www.kodnest.com

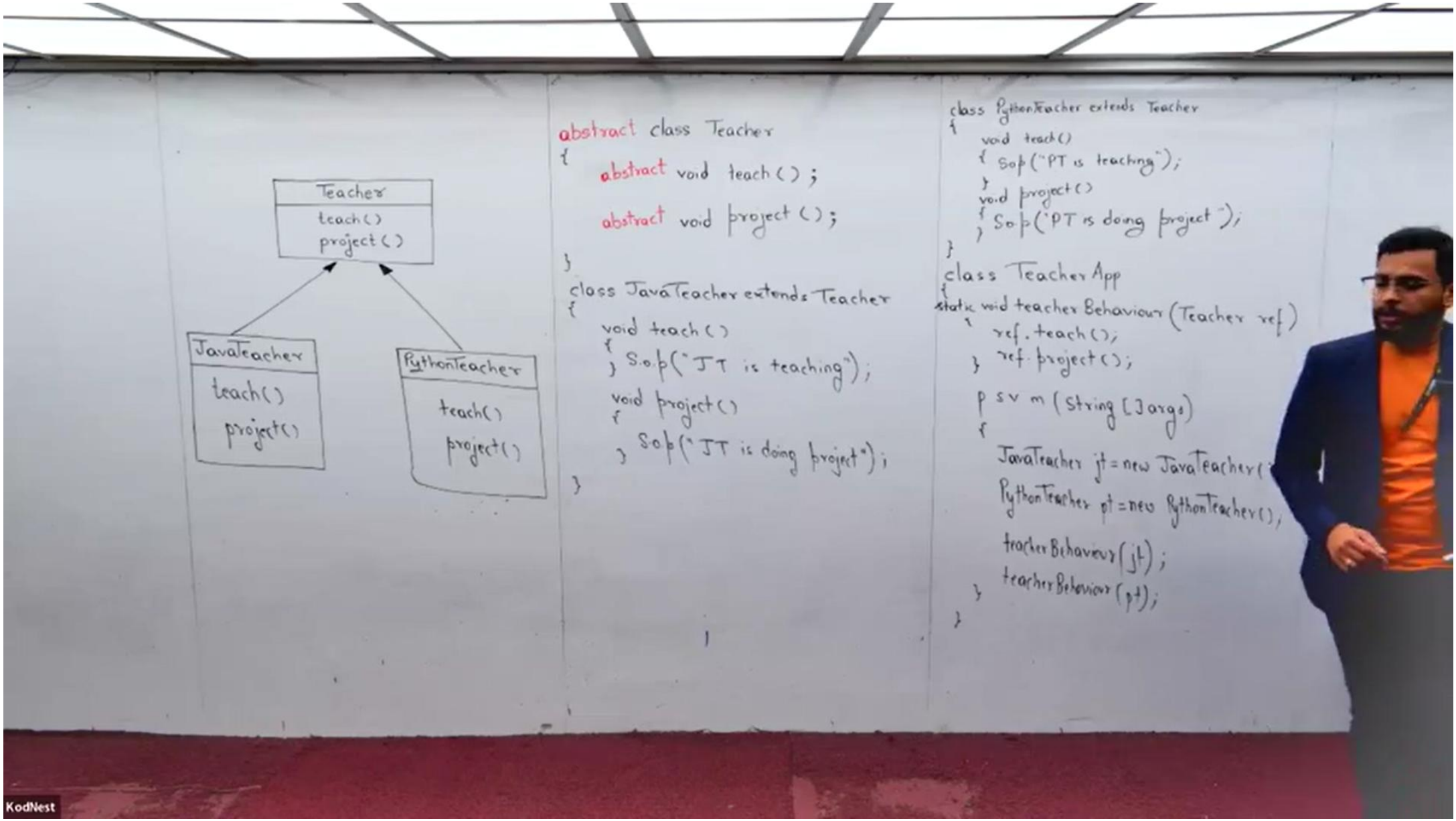
1-Abstraction :-

It is the process of hidding irrelevant d
It can be achieved using abstract classes or interfa

0 Abstraction using abstract classes..

S1

abstract keyword is used for creating an ab


Teacher
teach()
project()
JavaTeacher
teach()
project()
PythonTeacher
teach()
project()
abstract class Teacher
{
abstract void teach();
abstract void project();
}
class JavaTeacher extends Teacher
{
void teach()
{
} S.o.p("JT is teaching");
void project()
{
} Sop("JT is doing project");
}
class PythonTeacher extends Teacher
{
void teach()
{ Sop("PT is teaching");
}
void project()
{ Sop("PT is doing project");
}
}
class TeacherApp
{
static void teacherBehaviour(Teacher ref)
{
ref.teach();
} ref.project();
p s v m (String[]args)
{
JavaTeacher jt=new JavaTeacher(
PythonTeacher pt=new PythonTeacher();
teacherBehaviour(jt);
} teacherBehaviour(pt);
}
KodNest

It is the process of hiding
It can be achieved using abstract classes or interfaces

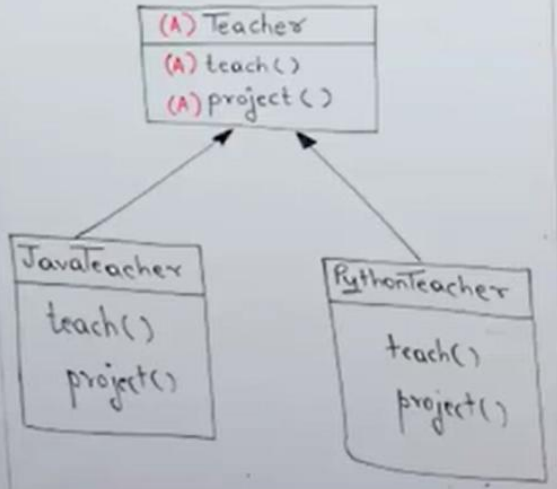
Abstraction using abstract classes.

S1

abstract keyword is used for creating an abstract class and an abstract method.

abstract methods are such methods which will have the method body. also, there is a rule in Java that if a class consist of any abstract then the class also must be declared as abstract class.

S2




```
abstract class Teacher
{
    abstract void teach();

    abstract void project();

}
class JavaTeacher extends Teacher
{
    void teach()
    {
    } S.o.p("JT is teaching");
    void project()
    {
    } S.o.p("JT is doing project");
}
```

```
class PythonTeacher extends Teacher
{
    void teach()
    { S.o.p("PT is teaching");
    }
    void project()
    { S.o.p("PT is doing project");
    }
}
class TeacherApp
{
static void teacherBehaviour(Teacher ref)
    {
      ref.teach();
    } ref.project();

    p s v m (String[]args)
    {
      JavaTeacher jt = new JavaTeacher();
      PythonTeacher pt = new PythonTeacher();

      teacherBehaviour(jt);
    } teacherBehaviour(pt);
}
```

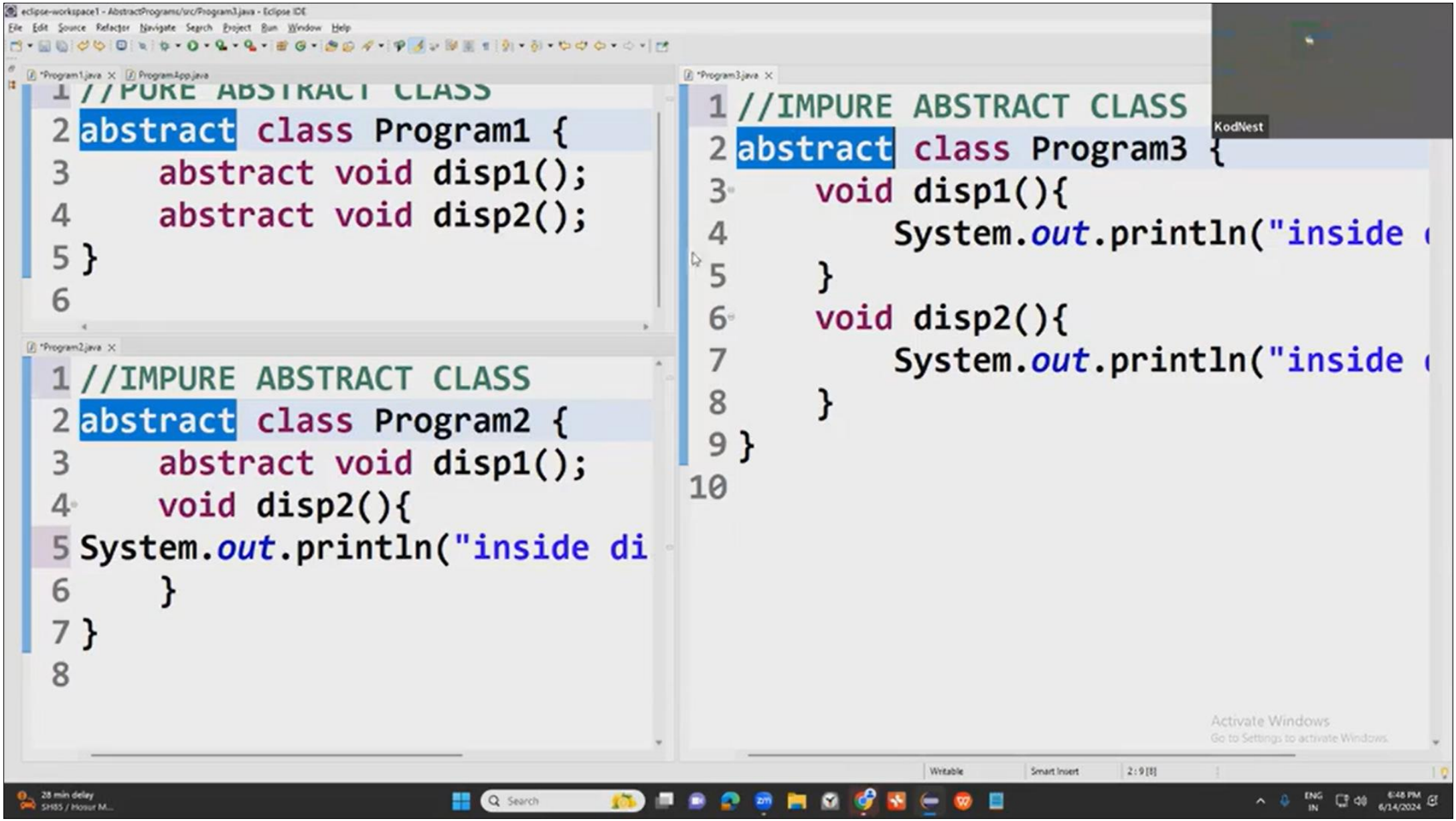
· Java that if a class consist of any abstract meth...
then. the class also must be decibared as abstract
class.

