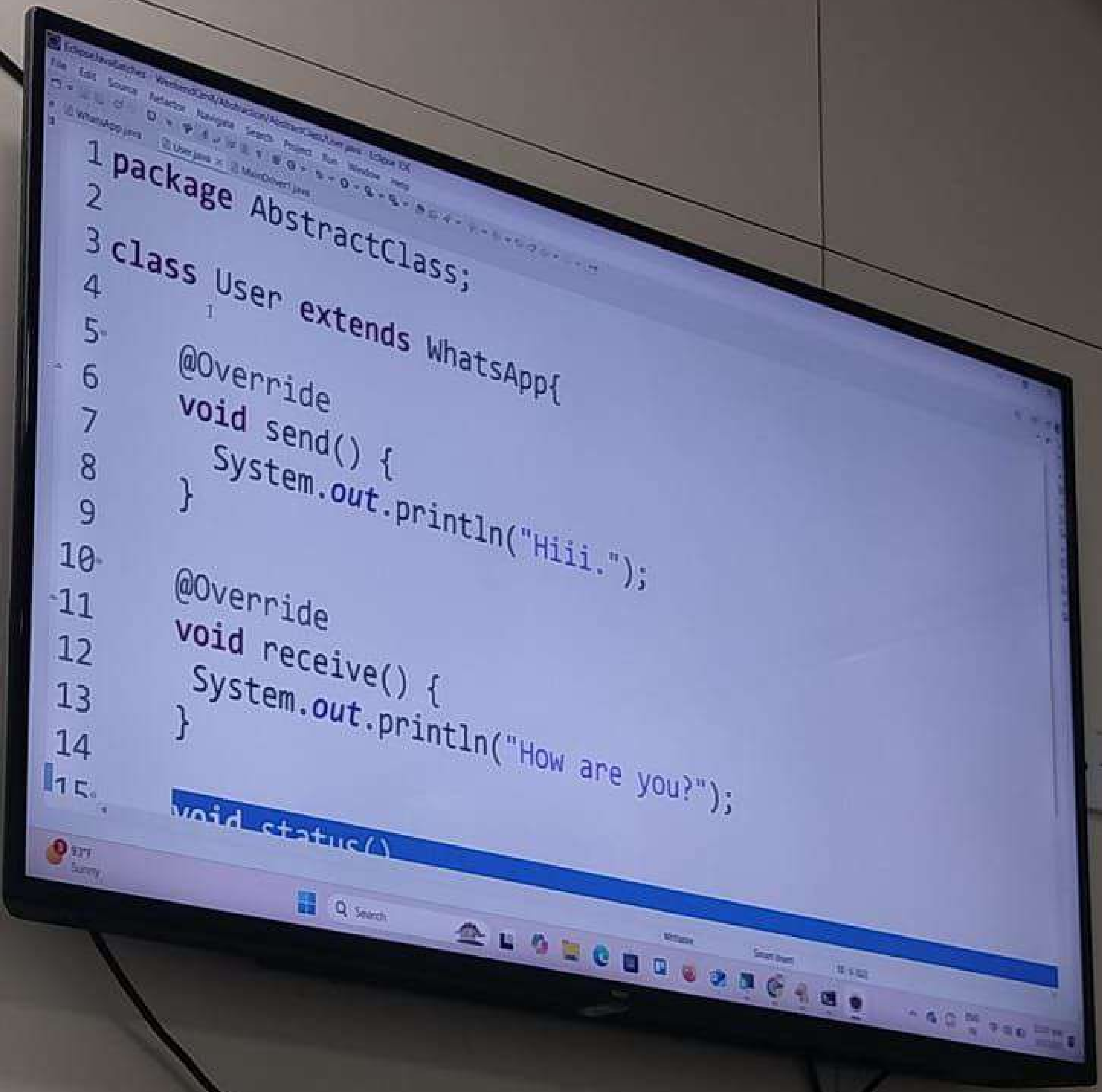
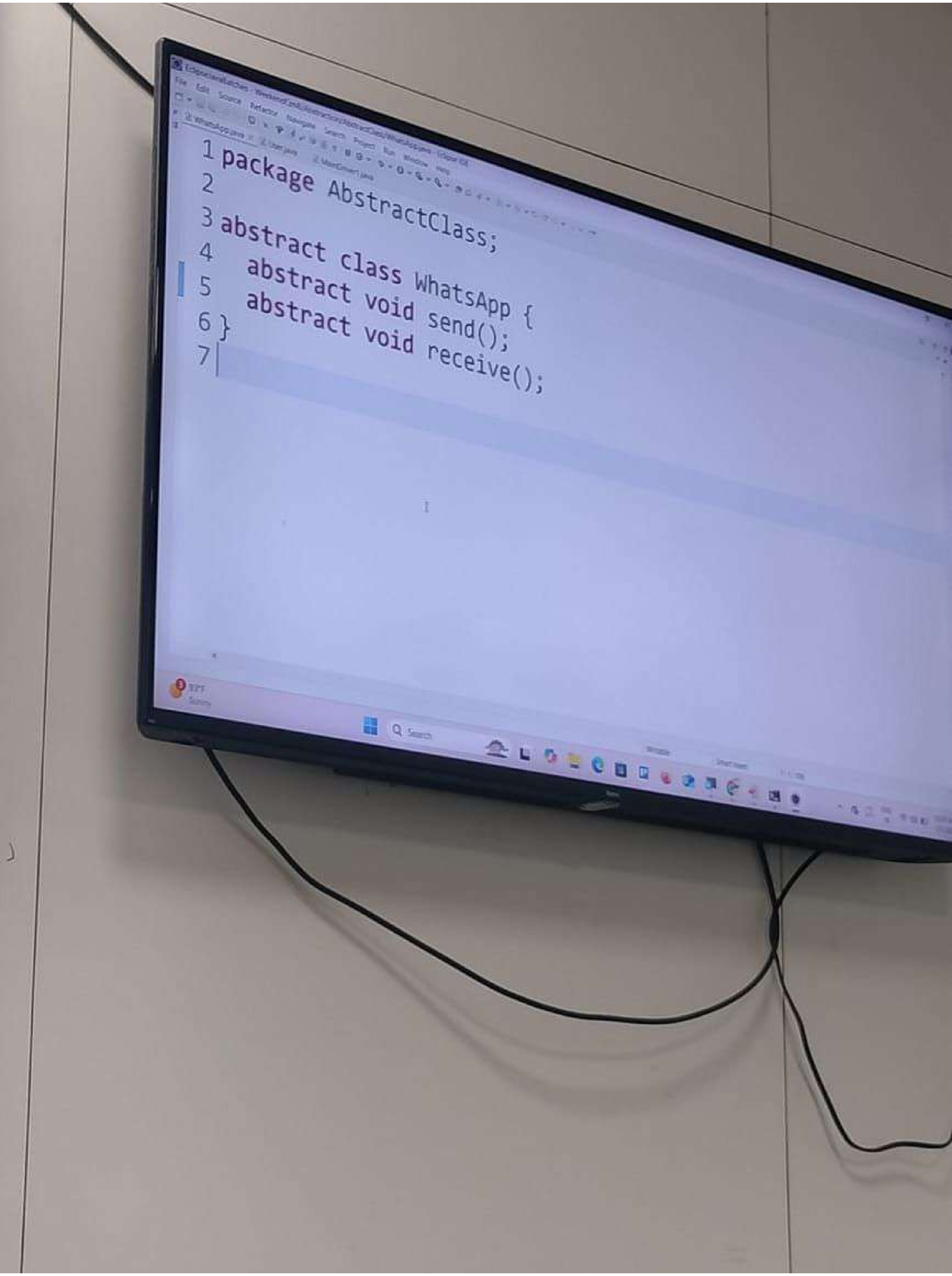
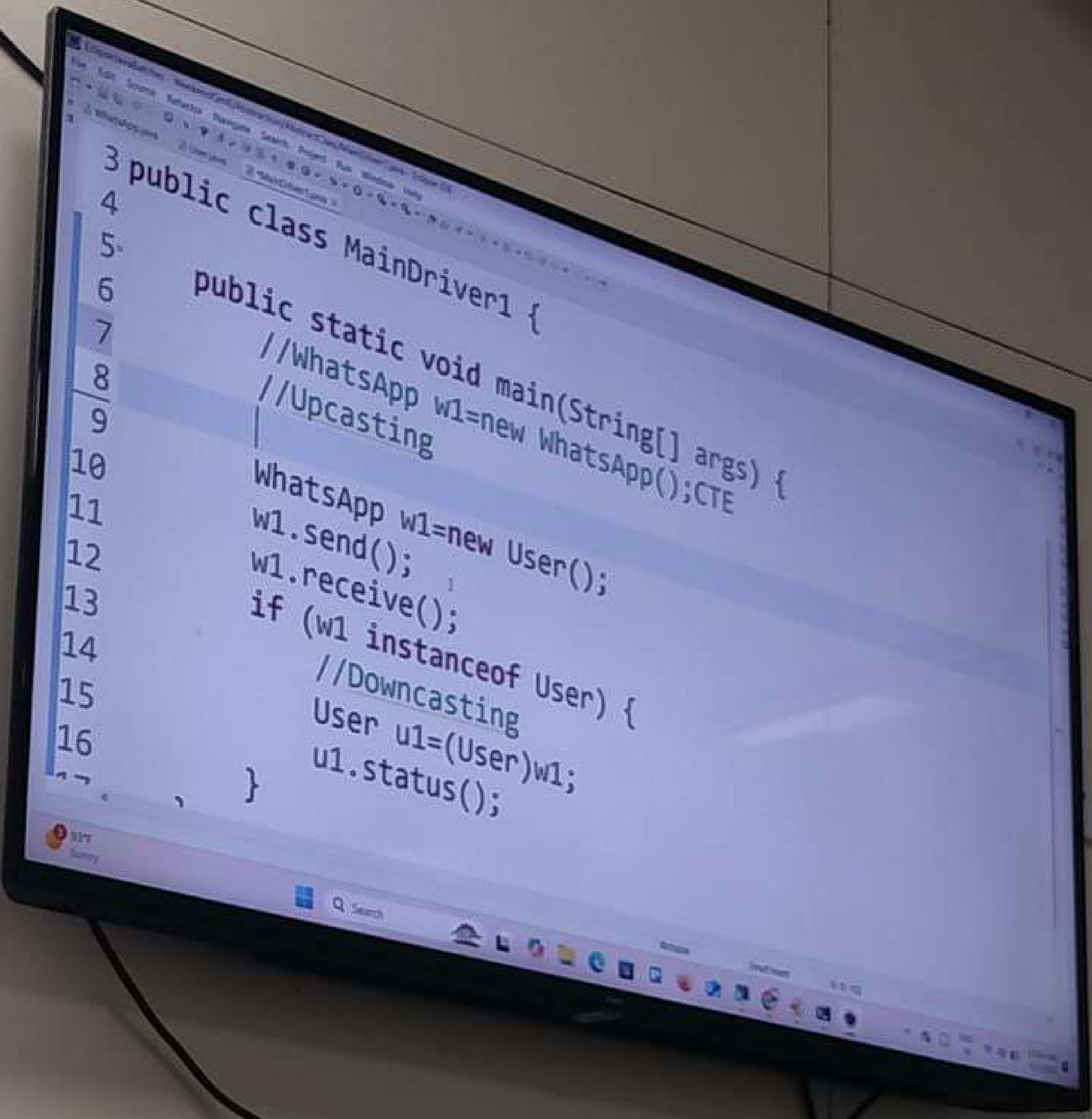


