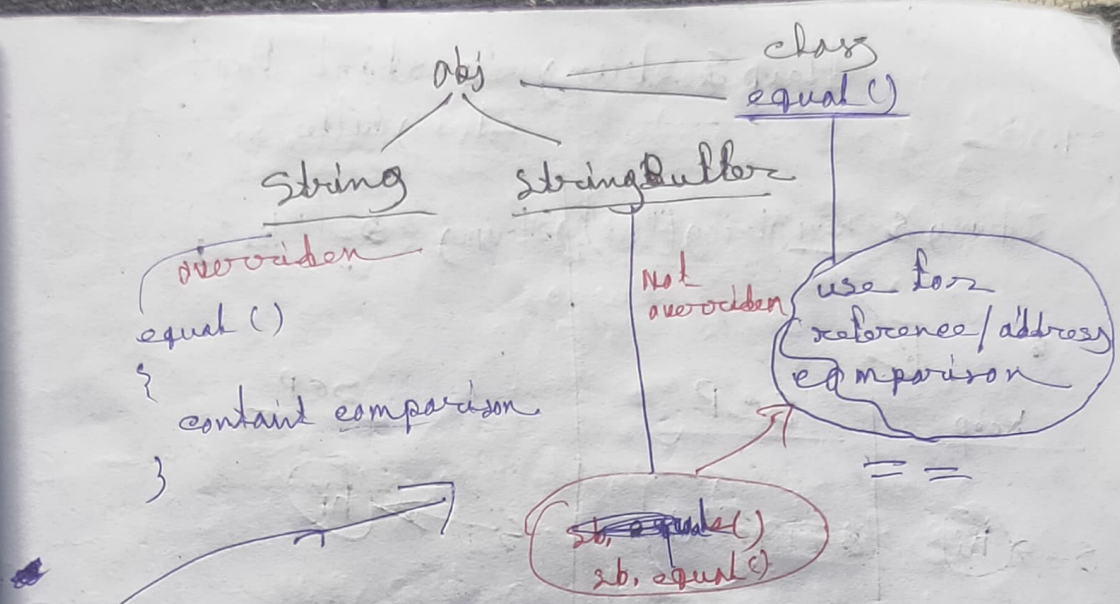


* In
always
in
refere

* The
the
refer
not

In
overrid

In



* In case of String object equal() method always use for content comparison. But in case of StringBuffer object use for reference comparison.

* The equal() method is present in the object class and the equal() use for reference/address comparison (==) only, But not for content comparison.

In String class equal() method are overriden used for content comparison.

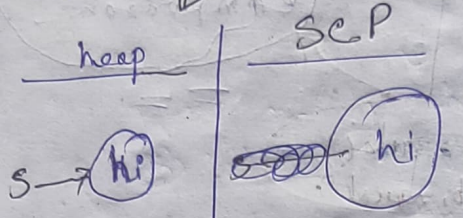
In

Heap & String Constant Pool

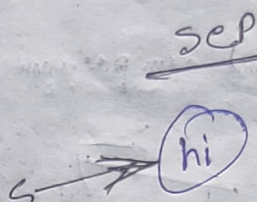
37:53

String literal.

String s = new String("hi");



String s = "hi";



ICU method Area/PERM GEN

1.7v Heap.

eg: s is reference variable.

- In this case two object will be created.
- new operator use to create object in the heap area.

- In this case only one object is created.
- For every string literal string object will created in SCP area

String Constant Pool

- Object creation in the heap area always mandatory when we use new operator.

In case of string literal

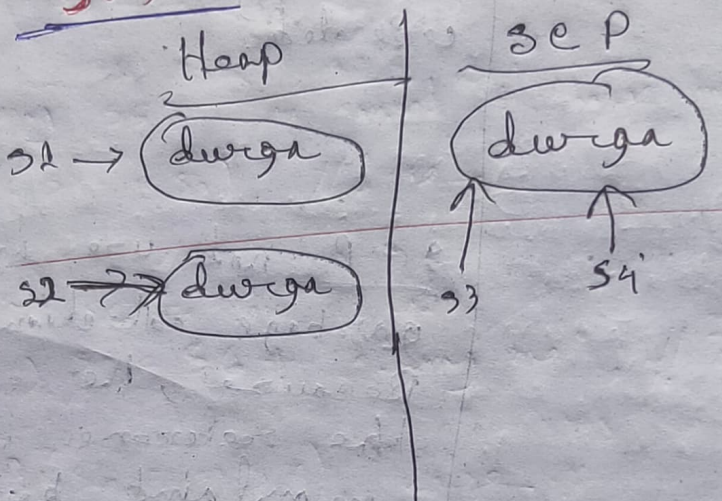
- Here, object creation is not mandatory, first the JVM will check in the SCP area do you have any object with this contains ("hi") or not. If obj is there, then reuse the same object, if not there then only obj will be created.

SCP not use in garbage collection. Because, here the reference is maintain by JVM.

```
String s1 = new String("durga");
String s2 = new String("durga");
String s3 = "durga";
String s4 = "durga"
```

How many object is created?

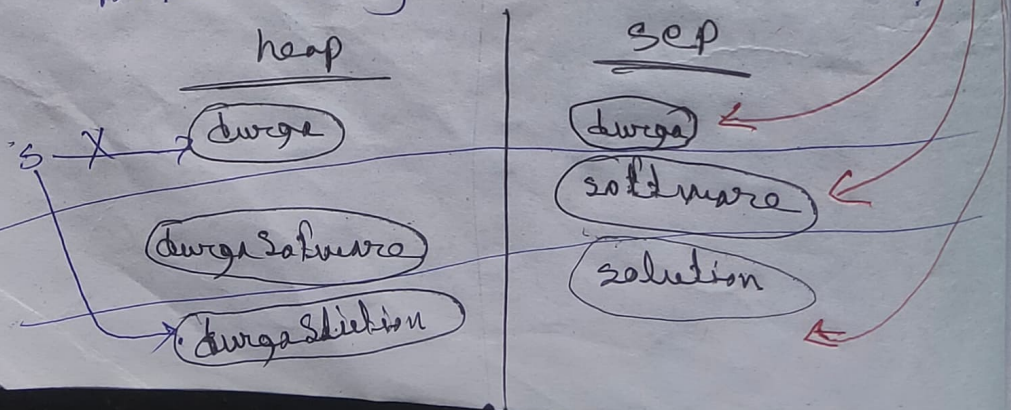
3 object



Ex

```
String s = new String("durga");
s.concat("software");
s = s.concat("solution");
```

Total How many object is created?

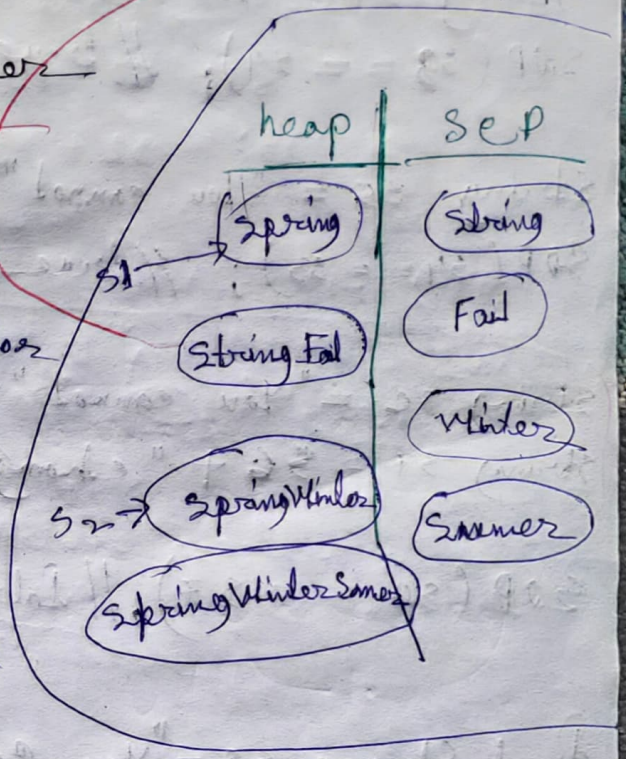


56:26
Part-11

```
String s1 = new String("Spring");
s1.concate("Fall");
String s2 = s1.concate("Winter");
s2.concate("Summer");
S.O.P(s1) // Spring
S.O.P(s2) // SpringWinner
```

runtime / method call

runtime



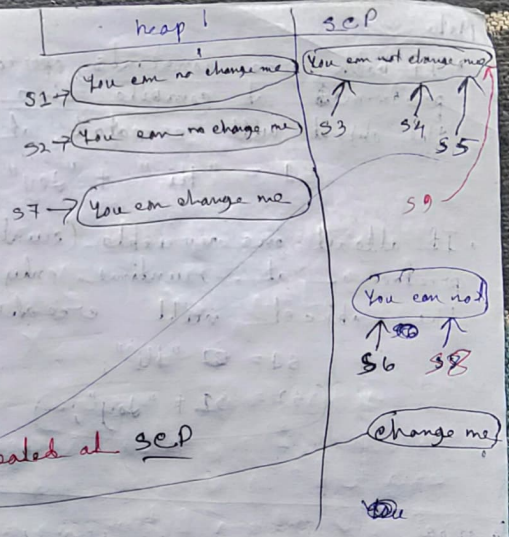
* For every new operator one object will created in heap area, For every string literal ("Spring") or s1 = "String" one object will created in the SCP.

* Because of runtime operation, like method called, if a new string object is requaire to create, the object will created in heap area

9.2.20

```
String s1 = new String("You cannot change me");
String s2 = new String("You cannot change me");
S.O.P.(s1 == s2); // false  reference comparison
String s3 = "You cannot change me"; // SCP
SOP(s1 == s3); // false
String s4 = "You cannot change me"; // SCP
SOP(s3 == s4); // true
String s5 = "You cannot" + "change me"; // In compile time, as both the are constant, object created at SCP
SOP(s4 == s5); // true
String s6 = "You cannot"
String s7 = s6 + "change me"; // In operation, if any variable is there, then the operation will preformed at runtime, so object will created at heap
SOP(s6 == s7); // false
final String s8 = "You cannot"
String s9 = s8 + "change me"; // s8 is a final variable which is a constant, (every final variable will be replace by compiler only) so s8 - is constant "change me" is constant, both are constant, so, the compiler will do the operation So object will created in SCP
SOP(s8 == s9); // true
```

compilation will done



Note

- If both are constants, operation will performed at compile time only and the object will created at SCP area

```
S1 = "Hi" + "Joy";
```

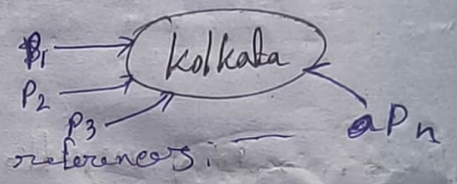
- If atleast one variable (runtime variable), operation will perform at runtime only and the object will created at heap area.

```
S1 = "Hi";
S2 = S1 + "Joy";
```

1.17.00

SCP - String Object Pool

* Same object in SCP will be referenced with multiple references, so memory utilization will be improve.



By one reference (P_3) try to change the content of the object, then all the remaining references will be effected.

To prevent ~~that~~ this problem, immutability concept is introduced by StringBuffer.

* Immutability is require because of the SCP. As if the reuseability of same concepts is not there immutability not require.

1.19.36

Importance of SCP: (String Constant Pool)

1:34:02

FAQ:

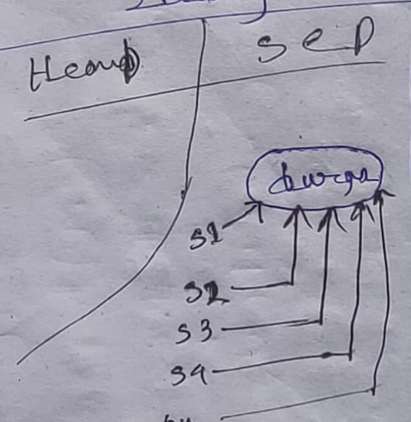
① Why SCP concept is available only for string object but not for StringBuffer?

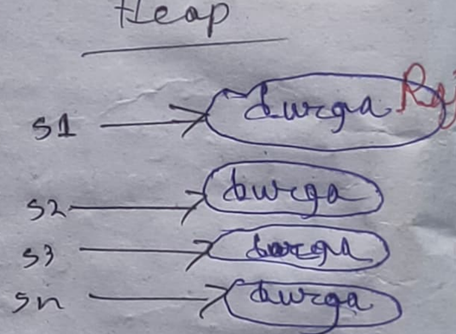
String ✓ ← SCP   StringBuffer ✗

String is most commonly used object in java, that why a special memory management is provided called SCP. But the StringBuffer is not regular used object. So no SCP is there.