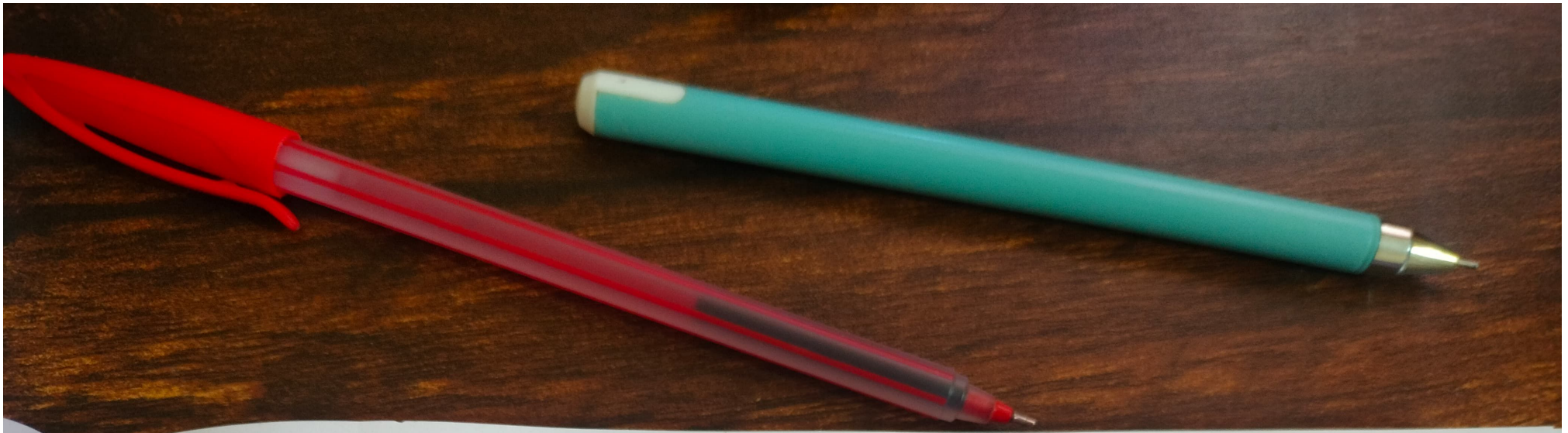
S2

abstract classes can consist of only. abstract.
, combination of abstract and concreat and.
only Concolte meths.

what is an abstract class:- for which we can't
creat the object:.

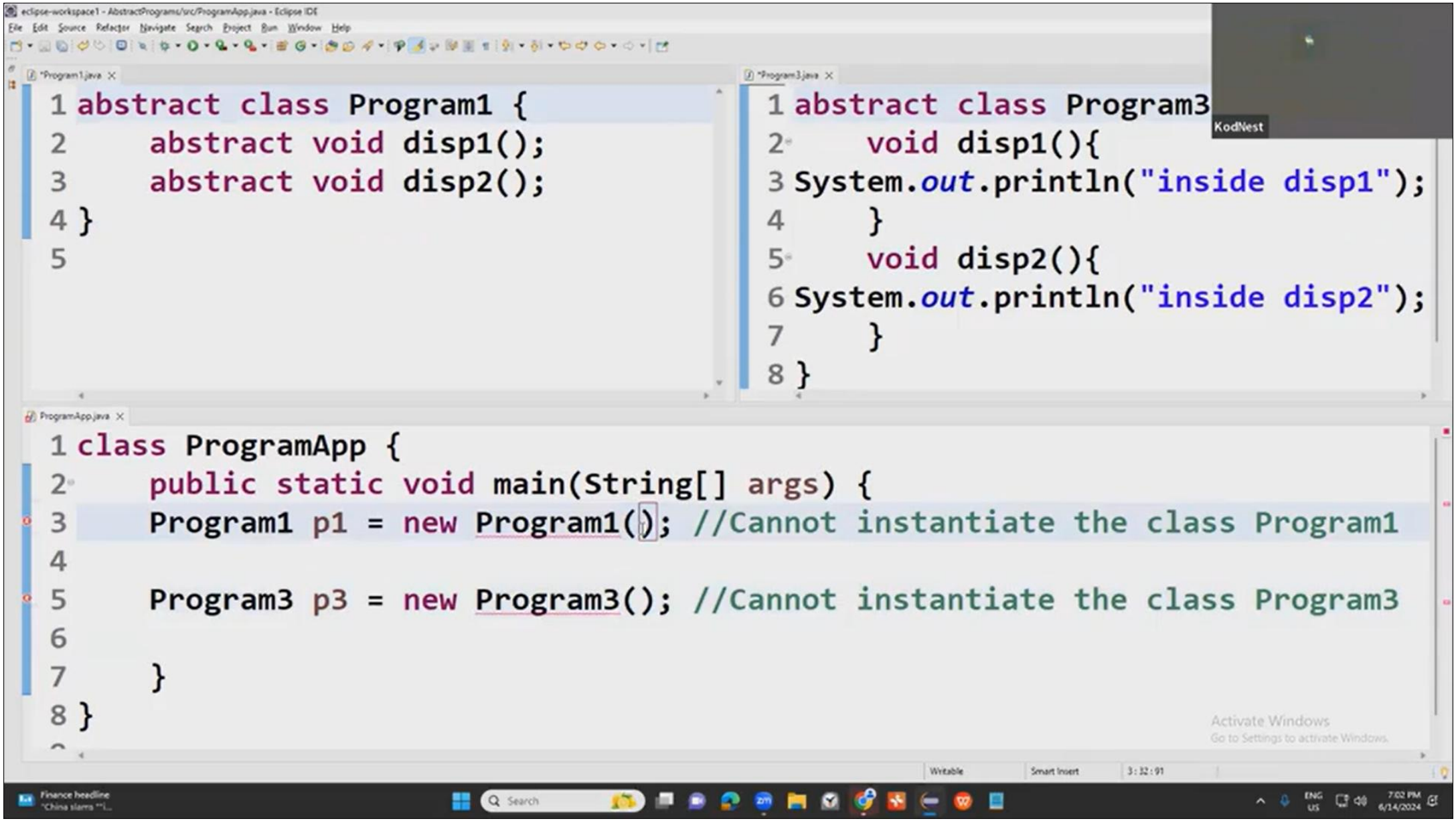
S3


eclipse-workspace1 - AbstractPrograms/src/Program3.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
*Program1.java
Program4pp.java
1 //PURE ABSTRACT CLASS
2 abstract class Program1 {
3 abstract void disp1();
4 abstract void disp2();
5 }
6
*Program2.java
1 //IMPURE ABSTRACT CLASS
2 abstract class Program2 {
3 abstract void disp1();
4 void disp2(){
5 System.out.println("inside di
6 }
7 }
8
*Program3.java
1 //IMPURE ABSTRACT CLASS
2 abstract class Program3 {
3 void disp1(){
4 System.out.println("inside
5 }
6 void disp2(){
7 System.out.println("inside
8 }
9 }
10
KodNest
Activate Windows
Go to Settings to activate Windows.
Writable
Smart Insert
2 : 9 [8]
28 min delay
SHBS / Hosur M...
Search
ENG
IN
6:48 PM
6/14/2024



note:- If a child class is inheriting the parent class and parent class consists of abstract methods then child class must override and provide the body for every abstract method.

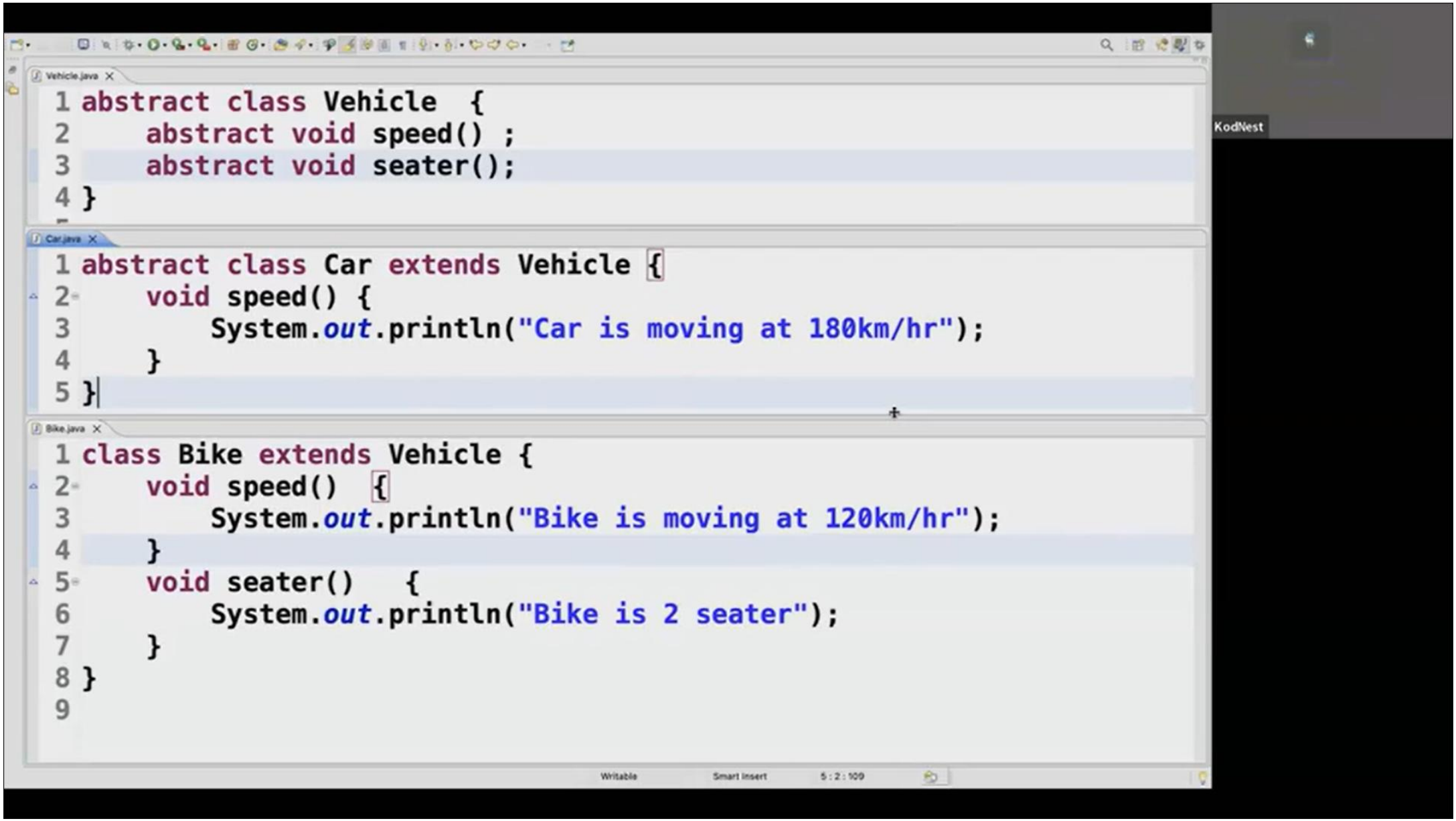
otherwise the child class should be declared as abstract.


