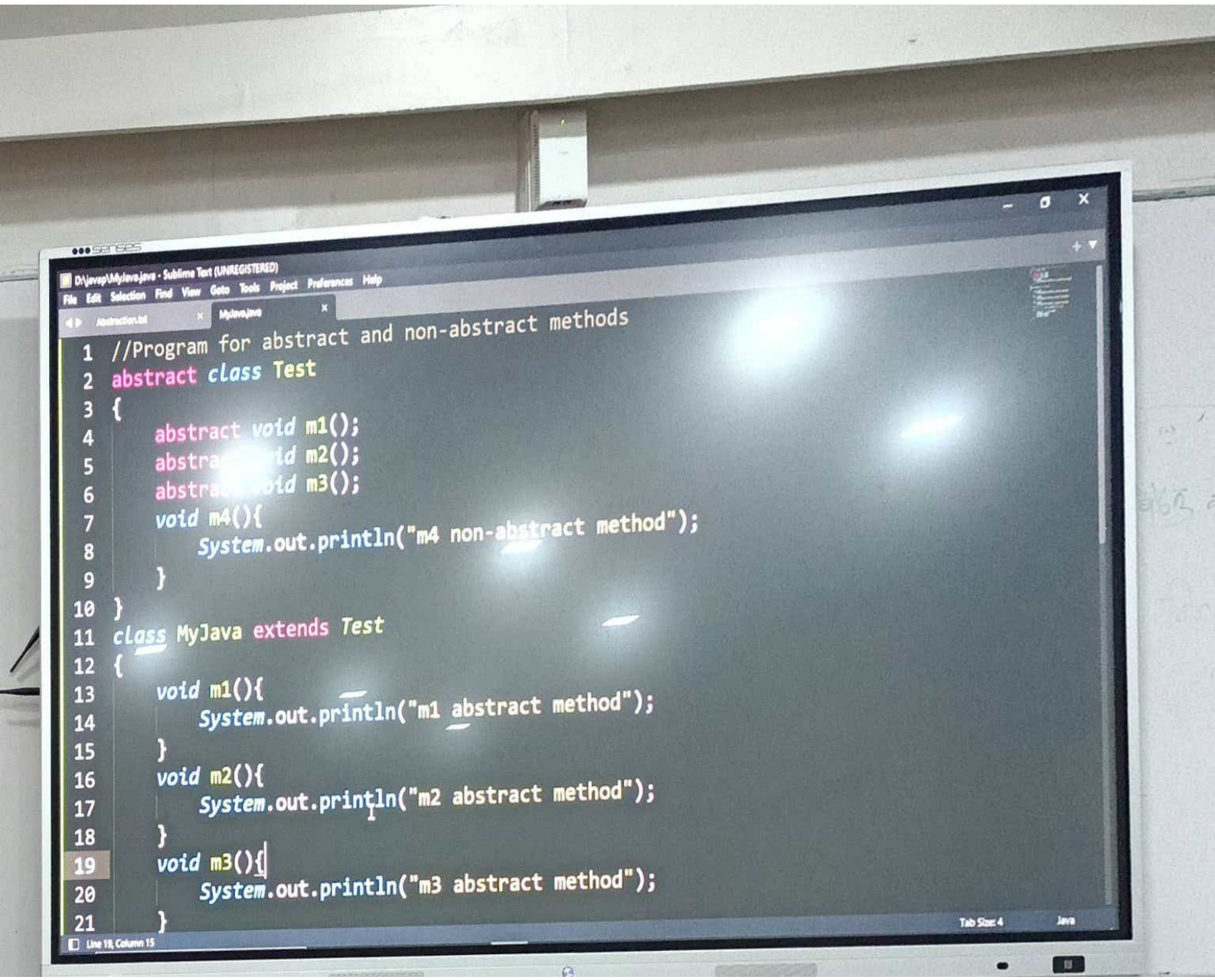
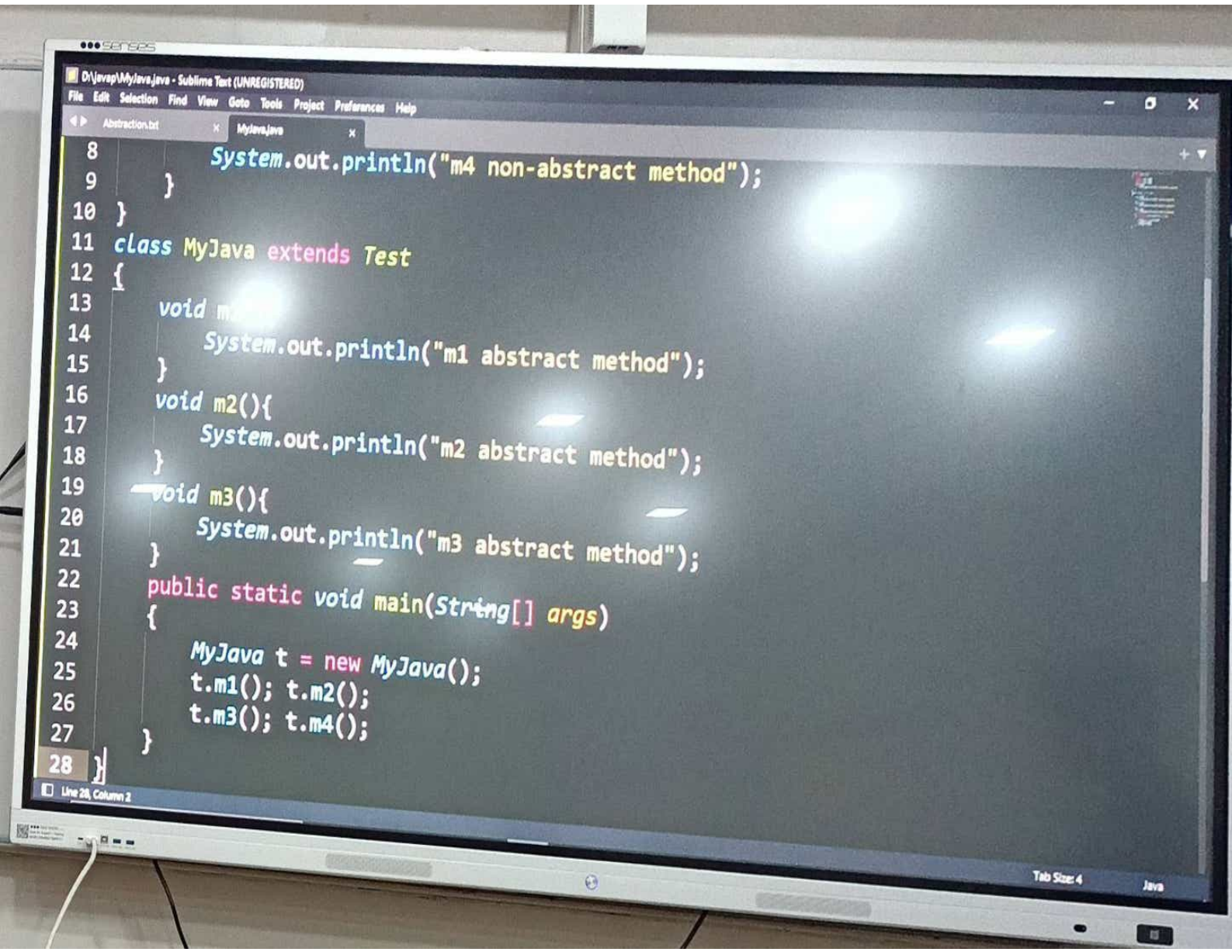
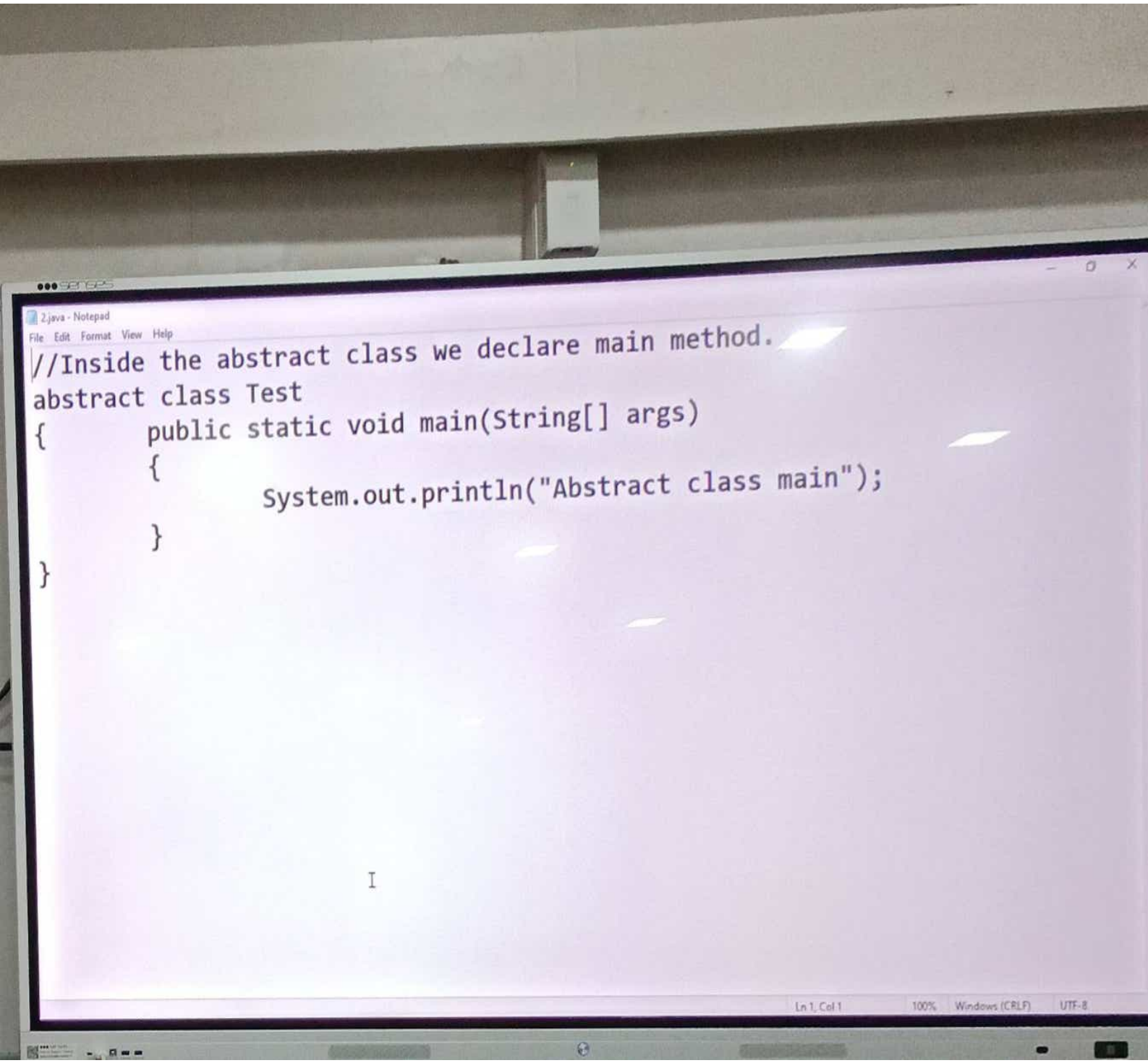
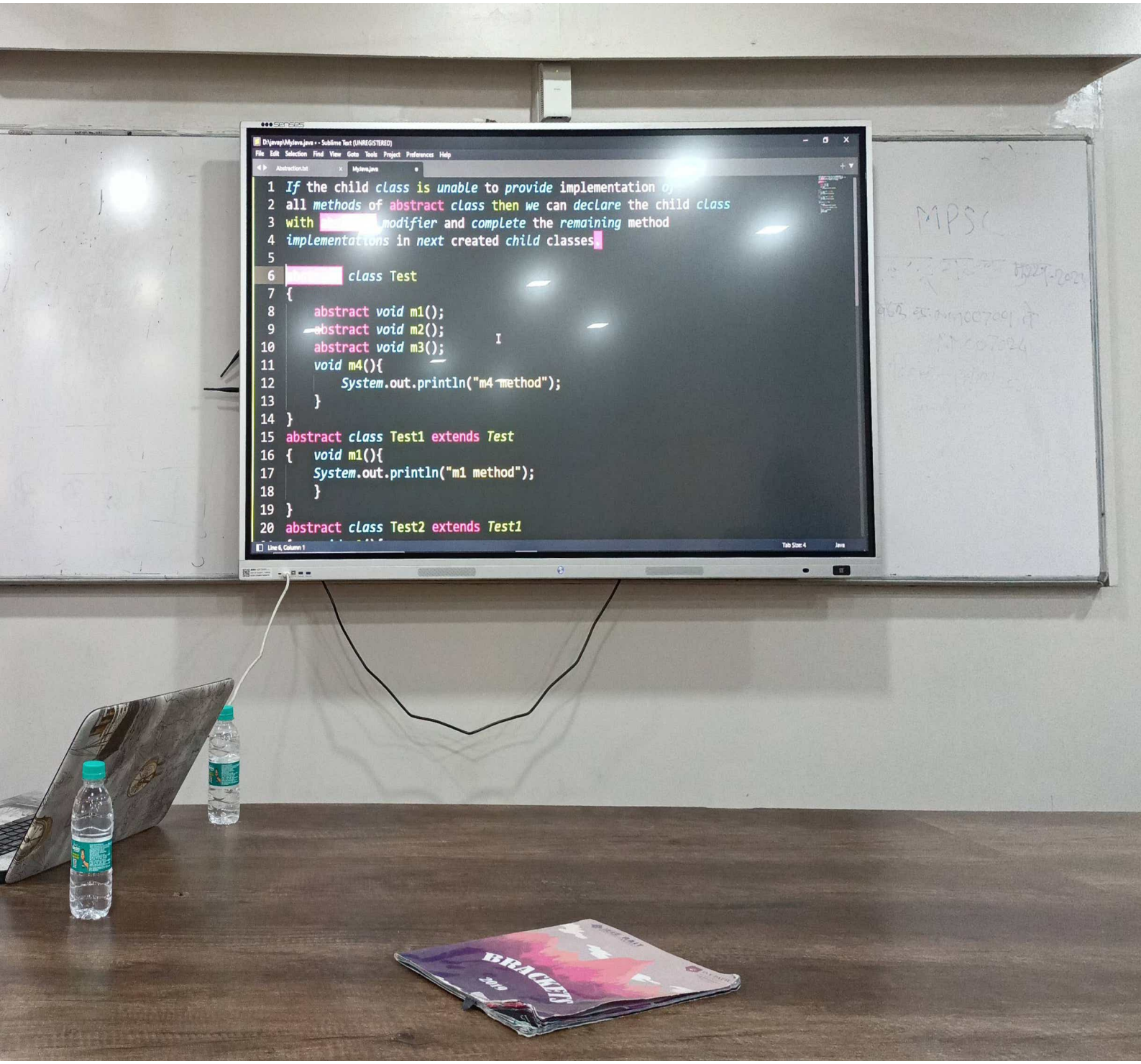