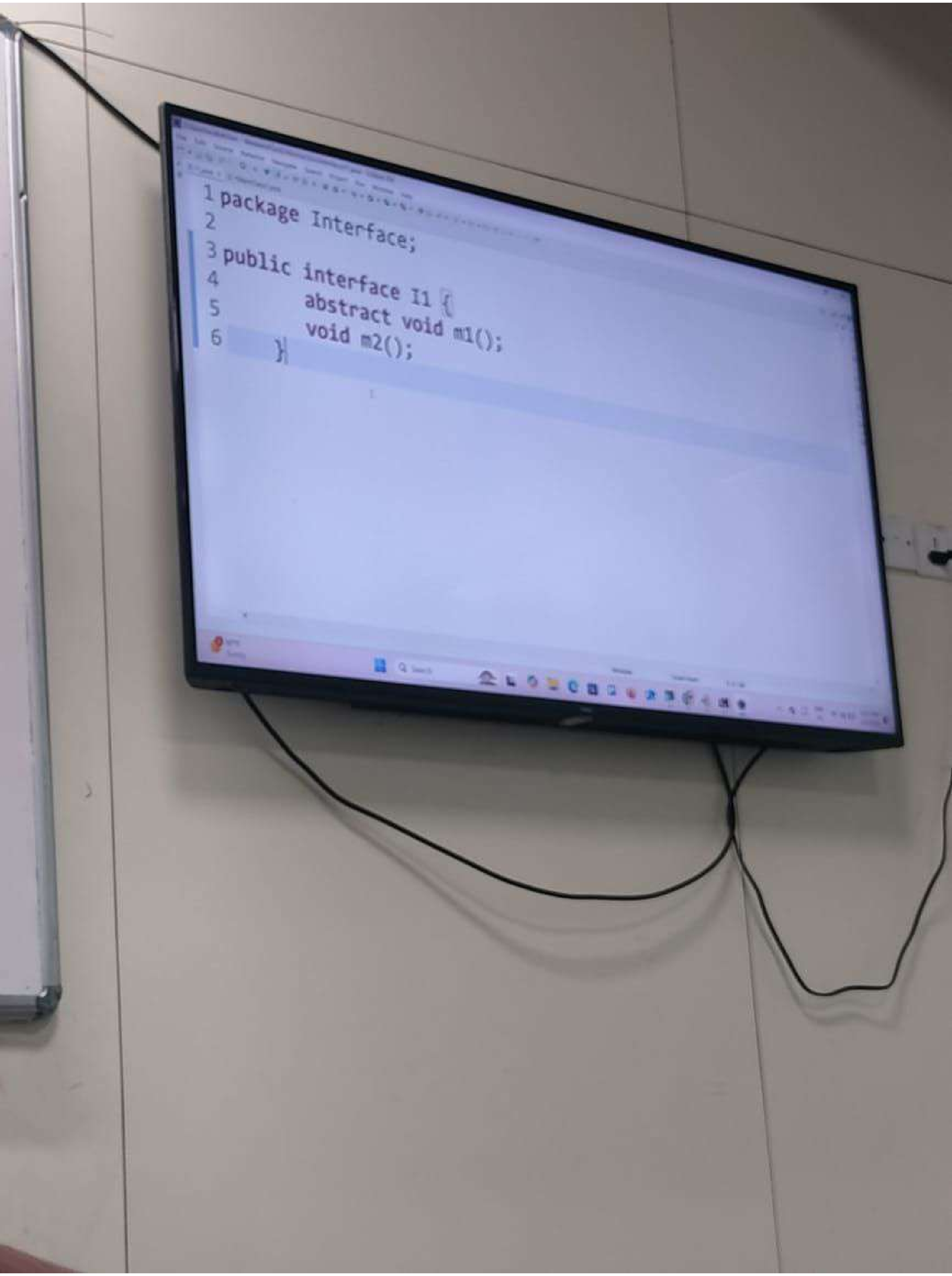
```
package AbstractClass;

class User extends WhatsApp{

    @Override
    void send() {
      System.out.println("Hiii.");
    }

    @Override
    void receive() {
     System.out.println("How are you?");
    }

    void status()
```



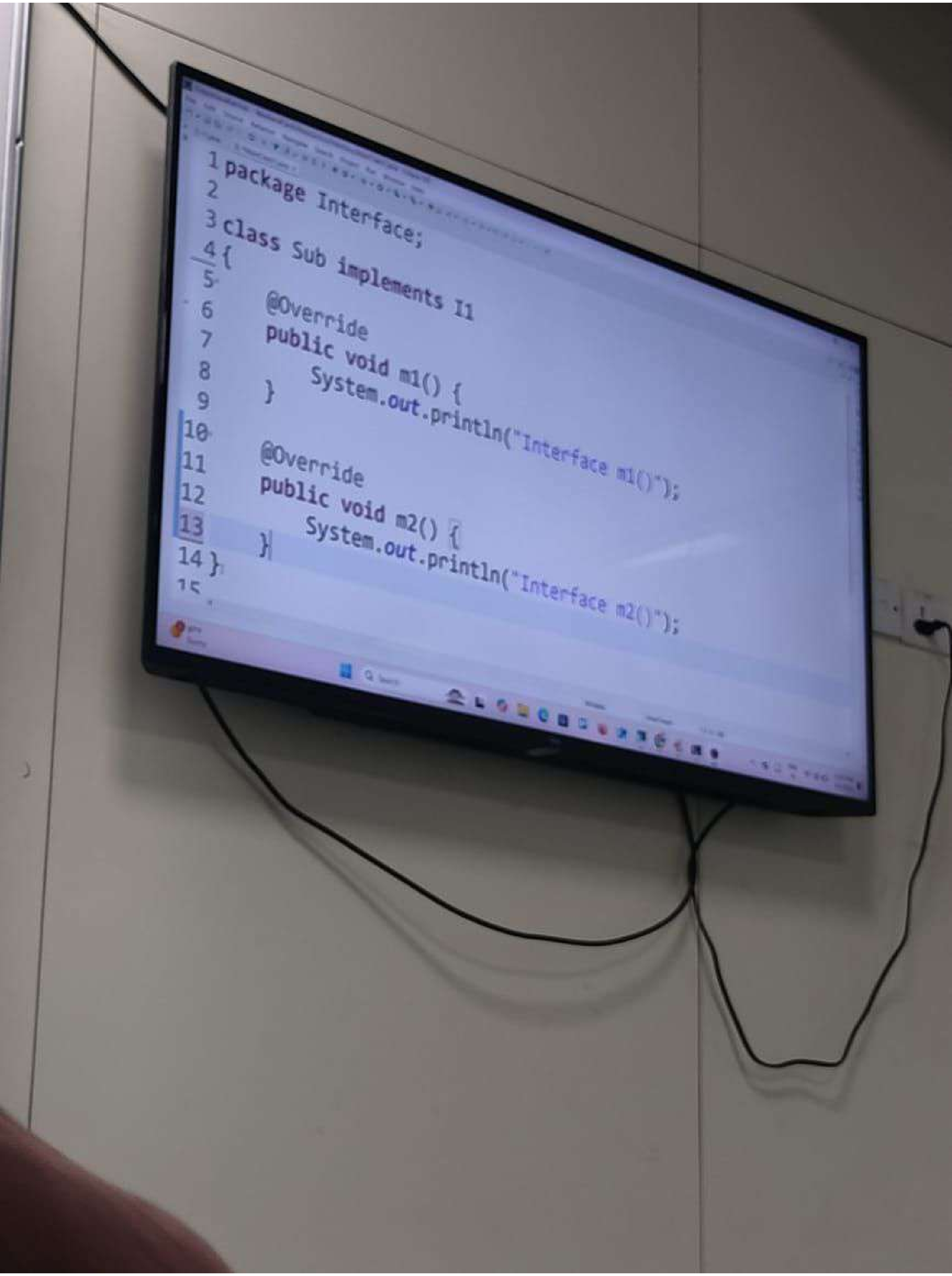
```java
package AbstractClass;

abstract class WhatsApp {
  abstract void send();
  abstract void receive();
}
```



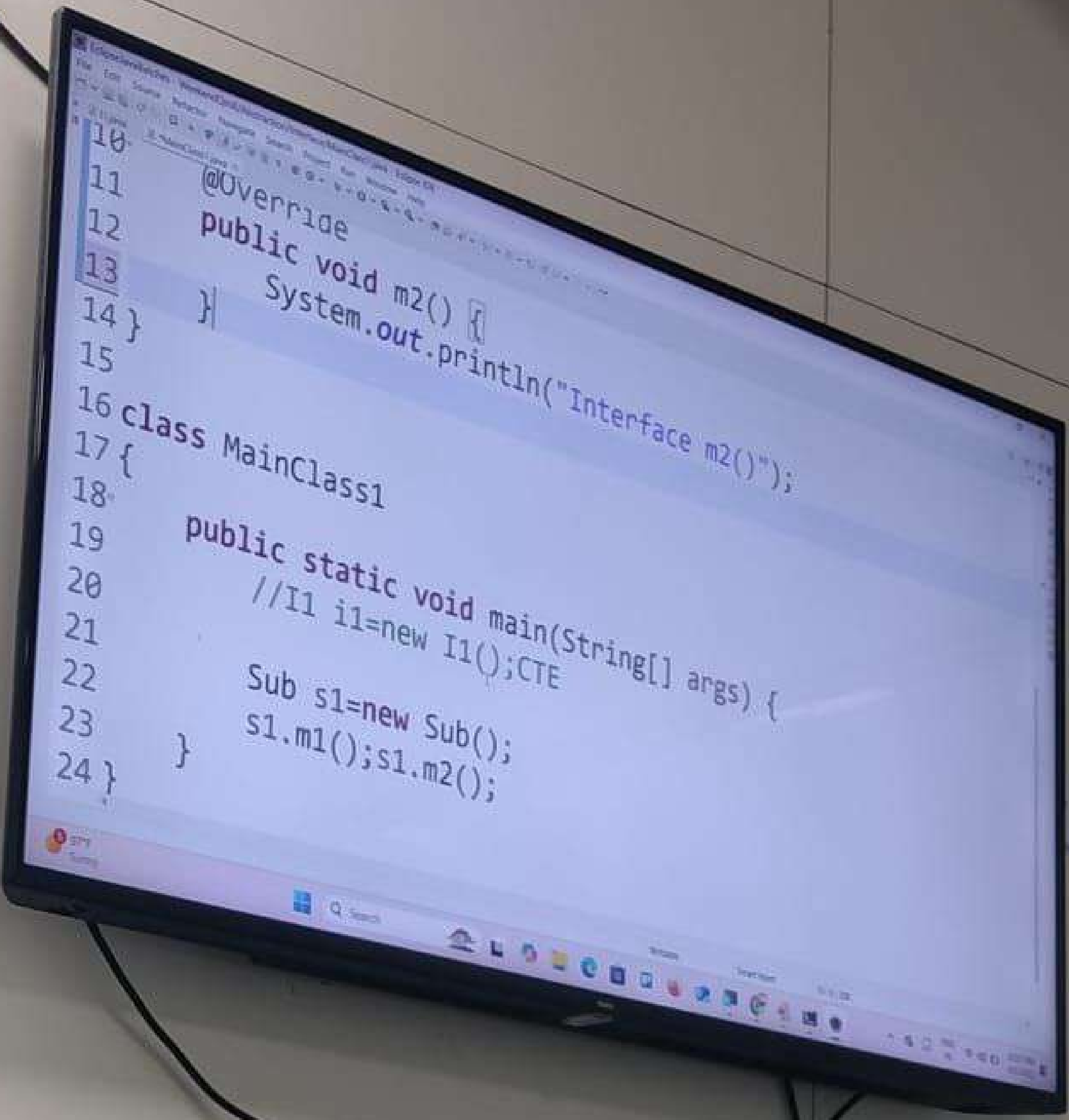
```
public class MainDriver1 {
    public static void main(String[] args) {
        //WhatsApp w1=new WhatsApp();CTE
        //Upcasting
        
        WhatsApp w1=new User();
        w1.send();
        w1.receive();
        if (w1 instanceof User) {
            //Downcasting
            User u1=(User)w1;
            u1.status();
        }
```



