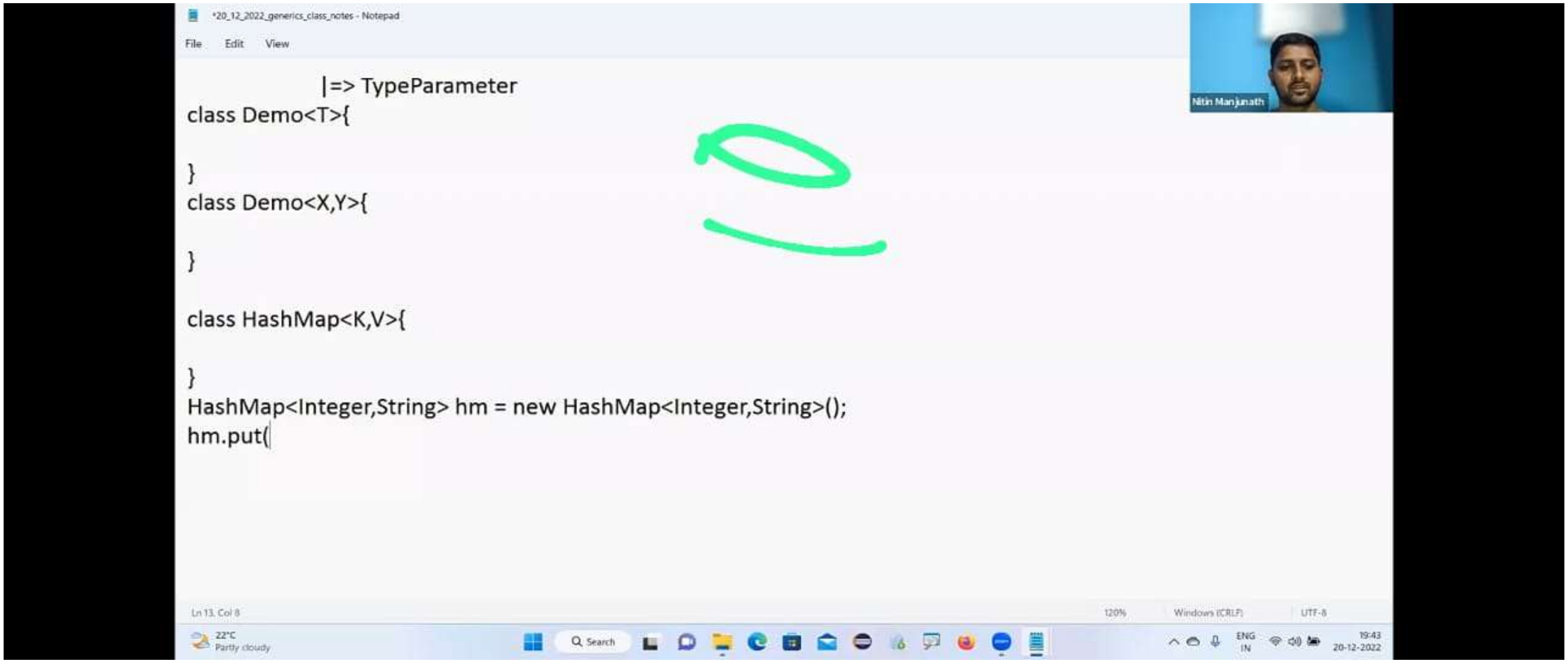
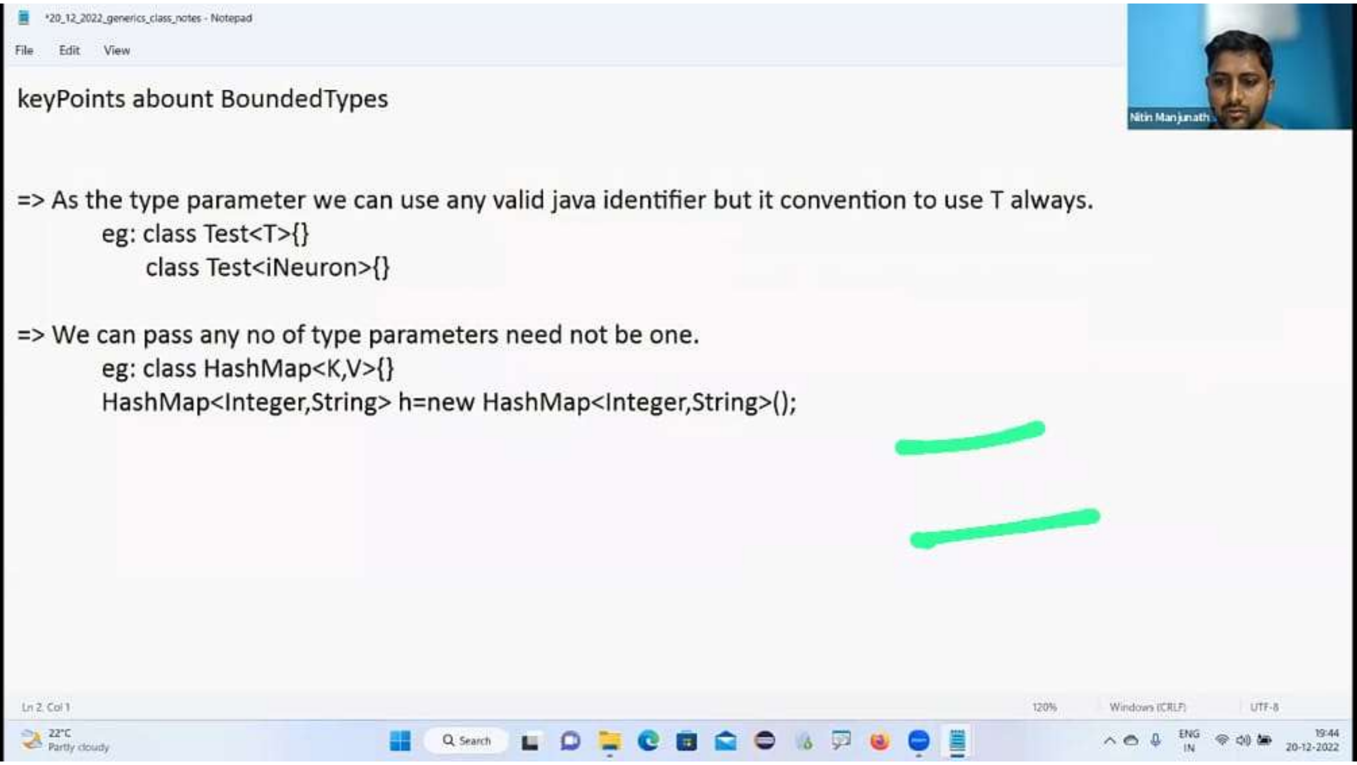
Jenerics Part 2

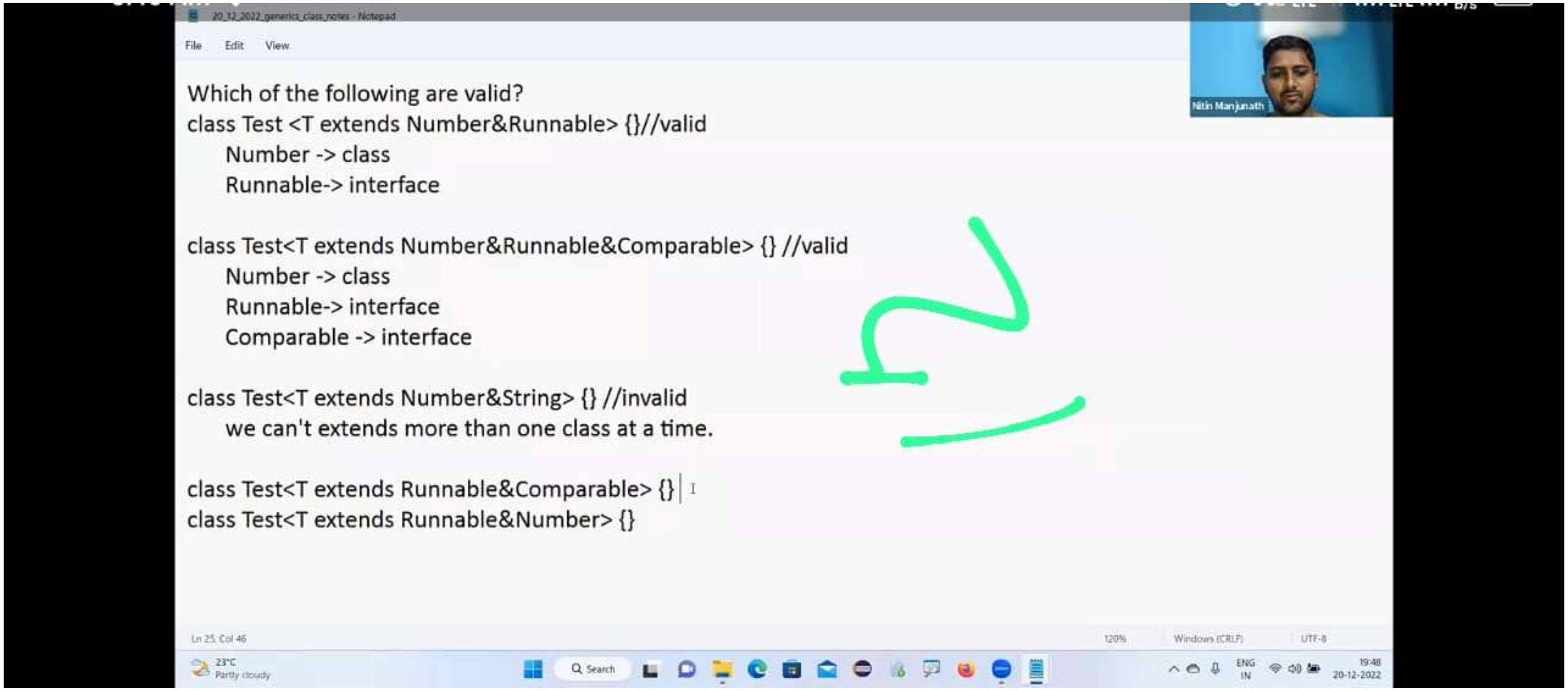


```
                |=> TypeParameter
class Demo<T>{

}
