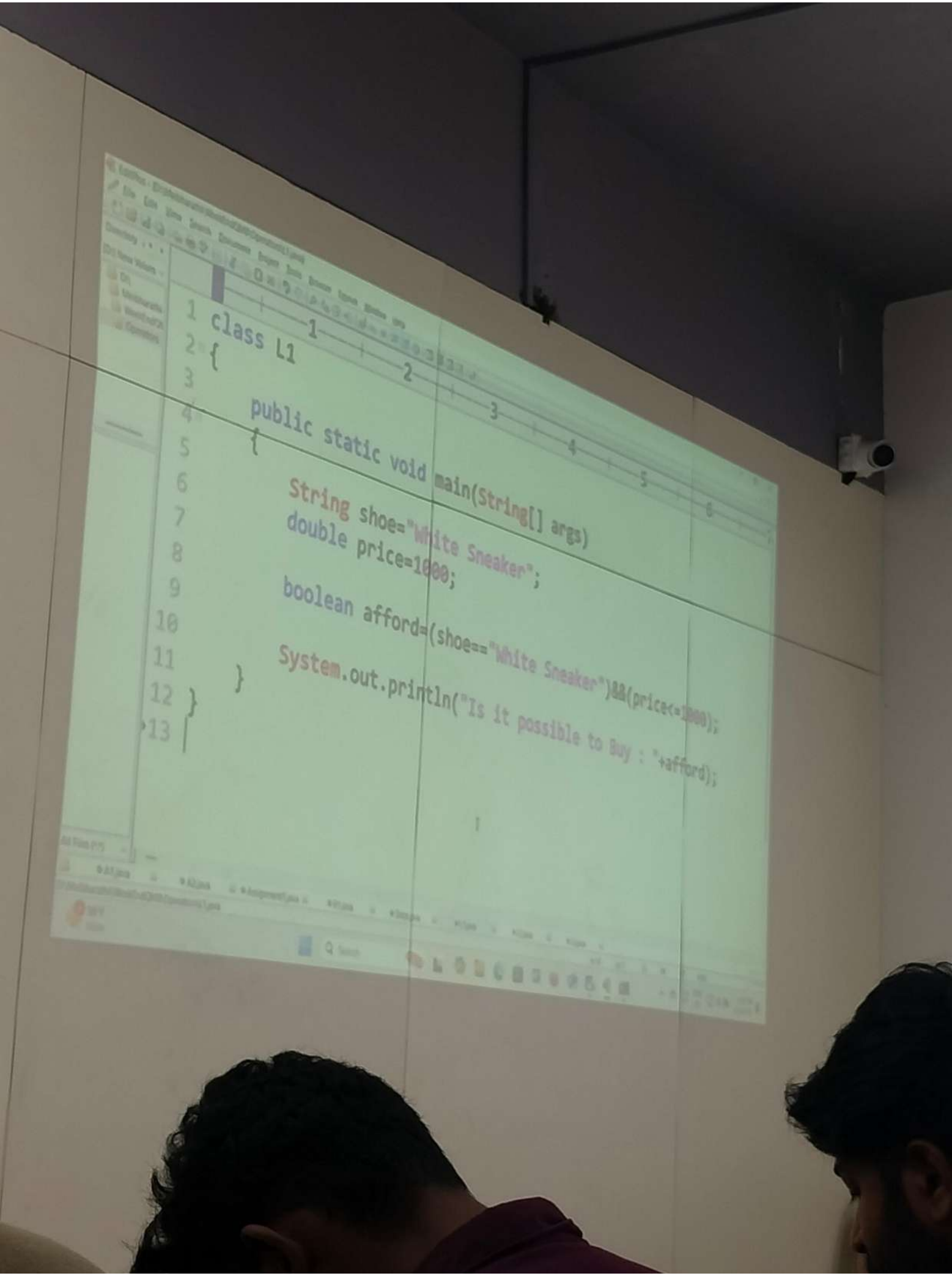
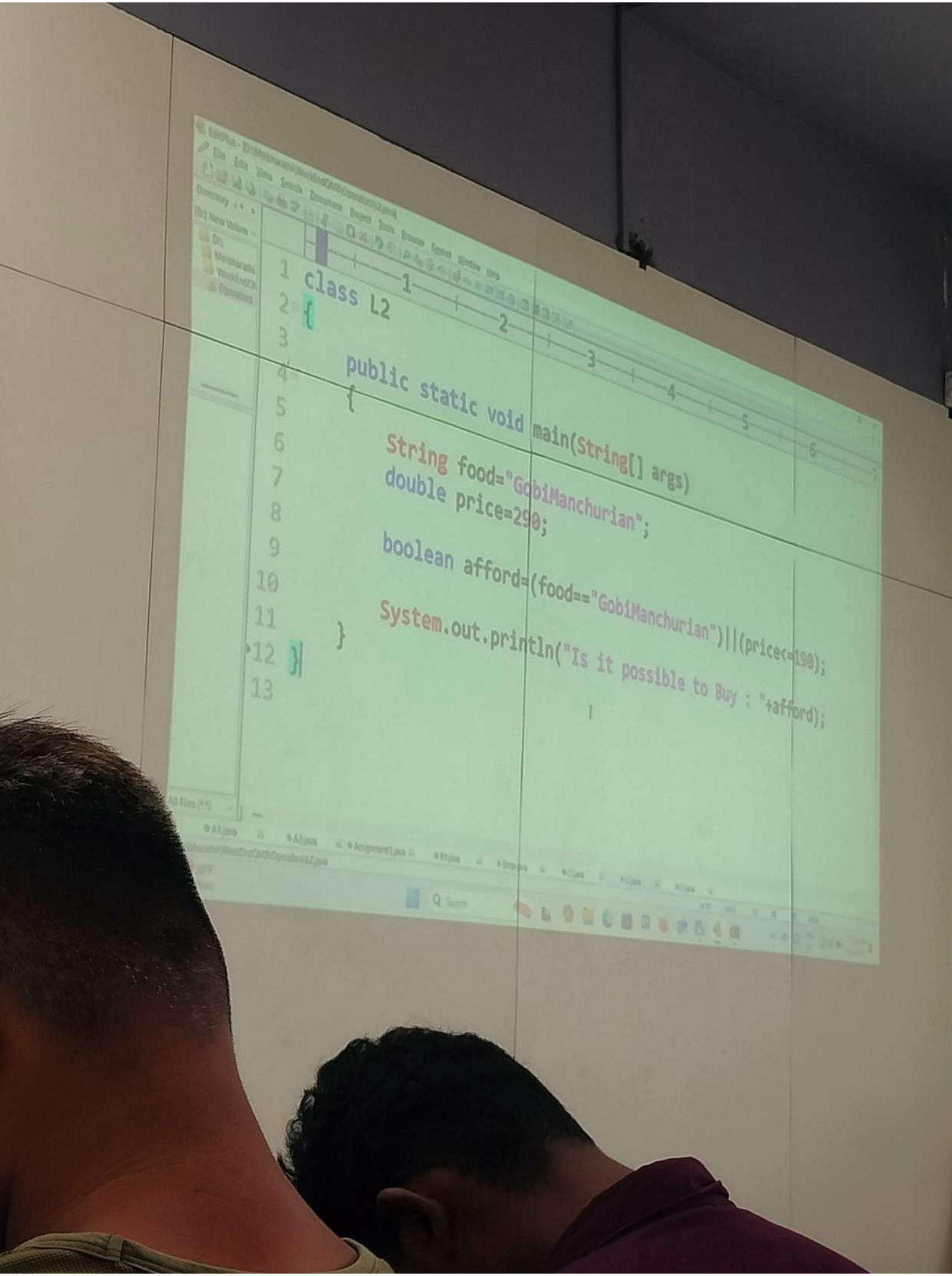
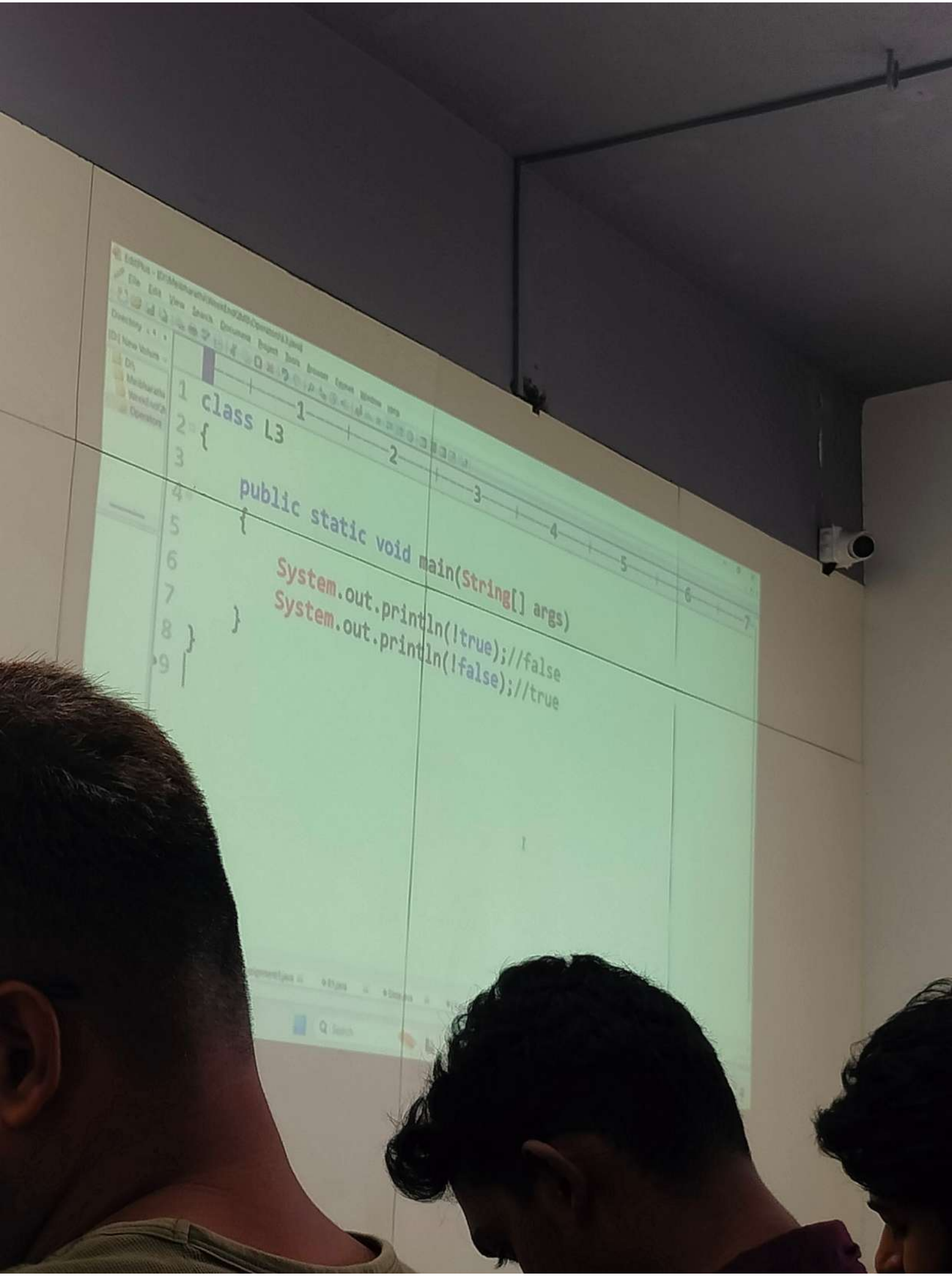
