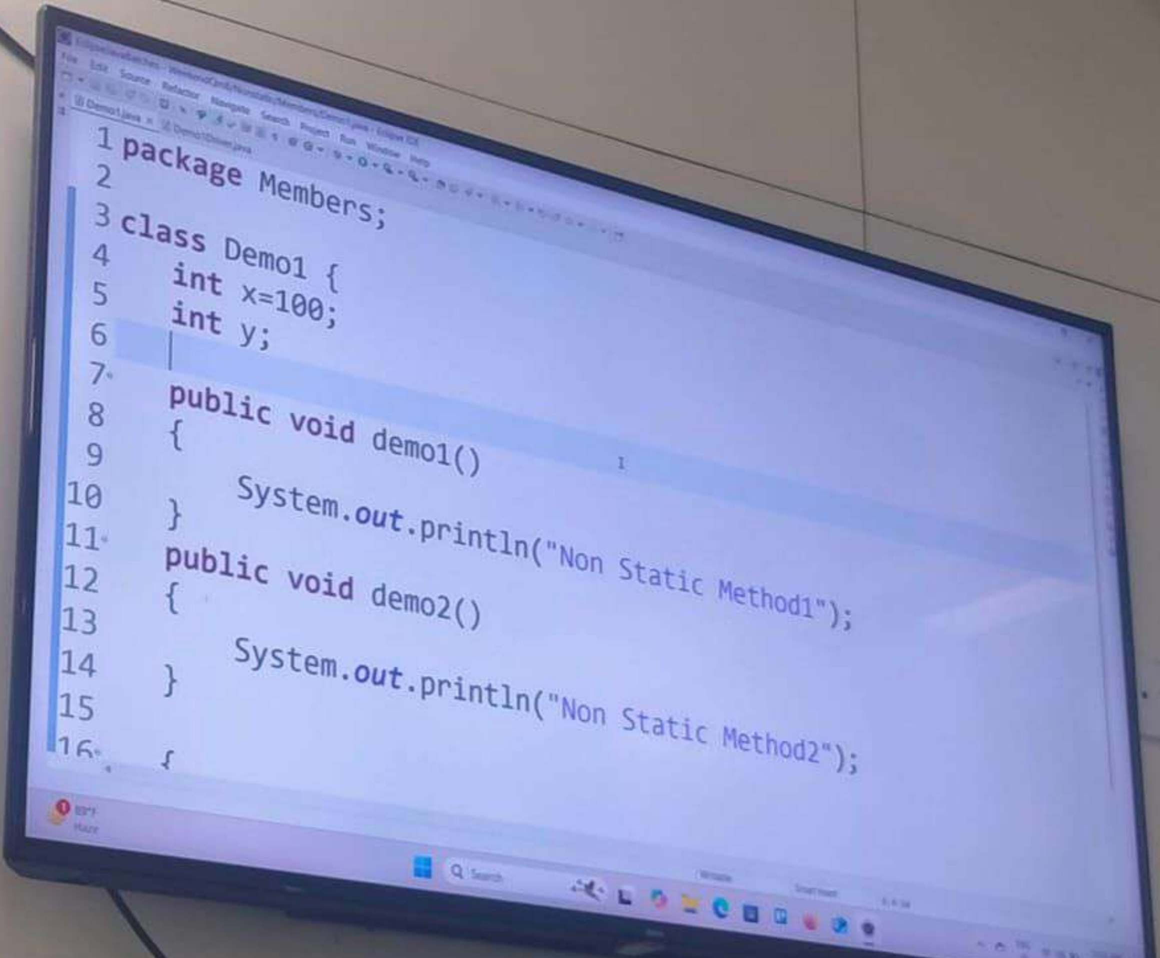
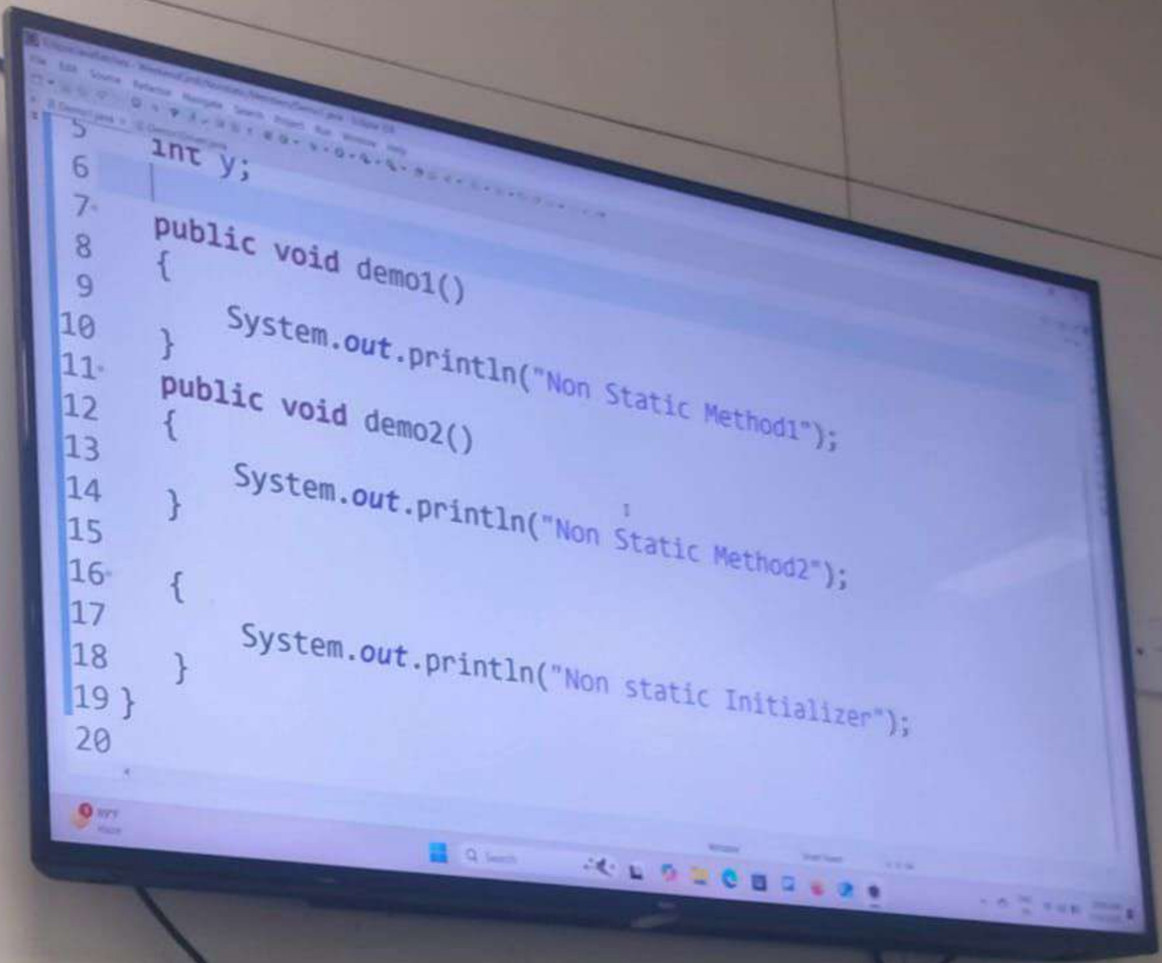
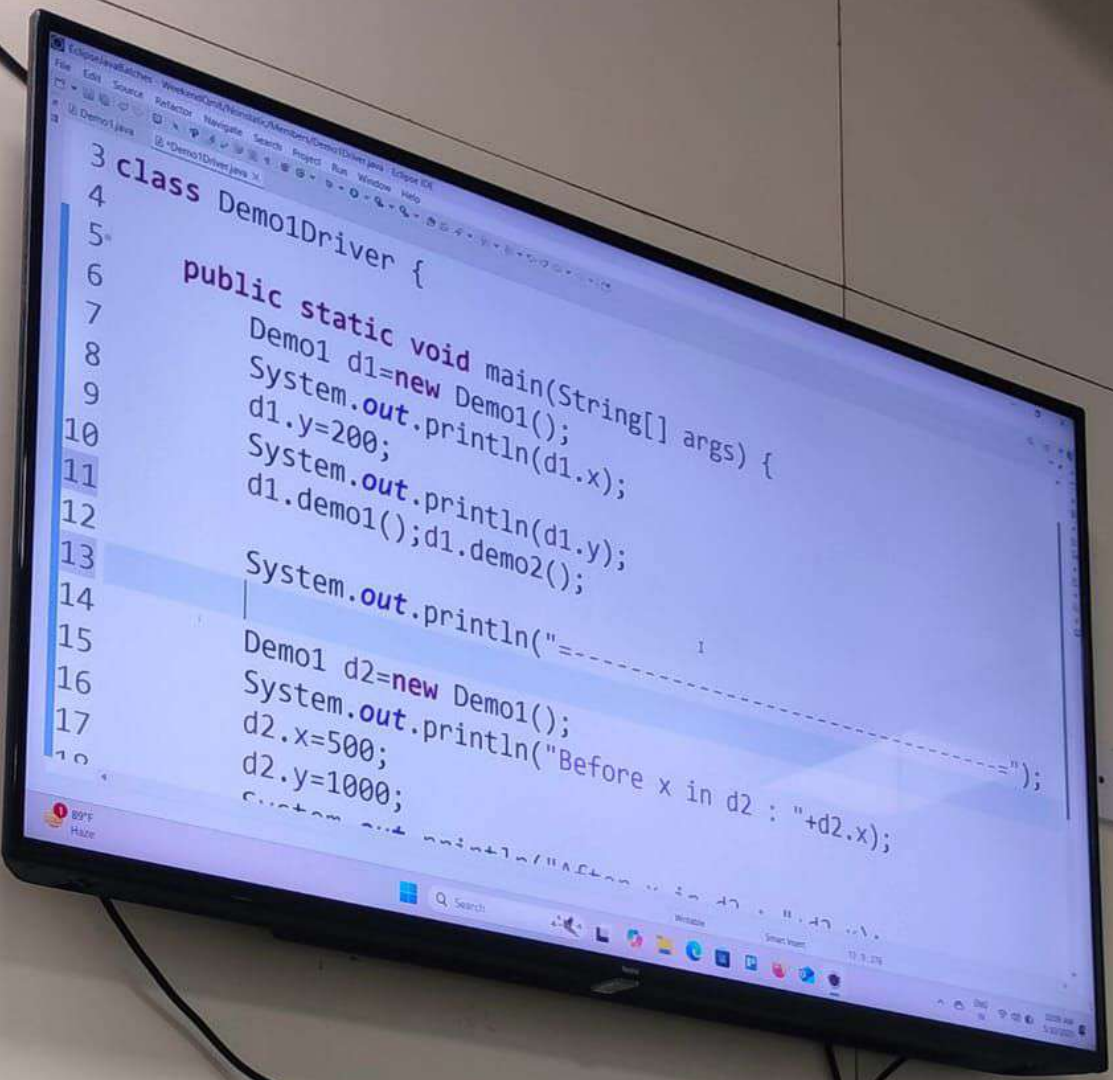


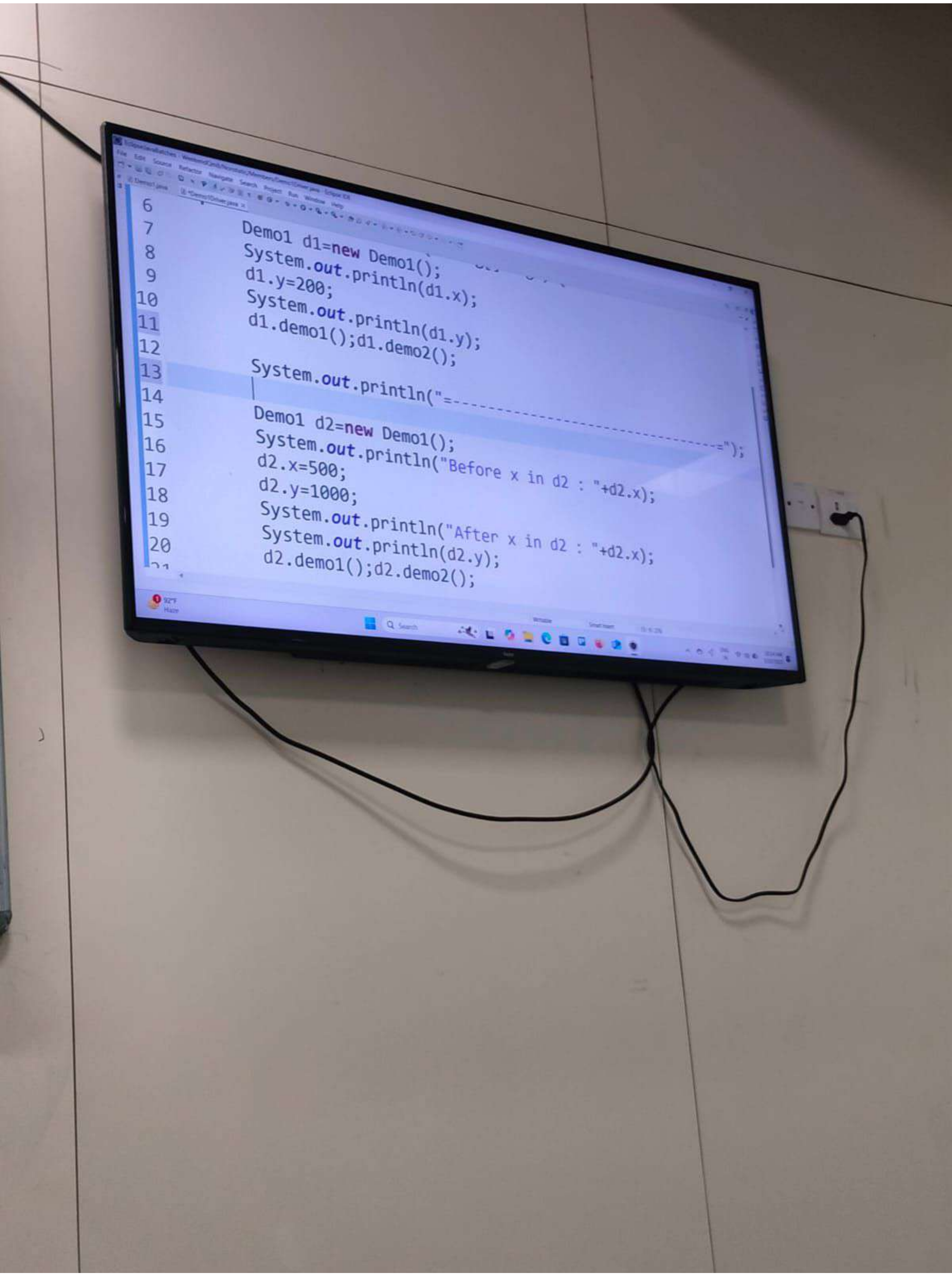
```java
package Members;
class Demo1 {
    int x=100;
    int y;
    
    public void demo1()
    {
        System.out.println("Non Static Method1");
    }
    public void demo2()
    {
        System.out.println("Non Static Method2");
    }

    {
```