② Why String object is immutable where as StringBuffer is mutable?

* String | StringBuffer





Because of SCP, same object can be reuse multiple time. By using one reference If we are try change containg of the object, then the remaing referencees will be effected. to prevent That ~~is why~~ immutability

Here SCP concept is not present, we cannot reuse same object is not possible. Every time seperated object will be created. If we are try to change containg of one object, then remaing referenced object will not be

concept is required.

affected. So immutability is not required.

(3) In addition to String object any other object are immutable or not?

All wrapper class objects are also immutable

In wrapper class, after a set of range Wrapper class object can be reused.

Integer Float

Byte

* char[] ch = {'j', 'a', 'v', 'a'}

String s = new String(ch);

SOP(s) // java

* byte[] b = {97, 98, 99};

String s = new String(b);

SOP(s); // abc

* String s = "durga"

SOP(s.length()); // 5

↓ method

int x = {1, 2, 3}

SOP(x.length) // 3

↓ variable

(2) * String s = "durga"

~~s.index~~

SOP(s.indexOf('z')); // -1

3.15.20 — String Immutability!
Creation of our own Immutable class

All the immutable class are declare as final.

3:26:00 — Final vs immutability.

By declaring a reference variable as final we won't give immutability nature.

Final	Immutable
① final is related to reference variable.	① Immutability is related to object

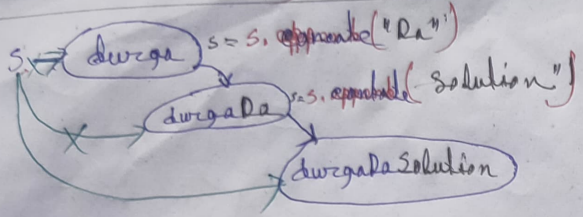
```
class Test
{
  psvm(---)
  {
    final StringBuffer sb
        = new StringBuffer
            ("durga");
    sb.append("software");
    SOP(sb) // durgasoftware
    sb = new StringBuffer("Raj");
  }
}
```

CE

reassignment is not possible

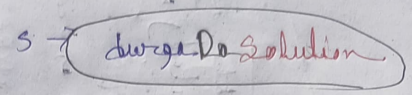
4:03:21

String



For every change in string object new object will be created.

StringBuffer



All the required changes will be performed in existing object only. For every changes, new object not will be created.

```
public class Test
{
    p s v main(String[] args)
    {
        StringBuilder sb = new StringBuilder(5);
        String s = "";
        if (sb.equals(s)) false
        {
            Sopln("match1");
        }
        else if (sb.toString().equals(s.toString())) true
        {
            Sopln("match2");
        }
        else
        {
            Sopln("no match");
        }
    }
}
```

sb →

s →

the Result ?

b) match2 c) no match

exception is thrown at runtime

Q7. Given the code fragment:

```
public class Test
{
    public static void main(String[] args)
    {
        StringBuilder sb1= new StringBuilder("Durga");
        String str1=sb1.toString();
        //insert code here==>Line-1
        System.out.println(str1==str2);
    }
}
```

Which code fragment,when inserted at Line-1,enables the code to print true?

A. String str2=str1;
B. String str2=new String(str1);
C. String str2=sb1.toString();
D. String str2="Durga";

Answer: A

Explanation: str1==str2 returns true iff both references pointing to the same object

```
01. Q8. Which statement will empty the contents of a StringBuilder variable named sb?
02.
03. A. sb.deleteAll();
04. B. sb.delete(0,sb.size());
05. C. sb.delete(0,sb.length());
06. D. sb.removeAll();
07.
08. Answer: C
09.
10. Explanation:
11. On the StringBuilder object we cannot apply deleteAll(),size() and removeAll() methods.
12. But we can apply delete() and length() methods.
13. sb.delete(begin,end) removes all characters from begin index to end-1 index.
14. sb.delete(0,sb.length()) removes all characters from the StringBuilder.
```

Q) Given the following code:

```
class myString
{
    String msg;
    myString(String msg)
    {
        this.msg = msg;
    }
}
public class Test
{
    p s v main(String[] args)
    {
        Sopln("Hello " + new StringBuilder("Java SE 8"));
        Sopln("Hello " + new myString("Java SE 8"));
    }
}
```

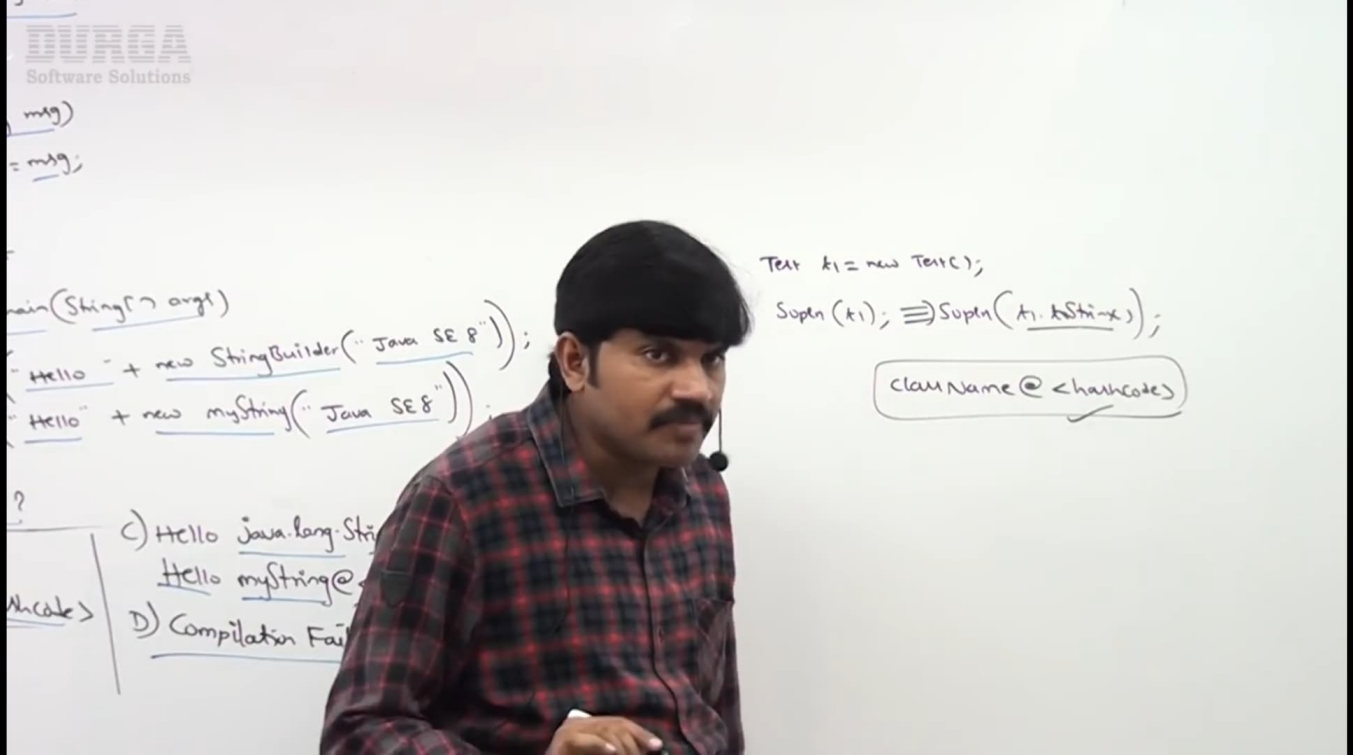
what is the result?

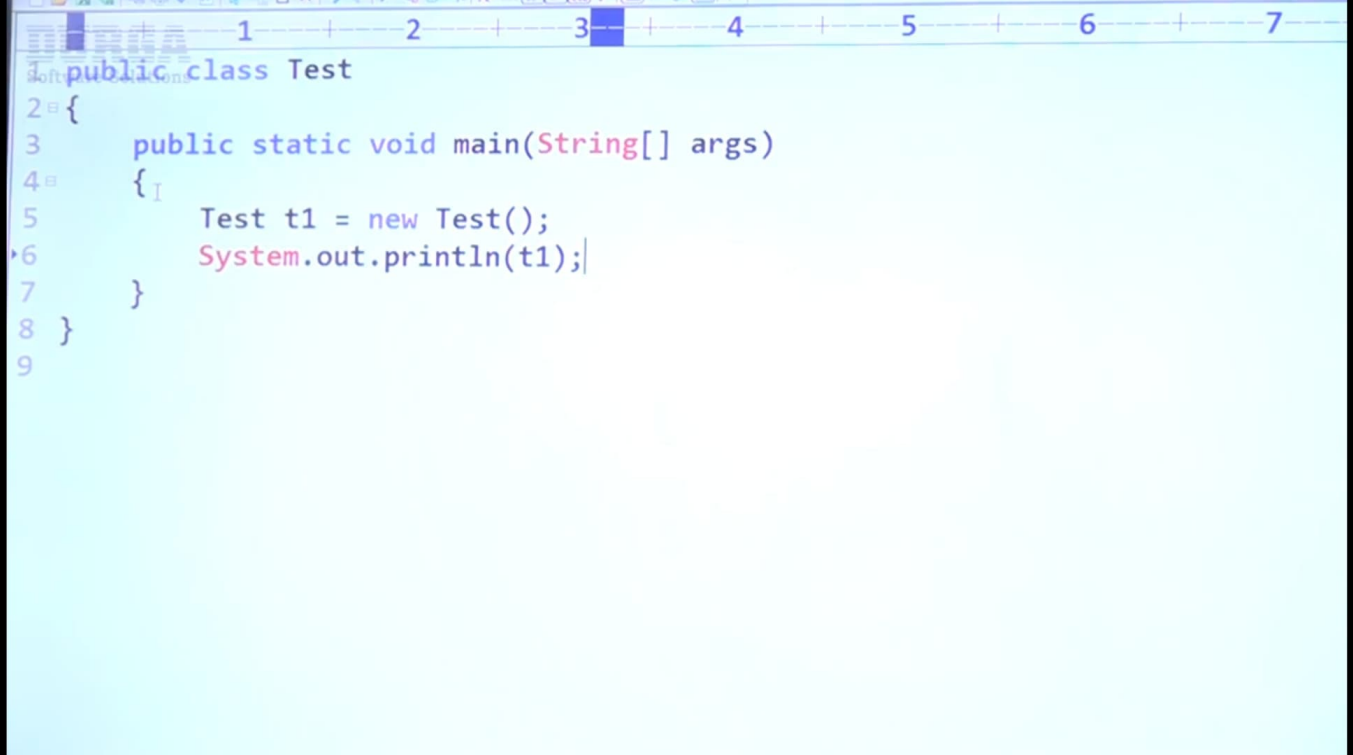
A) Hello Java SE 8
   Hello myString@<hashcode>

