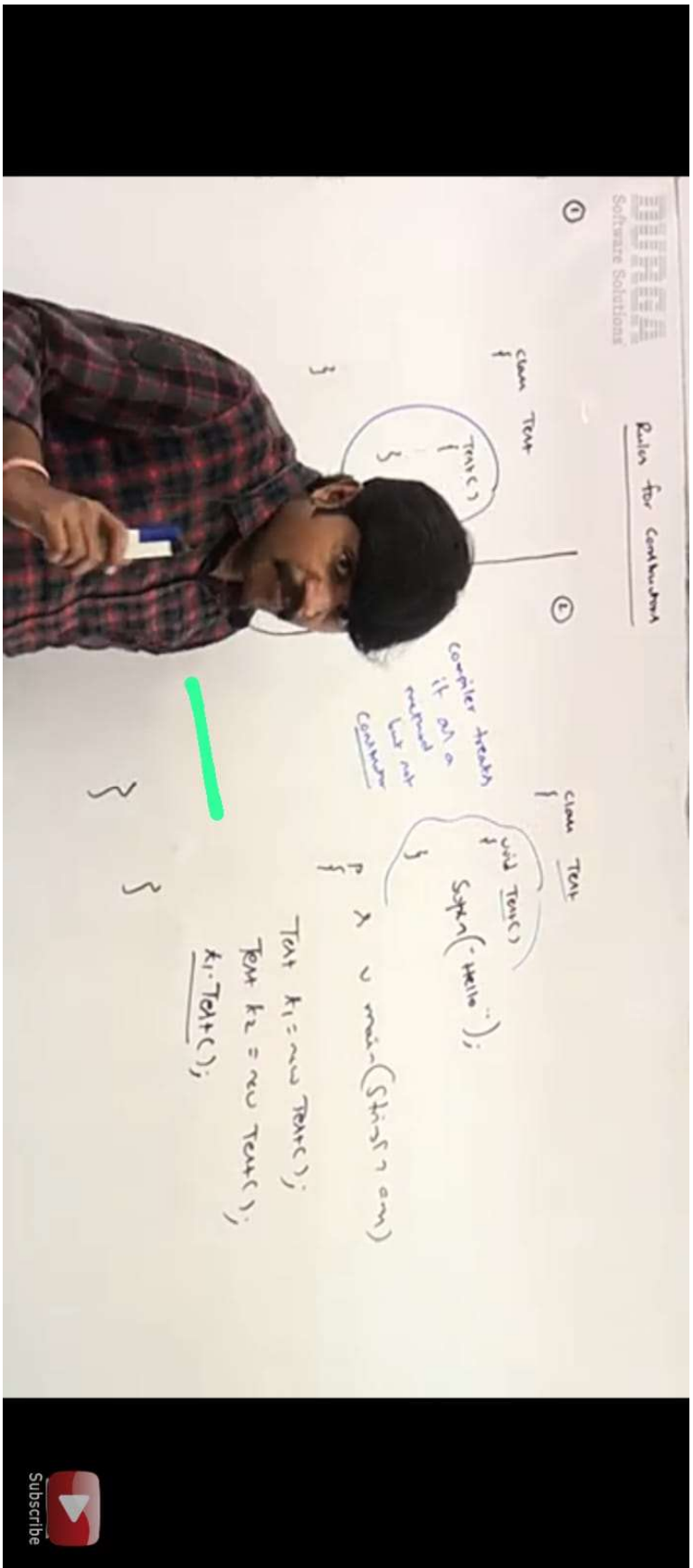
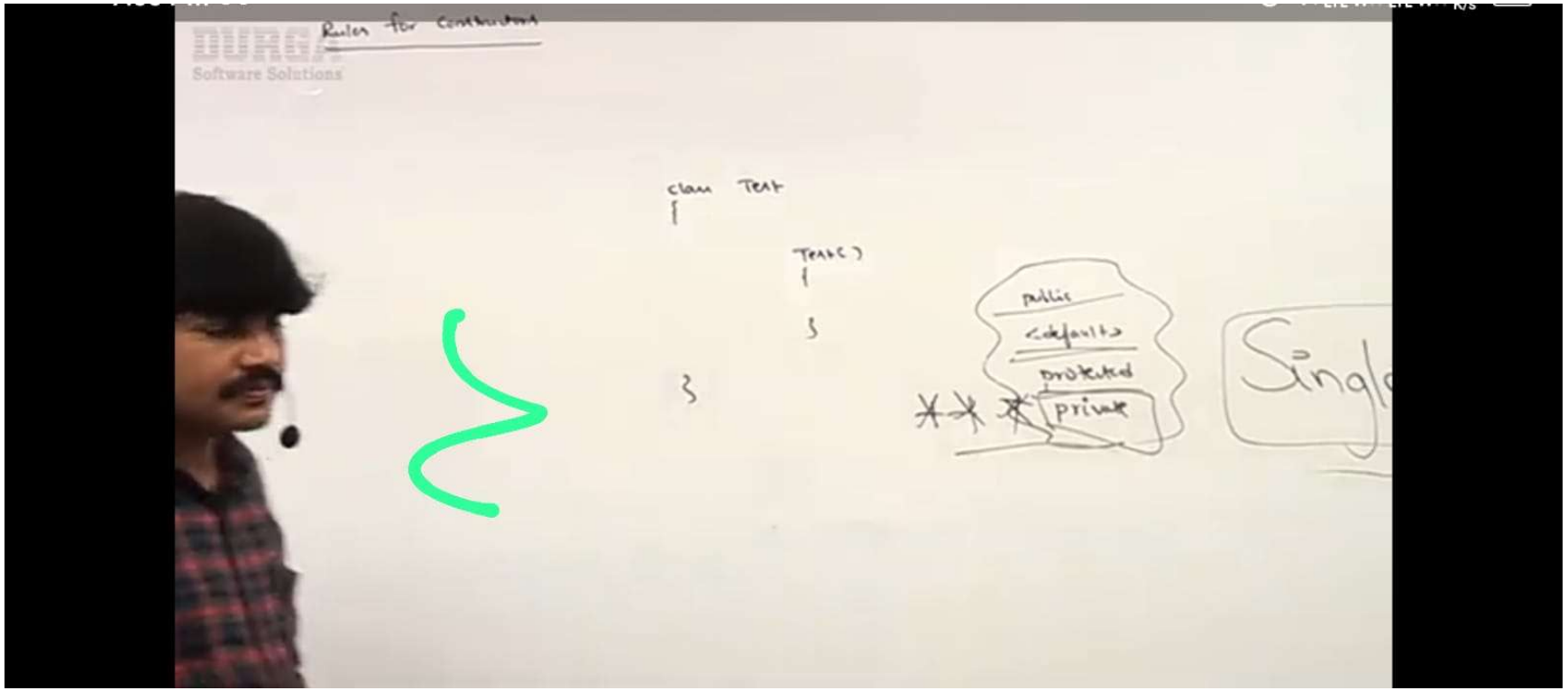