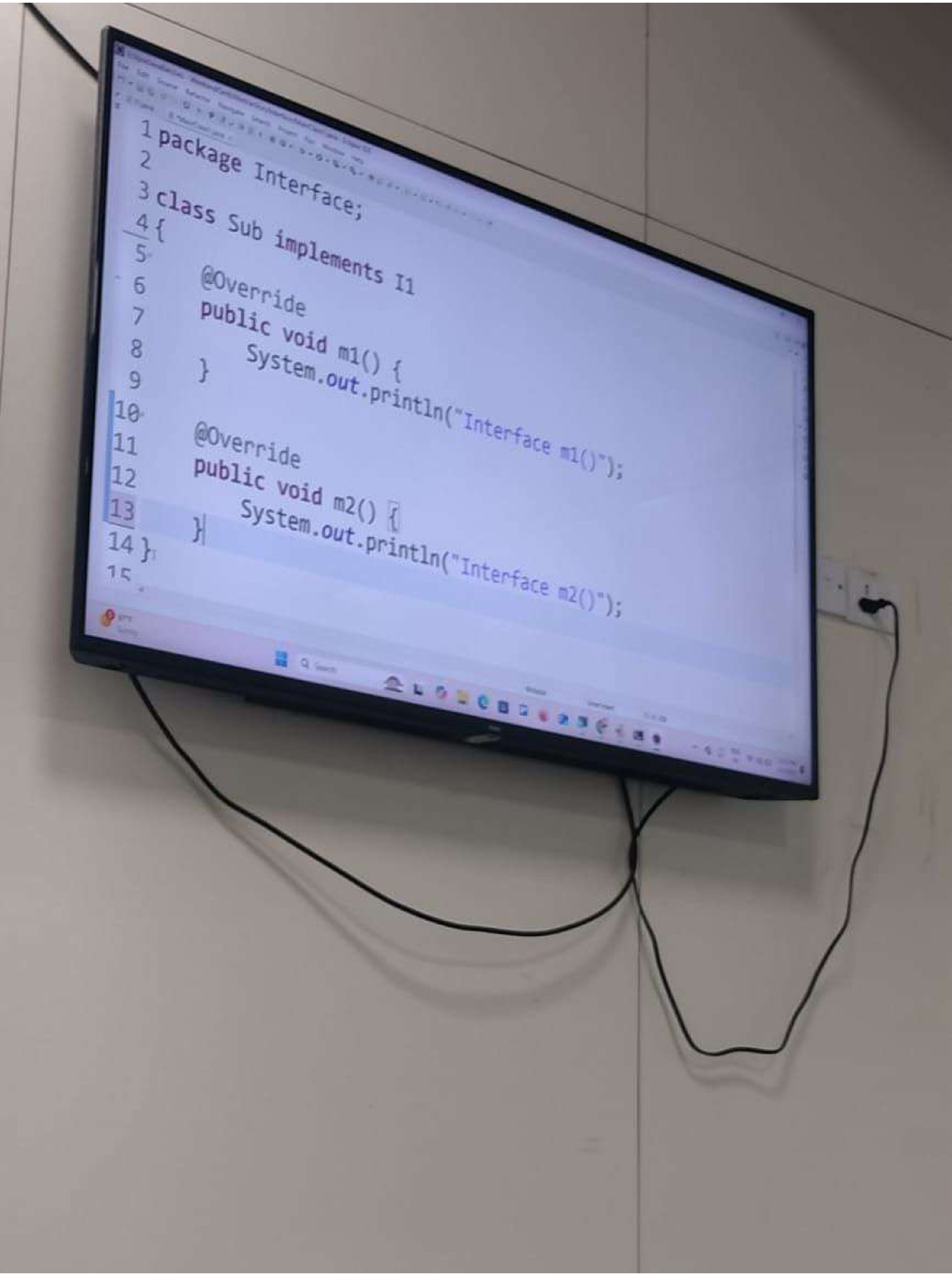
```
package Interface;

public interface I1 {
        abstract void m1();
        void m2();
    }
```



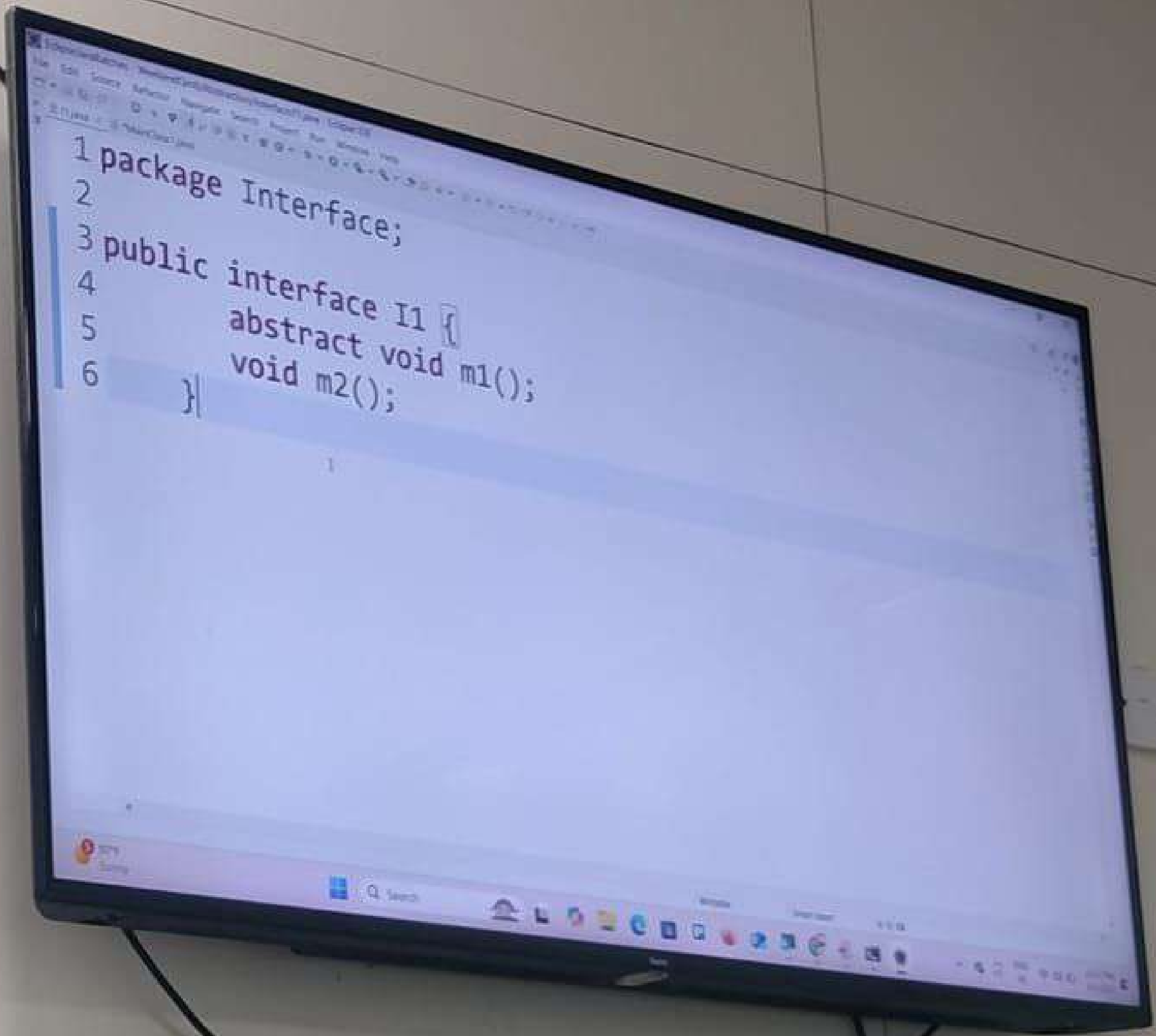
```
package Interface;

class Sub implements I1
{
    @Override
    public void m1() {
        System.out.println("Interface m1()");
    }

    @Override
    public void m2() {
        System.out.println("Interface m2()");
    }
}
```



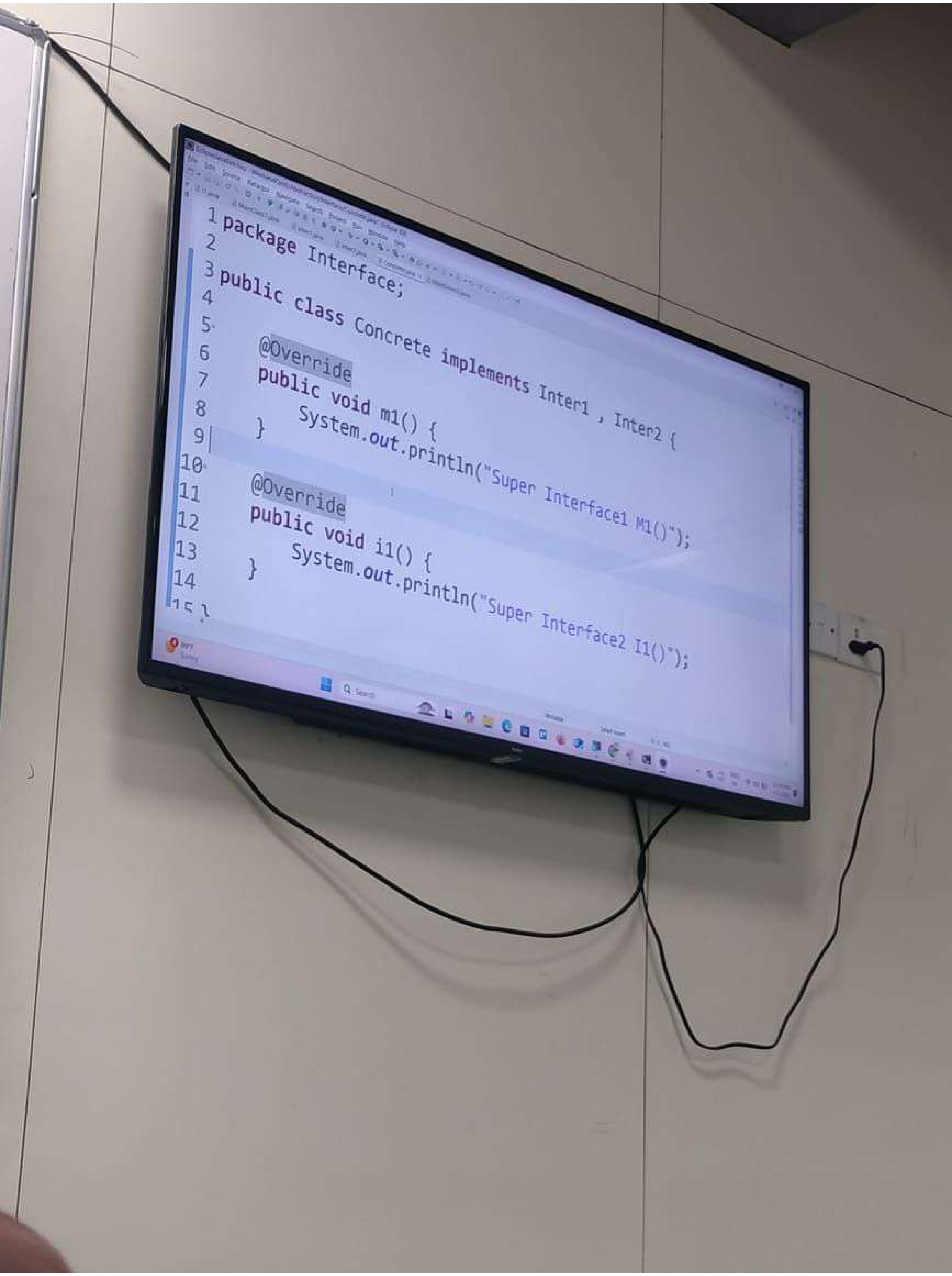
```
    @Override
    public void m2() {
        System.out.println("Interface m2()");
    }
}

class MainClass1
{
    public static void main(String[] args) {
        //I1 i1=new I1();CTE

        Sub s1=new Sub();
        s1.m1();s1.m2();
    }
}
```