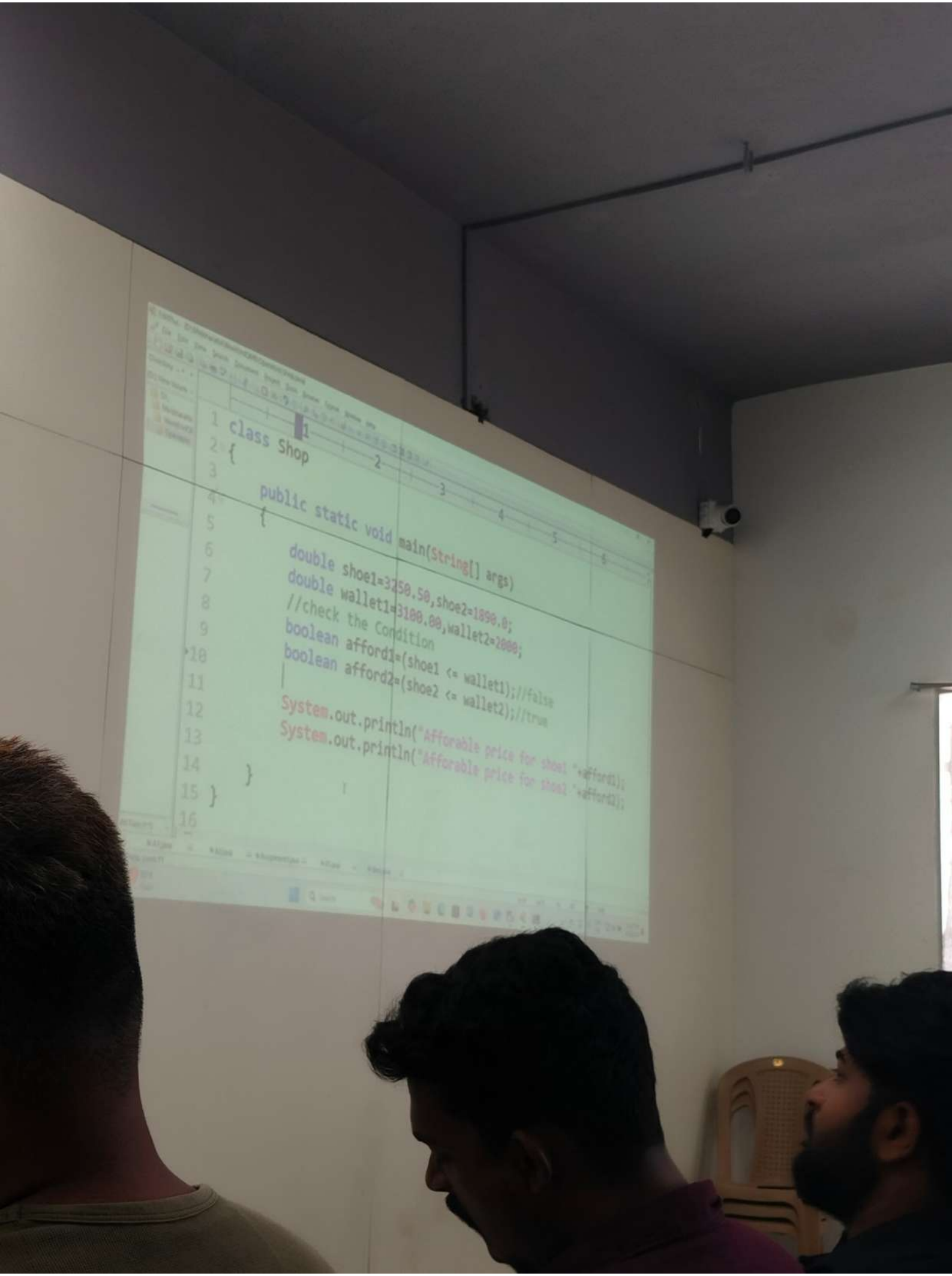
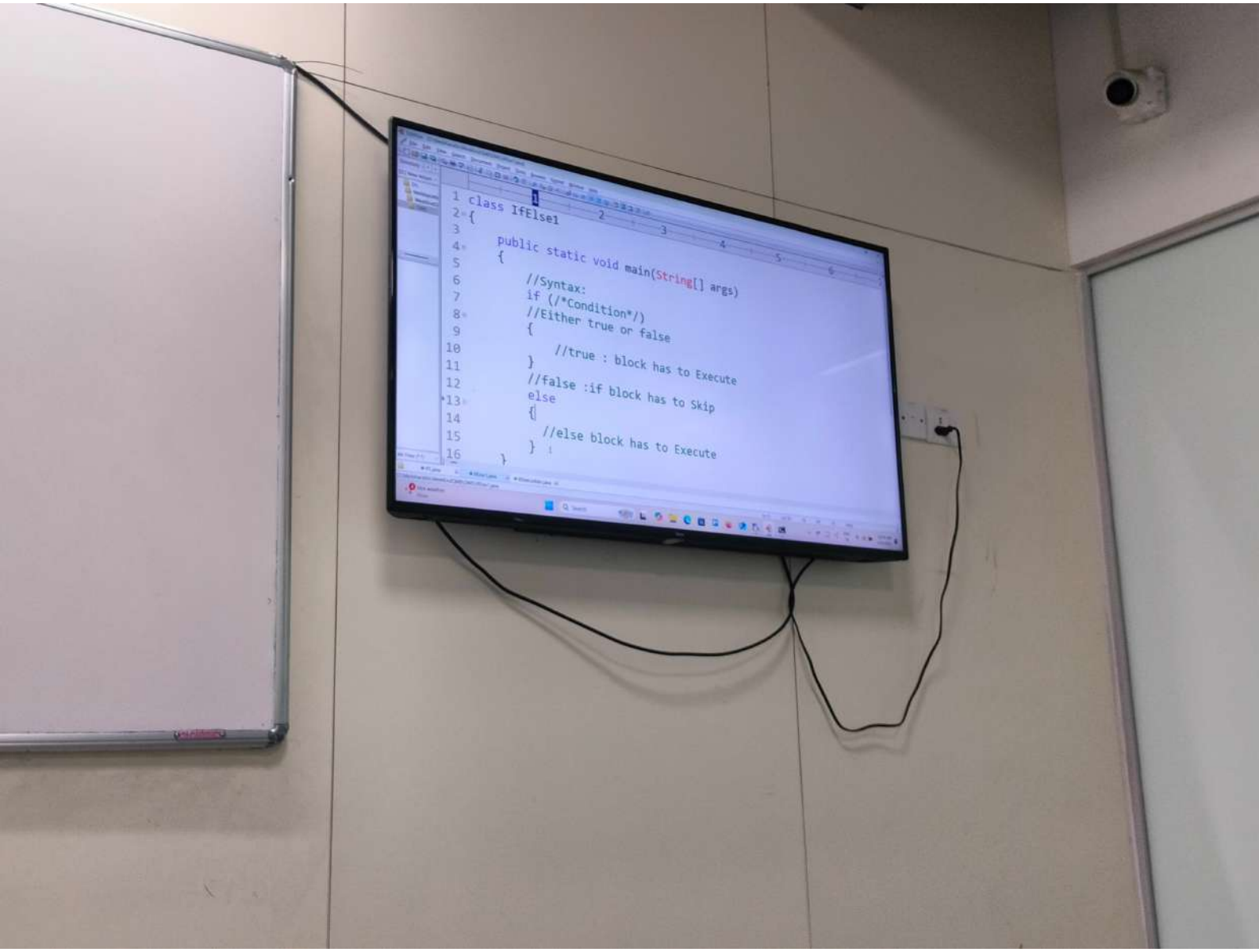
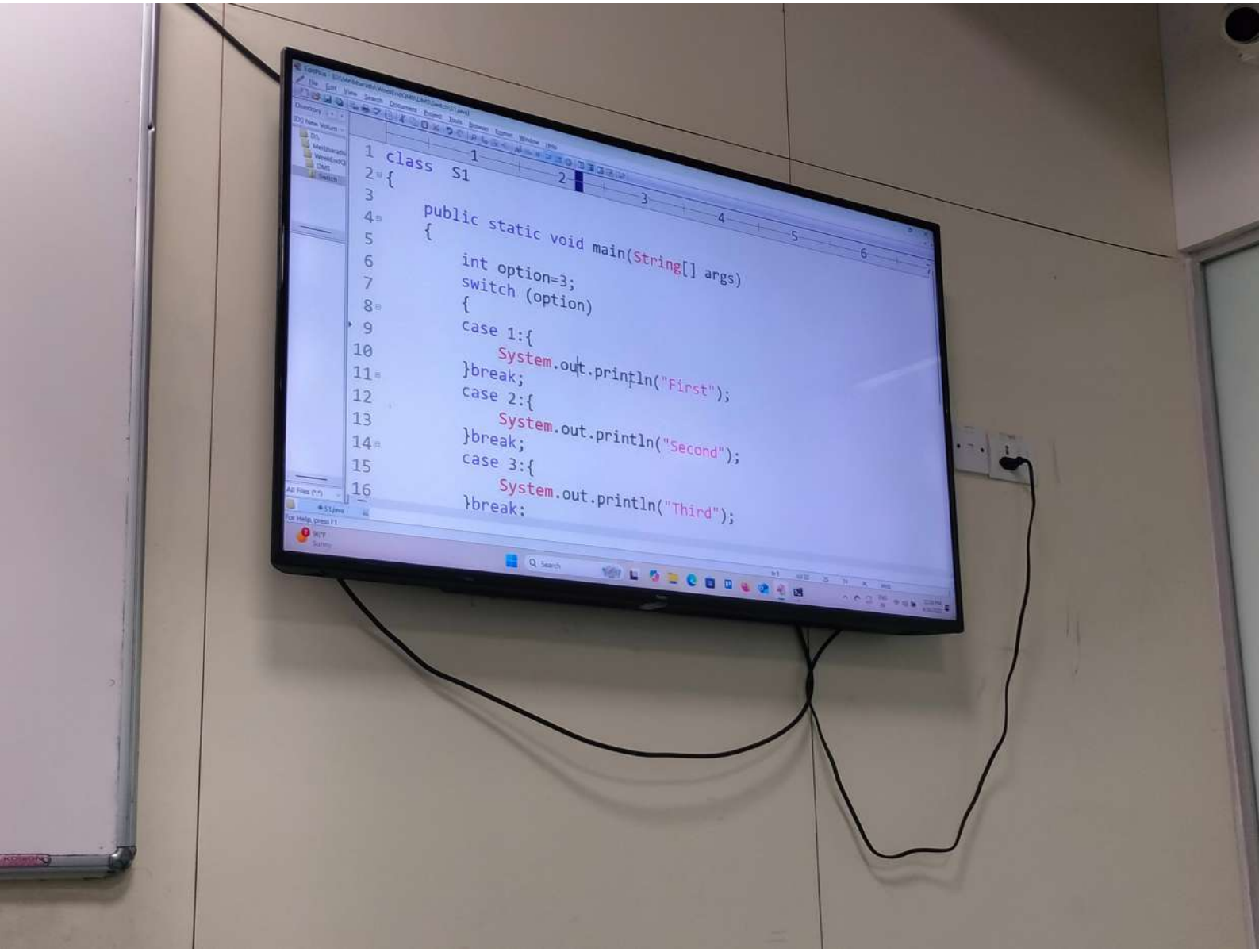