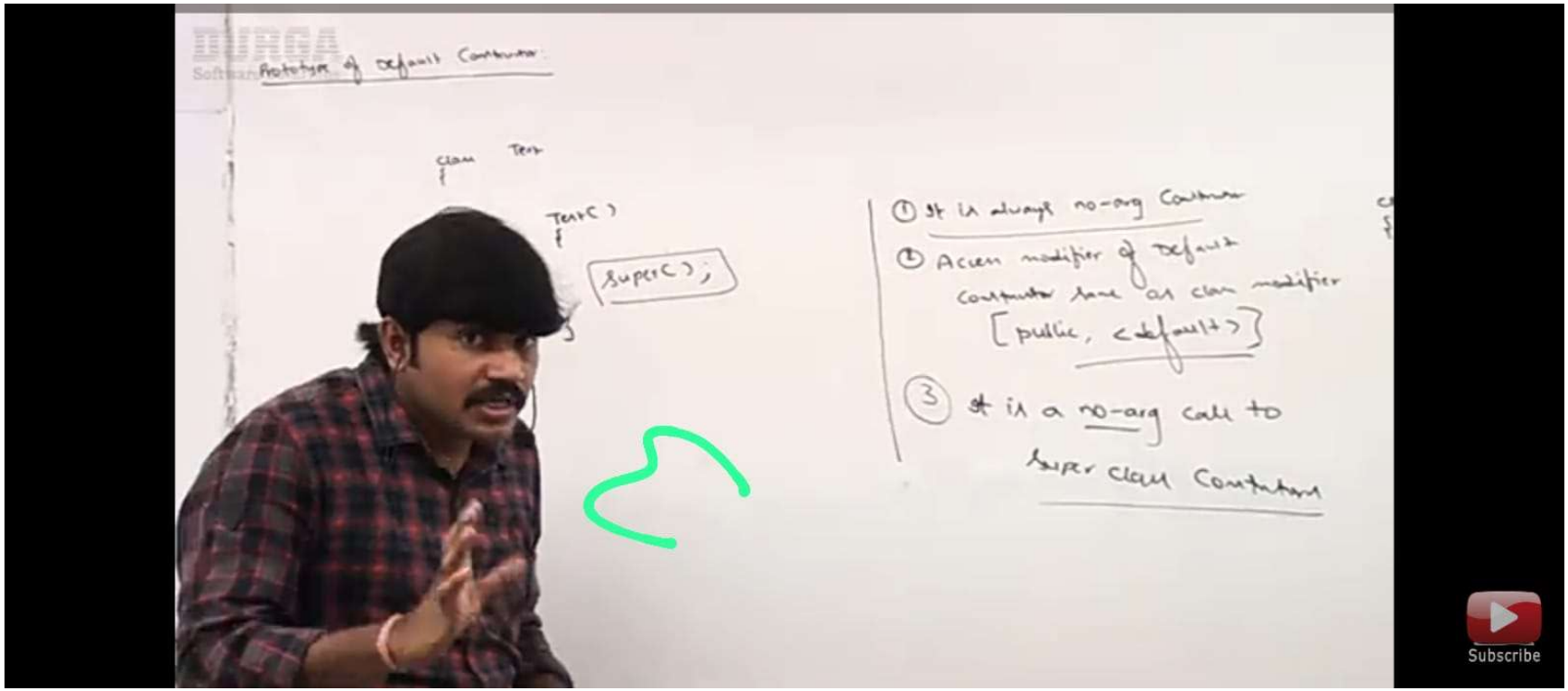
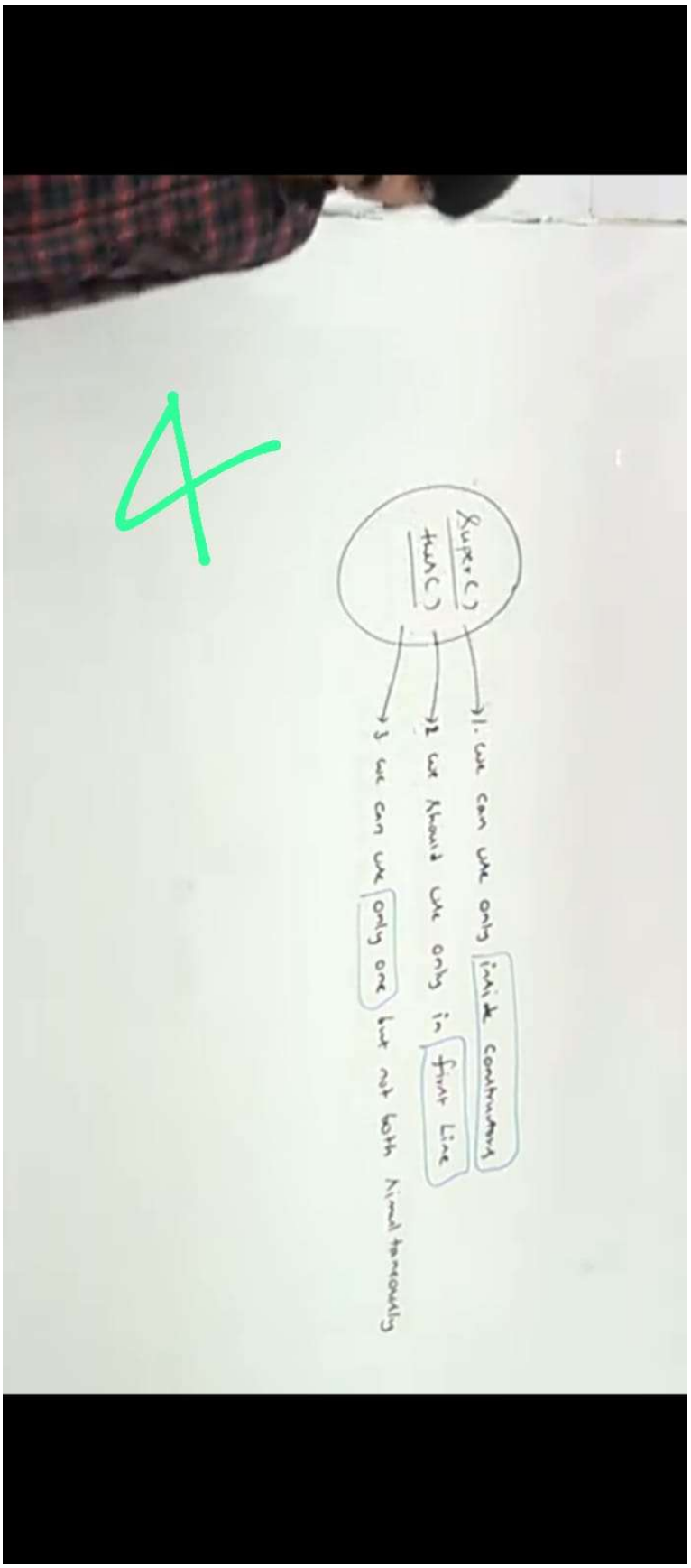