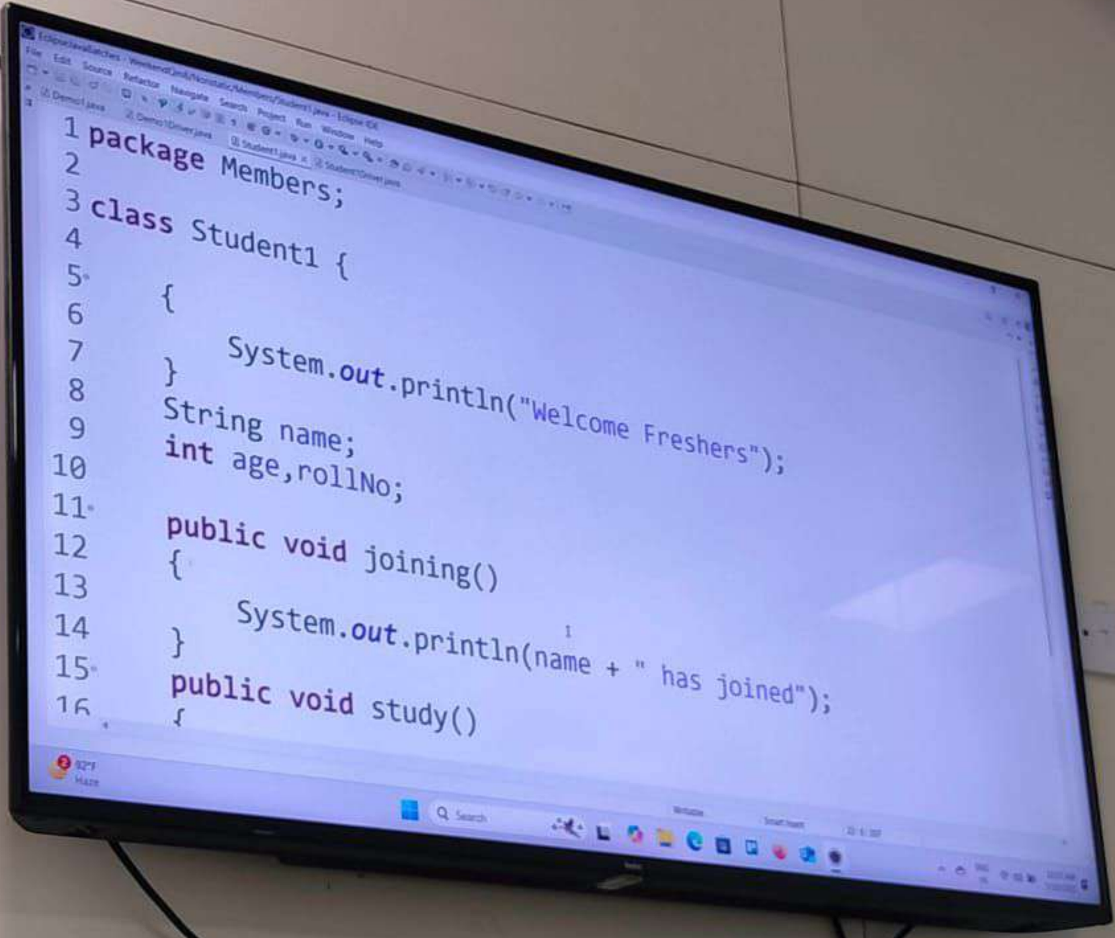
```
	int y;

	public void demo1()
	{
		System.out.println("Non Static Method1");
	}
	public void demo2()
	{
		System.out.println("Non Static Method2");
	}
	{
		System.out.println("Non static Initializer");
	}
}
```


File Edit Source Refactor Navigate Search Project Run Window Help
Demo1.java
*Demo1Driver.java

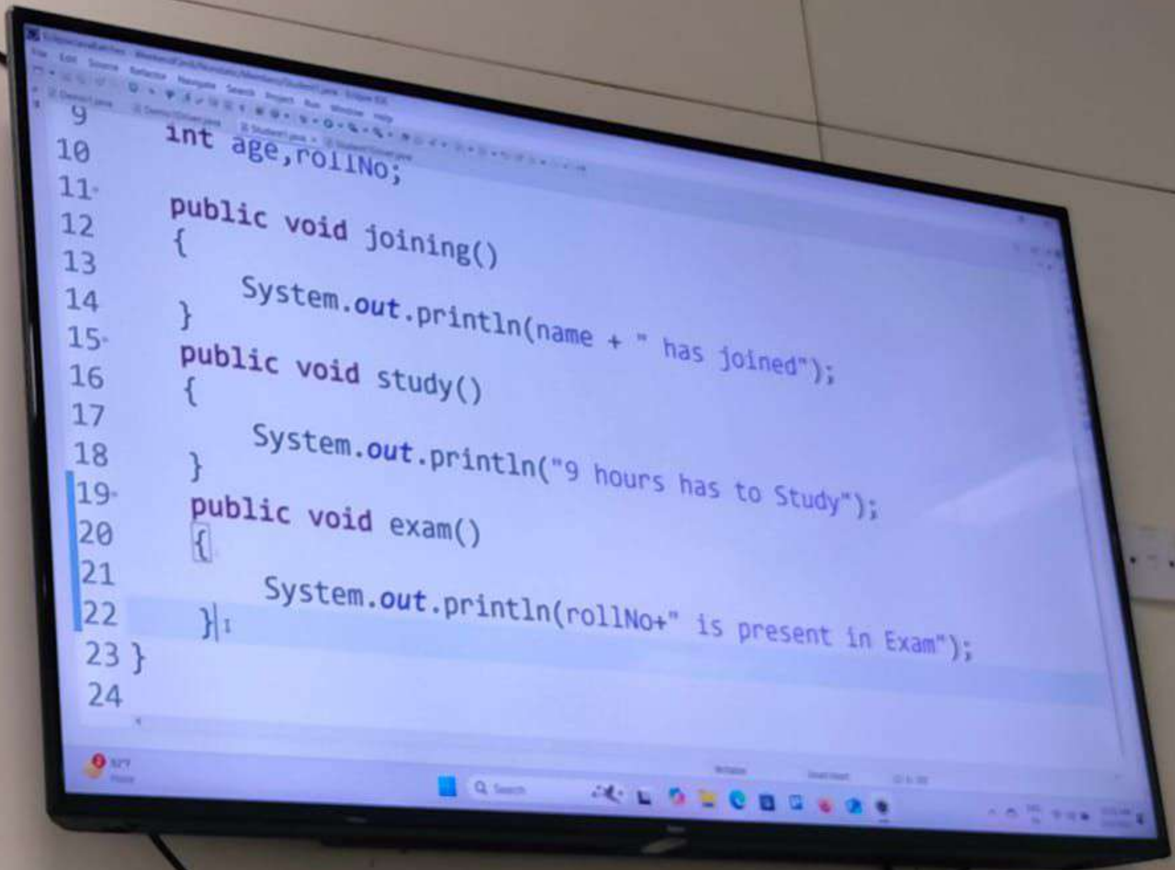
```
class Demo1Driver {

    public static void main(String[] args) {
        Demo1 d1=new Demo1();
        System.out.println(d1.x);
        d1.y=200;
        System.out.println(d1.y);
        d1.demo1();d1.demo2();
        System.out.println("=----------------------------------------=");

        Demo1 d2=new Demo1();
        System.out.println("Before x in d2 : "+d2.x);
        d2.x=500;
        d2.y=1000;
        System.out.println("After x in d2 : "+d2.x);
```



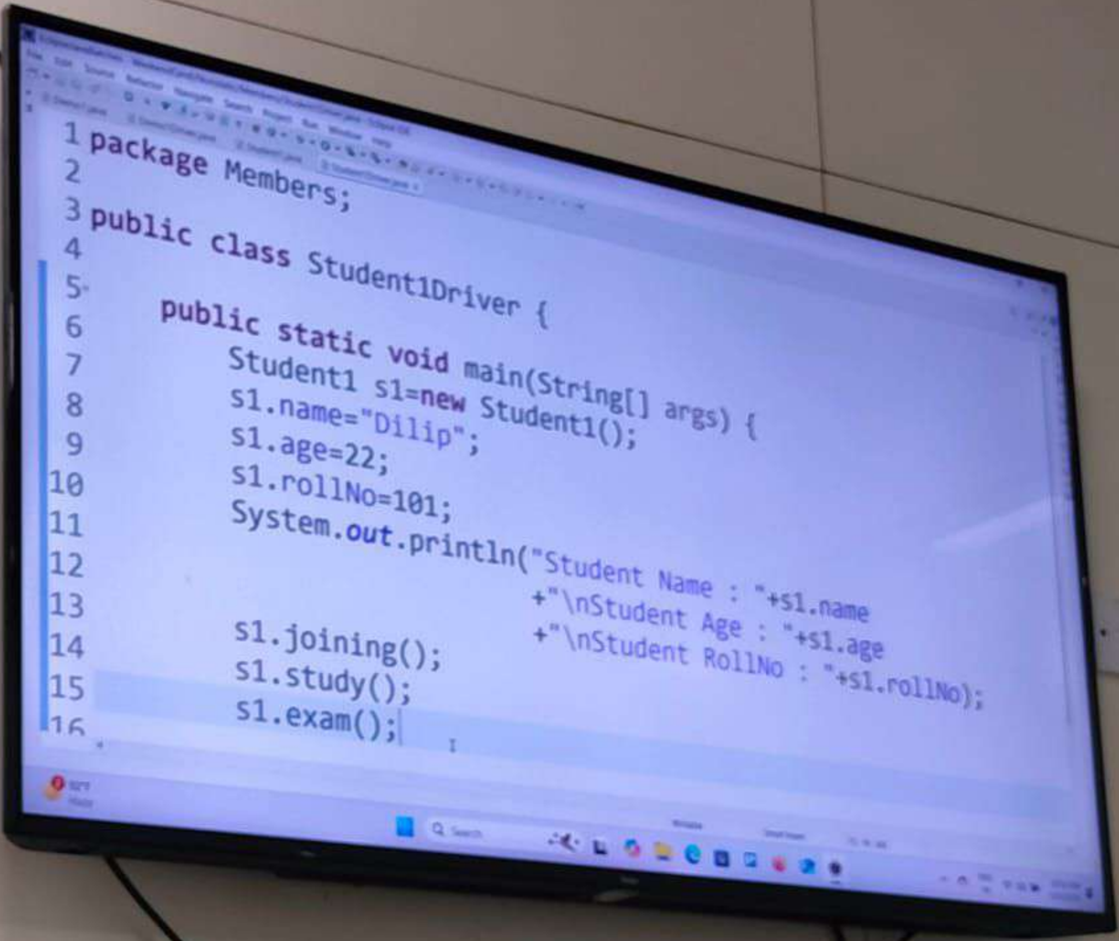
```
Demo1 d1=new Demo1();
System.out.println(d1.x);
d1.y=200;
System.out.println(d1.y);
d1.demo1();d1.demo2();

System.out.println("=-------------------------------------------=");

Demo1 d2=new Demo1();
System.out.println("Before x in d2 : "+d2.x);
d2.x=500;
d2.y=1000;
System.out.println("After x in d2 : "+d2.x);
System.out.println(d2.y);
d2.demo1();d2.demo2();
```



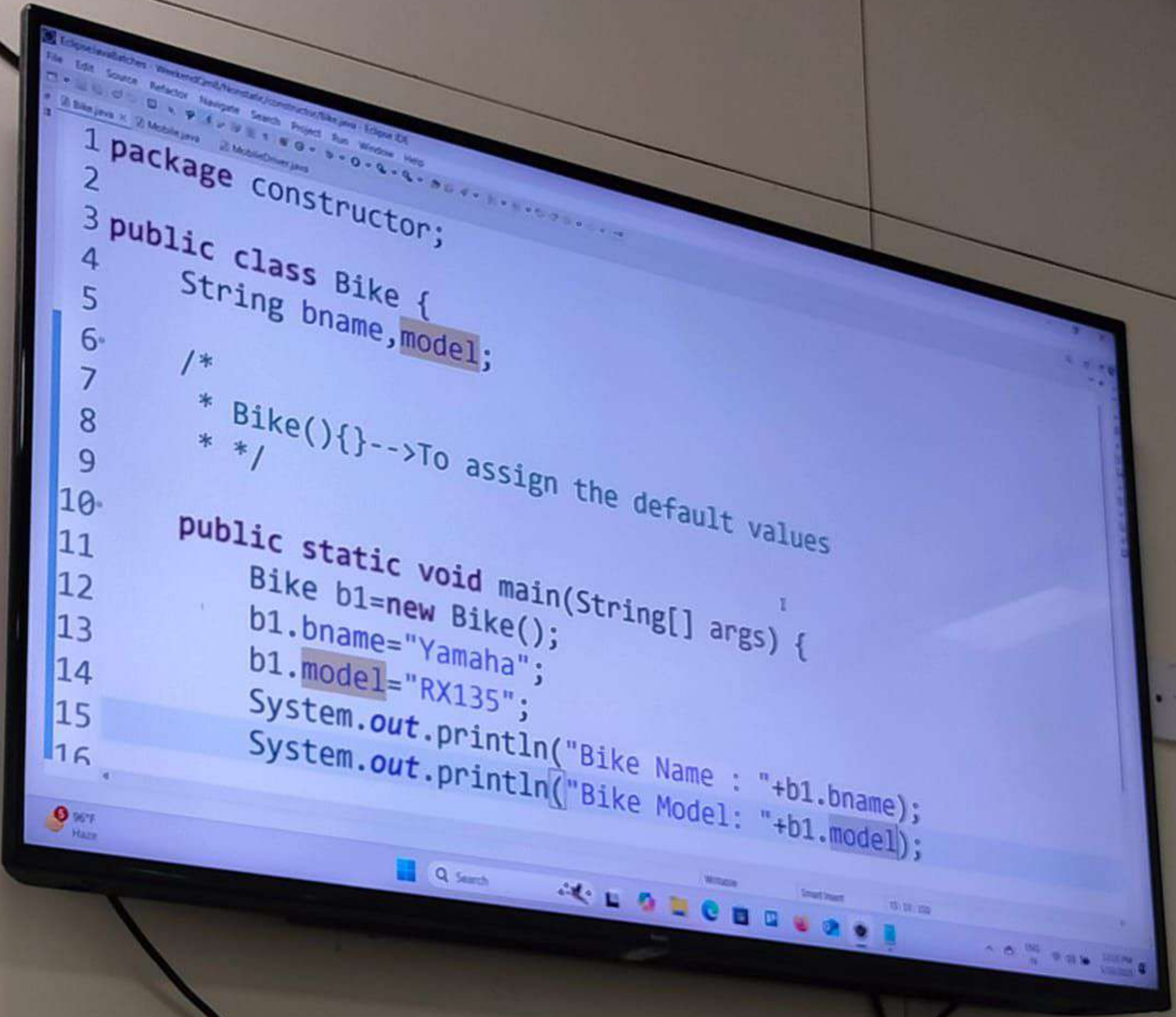
```
package Members;

class Student1 {
    {
        System.out.println("Welcome Freshers");
    }
    String name;
    int age,rollNo;

    public void joining()
    {
        System.out.println(name + " has joined");
    }
    public void study()
    {
```