class L1
{
public static void main(String[] args)
{
String shoe="White Sneaker";
double price=1000;
boolean afford=(shoe=="White Sneaker")&&(price<=1000);
System.out.println("Is it possible to Buy : "+afford);
}
}

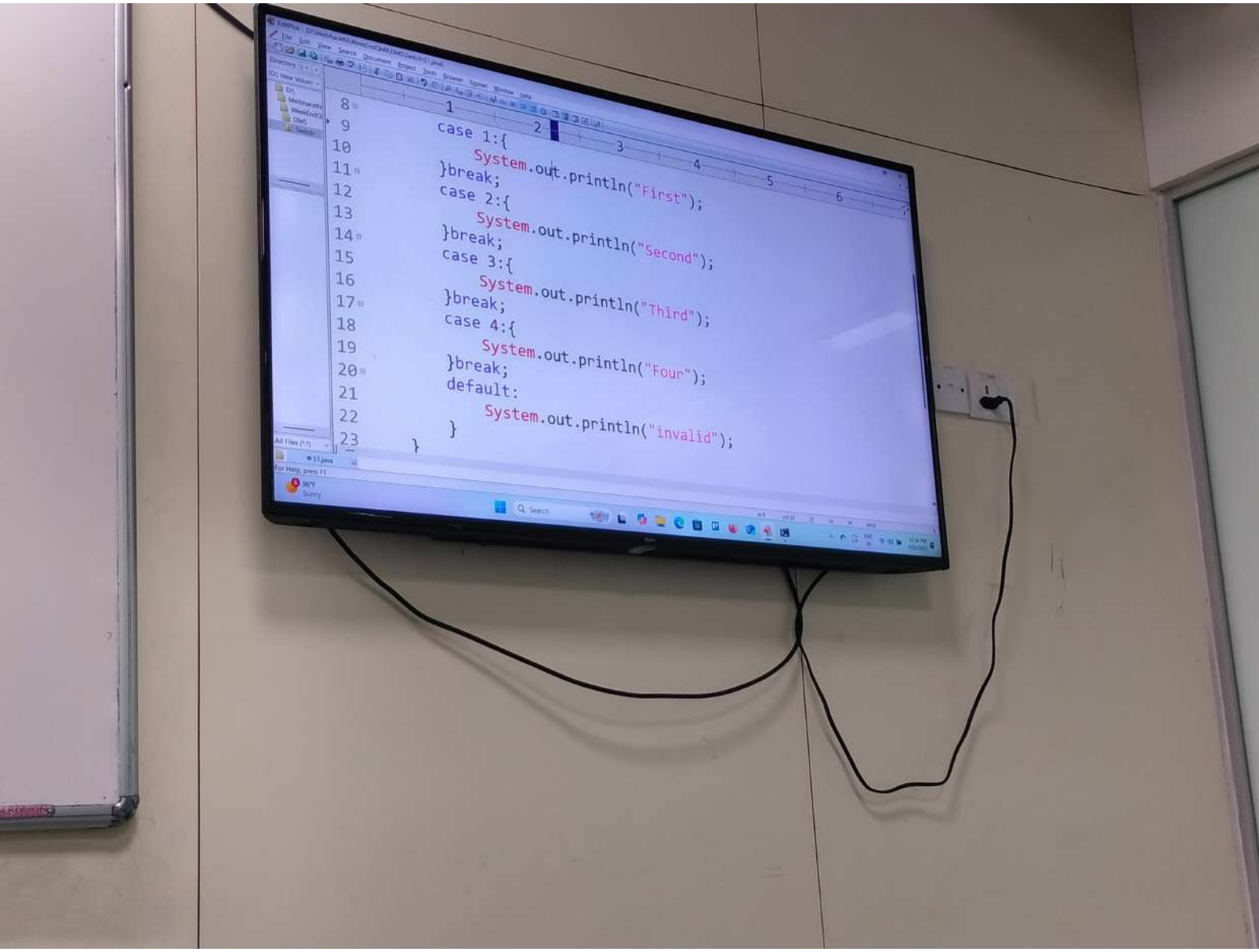

class L2
{
public static void main(String[] args)
{
String food="GobiManchurian";
double price=290;
boolean afford=(food=="GobiManchurian")||(price<=190);
System.out.println("Is it possible to Buy : "+afford);
}
}

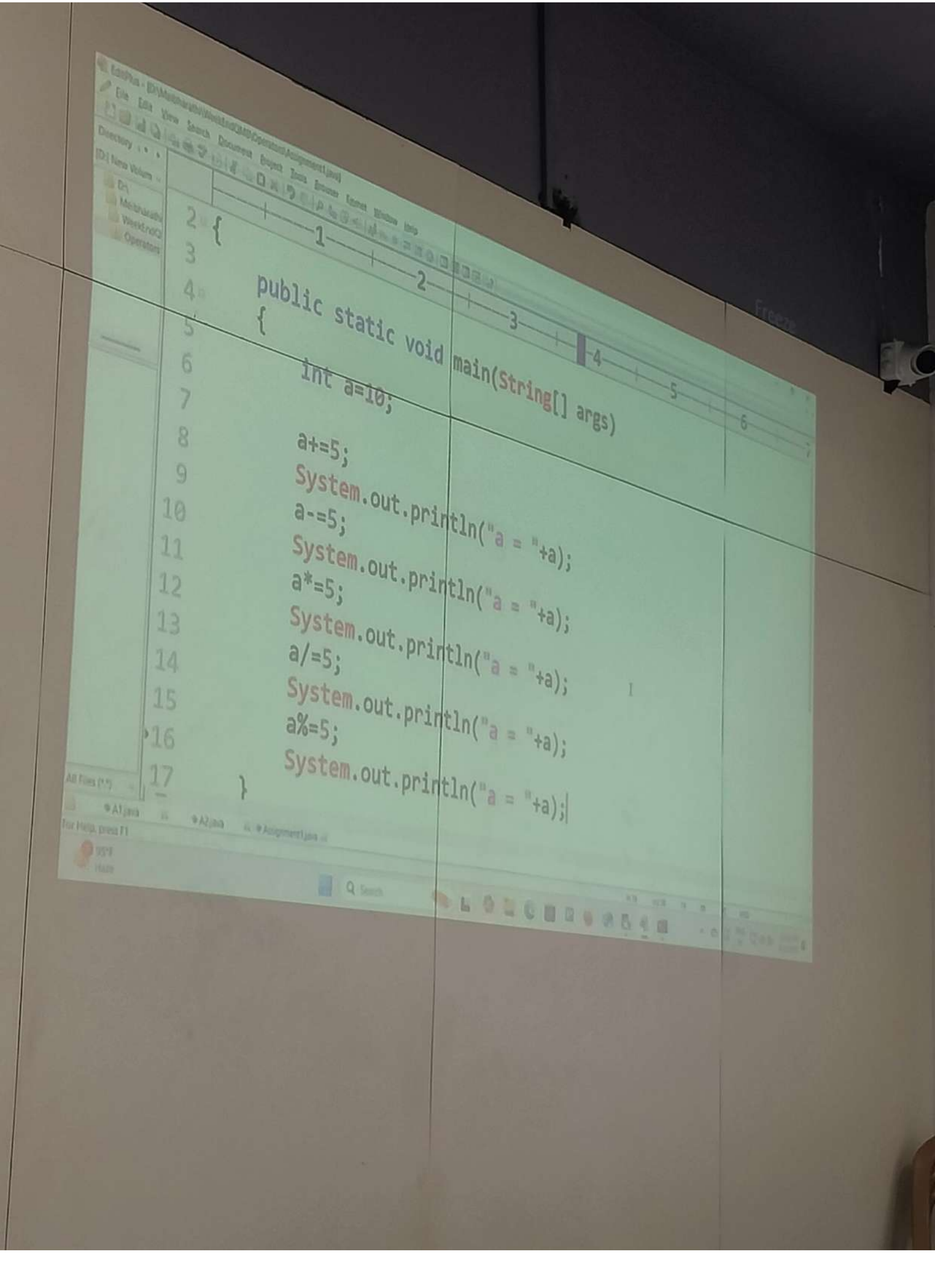

class L3
{
public static void main(String[] args)
{
System.out.println(!true);//false
System.out.println(!false);//true
}
}

