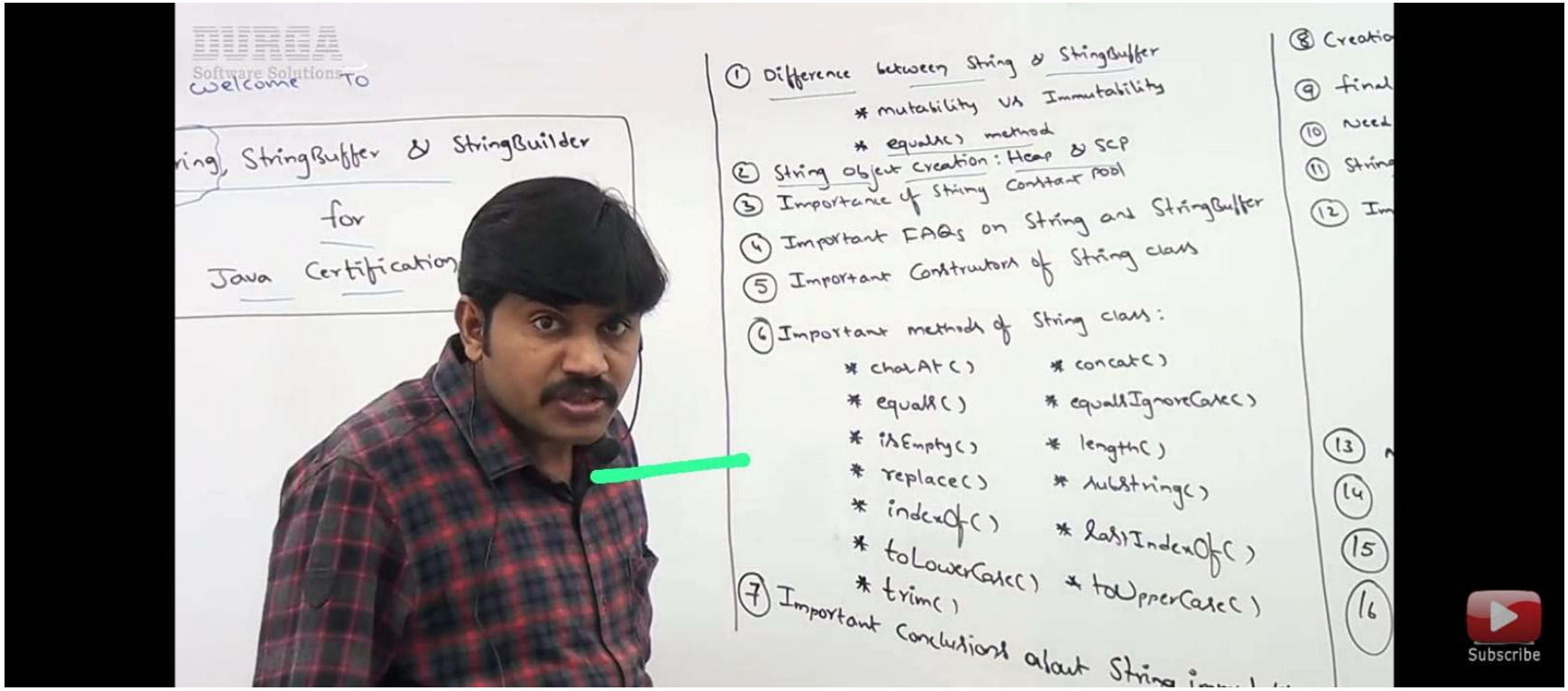
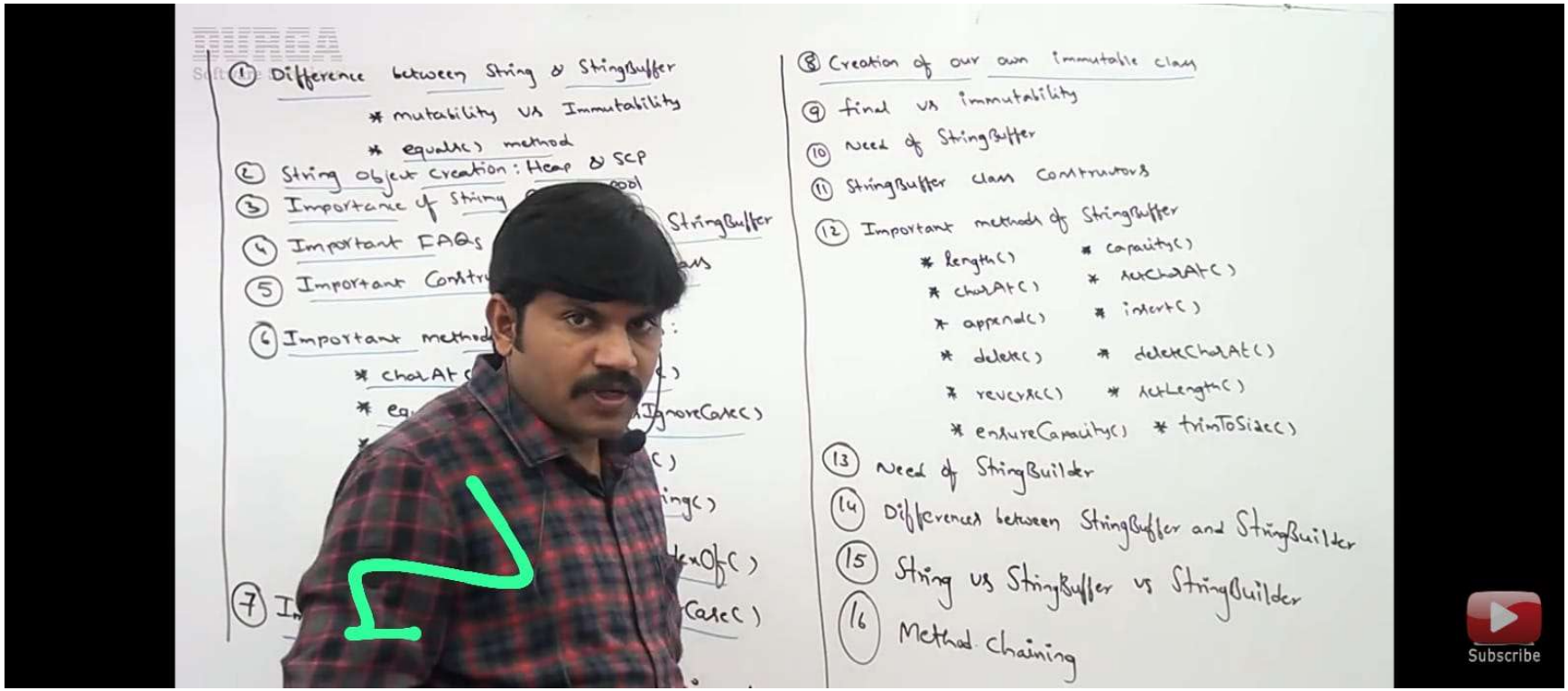
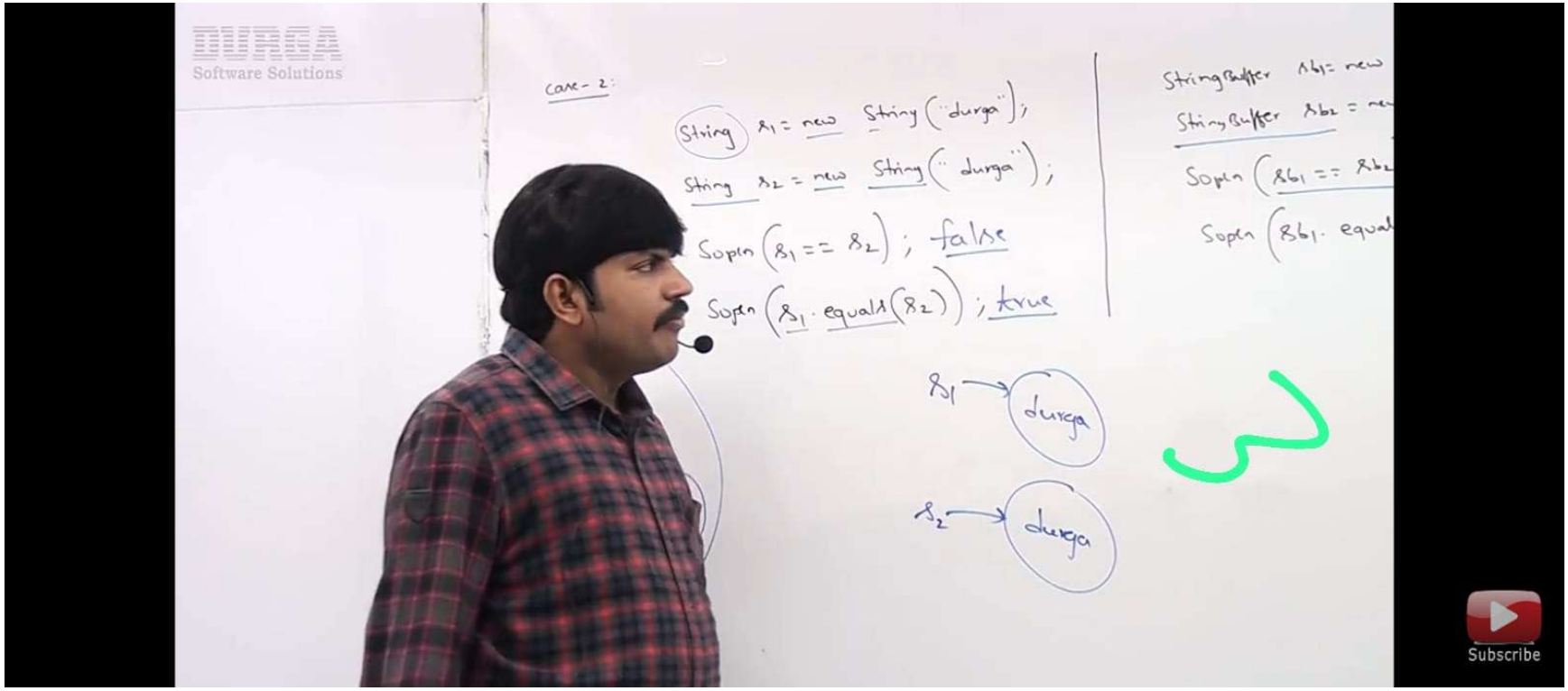