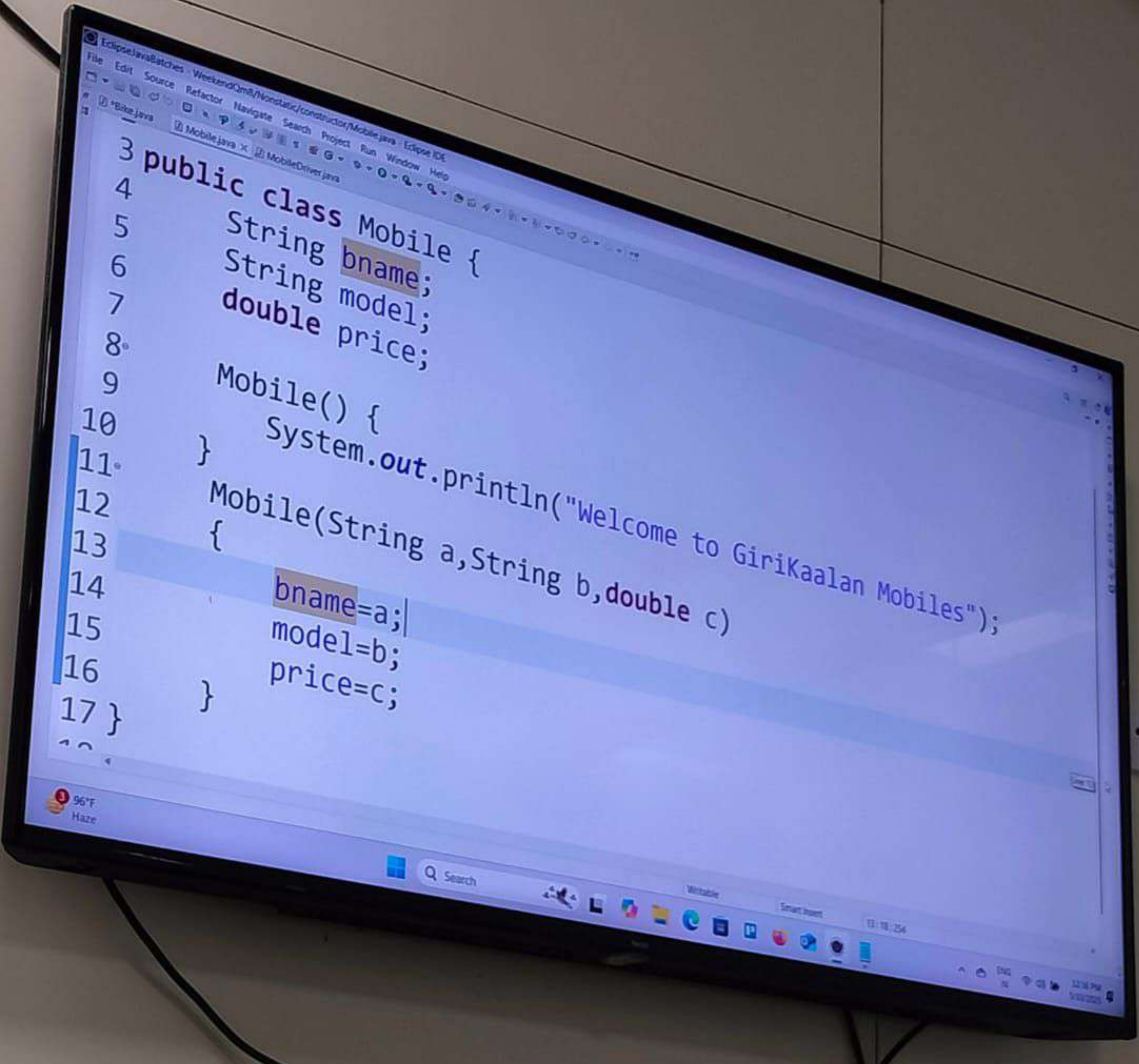
```
    int age,rollNo;

    public void joining()
    {
        System.out.println(name + " has joined");
    }
    public void study()
    {
        System.out.println("9 hours has to Study");
    }
    public void exam()
    {
        System.out.println(rollNo+" is present in Exam");
    }
}
```



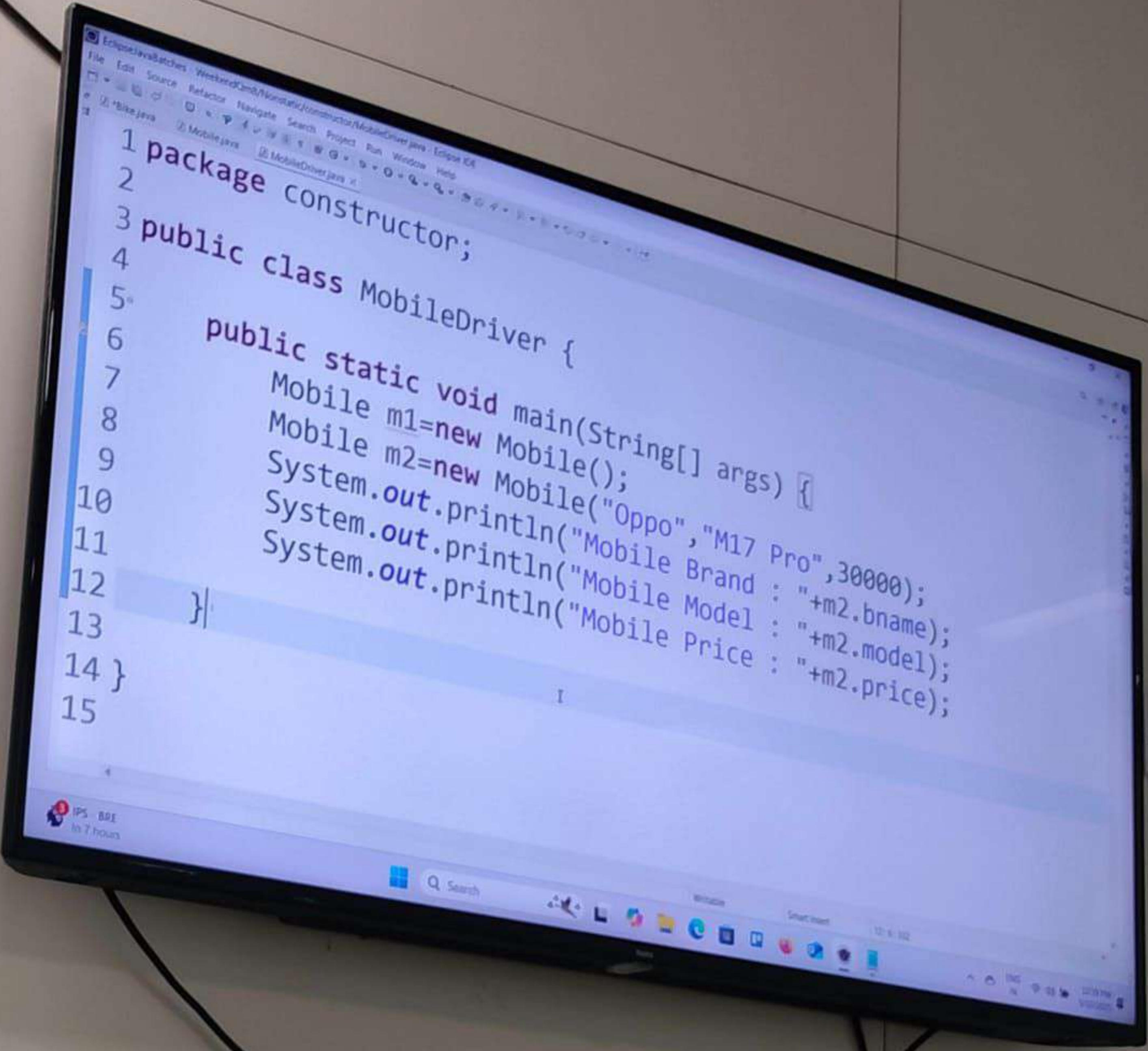
```
package Members;

public class Student1Driver {

    public static void main(String[] args) {
        Student1 s1=new Student1();
        s1.name="Dilip";
        s1.age=22;
        s1.rollNo=101;
        System.out.println("Student Name : "+s1.name
                                +"\nStudent Age : "+s1.age
                                +"\nStudent RollNo : "+s1.rollNo);
        s1.joining();
        s1.study();
        s1.exam();
```



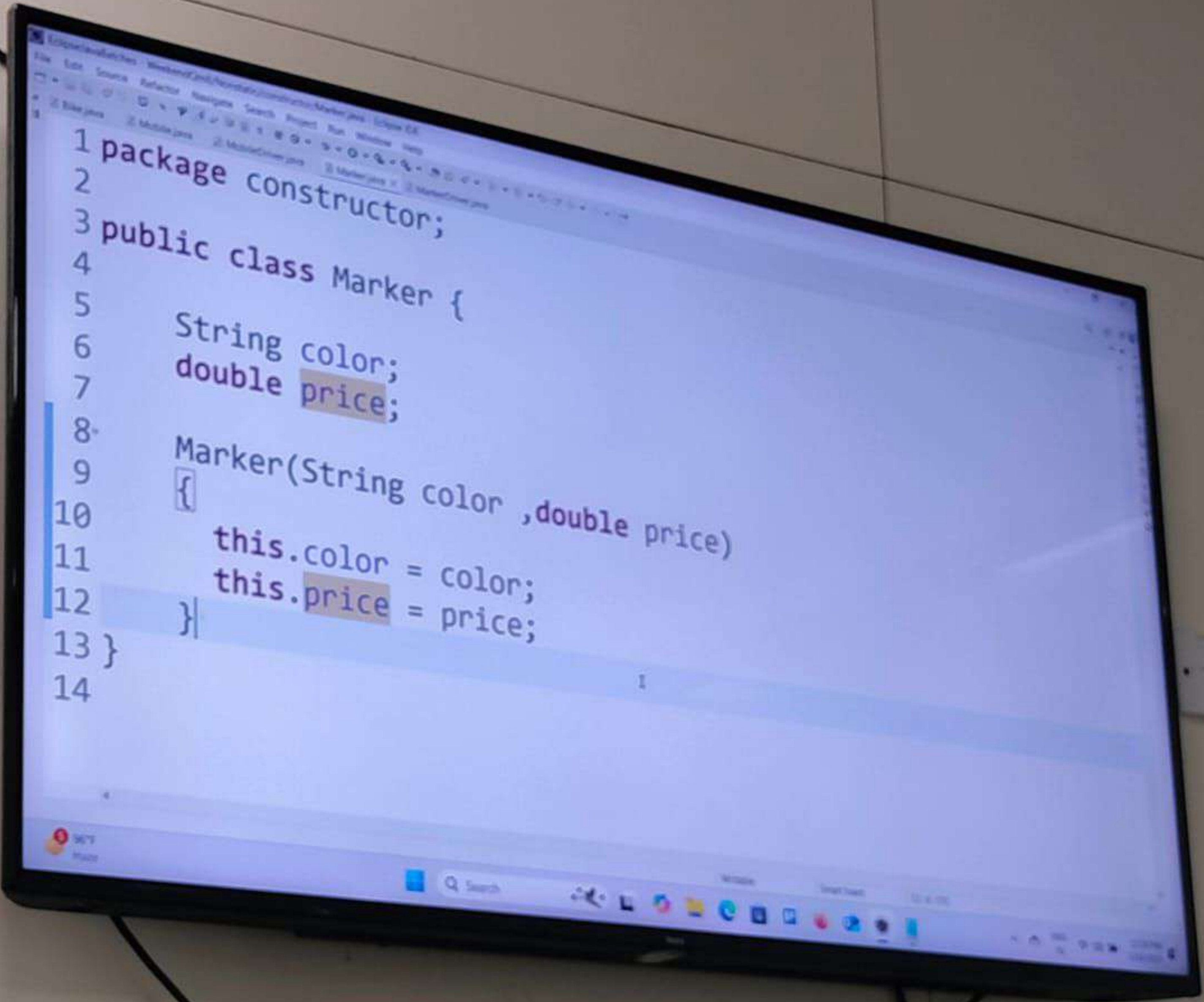
```
package constructor;

public class Bike {
	String bname,model;
	/*
	 * Bike(){}-->To assign the default values
	 * */

	public static void main(String[] args) {
		Bike b1=new Bike();
		b1.bname="Yamaha";
		b1.model="RX135";
		System.out.println("Bike Name : "+b1.bname);
		System.out.println("Bike Model: "+b1.model);
```



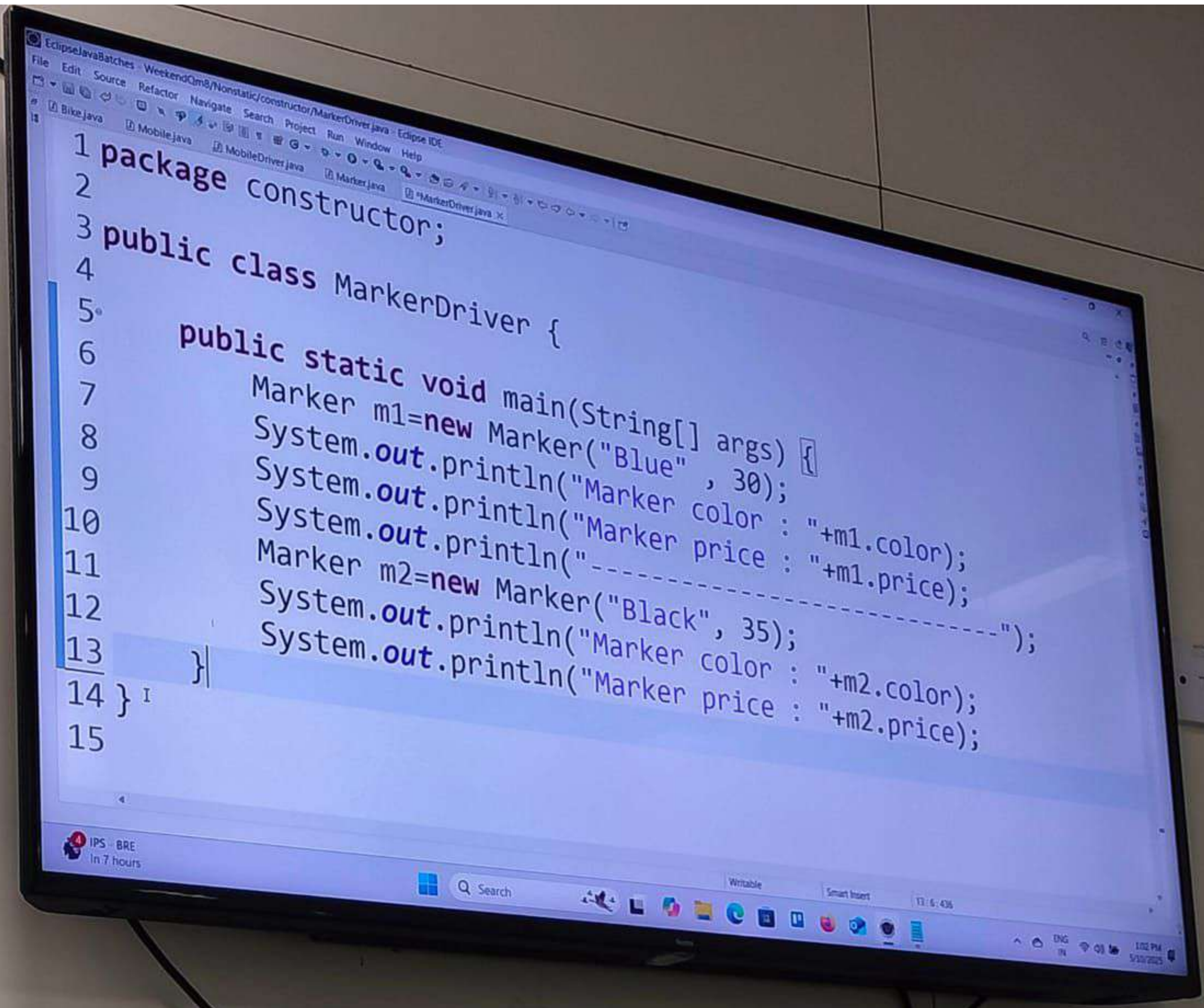
```
public class Mobile {
	String bname;
	String model;
	double price;

	Mobile() {
		System.out.println("Welcome to GiriKaalan Mobiles");
	}
	Mobile(String a,String b,double c)
	{
		bname=a;
		model=b;
		price=c;
	}
}
```