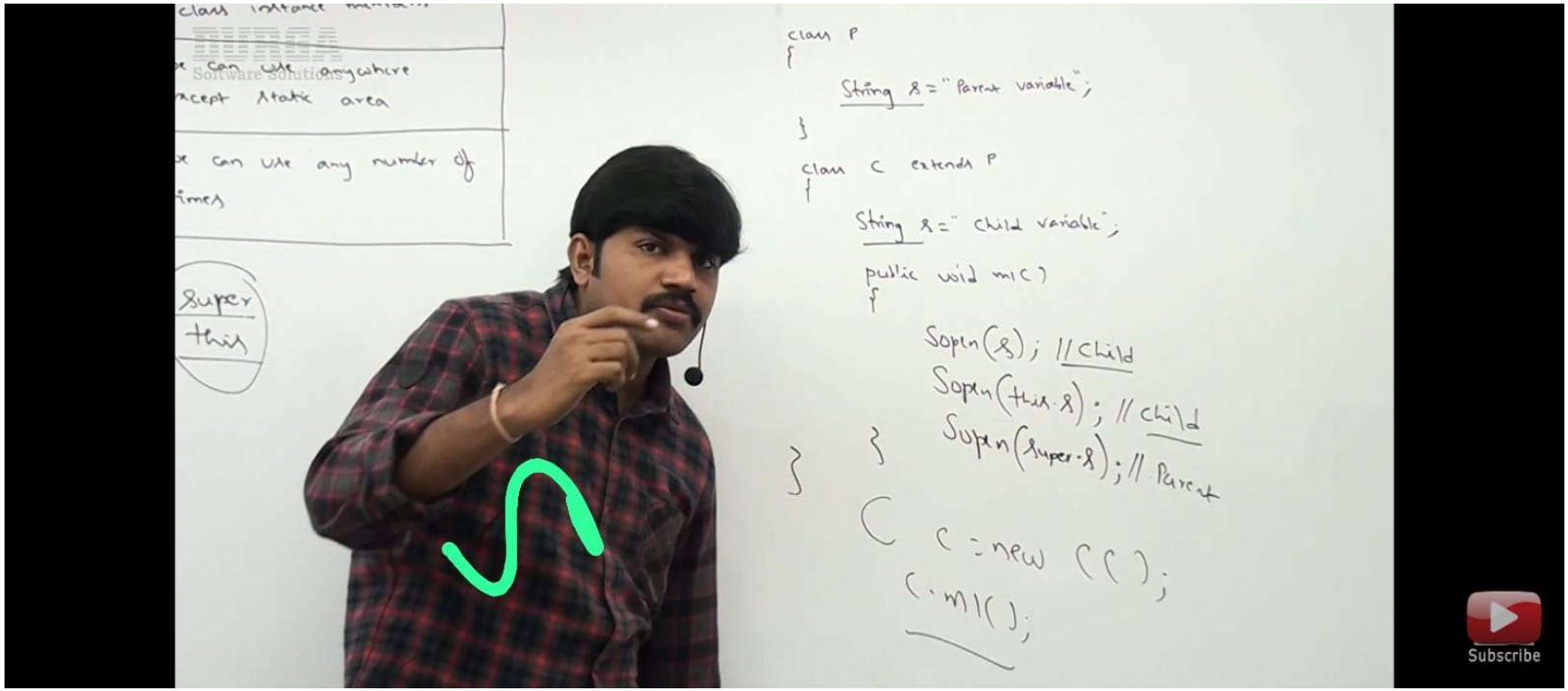
Constructor,this,super