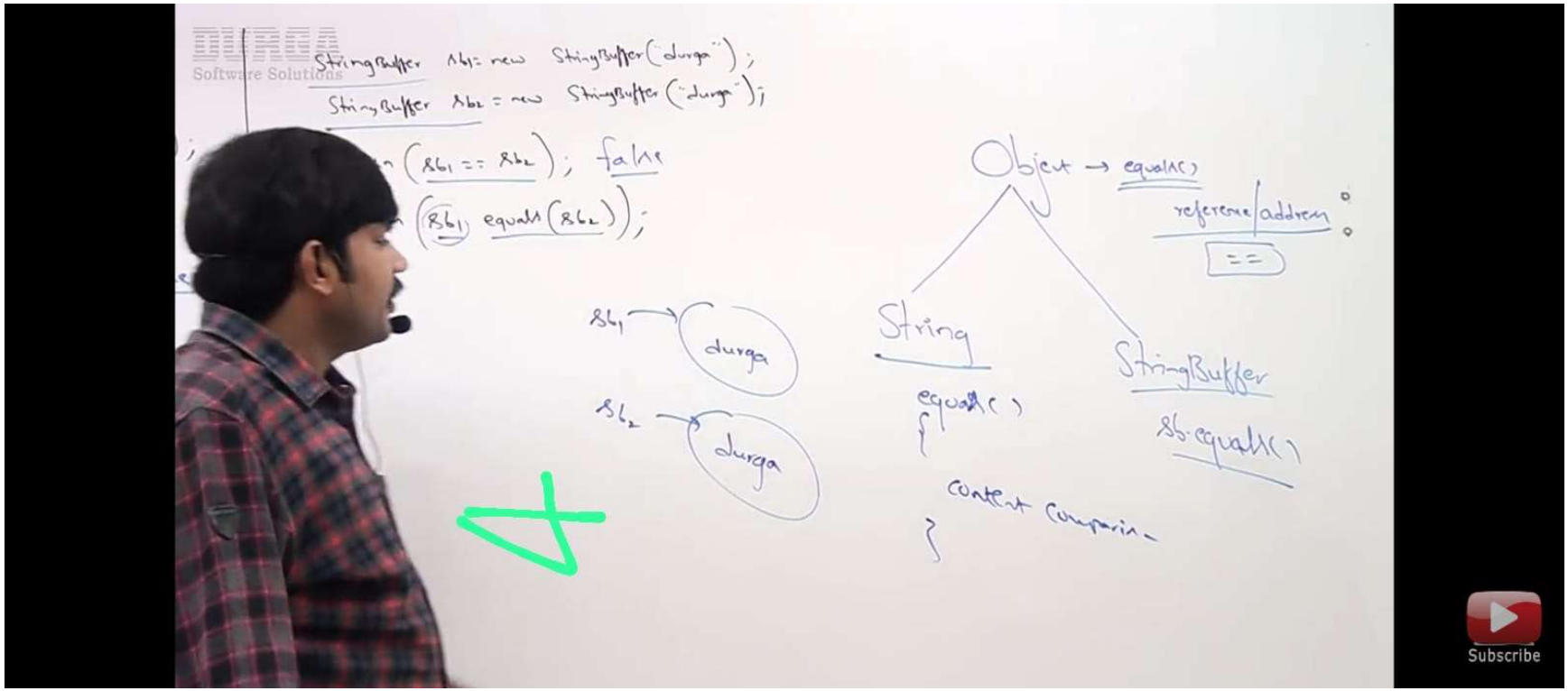
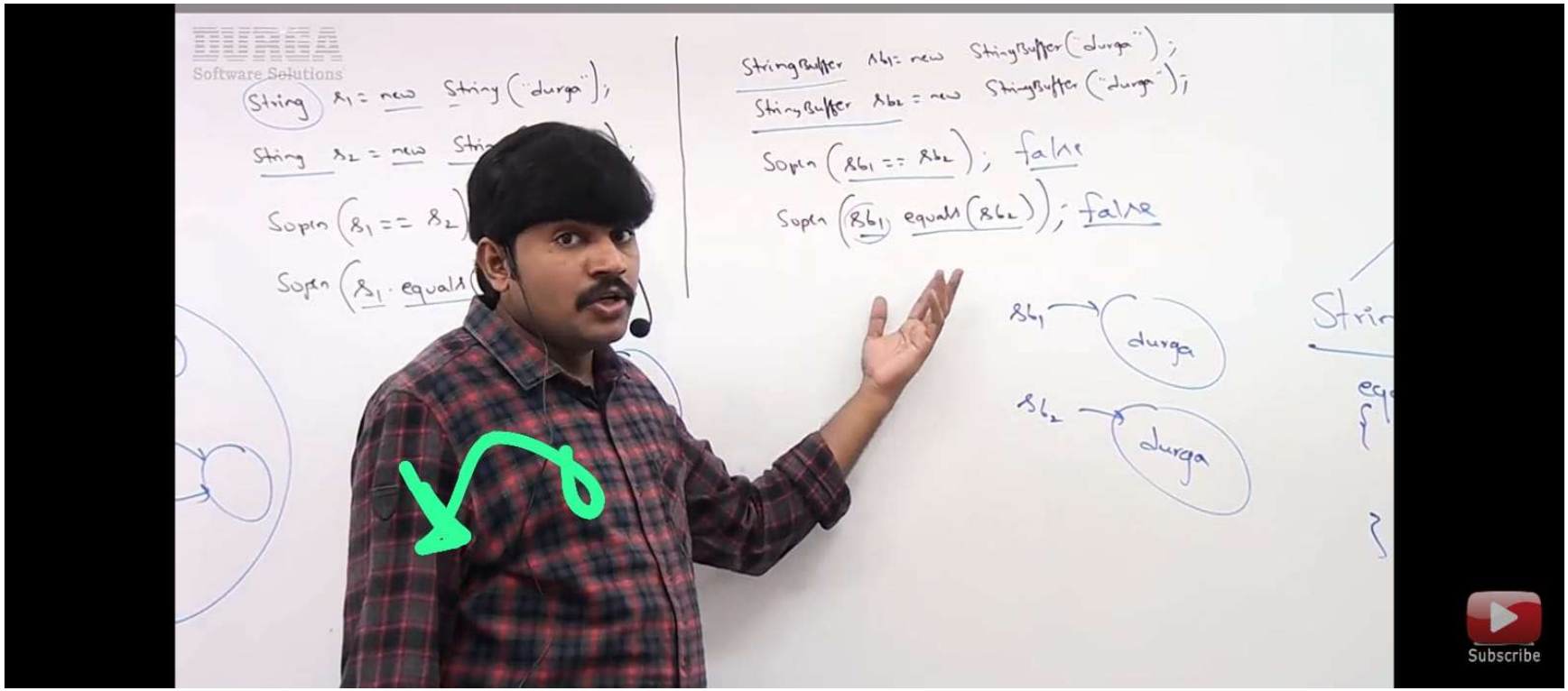
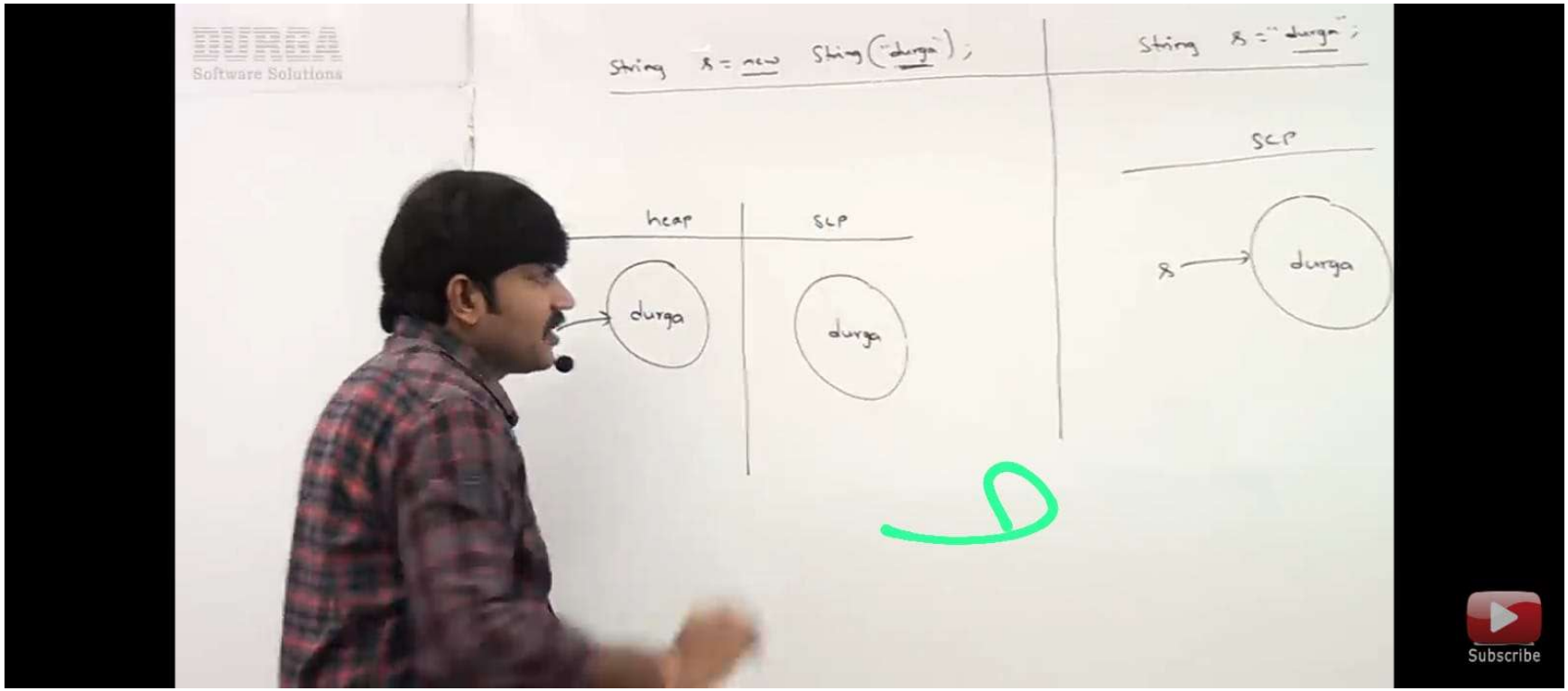