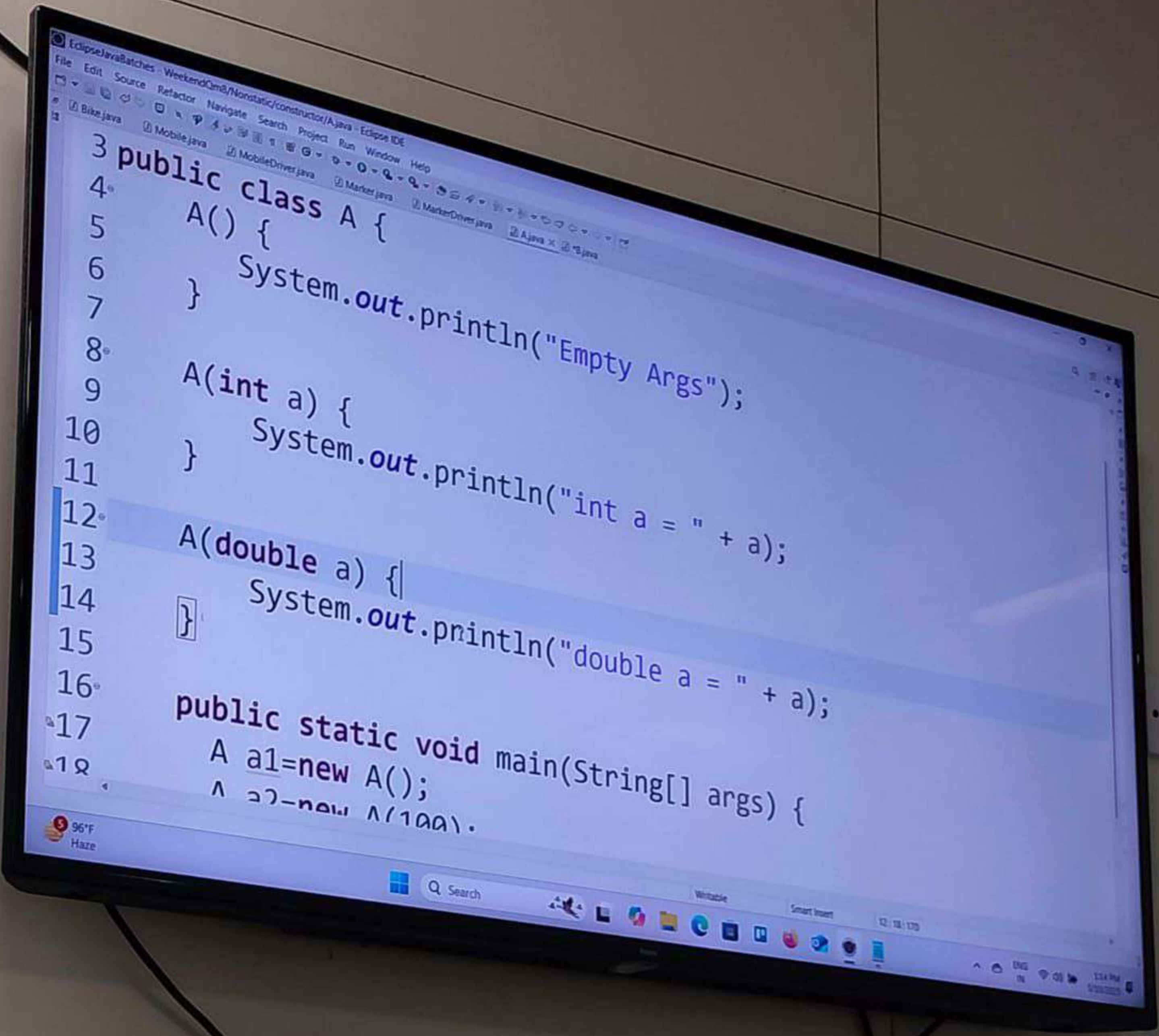
```java
package constructor;

public class MobileDriver {

    public static void main(String[] args) {
        Mobile m1=new Mobile();
        Mobile m2=new Mobile("Oppo","M17 Pro",30000);
        System.out.println("Mobile Brand : "+m2.bname);
        System.out.println("Mobile Model : "+m2.model);
        System.out.println("Mobile Price : "+m2.price);
    }
}
```



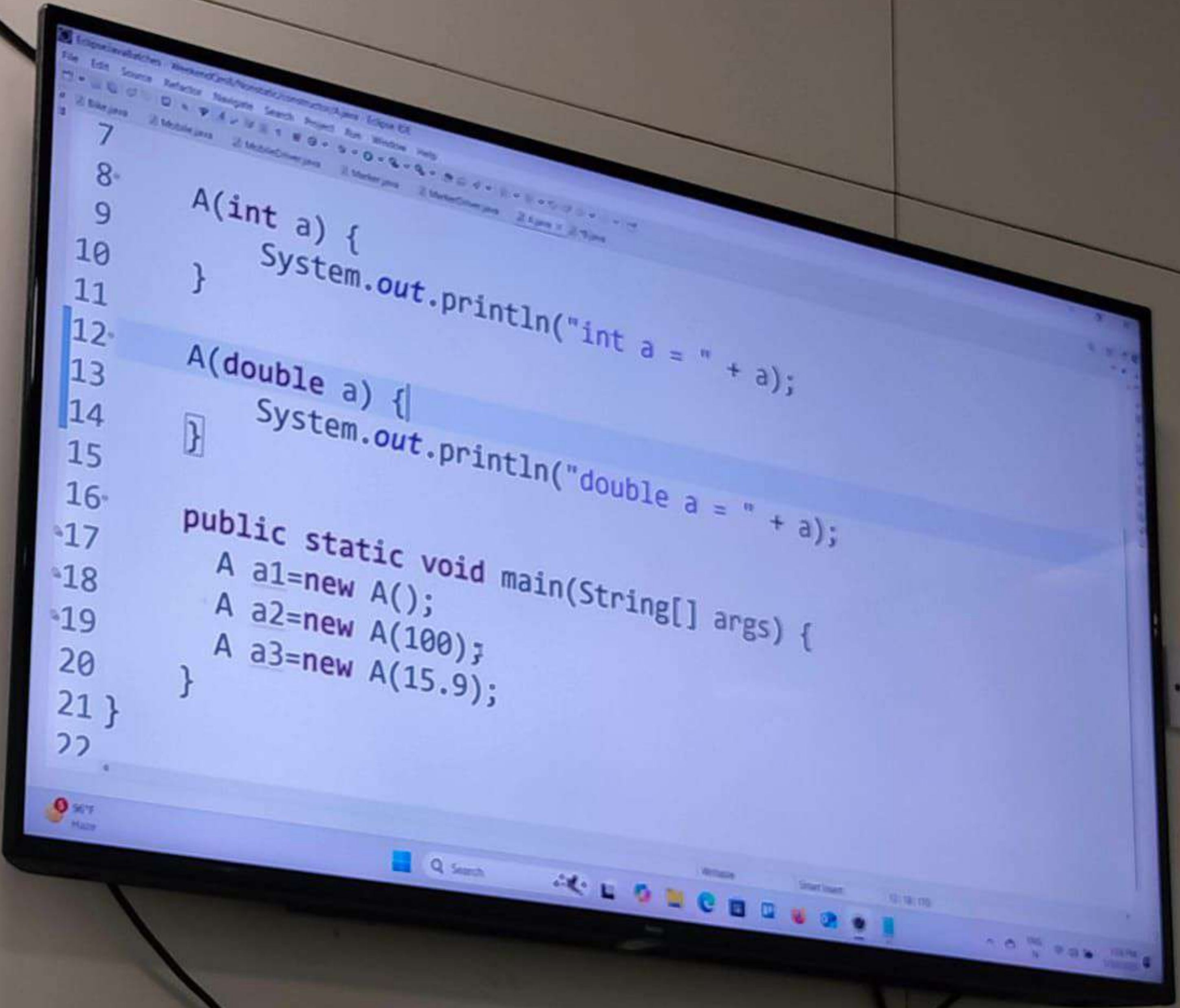
```java
package constructor;
public class Marker {

	String color;
	double price;

	Marker(String color ,double price)
	{
		this.color = color;
		this.price = price;
	}
}

```



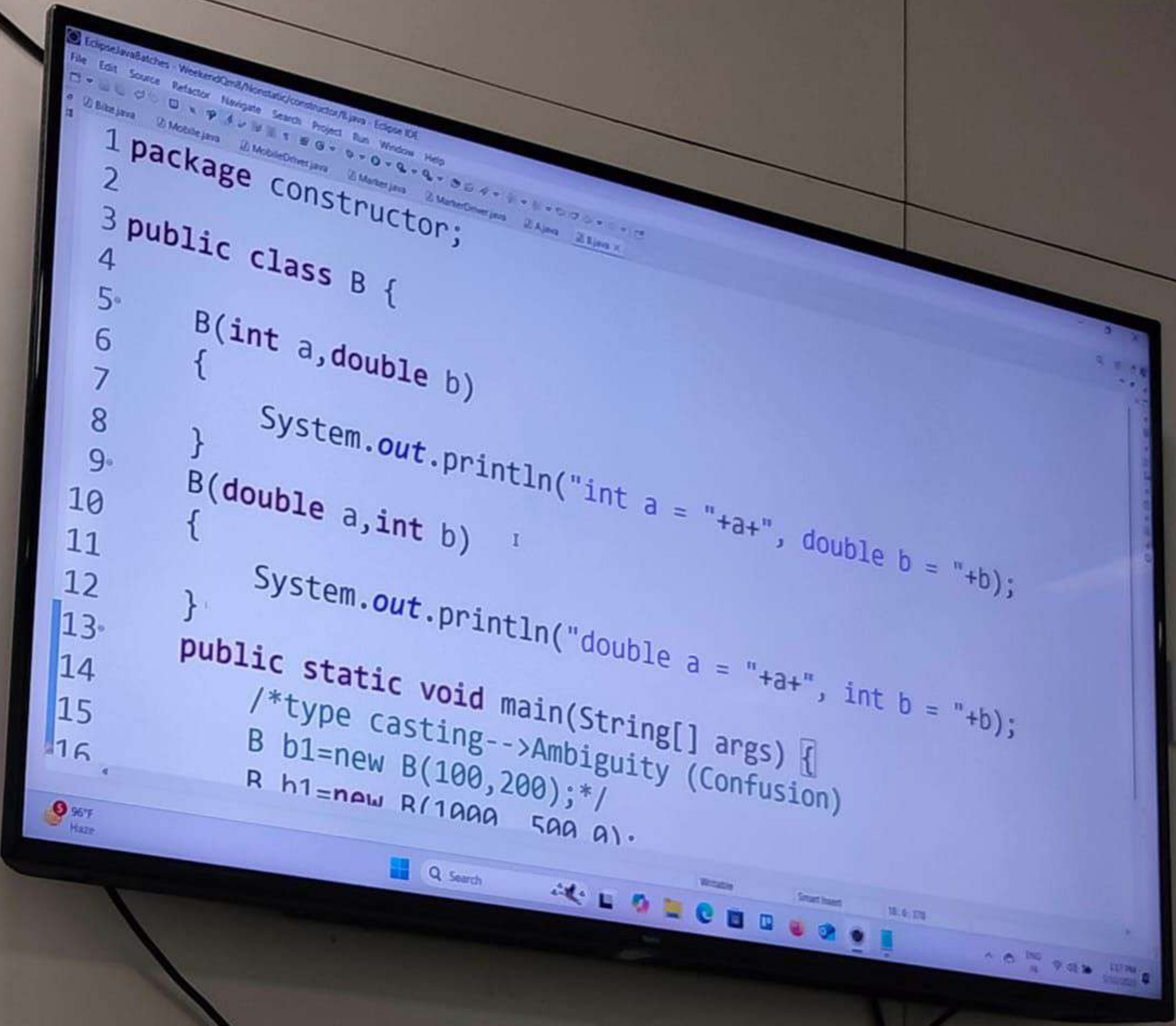
```java
package constructor;

public class MarkerDriver {

    public static void main(String[] args) {
        Marker m1=new Marker("Blue" , 30);
        System.out.println("Marker color : "+m1.color);
        System.out.println("Marker price : "+m1.price);
        System.out.println("---------------------------");
        Marker m2=new Marker("Black", 35);
        System.out.println("Marker color : "+m2.color);
        System.out.println("Marker price : "+m2.price);
    }
}
```