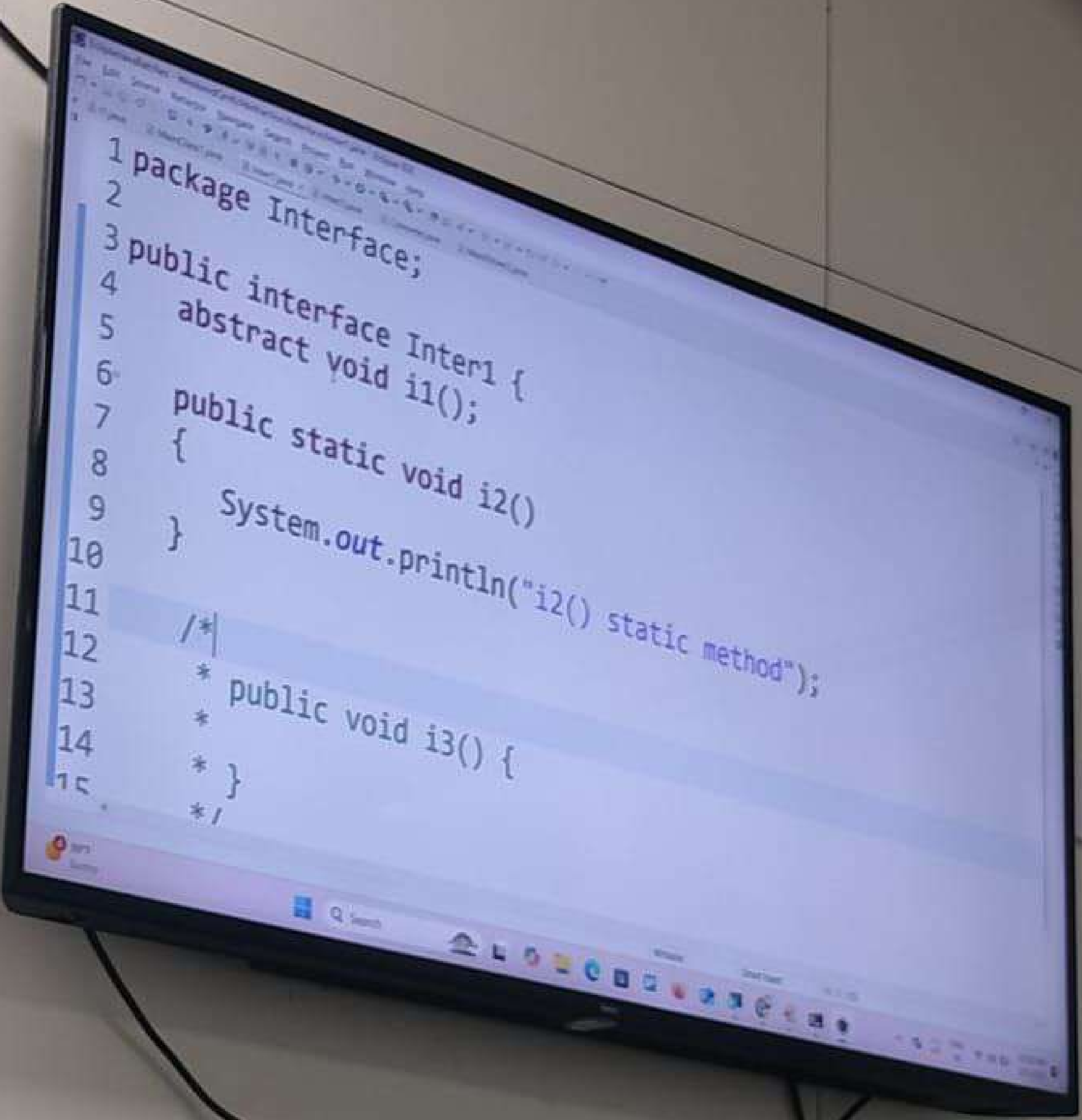
```java
package Interface;

class Sub implements I1
{
	@Override
	public void m1() {
		System.out.println("Interface m1()");
	}

	@Override
	public void m2() {
		System.out.println("Interface m2()");
	}
}
```



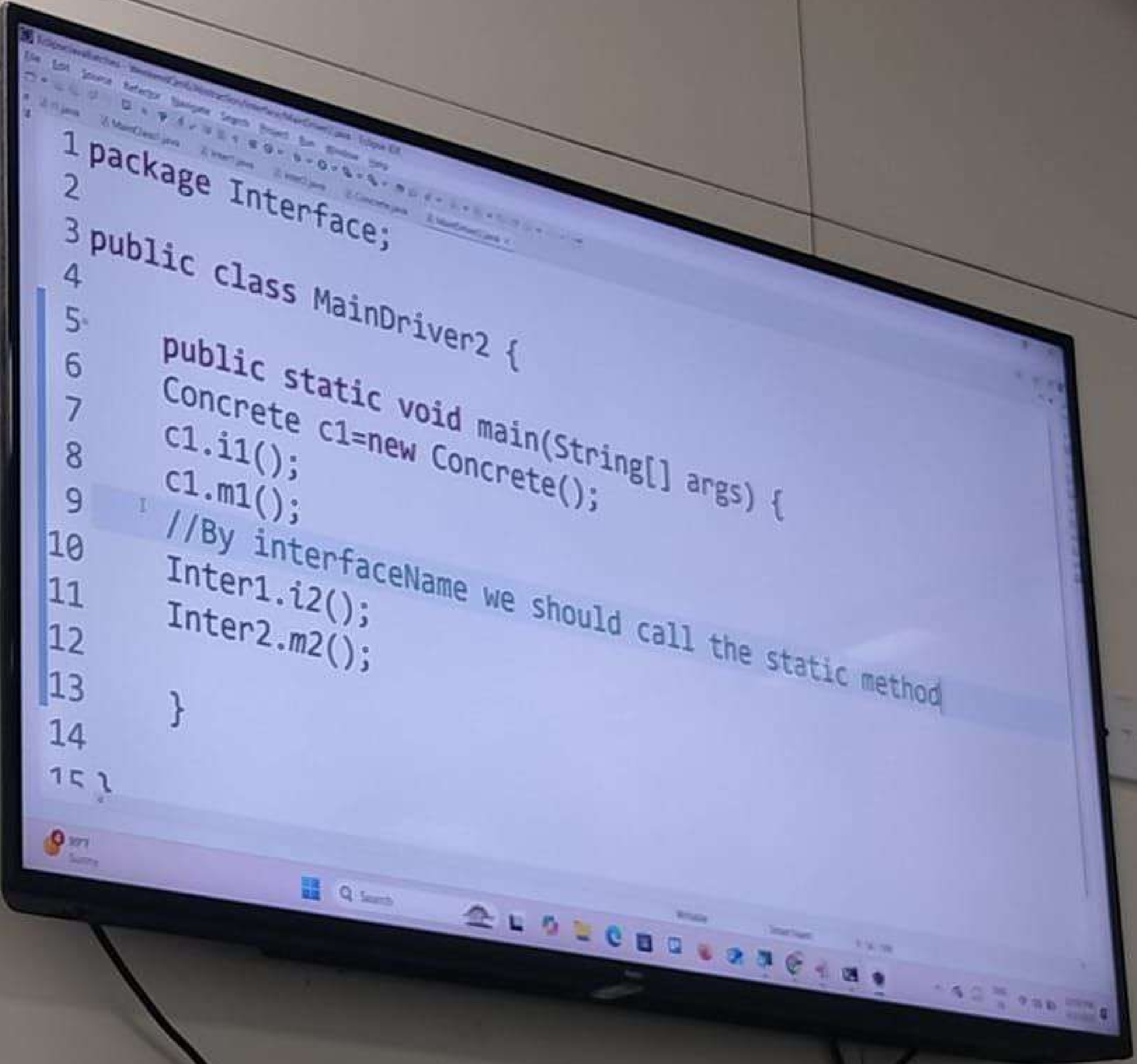
```
package Interface;

public interface I1 {
	abstract void m1();
	void m2();
	}
```



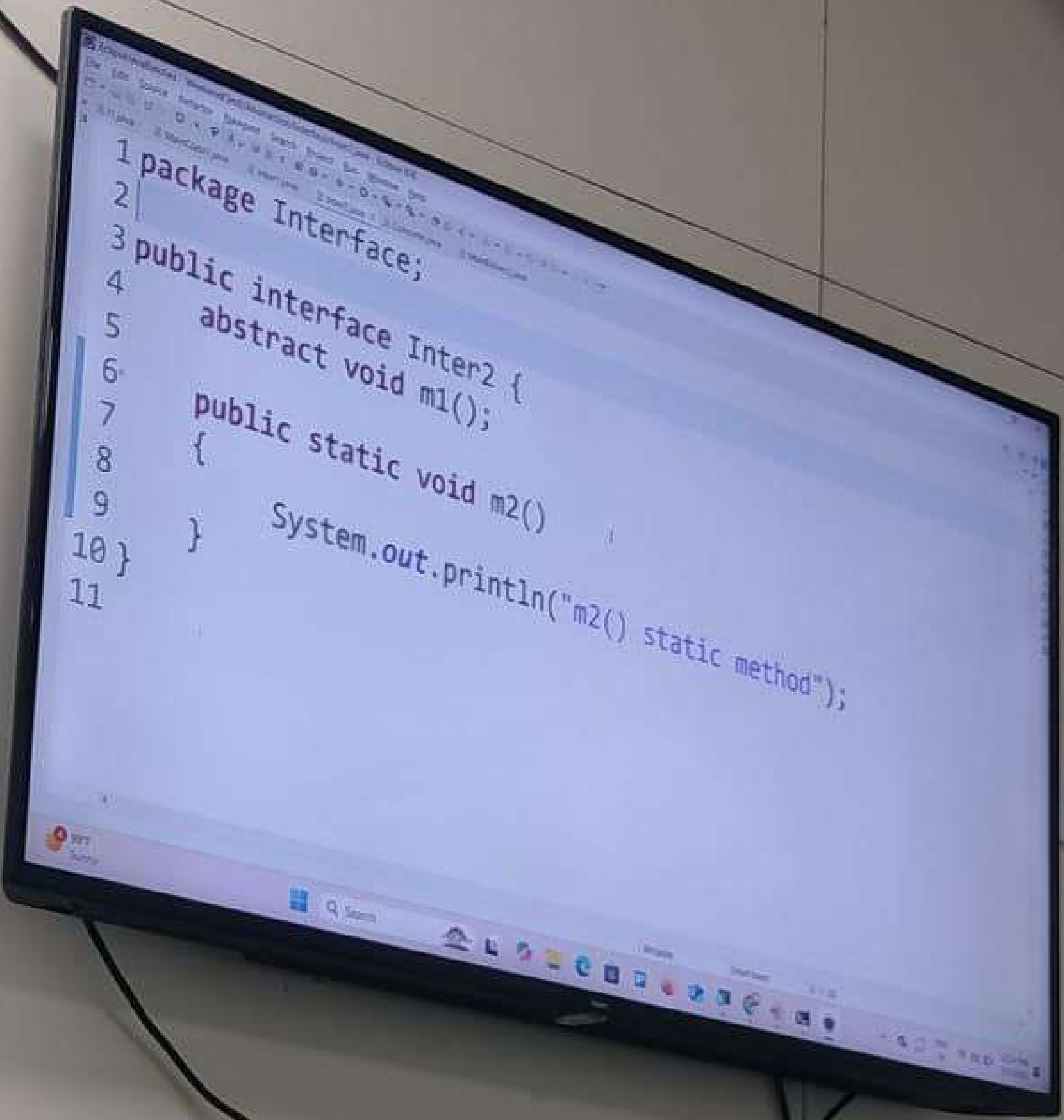
```java
package Interface;

public class Concrete implements Inter1 , Inter2 {

    @Override
    public void m1() {
        System.out.println("Super Interface1 M1()");
    }

    @Override
    public void i1() {
        System.out.println("Super Interface2 I1()");
    }
}
```