BURRA
Software Solutions
Welcome To
ring, StringBuffer & StringBuilder
for
Java Certification
1) Difference between String & StringBuffer
* mutability VS Immutability
* equals() method
2) String object creation: Heap & SCP
3) Importance of String Constant pool
4) Important FAQs on String and StringBuffer
5) Important Constructors of String class
6) Important methods of String class:
* charAt()
* concat()
* equals()
* equalsIgnoreCase()
* isEmpty()
* length()
* replace()
* substring()
* indexOf()
* lastIndexOf()
* toLowerCase()
* toUpperCase()
* trim()
7) Important Conclusions about String
8) Creatio
9) final
10) Need
11) String
12) Im
13)
14)
15)
16)
Subscribe

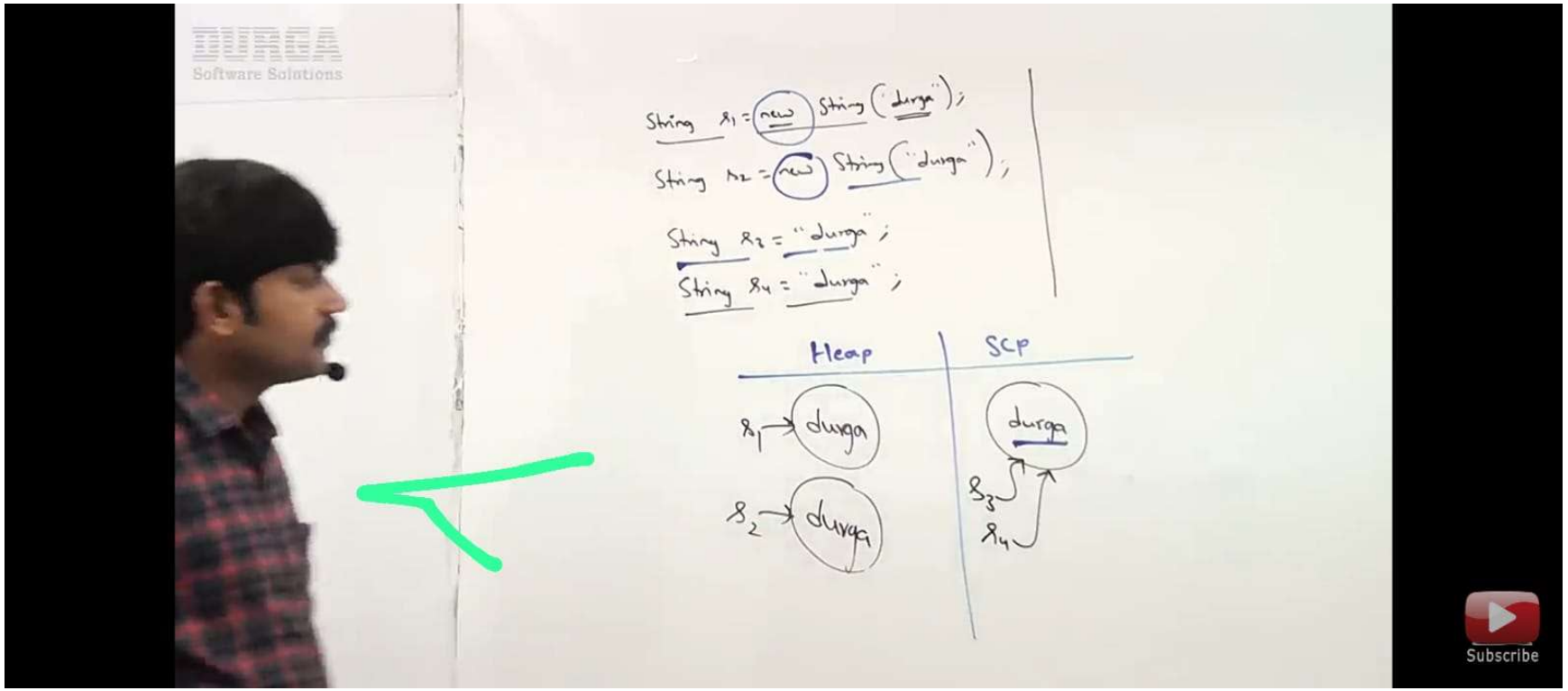

1. Difference between String & StringBuffer
* mutability VS Immutability
* equals() method
2. String object creation: Heap & SCP
3. Importance of String
4. Important FAQs
5. Important Constr
6. Important method
* charAt(
* eq
7. I
StringBuffer
IgnoreCase()
ing()
rmOf()
Case()
8. Creation of our own immutable class
9. final VS immutability
10. Need of StringBuffer
11. StringBuffer class Constructors
12. Important methods of StringBuffer
* length()
* charAt()
* append()
* delete()
* reverse()
* ensureCapacity()
* capacity()
* setCharAt()
* insert()
* deleteCharAt()
* setLength()
* trimToSize()
13. Need of StringBuilder
14. Differences between StringBuffer and StringBuilder
15. String vs StringBuffer vs StringBuilder
16. Method Chaining
Subscribe


DURGA
Software Solutions
case-2:
String s1 = new String("durga");
String s2 = new String("durga");
Sopln(s1 == s2); false
Sopln(s1.equals(s2)); true
StringBuffer sb1= new
StringBuffer sb2 = ne
Sopln(sb1 ==
Sopln(sb1. equal
s1 → durga
s2 → durga
Subscribe

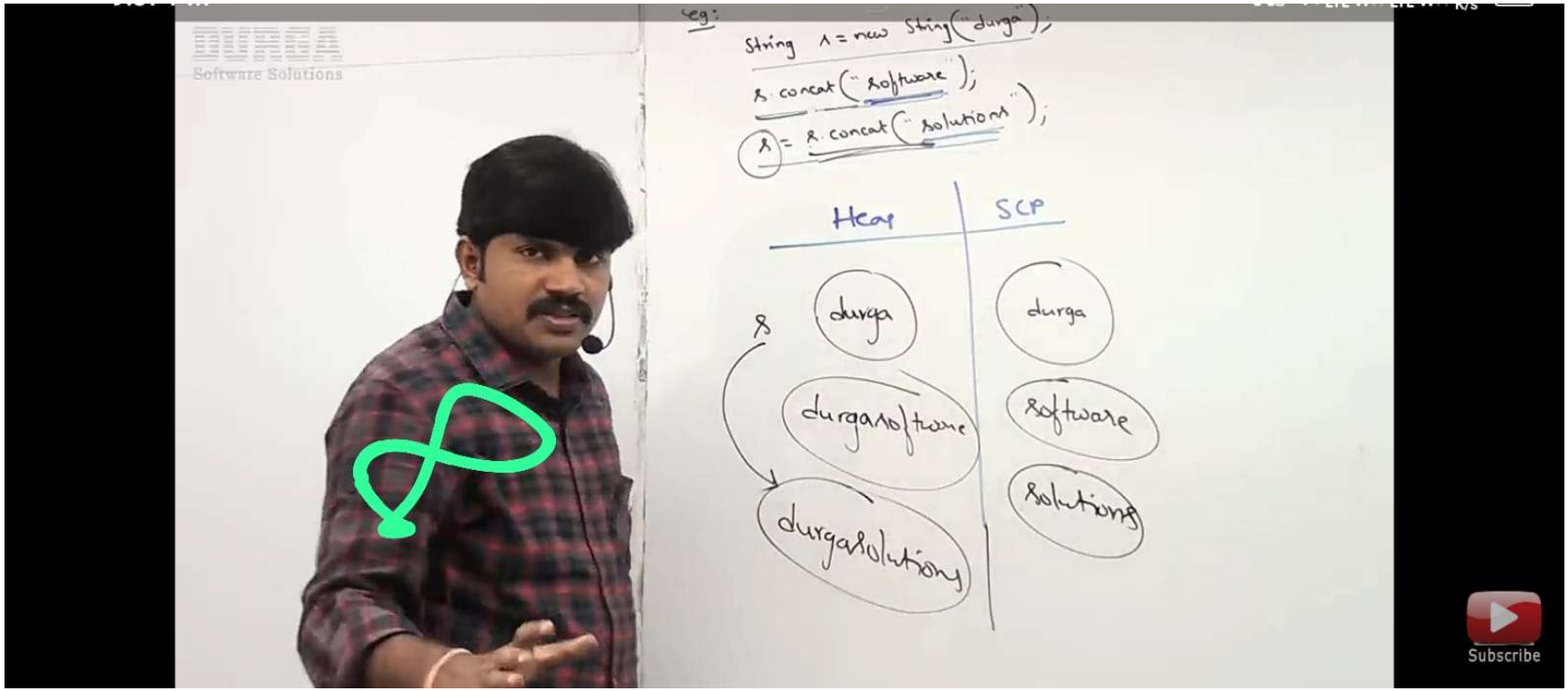

StringBuffer sb1 = new StringBuffer("durga");
StringBuffer sb2 = new StringBuffer("durga");
(sb1 == sb2); false
(sb1.equals(sb2));
Object → equals()
reference/address
==
sb1 → durga
sb2 → durga
String
equals()
{
content comparison
}
StringBuffer
sb.equals()
Subscribe

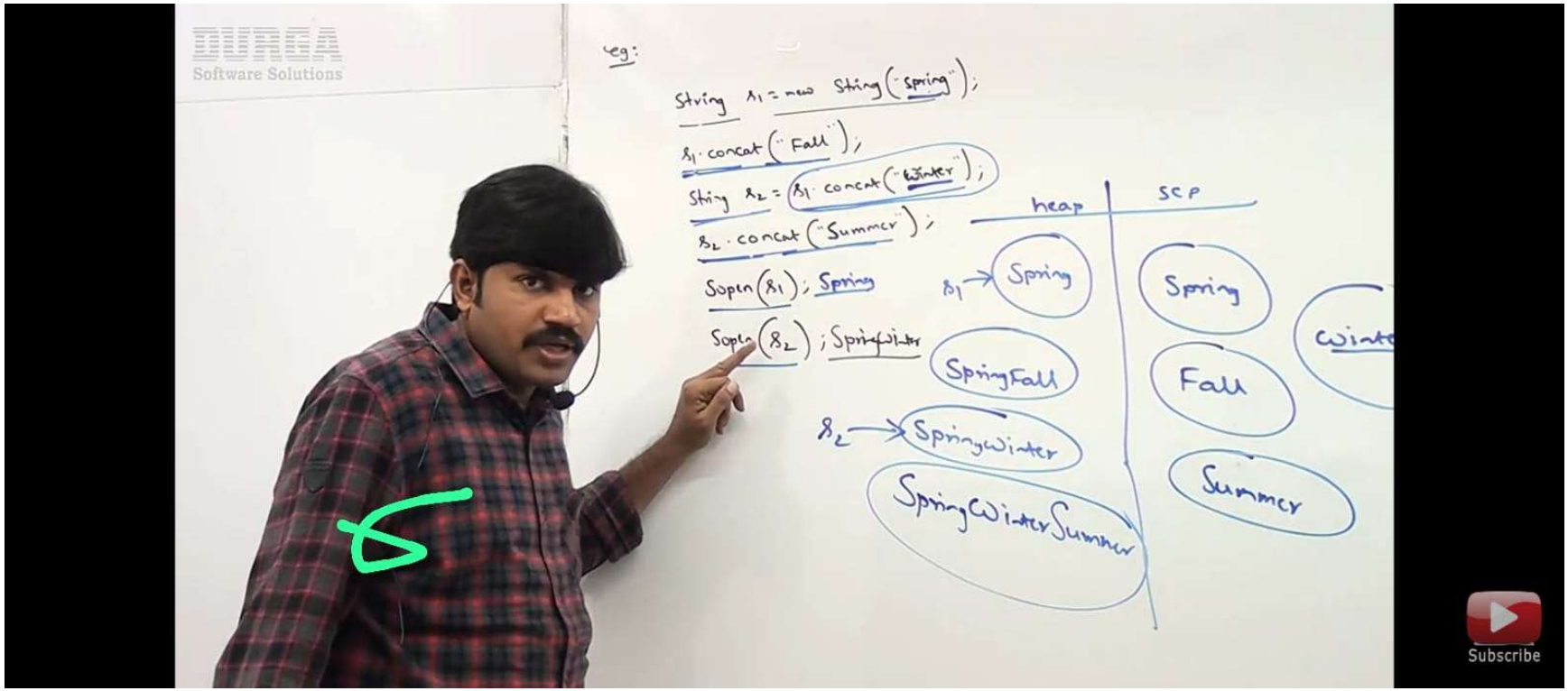

DURGA
Software Solutions
String s1 = new String("durga");
String s2 = new Stri
Sopln (s1 == s2)
Sopln (s1.equals(
StringBuffer sb1 = new StringBuffer("durga");
StringBuffer sb2 = new StringBuffer("durga");
Sopln (sb1 == sb2); false
Sopln (sb1.equals(sb2)); false
sb1
durga
sb2
durga
Strin
Subscribe