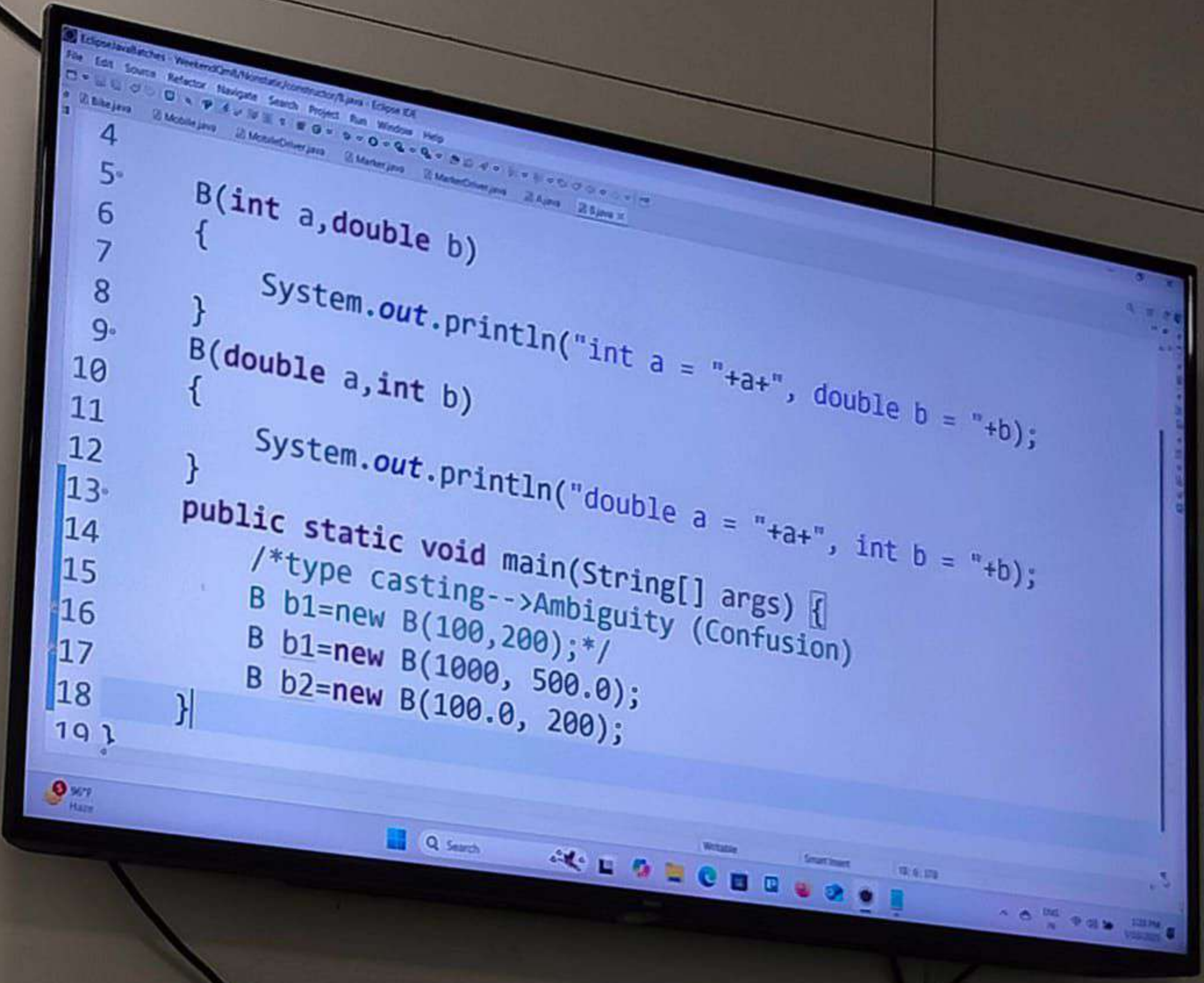
```java
public class A {
    A() {
        System.out.println("Empty Args");
    }

    A(int a) {
        System.out.println("int a = " + a);
    }

    A(double a) {
        System.out.println("double a = " + a);
    }

    public static void main(String[] args) {
        A a1=new A();
```



```
    A(int a) {
        System.out.println("int a = " + a);
    }

    A(double a) {
        System.out.println("double a = " + a);
    }

    public static void main(String[] args) {
      A a1=new A();
      A a2=new A(100);
      A a3=new A(15.9);
    }
}
```



```java
package constructor;

public class B {

    B(int a,double b)
    {
        System.out.println("int a = "+a+", double b = "+b);
    }
    B(double a,int b)
    {
        System.out.println("double a = "+a+", int b = "+b);
    }
    public static void main(String[] args) {
        /*type casting-->Ambiguity (Confusion)
        B b1=new B(100,200);*/
        B b1=new B(1000, 500.0);
```



```
	B(int a,double b)
	{
		System.out.println("int a = "+a+", double b = "+b);
	}
	B(double a,int b)
	{
		System.out.println("double a = "+a+", int b = "+b);
	}
	public static void main(String[] args) {
		/*type casting-->Ambiguity (Confusion)
		B b1=new B(100,200);*/
		B b1=new B(1000, 500.0);
		B b2=new B(100.0, 200);
	}
}
```

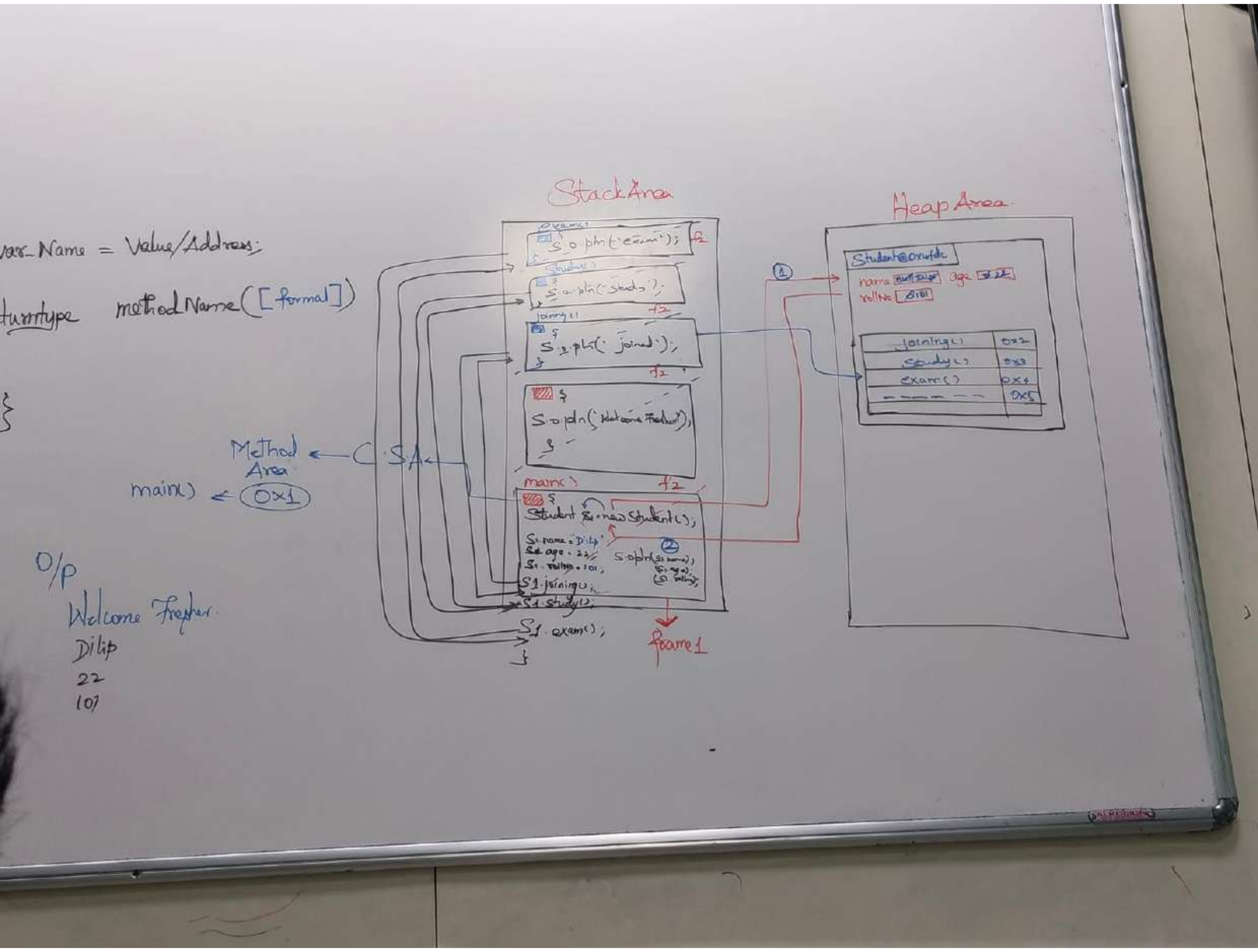
non static member

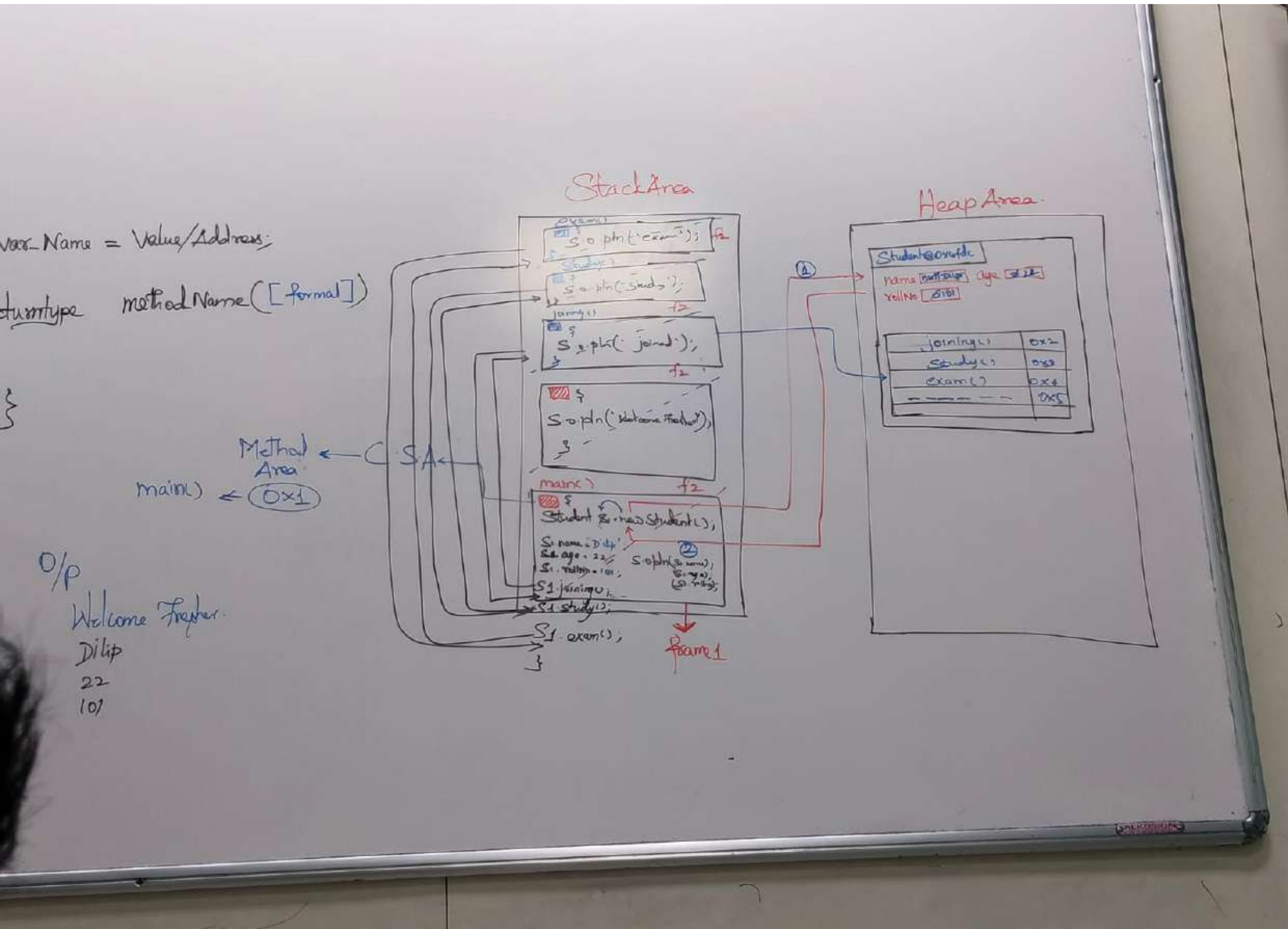
* non static variable → datatype var_Name = Value/Address;
* non static method → [AM] returntype methodName([formal])
  { }
* non static Initializer → { }

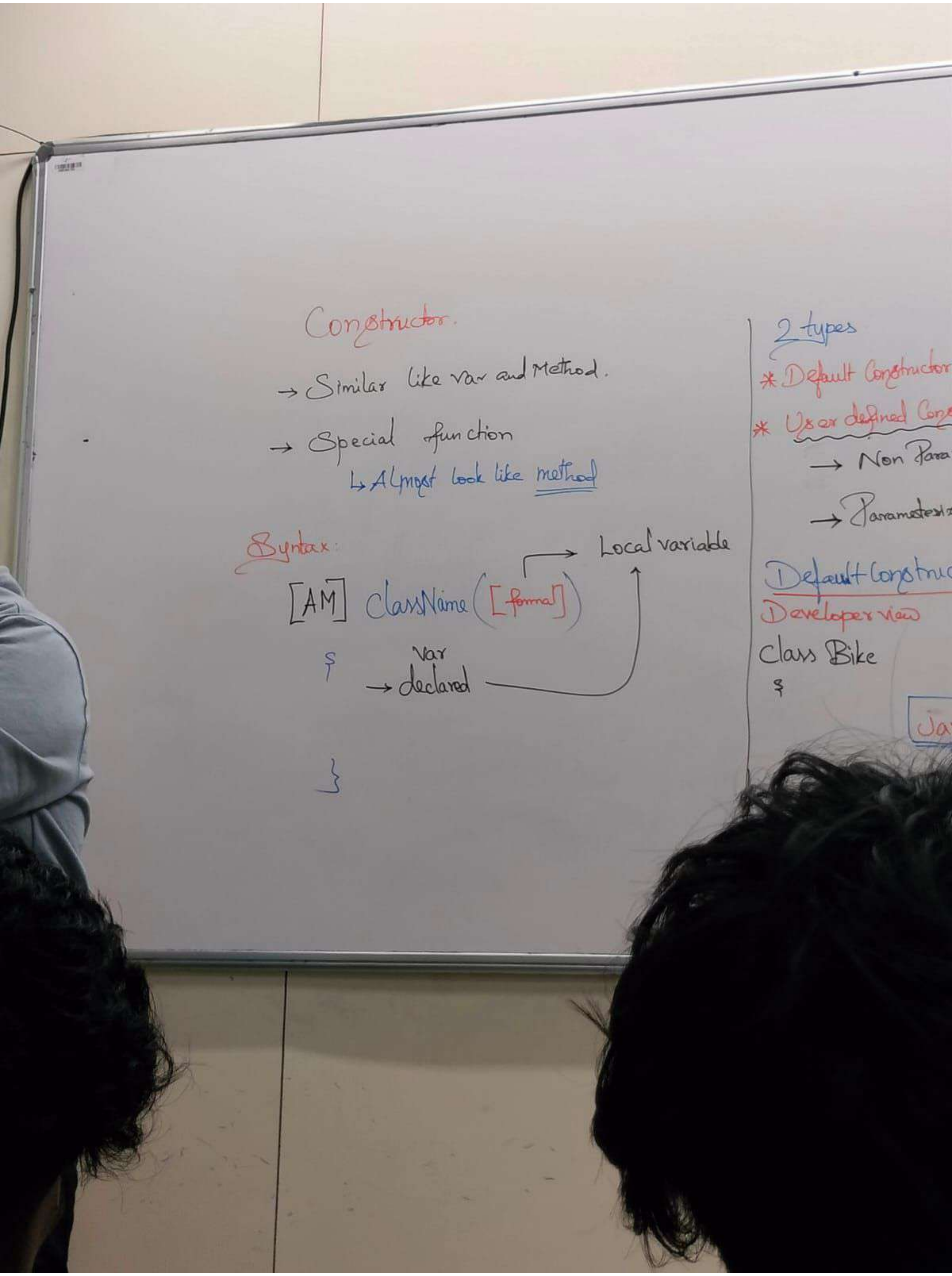
* Constructor

→ Object Members

→ Multiple Copies


var_Name = Value/Address;
turntype methodName([formal])
}
Method Area
C.S.A
main() ← 0x1
O/p
Welcome Fresher
Dilip
22
101
Stack Area
exam()
S.o.pln("exam");
f2
study()
S.o.pln("study");
f2
joining()
S.o.pln("joined");
f2
S.o.pln("Welcome Fresher");
main()
f2
Student S1 = new Student();
S1.name = "Dilip"
S1.age = 22;
S1.rollNo = 101;
S.o.pln(S1.name), (S1.age), (S1.rollNo);
S1.joining();
S1.study();
S1.exam();
frame 1
Heap Area
Student@0x4fdc
name Dilip
age 22
rollNo 101
joining() 0x2
study() 0x3
exam() 0x4
0x5
1
2


var_Name = Value/Address;
turntype methodName([formal])
}
Method Area
main() ← 0x1
C.S.A
O/p
Welcome Fresher
Dilip
22
101
StackArea
exam()
{
S.o.pln("exam");
}
f2
Study()
{
S.o.pln("study");
}
f2
joining()
{
S.o.pln("joined");
}
f2
{
S.o.pln("Welcome Fresher");
}
main()
f2
{
Student s1 = new Student();
S1.name = "Dilip";
S1.age = 22;
S1.rollno = 101;
S.o.pln(s1.name);
(s1.age);
(s1.rollno);
S1.joining();
S1.study();
S1.exam();
}
Frame 1
1
2
HeapArea
Student@0x4fdc
name null Dilip
age 0 22
rollNo 0 101
joining() 0x2
Study() 0x3
exam() 0x4
0x5



Constructor.