eclipse-workspace1 - AbstractPrograms/src/ProgramApp.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
*Program1.java
1 abstract class Program1 {
2 abstract void disp1();
3 abstract void disp2();
4 }
5
*Program3.java
1 abstract class Program3
2 void disp1(){
3 System.out.println("inside disp1");
4 }
5 void disp2(){
6 System.out.println("inside disp2");
7 }
8 }
KodNest
ProgramApp.java
1 class ProgramApp {
2 public static void main(String[] args) {
3 Program1 p1 = new Program1(); //Cannot instantiate the class Program1
4
5 Program3 p3 = new Program3(); //Cannot instantiate the class Program3
6
7 }
8 }
Activate Windows
Go to Settings to activate Windows.
Writable
Smart Insert
3 : 32 : 91
Finance headline
Search
ENG
US
7:02 PM
6/14/2024

note: ... child class is inheriting the parent class and parent class consists of abstract methods then child class must override and provide the body for every abstract method. otherwise the child class should be declared as abstract.

note:- abstract method cannot be declared as Final and static. (because final mtlwe usko override nikrskte.

abstract methods can't called as final because abstract's methods body has to be provided by t
abstract's methods and final method' cannot be over


Vehicle.java
abstract class Vehicle {
abstract void speed() ;
abstract void seater();
}
Car.java
abstract class Car extends Vehicle {
void speed() {
System.out.println("Car is moving at 180km/hr");
}
}
Bike.java
class Bike extends Vehicle {
void speed() {
System.out.println("Bike is moving at 120km/hr");
}
void seater() {
System.out.println("Bike is 2 seater");
}
}
KodNest
Writable
Smart Insert
5 : 2 : 109

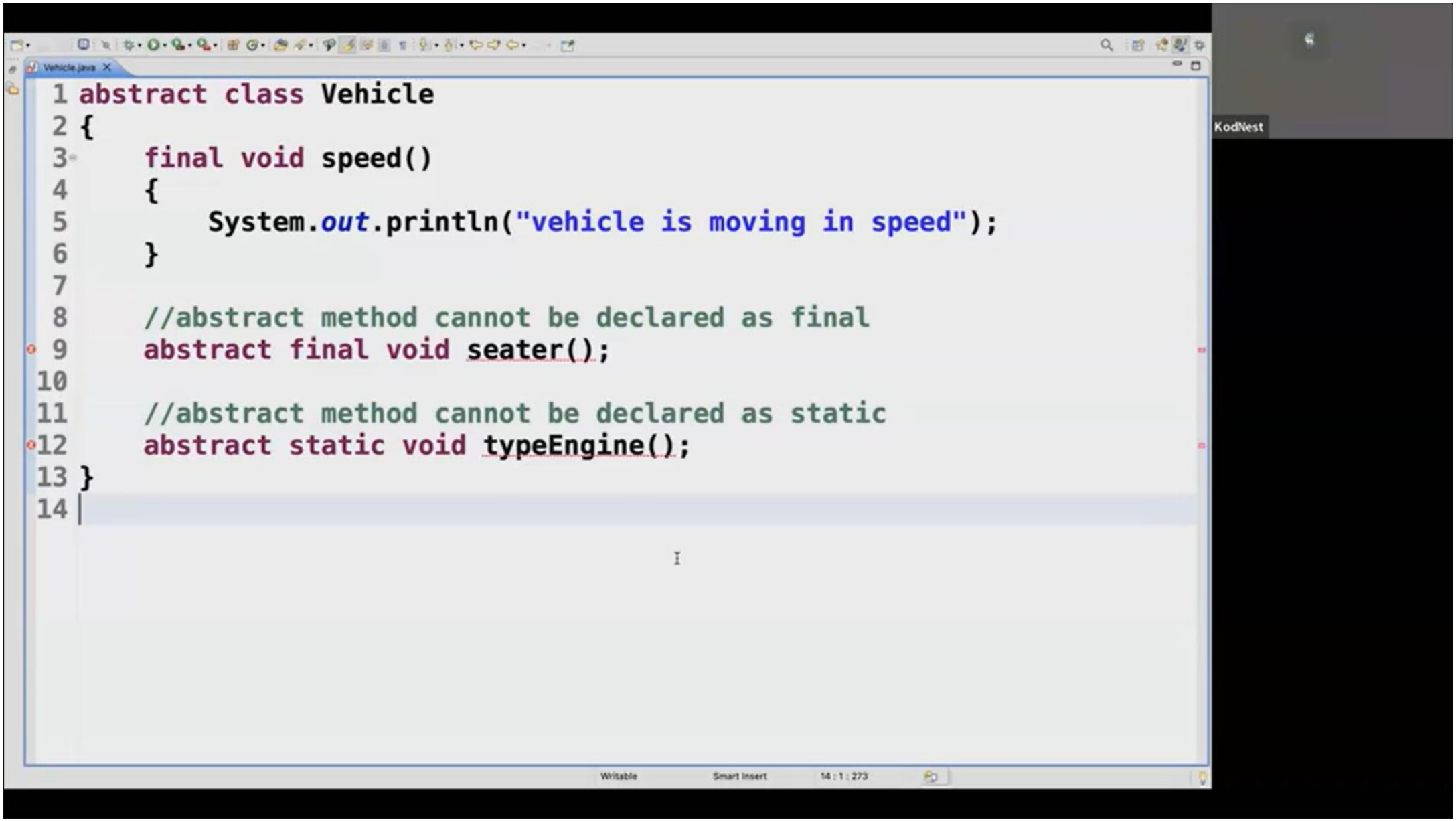
final and static (because final mtwe uske.
override nikrskte.

