5.9:. Complete setter getter :-

2## Inheritance :- Is the process. Inheriting or accuring the properties of one class into another class.

In Java. inheritance can be achtive. extends key coord.

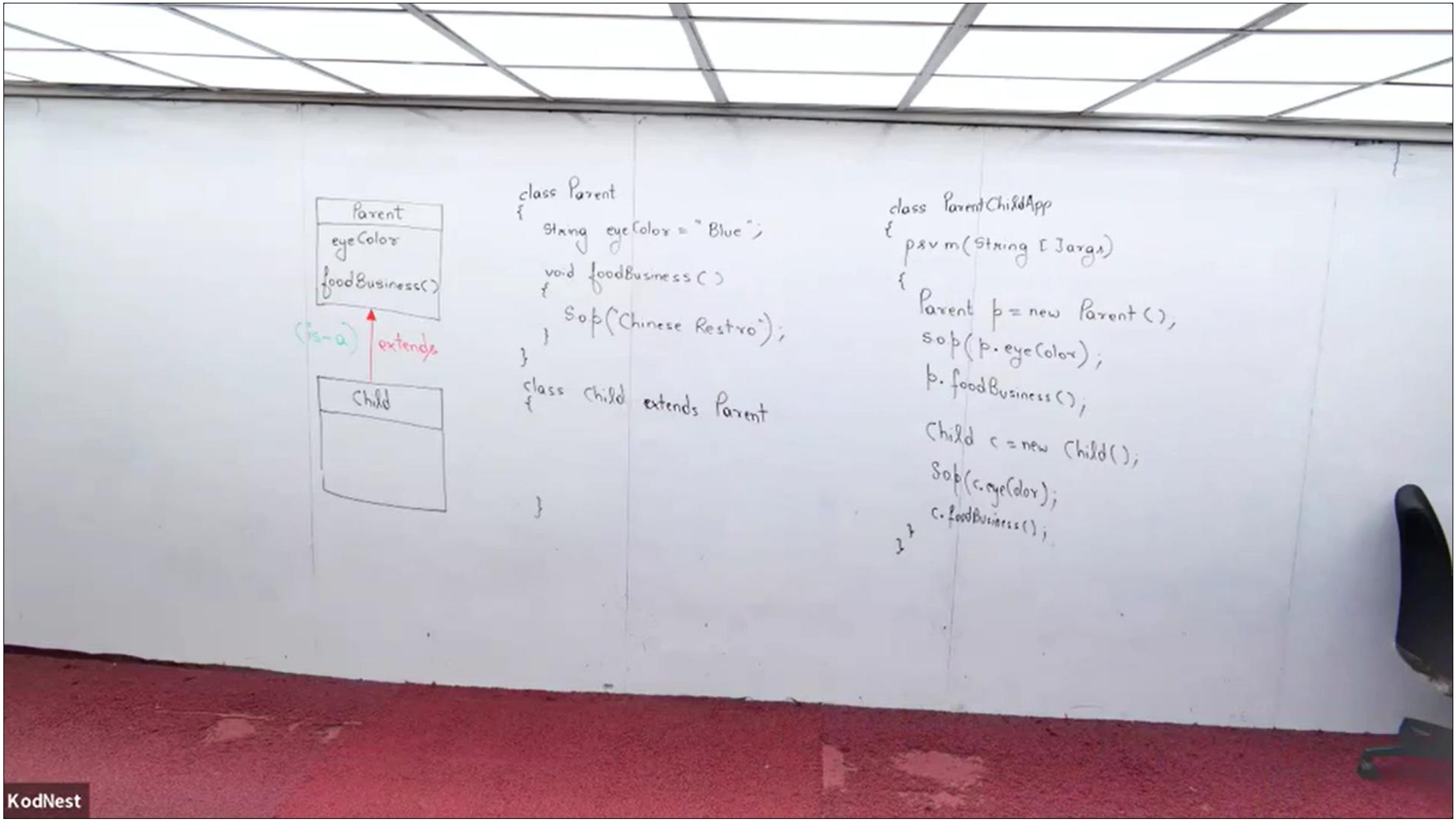
The class which gives properties and behaviour is called as parent class and the class which takes Properties and behaviour is called as child class.

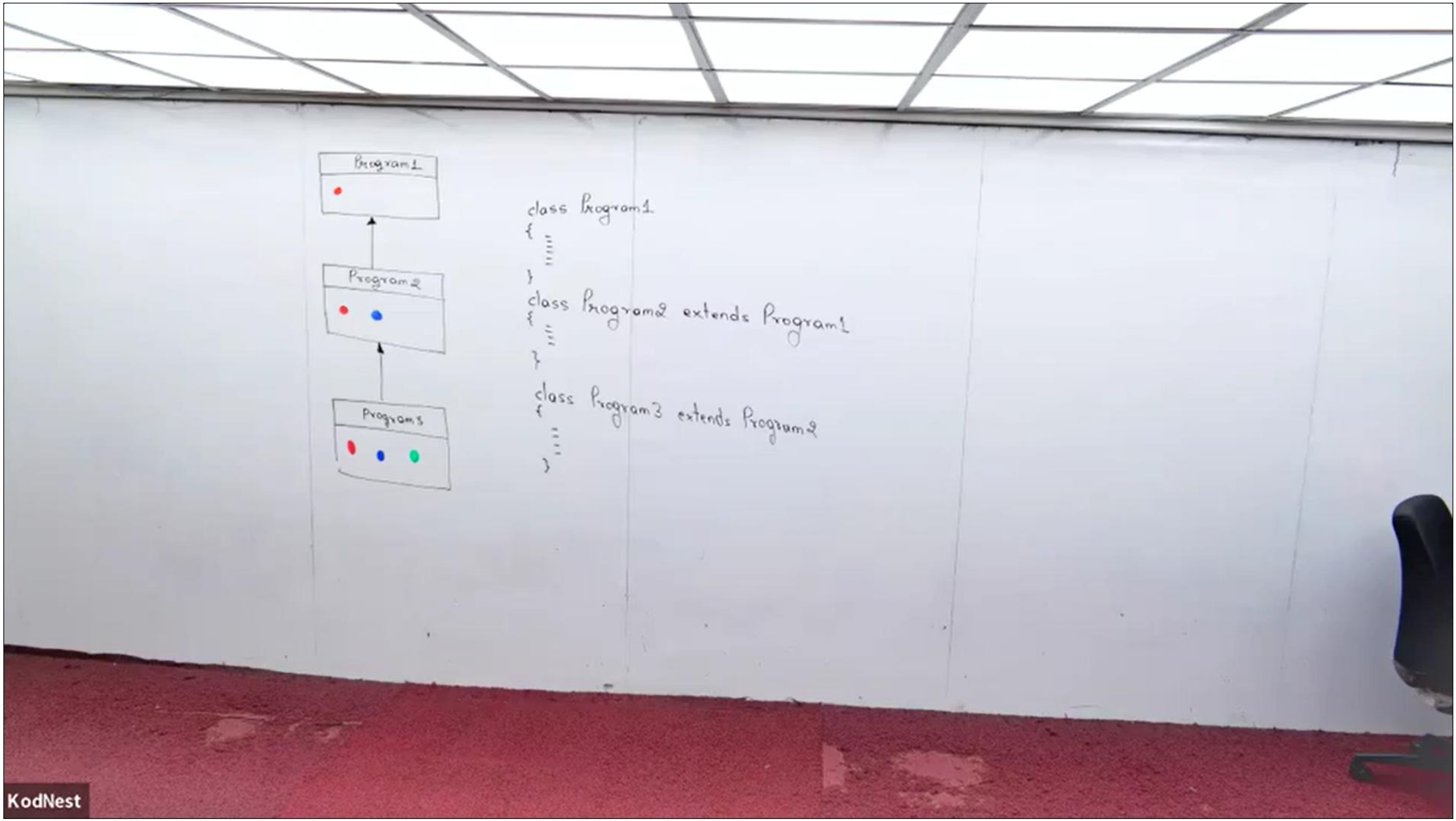
parent class is also called as Supper class and base class

child. class. is also called as drivied and Sub class

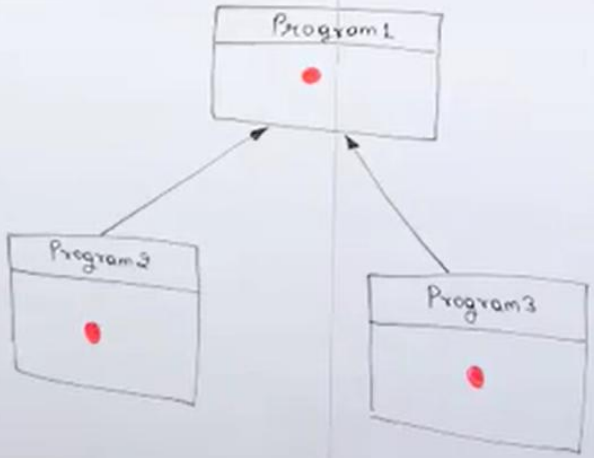
S1. S3

* We can achieve inheritance with extends keycoord

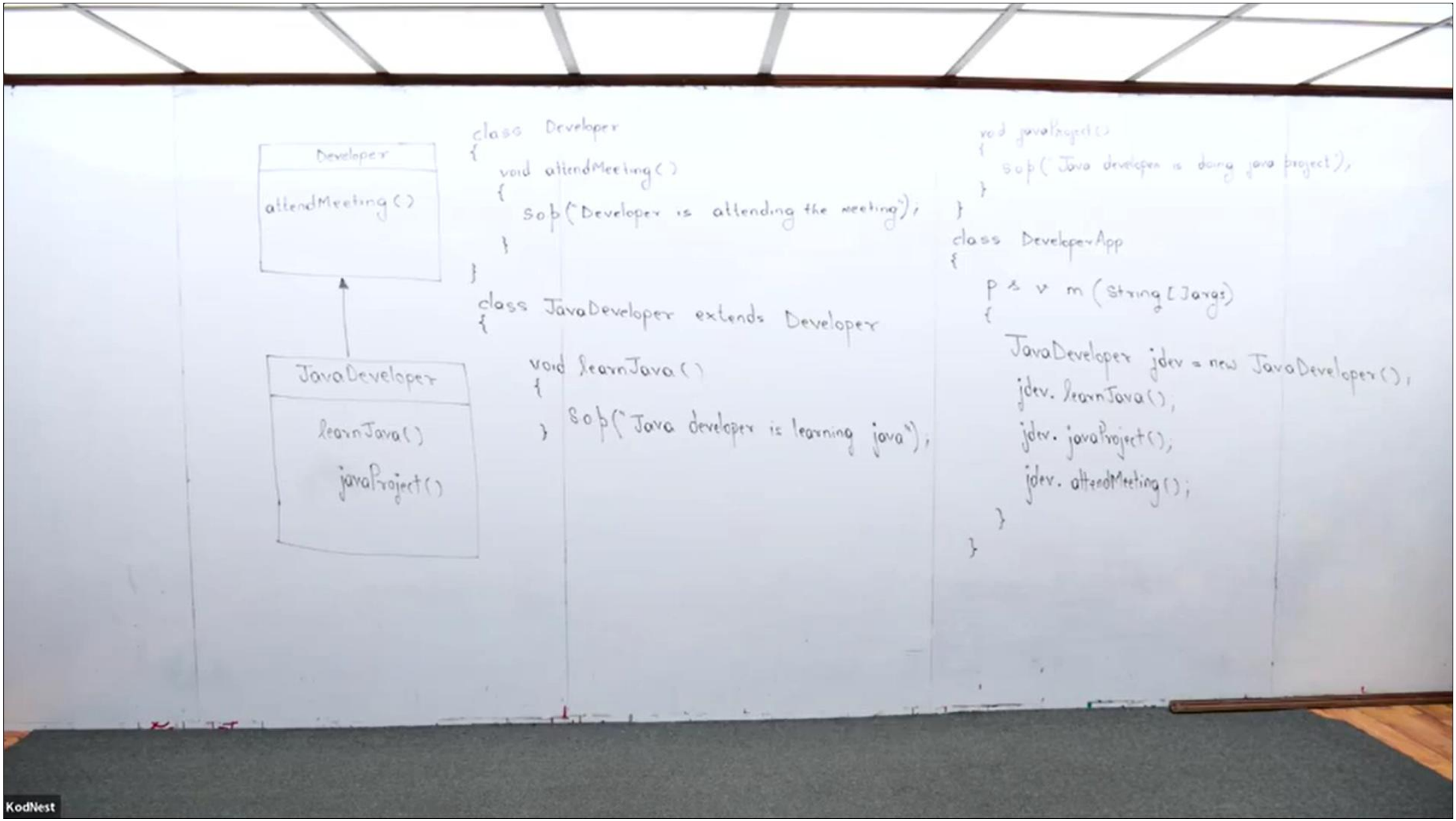

Parent
eyeColor
foodBusiness()
("is-a")
extends
Child
class Parent
{
String eyeColor = "Blue";
void foodBusiness()
{
Sop("Chinese Restro");
}
}
class Child extends Parent
{
}
class ParentChildApp
{
psvm(String [] args)
{
Parent p = new Parent();
Sop(p.eyeColor);
p.foodBusiness();
Child c = new Child();
Sop(c.eyeColor);
c.foodBusiness();
}
}
KodNest