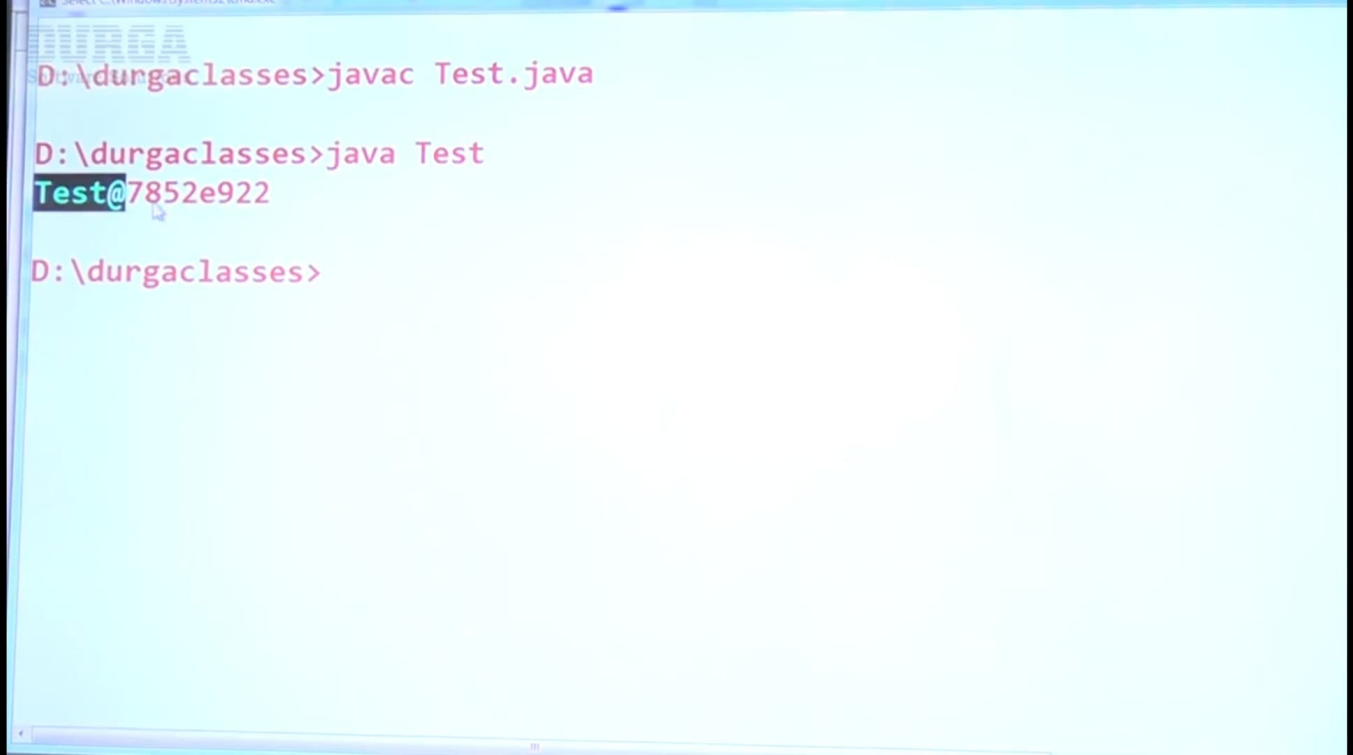
B) Hello Java SE 8
   Hello Java SE 8

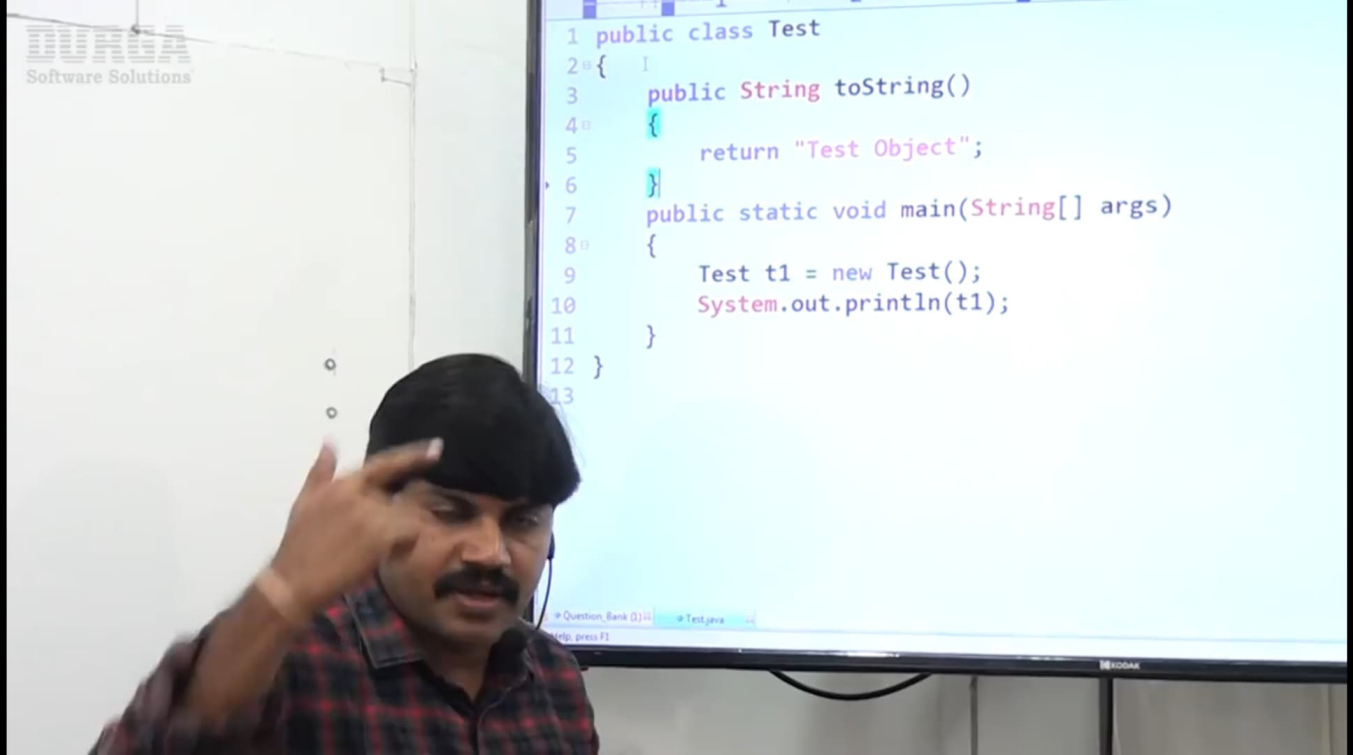
C) Hello java.lang.StringBuilder@<hashc
   Hello myString@<hashcode>

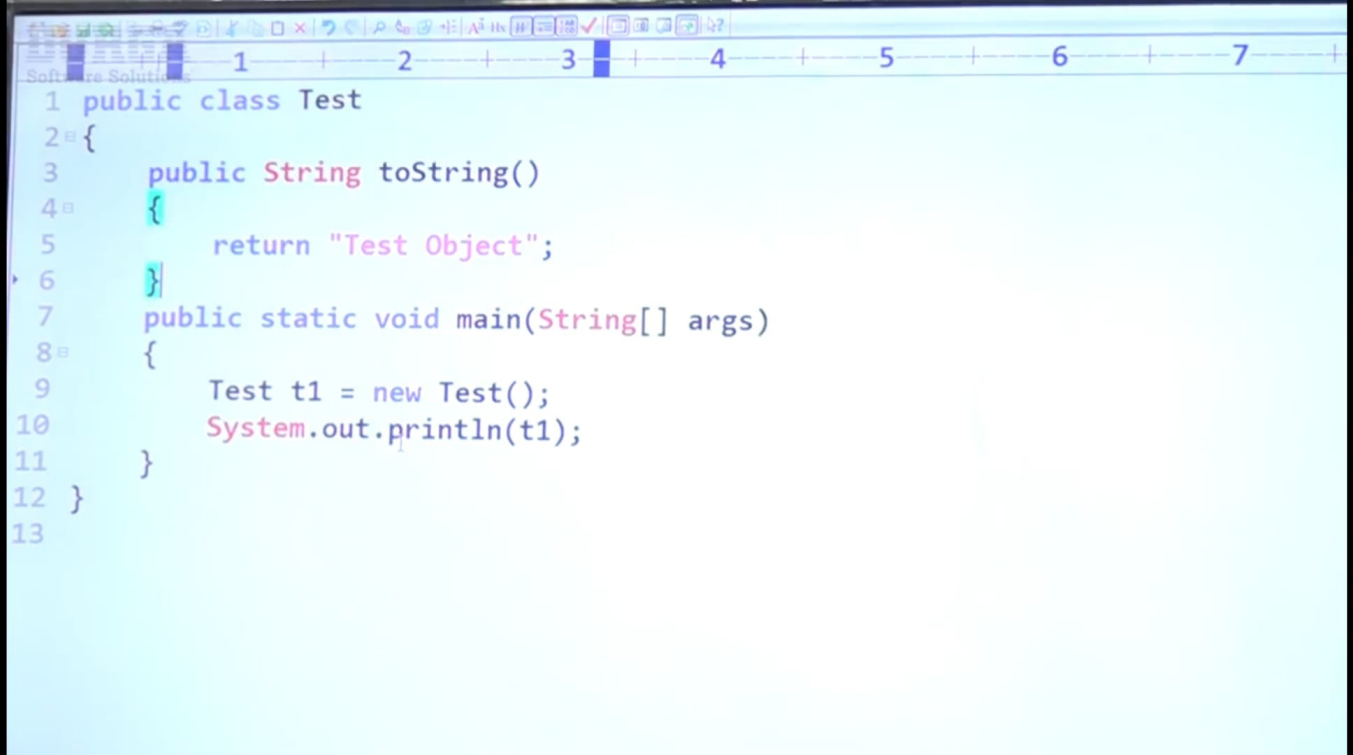
D) Compilation Fails

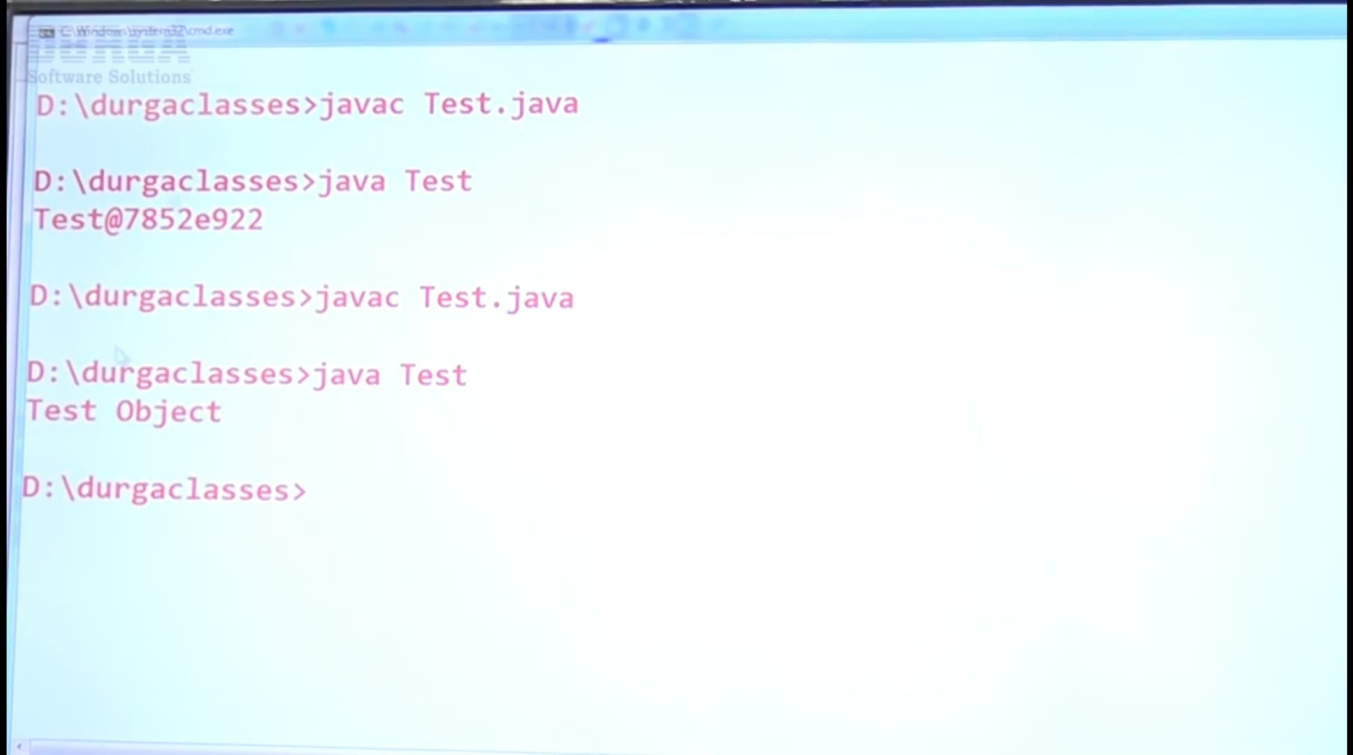

DURGA
Software Solutions
msg)
= msg;
Test t1 = new Test();
Sopln(t1); ⟹ Sopln(t1.toString());
className@<hashcode>
Hello" + new StringBuilder("Java SE 8"));
Hello" + new myString("Java SE 8"));
C) Hello java.lang.Str
Hello myString@
D) Compilation Fai