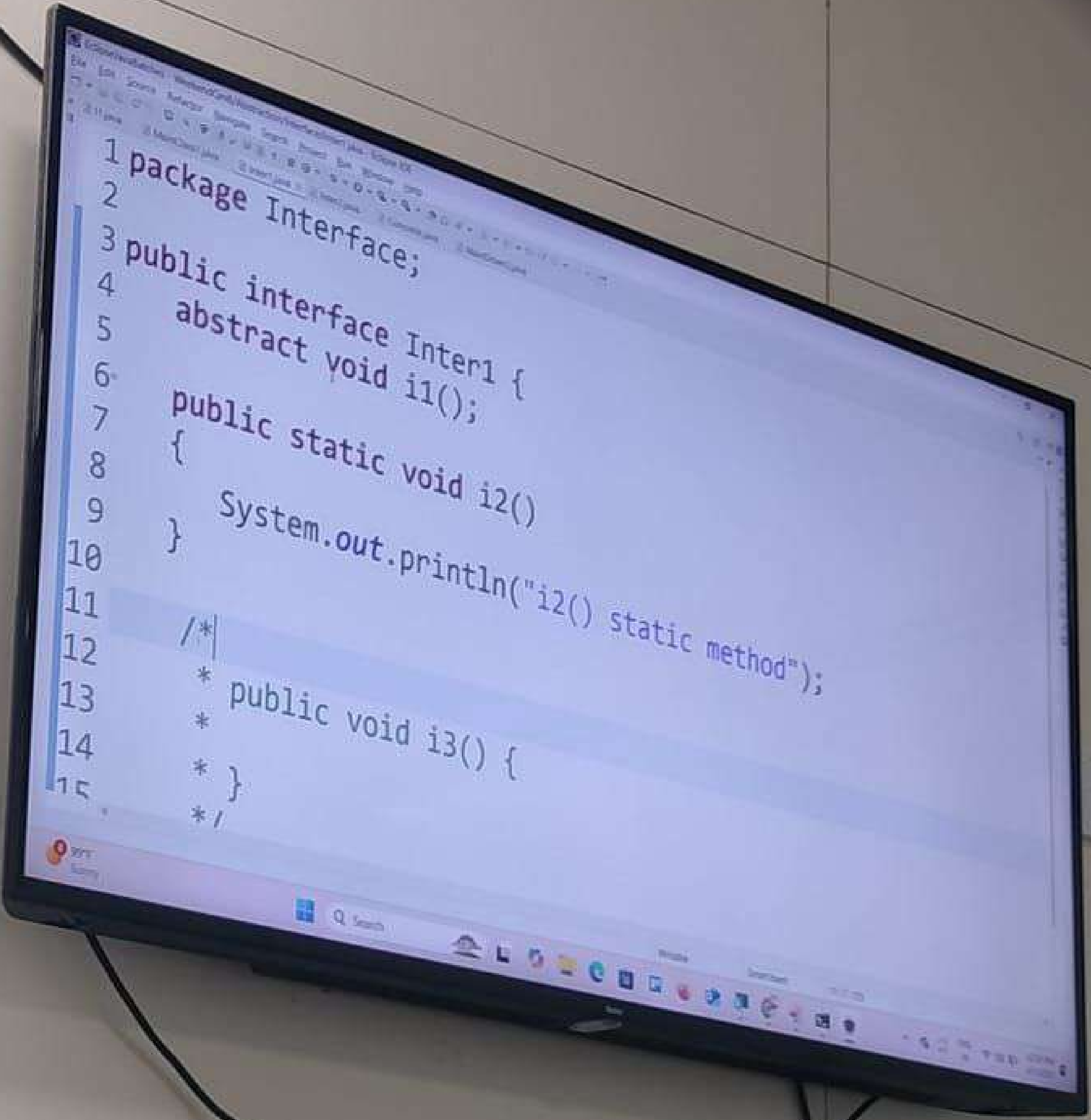
```java
package Interface;

public interface Inter1 {
    abstract void i1();
    public static void i2()
    {
        System.out.println("i2() static method");
    }

    /*
     * public void i3() {
     *
     * }
     */
```



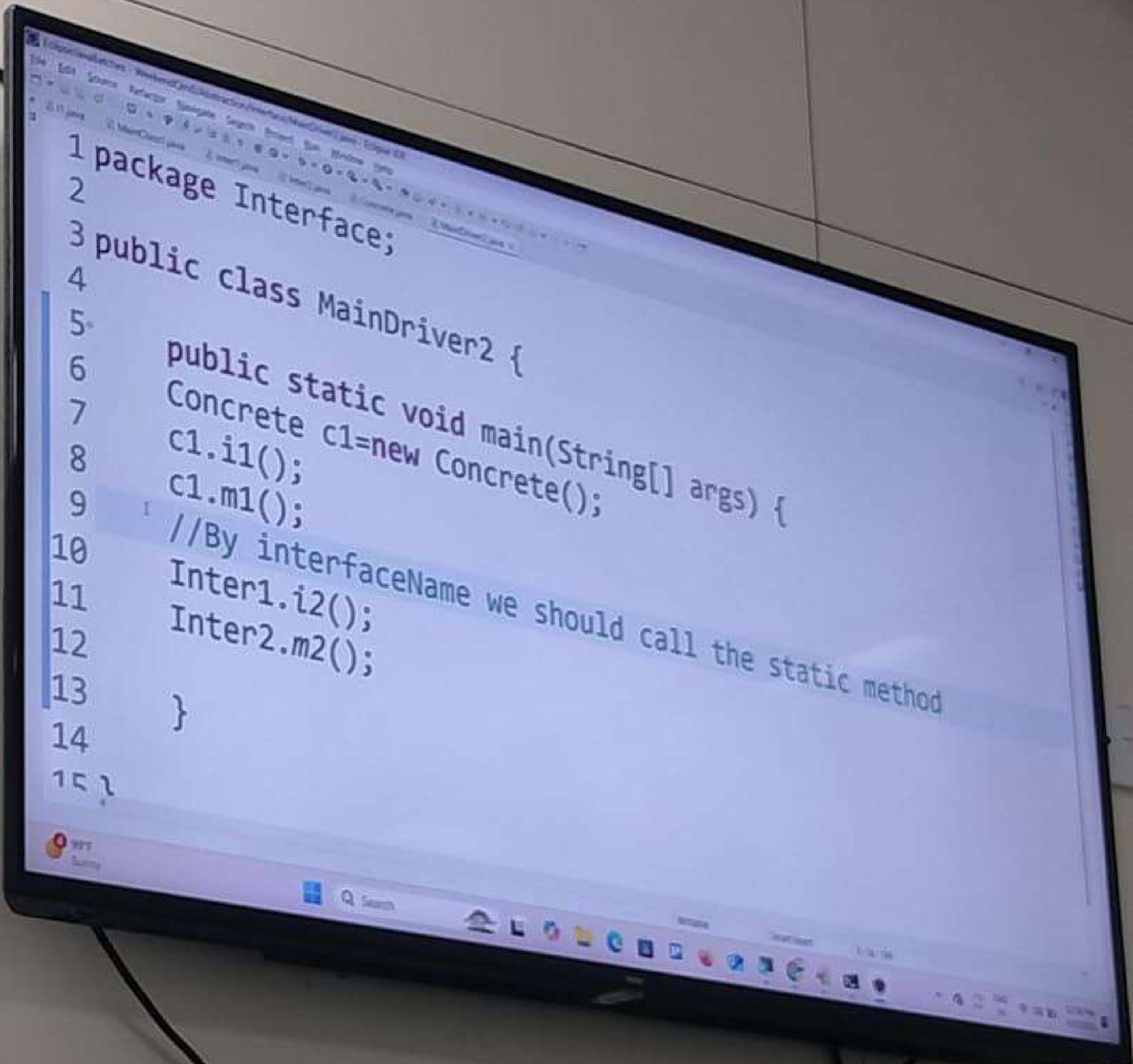
```java
package Interface;

public class MainDriver2 {


    public static void main(String[] args) {
    Concrete c1=new Concrete();
    c1.i1();
    c1.m1();
    //By interfaceName we should call the static method
    Inter1.i2();
    Inter2.m2();

    }

}
```



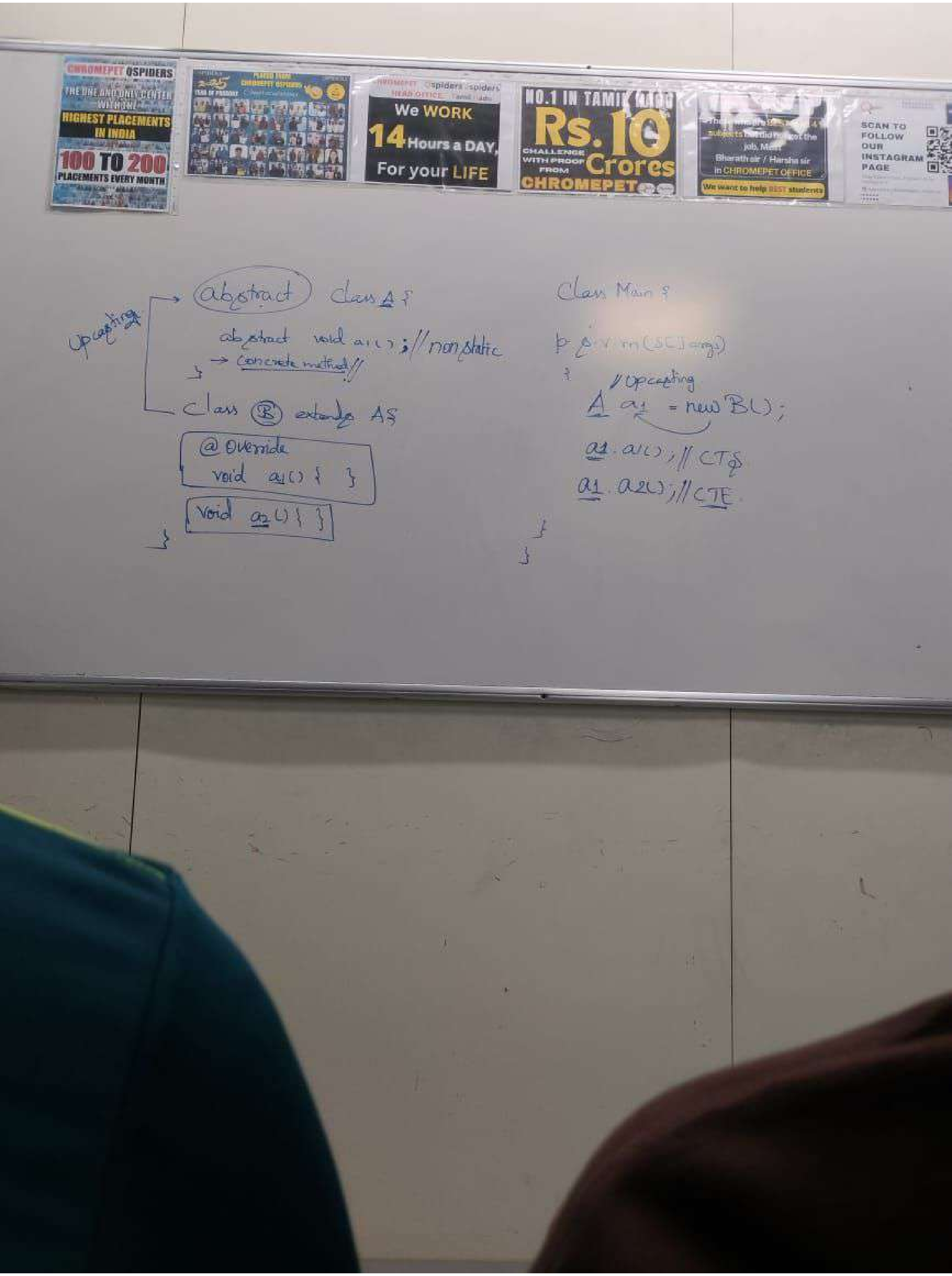
```
package Interface;

public interface Inter2 {
    abstract void m1();
    public static void m2()
    {
        System.out.println("m2() static method");
    }
}
```



```
package Interface;

public interface Inter1 {
    abstract void i1();
    public static void i2()
    {
        System.out.println("i2() static method");
    }

    /*
     * public void i3() {
     *
     * }
     */
```