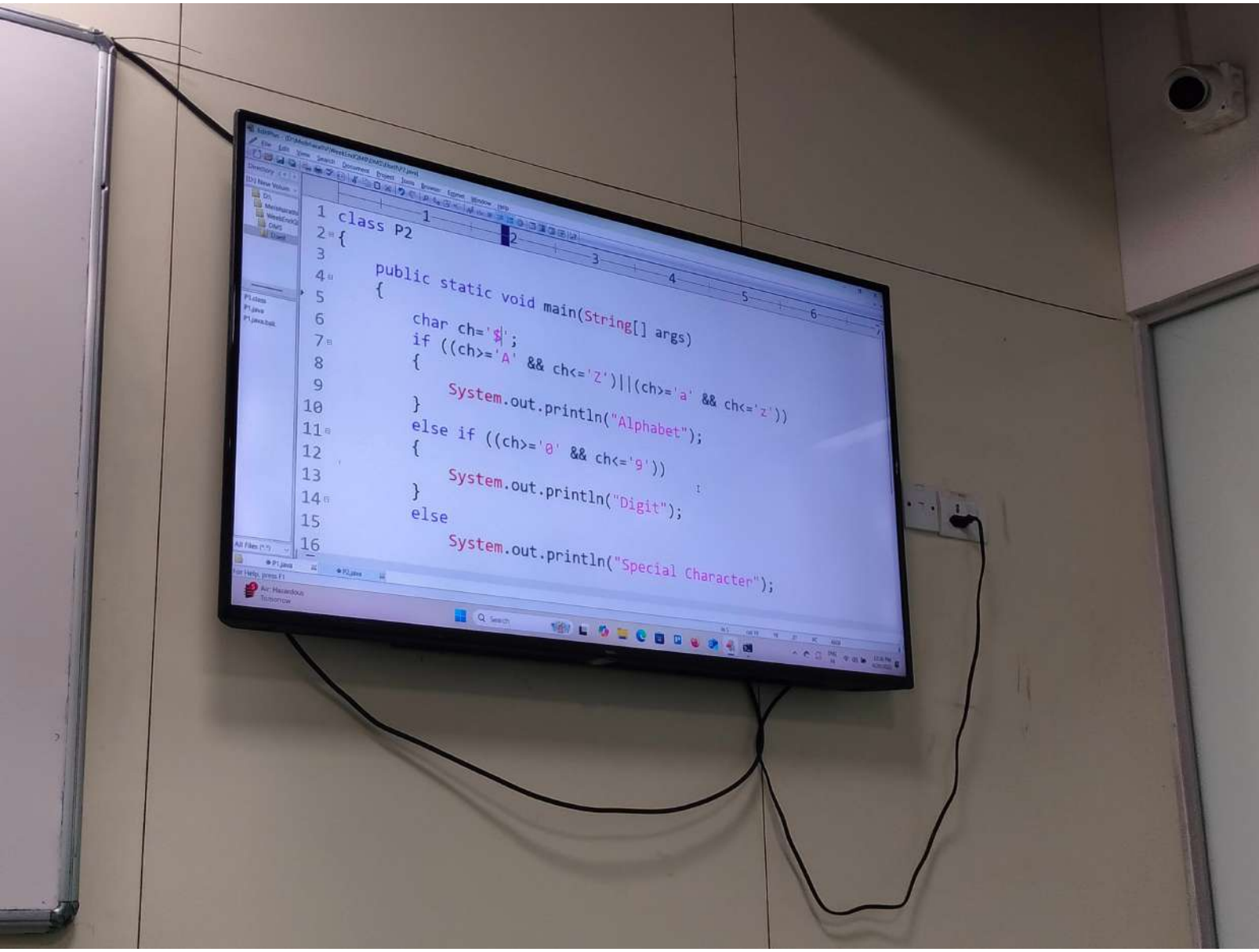

class Shop
{
public static void main(String[] args)
{
double shoe1=3250.50,shoe2=1890.0;
double wallet1=3100.00,wallet2=2000;
//check the Condition
boolean afford1=(shoe1 <= wallet1);//false
boolean afford2=(shoe2 <= wallet2);//true
System.out.println("Afforable price for shoe1 "+afford1);
System.out.println("Afforable price for shoe2 "+afford2);
}
}

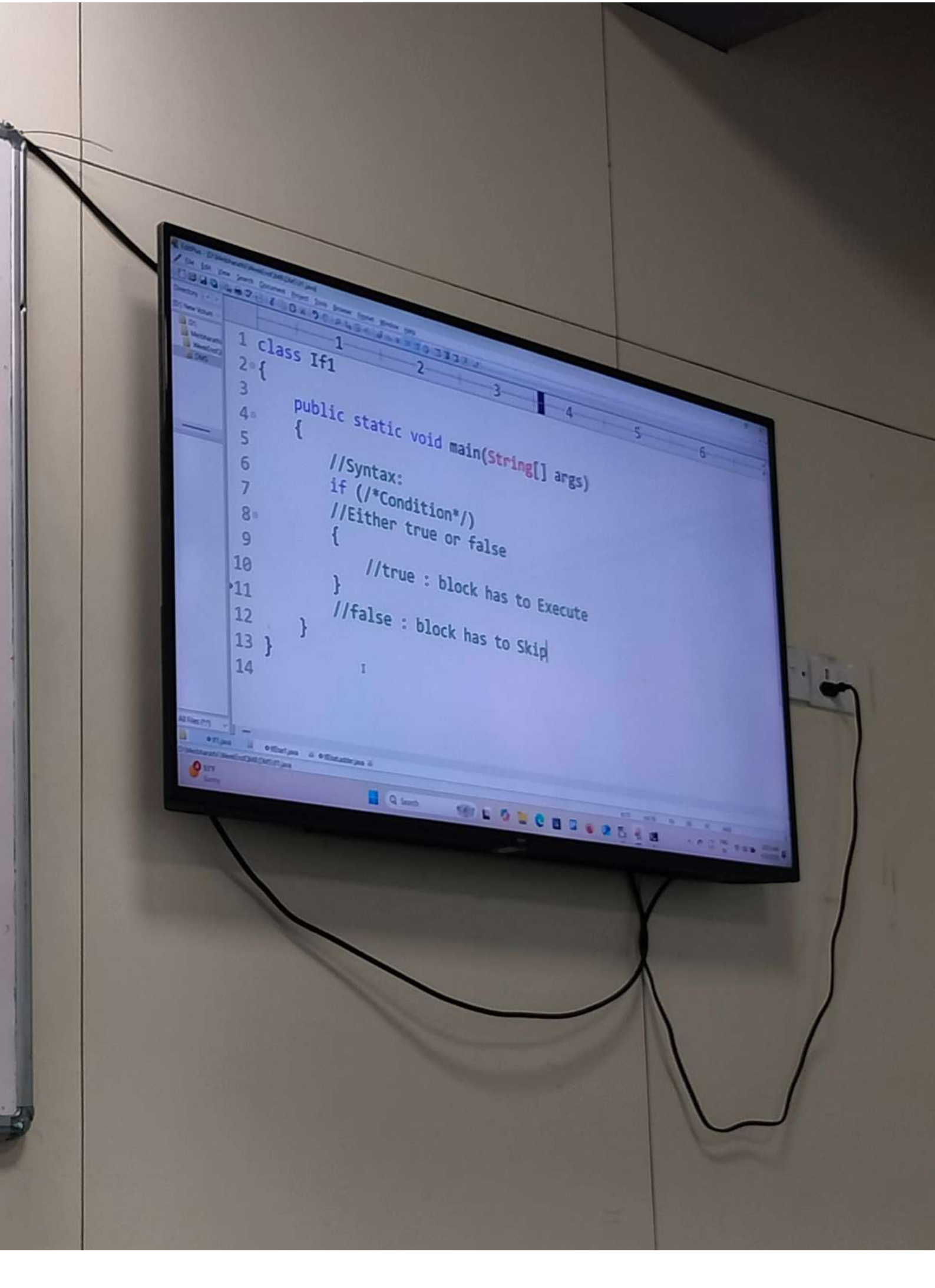

class IfElse1
{
public static void main(String[] args)
{
//Syntax:
if (/*Condition*/)
//Either true or false
{
//true : block has to Execute
}
//false :if block has to Skip
else
{
//else block has to Execute
}
}