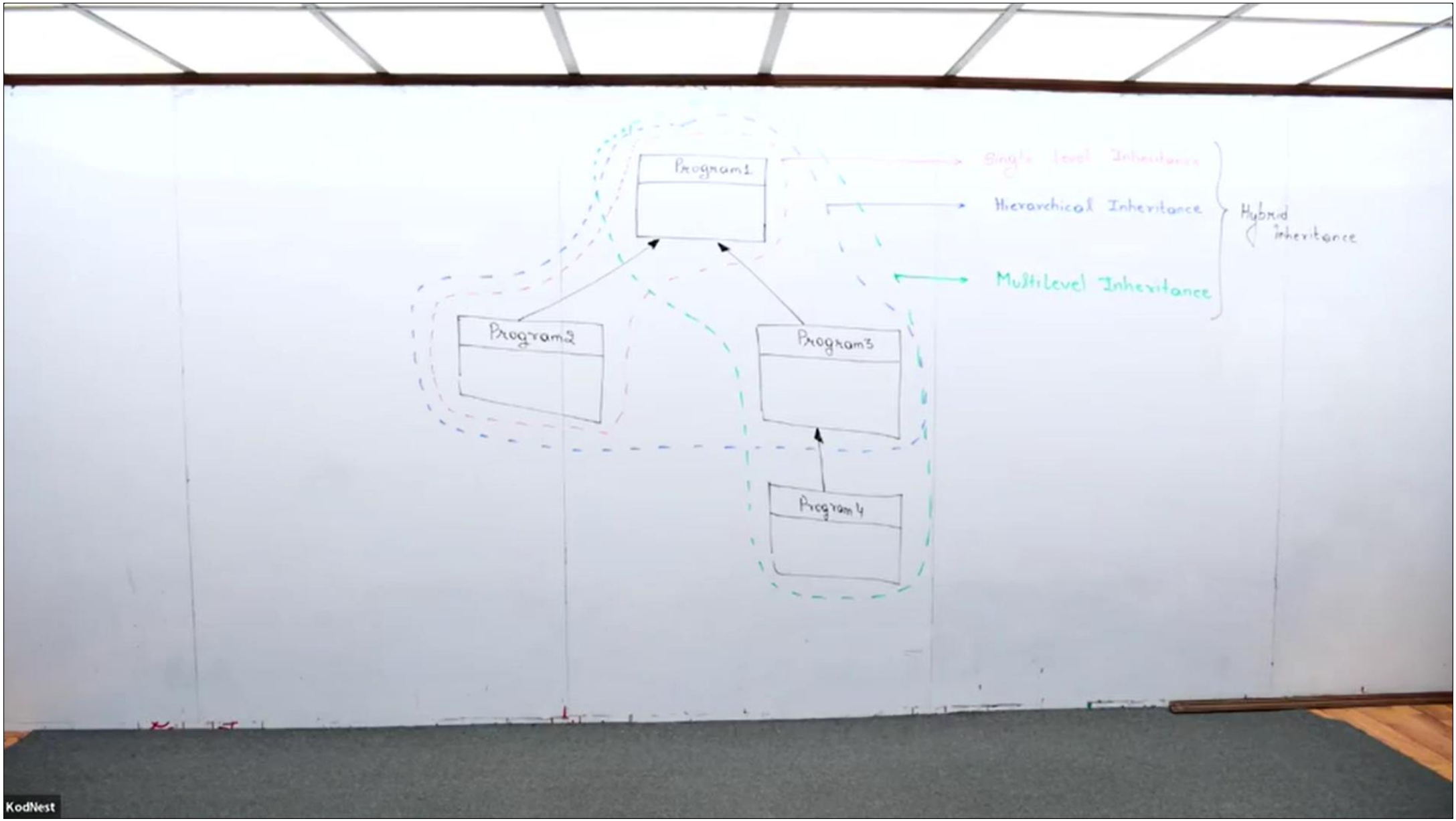

Program1
Program2
Program3
class Program1
{
}
class Program2 extends Program1
{
}
class Program3 extends Program2
{
}
KodNest

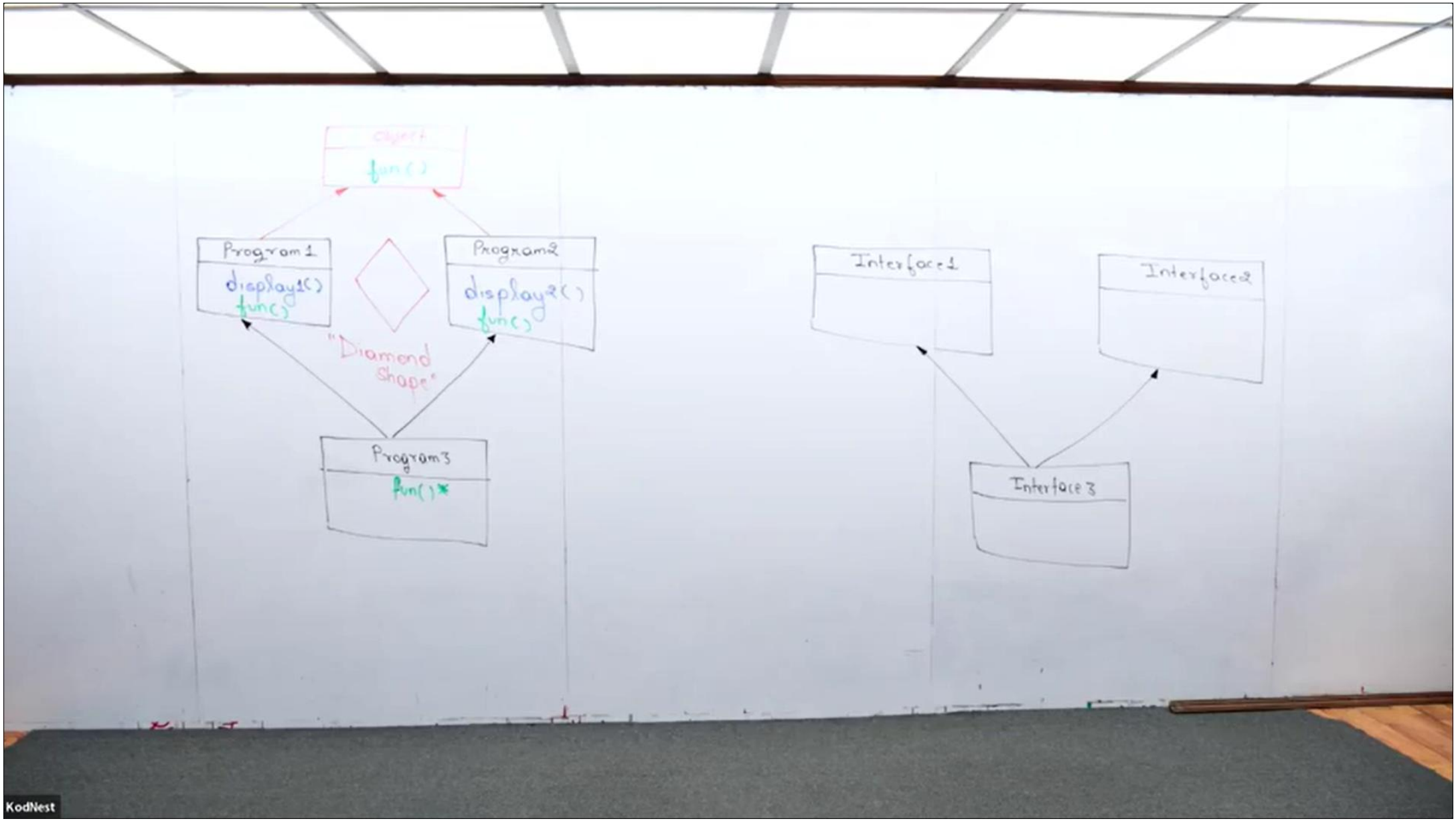
Hierarchical




```
class Program1
{
  ::
}
class Program2 extends Program1
{
  ::
}
class Program3 extends Program1
{
  ::
}
```


Developer
attendMeeting()
JavaDeveloper
learnJava()
javaProject()
class Developer
{
void attendMeeting()
{
sop("Developer is attending the meeting");
}
}
class JavaDeveloper extends Developer
{
void learnJava()
{
sop("Java developer is learning java");
}
void javaProject()
{
sop("Java developer is doing java project");
}
}
class DeveloperApp
{
p s v m (String[]args)
{
JavaDeveloper jdev = new JavaDeveloper();
jdev.learnJava();
jdev.javaProject();
jdev.attendMeeting();
}
}
KodNest


Program1
Program2
Program3
Program4
Single level Inheritance
Hierarchical Inheritance
Multilevel Inheritance
Hybrid Inheritance


fun()
Program 1
display1()
fun()
Program2
display2()
fun()
"Diamond Shape"
Program3
fun()
Interface1
Interface2
Interface3

Types of Inheritance

① Single level Inheritance :- Is such a Inheritance in which one parent class is inherited by one child class

ex:- As above

② multi level Inheritance :- Is such a Inheritance in which a class will be inherited by its child class and the child class will again inherited by its child class.

③ heirarchical Inheritance :- In which one parent class will be Inherited by multiple child class

ex.

④ Hyboid Inheritance :- which is a combination of two or more type of Inheritance.

5) Multiple Inheritance:- It is the type of Inheritance in which a child class will be Inheriting from multiple parent classes.

96.

In Java multiple Inheritance is not alowed using the Java classes because by default every class in Java will have the parent as object class hence it will lead to diamond shap problem due to which multiple Inhertance is not allowed using class..

But in Java muliple inheritence can be achieved using the interfaces because Interfaces do not have any default parent.

6. Cyclic Inheritance. In which child class will be Inheriting by the parent class and parent class also inheriting the child class

S7

This type of inheritance is not alowed in Java.

methods

(1) Inherited :- Jo as it is Inherited ho

(2) Overidden :- Inherite kro ne ke baad modified

(3) child specific :- ye vo method hai jo dayse child class mai bani jay.

(1). ~~which are~~

(2). These are such methods which will be inherited from parent class to child class, but child class will modify the body according to the requirment

(3) C.S :- which are only ~~paren~~ present only inside the child class and not present ~~in~~ inside the parent class.

S8.