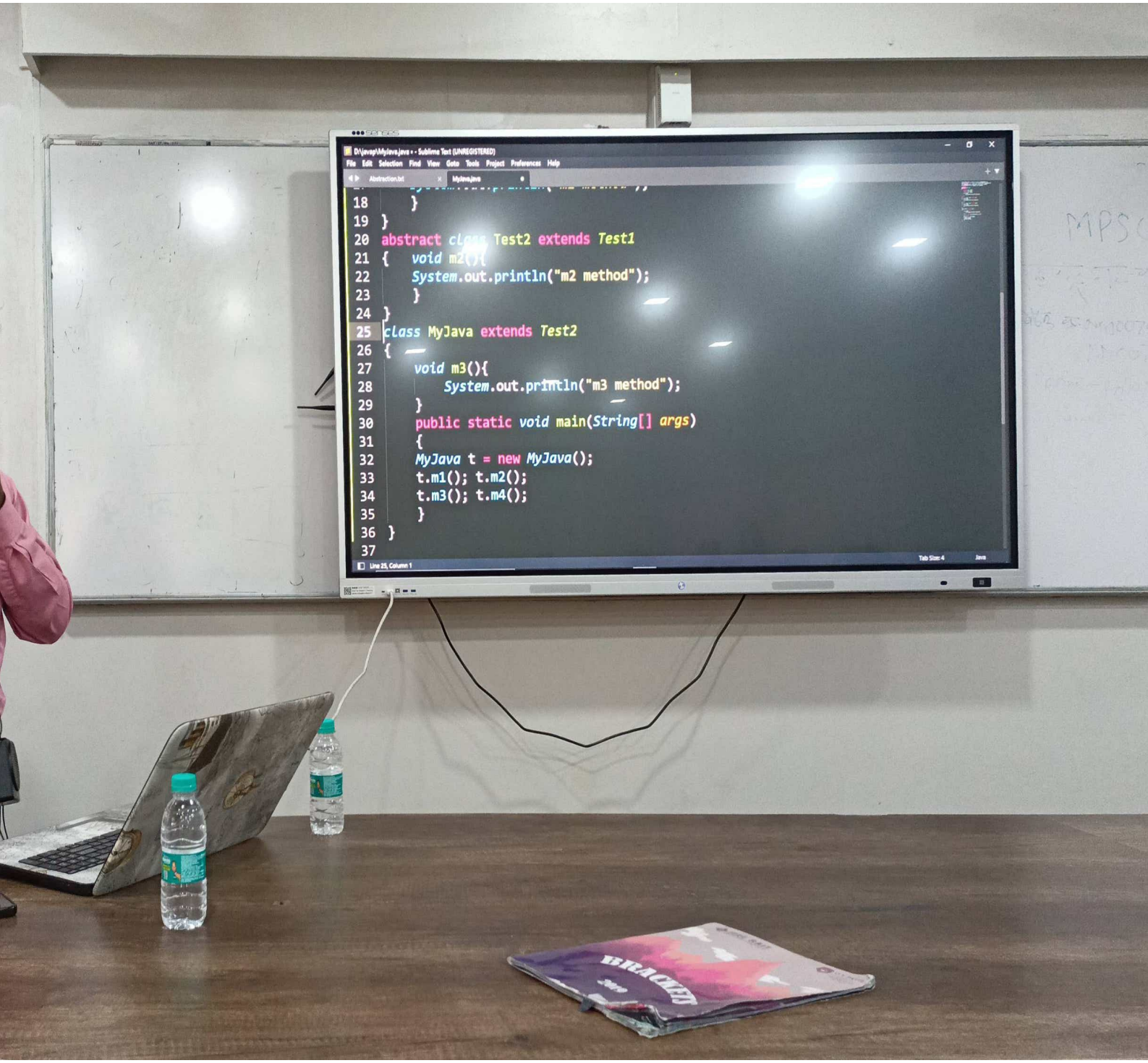
- A class must be prefixed with **abstract** if it has one or more methods with **abstract** keyword known as **abstract class**
- An **abstract** method is only declared but not implemented
- An **abstract class** cannot be instantiated but can be inherited by another class
- Abstract class may contain **abstract** and non **abstract methods**
- The inheriting class must implement all the **abstract** methods or else the subclass should also be declared as **abstract**