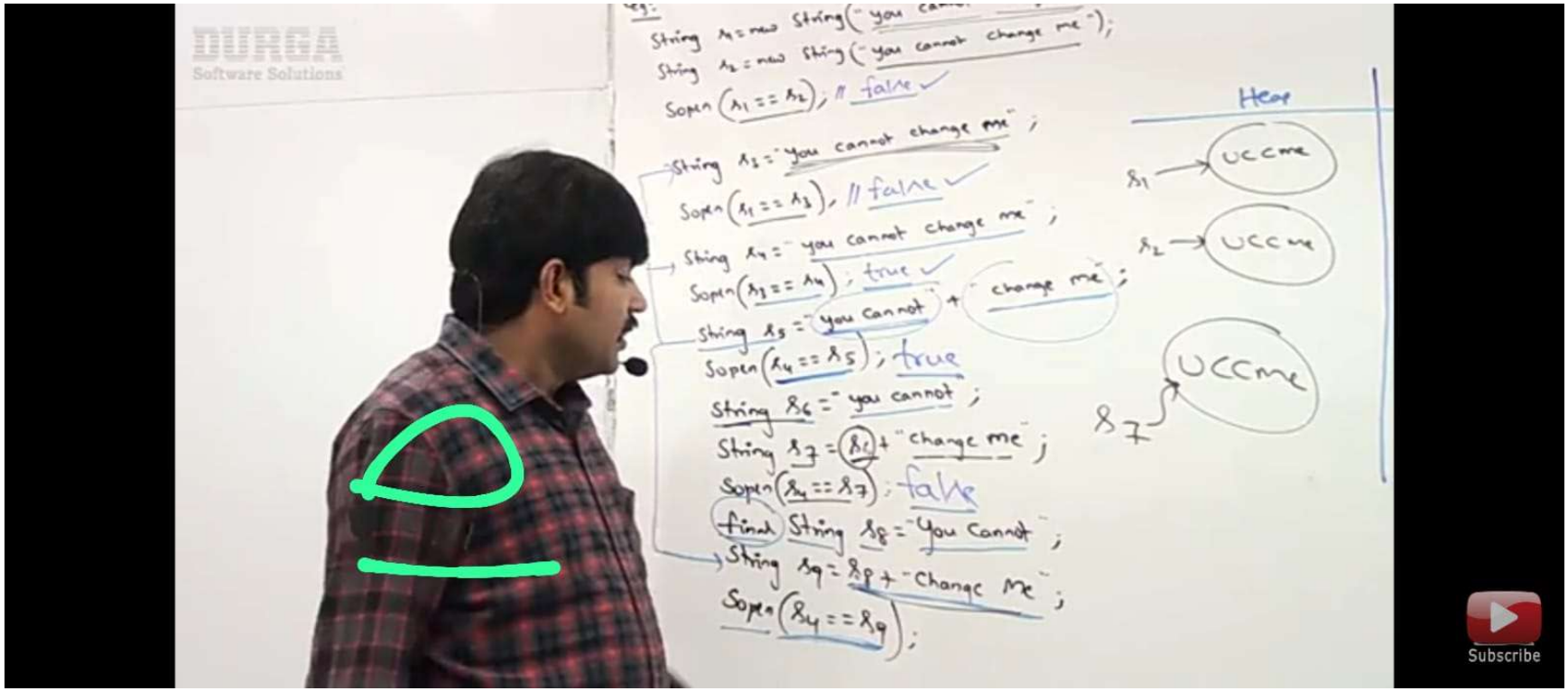

DURGA
Software Solutions
String s = new String("durga");
String s = "durga";
heap
SCP
durga
durga
SCP
s
durga
Subscribe

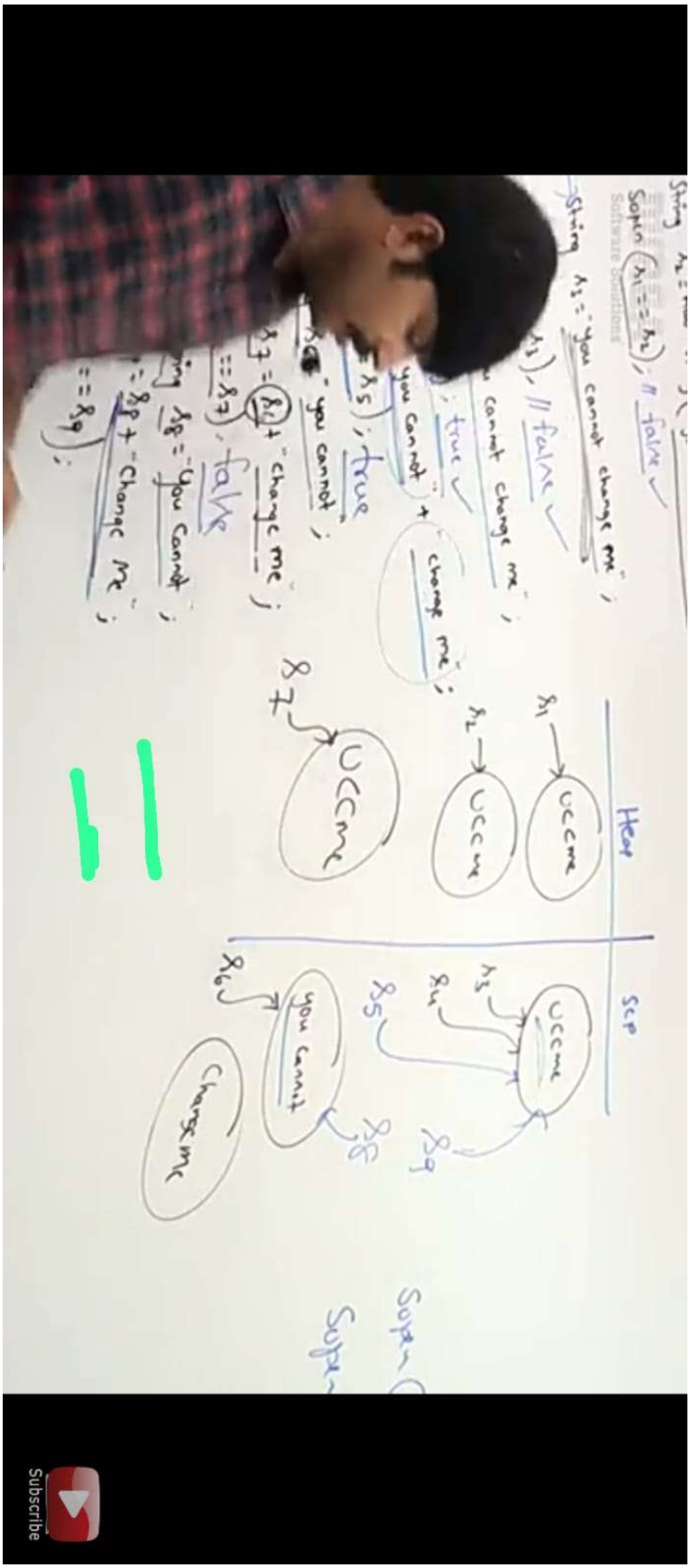

DURGA
Software Solutions
String s1 = new String("durga");
String s2 = new String("durga");
String s3 = "durga";
String s4 = "durga";
Heap
SCP
s1
durga
s2
durga
durga
s3
s4
Subscribe

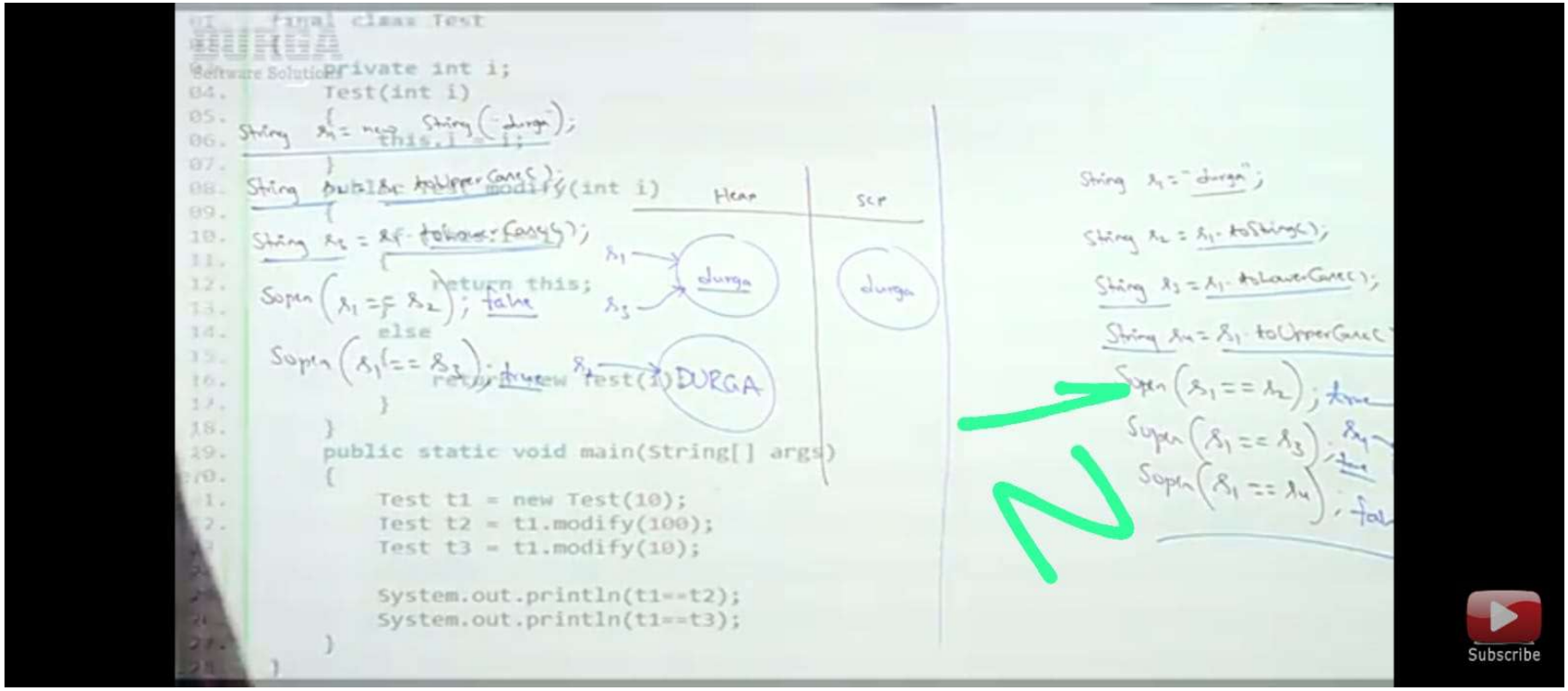
∞

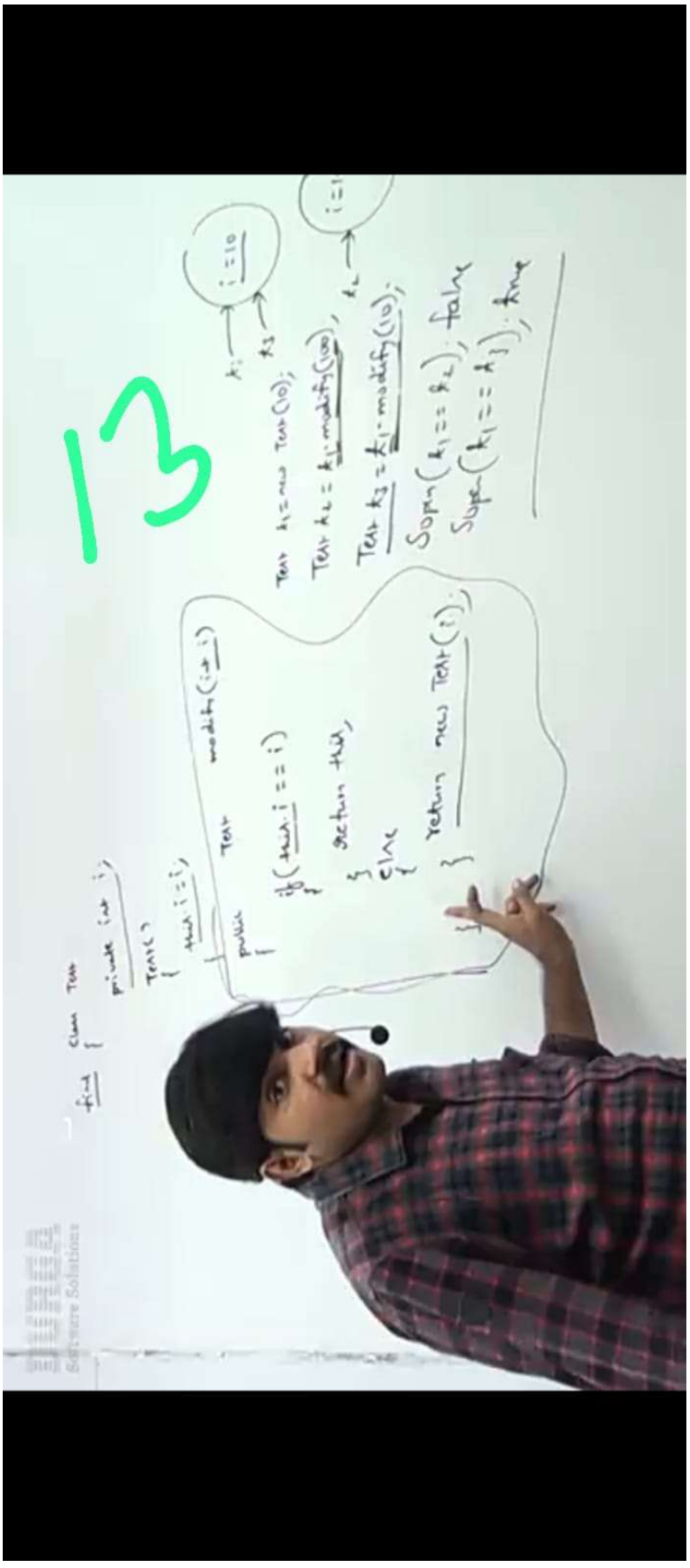

eg:
String s = new String("durga");
s.concat("software");
s = s.concat("solutions");
Heap
SCP
durga
durgasoftware
durgasolutions
durga
software
solutions
s
DURGA
Software Solutions
Subscribe