class Demo<X,Y>{

}

class HashMap<K,V>{

}
HashMap<Integer,String> hm = new HashMap<Integer,String>();
hm.put(
```



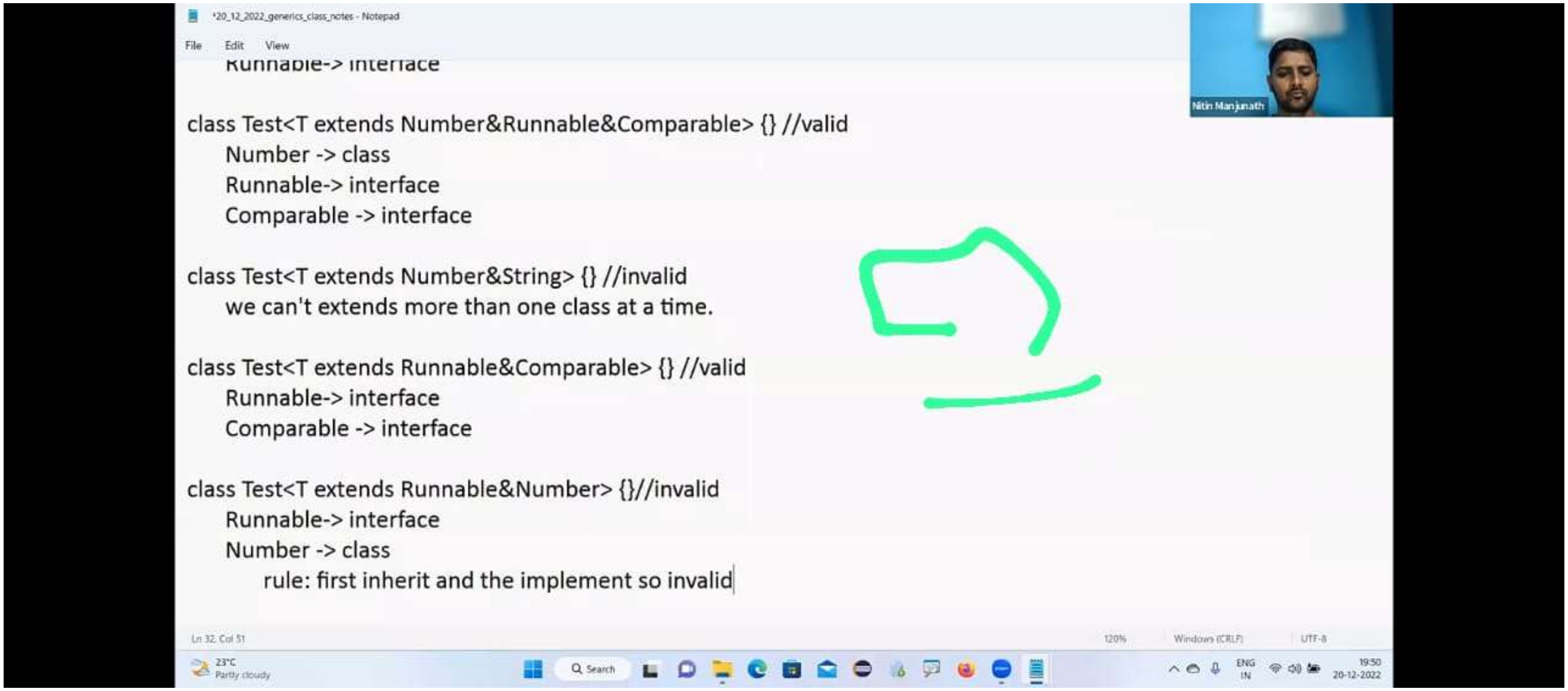
keyPoints abount BoundedTypes

```
=> As the type parameter we can use any valid java identifier but it convention to use T always.
        eg: class Test<T>{}
            class Test<iNeuron>{}

=> We can pass any no of type parameters need not be one.