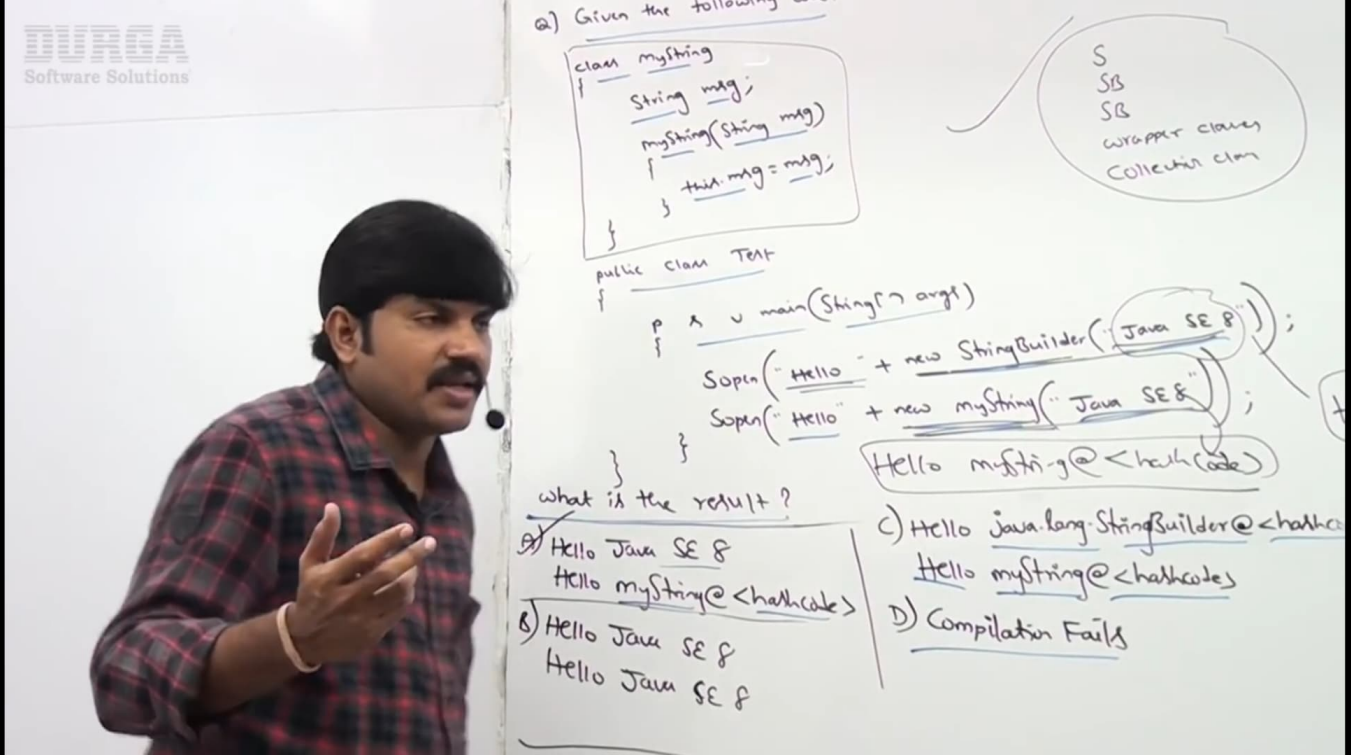

1 2 3 4 5 6 7
1 public class Test
2 {
3 public static void main(String[] args)
4 {
5 Test t1 = new Test();
6 System.out.println(t1);
7 }
8 }
9

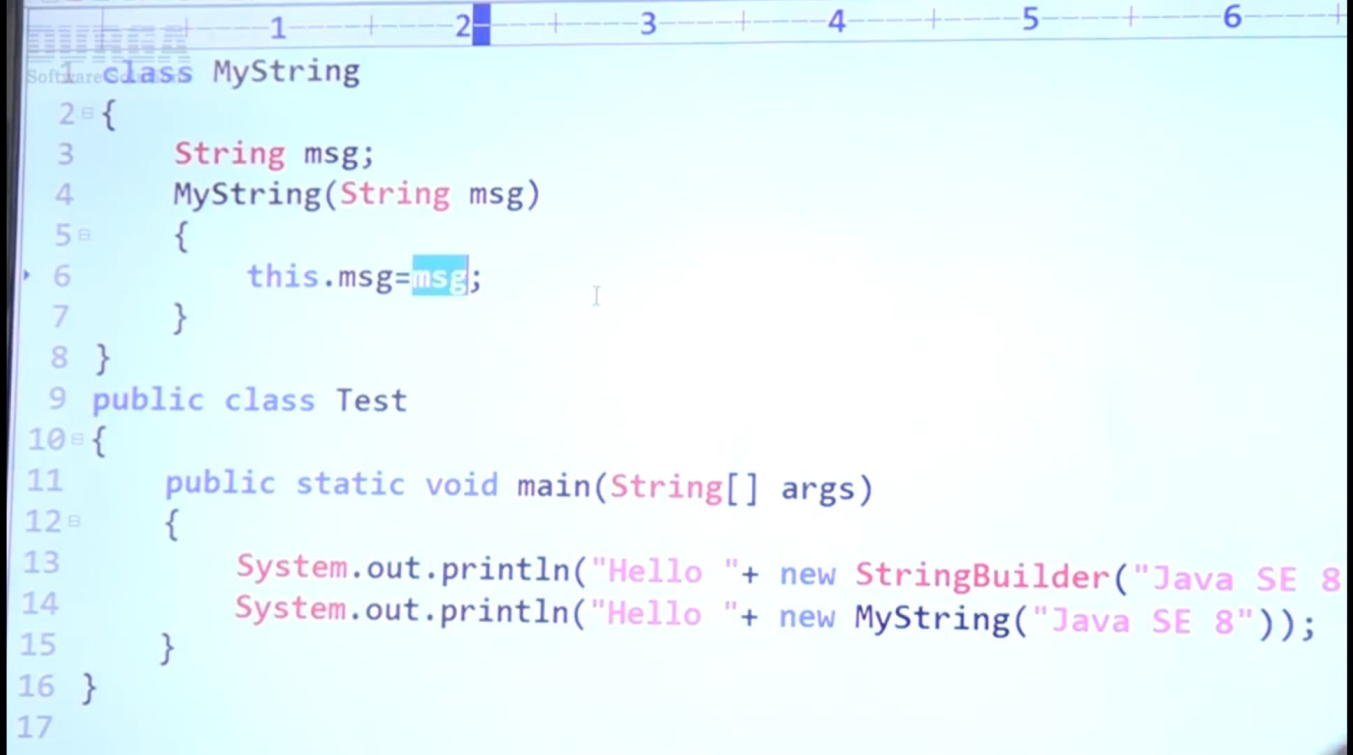

D:\durgaclasses>javac Test.java
D:\durgaclasses>java Test
Test@7852e922
D:\durgaclasses>


DURGA
Software Solutions
public class Test
{
public String toString()
{
return "Test Object";
}
public static void main(String[] args)
{
Test t1 = new Test();
System.out.println(t1);
}
}


1 2 3 4 5 6 7
1 public class Test
2 {
3 public String toString()
4 {
5 return "Test Object";
6 }
7 public static void main(String[] args)
8 {
9 Test t1 = new Test();
10 System.out.println(t1);
11 }
12 }
13


C:\Windows\system32\cmd.exe
Software Solutions
D:\durgaclasses>javac Test.java
D:\durgaclasses>java Test
Test@7852e922
D:\durgaclasses>javac Test.java
D:\durgaclasses>java Test
Test Object
D:\durgaclasses>


DURGA
Software Solutions
Q) Given the following
class MyString
{
String msg;
MyString(String msg)
{
this.msg = msg;
}
}
public class Test
{
p s v main(String[] args)
{
Sopln("Hello" + new StringBuilder("Java SE 8"));
Sopln("Hello" + new MyString("Java SE 8"));
}
}
S
SB
SB
wrapper class
Collection clas
Hello MyStri-g@<hashcode>
what is the result ?
A) Hello Java SE 8
Hello MyString@<hashcode>
B) Hello Java SE 8
Hello Java SE 8
C) Hello java.lang.StringBuilder@<hashc
Hello MyString@<hashcode>
D) Compilation Fails


1 2 3 4 5 6
1 class MyString
2 {
3 String msg;
4 MyString(String msg)
5 {
6 this.msg=msg;
7 }
8 }
9 public class Test
10 {
11 public static void main(String[] args)
12 {
13 System.out.println("Hello "+ new StringBuilder("Java SE 8
14 System.out.println("Hello "+ new MyString("Java SE 8"));
15 }
16 }
17

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

D:\durgaclasses>java Test
Hello Java SE 8
Hello MyString@7852e922

D:\durgaclasses>
```

Q9. Given the following code:

```
class MyString
{
    String msg;
    MyString(String msg)
    {
        this.msg=msg;
    }
}
public class Test
{
    public static void main(String[] args)
    {
        System.out.println("Hello "+ new StringBuilder("Java SE 8"));
        System.out.println("Hello "+ new MyString("Java SE 8"));
    }
}
```

What is the result?

A.
Hello Java SE 8
Hello MyString@<hashcode>

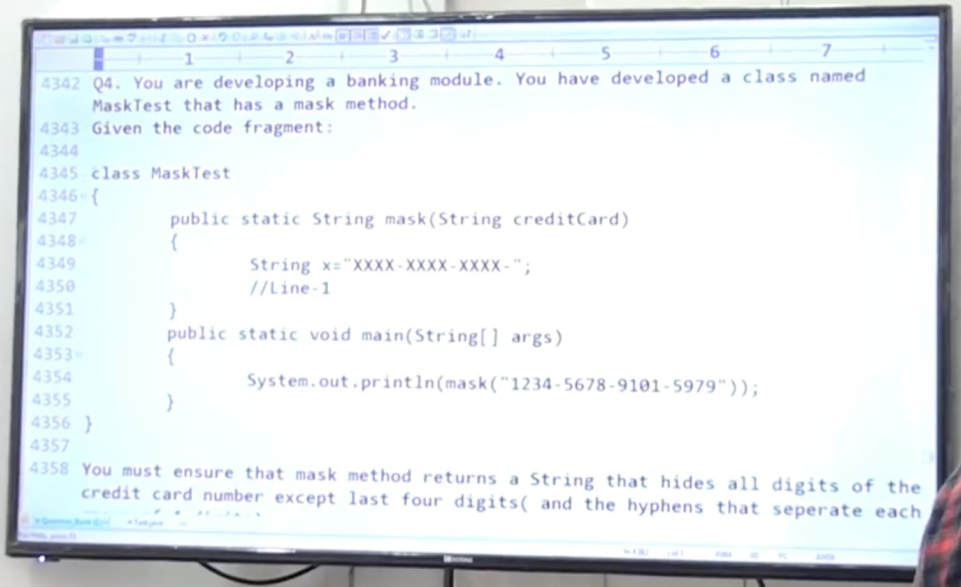
B.
Hello Java SE 8
Hello Java SE 8

C.
Hello java.lang.StringBuilder@<hashcode>
Hello MyString@<hashcode>

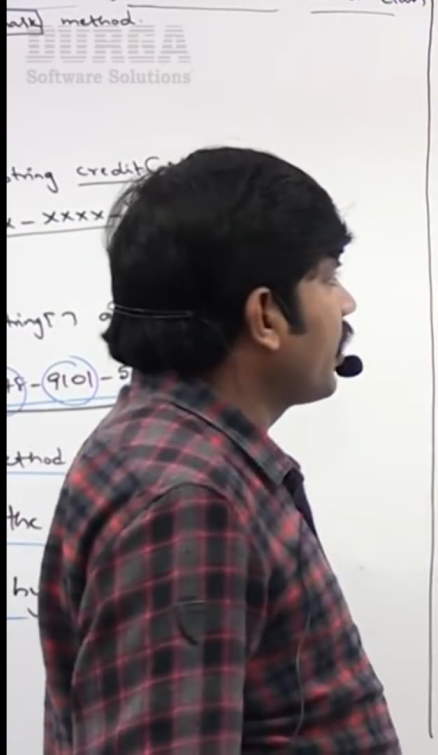
D. Compilation Fails

Answer: A

Explanation:
In StringBuilder class toString() method is overridden for meaningful String representation.
But in MyString class toString() method is not overridden and hence object class toString() method will be called which returns the String in the following format: classname@<hashcode_in_hexadecimal_string_form>