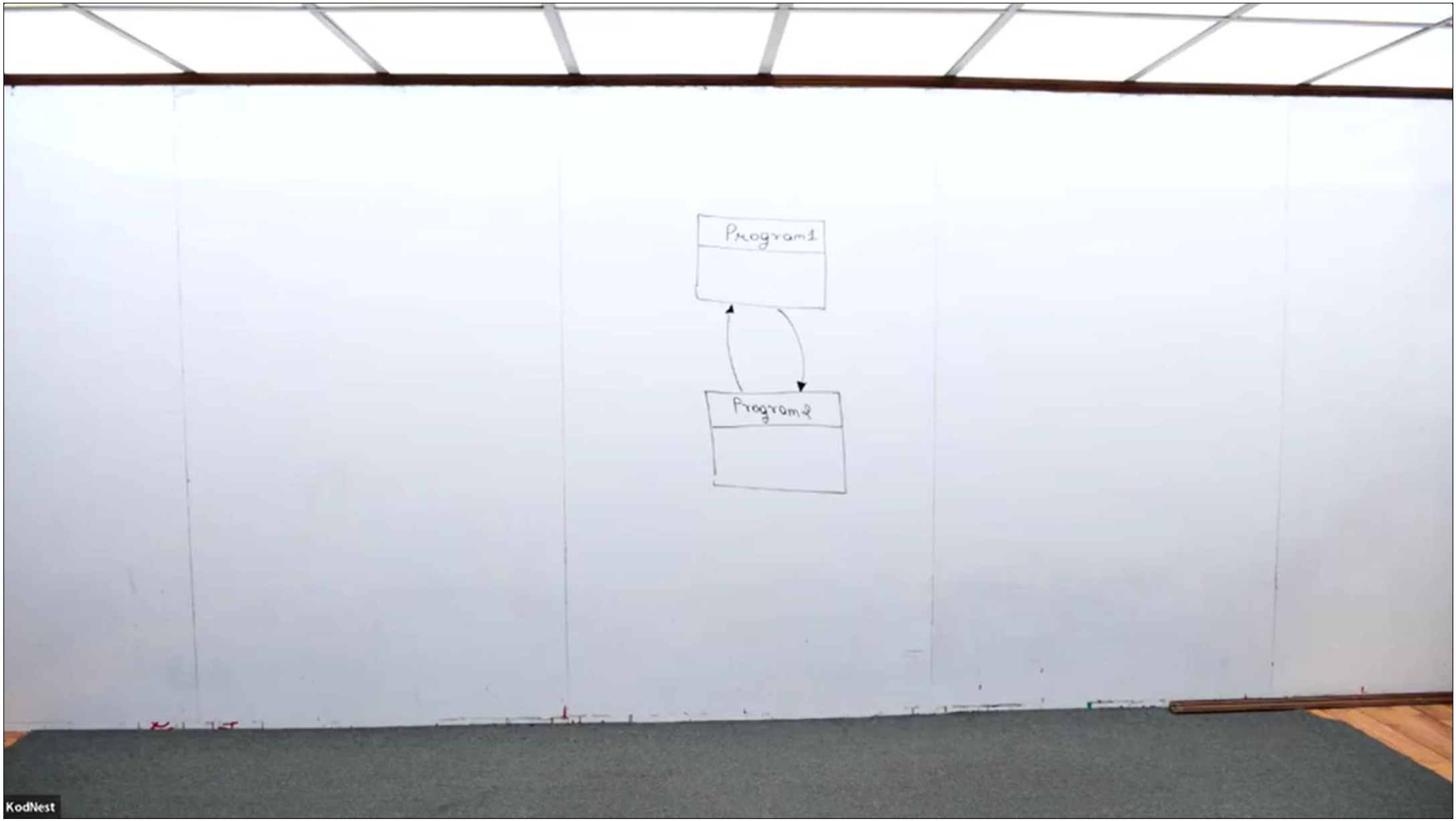
Constructor :- in

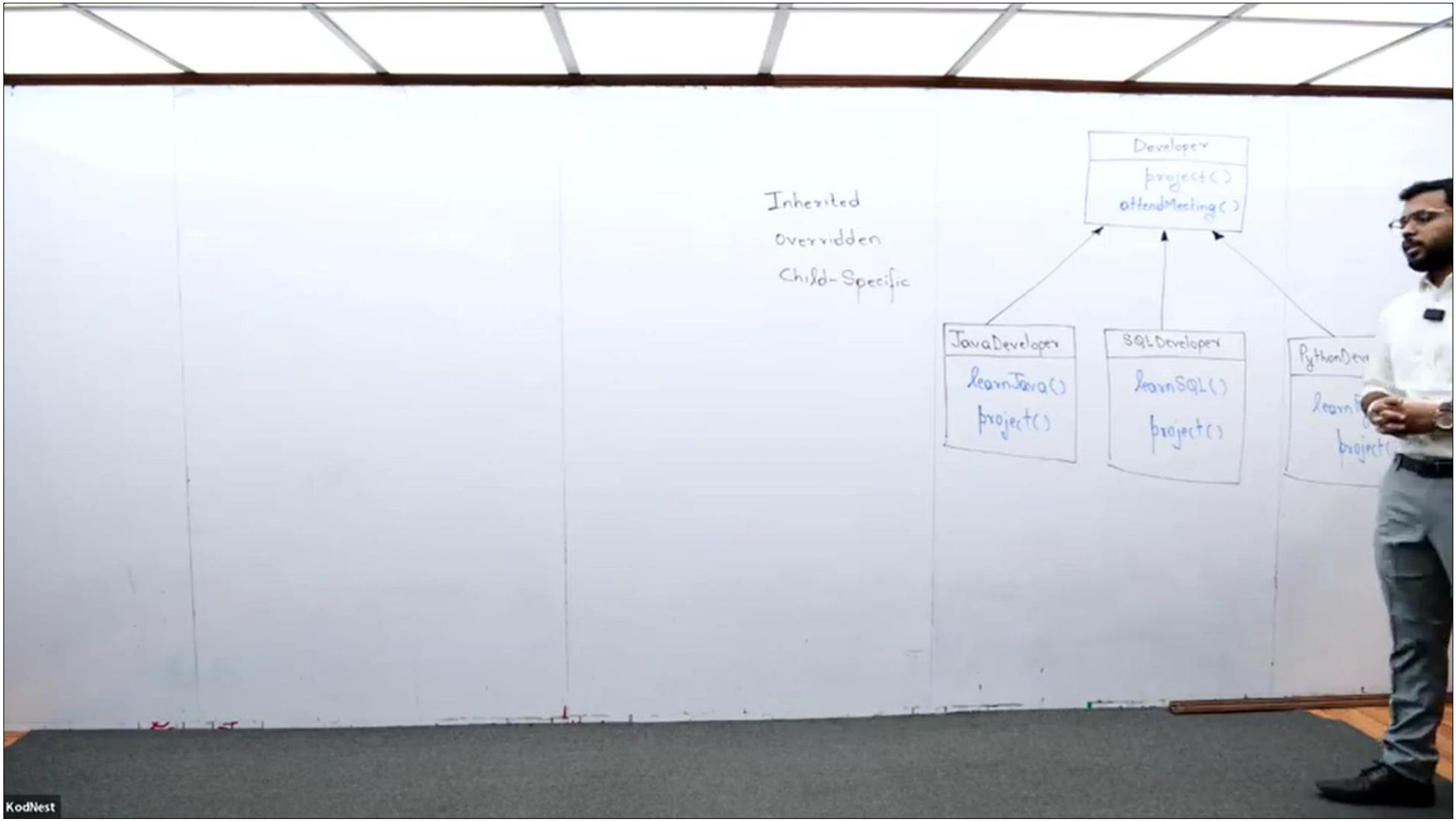
final keyword :- Can be used with class, method, variable.

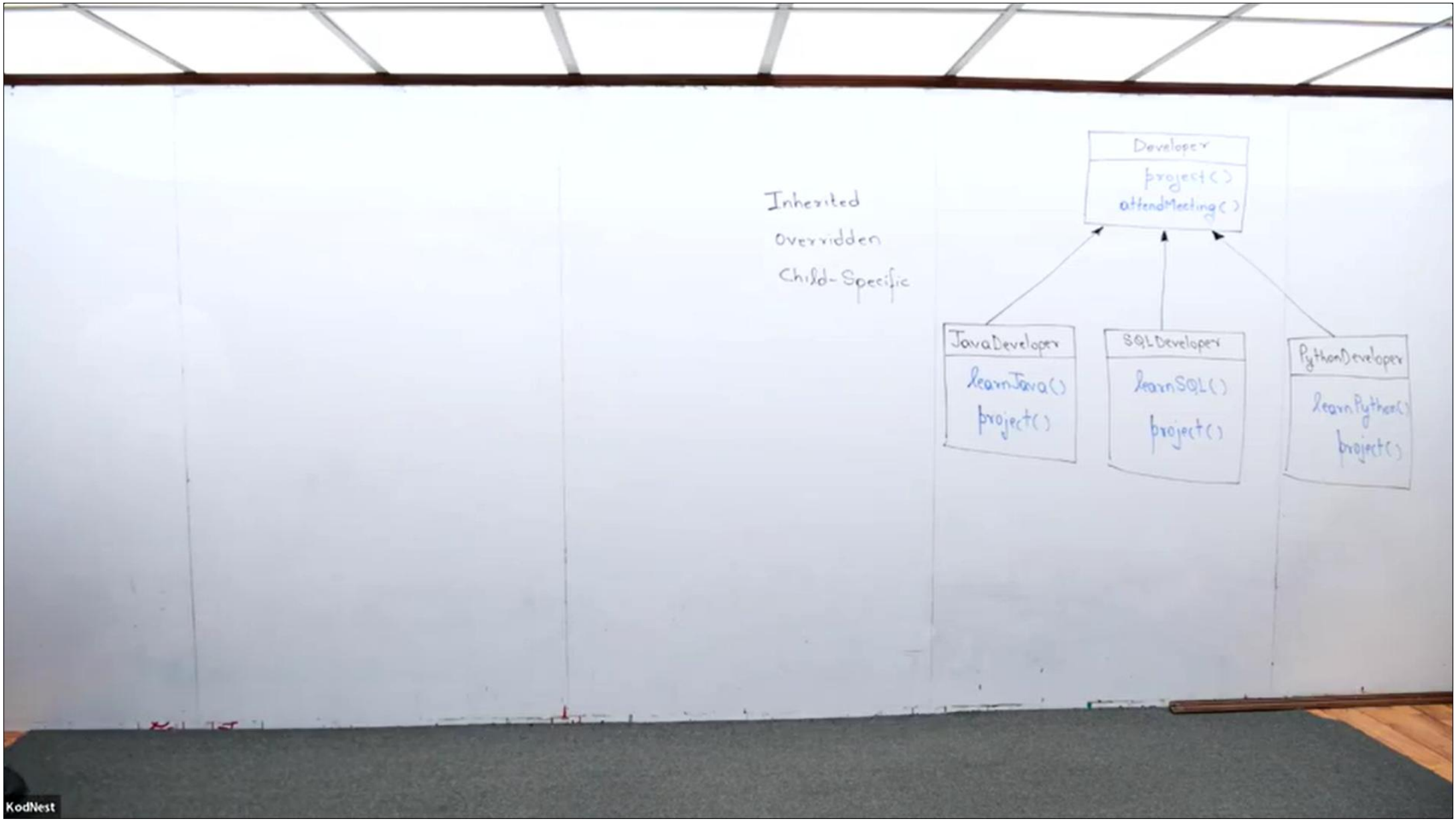
S9:33

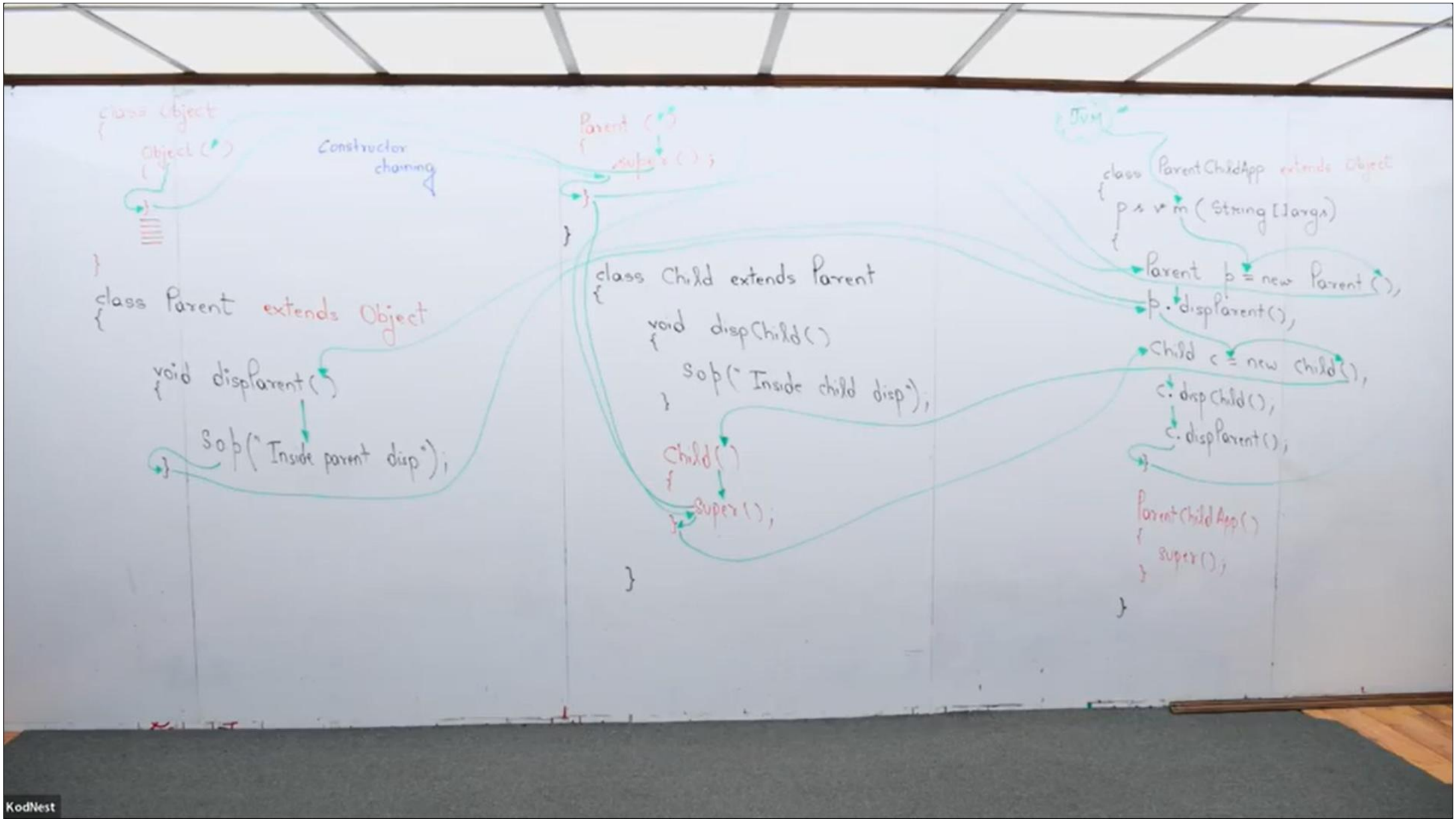
1) final keyword with class : is used to prevent Inheritance in other words. final classes can not be Inherited.

S9


Program1
Program2
KodNest


Inherited
Overridden
Child-Specific
Developer
project()
attendMeeting()
JavaDeveloper
learnJava()
project()
SQLDeveloper
learnSQL()
project()
KodNest


Inherited
Overridden
Child-Specific
Developer
project()
attendMeeting()
JavaDeveloper
learnJava()
project()
SQLDeveloper
learnSQL()
project()
PythonDeveloper
learnPython()
project()
KodNest


class Object
{
Object ()
{
}
}
Constructor chaining
class Parent extends Object
{
void displParent()
{
Sop("Inside parent disp");
}
}
Parent ()
{
super();
}
}
class Child extends Parent
{
void dispChild()
{
Sop("Inside child disp");
}
Child()
{
super();
}
}
JVM
class ParentChildApp extends Object
{
p s v m (String [] args)
{
Parent p = new Parent();
p.displParent();
Child c = new Child();
c.dispChild();
c.displParent();
}
ParentChildApp()
{
super();
}
}
KodNest

Access modifier:-

SSS

① Public:- Protected. private.

anywhere, diff. package we can access. we can not access in other class.

This variable:- removing shadowing problem.

upper variable:- accessing parent class variable

this method :- Constructor chaining in same class

supper method :- Constructor chaining in child class and parent class.

S12.