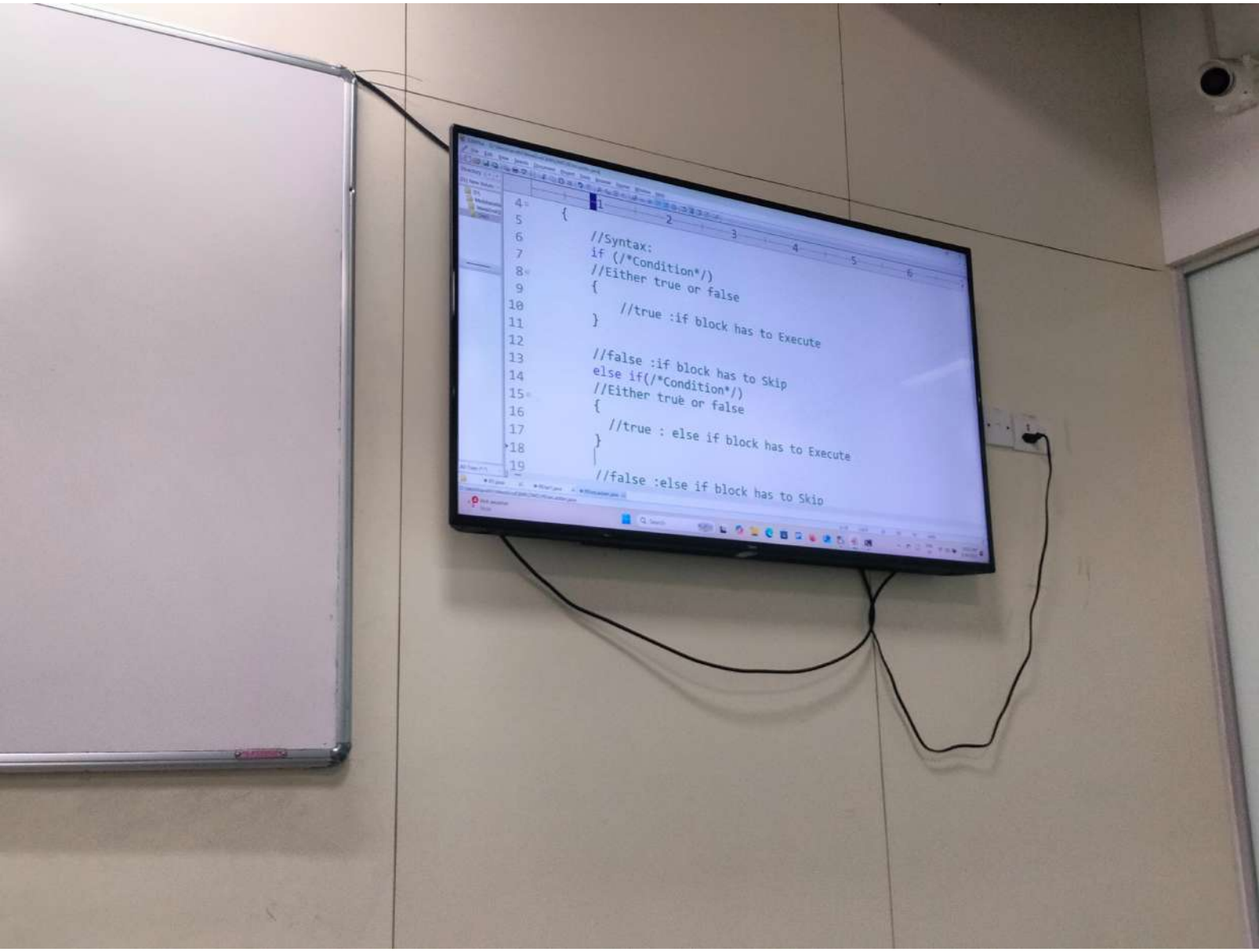

class S1
{
public static void main(String[] args)
{
int option=3;
switch (option)
{
case 1:{
System.out.println("First");
}break;
case 2:{
System.out.println("Second");
}break;
case 3:{
System.out.println("Third");
}break;

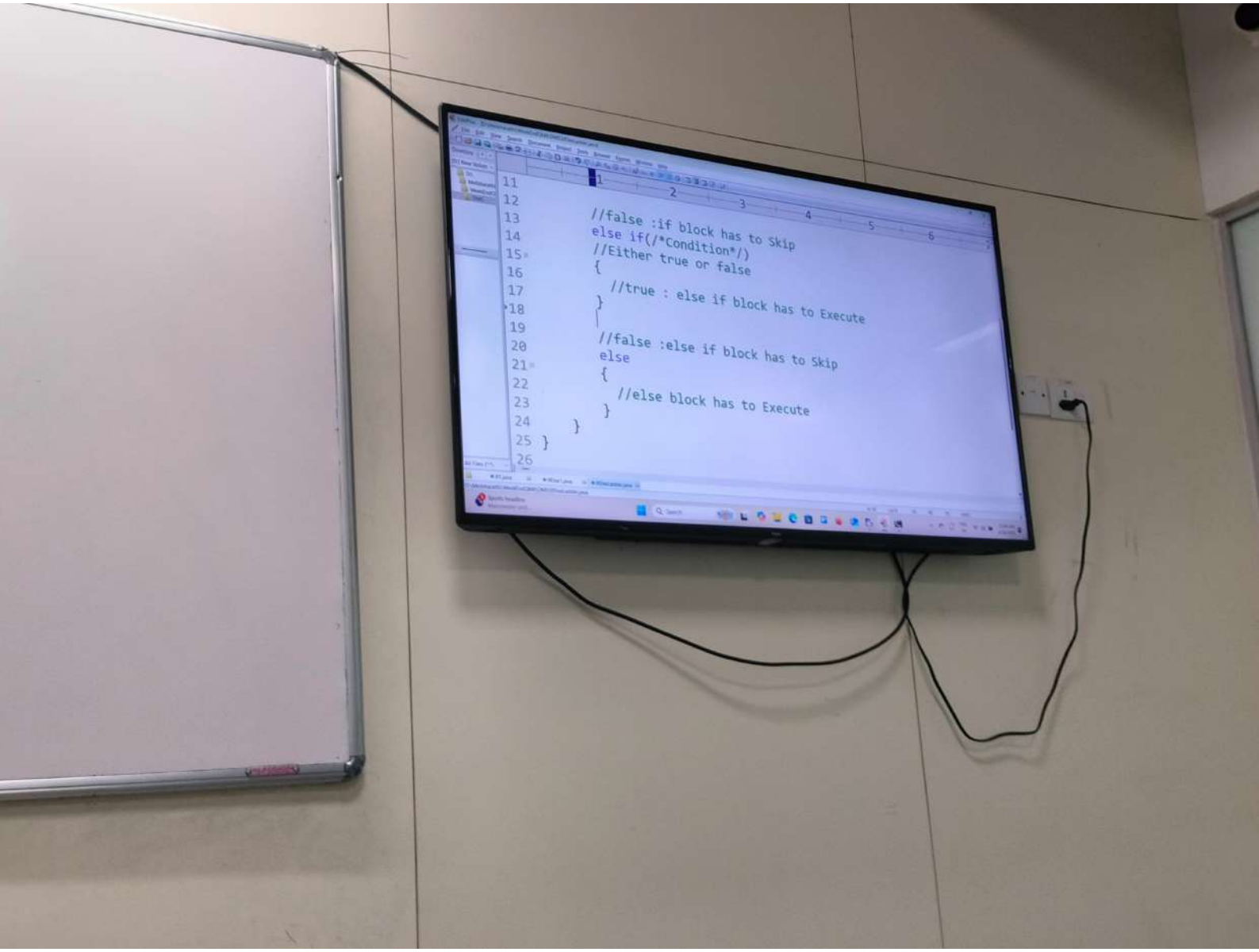

case 1:{
System.out.println("First");
}break;
case 2:{
System.out.println("Second");
}break;
case 3:{
System.out.println("Third");
}break;
case 4:{
System.out.println("Four");
}break;
default:
System.out.println("invalid");
}
}

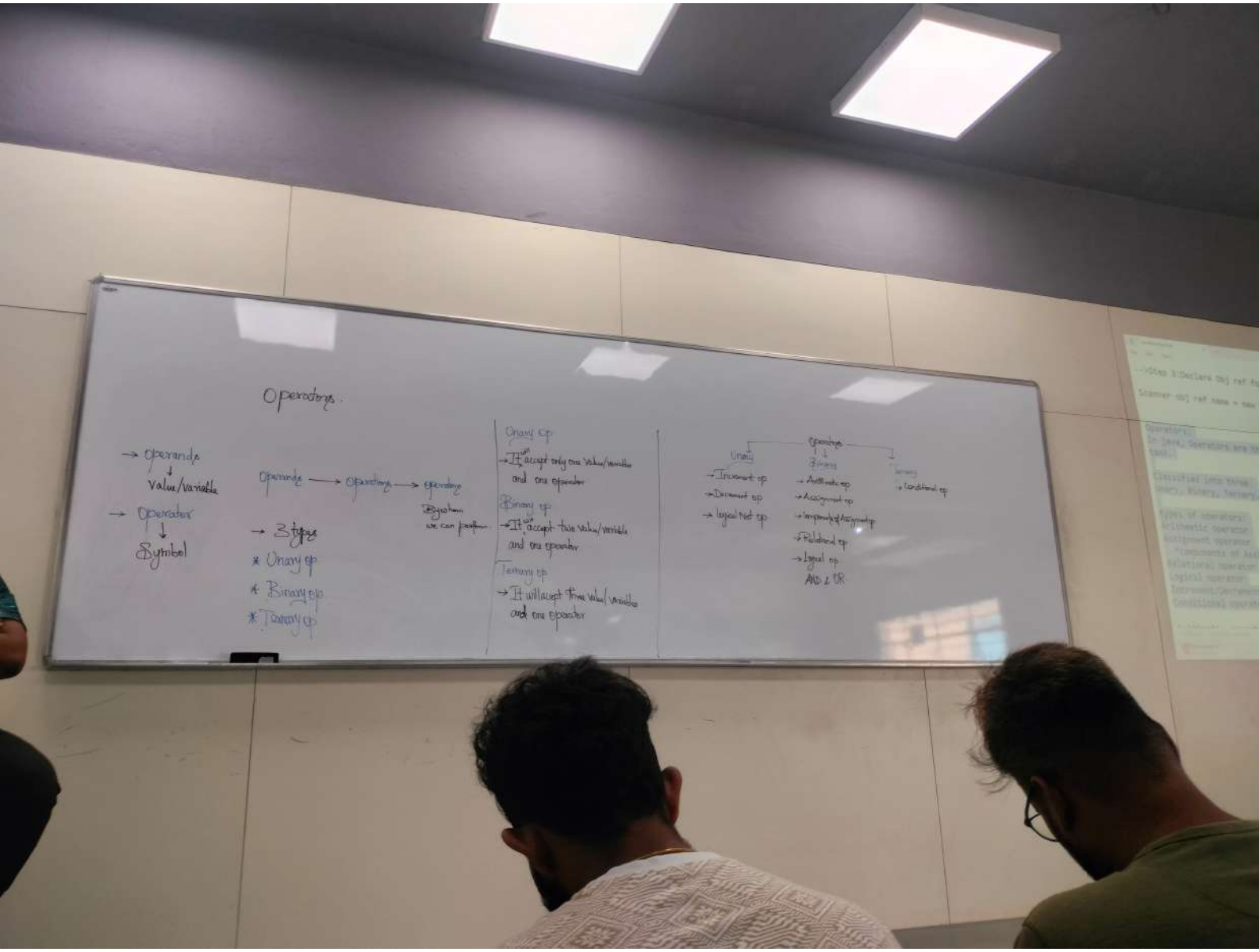

{
public static void main(String[] args)
{
int a=10;
a+=5;
System.out.println("a = "+a);
a-=5;
System.out.println("a = "+a);
a*=5;
System.out.println("a = "+a);
a/=5;
System.out.println("a = "+a);
a%=5;
System.out.println("a = "+a);
}