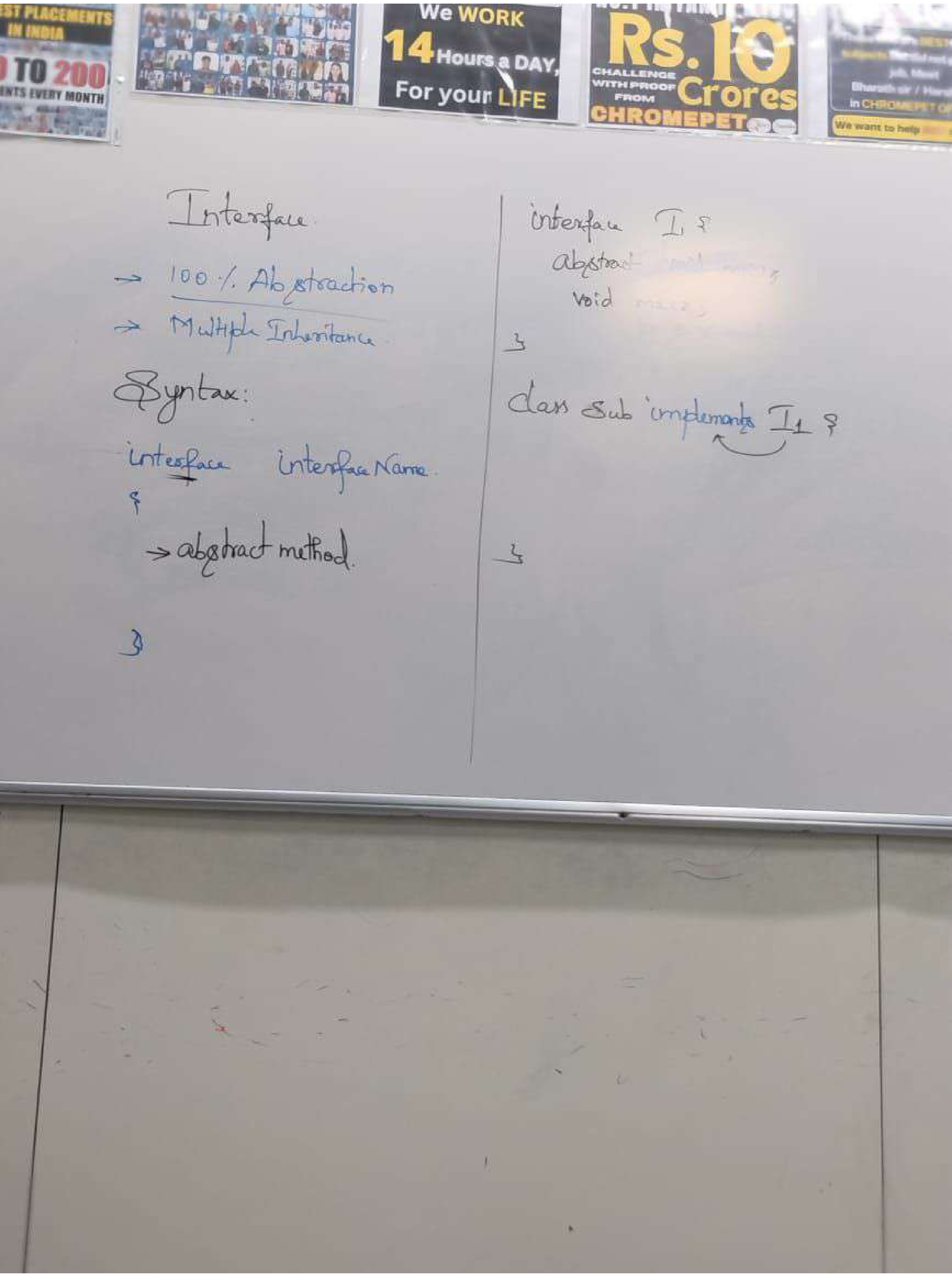


```
package Interface;

public class MainDriver2 {

	public static void main(String[] args) {
	Concrete c1=new Concrete();
	c1.i1();
	c1.m1();
	//By interfaceName we should call the static method
	Inter1.i2();
	Inter2.m2();

	}

}
```



CHROMEPET QSPIDERS
THE ONE AND ONLY CENTER WITH ONLY HIGHEST PLACEMENTS IN INDIA
100 TO 200 PLACEMENTS EVERY MONTH
We WORK 14 Hours a DAY, For your LIFE
NO.1 IN TAMIL NADU
Rs. 10 Crores
CHALLENGE WITH PROOF FROM CHROMEPET
Bharath sir / Harsha sir in CHROMEPET OFFICE
We want to help BEST students
SCAN TO FOLLOW OUR INSTAGRAM PAGE

```
Upcasting
abstract class A{
    abstract void a1();//non static
    → concrete method//
}
class B extends A{
    @Override
    void a1() { }
    void a2() { }
}

class Main{
    p s v m(S[] args)
    {
        //Upcasting
        A a1 = new B();
        a1.a1();//CTS
        a1.a2();//CTE
    }
}
```



PLACEMENTS
IN INDIA
TO 200
ENTS EVERY MONTH
We WORK
14 Hours a DAY,
For your LIFE
Rs.10
CHALLENGE
WITH PROOF
FROM
Crores
CHROMEPET
in CHROMEPET
We want to help

Interface

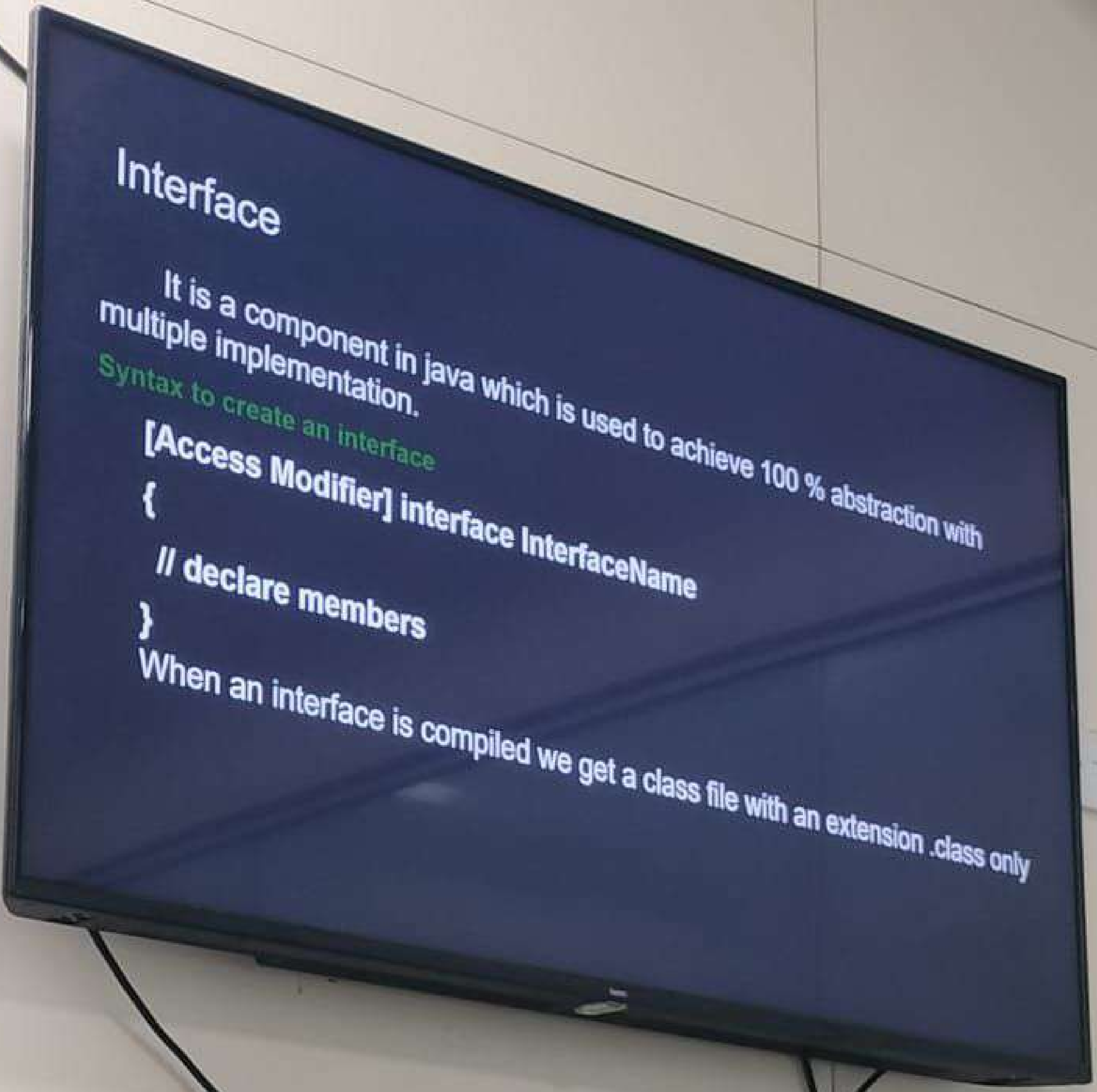
→ 100 % Abstraction

→ Multiple Inheritance

Syntax:

```
interface  interfaceName
{
  → abstract method

}
```

```
interface  I1 {
   abstract [illegible]
     void [illegible]
}

class Sub implements I1 {


}
```



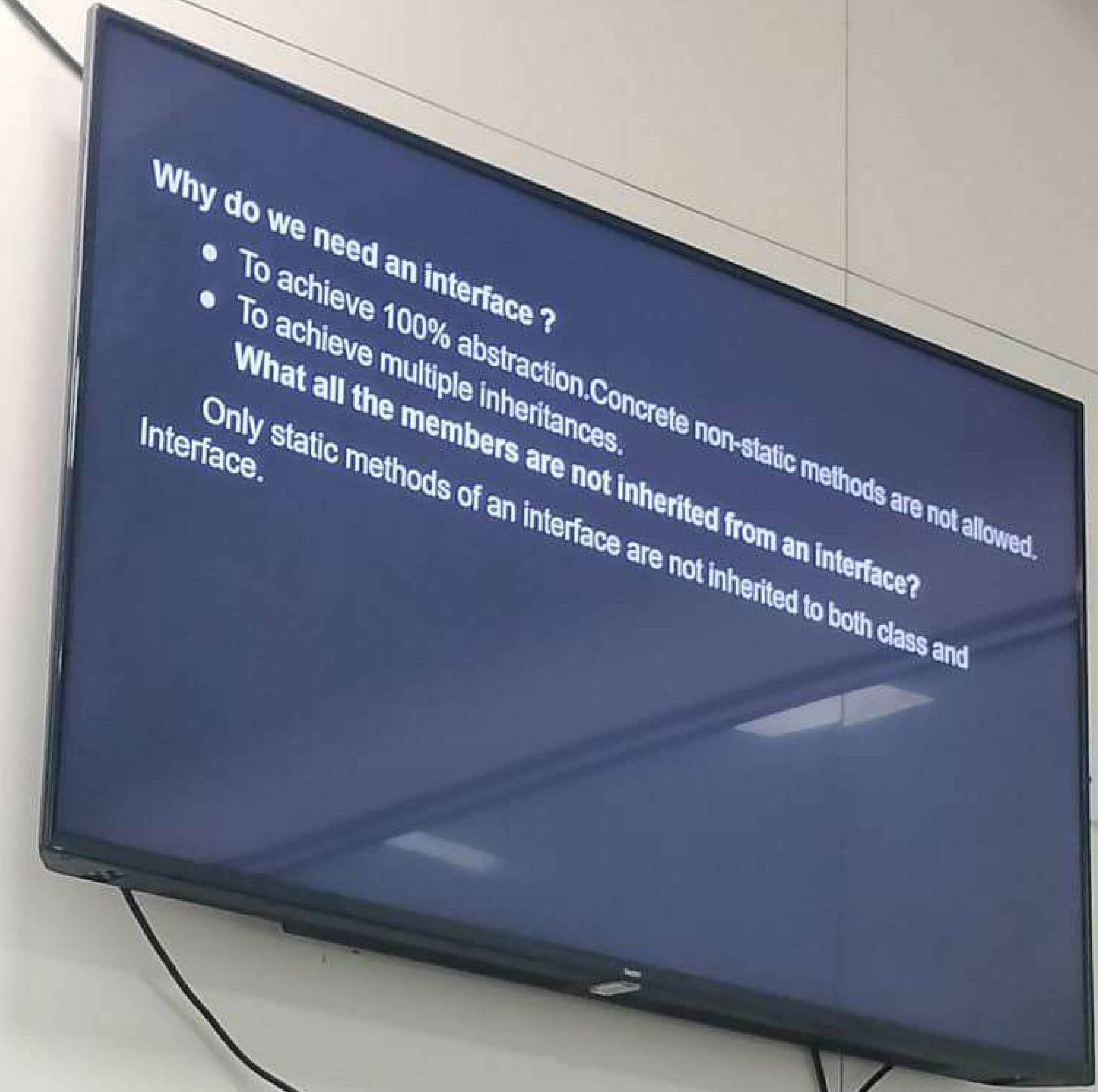
Interface

It is a component in java which is used to achieve 100 % abstraction with multiple implementation.