---

al keyword:- we can use with class, method and variable.

class with final keyword- Is used to prevent Inheri-tance (final class cannot be inherited)

S13.

final class P1 {

}

class P2 extends P1 { // error

(2) method :- final keyword with method is used to prevent method overriding (final methods cannot be overridden (child class can use it as it is)

S14;

(3) Variable :-

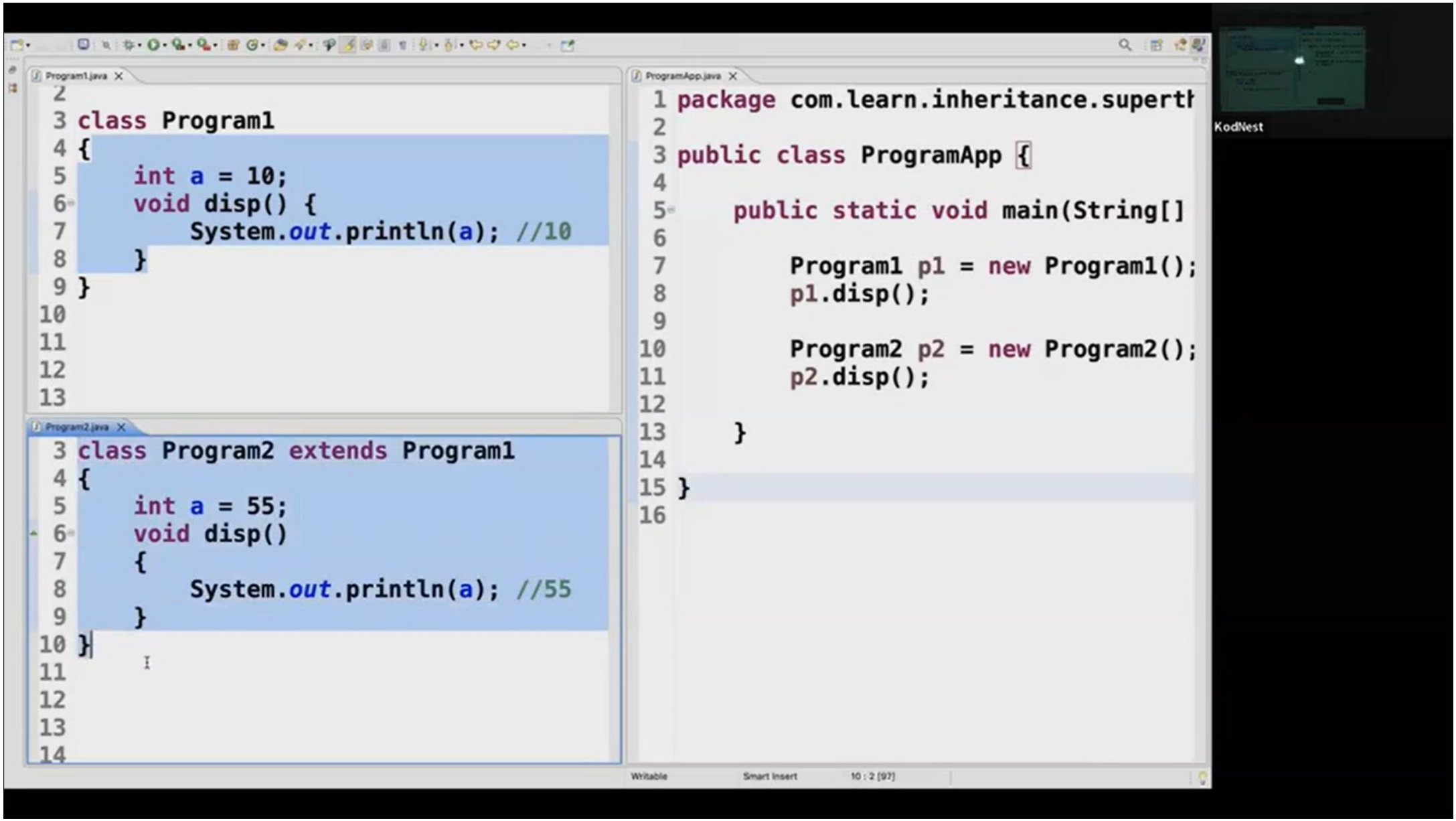
(1) The access modifier of parent class method and child class overriden method must be same if. It is different will get an error.

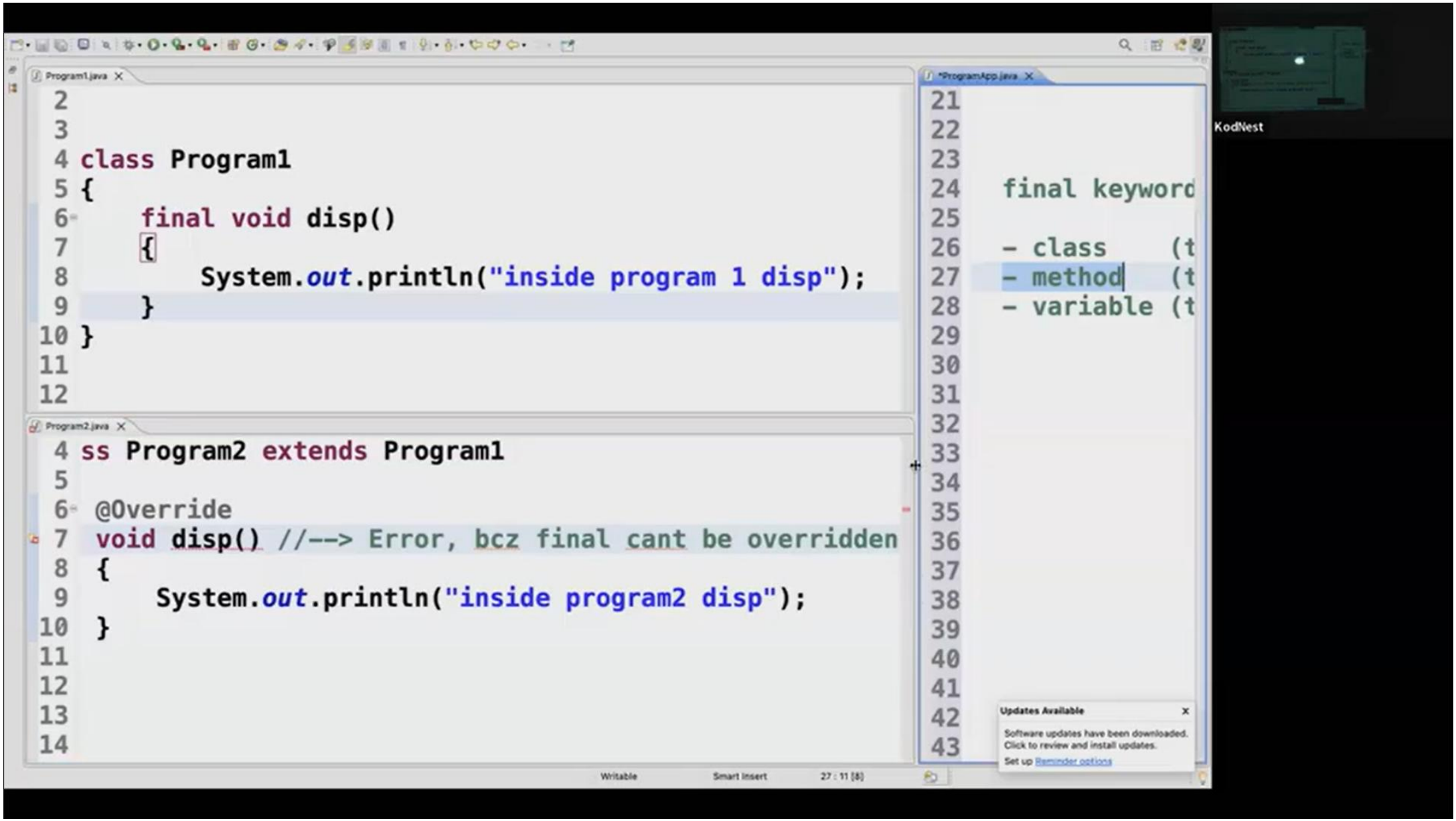
S15.

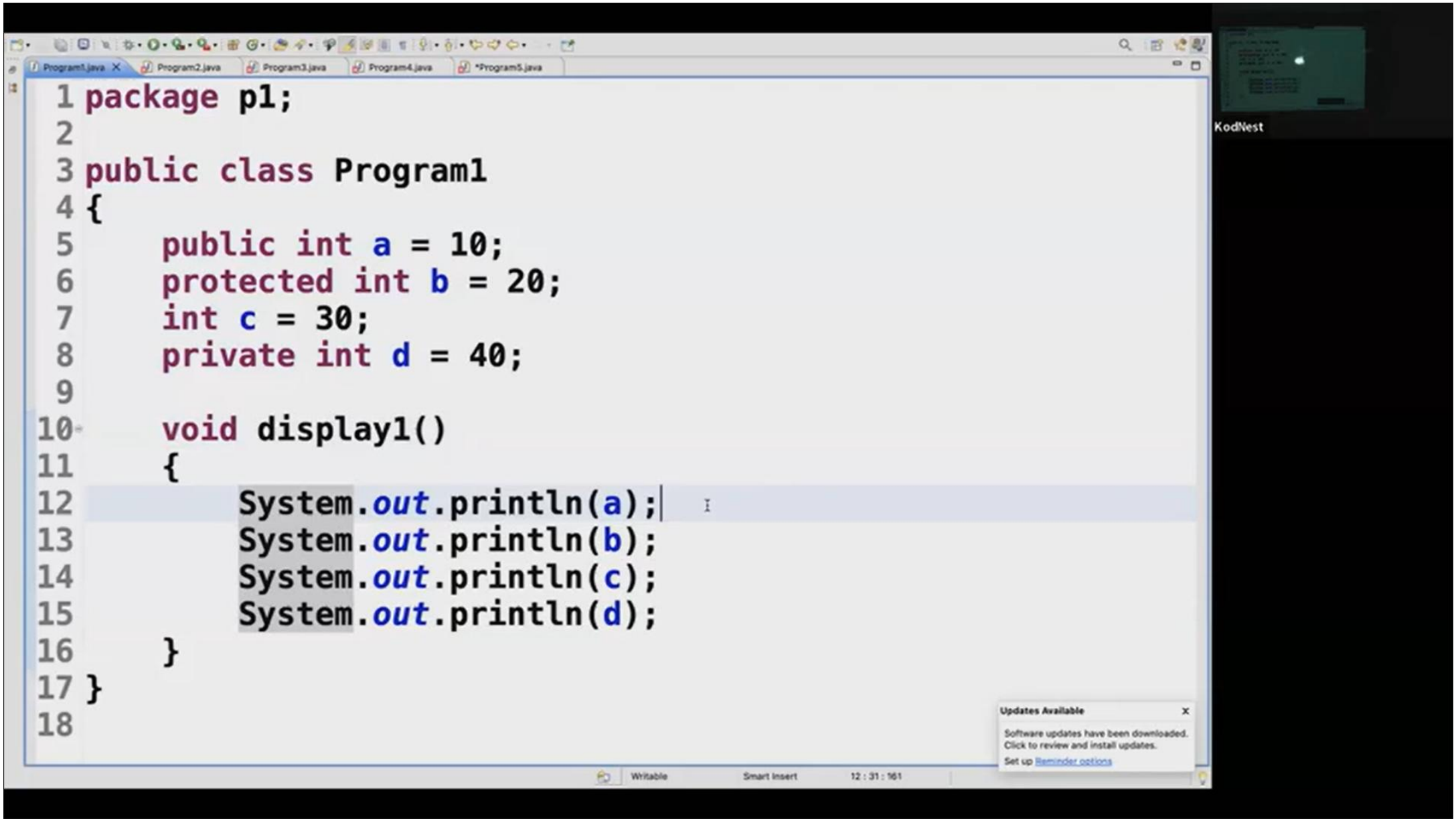
(2) The access modifier of parent class method and child class overriden method can be different if visibility and accessibility is increased, but if it is decreased then it is not allowed.

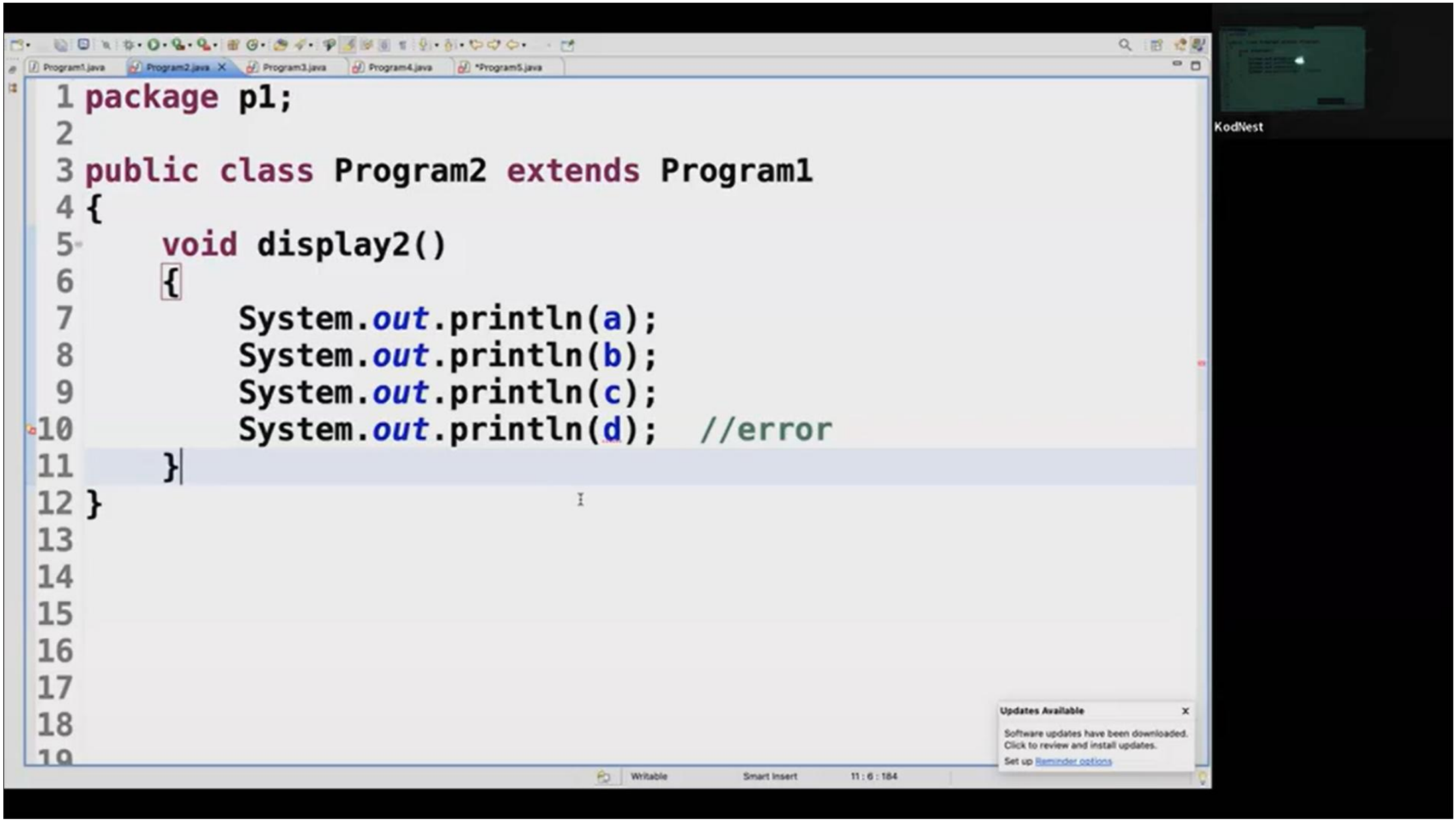
S16.