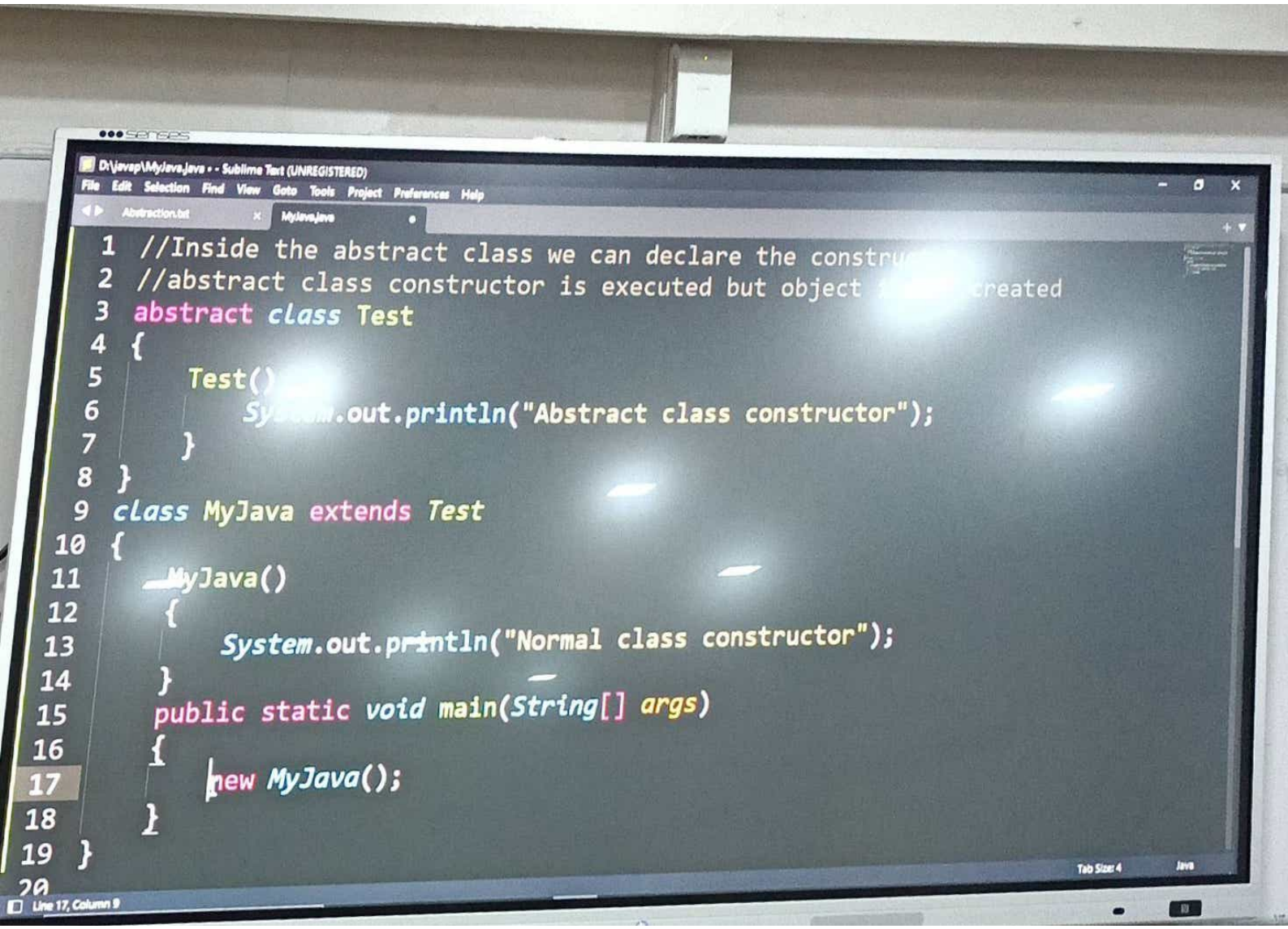

D:\javap\MyJava.java - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
Abstraction.txt
MyJava.java
1 //Program for abstract and non-abstract methods
2 abstract class Test
3 {
4 abstract void m1();
5 abstract void m2();
6 abstract void m3();
7 void m4(){
8 System.out.println("m4 non-abstract method");
9 }
10 }
11 class MyJava extends Test
12 {
13 void m1(){
14 System.out.println("m1 abstract method");
15 }
16 void m2(){
17 System.out.println("m2 abstract method");
18 }
19 void m3(){
20 System.out.println("m3 abstract method");
21 }
Line 18, Column 15
Tab Size: 4
Java


D:\javap\MyJava.java - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
Abstraction.txt
MyJava.java
System.out.println("m4 non-abstract method");
}
}
class MyJava extends Test
{
void m1(){
System.out.println("m1 abstract method");
}
void m2(){
System.out.println("m2 abstract method");
}
void m3(){
System.out.println("m3 abstract method");
}
public static void main(String[] args)
{
MyJava t = new MyJava();
t.m1(); t.m2();
t.m3(); t.m4();
}
}
Line 28, Column 2
Tab Size: 4
Java