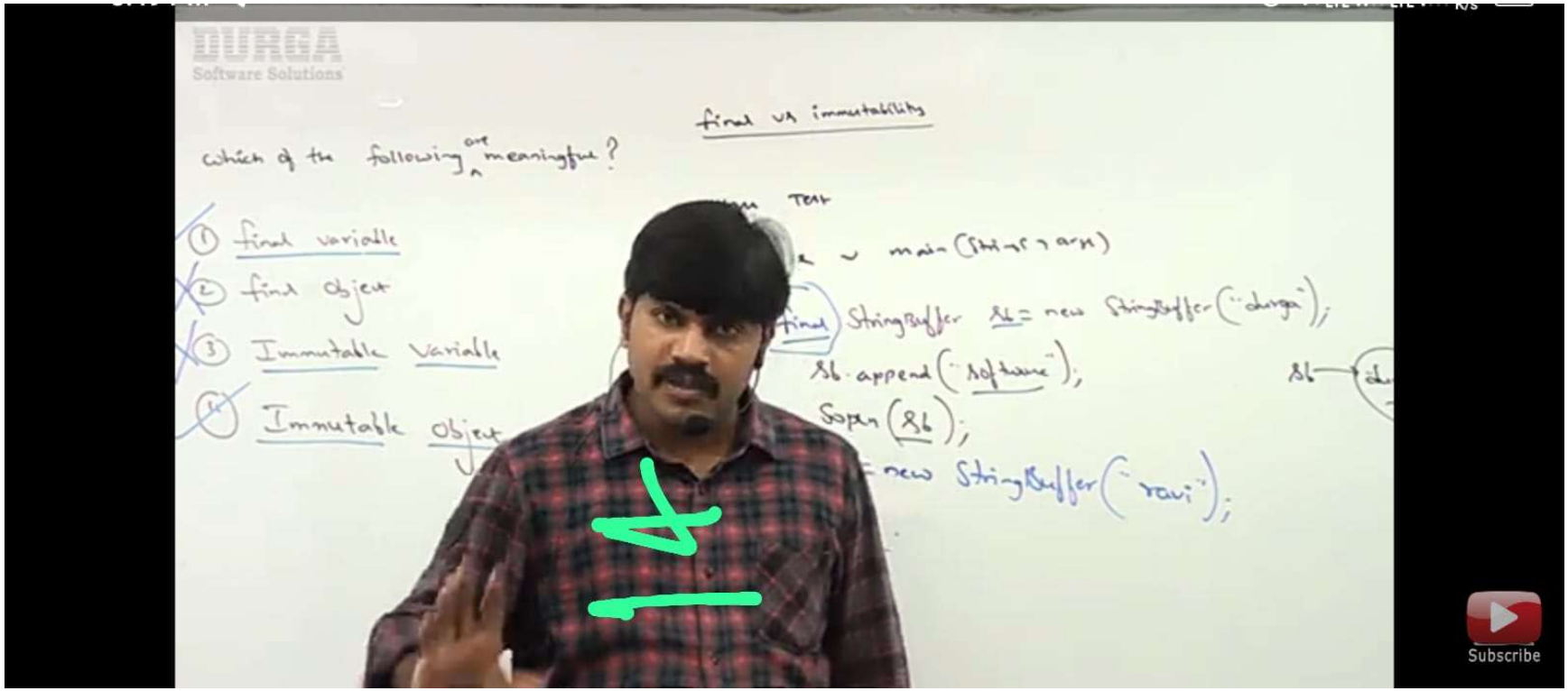

BURGA
Software Solutions
eg:
String s1 = new String("Spring");
s1.concat("Fall");
String s2 = s1.concat("Winter");
s2.concat("Summer");
Sopln(s1); Spring
Sopln(s2); Springwinter
heap
scp
s1
Spring
SpringFall
s2
Springwinter
SpringWinterSummer
Spring
Fall
Summer
Subscribe

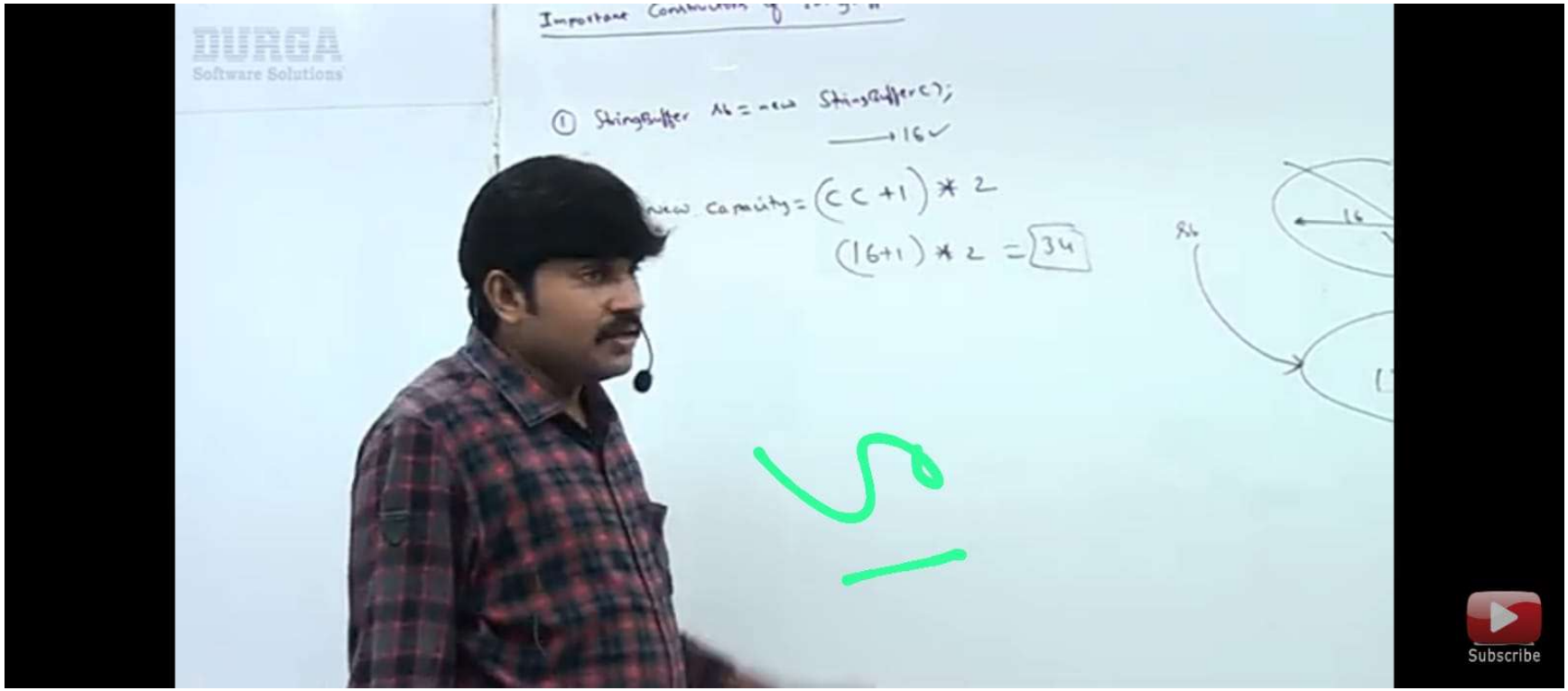

DURGA
Software Solutions
String s1 = new String("you cannot change me");
String s2 = new String("you cannot change me");
Sopln(s1 == s2); // false
String s3 = "you cannot change me";
Sopln(s1 == s3); // false
String s4 = "you cannot change me";
Sopln(s3 == s4); // true
String s5 = "you cannot" + "change me";
Sopln(s4 == s5); true
String s6 = "you cannot";
String s7 = s6 + "change me";
Sopln(s5 == s7); false
final String s8 = "You Cannot";
String s9 = s8 + "Change Me";
Sopln(s4 == s9);
Heap
s1
UCCme
s2
UCCme
s7
UCCme
Subscribe

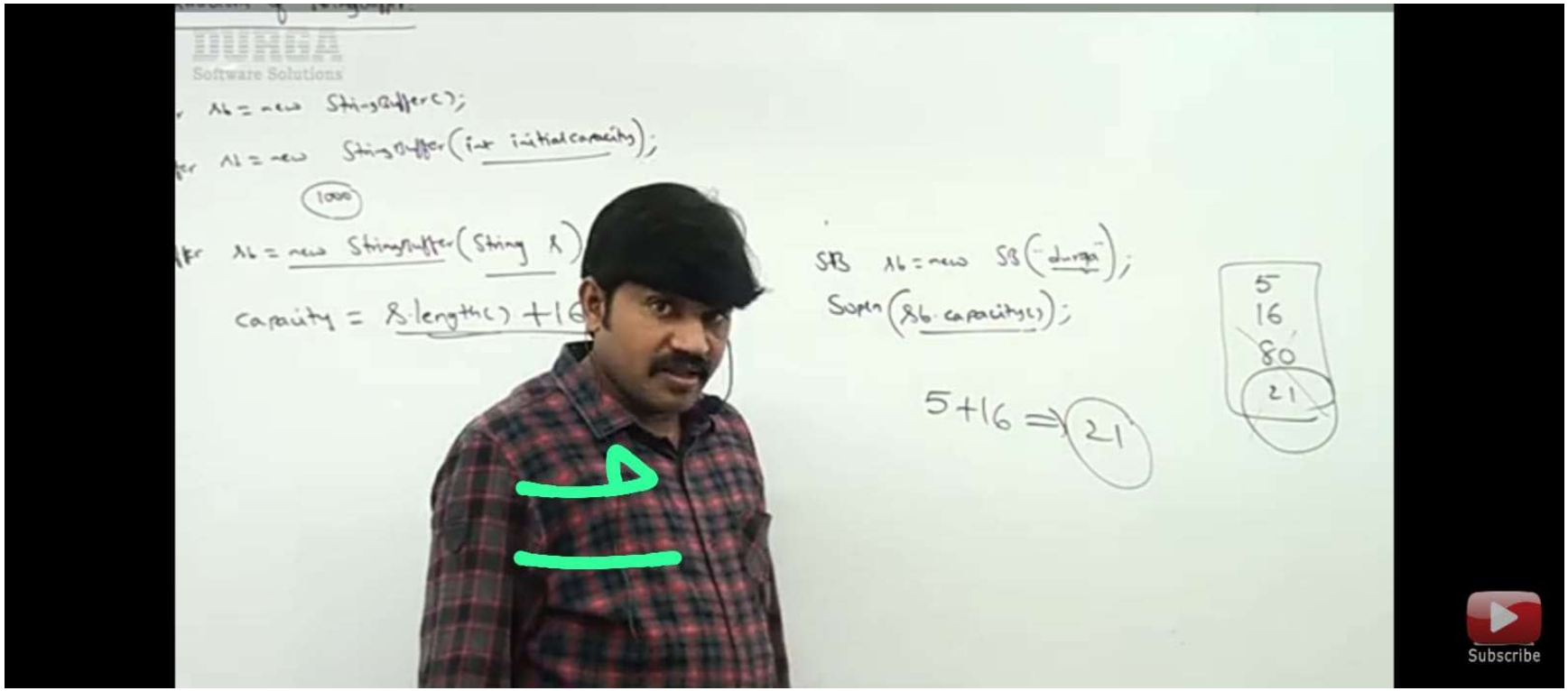

Soysa (x1 == x2) // false
String x2 = "you cannot change me";
x1) // false
cannot change me
true
you cannot change me;
x5); true
you cannot
x7 = (x1) + "change me
== x3) // false
ing x8 = "you cannot
+ "Change me
== x9);
Heap
Stack
UCCme
UCCme
UCCme
you cannot
Change me
x1
x2
x3
x4
x5
x6
x7
x8
x9
Subscribe