        eg: class HashMap<K,V>{}
        HashMap<Integer,String> h=new HashMap<Integer,String>();
```

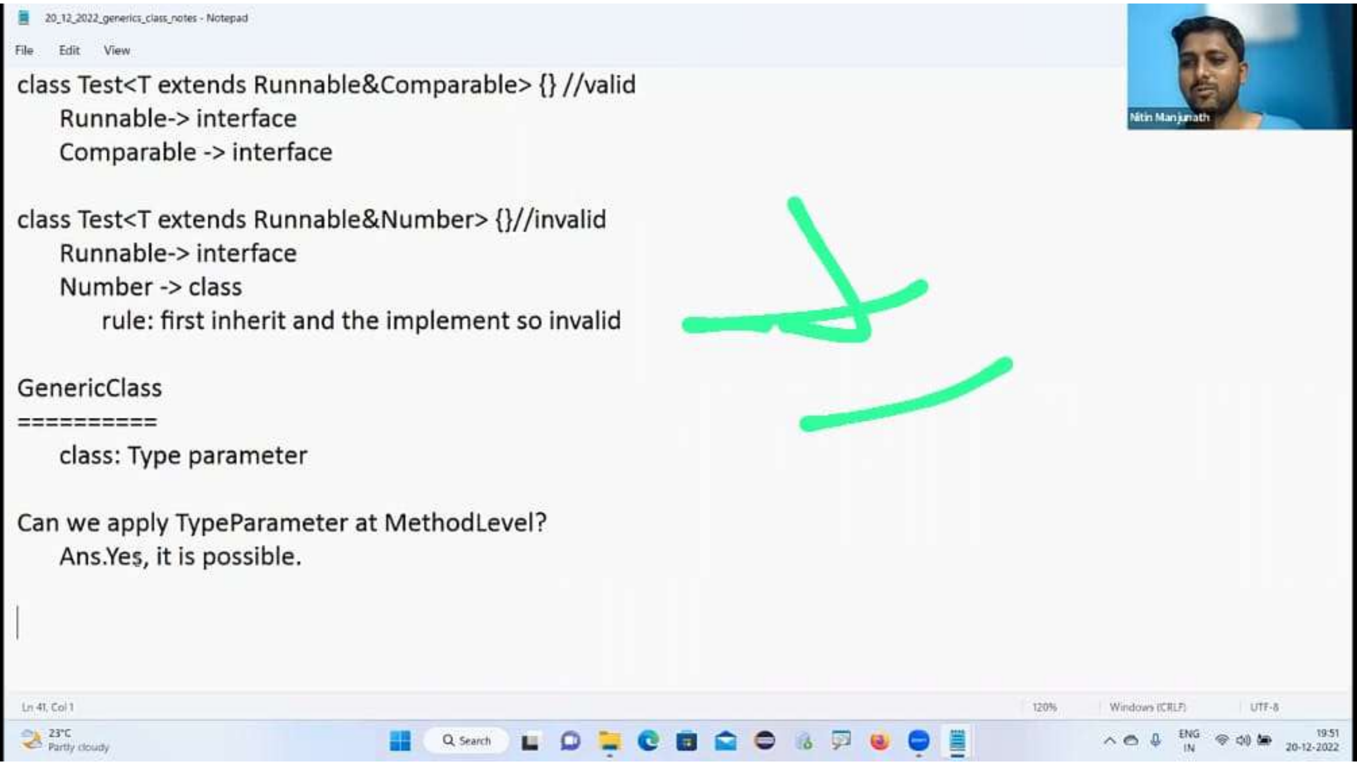


```
Which of the following are valid?
class Test <T extends Number&Runnable> {}//valid
    Number -> class
    Runnable-> interface

class Test<T extends Number&Runnable&Comparable> {} //valid
    Number -> class
    Runnable-> interface
    Comparable -> interface

class Test<T extends Number&String> {} //invalid
    we can't extends more than one class at a time.