```
4342 Q4. You are developing a banking module. You have developed a class named
     MaskTest that has a mask method.
4343 Given the code fragment:
4344
4345 class MaskTest
4346 {
4347         public static String mask(String creditCard)
4348         {
4349                 String x="XXXX-XXXX-XXXX-";
4350                 //Line-1
4351         }
4352         public static void main(String[] args)
4353         {
4354                 System.out.println(mask("1234-5678-9101-5979"));
4355         }
4356 }
4357
4358 You must ensure that mask method returns a String that hides all digits of the
     credit card number except last four digits( and the hyphens that seperate each
```



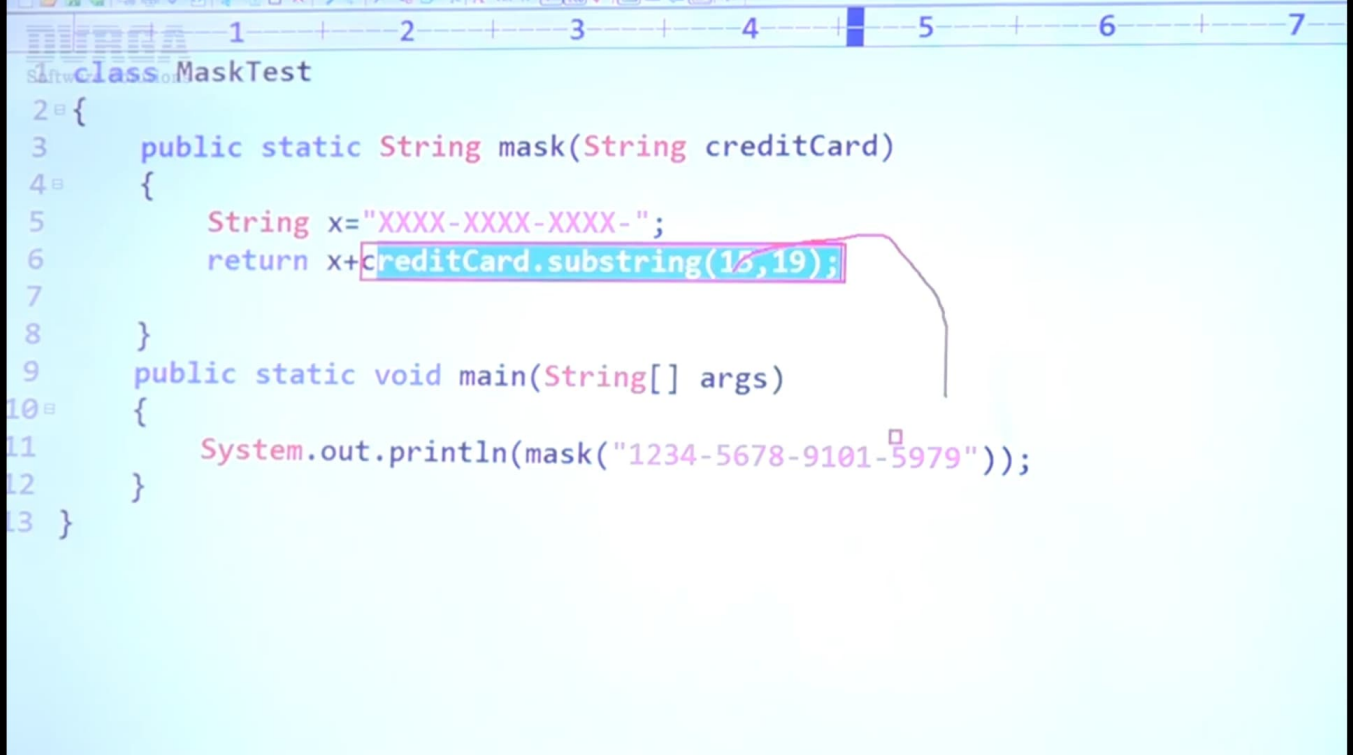
which two code fragments should you use at Line-1, independently to achieve the requirement ?

A)
```
StringBuilder sb = new StringBuilder(creditCard);
sb.subString(15,19);
return x+sb;
```
1234-5678-9101-5979

B) `return x+creditCard.subString(15,19);`

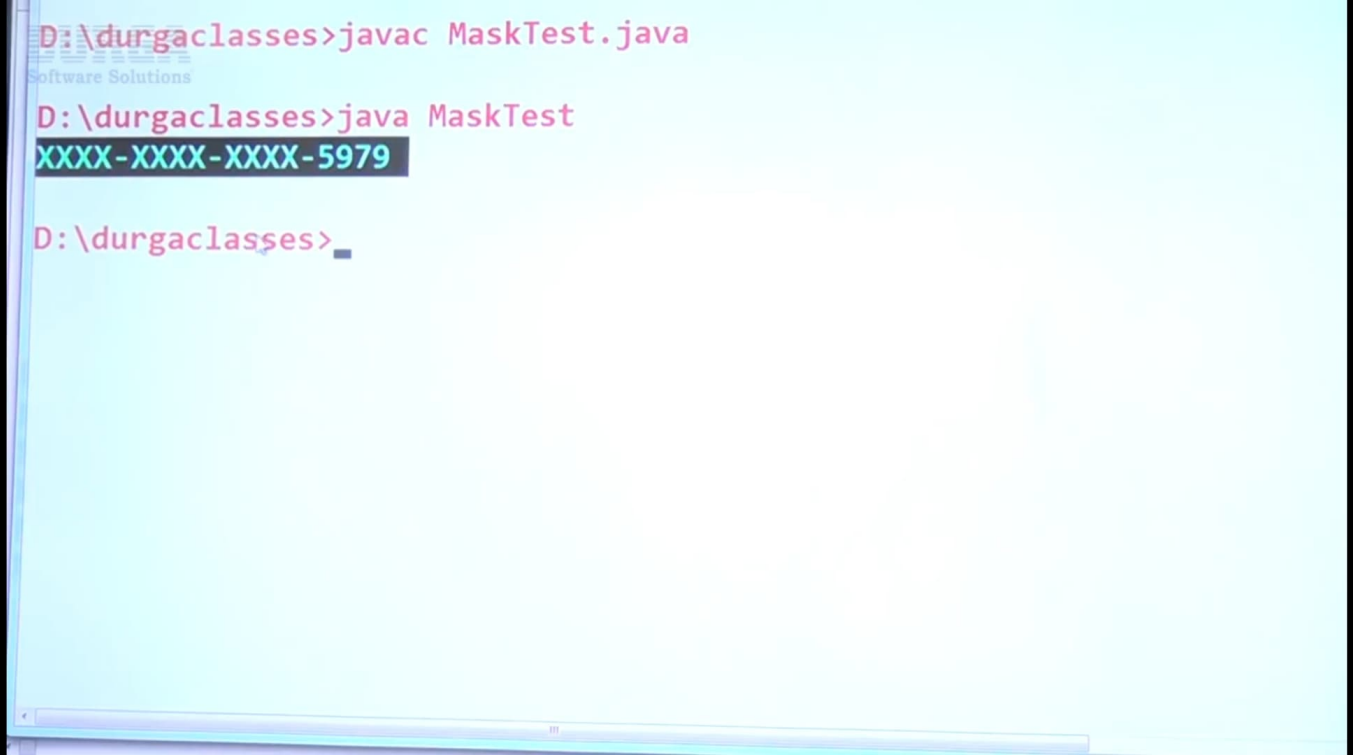
C)
```
StringBuilder sb = new StringBuilder(x);
sb.append(creditCard,15,19)
return sb.toString();
```

D)
```
StringBuilder sb = new StringBuilder(creditCard);
StringBuilder s = sb.insert(0,x);
return s.toString();
```


1 2 3 4 5 6 7
class MaskTest
{
public static String mask(String creditCard)
{
String x="XXXX-XXXX-XXXX-";
return x+creditCard.substring(15,19);
}
public static void main(String[] args)
{
System.out.println(mask("1234-5678-9101-5979"));
}
}

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

D:\durgaclasses>java MaskTest
XXXX-XXXX-XXXX-1234-5678-9101-5979

D:\durgaclasses>javac MaskTest.java

D:\durgaclasses>java MaskTest
XXXX-XXXX-XXXX-5979

D:\durgaclasses>
```

```java
class MaskTest
{
    public static String mask(String creditCard)
    {
        String x='XXXX-XXXX-XXXX-';
        StringBuilder sb=new StringBuilder(x);
        sb.append(creditCard,15,19);
        return sb.toString();

    }
    public static void main(String[] args)
    {
        System.out.println(mask("1234-5678-9101-5979"));
    }
}
```


D:\durgaclasses>javac MaskTest.java
Software Solutions
D:\durgaclasses>java MaskTest
XXXX-XXXX-XXXX-5979
D:\durgaclasses>

Q10. You are developing a banking module. You have developed a class named MaskTest that has a mask method.
Given the code fragment:

```
class MaskTest
{
    public static String mask(String creditCard)
    {
        String x="XXXX-XXXX-XXXX-";
        //Line-1
    }
    public static void main(String[] args)
    {
        System.out.println(mask("1234-5678-9101-5979"));
    }
}
```

You must ensure that mask method returns a String that hides all digits of the credit card number
except last four digits(and the hyphens that seperate each group of 4 digits)

Which two code fragments should you use at line 1, independently to achieve the requirement?

A.
```
StringBuilder sb=new StringBuilder(creditCard);
sb.substring(15,19);
return x+sb;
```

B.
```
return x+creditCard.substring(15,19);
```

C.
```
StringBuilder sb=new StringBuilder(x);
sb.append(creditCard,15,19);
return sb.toString();
```

D.
```
StringBuilder sb=new StringBuilder(creditCard);
StringBuilder s=sb.insert(0,x);
return s.toString();
```

Answer: B,C

Explanation:
Only in the following 2 cases the digits will be hidden and in remaining cases all
digits will be displayed to the end user.

1. `return x+creditCard.substring(15,19);`
2.
```
StringBuilder sb=new StringBuilder(x);
sb.append(creditCard,15,19);
return sb.toString();
```

Important Constructors of String class:

1. String s = new String();
   → Creates an empty String object
2. String s = new String(String literal);
3. String s = new String(StringBuffer sb);
4. String s = new String(StringBuilder sb);
5. String s = new String(char[] ch);

Constructors of String class:

Software Solutions

s = new String();
→ Creates an empty String object

s = new String(String literal);

s = new String(StringBuffer sb);

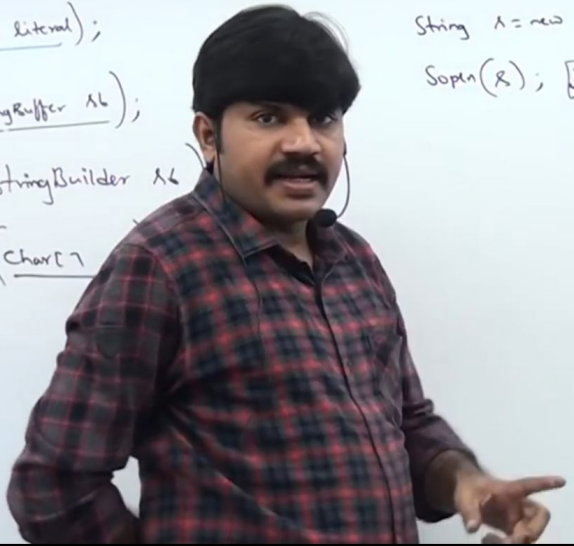
s = new String(StringBuilder sb);

s = new String(char[]

char[] ch = {'j', 'a', 'v', 'a'};

String s = new String(ch);

Sopln(s); Java ✓



Important methods of String class

5. public boolean isEmpty()
6. public int length()

String s = "durga";
Sopln(s.length()); 5
Sopln(s.length); CE

int[] x = {10,20,30,40};
Sopln(x.length); 4
Sopln(x.length()); CE

Important methods of String class

(7) public String replace(char old, char new);

(8) public String substring(int begin)
from begin index to end of String

(9) public String substring(int begin, int end);
from begin index to end-1 index

String s = "abcdefg";
Super(s.substring(3));
Super(s.substr
3t

DURGA
Software Solutions

Important methods of String class

10. public int indexOf(char ch);
11. public int lastIndexOf(char ch);
12. public String toLowerCase();
13. public String toUpperCase();