Syntax to create an interface

```
[Access Modifier] interface InterfaceName
{
  // declare members
}
```

When an interface is compiled we get a class file with an extension .class only

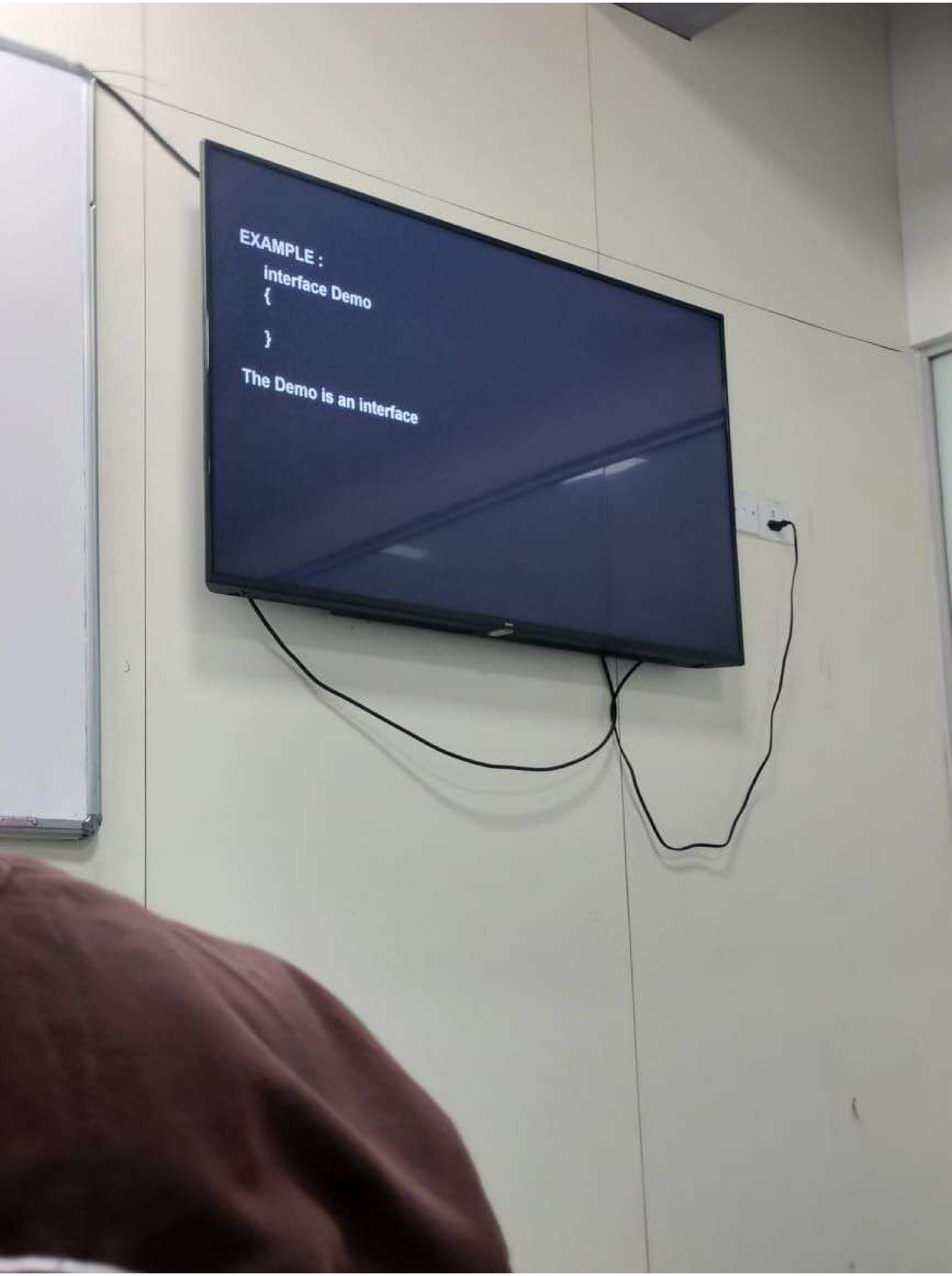


Why do we need an interface ?

- To achieve 100% abstraction.Concrete non-static methods are not allowed.
- To achieve multiple inheritances.

What all the members are not inherited from an interface?

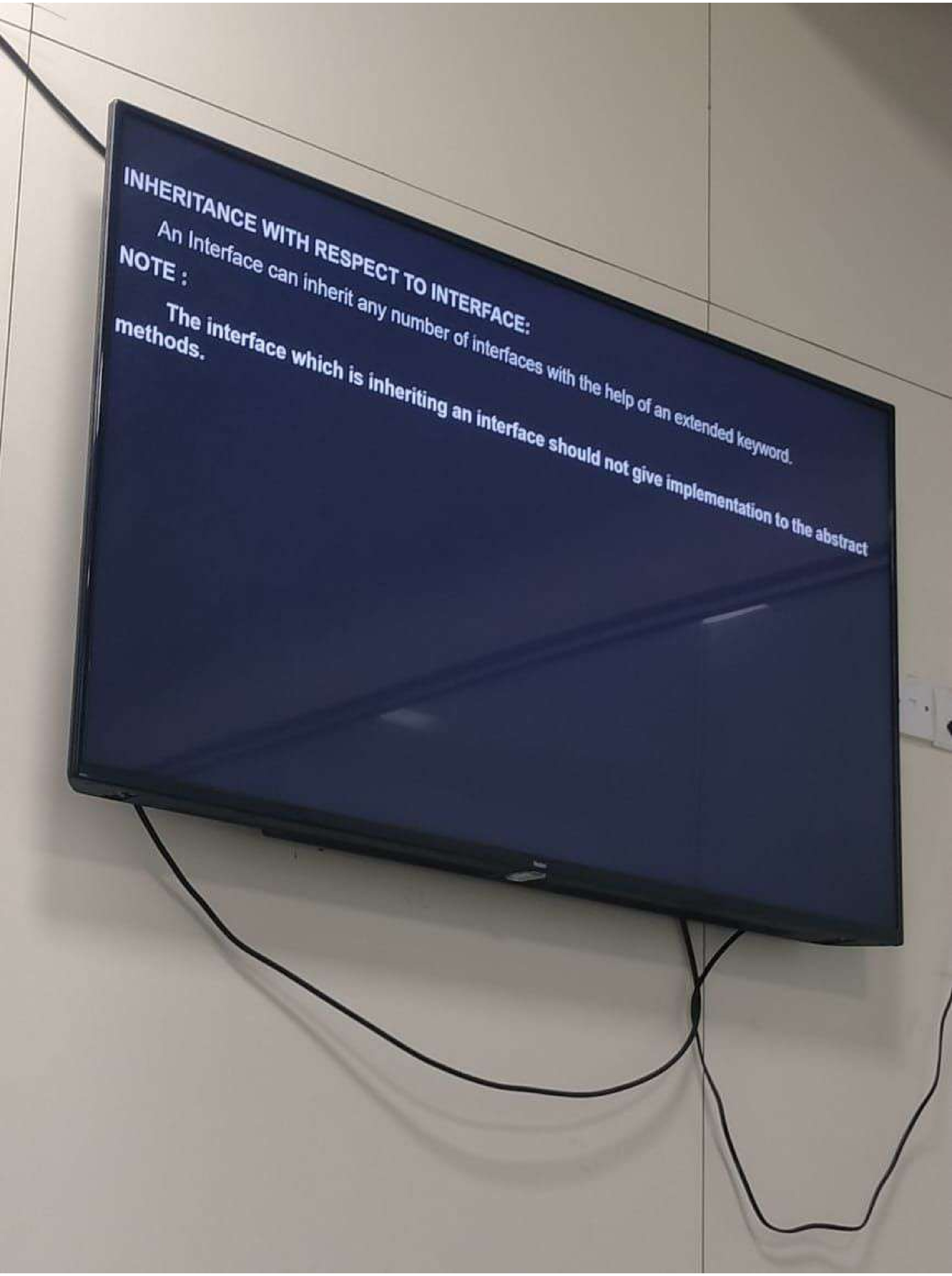
Only static methods of an interface are not inherited to both class and Interface.



EXAMPLE :

```
interface Demo
{

}
```

The Demo is an interface

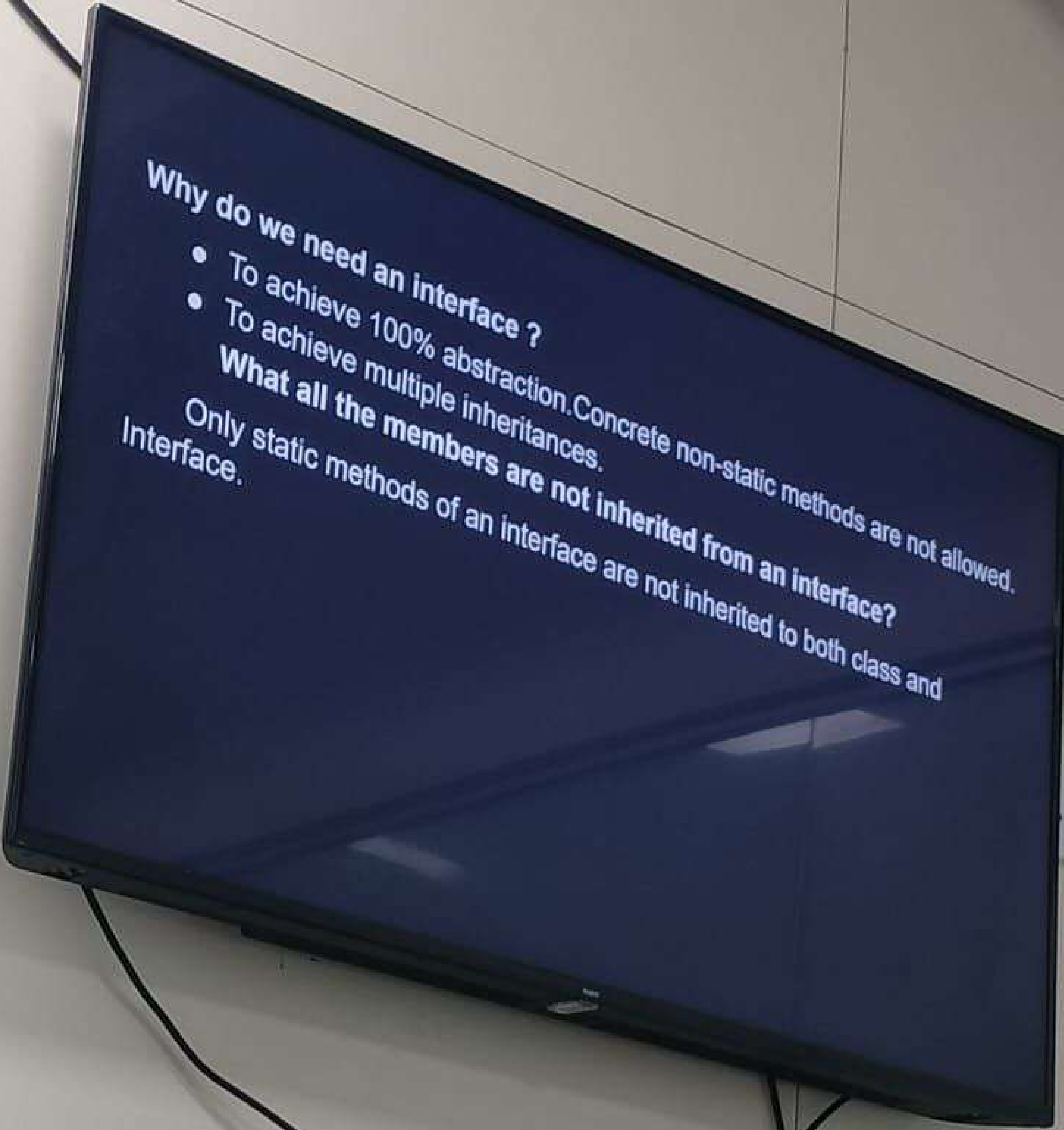


INHERITANCE WITH RESPECT TO INTERFACE:

An Interface can inherit any number of interfaces with the help of an extended keyword.

NOTE :

The interface which is inheriting an interface should not give implementation to the abstract methods.