→ Similar like var and Method.

→ Special function
  ↳ Almost look like <u>method</u>

Syntax:

```
[AM] ClassName([formal])  → Local variable
{
   var
   → declared
}
```

<u>2 types</u>

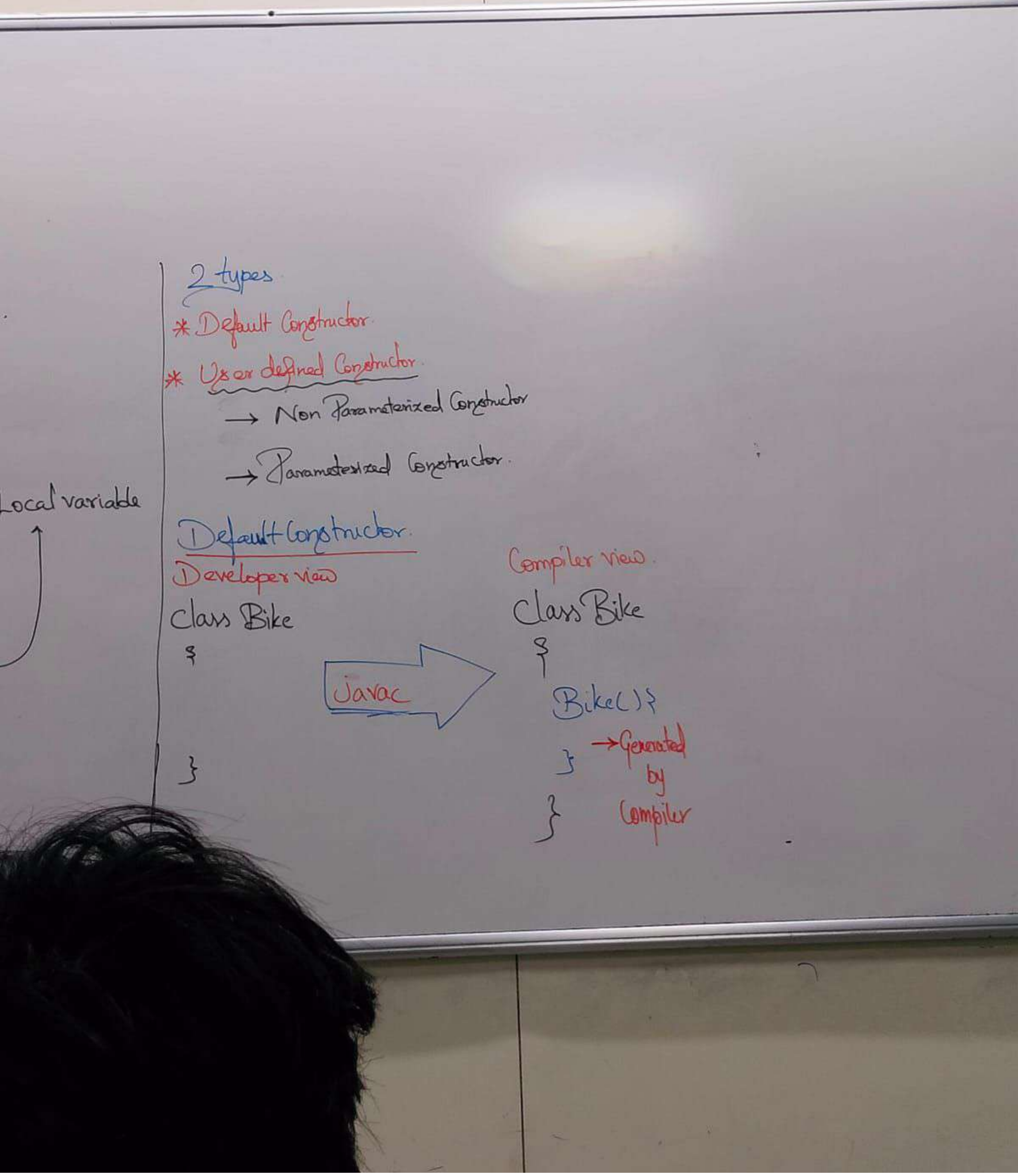
* Default Constructor
* User defined Con
  → Non Para
  → Parameteri

Default Construc

Developer view

```
Class Bike
{
```

Ja



Local variable

2 types

* Default Constructor
* User defined Constructor
  - → Non Parameterized Constructor
  - → Parameterized Constructor

Default Constructor

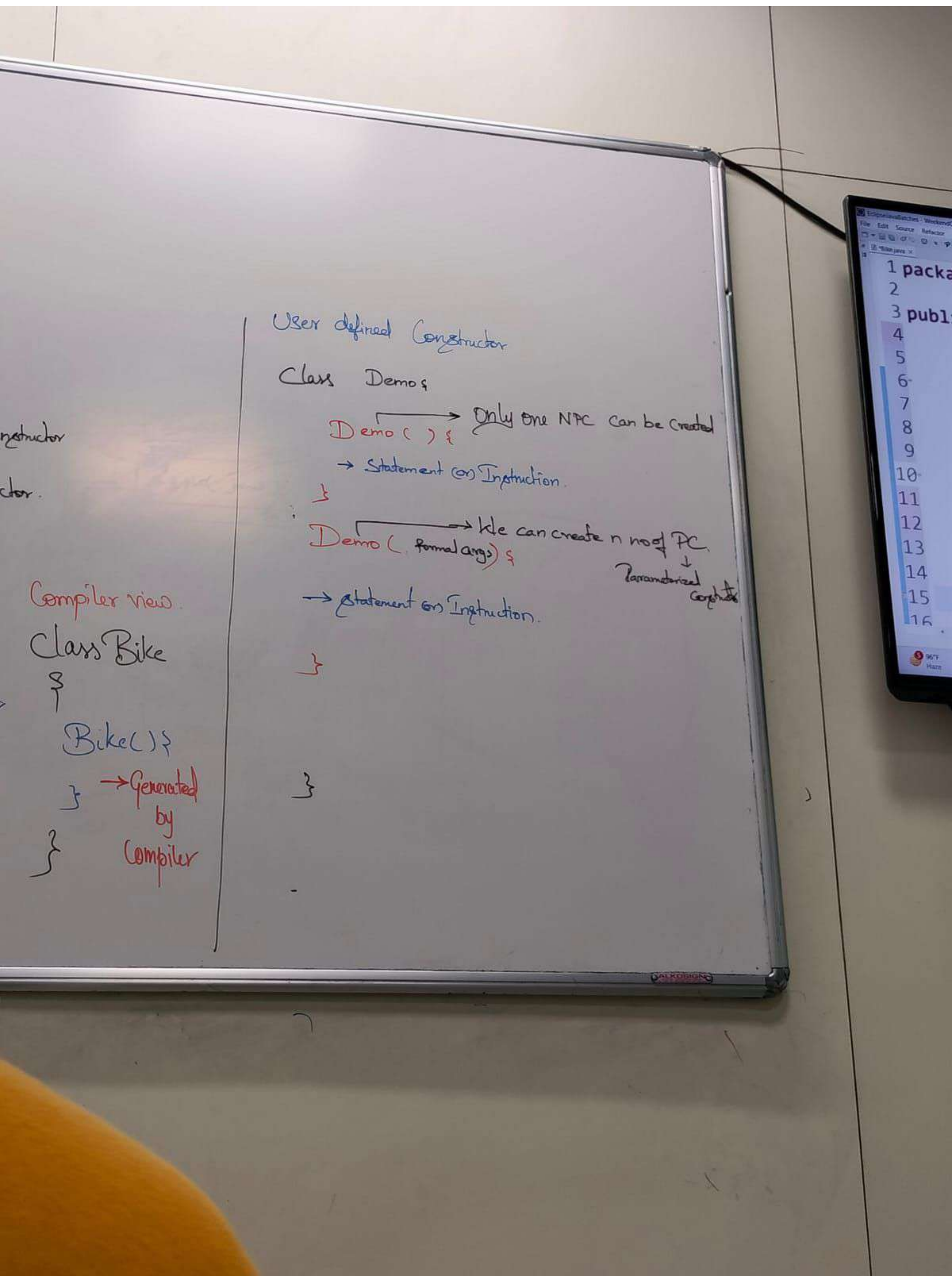
Developer view

```
Class Bike
{


}
```

Javac

Compiler view

```
Class Bike
{
  Bike(){
  }  → Generated
       by
       Compiler
}
```



nstructor

ctor.

Compiler view

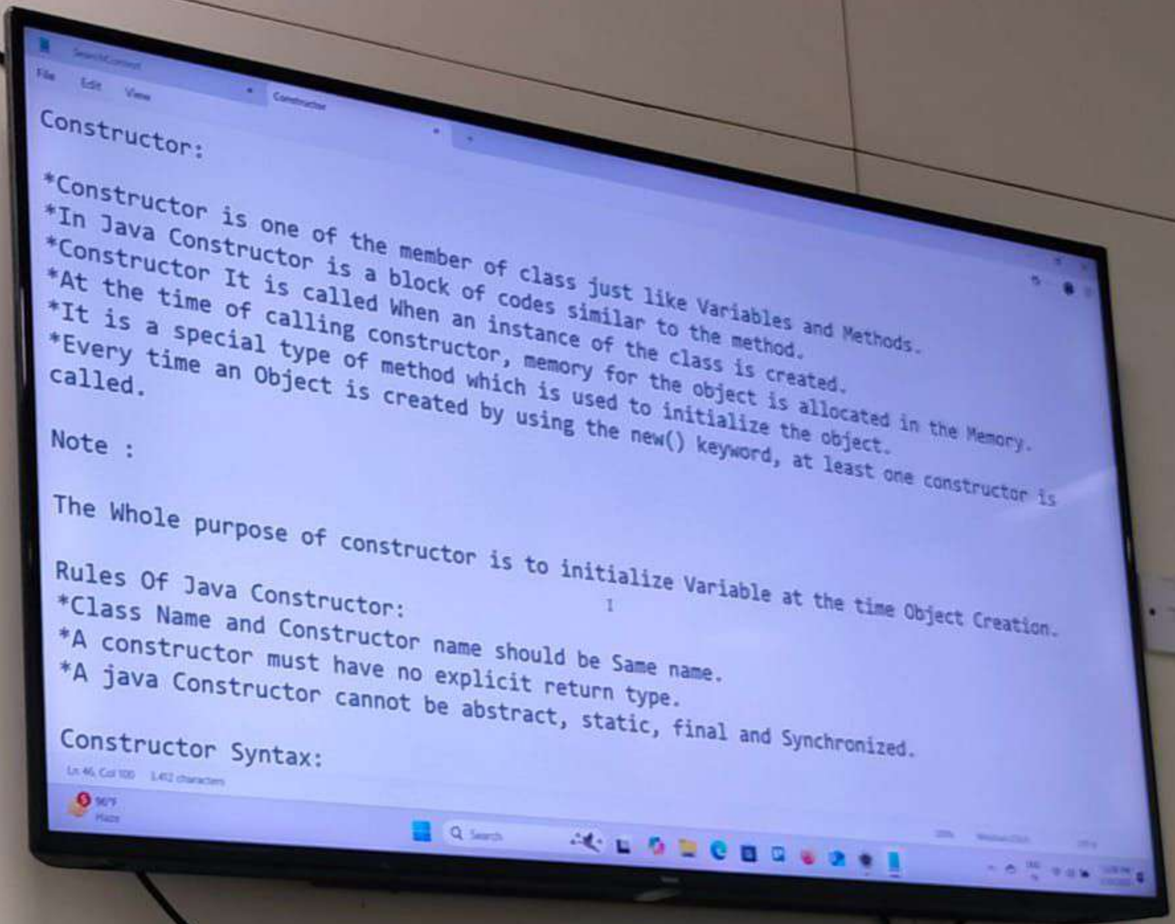
```
Class Bike
{
    Bike(){
    }  → Generated
         by
         Compiler
}
```

User defined Constructor

```
Class Demo{

    Demo ( ) {  → Only one NPC can be Created
    → Statement (or) Instruction.
    }

    Demo ( formal args) {  → We can create n no of PC.
                                                  ↓
                                        Parameterized
                                          Constructor
    → Statement (or) Instruction.

    }

}
```

packa

publ


File
Edit
View
Constructor

Constructor:

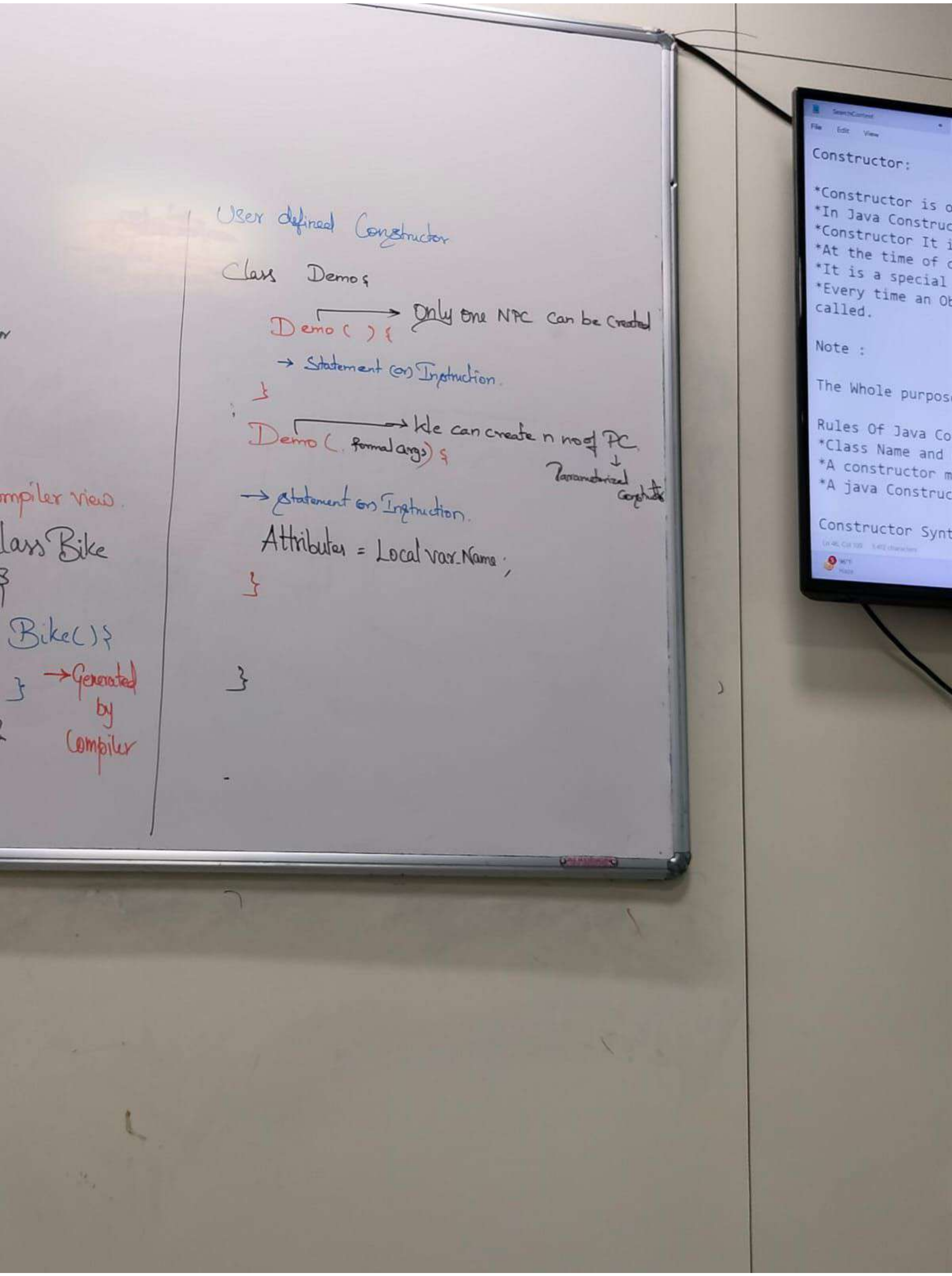
*Constructor is one of the member of class just like Variables and Methods.
*In Java Constructor is a block of codes similar to the method.
*Constructor It is called When an instance of the class is created.
*At the time of calling constructor, memory for the object is allocated in the Memory.
*It is a special type of method which is used to initialize the object.
*Every time an Object is created by using the new() keyword, at least one constructor is called.

Note :

The Whole purpose of constructor is to initialize Variable at the time Object Creation.

Rules Of Java Constructor:
*Class Name and Constructor name should be Same name.
*A constructor must have no explicit return type.
*A java Constructor cannot be abstract, static, final and Synchronized.

Constructor Syntax:



r

mpiler view

lass Bike

{

Bike(){

} →Generated by Compiler

}

User defined Constructor

```
Class Demo{
    Demo( ){ → Only one NPC can be created
    → Statement (or) Instruction
    }
    Demo( formal args){ → We can create n no of PC
                                  ↓
                          Parameterized Constructor
    → Statement or Instruction
    Attributes = Local var_Name;
    }
}
```

Constructor:

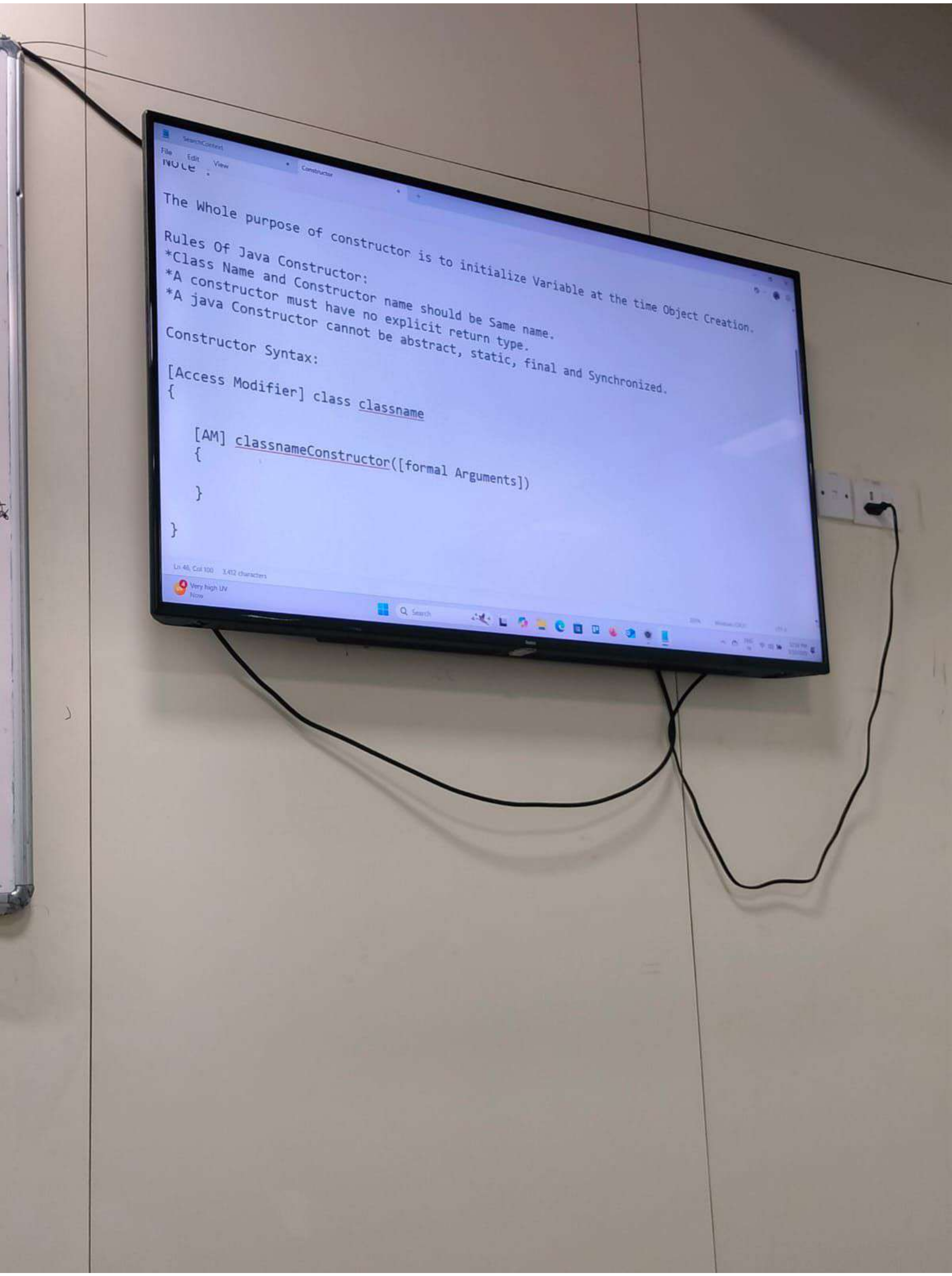
*Constructor is o
*In Java Construc
*Constructor It i
*At the time of c
*It is a special
*Every time an Ob
called.

Note :

The Whole purpos

Rules Of Java Co
*Class Name and
*A constructor m
*A java Construc

Constructor Synt



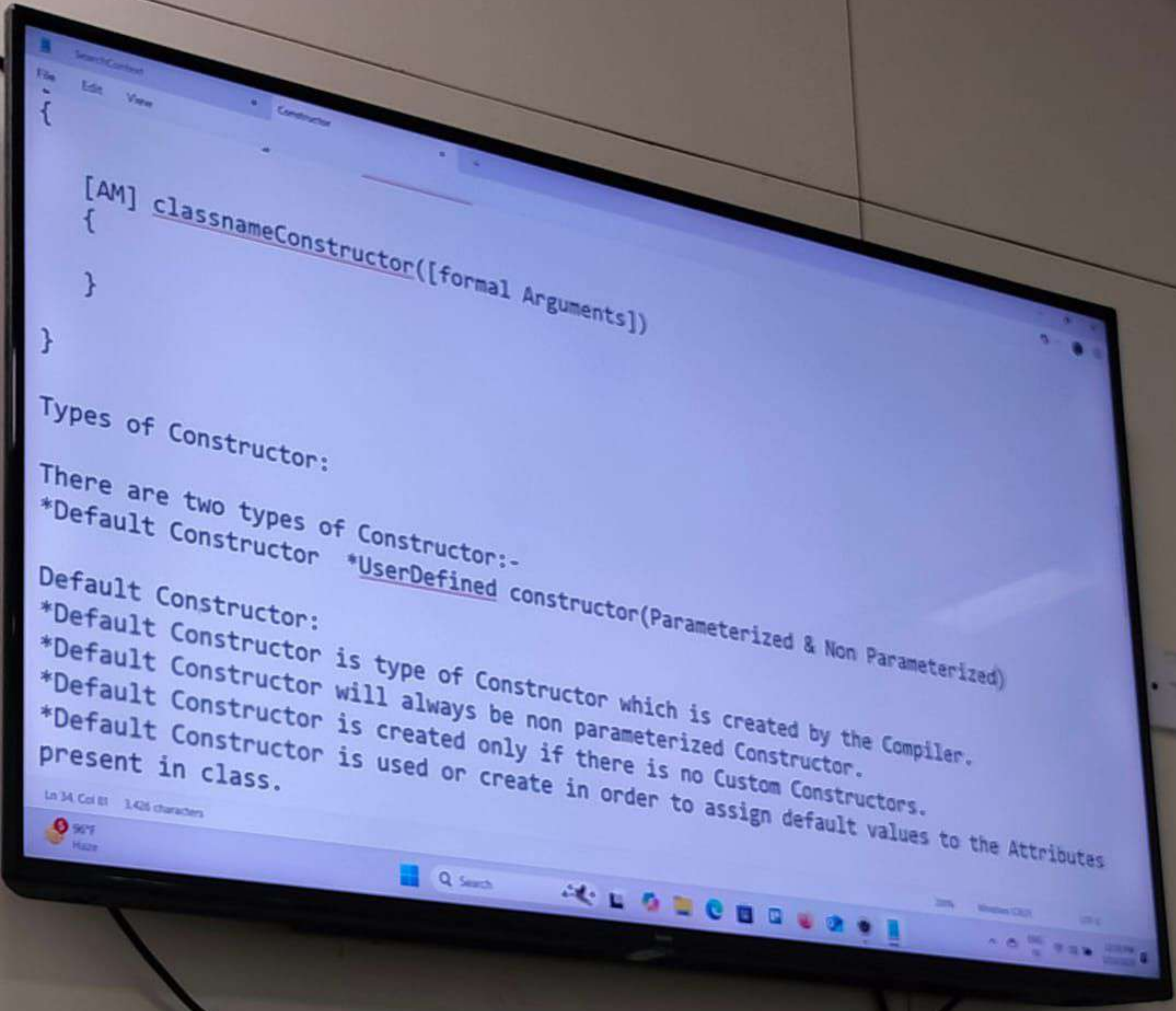
Note :

The Whole purpose of constructor is to initialize Variable at the time Object Creation.

Rules Of Java Constructor:
*Class Name and Constructor name should be Same name.
*A constructor must have no explicit return type.
*A java Constructor cannot be abstract, static, final and Synchronized.

Constructor Syntax:

```
[Access Modifier] class classname
{

    [AM] classnameConstructor([formal Arguments])
    {

    }

}
```

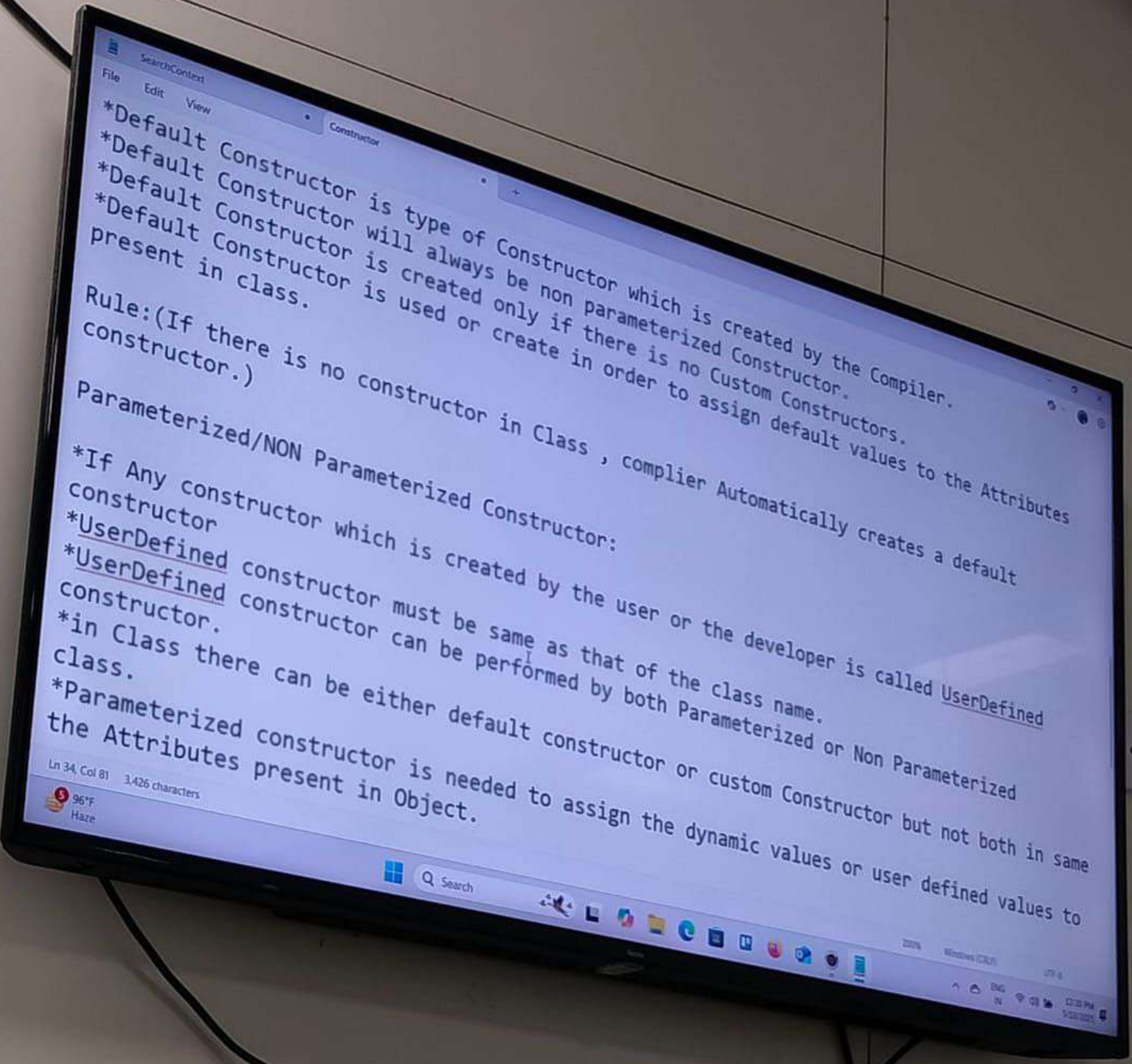


```
{

    [AM] classnameConstructor([formal Arguments])
    {

    }
}
```