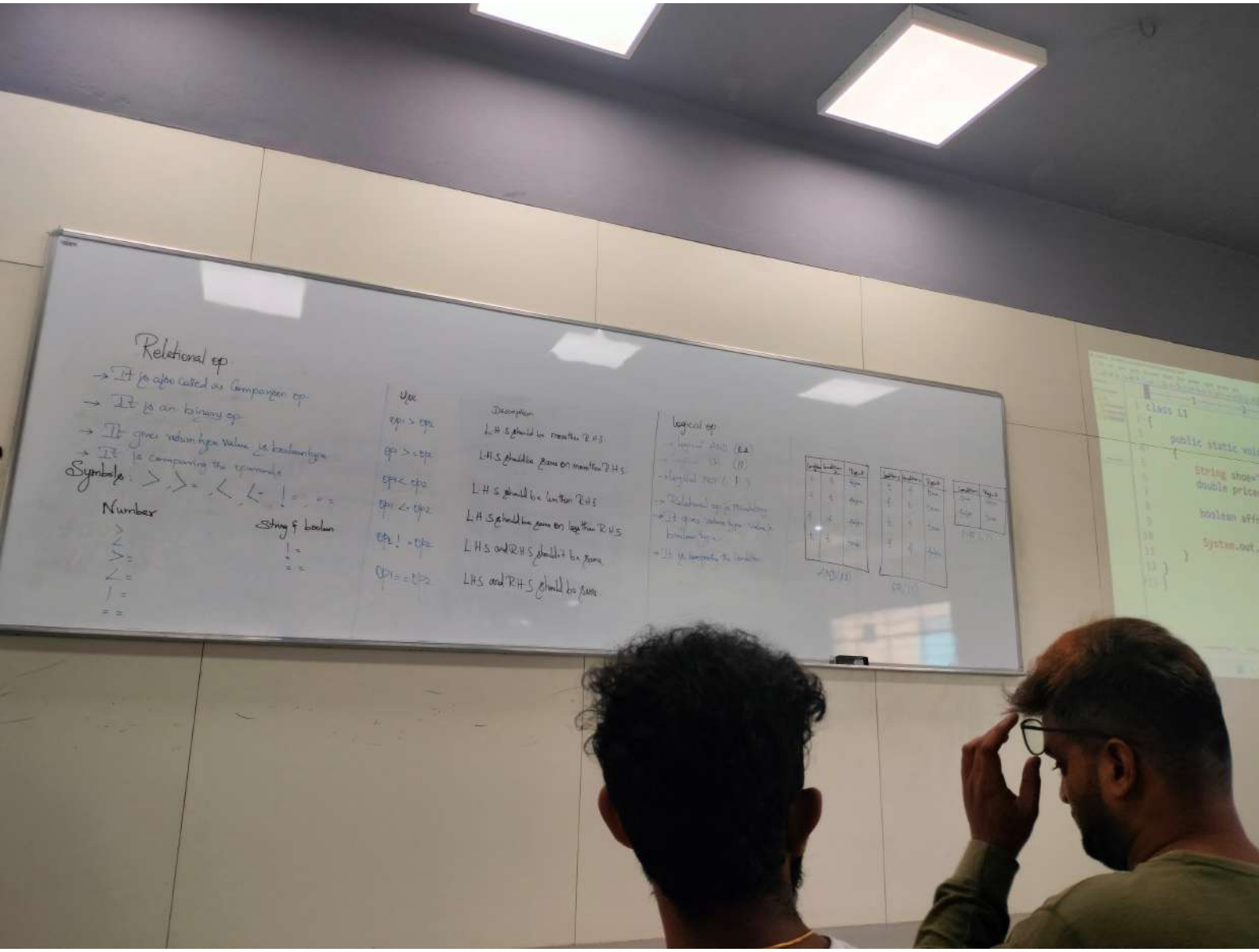

class P2
{
public static void main(String[] args)
{
char ch='$';
if ((ch>='A' && ch<='Z')||(ch>='a' && ch<='z'))
{
System.out.println("Alphabet");
}
else if ((ch>='0' && ch<='9'))
{
System.out.println("Digit");
}
else
System.out.println("Special Character");

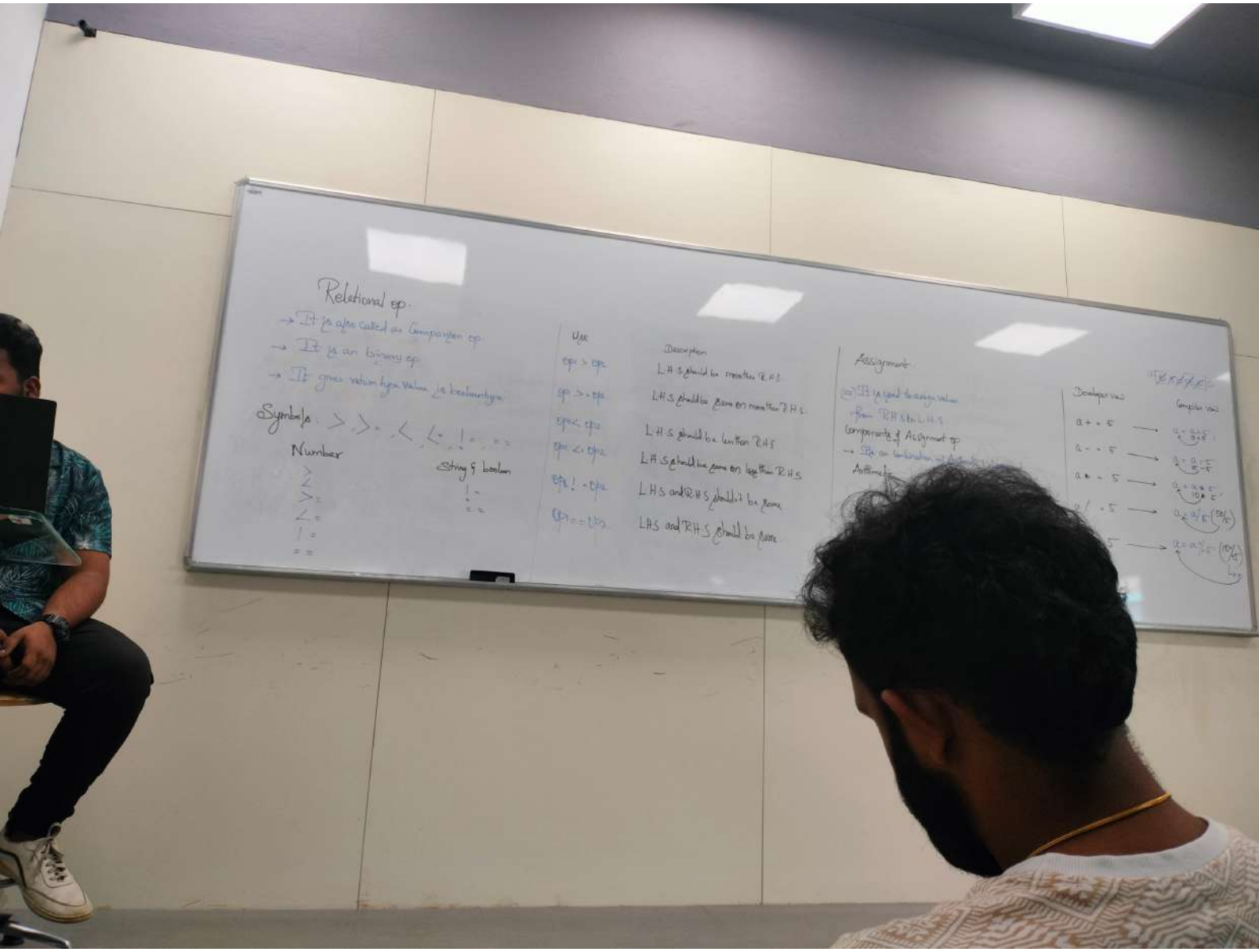

class If1
{
public static void main(String[] args)
{
//Syntax:
if (/*Condition*/)
//Either true or false
{
//true : block has to Execute
}
//false : block has to Skip
}
}


{
//Syntax:
if (/*Condition*/)
//Either true or false
{
//true :if block has to Execute
}
//false :if block has to Skip
else if(/*Condition*/)
//Either true or false
{
//true : else if block has to Execute
}
//false :else if block has to Skip


//false :if block has to Skip
else if(/*Condition*/)
//Either true or false
{
//true : else if block has to Execute
}
//false :else if block has to Skip
else
{
//else block has to Execute
}
}
}


Operators.
→ Operands
↓
Value/variable
→ Operater
↓
Symbol
Operands → Operators → operators
By default we can perform
→ 3 type
* Unary op
* Binary op
* Ternary op
Unary op
→ It will accept only one value/variable and one operator
Binary op
→ It will accept two value/variable and one operator
Ternary op
→ It will accept three value/variable and one operator
Operators
Unary
→ Increment op
→ Decrement op
→ logical not op
Binary
→ Arithmetic op
→ Assignment op
→ Compound Assignment op
→ Relational op
→ logical op
AND & OR
Ternary
→ Conditional op


Relational op
→ It is also called as Comparison op
→ It is an binary op
→ It gives return type value is boolean type
→ It is comparing the operands
Symbols: > >= < <= != ==
Number
>
<
>=
<=
!=
==
String & boolan
!=
==
Use
Description
logical op