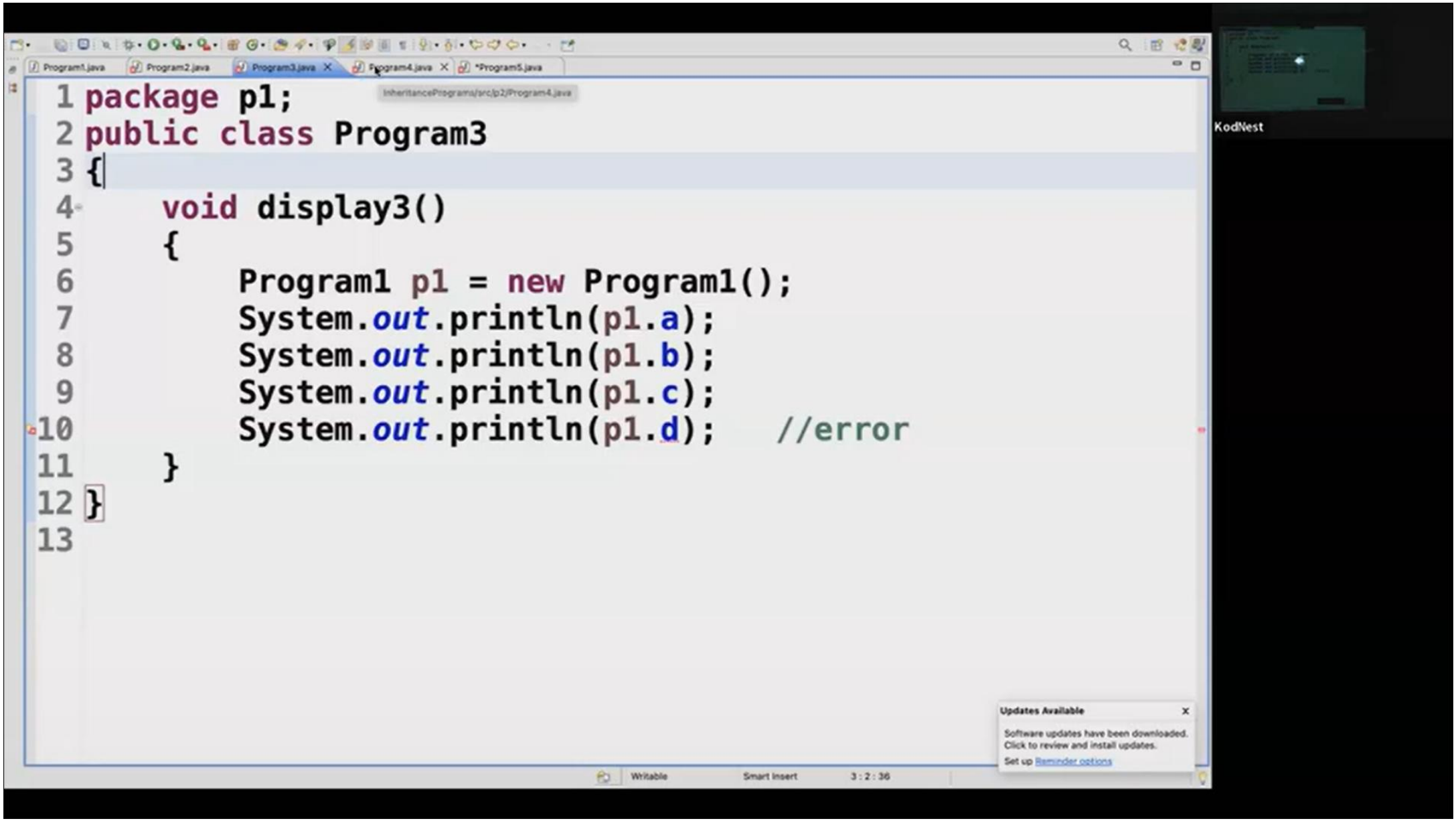
(3) Private never accessible.

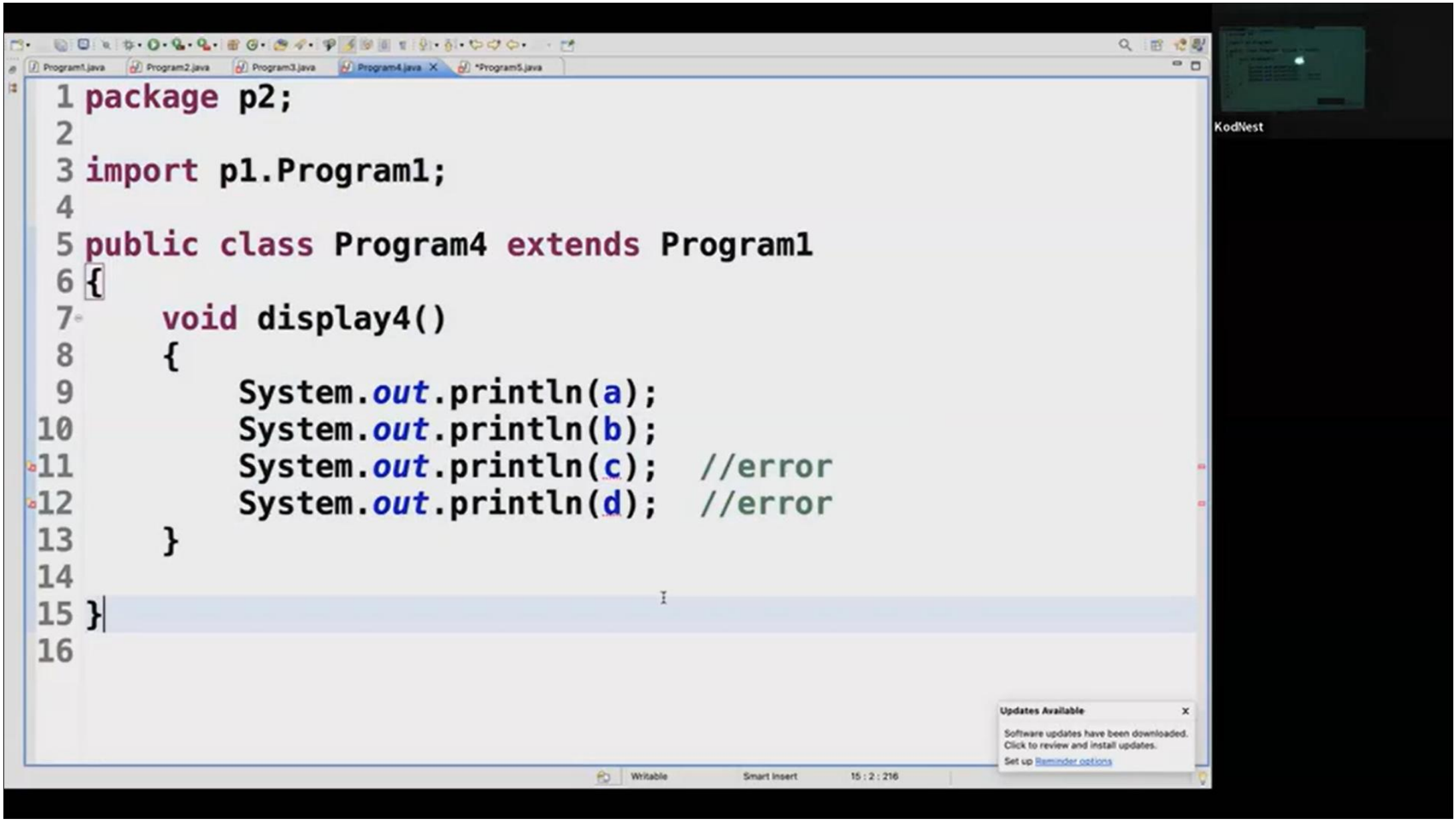

Program1.java
class Program1
{
int a = 10;
void disp() {
System.out.println(a); //10
}
}
Program2.java
class Program2 extends Program1
{
int a = 55;
void disp()
{
System.out.println(a); //55
}
}
ProgramApp.java
package com.learn.inheritance.superth
public class ProgramApp {
public static void main(String[]
Program1 p1 = new Program1();
p1.disp();
Program2 p2 = new Program2();
p2.disp();
}
}
Writable
Smart Insert
10 : 2 [97]
KodNest

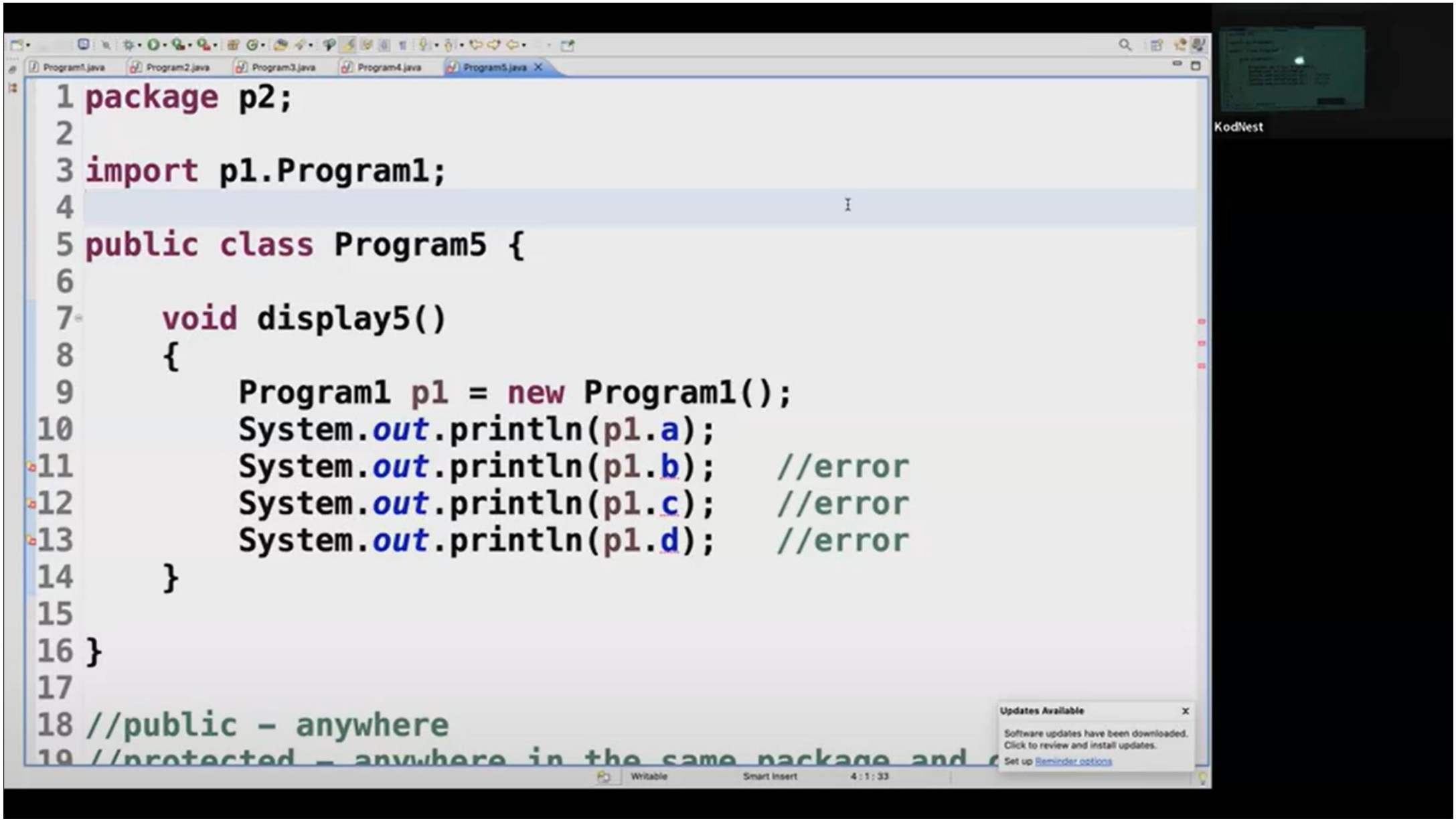

Program1.java
class Program1
{
final void disp()
{
System.out.println("inside program 1 disp");
}
}
Program2.java
ss Program2 extends Program1
@Override
void disp() //--> Error, bcz final cant be overridden
{
System.out.println("inside program2 disp");
}
*ProgramApp.java
final keyword
- class (t
- method (t
- variable (t
Updates Available
Software updates have been downloaded.
Click to review and install updates.
Set up Reminder options
Writable
Smart Insert
27 : 11 [8]
KodNest