Types of Constructor:

There are two types of Constructor:-
*Default Constructor *UserDefined constructor(Parameterized & Non Parameterized)

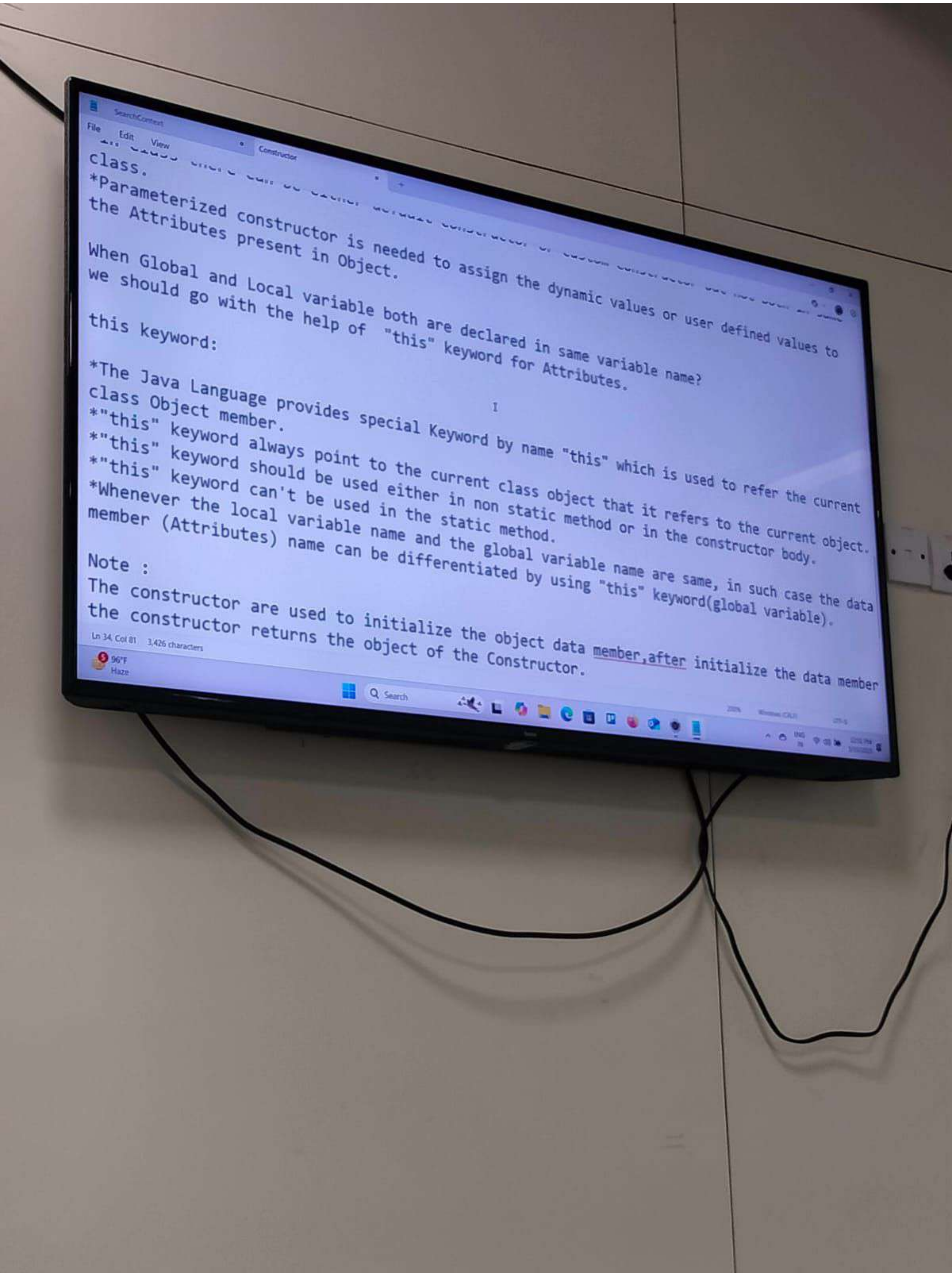
Default Constructor:
*Default Constructor is type of Constructor which is created by the Compiler.
*Default Constructor will always be non parameterized Constructor.
*Default Constructor is created only if there is no Custom Constructors.
*Default Constructor is used or create in order to assign default values to the Attributes present in class.


SearchContext
Constructor
File
Edit
View
Ln 34, Col 81
3,426 characters
96°F
Haze
Search

*Default Constructor is type of Constructor which is created by the Compiler.
*Default Constructor will always be non parameterized Constructor.
*Default Constructor is created only if there is no Custom Constructors.
*Default Constructor is used or create in order to assign default values to the Attributes present in class.
Rule:(If there is no constructor in Class , complier Automatically creates a default constructor.)

Parameterized/NON Parameterized Constructor:

*If Any constructor which is created by the user or the developer is called UserDefined constructor
*UserDefined constructor must be same as that of the class name.
*UserDefined constructor can be performed by both Parameterized or Non Parameterized constructor.
*in Class there can be either default constructor or custom Constructor but not both in same class.
*Parameterized constructor is needed to assign the dynamic values or user defined values to the Attributes present in Object.


SearchContext
File
Edit
View
Constructor
Ln 34, Col 81
3,426 characters
96°F
Haze
Search

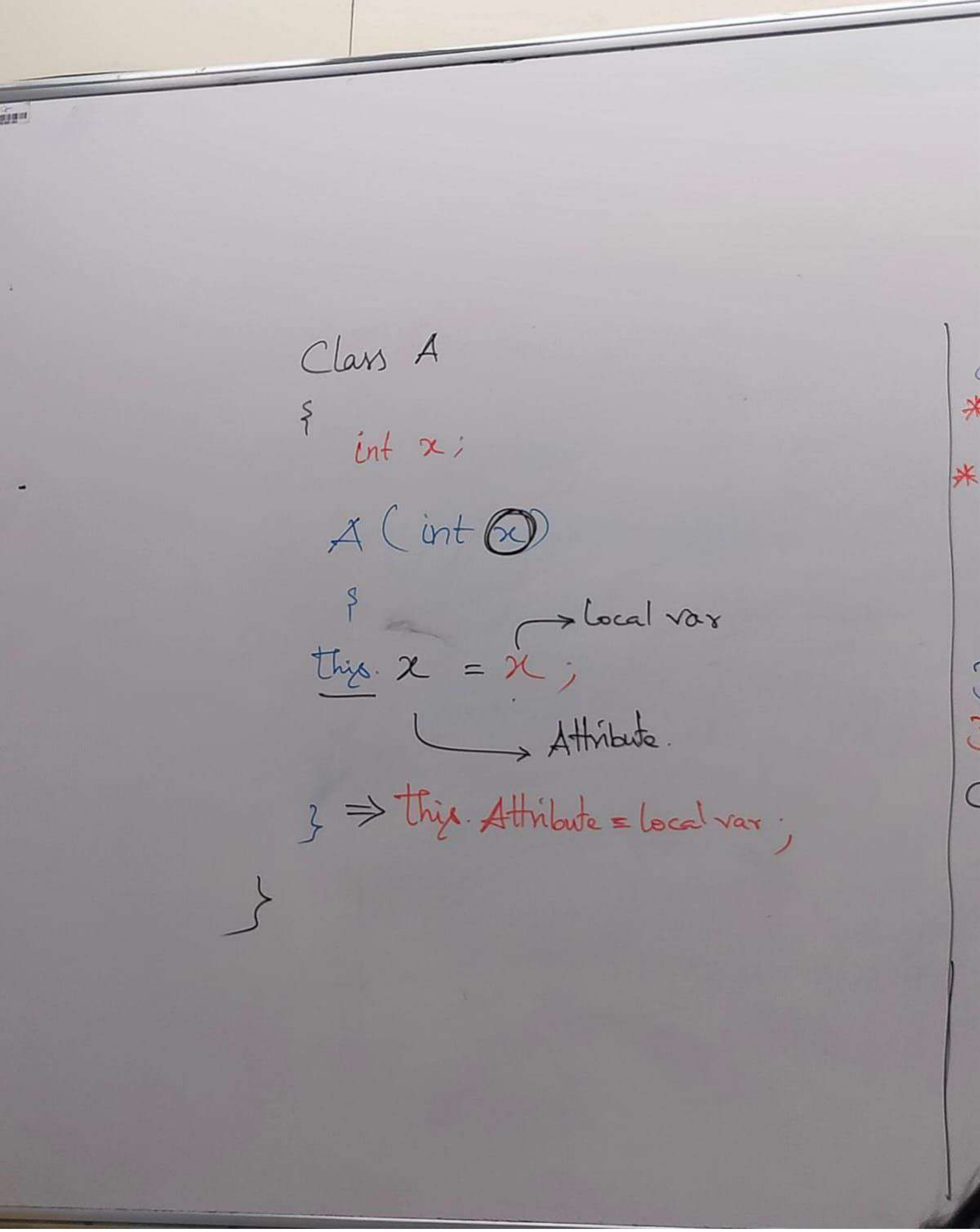
class.
*Parameterized constructor is needed to assign the dynamic values or user defined values to the Attributes present in Object.

When Global and Local variable both are declared in same variable name?
we should go with the help of "this" keyword for Attributes.

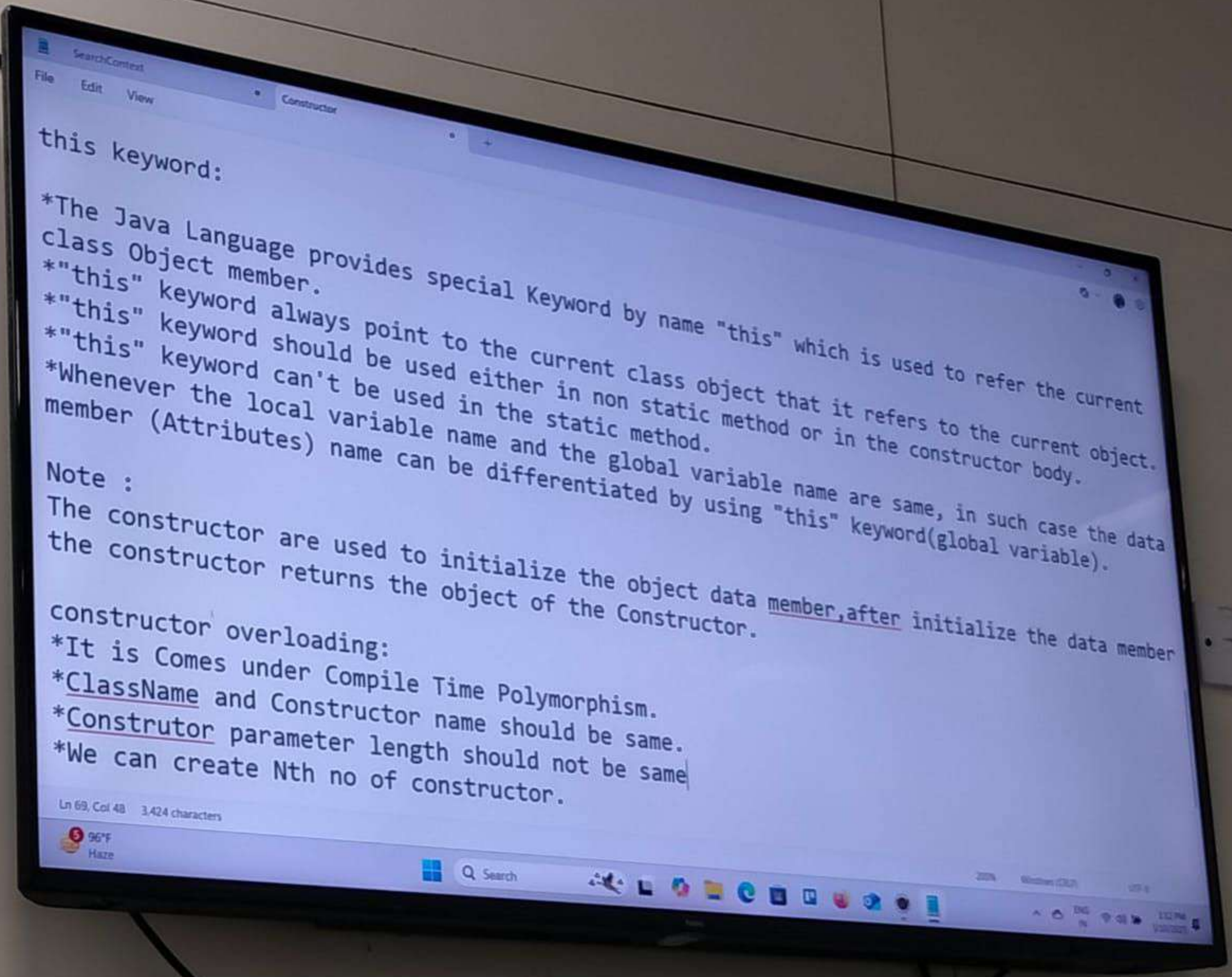
this keyword:

*The Java Language provides special Keyword by name "this" which is used to refer the current class Object member.
*"this" keyword always point to the current class object that it refers to the current object.
*"this" keyword should be used either in non static method or in the constructor body.
*"this" keyword can't be used in the static method.
*Whenever the local variable name and the global variable name are same, in such case the data member (Attributes) name can be differentiated by using "this" keyword(global variable).

Note :
The constructor are used to initialize the object data member,after initialize the data member the constructor returns the object of the Constructor.



```
Class A
{
    int x;

   A(int x)
    {
 this. x  =  x ;   → Local var
         → Attribute
   } ⇒ this. Attribute = local var ;
}
```



this keyword:

*The Java Language provides special Keyword by name "this" which is used to refer the current class Object member.
*"this" keyword always point to the current class object that it refers to the current object.
*"this" keyword should be used either in non static method or in the constructor body.
*"this" keyword can't be used in the static method.
*Whenever the local variable name and the global variable name are same, in such case the data member (Attributes) name can be differentiated by using "this" keyword(global variable).

Note :
The constructor are used to initialize the object data member,after initialize the data member the constructor returns the object of the Constructor.

constructor overloading:
*It is Comes under Compile Time Polymorphism.
*ClassName and Constructor name should be same.
*Construtor parameter length should not be same
*We can create Nth no of constructor.