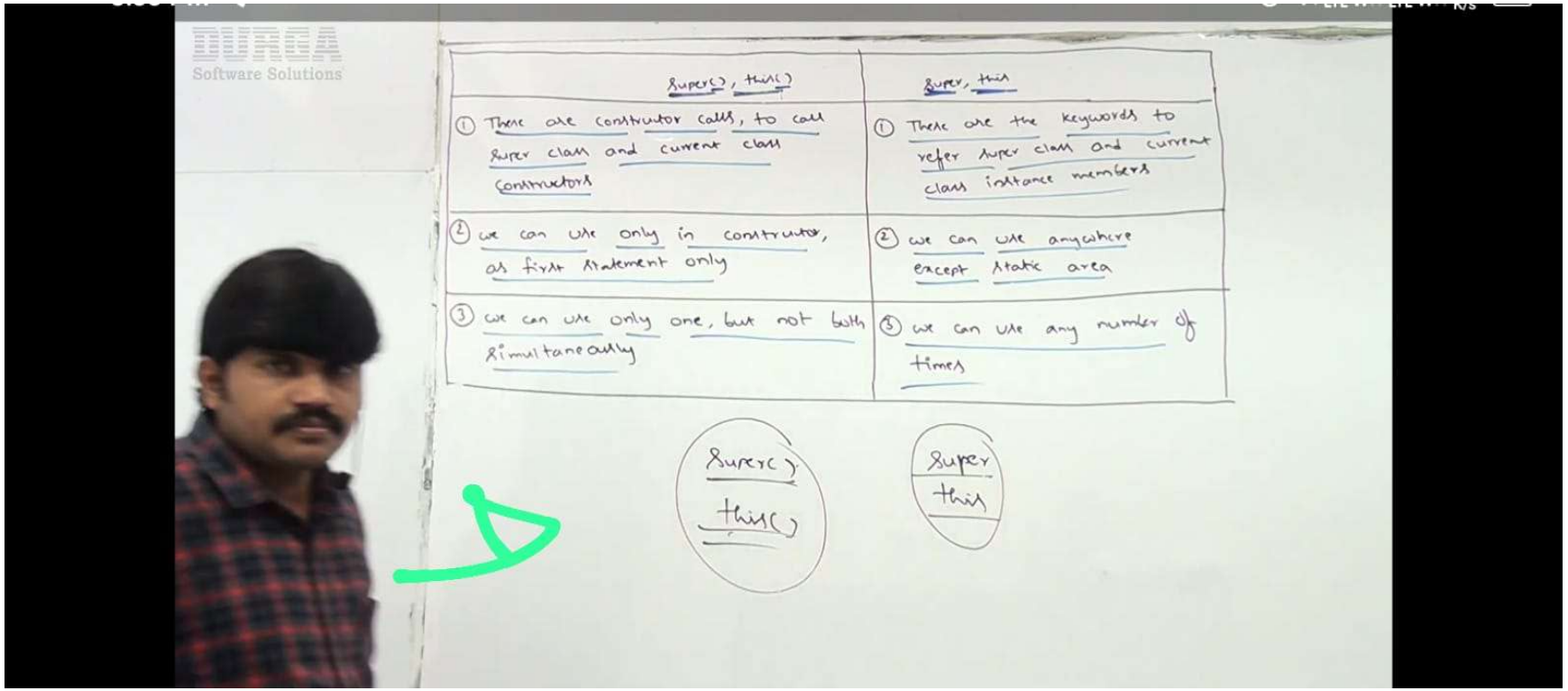


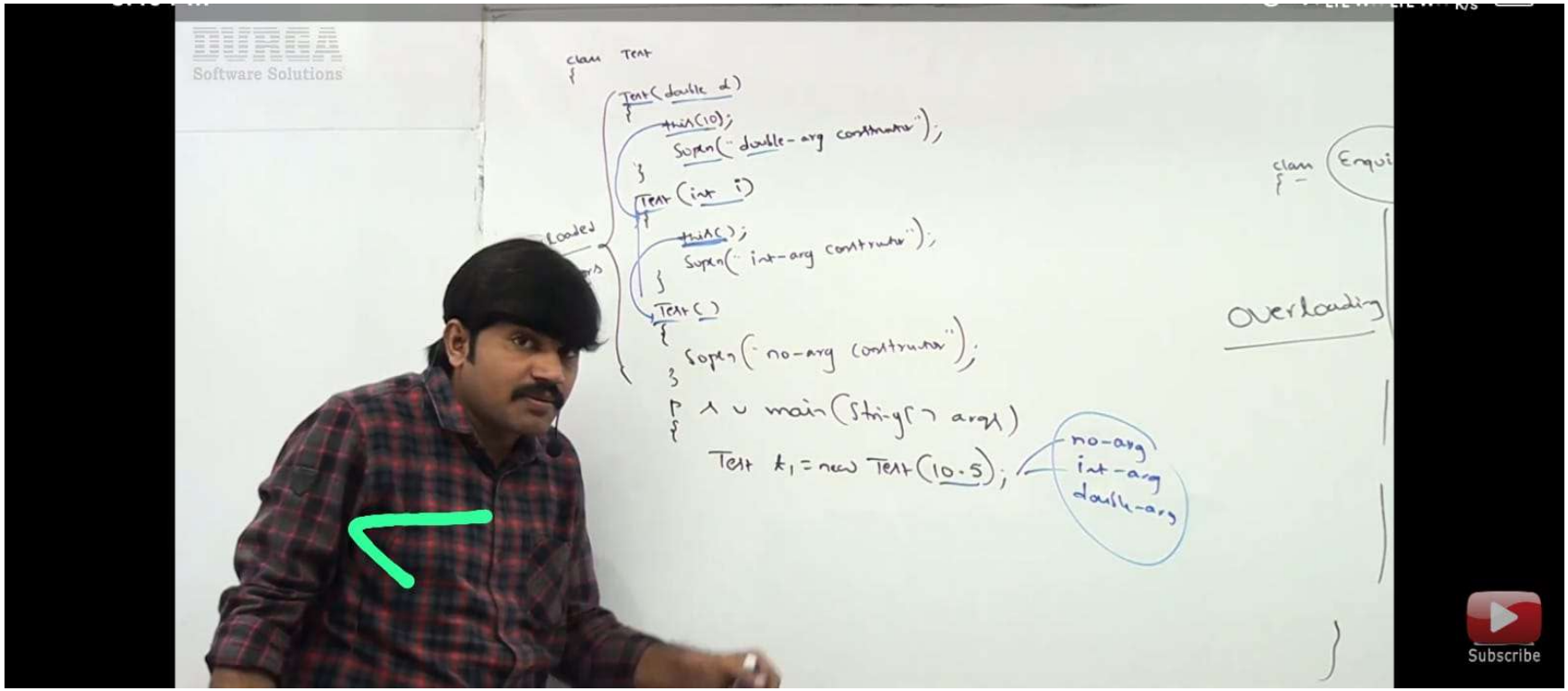

Rules for Constructors
DURGA
Software Solutions
class Test
{
Test()
{
}
}
public
<default>
protected
private
Single