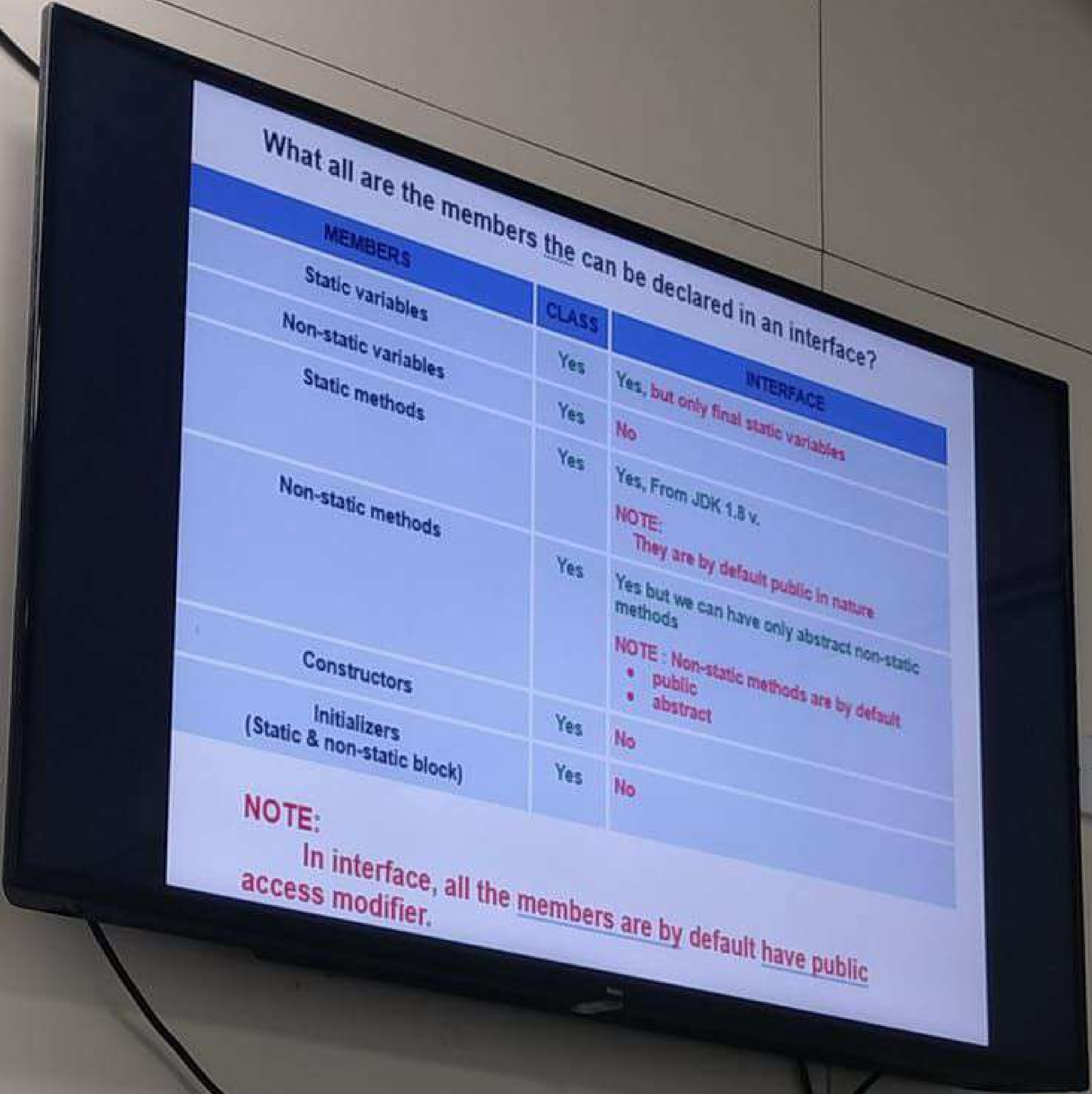


Why do we need an interface ?

- To achieve 100% abstraction.Concrete non-static methods are not allowed.
- To achieve multiple inheritances.

What all the members are not inherited from an interface?

Only static methods of an interface are not inherited to both class and Interface.

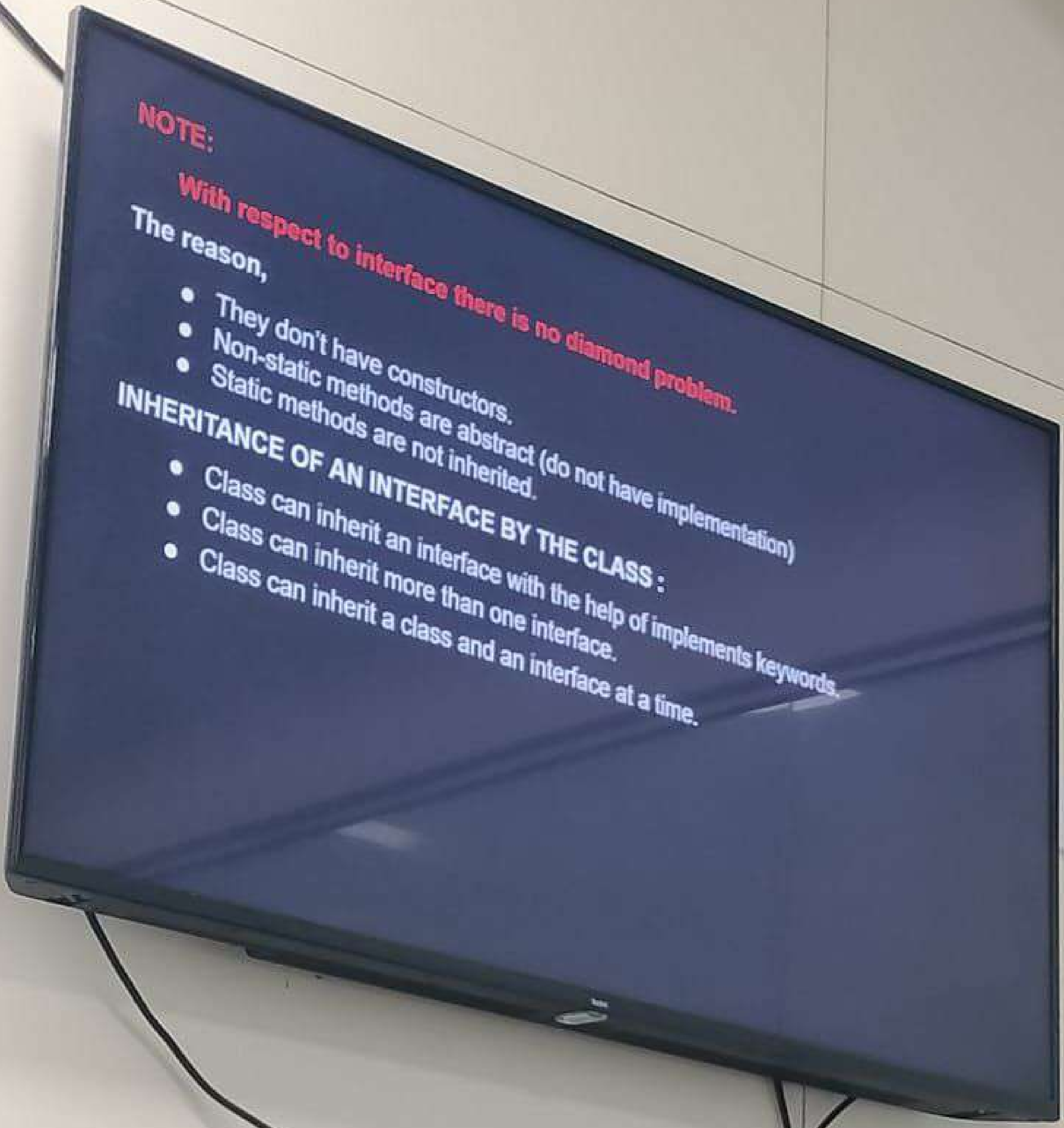


What all are the members the can be declared in an interface?

MEMBERS	CLASS	INTERFACE
Static variables	Yes	Yes, but only final static variables
Non-static variables	Yes	No
Static methods	Yes	Yes, From JDK 1.8 v. NOTE: They are by default public in nature
Non-static methods	Yes	Yes but we can have only abstract non-static methods NOTE : Non-static methods are by default • public • abstract
Constructors	Yes	No
Initializers (Static & non-static block)	Yes	No

NOTE:

In interface, all the members are by default have public access modifier.



NOTE:

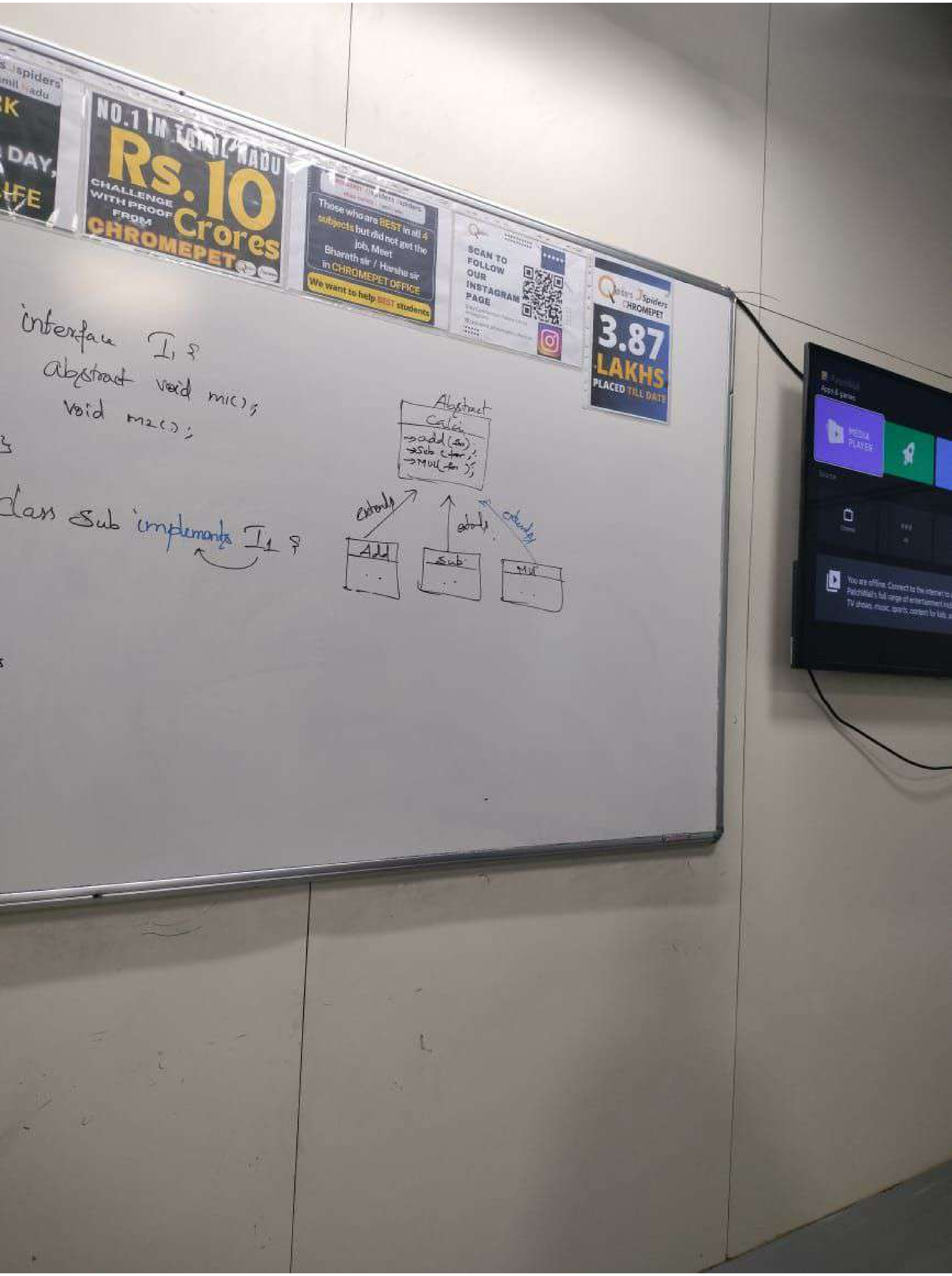
With respect to interface there is no diamond problem.

The reason,

- They don't have constructors.
- Non-static methods are abstract (do not have implementation)
- Static methods are not inherited.

INHERITANCE OF AN INTERFACE BY THE CLASS :

- Class can inherit an interface with the help of implements keywords.
- Class can inherit more than one interface.
- Class can inherit a class and an interface at a time.



```
interface I1 {
    abstract void m1();
        void m2();
}

class Sub implements I1 {
```

Abstract
Calci
->add(...);
->sub(...);
->mul(...);
extends
extends
extends
Add
Sub
Mul
NO.1 IN TAMIL NADU
Rs.10 Crores
CHALLENGE WITH PROOF FROM
CHROMEPET
Those who are BEST in all 4 subjects but did not get the job, Meet Bharath sir / Harsha sir in CHROMEPET OFFICE
We want to help BEST students
SCAN TO FOLLOW OUR INSTAGRAM PAGE
Qspiders Jspiders CHROMEPET
3.87 LAKHS
PLACED TILL DATE
MEDIA PLAYER