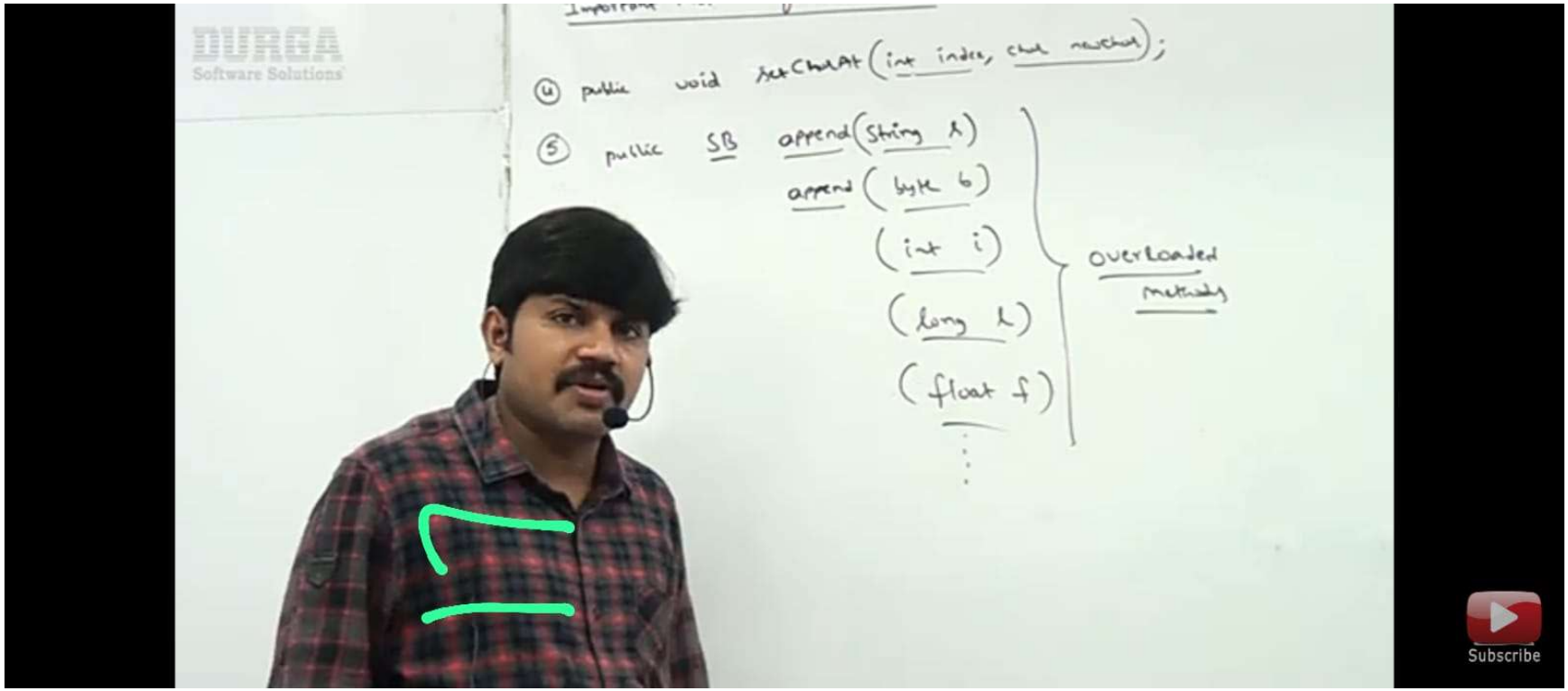

final class Test
{
private int i;
Test(int i)
{
this.i = i;
}
public Test modify(int i)
{
if(i==this.i)
return this;
else
return new Test(i);
}
public static void main(String[] args)
{
Test t1 = new Test(10);
Test t2 = t1.modify(100);
Test t3 = t1.modify(10);
System.out.println(t1==t2);
System.out.println(t1==t3);
}
}
Heap
Scp
durga
DURGA
String s1 = "durga";
String s2 = s1.toString();
String s3 = s1.toLowerCase();
String s4 = s1.toUpperCase();
Subscribe

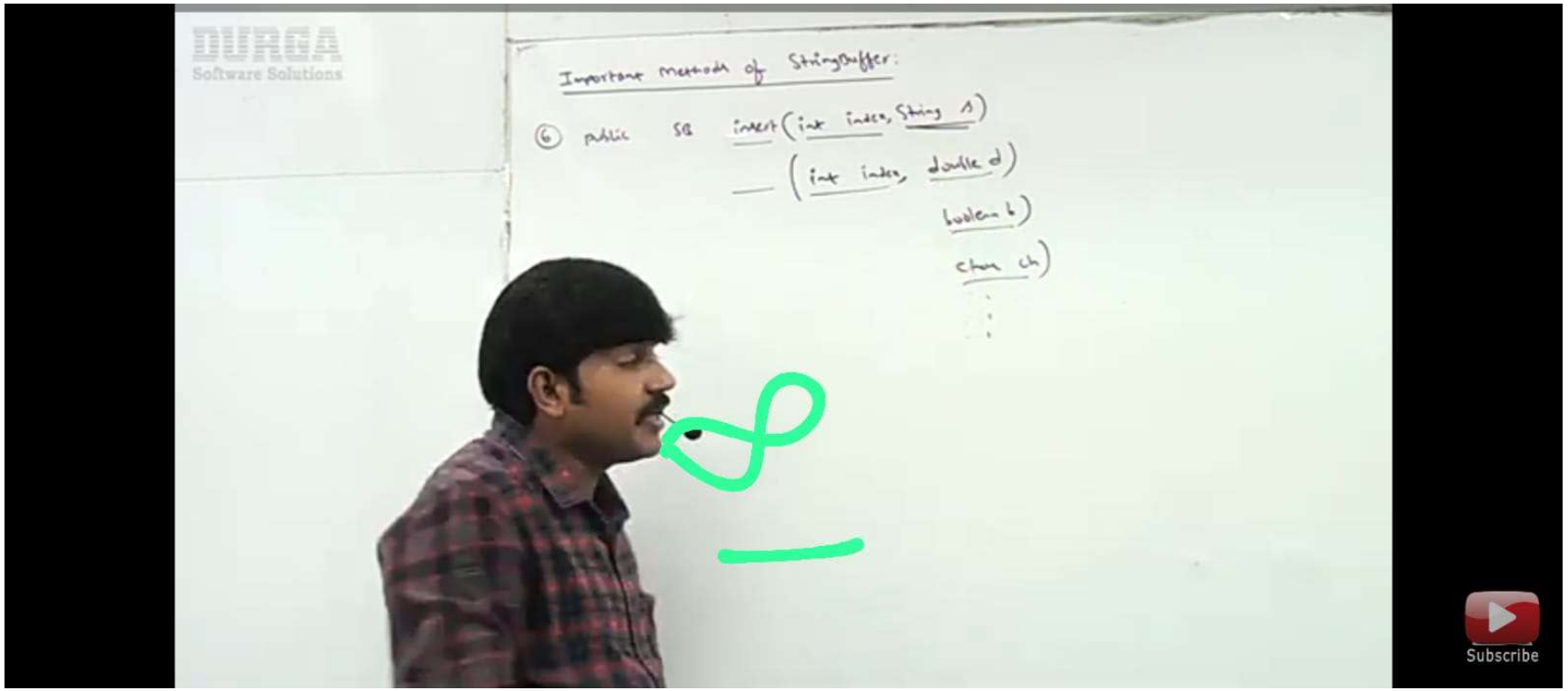

13
class Test
private int i;
Test()
{ this.i=i;
}
Test modify(int i)
{
if(this.i == i)
{
return this;
}
else
{
return new Test(i);
}
Test k1=new Test(10);
Test k2 = k1.modify(100);
Test k3 = k1.modify(10);
Sopn(k1 == k2); false
Sopn(k1 == k3); true
k1
k3
i=10
k2
i=1

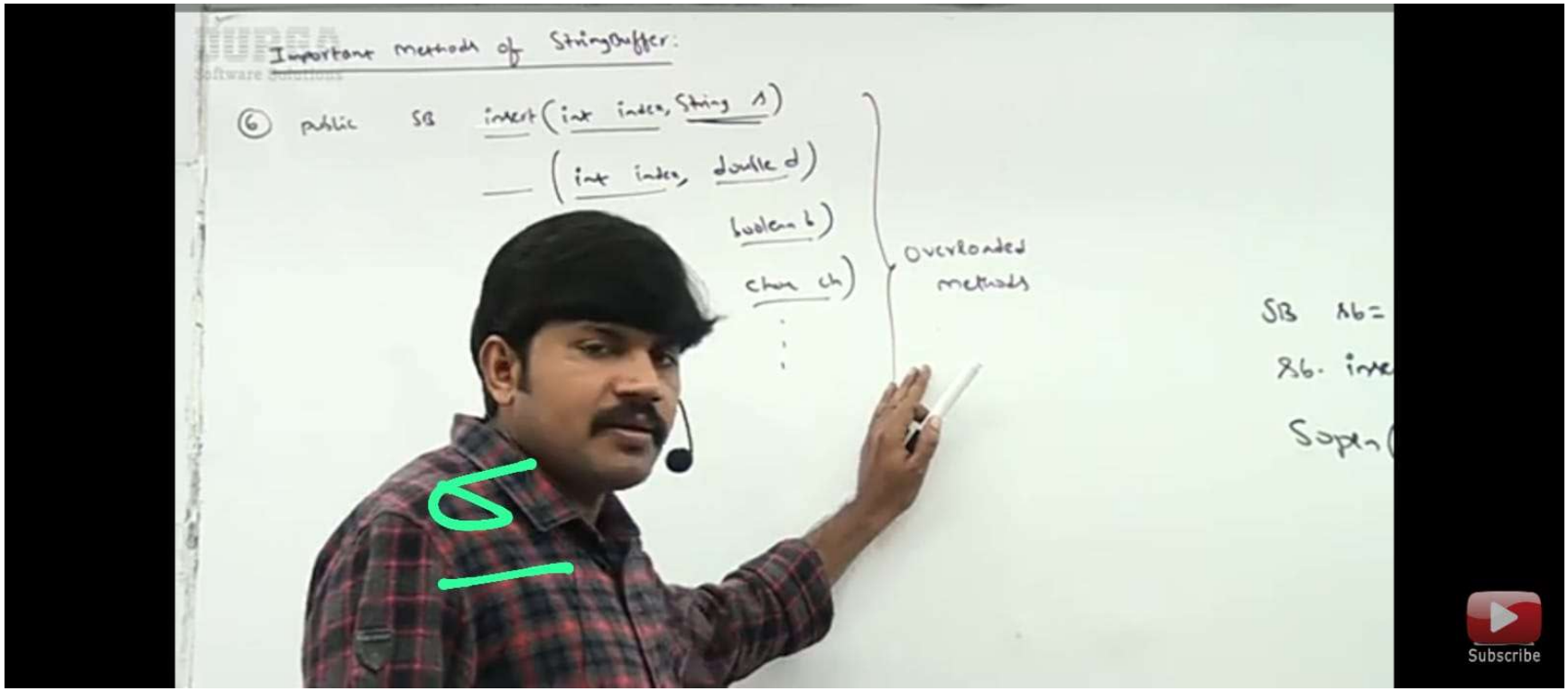

DURGA
Software Solutions
final vs immutability
which of the following are meaningful ?
1) final variable
2) final object
3) Immutable variable
4) Immutable object
Test
main (String[] args)
final StringBuffer sb = new StringBuffer("durga");
sb.append("software");
Sopln(sb);
= new StringBuffer("ravi");
Subscribe