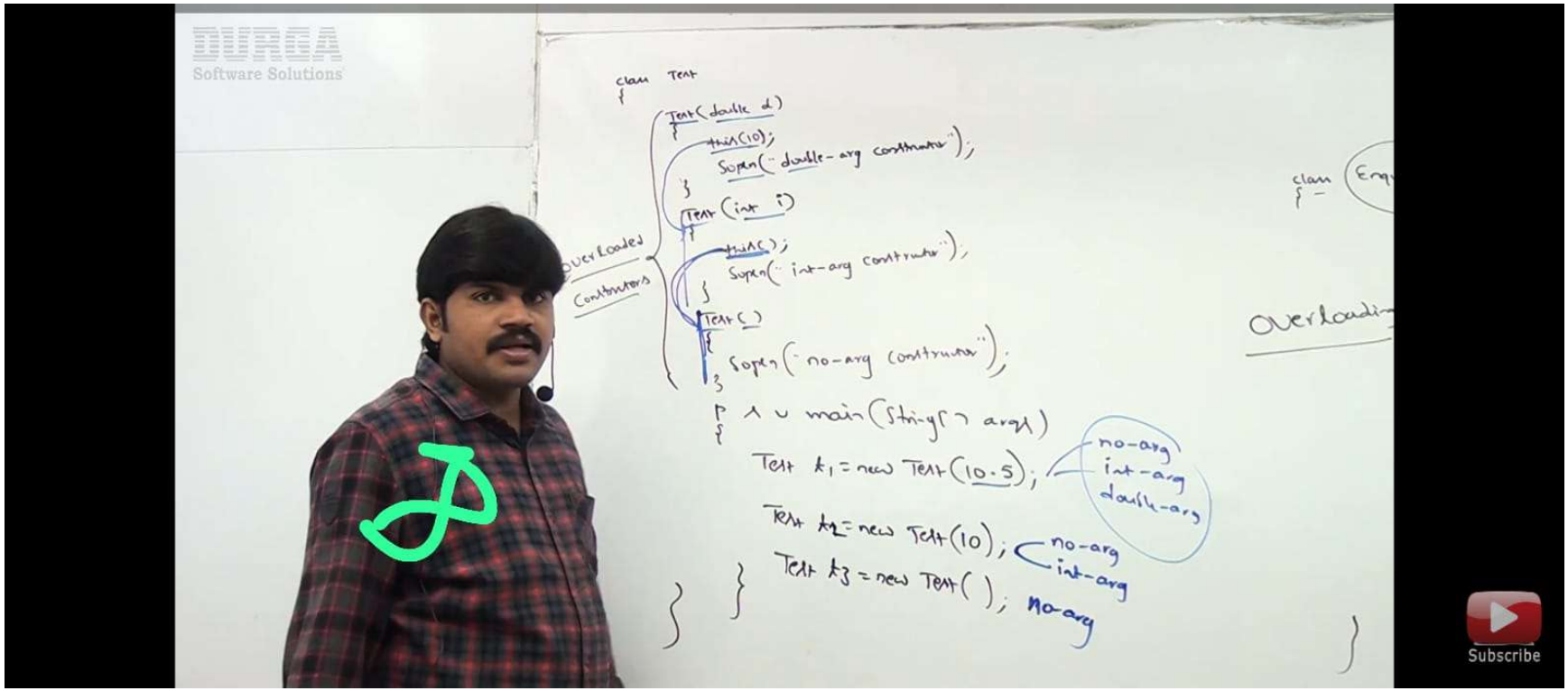

DURGA
Prototype of default Constructor
class Test
{
Test()
{
super();
}
1) It is always no-arg Constructor
2) Access modifier of default constructor have as class modifier
[public, <default>]
3) It is a no-arg call to super class Constructor
Subscribe