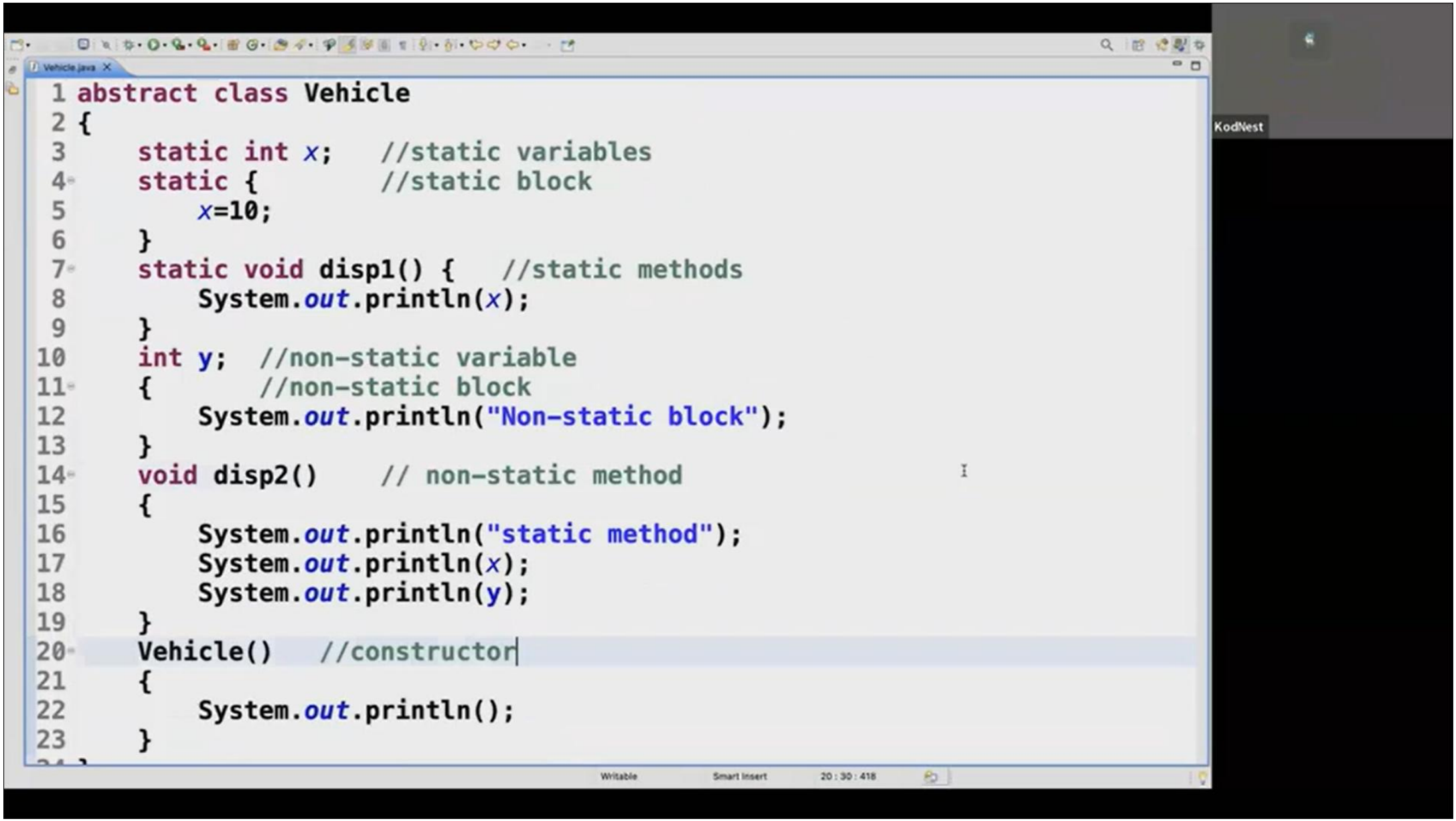
95

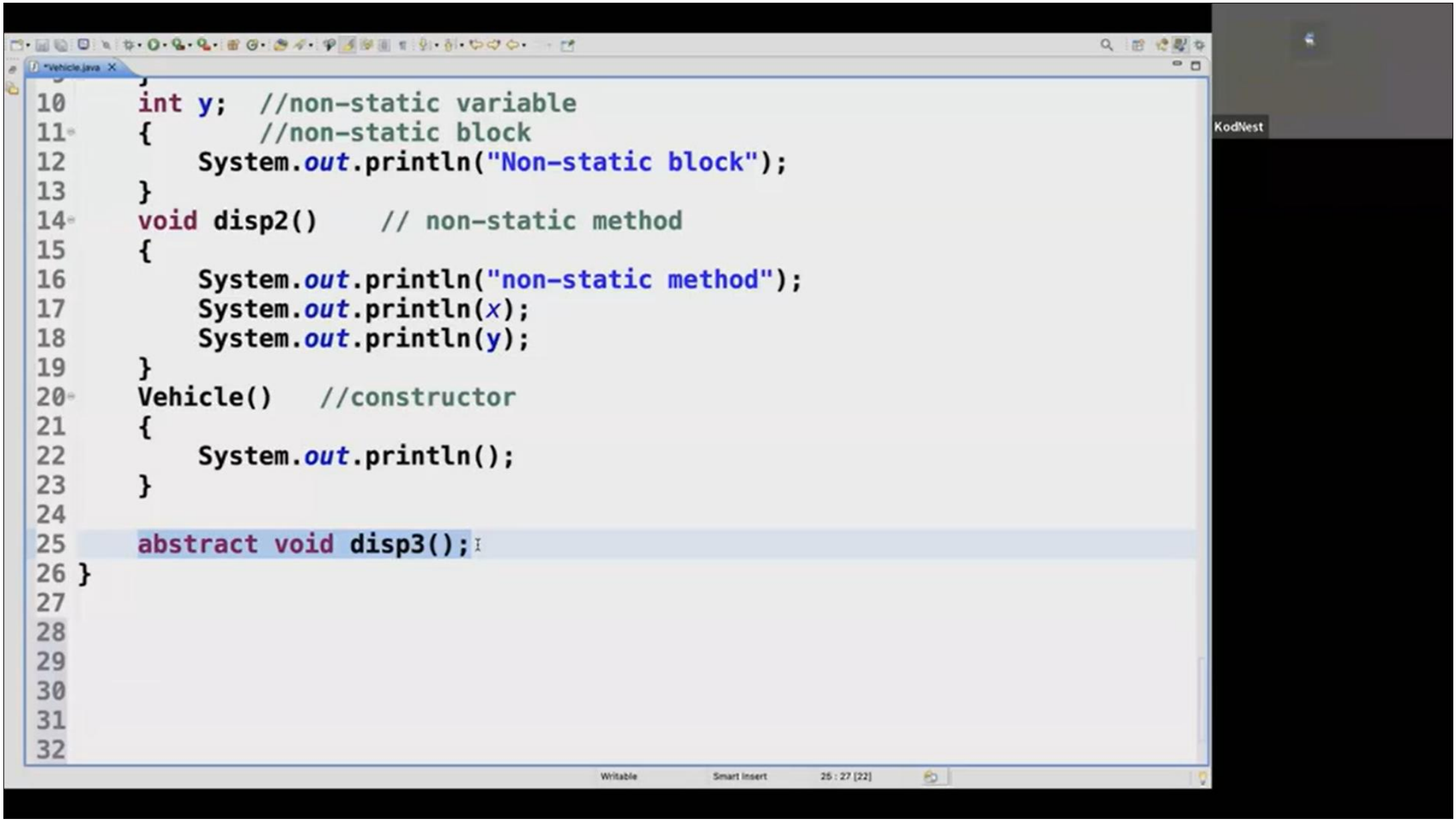
abstract methods can't called as final because abstract's methods body has to be provided by the child class's. and final method' cannot be overridden. which means both opposite to each other.

#A.M is can't declared as static because static methods are class level methods

96


Vehicle.java
1 abstract class Vehicle
2 {
3 final void speed()
4 {
5 System.out.println("vehicle is moving in speed");
6 }
7
8 //abstract method cannot be declared as final
9 abstract final void seater();
10
11 //abstract method cannot be declared as static
12 abstract static void typeEngine();
13 }
14
KodNest
Writable
Smart Insert
14 : 1 : 273


Vehicle.java
abstract class Vehicle
{
static int x; //static variables
static { //static block
x=10;
}
static void disp1() { //static methods
System.out.println(x);
}
int y; //non-static variable
{ //non-static block
System.out.println("Non-static block");
}
void disp2() // non-static method
{
System.out.println("static method");
System.out.println(x);
System.out.println(y);
}
Vehicle() //constructor
{
System.out.println();
}
Writable
Smart Insert
20 : 30 : 418
KodNest


*Vehicle.java
10 int y; //non-static variable
11 { //non-static block
12 System.out.println("Non-static block");
13 }
14 void disp2() // non-static method
15 {
16 System.out.println("non-static method");
17 System.out.println(x);
18 System.out.println(y);
19 }
20 Vehicle() //constructor
21 {
22 System.out.println();
23 }
24
25 abstract void disp3();
26 }
27
28
29
30
31
32
Writable
Smart Insert
25 : 27 [22]
KodNest

www.kodnest.com

An abstract class can consist of.

S6 - S7

Static variable
static block.
static method.
non-static variable
non-static block
non-static method
constructor.

Abstraction using interfaces :-

S8 - 9

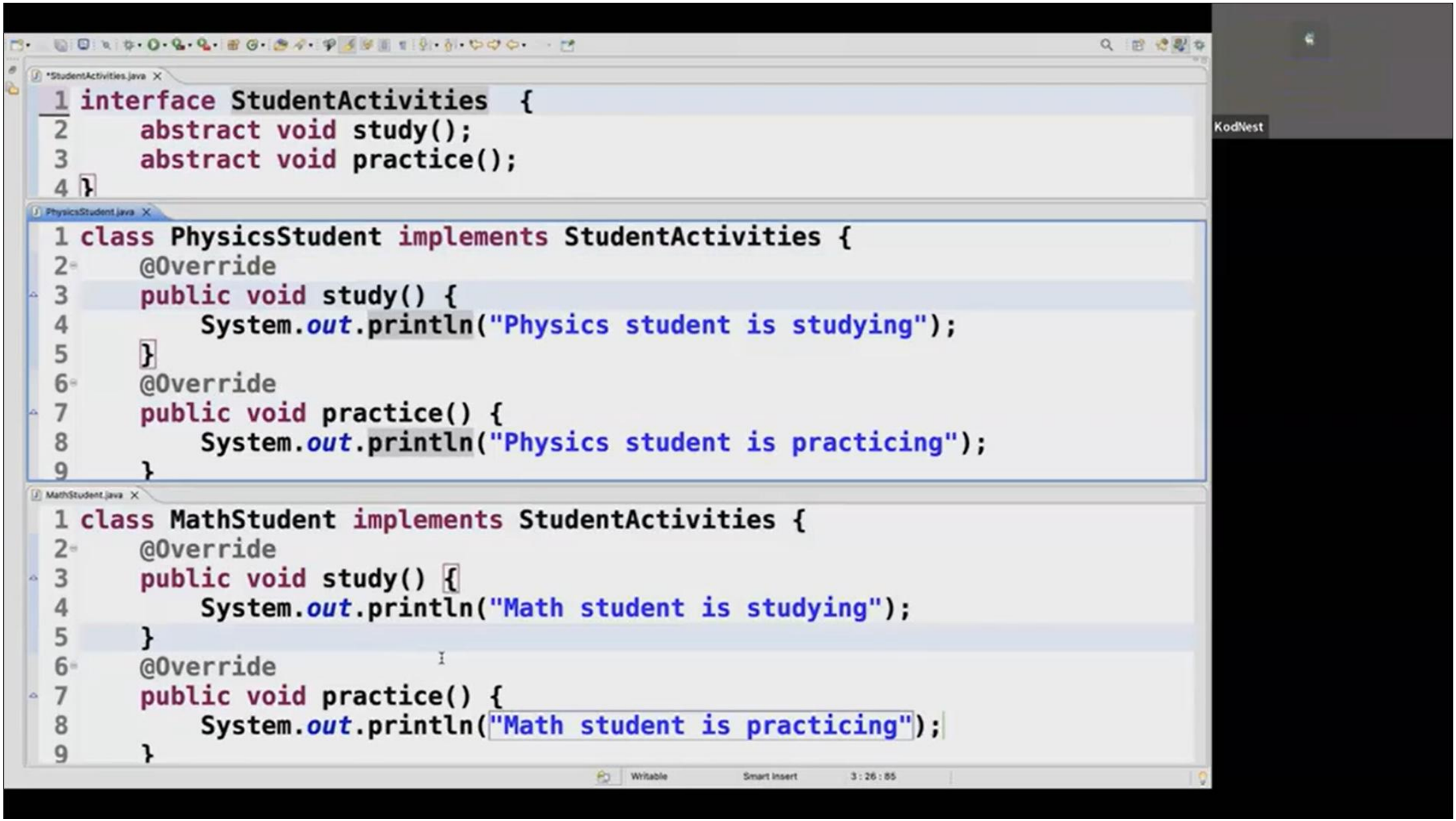
constructor.