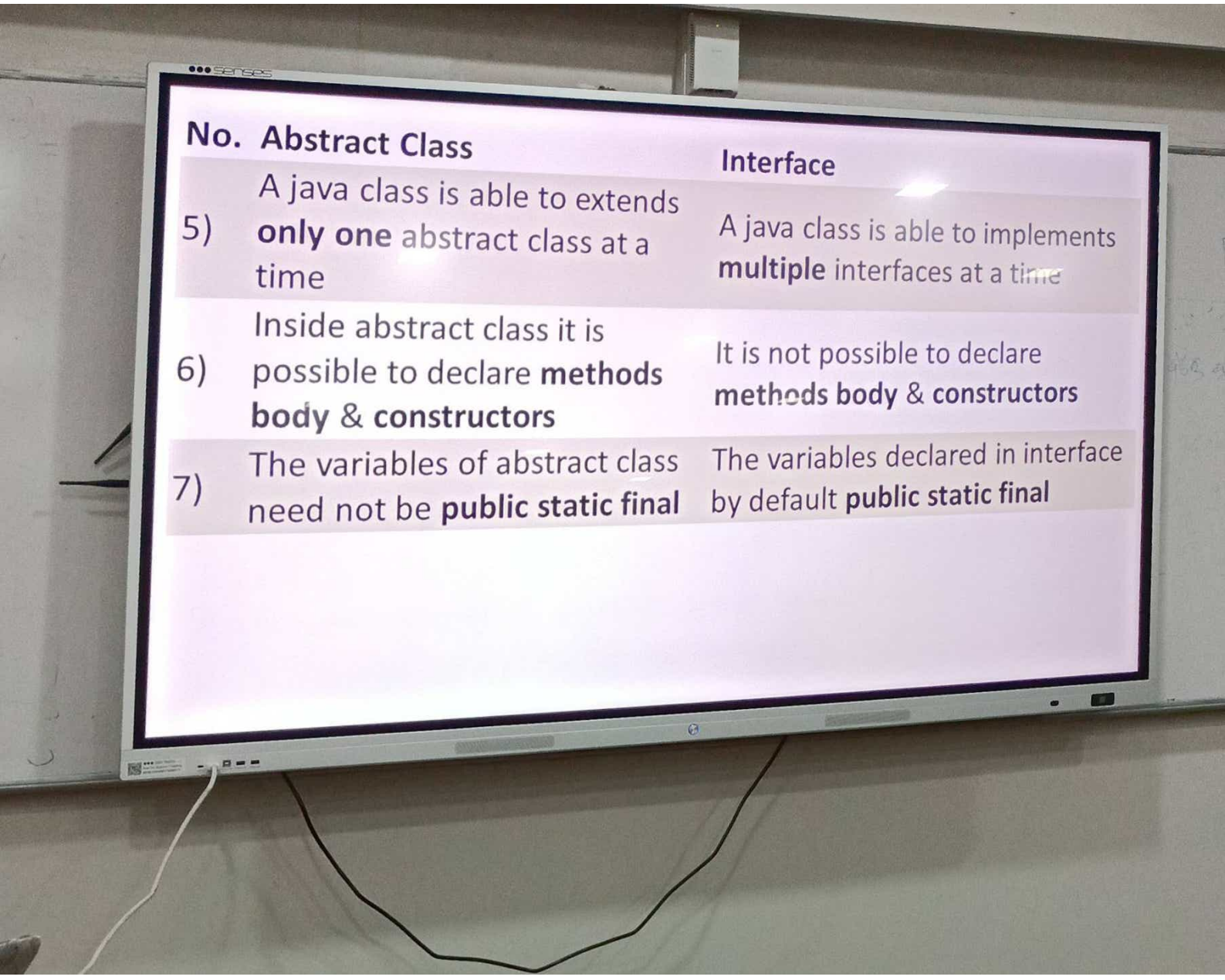

2.java - Notepad
File Edit Format View Help
//Inside the abstract class we declare main method.
abstract class Test
{ public static void main(String[] args)
{
System.out.println("Abstract class main");
}
}
Ln 1, Col 1
100%
Windows (CRLF)
UTF-8


If the child class is unable to provide implementation of
all methods of abstract class then we can declare the child class
with modifier and complete the remaining method
implementations in next created child classes.
class Test
{
abstract void m1();
abstract void m2();
abstract void m3();
void m4(){
System.out.println("m4 method");
}
}
abstract class Test1 extends Test
{ void m1(){
System.out.println("m1 method");
}
}
abstract class Test2 extends Test1
BRACKETS