String s="d
Sopn(s
Sopn(s
String s=
Sopn(s

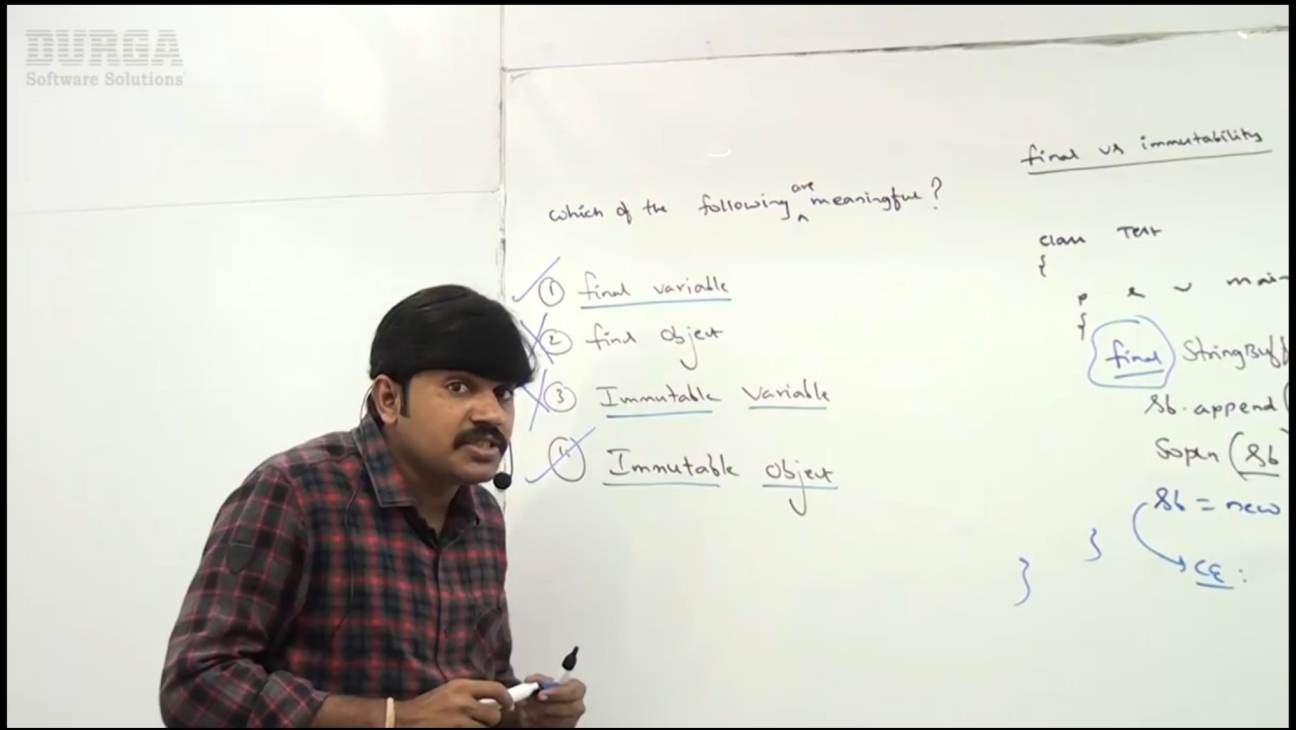
```java
import java.util.*;
class Test
{
    public static void main(String[] args)
    {
        Scanner sc= new Scanner(System.in);
        System.out.print("Enter Your City Name:");
        String name=sc.nextLine().toLowerCase().trim();
        if (name.equals("hyderabad"))
        {
            System.out.println("Hello Hyderabadi, Aadaab...");
        }
        else if (name.equals("chennai"))
        {
            System.out.println("Hello Madrasi, Vanakkam...");
        }
        else if (name.equals("bangalore"))
        {
            System.out.println("Hello Kannadiga, Namaskara...");
        }
        else
        {
            System.out.println("Please enter valid city name");
        }
    }
}
```

```java
final class Test
{
    private int i;
    Test(int i)
    {
        this.i = i;
    }
    public Test modify(int i)
    {
        if (this.i==i)
        {
            return this;
        }
        else
        {
            return new Test(i);
        }
    }
    public static void main(String[] args)
    {
        Test t1 = new Test(10);
        Test t2 = t1.modify(100);
        Test t3 = t1.modify(10);

        System.out.println(t1==t2);
        System.out.println(t1==t3);
    }
}
```

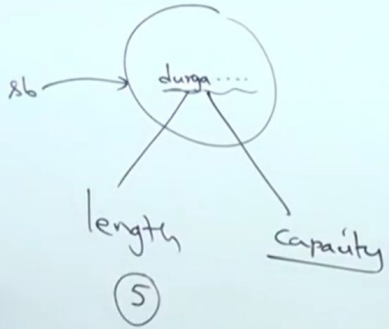
final vs immutability

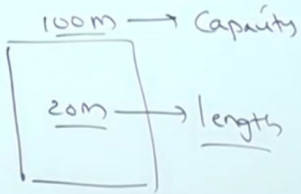
```
class Test
{
    p s v main(String[] arr)
    {
        final StringBuffer sb = new StringBuffer("durga");
        sb.append("software");
        Sop(sb);
        sb = new StringBuffer("ravi");
    }
    CE:
}
```

sb→


DURGA
Software Solutions
final vs immutability
which of the following are meaningful ?
1) final variable
2) final object
3) Immutable Variable
4) Immutable Object
class Test
final
StringBuffer
sb.append
Sopln (sb
sb = new
CE :

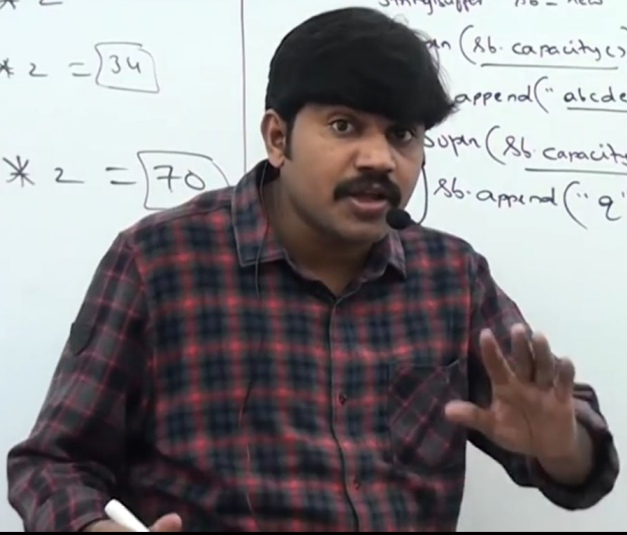
StringBuffer sb = new StringBuffer("durga");

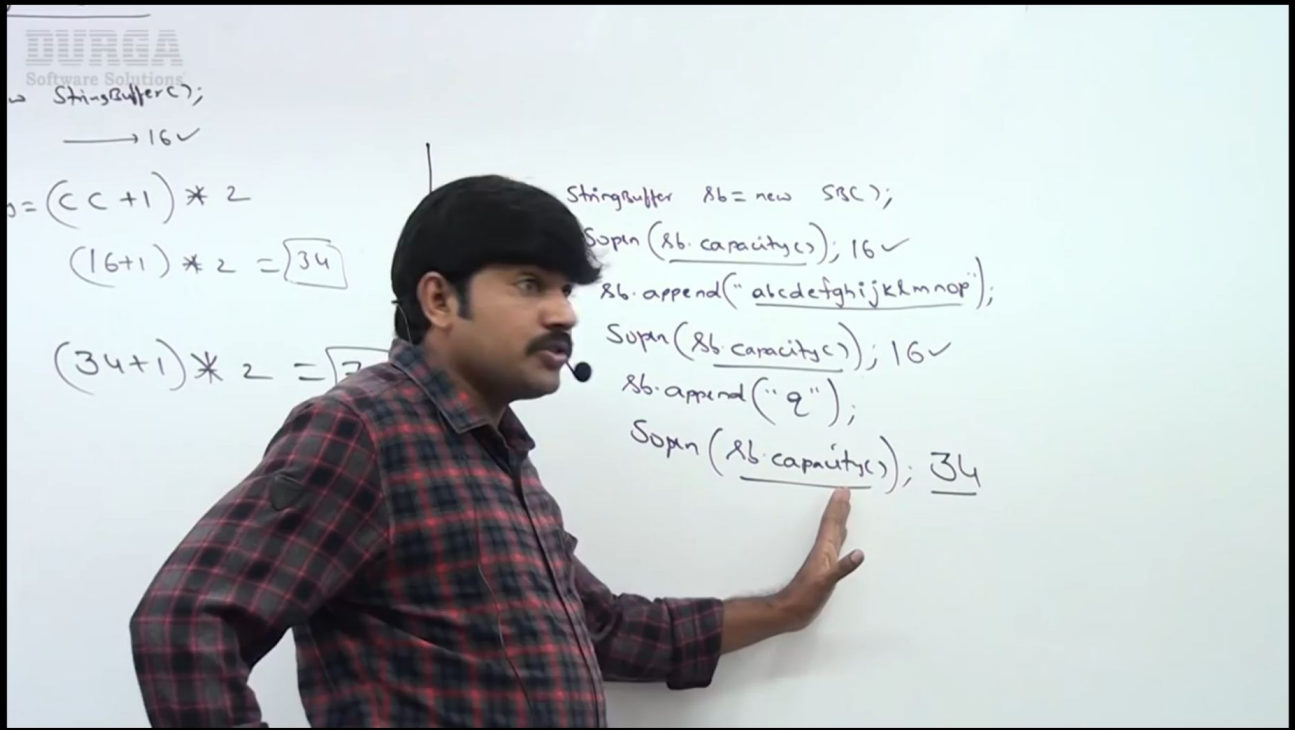


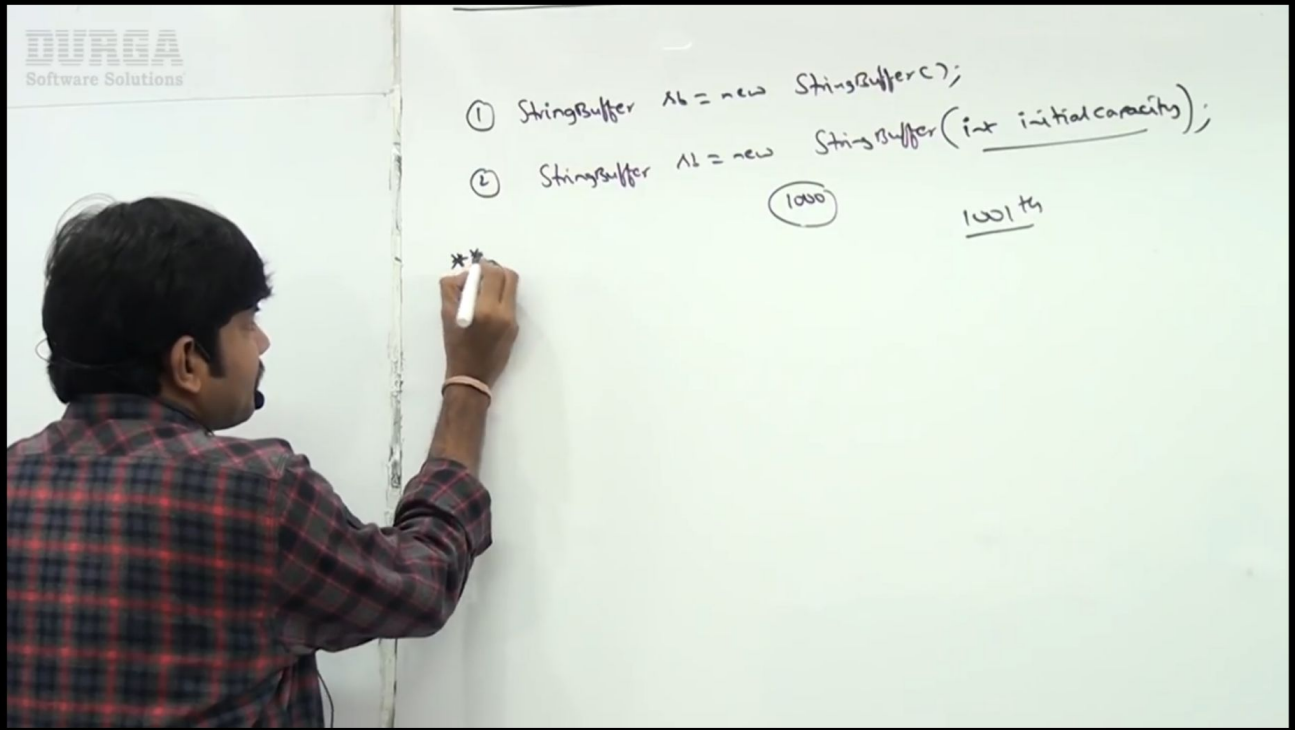


```
...w StringBuffer();
        ⟶ 16 ✓
y = (c c + 1) * 2
    (16+1) * 2 = 34
    (34+1) * 2 = 70
```