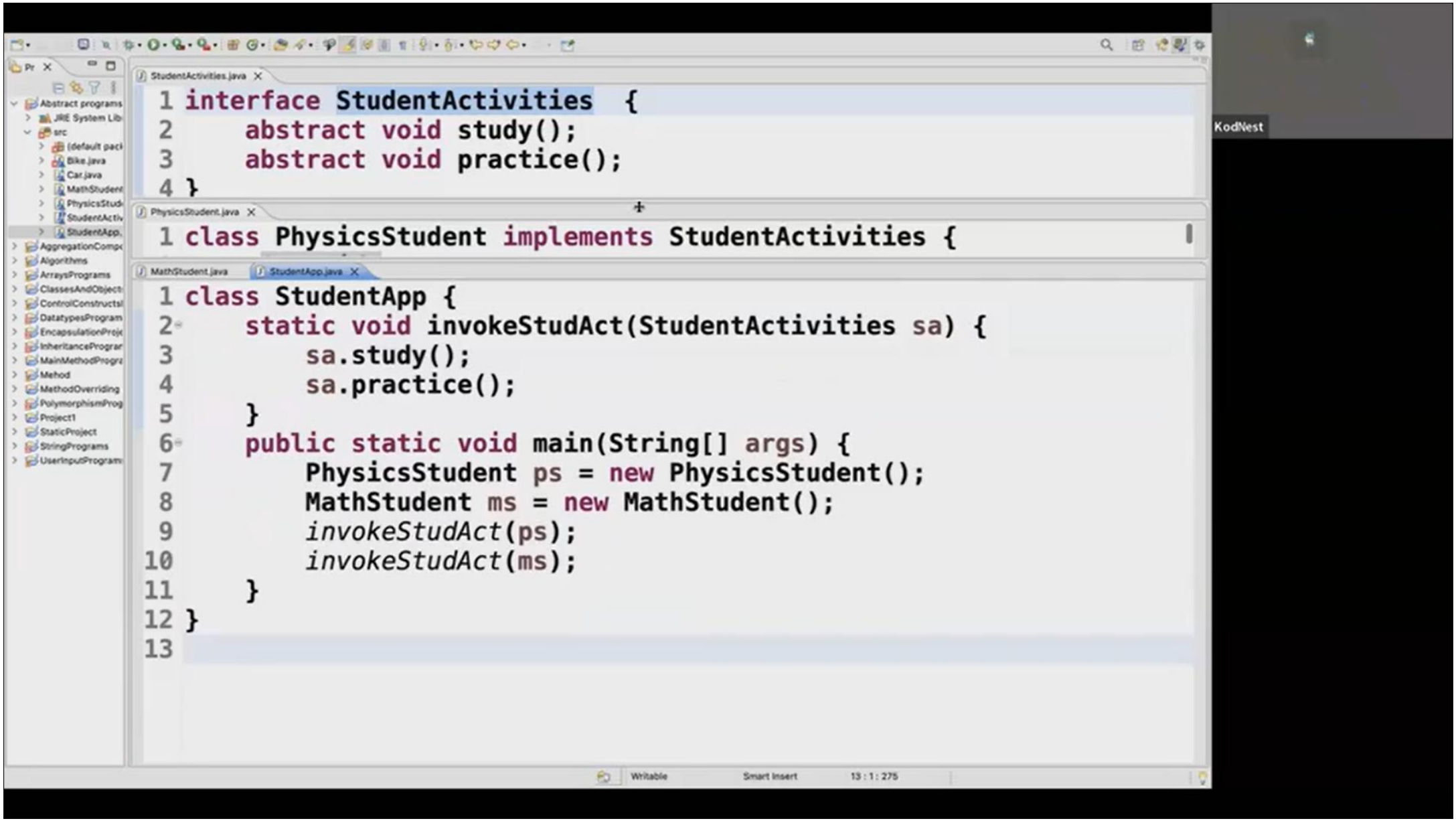
Abstraction using interfaces :- S8 - 9

note... Interface consist of abstract method.
It is another way of achieving abstraction
when a class connects with an Interface. "Implements"
keyword has to be used which means the Implemented class or child class has to provide the
body for all the abstract methods otherwise
It has to be declared as abstract

Interfaces can not have concreate methods.

Standardisation.


*StudentActivities.java
interface StudentActivities {
abstract void study();
abstract void practice();
}
PhysicsStudent.java
class PhysicsStudent implements StudentActivities {
@Override
public void study() {
System.out.println("Physics student is studying");
}
@Override
public void practice() {
System.out.println("Physics student is practicing");
}
MathStudent.java
class MathStudent implements StudentActivities {
@Override
public void study() {
System.out.println("Math student is studying");
}
@Override
public void practice() {
System.out.println("Math student is practicing");
}
KodNest
Writable
Smart Insert
3 : 26 : 85


StudentActivities.java
interface StudentActivities {
abstract void study();
abstract void practice();
}
PhysicsStudent.java
class PhysicsStudent implements StudentActivities {
MathStudent.java
StudentApp.java
class StudentApp {
static void invokeStudAct(StudentActivities sa) {
sa.study();
sa.practice();
}
public static void main(String[] args) {
PhysicsStudent ps = new PhysicsStudent();
MathStudent ms = new MathStudent();
invokeStudAct(ps);
invokeStudAct(ms);
}
}
Writable
Smart Insert
13 : 1 : 275
KodNest

because of the following reason.

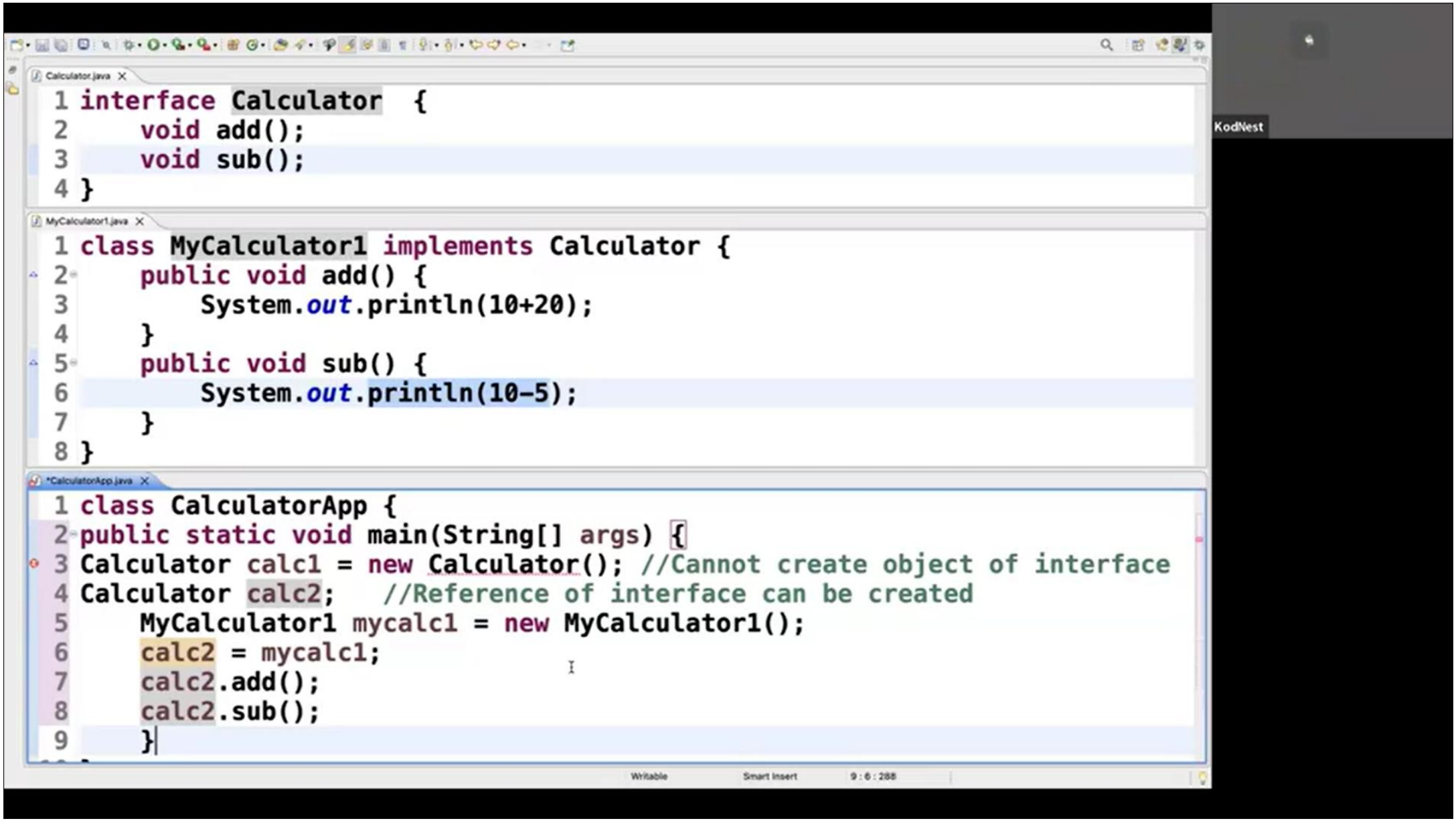
① to acheive standalization, i- which means inside the interface only method. declaration with provid In other words interfaces will tell what to do not how do do.

② using interfaces multipal inheritance can be achiev which can not be acheived by abstract class it is possible using the interface because interface do not have any parent by defaut.

S1

#2. all the methods inside interface are abstrac
all the method — — — by default public

S2 - S


Calculator.java
interface Calculator {
void add();
void sub();
}
MyCalculator1.java
class MyCalculator1 implements Calculator {
public void add() {
System.out.println(10+20);
}
public void sub() {
System.out.println(10-5);
}
}
*CalculatorApp.java
class CalculatorApp {
public static void main(String[] args) {
Calculator calc1 = new Calculator(); //Cannot create object of interface
Calculator calc2; //Reference of interface can be created
MyCalculator1 mycalc1 = new MyCalculator1();
calc2 = mycalc1;
calc2.add();
calc2.sub();
}
Writable
Smart Insert
9 : 6 : 288
KodNest

because of the following reason.

① to achieve standardization, i- which means inside the interface only method declaration with provided in other words interfaces will tell what to do not how do do.