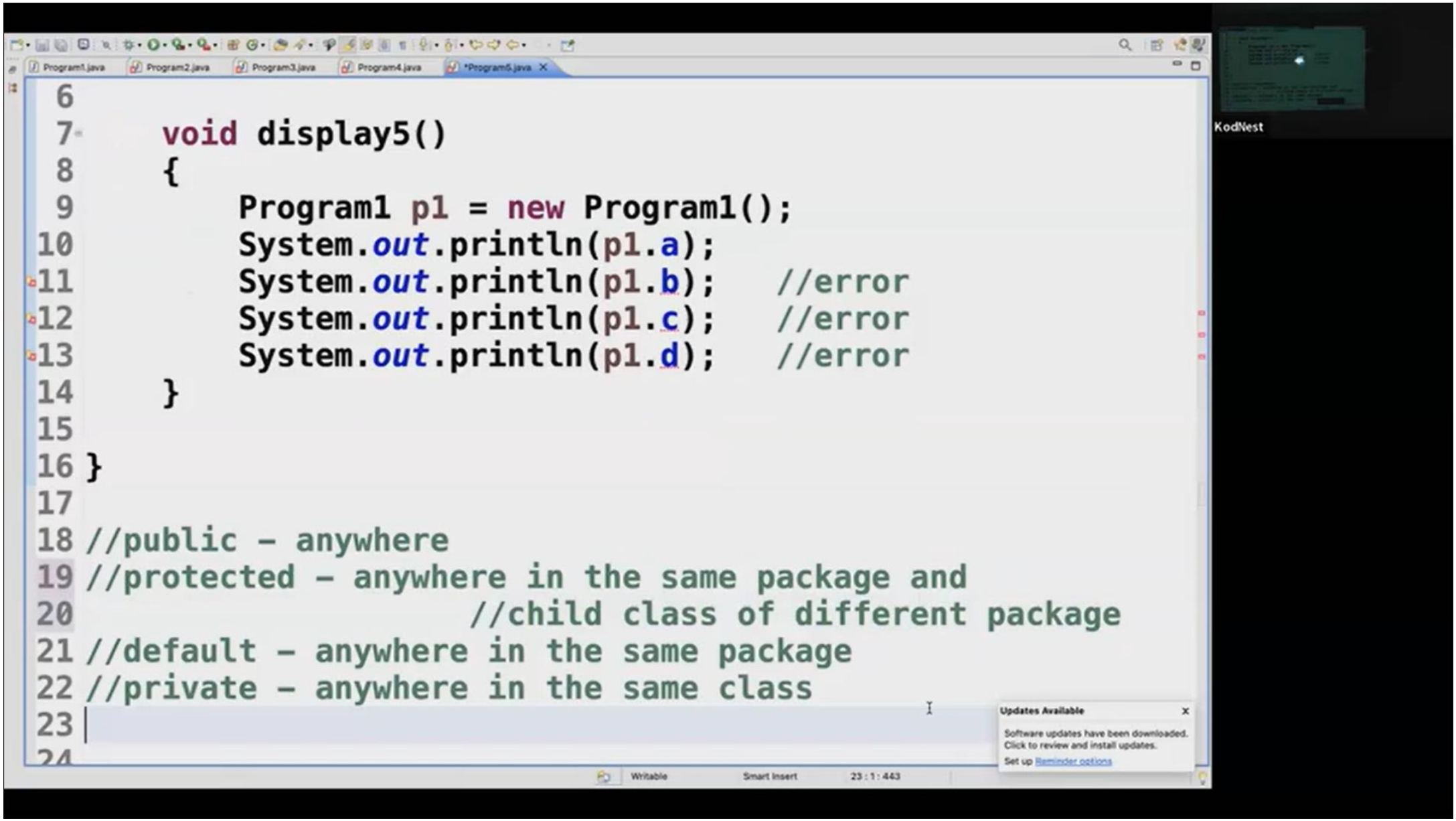

Program1.java
Program2.java
Program3.java
Program4.java
*Program5.java
1 package p1;
2
3 public class Program1
4 {
5 public int a = 10;
6 protected int b = 20;
7 int c = 30;
8 private int d = 40;
9
10 void display1()
11 {
12 System.out.println(a);
13 System.out.println(b);
14 System.out.println(c);
15 System.out.println(d);
16 }
17 }
18
KodNest
Updates Available
Software updates have been downloaded.
Click to review and install updates.
Set up Reminder options
Writable
Smart Insert
12 : 31 : 161

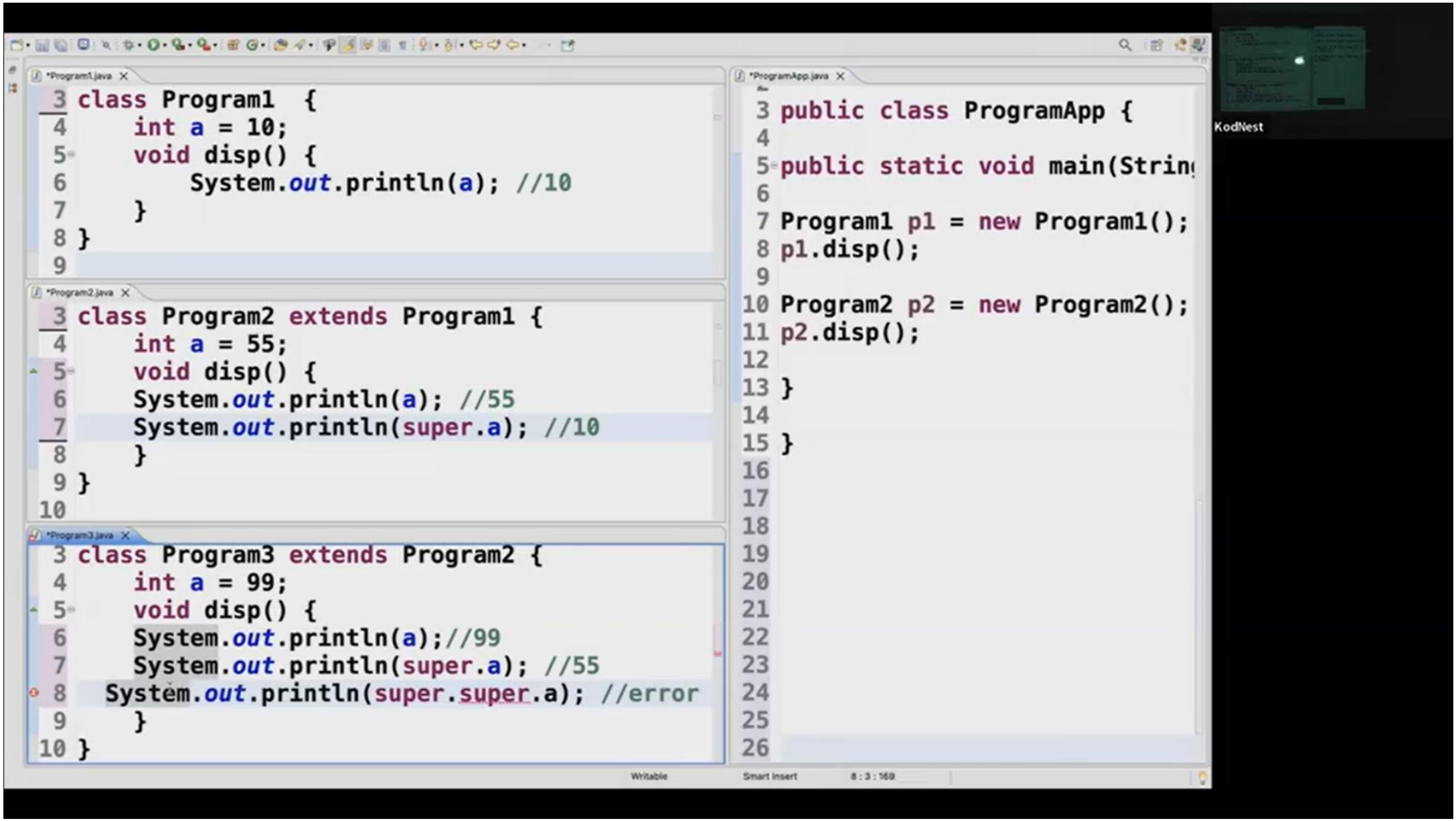

Program1.java
Program2.java
Program3.java
Program4.java
*Program5.java
1 package p1;
2
3 public class Program2 extends Program1
4 {
5 void display2()
6 {
7 System.out.println(a);
8 System.out.println(b);
9 System.out.println(c);
10 System.out.println(d); //error
11 }
12 }
KodNest
Updates Available
Software updates have been downloaded.
Click to review and install updates.
Set up Reminder options
Writable
Smart Insert
11 : 6 : 184

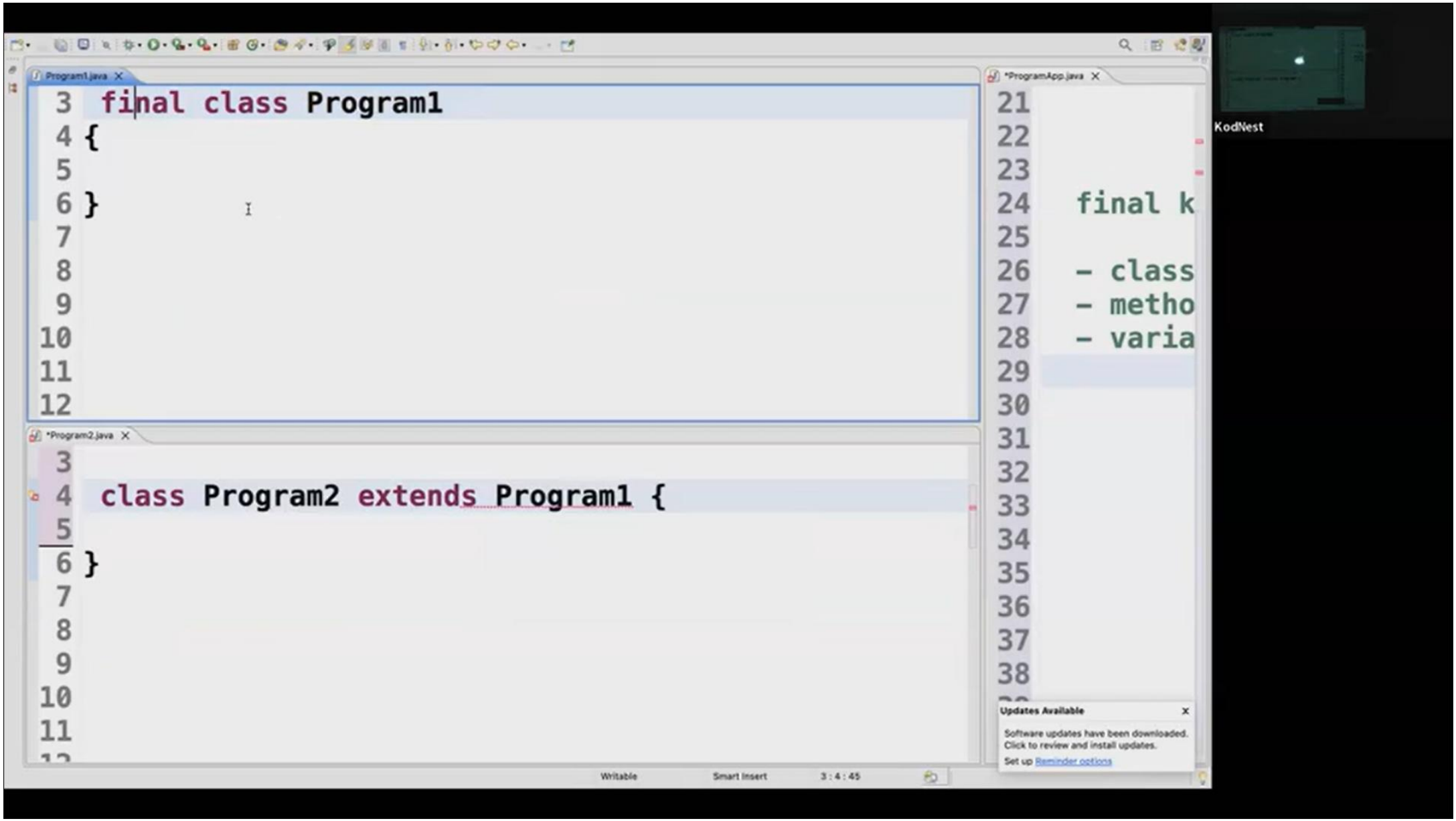

Program1.java
Program2.java
Program3.java
Program4.java
*Program5.java
InheritancePrograms/src/p2/Program4.java
1 package p1;
2 public class Program3
3 {
4 void display3()
5 {
6 Program1 p1 = new Program1();
7 System.out.println(p1.a);
8 System.out.println(p1.b);
9 System.out.println(p1.c);
10 System.out.println(p1.d); //error
11 }
12 }
13
KodNest
Updates Available
Software updates have been downloaded.
Click to review and install updates.
Set up Reminder options
Writable
Smart Insert
3 : 2 : 36