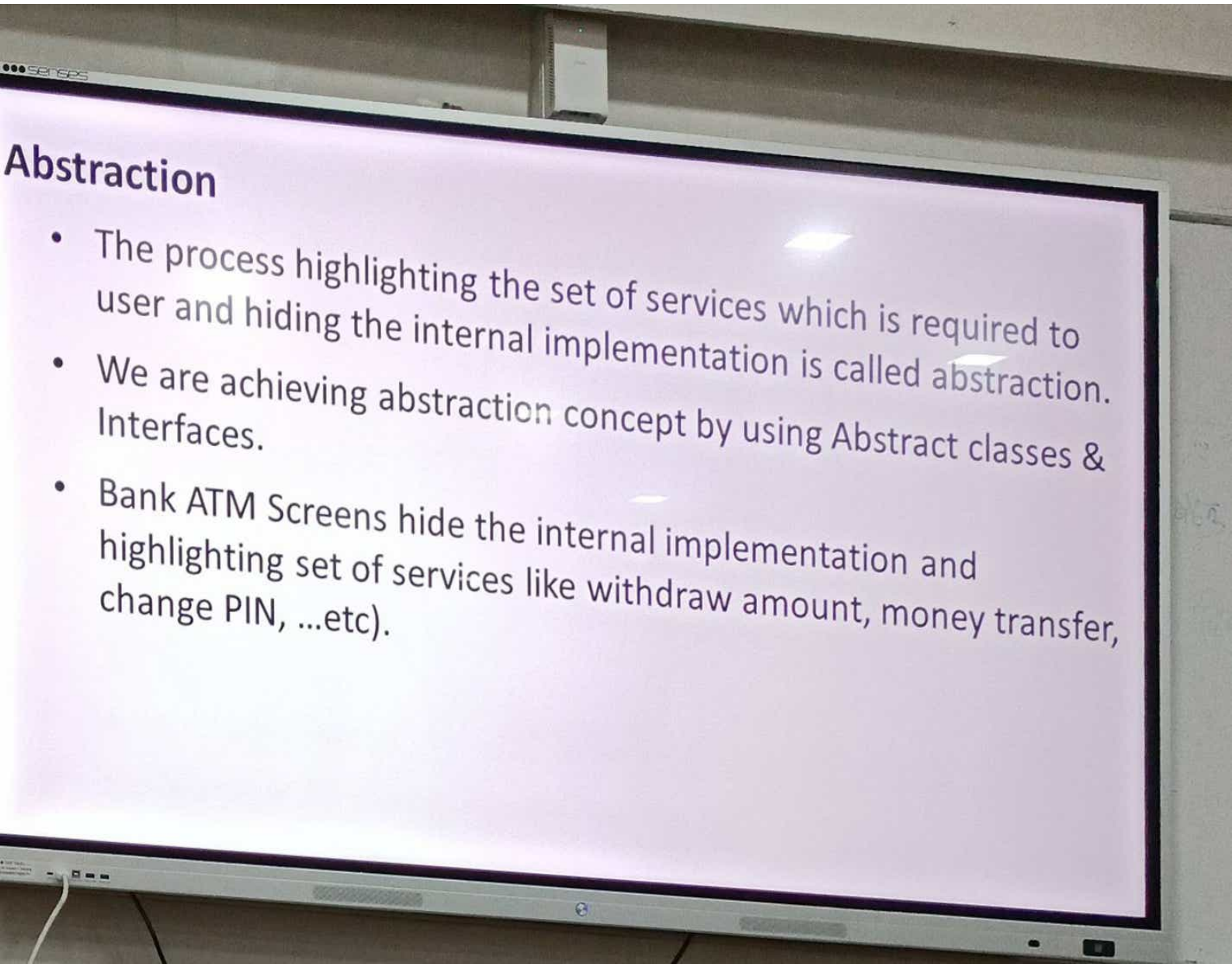

D:\javap\MyJava.java • - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
Abstraction.txt
MyJava.java
}
}
abstract class Test2 extends Test1
{ void m2(){
System.out.println("m2 method");
}
}
class MyJava extends Test2
{
void m3(){
System.out.println("m3 method");
}
public static void main(String[] args)
{
MyJava t = new MyJava();
t.m1(); t.m2();
t.m3(); t.m4();
}
}
Line 25, Column 1
Tab Size: 4
Java


D:\javap\MyJava.java • - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
Abstraction.txt
MyJava.java
1 //Inside the abstract class we can declare the constru
2 //abstract class constructor is executed but object
created
3 abstract class Test
4 {
5 Test()
6 .out.println("Abstract class constructor");
7 }
8 }
9 class MyJava extends Test
10 {
11 MyJava()
12 {
13 System.out.println("Normal class constructor");
14 }
15 public static void main(String[] args)
16 {
17 new MyJava();
18 }
19 }
20
Line 17, Column 9
Tab Size: 4
Java

No.	Abstract Class	Interface
1)	It is declared with **abstract** modifier	It is declare by using **interface** keyword
2)	The abstract allows declaring both abstract & concrete methods	The interface allows declaring only abstract methods
3)	Methods must declare with abstract modifier	Methods are by default **public abstract**
4)	In child class the implementation methods need not be **public** it means while overriding it is possible to declare any valid modifier	In implementation class the implementation methods must be **public**



No.	Abstract Class	Interface
5)	A java class is able to extends **only one** abstract class at a time	A java class is able to implements **multiple** interfaces at a time