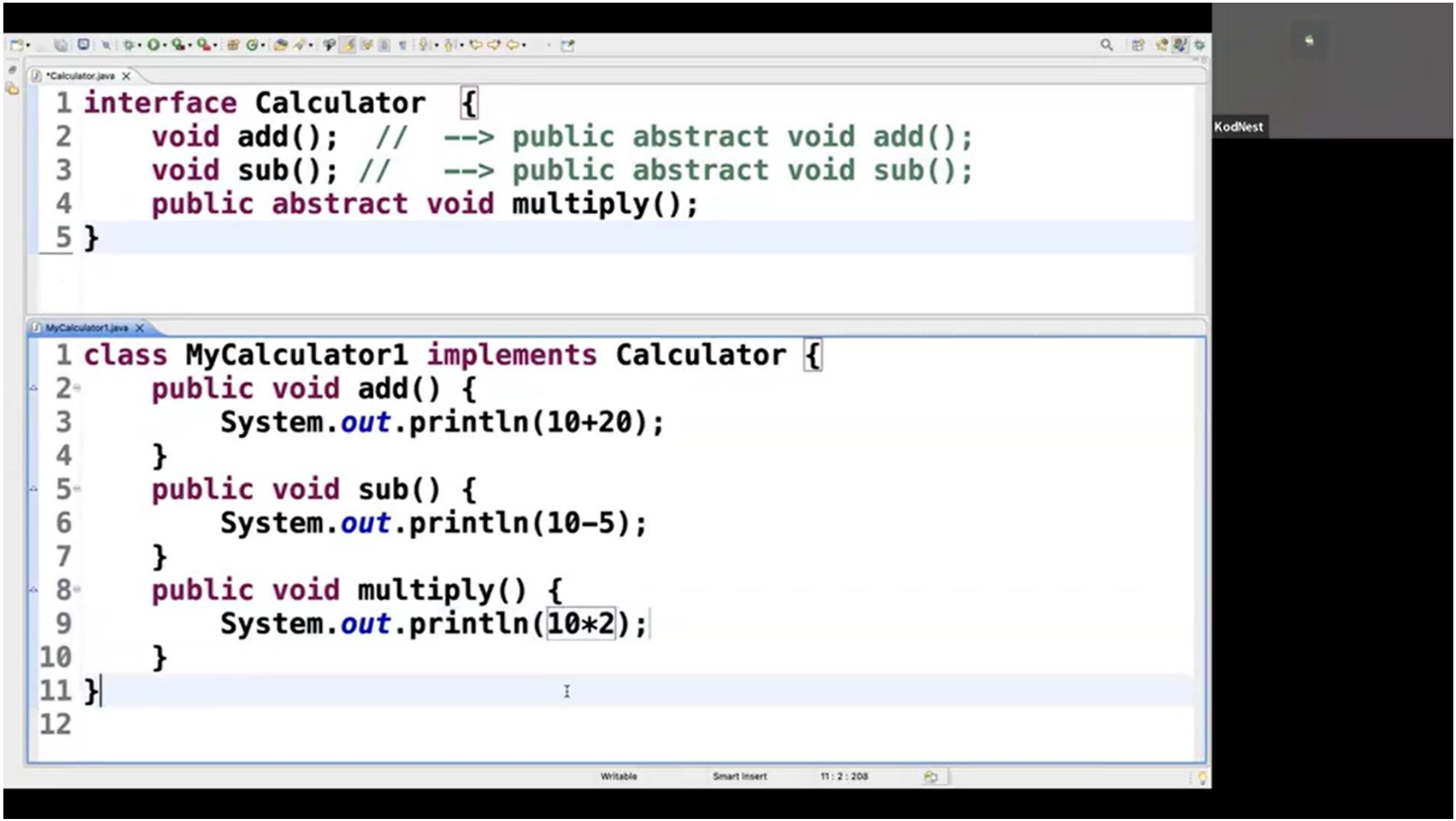
② using interfaces multipal inheritance can be achieve which can not be acheived by abstract class it is possible using the interface because interface do not have any parent by defaut.

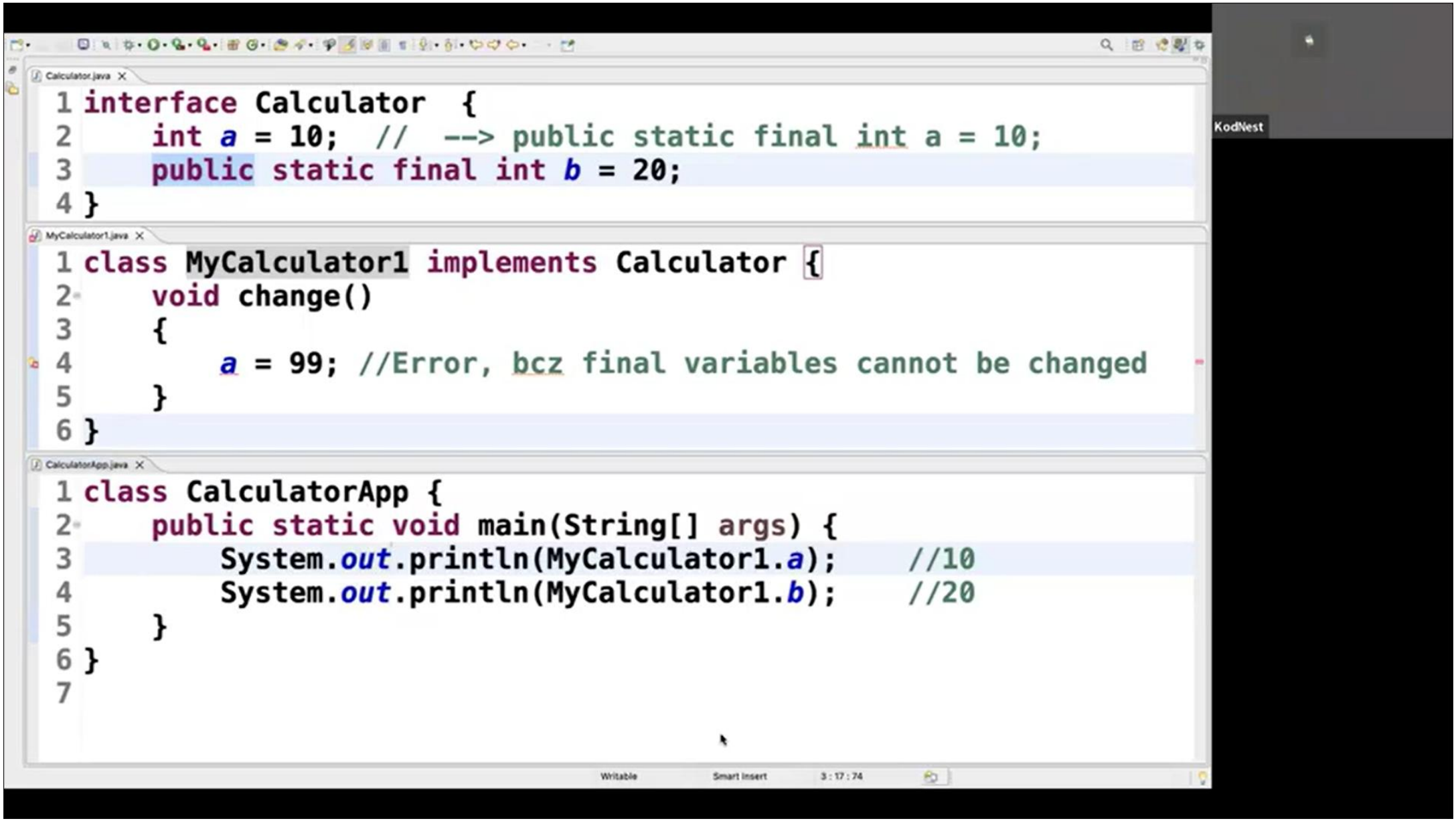
S1

#2. all the methods inside interface are abstract
all the method — — — by default public

S2 - S3

#b. All variable in interfaces are decla

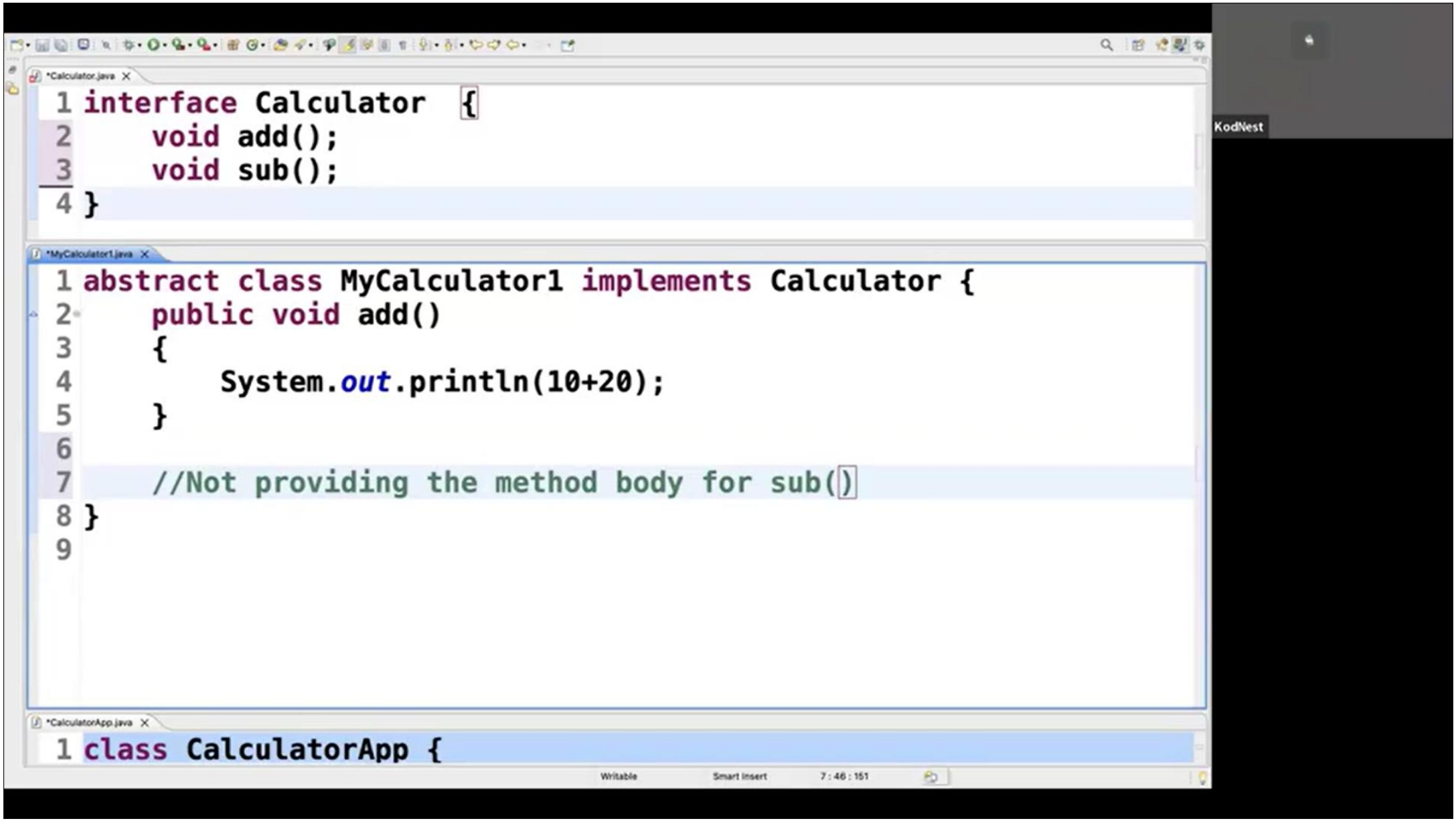

*Calculator.java
interface Calculator {
void add(); // --> public abstract void add();
void sub(); // --> public abstract void sub();
public abstract void multiply();
}
MyCalculator1.java
class MyCalculator1 implements Calculator {
public void add() {
System.out.println(10+20);
}
public void sub() {
System.out.println(10-5);
}
public void multiply() {
System.out.println(10*2);
}
}
Writable
Smart Insert
11 : 2 : 208
KodNest