class Test<T extends Runnable&Comparable> {}
class Test<T extends Runnable&Number> {}
```

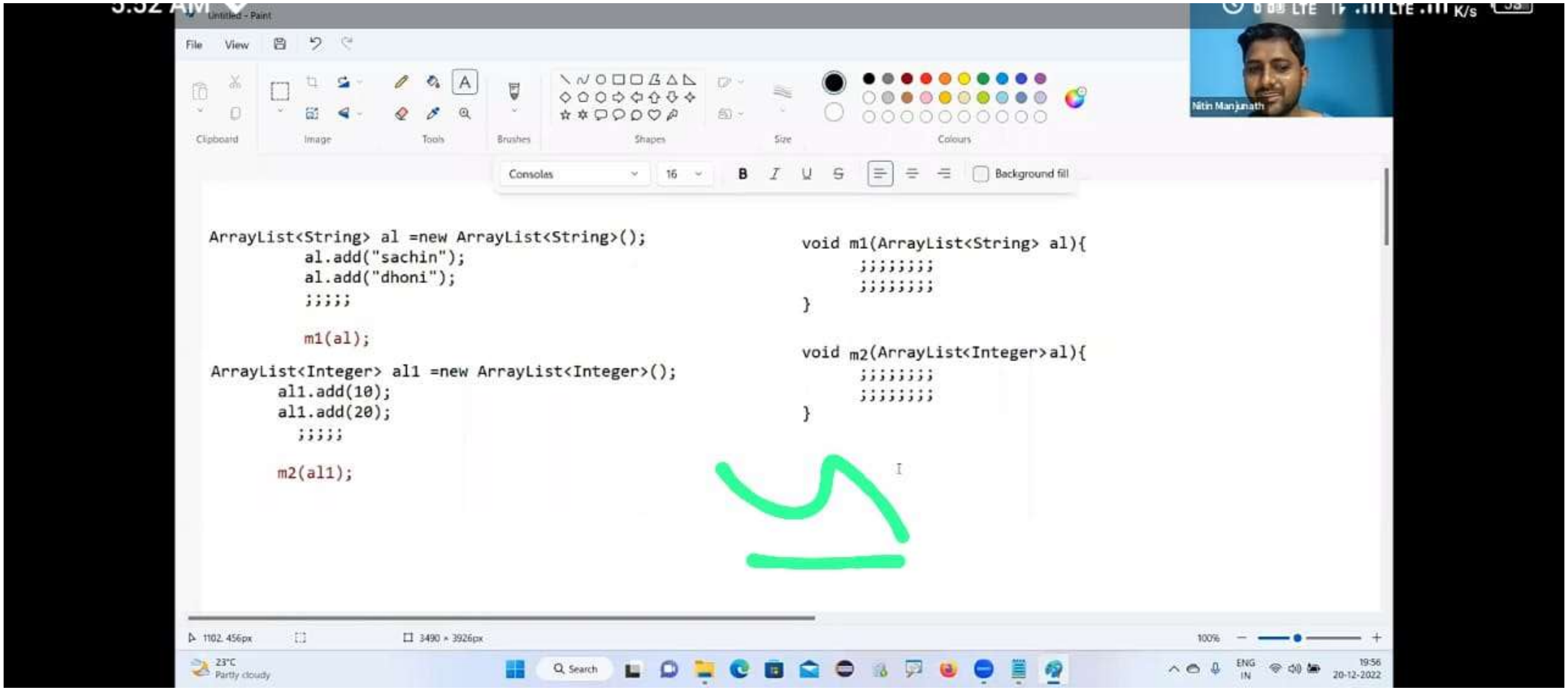
12



```
    Runnable-> interface

class Test<T extends Number&Runnable&Comparable> {} //valid
    Number -> class
    Runnable-> interface
    Comparable -> interface

class Test<T extends Number&String> {} //invalid
    we can't extends more than one class at a time.

class Test<T extends Runnable&Comparable> {} //valid
    Runnable-> interface
    Comparable -> interface

class Test<T extends Runnable&Number> {}//invalid
    Runnable-> interface
    Number -> class
        rule: first inherit and the implement so invalid
```