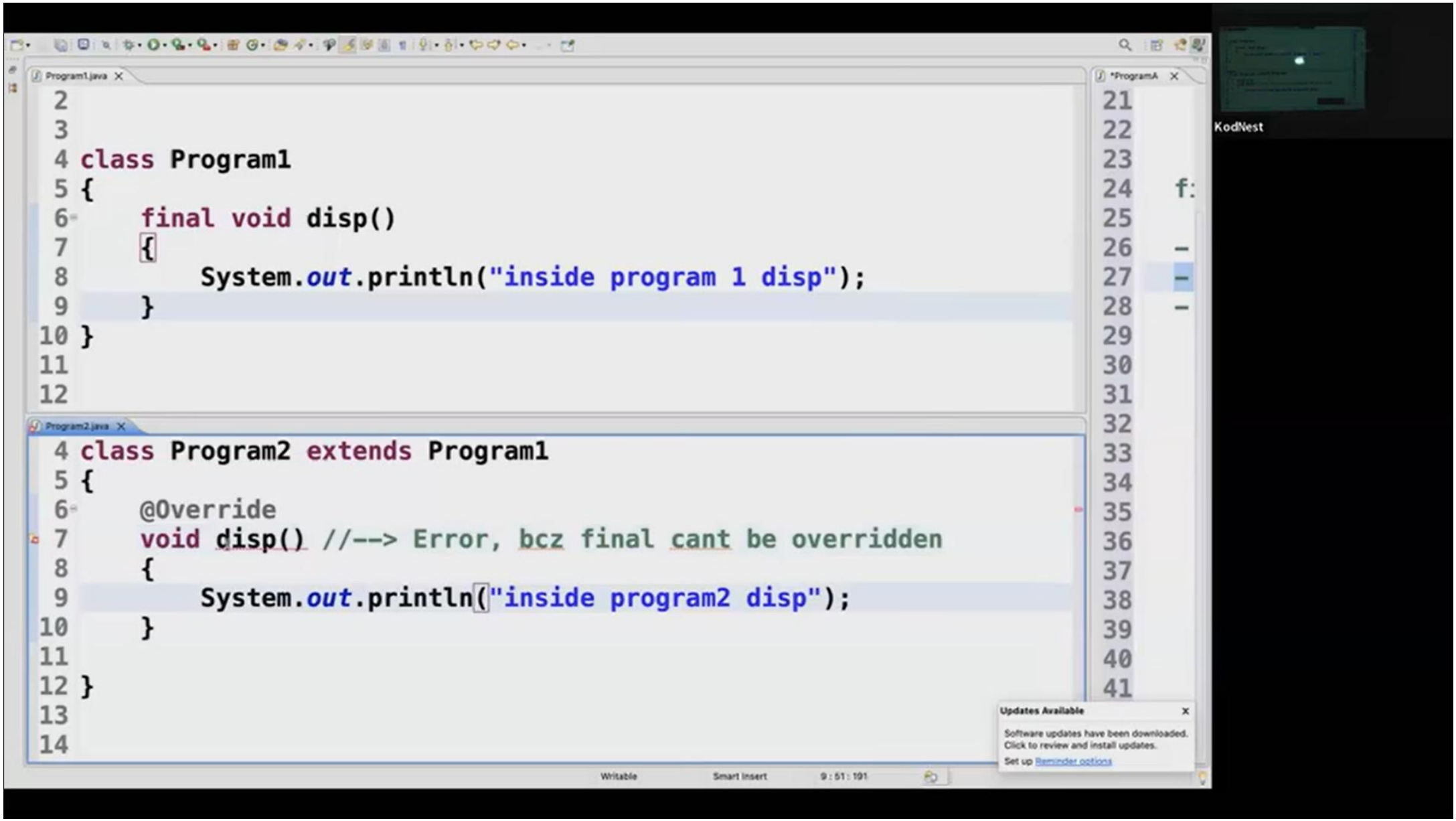

Program1.java
Program2.java
Program3.java
Program4.java
*Program5.java
1 package p2;
2
3 import p1.Program1;
4
5 public class Program4 extends Program1
6 {
7 void display4()
8 {
9 System.out.println(a);
10 System.out.println(b);
11 System.out.println(c); //error
12 System.out.println(d); //error
13 }
14
15 }
16
KodNest
Updates Available
Software updates have been downloaded.
Click to review and install updates.
Set up Reminder options
Writable
Smart Insert
15 : 2 : 216

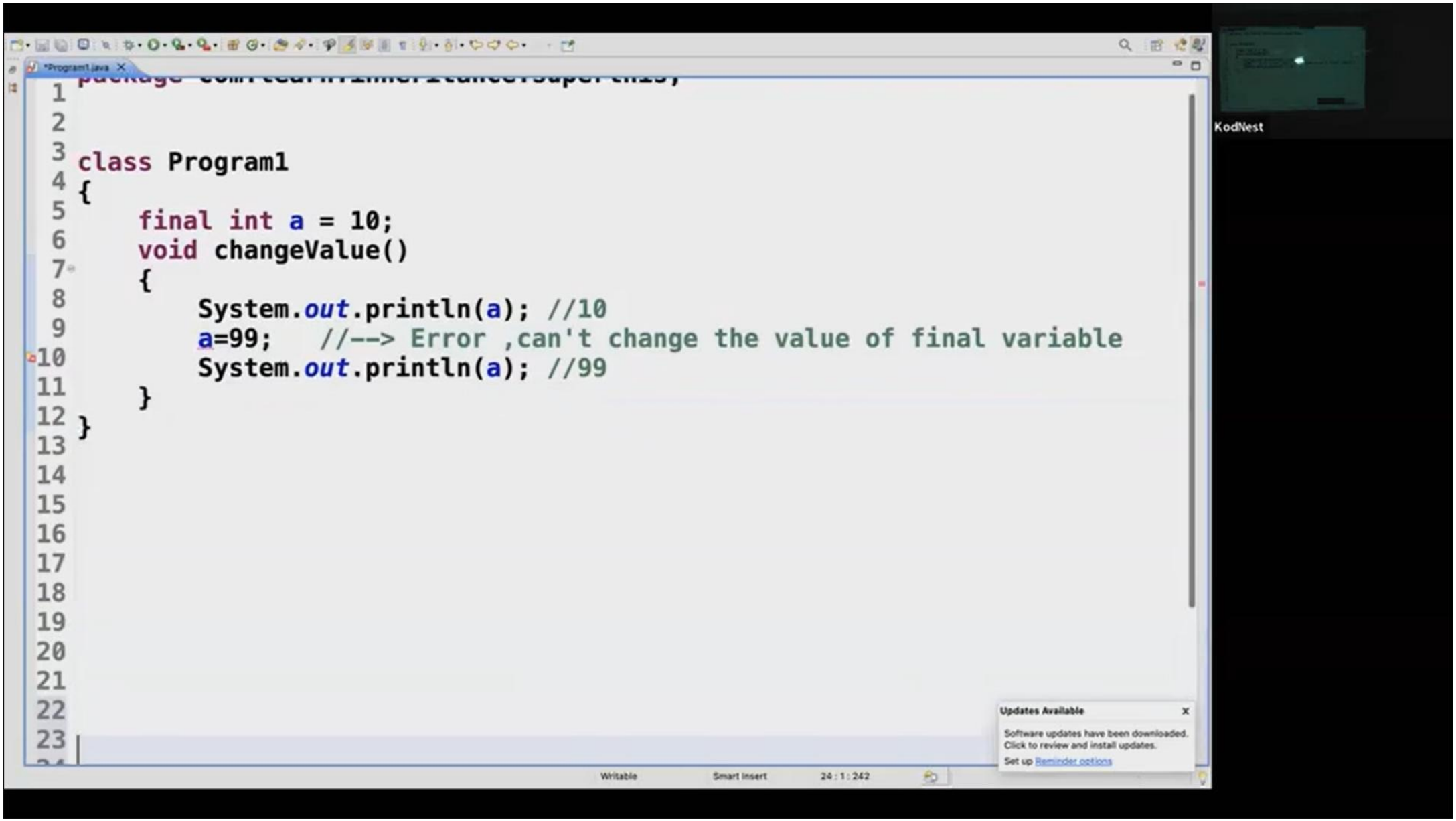

Program1.java
Program2.java
Program3.java
Program4.java
Program5.java
1 package p2;
2
3 import p1.Program1;
4
5 public class Program5 {
6
7 void display5()
8 {
9 Program1 p1 = new Program1();
10 System.out.println(p1.a);
11 System.out.println(p1.b); //error
12 System.out.println(p1.c); //error
13 System.out.println(p1.d); //error
14 }
15
16 }
17
18 //public - anywhere
Updates Available
Software updates have been downloaded.
Click to review and install updates.
Set up Reminder options
Writable
Smart Insert
4 : 1 : 33
KodNest

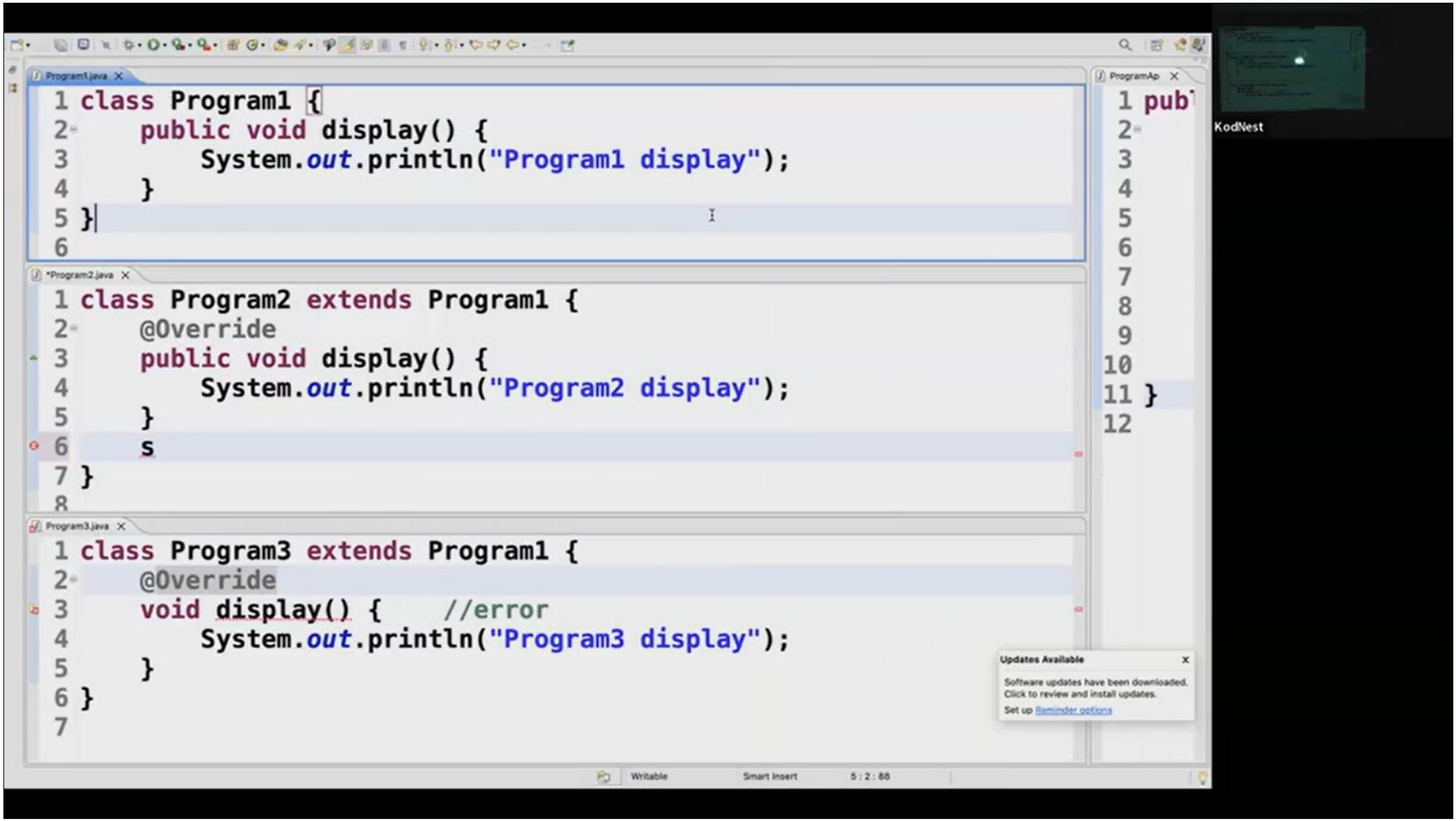

Program1.java
Program2.java
Program3.java
Program4.java
*Program5.java
KodNest
6
7 void display5()
8 {
9 Program1 p1 = new Program1();
10 System.out.println(p1.a);
11 System.out.println(p1.b); //error
12 System.out.println(p1.c); //error
13 System.out.println(p1.d); //error
14 }
15
16 }
17
18 //public - anywhere
19 //protected - anywhere in the same package and
20 //child class of different package
21 //default - anywhere in the same package
22 //private - anywhere in the same class
23
Updates Available
Software updates have been downloaded.
Click to review and install updates.
Set up Reminder options
Writable
Smart Insert
23 : 1 : 443