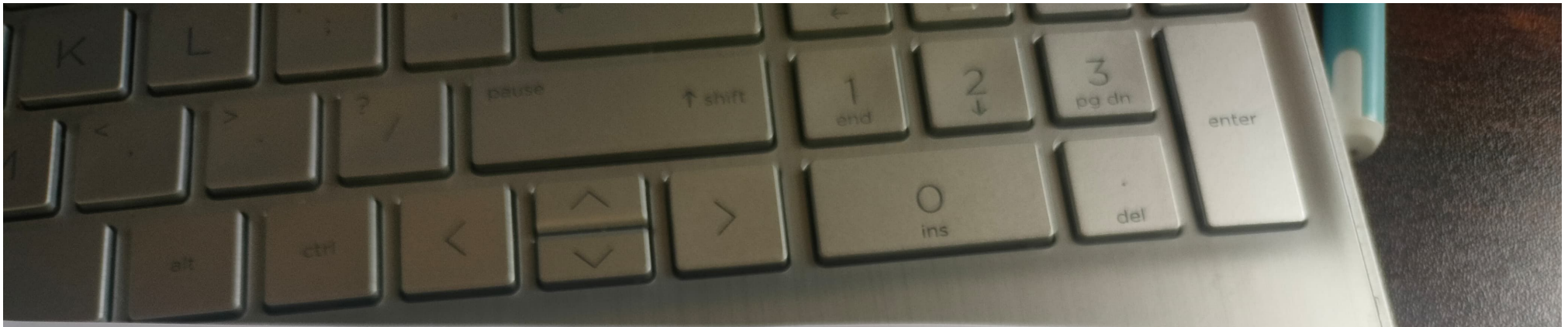

Calculator.java
interface Calculator {
int a = 10; // --> public static final int a = 10;
public static final int b = 20;
}
MyCalculator1.java
class MyCalculator1 implements Calculator {
void change()
{
a = 99; //Error, bcz final variables cannot be changed
}
}
CalculatorApp.java
class CalculatorApp {
public static void main(String[] args) {
System.out.println(MyCalculator1.a); //10
System.out.println(MyCalculator1.b); //20
}
}
KodNest
Writable
Smart Insert
3 : 17 : 74

#2. all the methods inside interface are abstract
all the method ——— by default public
S2 - S3

#3: All the variable .oin. interfaces are declaired as public static and final. ex. S4.

#. If any class is partially implementing (interface ki kuch method over oide kr ray hai or kuch nii) the interface (it is not providing the body all abst method then the abstract also have to be an abst class. S4 S.


*Calculator.java
1 interface Calculator {
2 void add();
3 void sub();
4 }
*MyCalculator1.java
1 abstract class MyCalculator1 implements Calculator {
2 public void add()
3 {
4 System.out.println(10+20);
5 }
6
7 //Not providing the method body for sub()
8 }
9
*CalculatorApp.java
1 class CalculatorApp {
KodNest
Writable
Smart Insert
7 : 46 : 151



multiple Inheritence:-

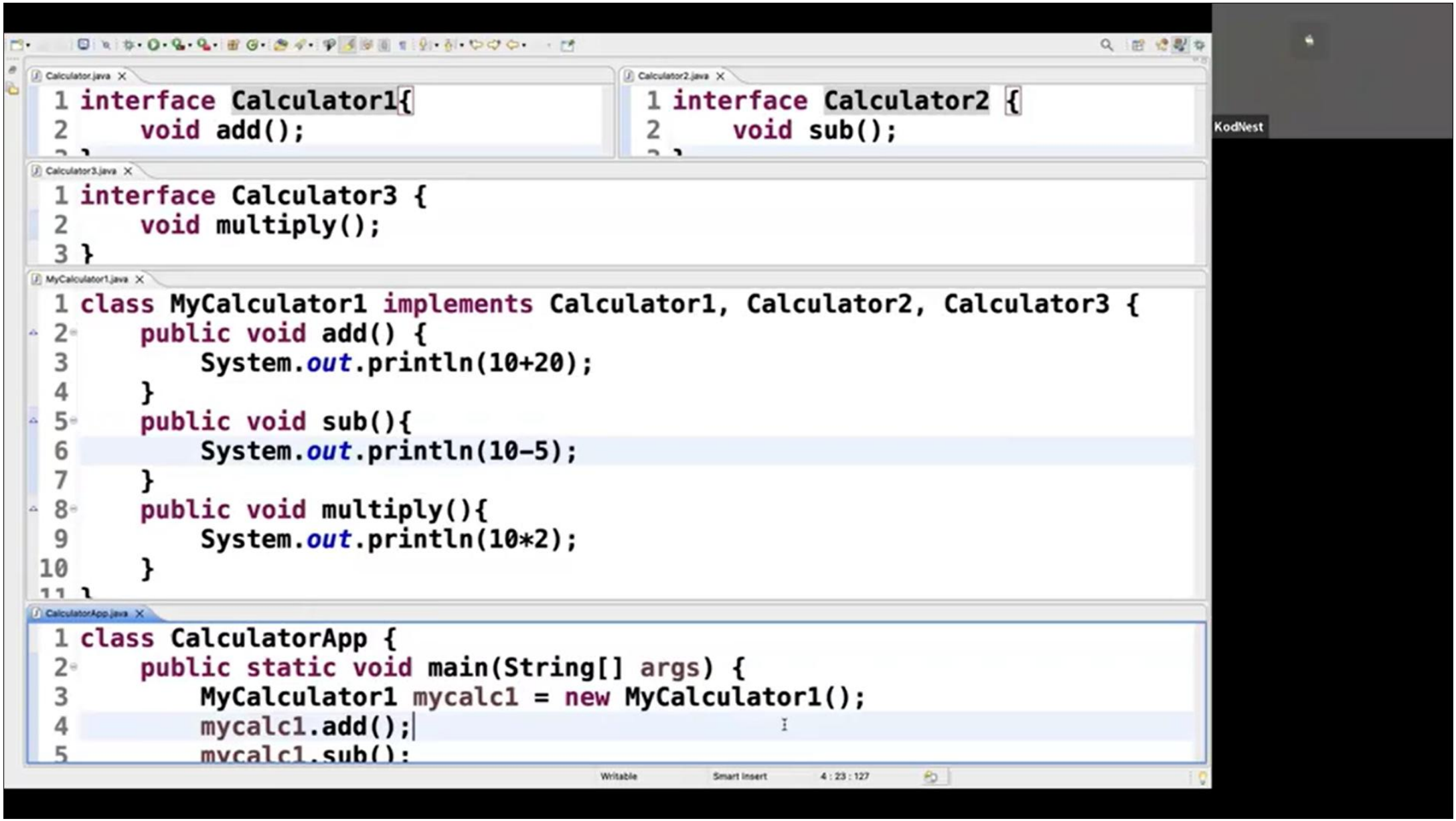
The above example. represents. multiple inheritence using Interface.

S6

Extends:- Is a.

Implements:- promise (i will override your methods)

interface can never implement another interface

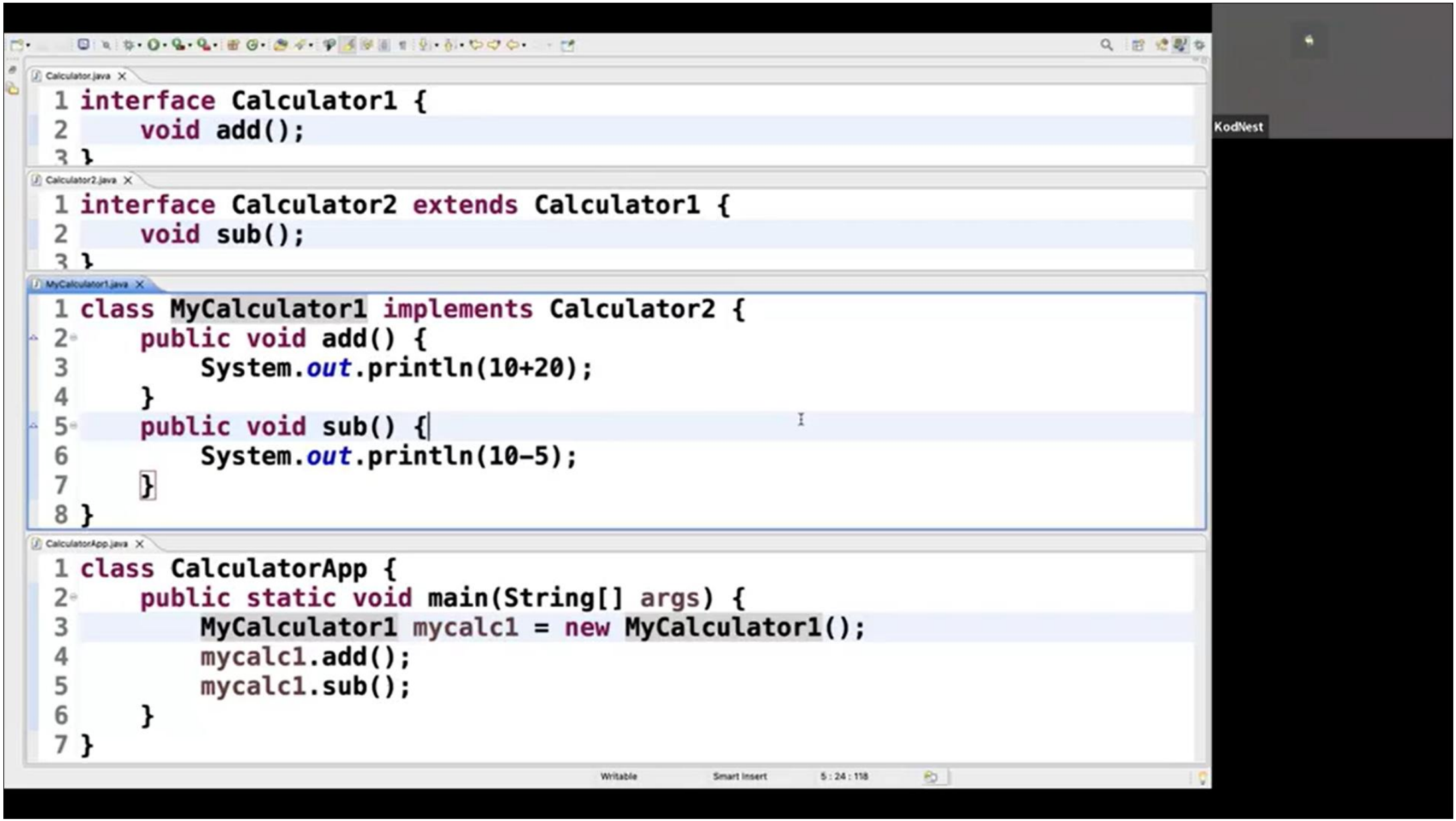

Calculator.java
interface Calculator1{
void add();
Calculator2.java
interface Calculator2 {
void sub();
Calculator3.java
interface Calculator3 {
void multiply();
}
MyCalculator1.java
class MyCalculator1 implements Calculator1, Calculator2, Calculator3 {
public void add() {
System.out.println(10+20);
}
public void sub(){
System.out.println(10-5);
}
public void multiply(){
System.out.println(10*2);
}
CalculatorApp.java
class CalculatorApp {
public static void main(String[] args) {
MyCalculator1 mycalc1 = new MyCalculator1();
mycalc1.add();
mycalc1.sub();
KodNest
Writable
Smart Insert
4 : 23 : 127