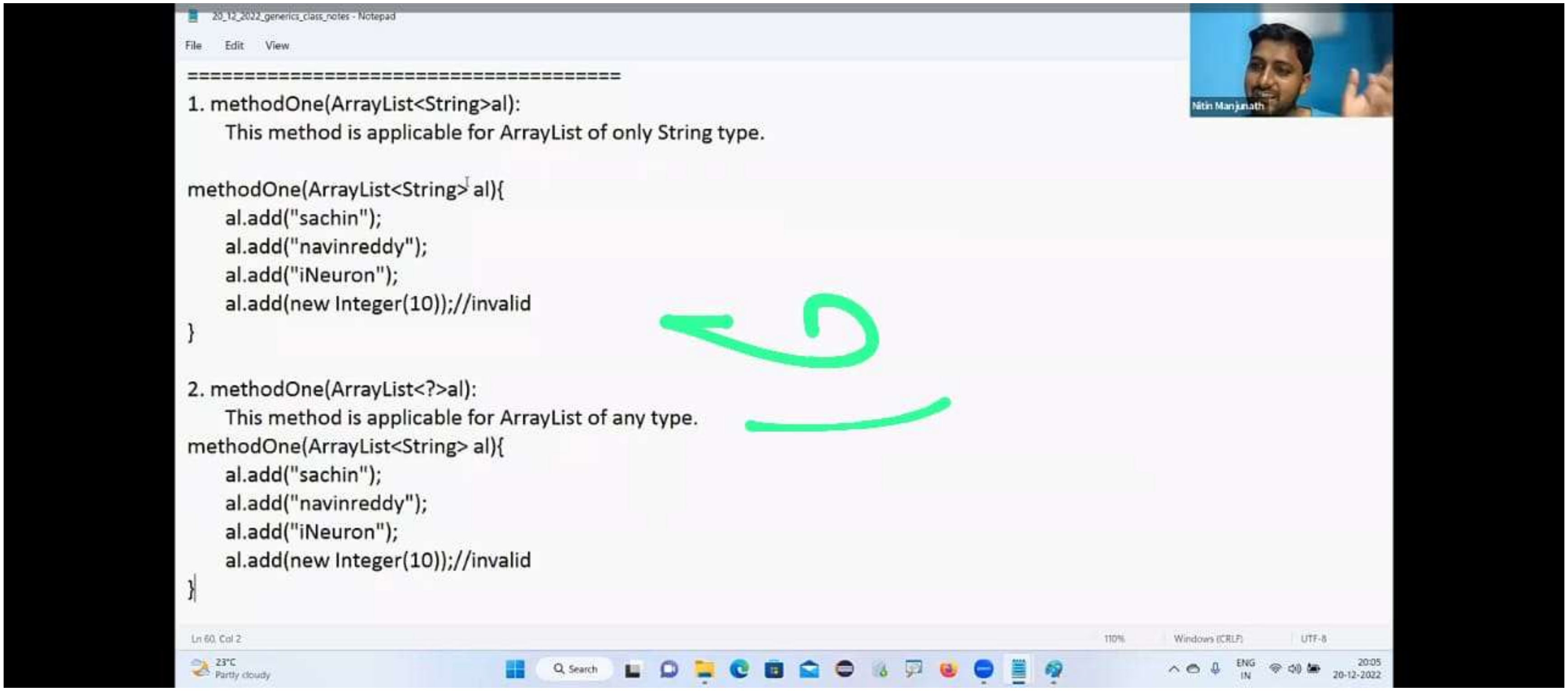


```
class Test<T extends Runnable&Comparable> {} //valid
    Runnable-> interface
    Comparable -> interface

class Test<T extends Runnable&Number> {}//invalid
    Runnable-> interface
    Number -> class
        rule: first inherit and the implement so invalid

GenericClass
==========
    class: Type parameter

Can we apply TypeParameter at MethodLevel?
    Ans.Yes, it is possible.
```

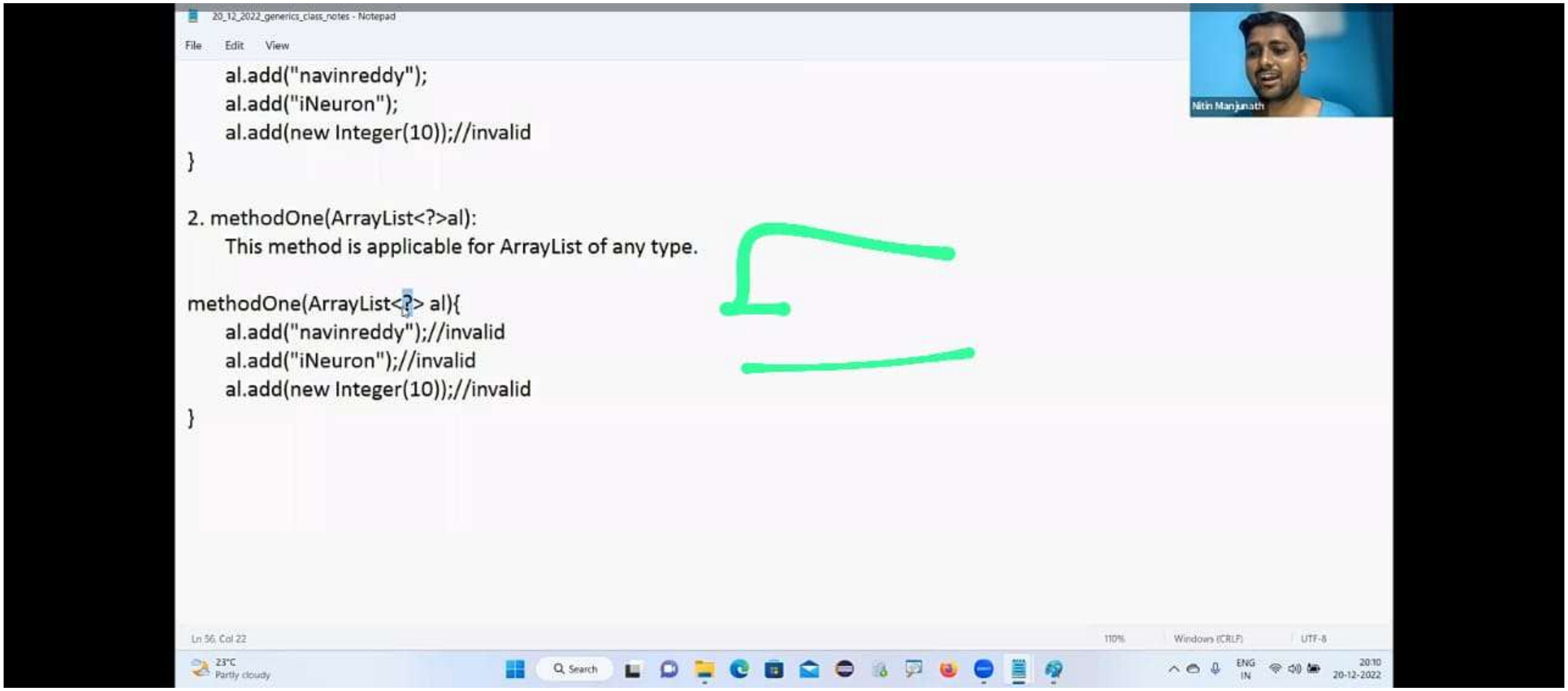


```
ArrayList<String> al =new ArrayList<String>();
          al.add("sachin");
          al.add("dhoni");
          ;;;;;

          m1(al);

ArrayList<Integer> al1 =new ArrayList<Integer>();
       al1.add(10);