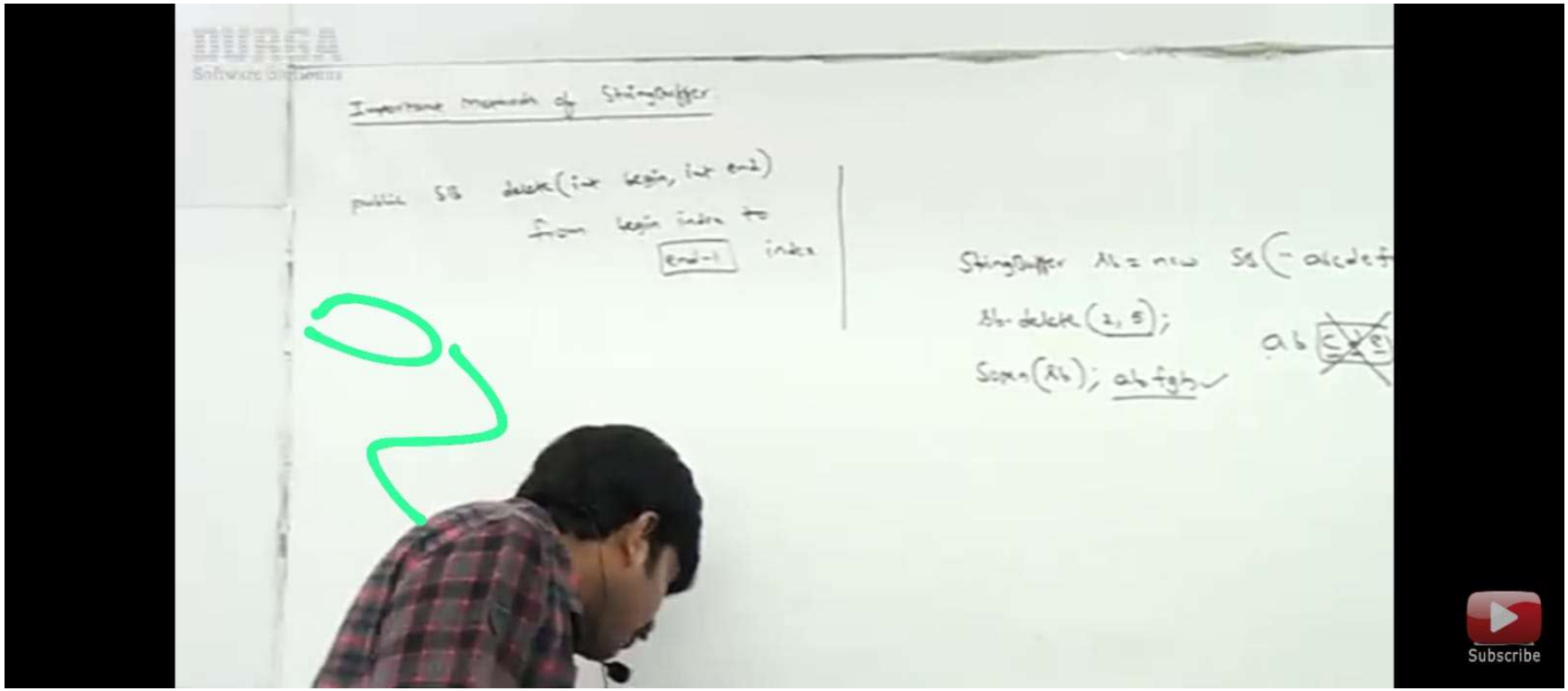

DURGA
Software Solutions
Important
① StringBuffer sb = new StringBuffer();
16
new capacity = (c + 1) * 2
(16+1) * 2 = 34
sb
16
Subscribe

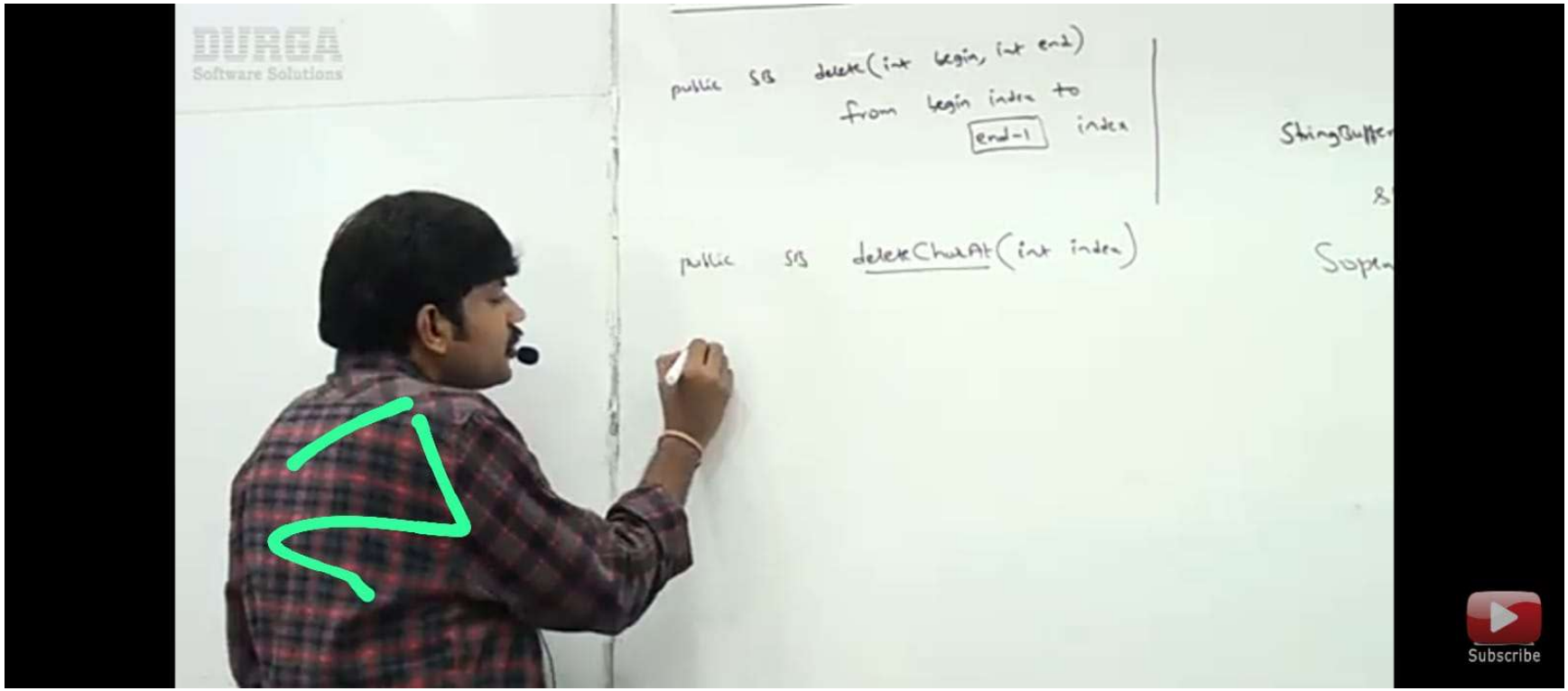

DURGA
Software Solutions
sb = new StringBuffer();
sb = new StringBuffer(int initialCapacity);
1000
sb = new StringBuffer(String s)
capacity = s.length() +16
SB sb = new SB("durga");
Sop(sb.capacity());
5+16 => 21
5
16
80
21
Subscribe

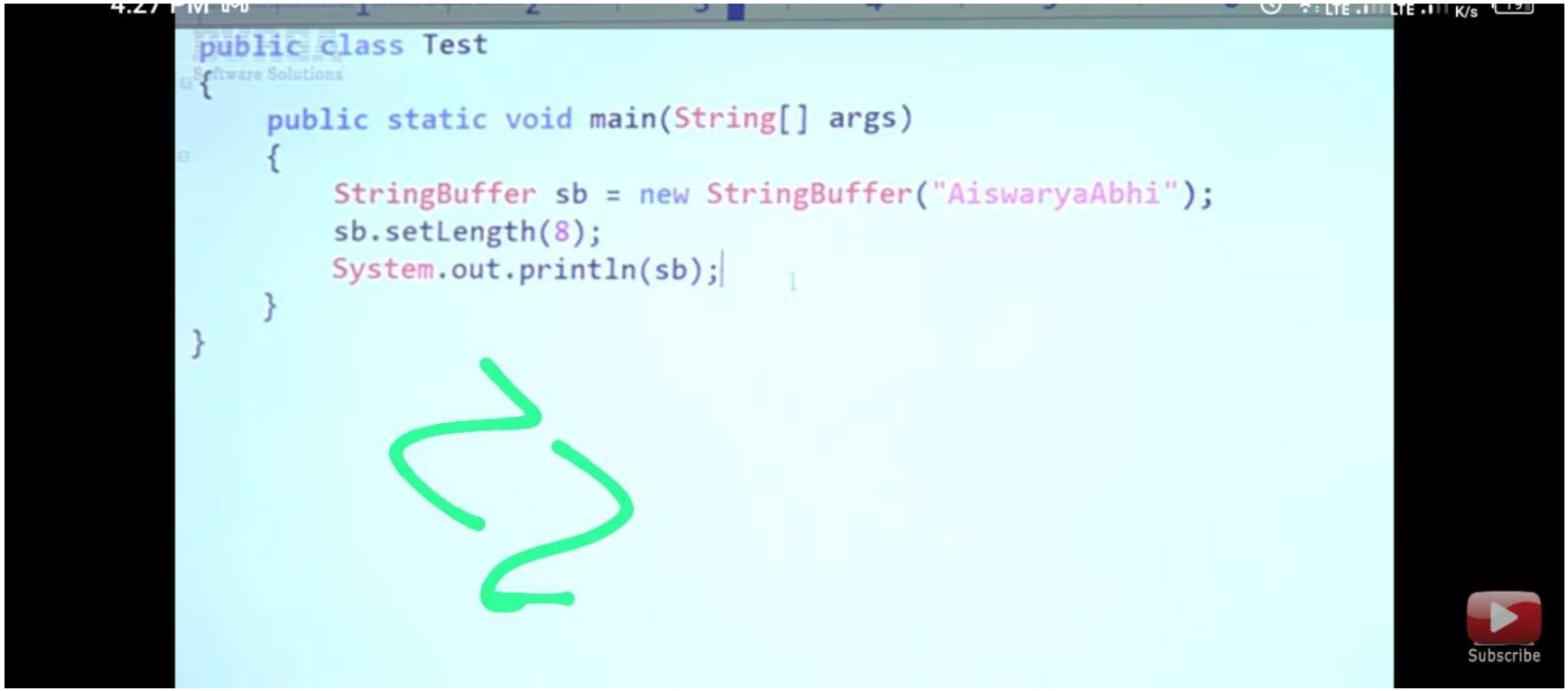

DURGA
Software Solutions
public void setCharAt(int index, char newChar);
public SB append(String s)
append(byte b)
(int i)
(long l)
(float f)
overloaded methods
Subscribe