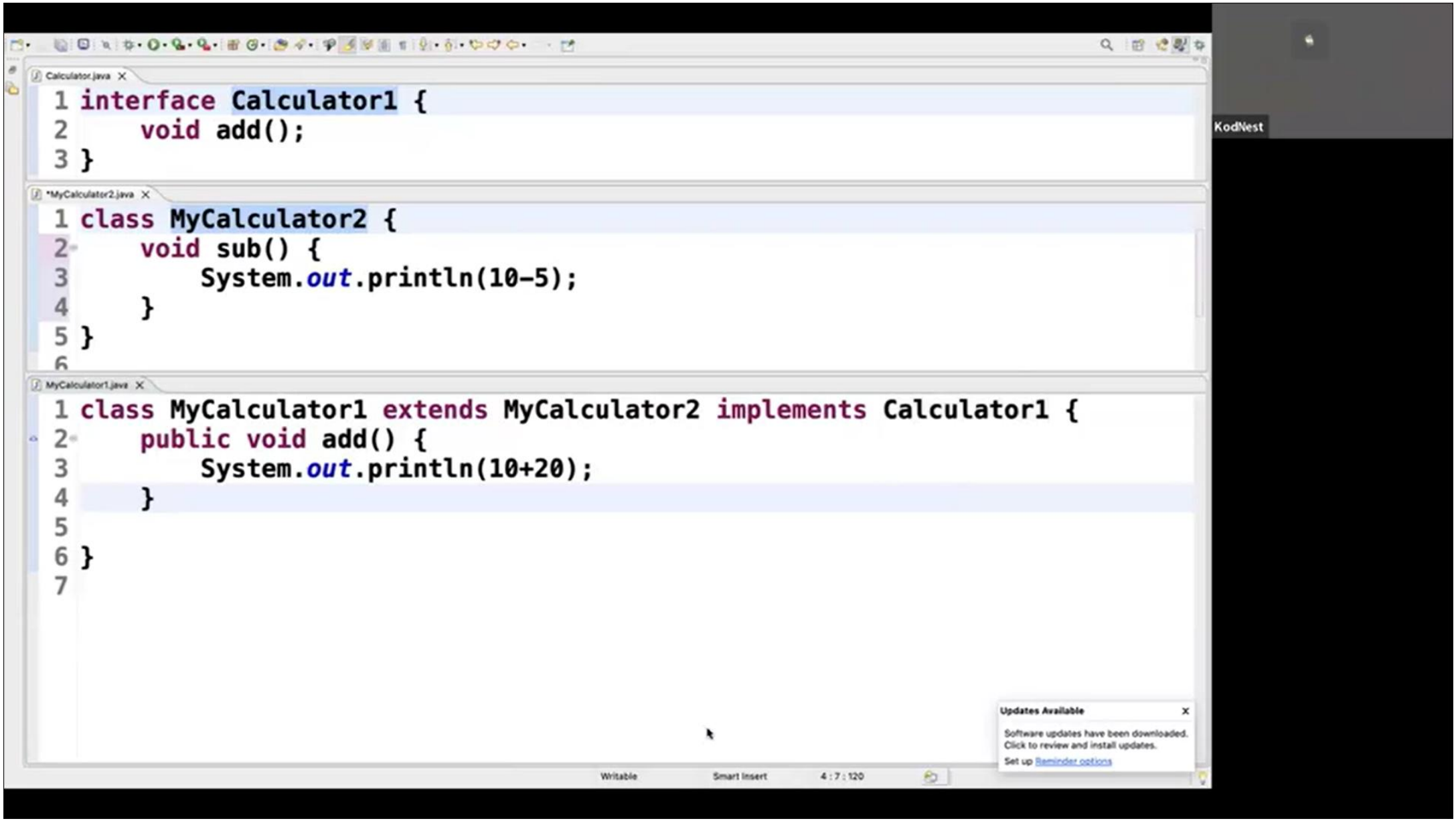
Extends :- Is a.

Implements :- promise (i will override your methods)

interface can never implement another interface rather it can extent another interface.

If a class is ~~ext~~ implementing the extended child interface then it has to provid the me body for all the methods of child interface and parent interface :- ex. - 97


Calculator.java
interface Calculator1 {
void add();
}
Calculator2.java
interface Calculator2 extends Calculator1 {
void sub();
}
MyCalculator1.java
class MyCalculator1 implements Calculator2 {
public void add() {
System.out.println(10+20);
}
public void sub() {
System.out.println(10-5);
}
}
CalculatorApp.java
class CalculatorApp {
public static void main(String[] args) {
MyCalculator1 mycalc1 = new MyCalculator1();
mycalc1.add();
mycalc1.sub();
}
}
Writable
Smart Insert
5 : 24 : 118
KodNest


Calculator.java
interface Calculator1 {
void add();
}
*MyCalculator2.java
class MyCalculator2 {
void sub() {
System.out.println(10-5);
}
}
MyCalculator1.java
class MyCalculator1 extends MyCalculator2 implements Calculator1 {
public void add() {
System.out.println(10+20);
}
}
KodNest
Updates Available
Software updates have been downloaded.
Click to review and install updates.
Set up Reminder options
Writable
Smart Insert
4 : 7 : 120

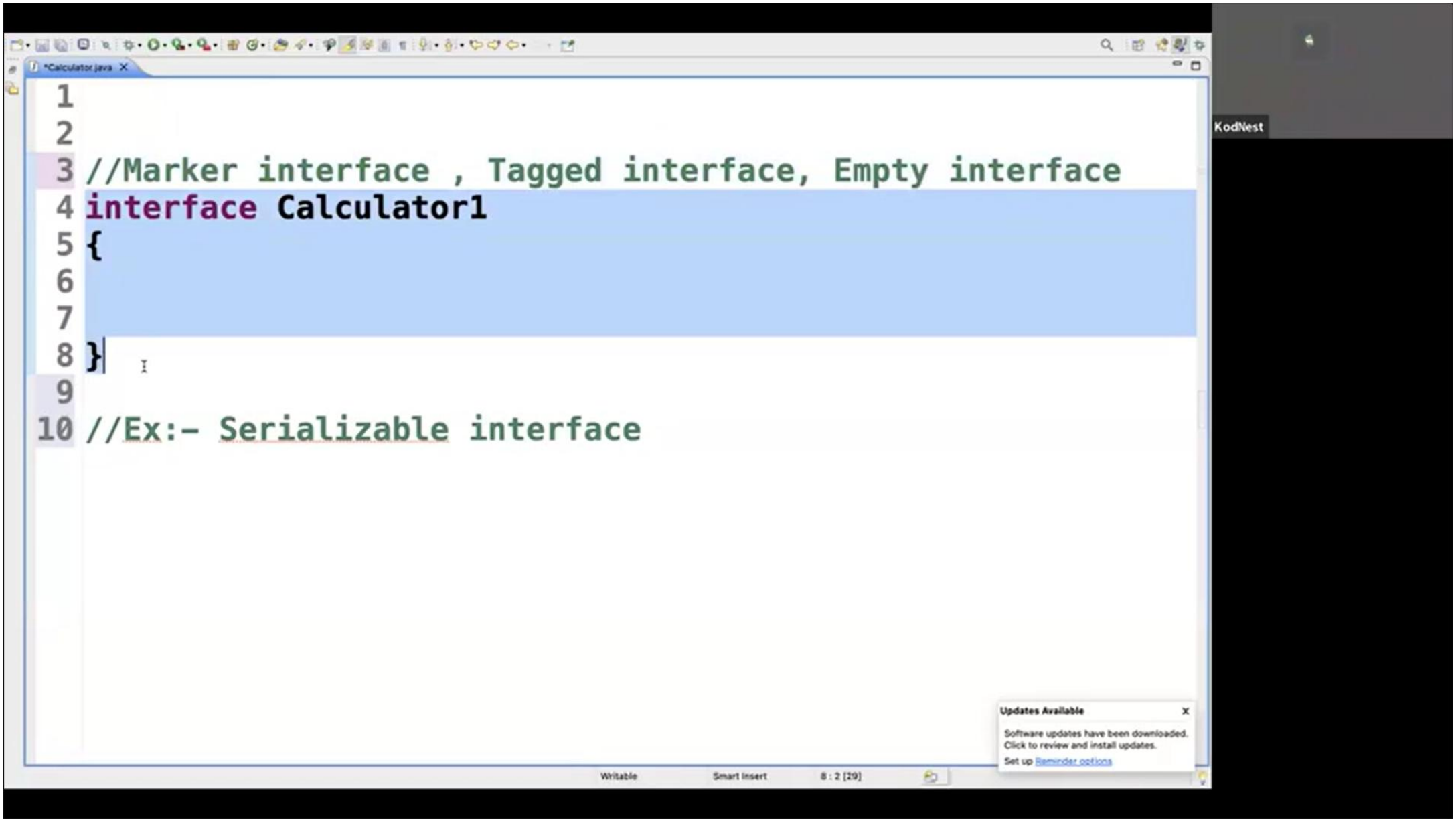
child interface
body for all the methods
and parent interface:- ex - 97

a class can extend another class and implements an interface at the same time.

58

marker interface, Tagged interface, empty interface

empty interfaces are called as marker interfaces
they also consider as taged interface.
serializable interface is the marked interface


*Calculator.java
1
2
3 //Marker interface , Tagged interface, Empty interface
4 interface Calculator1
5 {
6
7
8 }
9
10 //Ex:- Serializable interface
KodNest
Updates Available
Software updates have been downloaded.
Click to review and install updates.
Set up Reminder options
Writable
Smart Insert
8 : 2 [29]