Relational op.
→ It is also called as Comparison op.
→ It is an binary op.
→ It gives return type value is boolean type.
Symbols: > , >= , < , <= , != , ==
Number
>
>=
<
<=
!=
==
String & boolean
!=
==
Use
Description
op1 > op2
L.H.S should be greater than R.H.S
op1 >= op2
L.H.S should be same or greater than R.H.S
op1 < op2
L.H.S should be lesser than R.H.S
op1 <= op2
L.H.S should be same or lesser than R.H.S
op1 != op2
L.H.S and R.H.S should not be same
op1 == op2
L.H.S and R.H.S should be same
Assignment
Developer view
Compiler view

Mei bharathi ravi

offlin

These chats are monitored by admin

12/4/2026

```
import java.util.Scanner;
class W1
{
 public static void main(String[] args)
 {
  Scanner s=new Scanner(System.in);
  System.out.print("Enter the Byte Value : ");
  byte b1=s.nextByte();
  short s1=b1;
  int i=b1;
  long l=b1;
  float f=b1;
  double d=b1;
  System.out.println(s1+"\n"+i+"\n"+l+"\n"+f+"\n"+d);
  s.close();
 }
}
```

QCR-SJFCJE-M8 01:11 PM

```
import java.util.Scanner;
class N1
{
 public static void main(String[] args)
 {
  Scanner s=new Scanner(System.in);
  System.out.print("Enter the Double Value : ");
  double d=s.nextDouble();
  float f=(float)d;
  long l=(long)d;
  int i=(int)d;
  short s1=(short)d;
```

Message...

offline

These chats are monitored by admin

12/4/2025

```
    s.close();
  }
}
```

QCR-SJFCJE-M8 01:11 PM

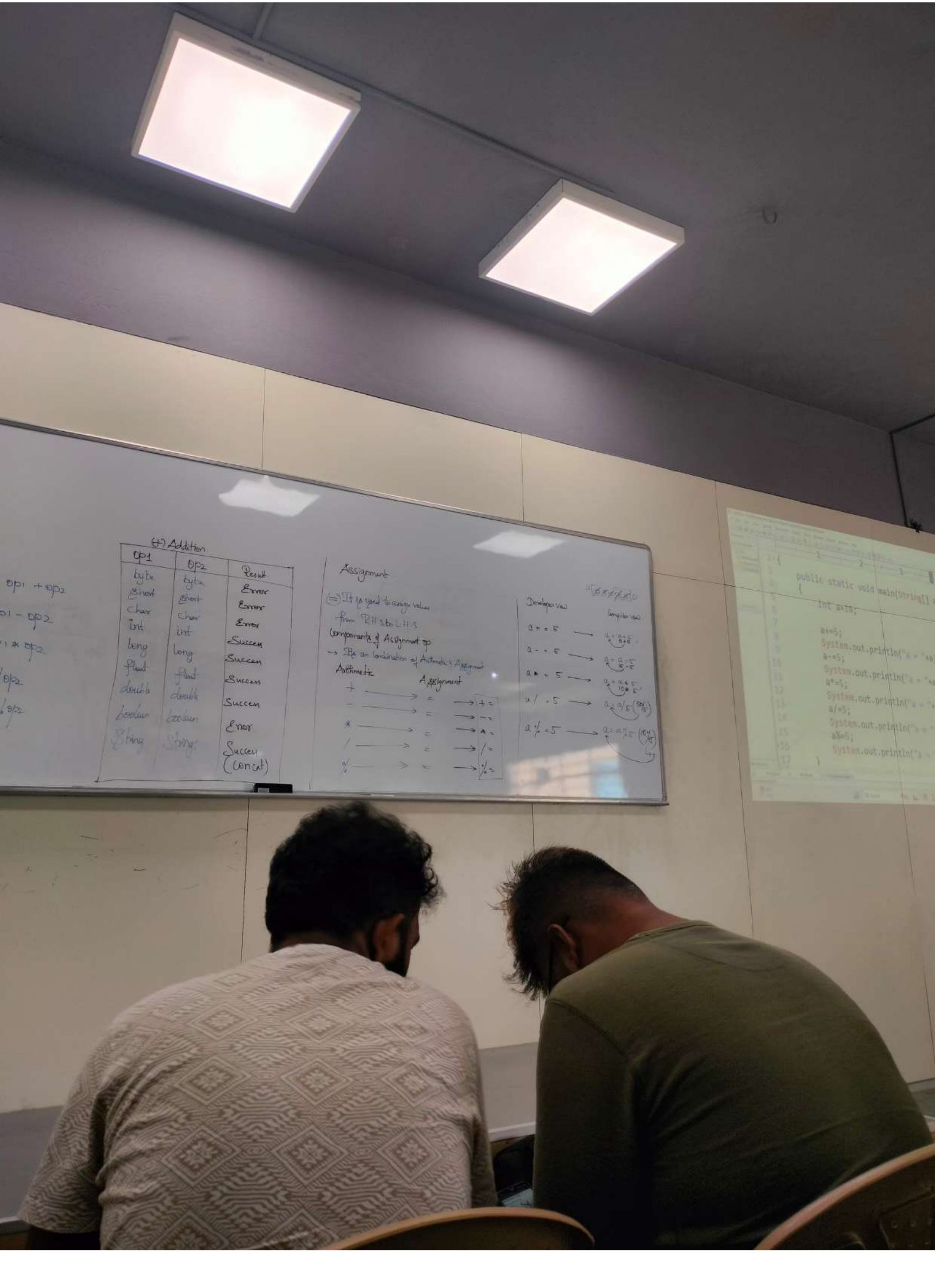
```
import java.util.Scanner;
class N1
{
 public static void main(String[] args)
 {
  Scanner s=new Scanner(System.in);
  System.out.print("Enter the Double Value : ");
  double d=s.nextDouble();
  float f=(float)d;
  long l=(long)d;
  int i=(int)d;
  short s1=(short)d;
  byte b=(byte)d;
  System.out.println(f+"\n"+l+"\n"+i+"\n"+s1+"\n"+b);
 s.close();
 }

}
```