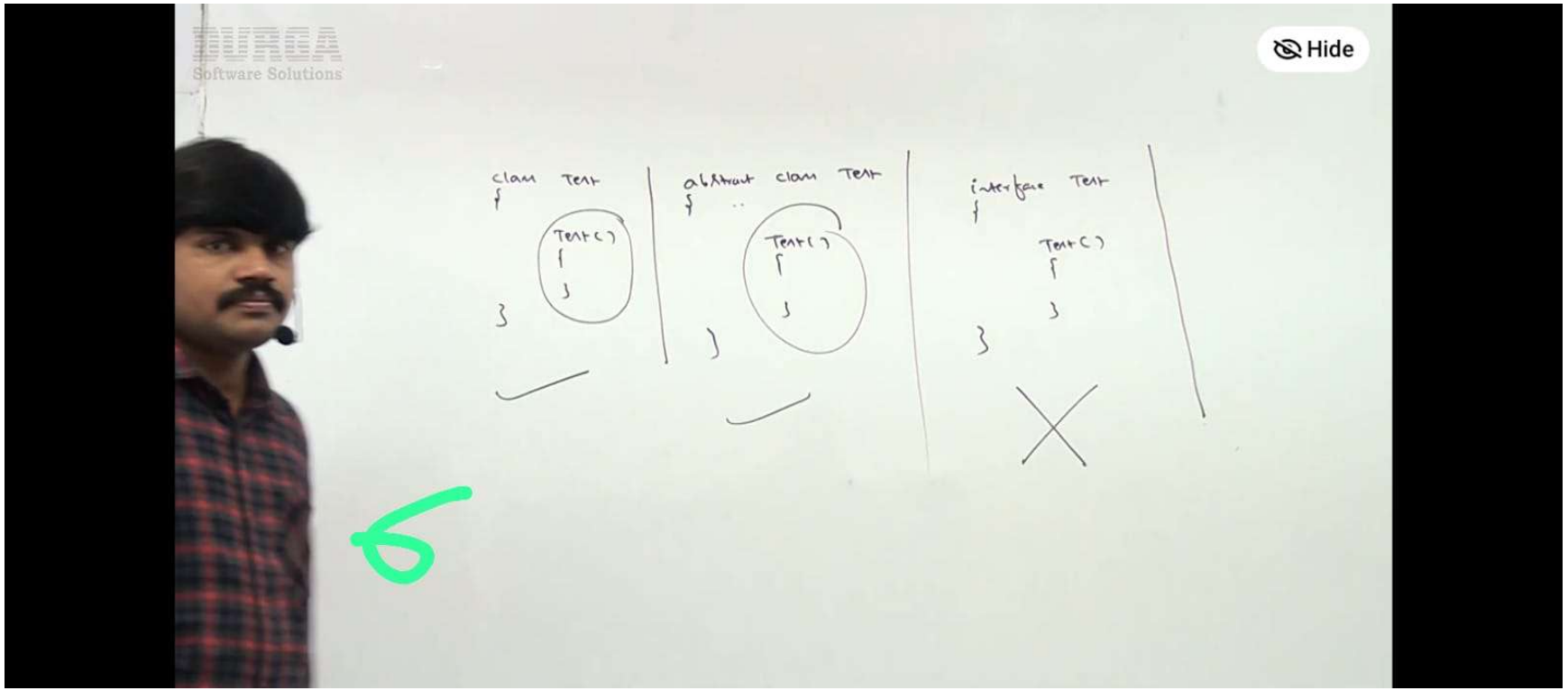

4
super()
this()
1. we can use only inside constructors
2 we should use only in first line
3 we can use only one but not both

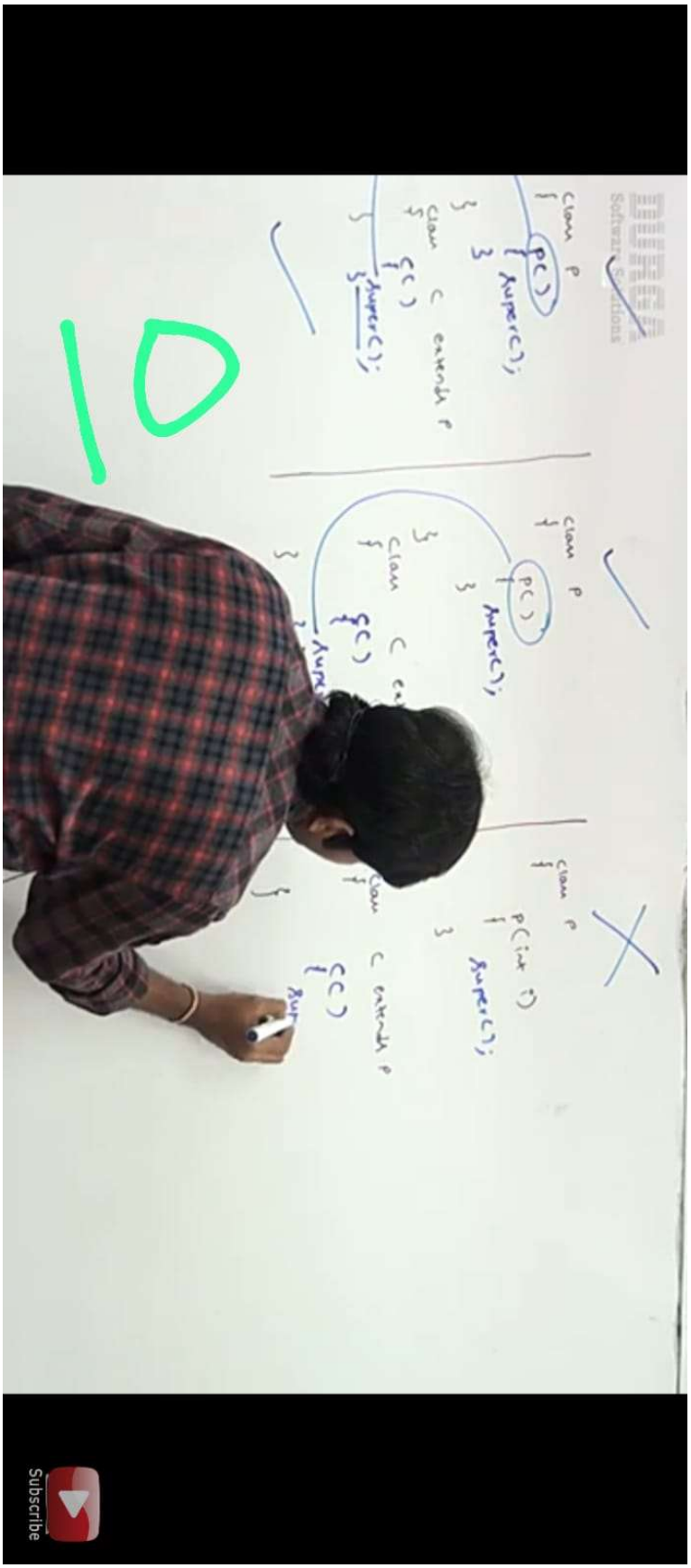

can use anywhere
except static area
can use any number of
times
super
this
class P
{
String s = "Parent variable";
}
class C extends P
{
String s = "child variable";
public void m1()
{
Sopln(s); // child
Sopln(this.s); // child
Sopln(super.s); // Parent
}
}
C c = new C();
c.m1();
Subscribe