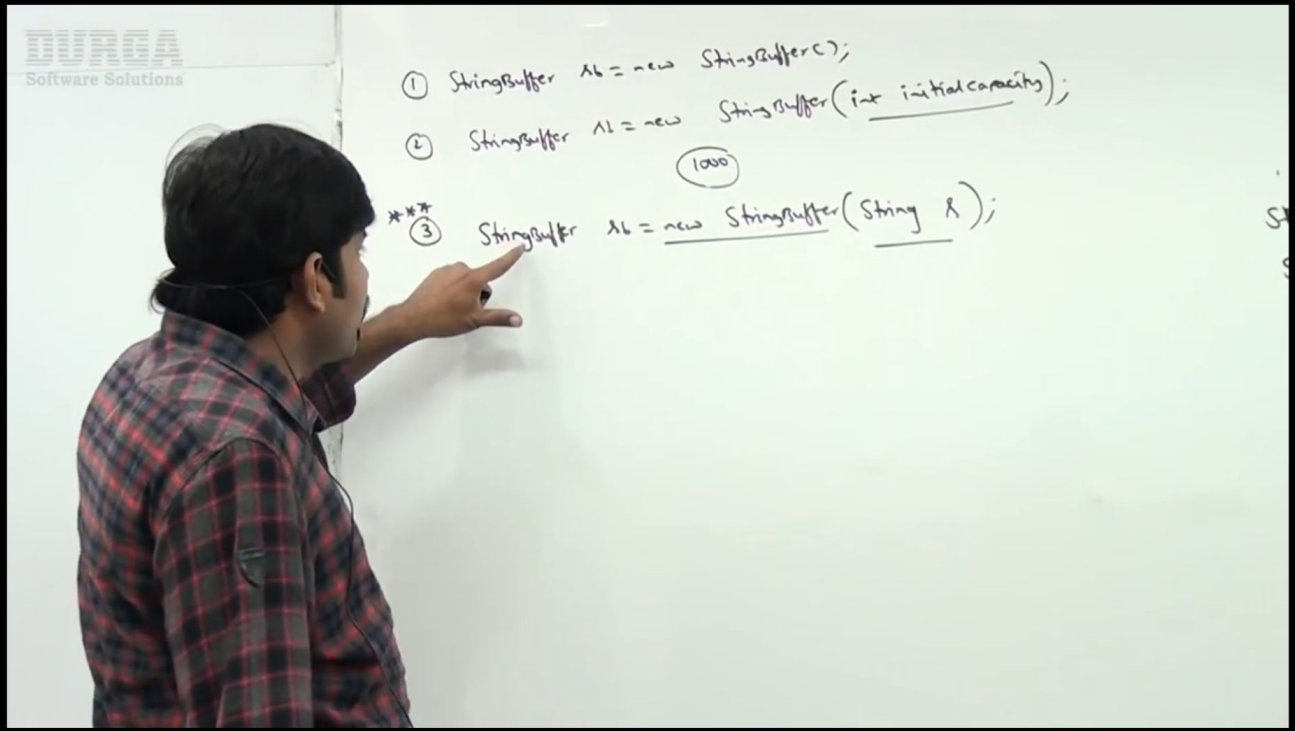
```
StringBuffer sb = new SBC);
...n (sb.capacity());  16 ✓
...append("abcdefghijklmnop");
...sb.capacity());  16 ✓
sb.append("q");
```

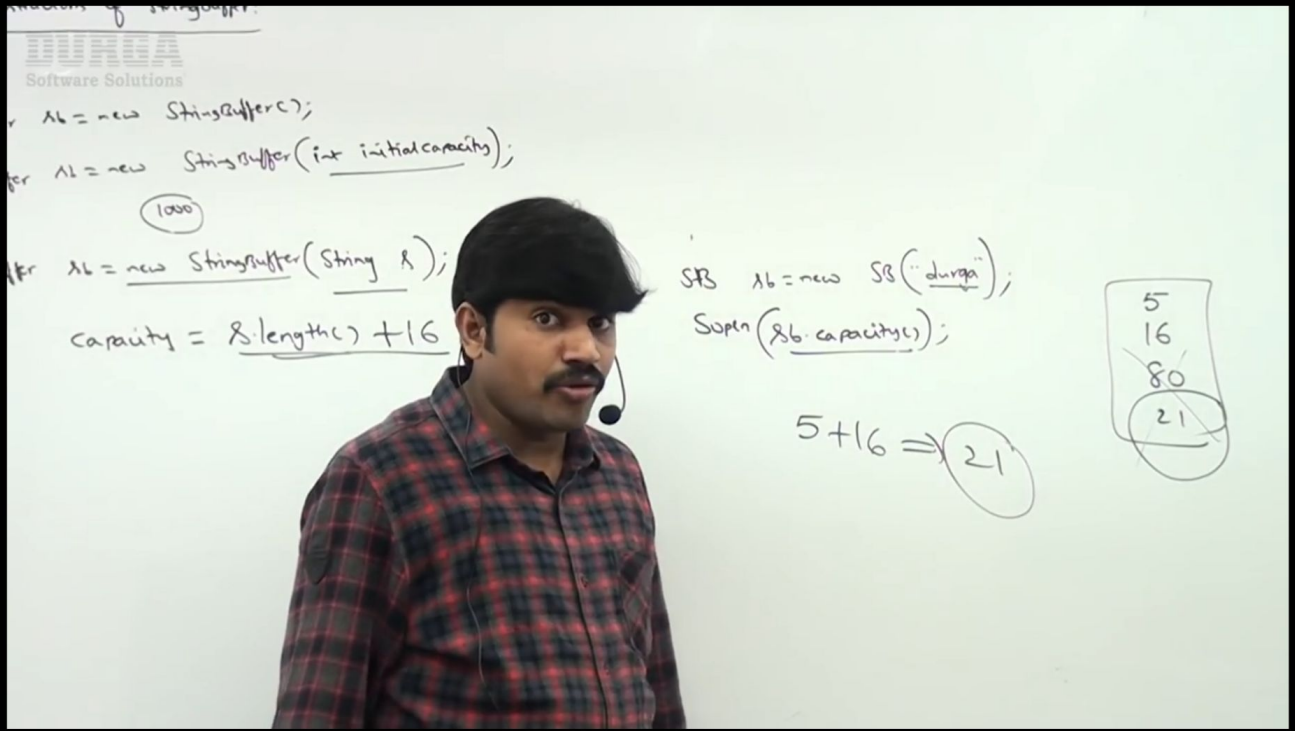



DURGA
Software Solutions
StringBuffer();
16
(cc+1) * 2
(16+1) * 2 = 34
(34+1) * 2 =
StringBuffer sb = new SBC();
Sopln(sb.capacity()); 16
sb.append("abcdefghijklmnop");
Sopln(sb.capacity()); 16
sb.append("q");
Sopln(sb.capacity()); 34


DURGA
Software Solutions
1) StringBuffer sb = new StringBuffer();
2) StringBuffer sb = new StringBuffer(int initialcapacity);
1000
1001th

```
1  public class Test
2  {
3      public static void main(String[] args)
4      {
5          StringBuffer sb = new StringBuffer(1000);
6          System.out.println(sb.capacity());
7      }
8  }
```


DURGA
Software Solutions
1) StringBuffer sb = new StringBuffer();
2) StringBuffer sb = new StringBuffer(int initialcapacity);
1000
***3) StringBuffer sb = new StringBuffer(String s);


DURGA
Software Solutions
StringBuffer sb = new StringBuffer();
StringBuffer sb = new StringBuffer(int initialCapacity);
1000
StringBuffer sb = new StringBuffer(String s);
capacity = s.length() + 16
SB sb = new SB("durga");
Sopln(sb.capacity());
5+16 => 21
5
16
80
21

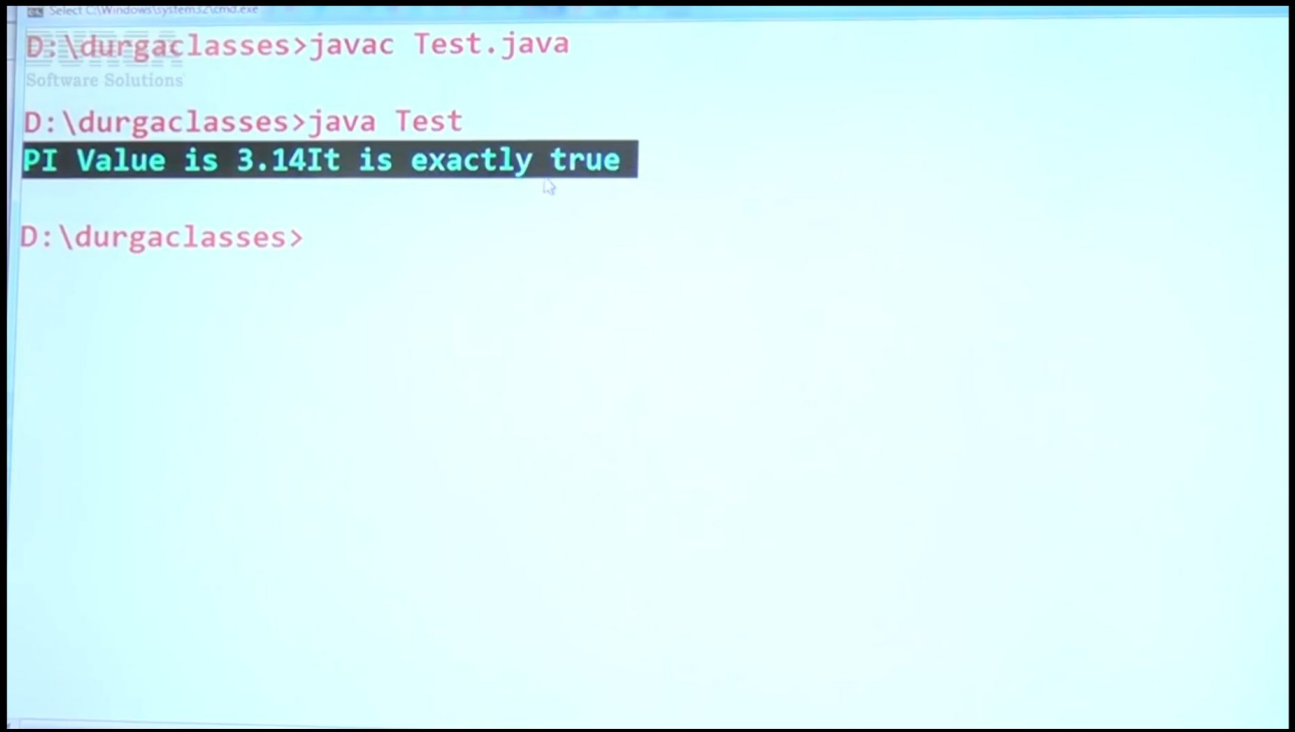
Important

④ public void setCharAt(int index, char newChar);

⑤ public SB append(String s)
append(byte b)
(int i)
(long l)
(float f)
:
:

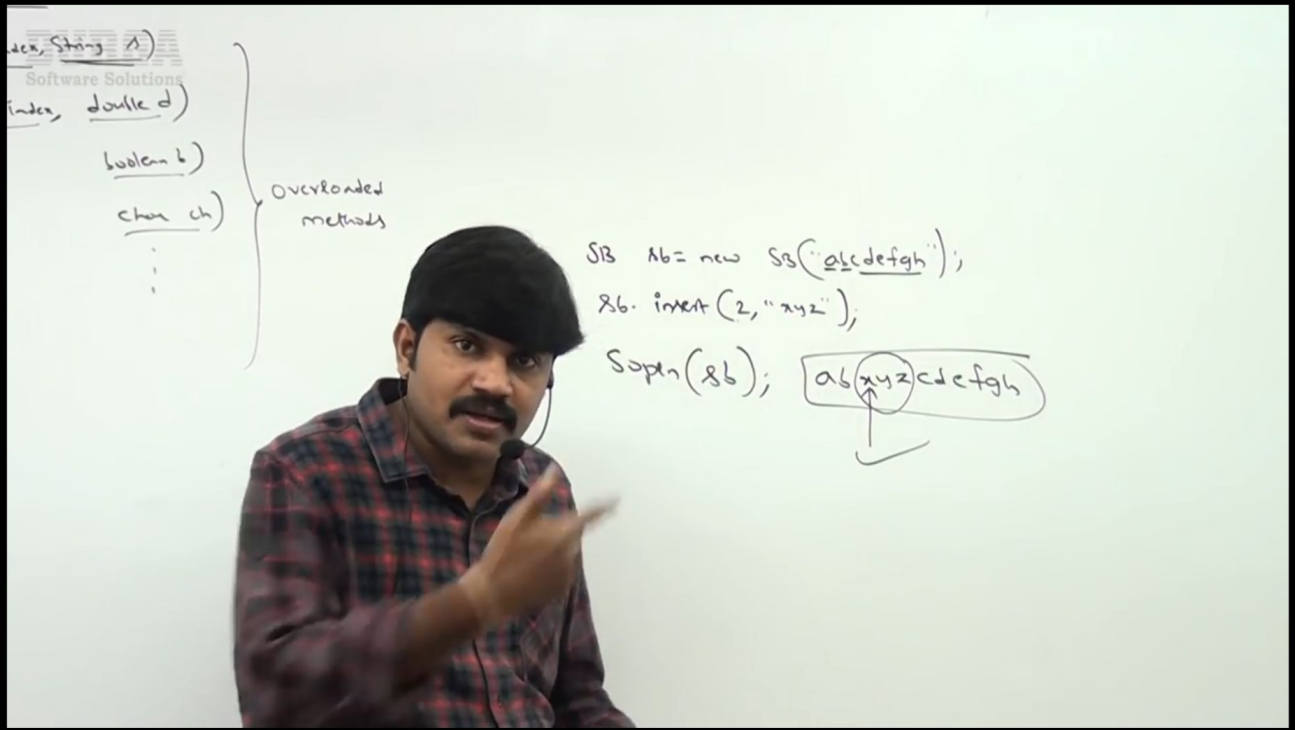
overloaded methods

```
1  public 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);
11     }
12 }
```


D:\durgaclasses>javac Test.java
Software Solutions
D:\durgaclasses>java Test
PI Value is 3.14It is exactly true
D:\durgaclasses>

Important methods of StringBuffer:

⑥ public SB insert(int index, String s)
___ (int index, double d)
boolean b)
char ch)
:
:
} Overloaded methods


Software Solutions
Overloaded methods
SB sb = new SB("abcdefgh");
sb.insert(2, "xyz");
Sopln(sb);
abxyzcdefgh

SB

delete(int begin, int end)

from begin index to

end-1 index

StringBuffer sb = new SB("abcdefgh");

sb.delete(2, 5);

sop(sb); abfgh ✓

ab c d e fgh

delete(int begin, int end)

from begin index to end-1 index

SB deleteCharAt(int

StringBuffer sb = new SB("abcdefgh");

sb.deleteCharAt(3);

Sopln(sb); abcefgh

delete(int begin, int end)

from begin index to end-1 index

deleteCharAt(int in

SB reverse()

StringBuffer sb = new SB("durga");

sb.reverse();

opln(sb);

agrud

```
SB sb = new SB("AishwaryaAbhi");
sb.setLength(8);
Supan(sb);
```

Aishwarya

of StringBuffer:

setLength(int length)

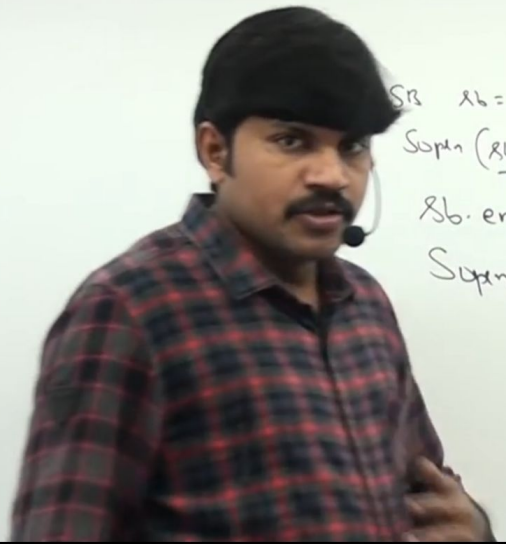
ensureCapacity(int capacity)

```
SB sb = new SB();
Sopln(sb.capacity()); 16
sb.ensureCapacity(1000);
Sopln(sb.capacity());
```

int length

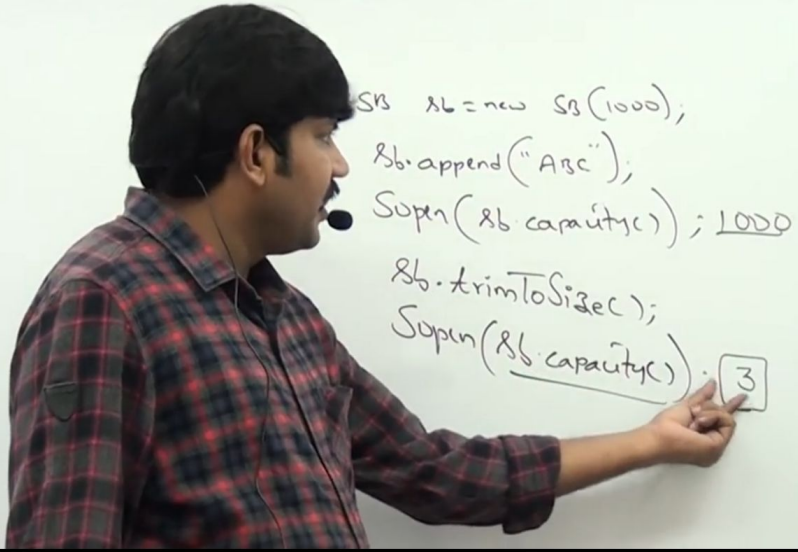
apacity(int capacity)

```
SB sb = new SB();
Sopln(sb.capacity()); 16
sb.ensureCapacity(1000);
Sopln(sb.capacity()); 1000
```



setLength(int length)

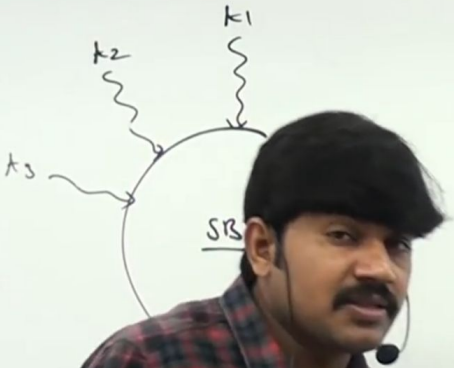
void ensureCapacity(int capacity)



```
SB sb = new SB(1000);
sb.append("ABC");
Sopln(sb.capacity()); 1000
sb.trimToSize();
Sopln(sb.capacity()); 3
```

StringBuffer	StringBuilder
Every Method Present In Stringbuffer Is Synchronized.	No Method Present In Stringbuilder Is Synchronized.
At A Time Only One Thread Is Allow To Operate On Stringbuffer Object And Hence It Is Thread Safe.	At A Time Multiple Thread Are Allowed To Operate On Stringbuilder Object And Hence It Is Not Thread Safe.
Threads Are Required To Wait To Operate On Stringbuffer Object And Hence Relatively Performance Is Slow.	Threads Are Not Required To Wait To Operate On Stringbuilder Object And Hence Relatively Performance Is High.
Introduced In 1.0 Version.	Introduced In 1.5 Version.

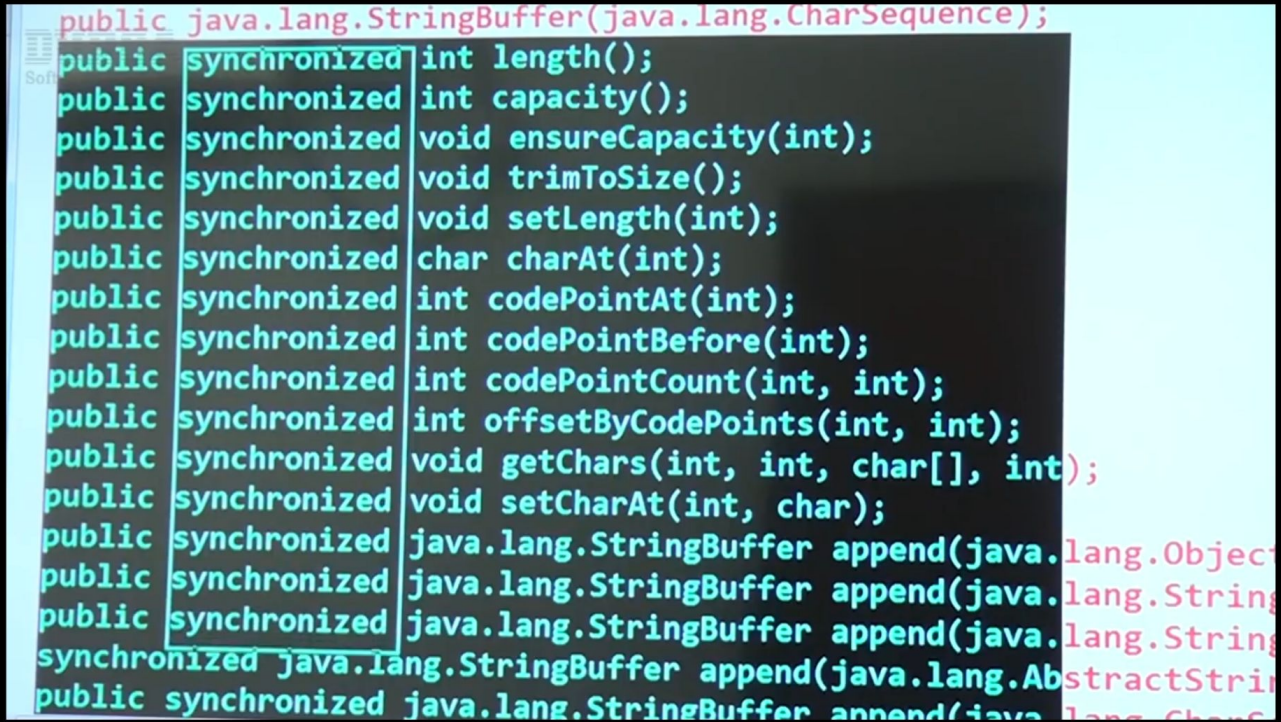
StringBuilder



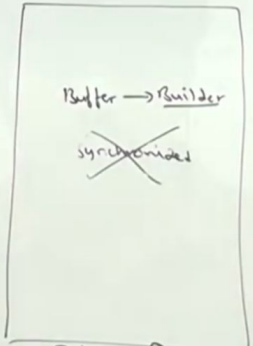
1.5 V

StringBuilder

```
D:\durgaclasses>javap java.lang.StringBuffer
```


public java.lang.StringBuffer(java.lang.CharSequence);
public synchronized int length();
public synchronized int capacity();
public synchronized void ensureCapacity(int);
public synchronized void trimToSize();
public synchronized void setLength(int);
public synchronized char charAt(int);
public synchronized int codePointAt(int);
public synchronized int codePointBefore(int);
public synchronized int codePointCount(int, int);
public synchronized int offsetByCodePoints(int, int);
public synchronized void getChars(int, int, char[], int);
public synchronized void setCharAt(int, char);
public synchronized java.lang.StringBuffer append(java.
public synchronized java.lang.StringBuffer append(java.
public synchronized java.lang.StringBuffer append(java.
synchronized java.lang.StringBuffer append(java.lang.Ab
public synchronized java.lang.StringBuffer

StringBuilder



1.5 V

StringBuilder

DURGA
Software Solutions

Method chaining

sb.m1()

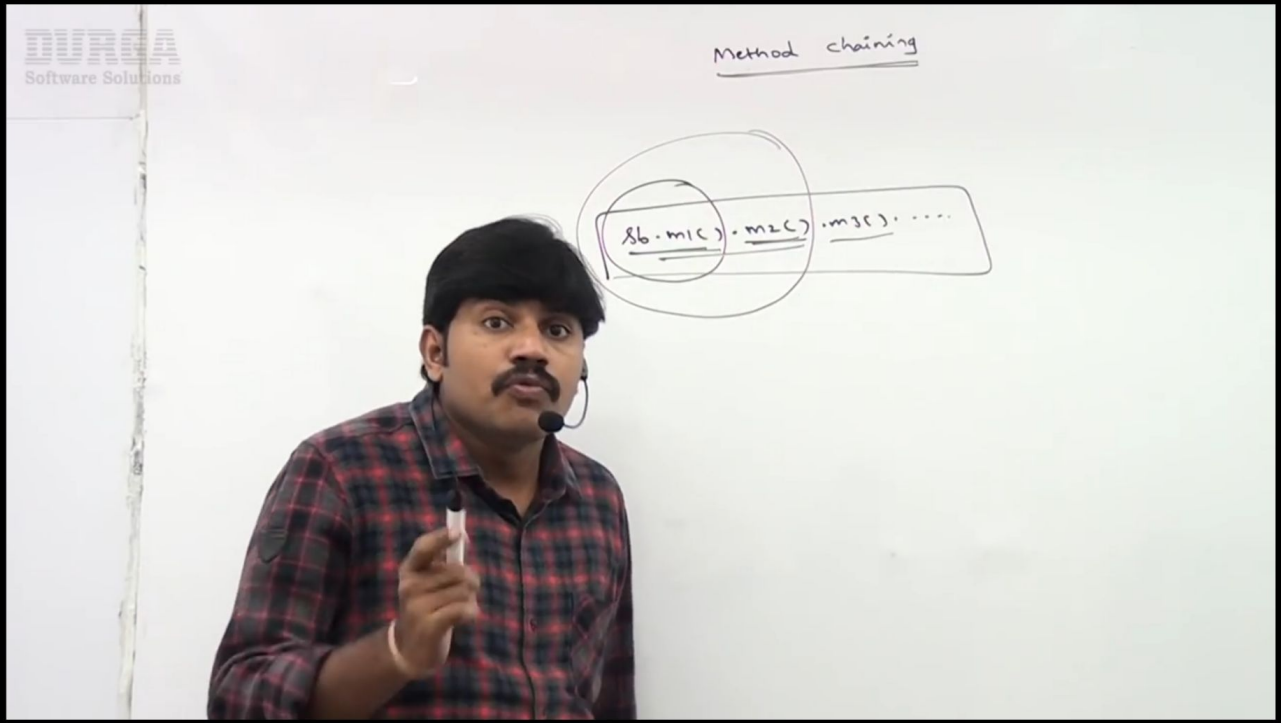
StringBuilder

SB append()

SB reverse()

SB insert()

SB delete()



Method chaining

sb.m1().m2().m3()....

eg:

```
StringBuilder sb = new StringBuilder();
sb.append("durga").append("solutions").reverse().insert(
System.out.println(sb);
```