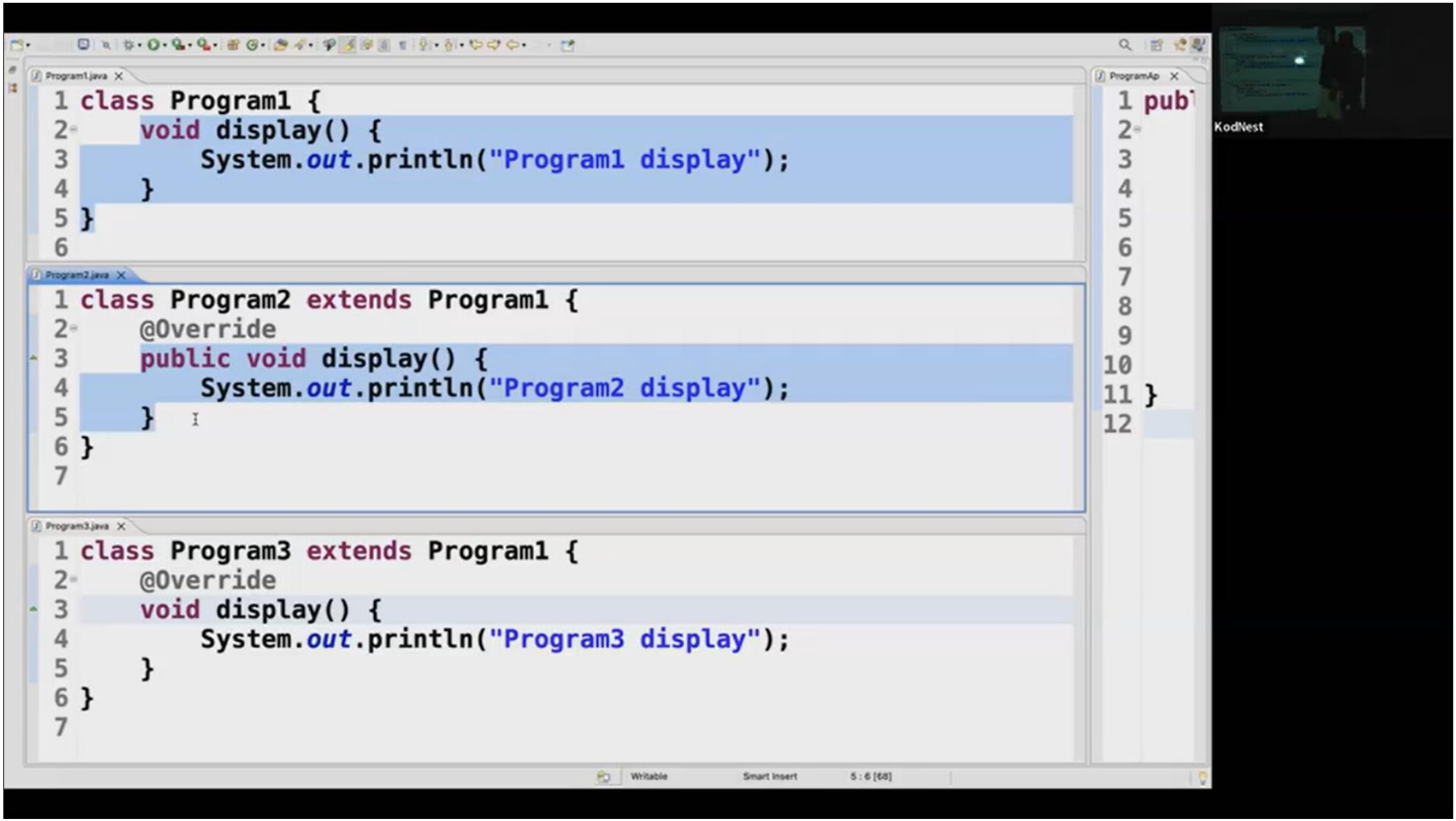

*Program1.java
class Program1 {
int a = 10;
void disp() {
System.out.println(a); //10
}
}
*Program2.java
class Program2 extends Program1 {
int a = 55;
void disp() {
System.out.println(a); //55
System.out.println(super.a); //10
}
}
*Program3.java
class Program3 extends Program2 {
int a = 99;
void disp() {
System.out.println(a);//99
System.out.println(super.a); //55
System.out.println(super.super.a); //error
}
}
*ProgramApp.java
public class ProgramApp {
public static void main(Strin
Program1 p1 = new Program1();
p1.disp();
Program2 p2 = new Program2();
p2.disp();
}
}
KodNest
Writable
Smart Insert
8 : 3 : 169

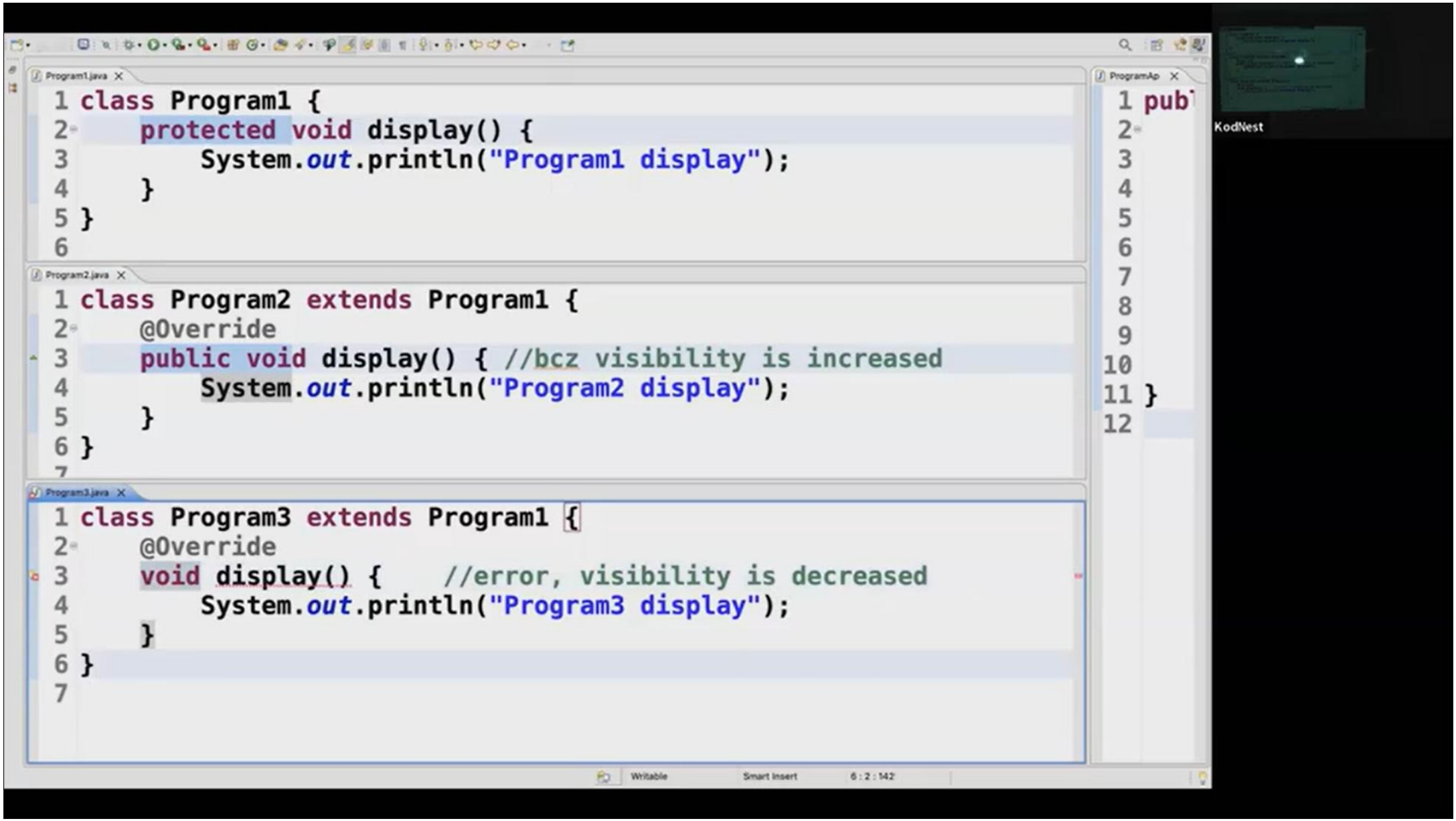

Program1.java
final class Program1
{
}
*Program2.java
class Program2 extends Program1 {
}
*ProgramApp.java
final k
- class
- metho
- varia
Updates Available
Software updates have been downloaded.
Click to review and install updates.
Set up Reminder options
Writable
Smart Insert
3 : 4 : 45
KodNest

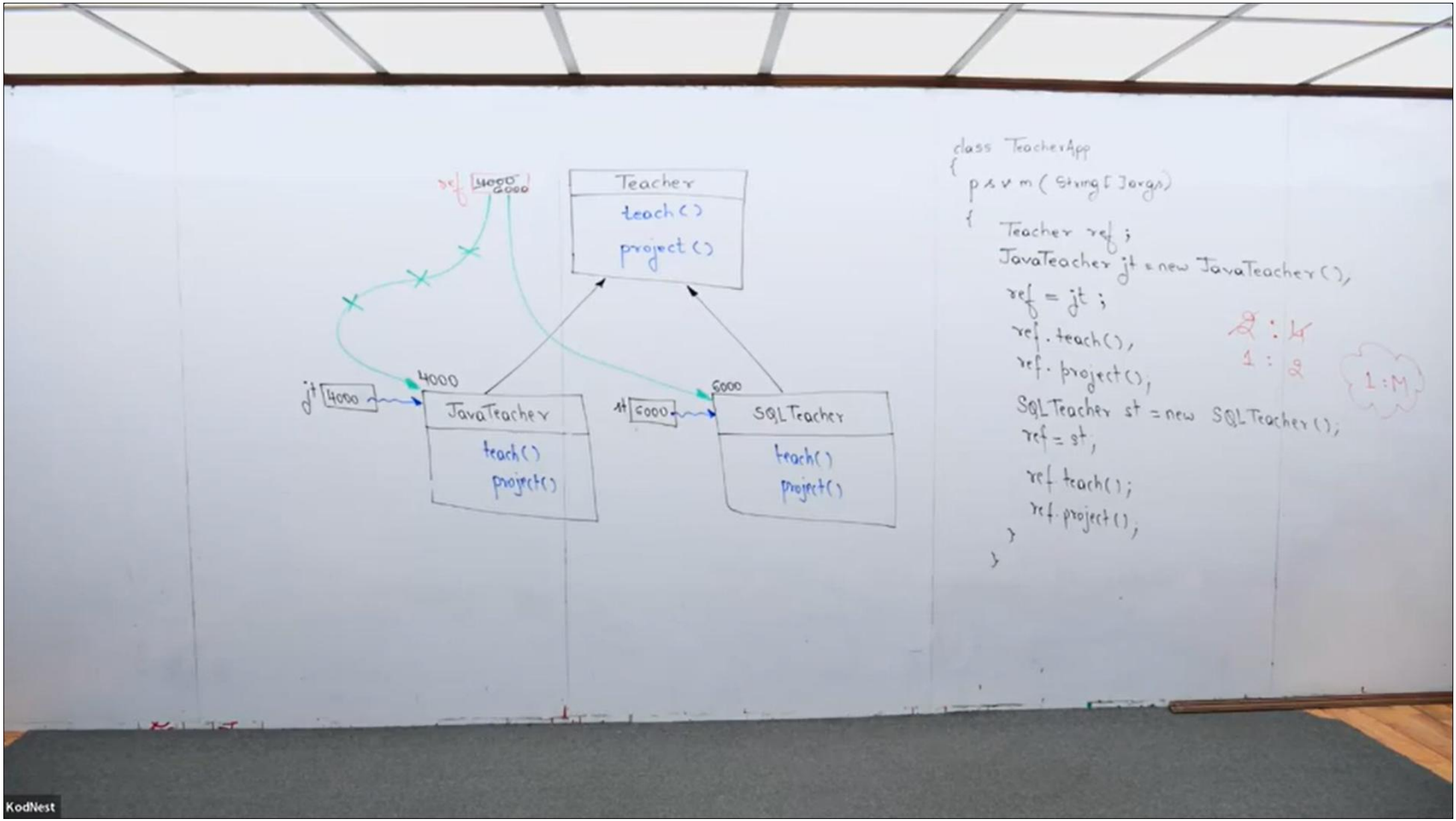

Program1.java
class Program1
{
final void disp()
{
System.out.println("inside program 1 disp");
}
}
Program2.java
class Program2 extends Program1
{
@Override
void disp() //--> Error, bcz final cant be overridden
{
System.out.println("inside program2 disp");
}
}
*ProgramA
f:
KodNest
Updates Available
Software updates have been downloaded.
Click to review and install updates.
Set up Reminder options
Writable
Smart Insert
9 : 51 : 191


*Program1.java
1
2
3 class Program1
4 {
5 final int a = 10;
6 void changeValue()
7 {
8 System.out.println(a); //10
9 a=99; //--> Error ,can't change the value of final variable
10 System.out.println(a); //99
11 }
12 }
13
14
15
16
17
18
19
20
21
22
23
KodNest
Updates Available
Software updates have been downloaded.
Click to review and install updates.
Set up Reminder options
Writable
Smart Insert
24 : 1 : 242


Program1.java
class Program1 {
public void display() {
System.out.println("Program1 display");
}
}
*Program2.java
class Program2 extends Program1 {
@Override
public void display() {
System.out.println("Program2 display");
}
s
}
Program3.java
class Program3 extends Program1 {
@Override
void display() { //error
System.out.println("Program3 display");
}
}
ProgramAp
pub
}
KodNest
Updates Available
Software updates have been downloaded. Click to review and install updates.
Set up Reminder options
Writable
Smart Insert
5 : 2 : 86


Program1.java
class Program1 {
void display() {
System.out.println("Program1 display");
}
}
Program2.java
class Program2 extends Program1 {
@Override
public void display() {
System.out.println("Program2 display");
}
}
Program3.java
class Program3 extends Program1 {
@Override
void display() {
System.out.println("Program3 display");
}
}
ProgramAp
pub
}
KodNest
Writable
Smart Insert
5 : 6 [68]


Program1.java
class Program1 {
protected void display() {
System.out.println("Program1 display");
}
}
Program2.java
class Program2 extends Program1 {
@Override
public void display() { //bcz visibility is increased
System.out.println("Program2 display");
}
}
Program3.java
class Program3 extends Program1 {
@Override
void display() { //error, visibility is decreased
System.out.println("Program3 display");
}
}
ProgramAp
pub
}
KodNest
Writable
Smart Insert
6 : 2 : 142


ref 4000 6000
Teacher
teach()
project()
jt 4000
4000
JavaTeacher
teach()
project()
st 6000
6000
SQLTeacher
teach()
project()
class TeacherApp
{
psvm(String[]args)
{
Teacher ref;
JavaTeacher jt = new JavaTeacher();
ref = jt;
ref.teach();
ref.project();
SQLTeacher st = new SQLTeacher();
ref = st;
ref.teach();
ref.project();
}
}
2 : 4
1 : 2
1 : M