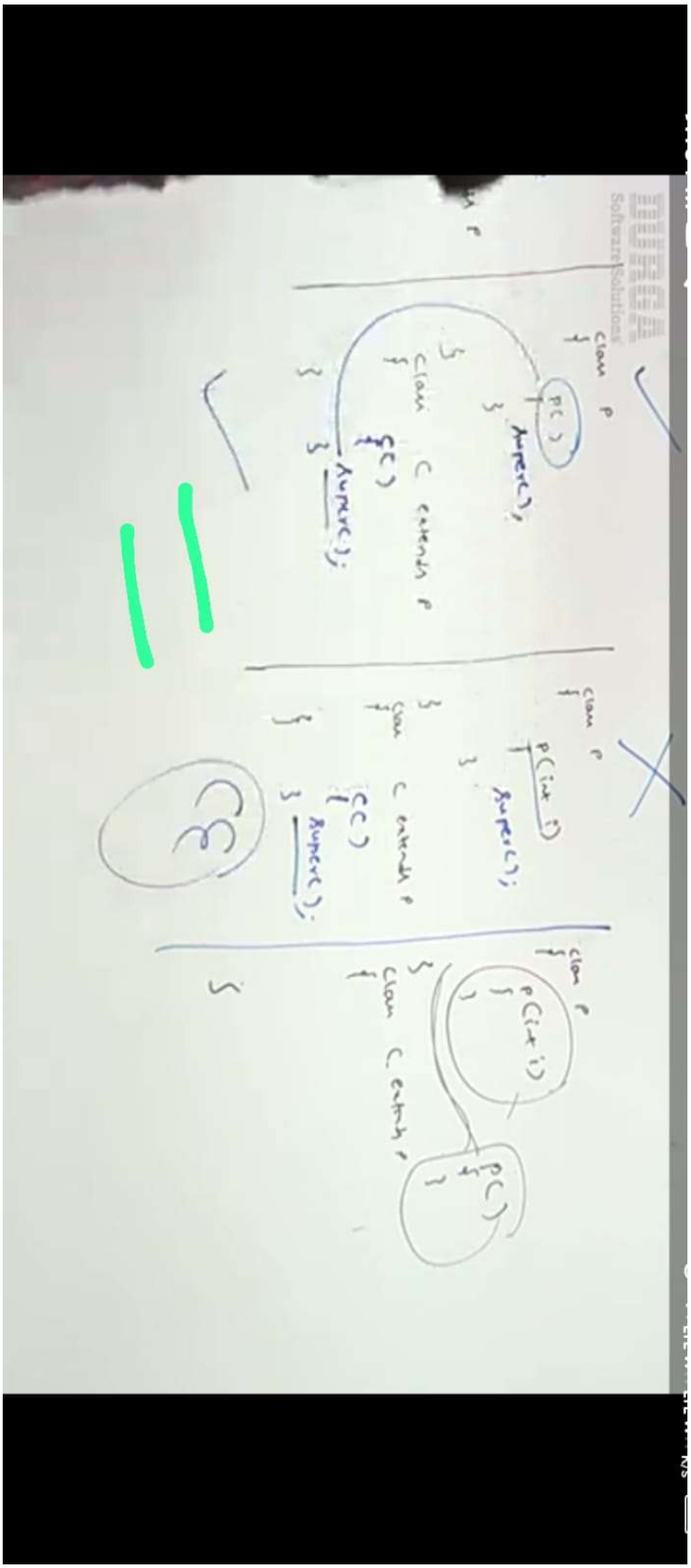

DURGA
Software Solutions
super(), this()
super, this
① These are constructor calls, to call super class and current class constructors
② we can use only in constructor, as first statement only
③ we can use only one, but not both simultaneously
① These are the keywords to refer super class and current class instance members
② we can use anywhere except static area
③ we can use any number of times
super()
this()
super
this


DURGA
Software Solutions
class Test
{
Test(double d)
{
this(10);
Sopln("double-arg constructor");
}
Test(int i)
{
this();
Sopln("int-arg constructor");
}
Test()
{
Sopln("no-arg constructor");
}
public static void main(String[] args)
{
Test t1 = new Test(10.5);
}
}
Loaded
class Enqui
Overloading
no-arg
int-arg
double-arg
Subscribe


DURGA
Software Solutions
class Test
Test(double d)
this(10);
Sopln("double-arg constructor");
Test(int i)
this();
Sopln("int-arg constructor");
Test()
Sopln("no-arg constructor");
OverLoaded Constructors
p s v main(String args[])
Test t1 = new Test(10.5);
Test t2 = new Test(10);
Test t3 = new Test();
no-arg
int-arg
double-arg
no-arg
int-arg
no-arg
Overloading
Subscribe


DURGA
Software Solutions
Hide
class Test
{
Test()
{
}
}
abstract class Test
{
Test()
{
}
}
interface Test
{
Test()
{
}
}
6


Subscribe


class P
P()
super();
class C extends P
C()
super();
P(int i)
super();
class C extends P
C()
super();
class P
P(int i)
class C extends P
P()