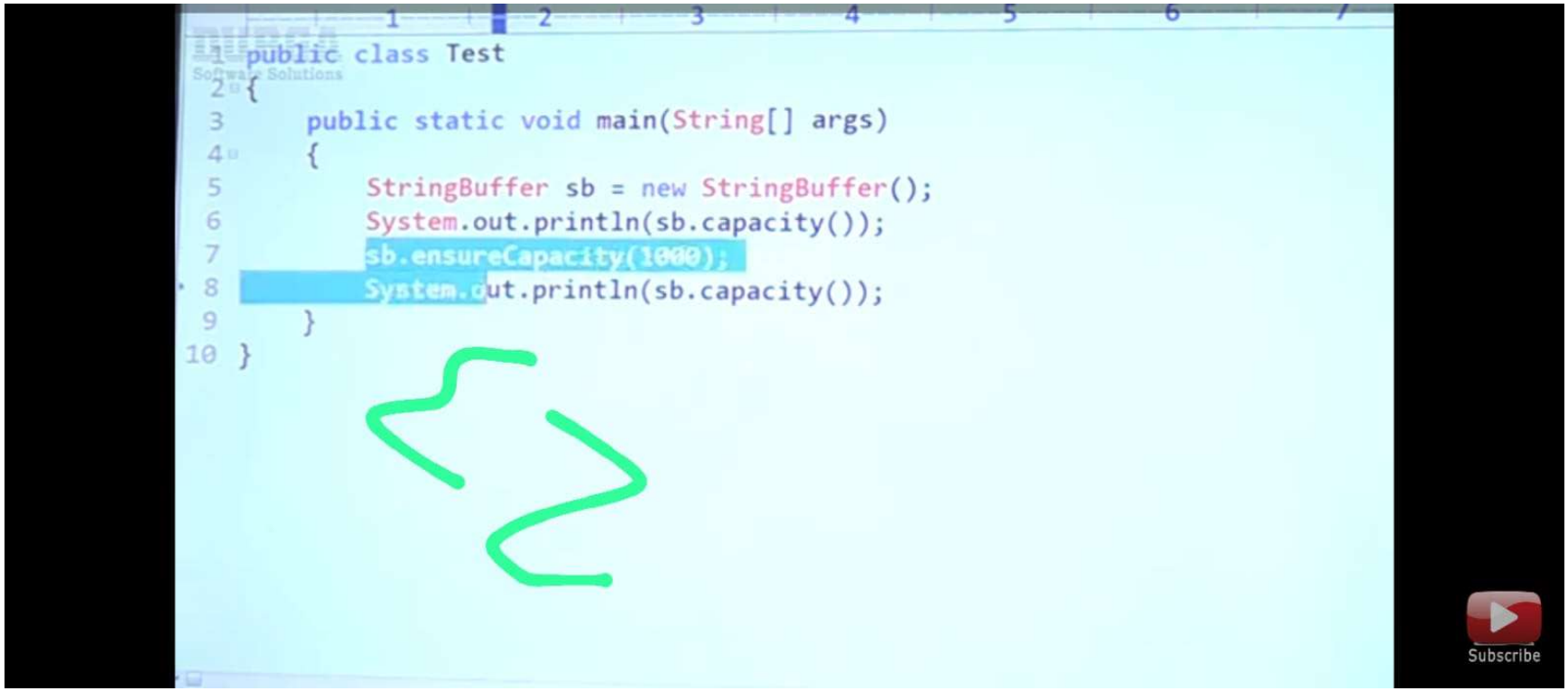

DURGA
Software Solutions
Important methods of StringBuffer:
⑥ public SB insert(int index, String s)
(int index, double d)
boolean b)
char ch)
Subscribe

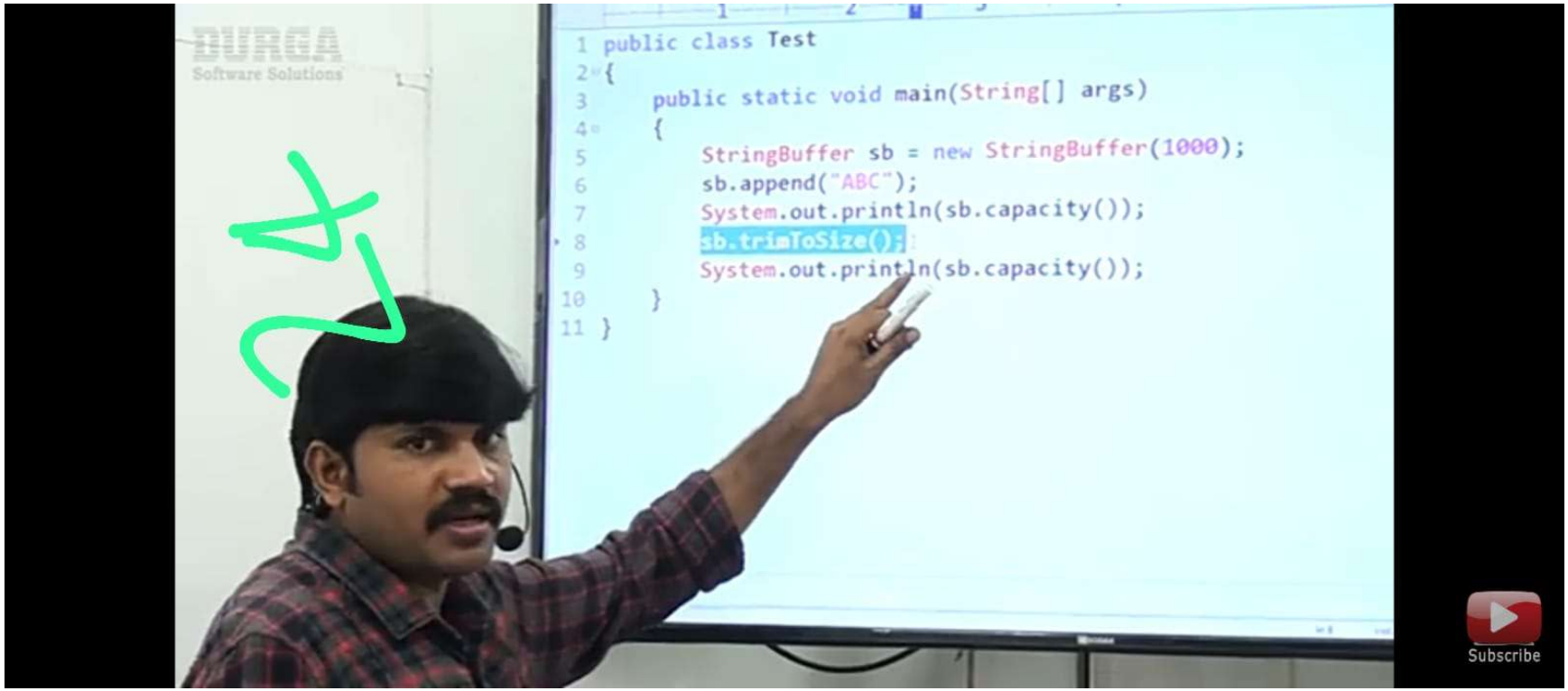

Important methods of StringBuffer:
6 public SB insert(int index, String s)
(int index, double d)
boolean b)
char ch)
overloaded methods
SB sb=
sb. inse
Sopln(
Subscribe

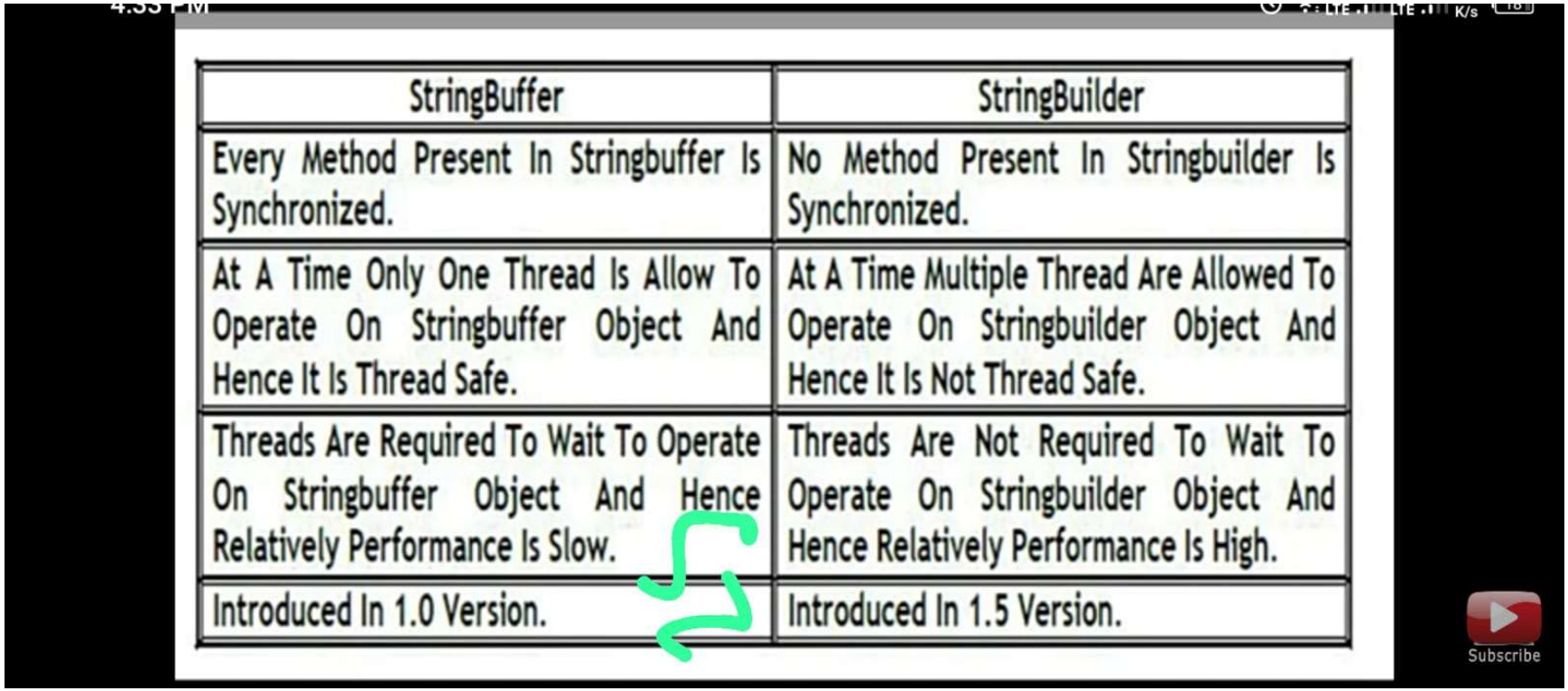

BURSA
Software Solutions
Important methods of StringBuffer
public SB delete(int begin, int end)
from begin index to
end-1 index
StringBuffer Ab = new SB(-abcdef
Ab.delete(2, 5);
Sopn(Ab); abfgh
ab
Subscribe


DURGA
Software Solutions
public SB delete(int begin, int end)
from begin index to
end-1 index
public SB deleteCharAt(int index)
Subscribe


public class Test
{
public static void main(String[] args)
{
StringBuffer sb = new StringBuffer("AiswaryaAbhi");
sb.setLength(8);
System.out.println(sb);
}
}
Subscribe


1 2 3 4 5 6 7
1 public class Test
2 {
3 public static void main(String[] args)
4 {
5 StringBuffer sb = new StringBuffer();
6 System.out.println(sb.capacity());
7 sb.ensureCapacity(1000);
8 System.out.println(sb.capacity());
9 }
10 }
Subscribe


DURGA
Software Solutions
24
1 public class Test
2 {
3 public static void main(String[] args)
4 {
5 StringBuffer sb = new StringBuffer(1000);
6 sb.append("ABC");
7 System.out.println(sb.capacity());
8 sb.trimToSize();
9 System.out.println(sb.capacity());
10 }
11 }
Subscribe



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.