6)	Inside abstract class it is possible to declare **methods body** & **constructors**	It is not possible to declare **methods body** & **constructors**
7)	The variables of abstract class need not be **public static final**	The variables declared in interface by default **public static final**


Abstraction
• The process highlighting the set of services which is required to user and hiding the internal implementation is called abstraction.
• We are achieving abstraction concept by using Abstract classes & Interfaces.
• Bank ATM Screens hide the internal implementation and highlighting set of services like withdraw amount, money transfer, change PIN, ...etc).

Encapsulation

- The process of binding the data(variables) and code(methods) as a single unit is called encapsulation.
- The process of hiding the implementation details to user is called encapsulation.
- We are achieving this concept by declaring variables as a private modifier because it is possible to access private members with in the class only but not outside of that class.

Data Hiding

- The main objective is data hiding is security and it is possible to hide the data by using **private modifier**.
- If any variable declared as a **private** it is possible to access those variables only inside the class is called data hiding.

Data Hiding

- The main objective is data hiding is security and it is possible to hide the data by using **private modifier**.
- If any variable declared as a **private** it is possible to access those variables only inside the class is called data hiding.

Fully or Tightly encapsulated class

The class contains only **private properties** that class is said to be tightly encapsulated class.

```
class Emp
{       private int eid;
        private String ename;
}
```