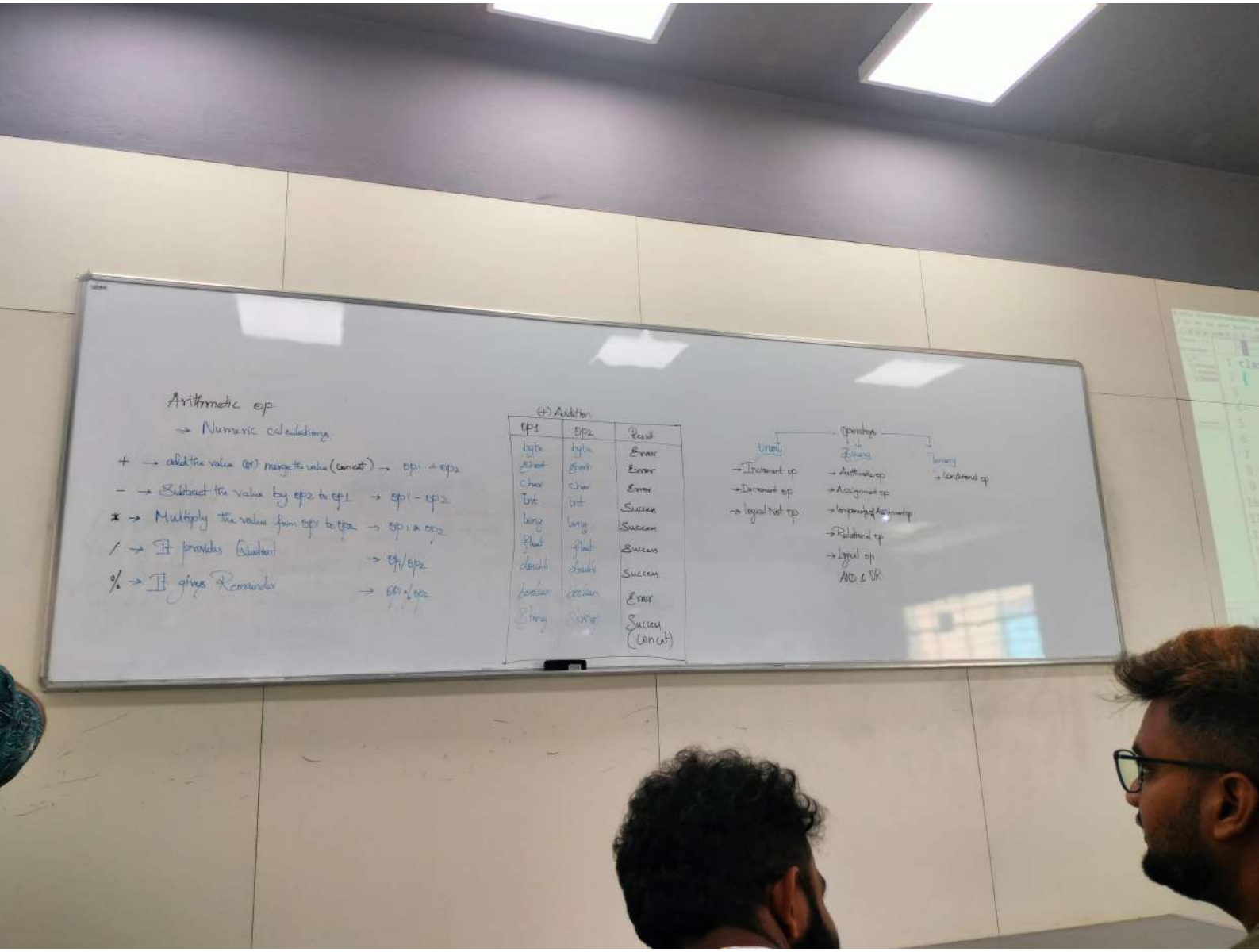
QCR-SJFCJE-M8 01:11 PM

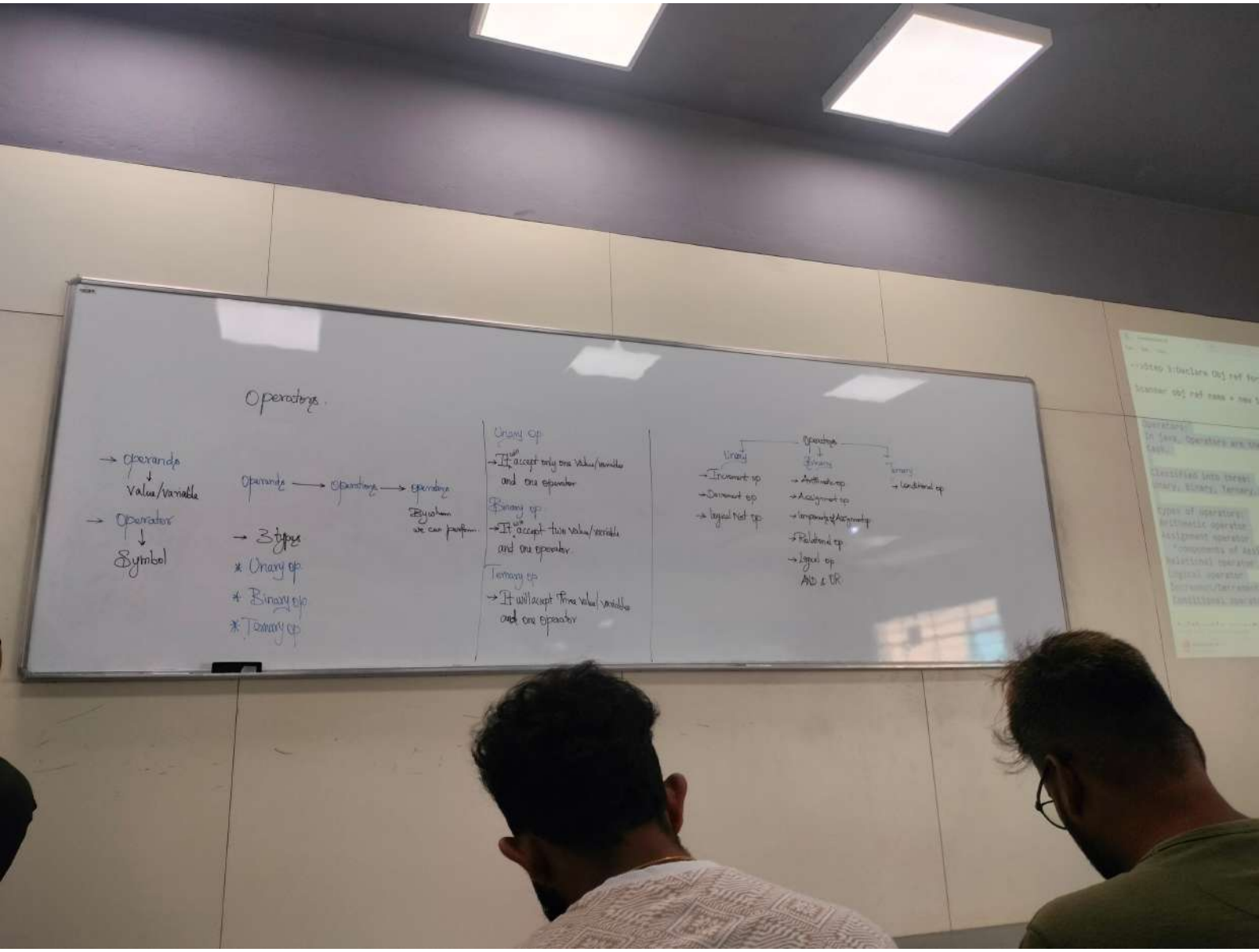
Assignment:
-->WAP to convert value into char datatype literals.
-->Execute remaining datatypes in widening and narrowing

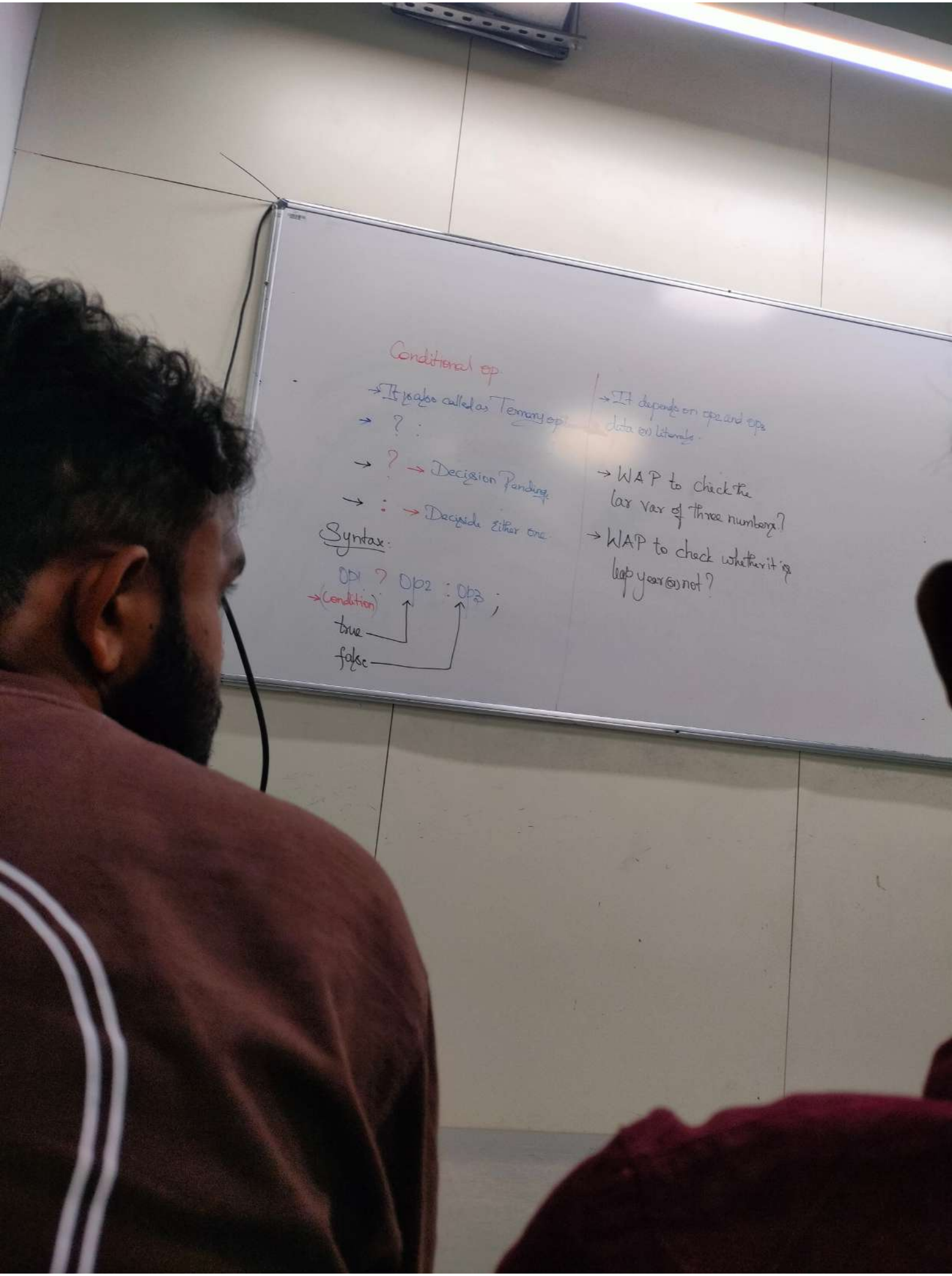
QCR-SJFCJE-M8 01:12 PM


(+) Addition
Op1
Op2
Result
byte
byte
Error
short
short
Error
char
char
Error
int
int
Success
long
long
Success
float
float
Success
double
double
Success
boolean
boolean
Error
String
String
Success (concat)
Assignment
(=) It is used to assign value from RHS to LHS
Components of Assignment op
→ It's an combination of Arithmetic & Assignment
Arithmetic
Assignment
Developer view
Compiler view
a+=5
a-=5
a*=5
a/=5
a%=5

| Operators | Associativity | Type |
|---|---|---|
| ++ -- | Right to left | Unary postfix |
| ++ -- + - ~ ! (type) | Right to left | Unary prefix |
| * / % | Left to right | Multiplicative |
| + - | Left to right | Additive |
| << >> >>> | Left to right | Shift |
| < <= > >= | Left to right | Relational |
| == != | Left to right | Equality |
| & | Left to right | Boolean Logical AND |
| ^ | Left to right | Boolean Logical Exclusive OR |
| \| | Left to right | Boolean Logical Inclusive OR |
| && | Left to right | Conditional AND |
| \|\| | Left to right | Conditional OR |
| ?: | Right to left | Conditional |
| = += -= *= /= %= | Right to left | Assignment |


Arithmetic op
→ Numeric calculations
+ → add the value (or) merge the value (concat) → op1 + op2
- → Subtract the value by op2 to op1 → op1 - op2
* → Multiply the value from op1 to op2 → op1 * op2
/ → It provides Quotient → op1/op2
% → It gives Remainder → op1 % op2
(+) Addition
op1 op2 Result
byte byte Error
Short Short Error
Char Char Error
Int Int Success
long long Success
float float Success
double double Success
boolean boolean Error
String String Success (concat)
Operators
Unary
→ Increment op
→ Decrement op
→ logical not op
Binary
→ Arithmetic op
→ Assignment op
→ Comparison op
→ Relational op
→ Logical op
AND & OR
Ternary
→ Conditional op


Operators.
→ operands
↓
Value/variable
→ Operators
↓
Symbol
Operands → Operators → operators
By whom we can perform
→ 3 types
* Unary op
* Binary op
* Ternary op
Unary op
→ It will accept only one Value/variable and one operator
Binary op
→ It will accept two value/variable and one operator.
Ternary op
→ It will accept three Value/variable and one operator
Operators
Unary
→ Increment op
→ Decrement op
→ logical Not op
Binary
→ Arithmetic op
→ Assignment op
→ Relational op
→ Logical op
AND & OR
Ternary
→ Conditional op


Conditional op
→ It is also called as Ternary op
→ ? :
→ ? → Decision Pending
→ : → Decide either one
Syntax:
Op1 ? Op2 : Op3 ;
→ (condition)
true
false
→ It depends on op1 and op2 data ex) literals
→ WAP to check the lar var of three numbers?
→ WAP to check whether it is leap year (or) not?