       al1.add(20);
         ;;;;;

       m2(al1);
```

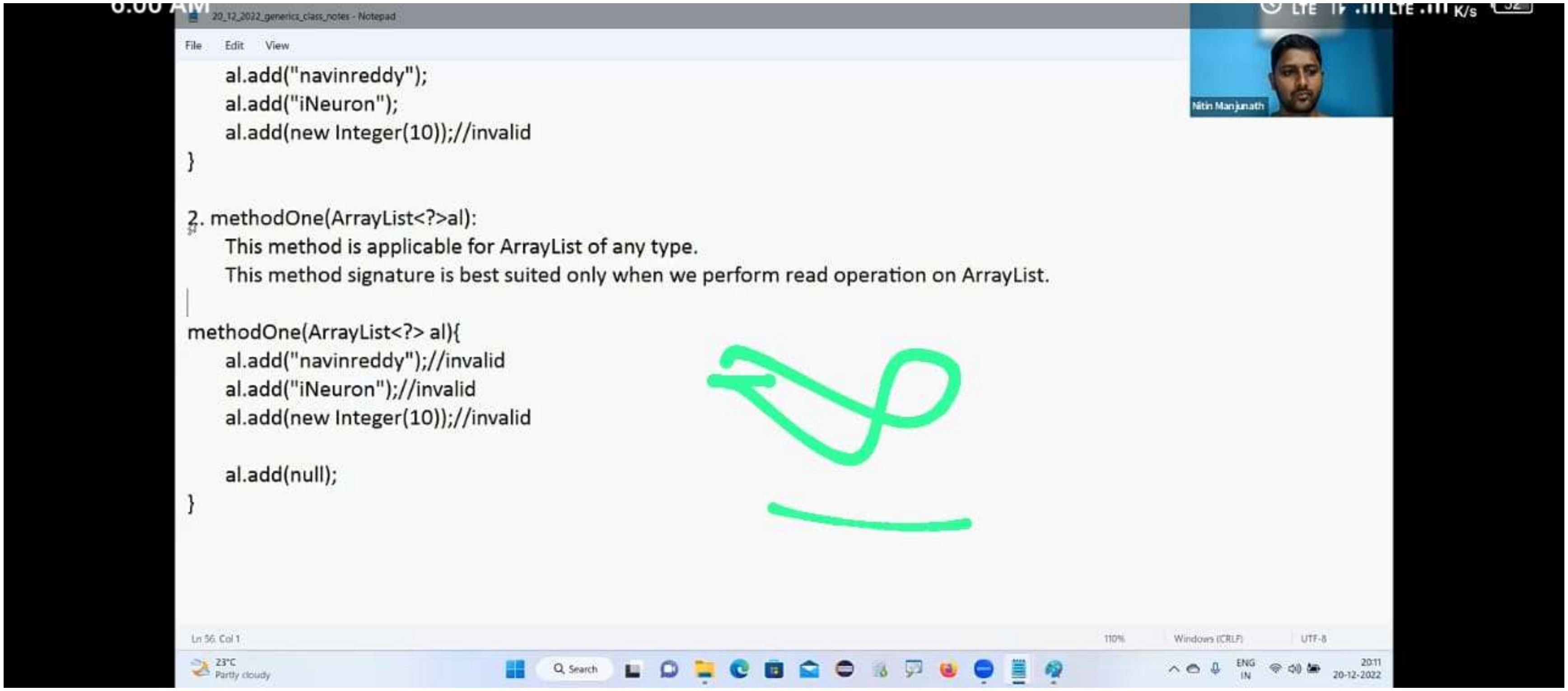
```
void m1(ArrayList<String> al){
      ;;;;;;;;
      ;;;;;;;;
}

void m2(ArrayList<Integer>al){
      ;;;;;;;;
      ;;;;;;;;
}
```



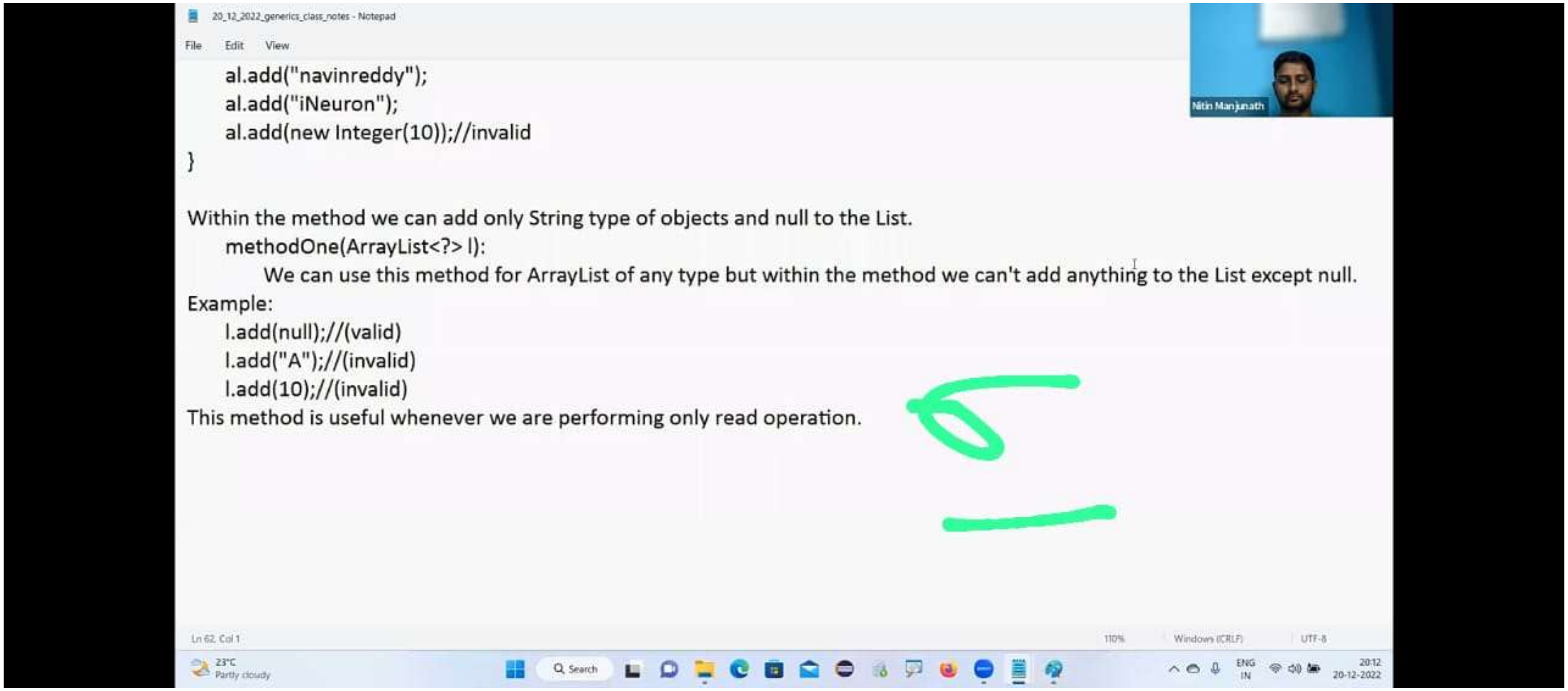
```
======================================
1. methodOne(ArrayList<String>al):
    This method is applicable for ArrayList of only String type.

methodOne(ArrayList<String> al){
    al.add("sachin");
    al.add("navinreddy");
    al.add("iNeuron");
    al.add(new Integer(10));//invalid
}

2. methodOne(ArrayList<?>al):
    This method is applicable for ArrayList of any type.
methodOne(ArrayList<String> al){
    al.add("sachin");
    al.add("navinreddy");
    al.add("iNeuron");
    al.add(new Integer(10));//invalid
}
```



```
    al.add("navinreddy");
    al.add("iNeuron");
    al.add(new Integer(10));//invalid
}

2. methodOne(ArrayList<?>al):
    This method is applicable for ArrayList of any type.

methodOne(ArrayList<?> al){
    al.add("navinreddy");//invalid
    al.add("iNeuron");//invalid
    al.add(new Integer(10));//invalid
}
```



```
    al.add("navinreddy");
    al.add("iNeuron");
    al.add(new Integer(10));//invalid
}

2. methodOne(ArrayList<?>al):
    This method is applicable for ArrayList of any type.
    This method signature is best suited only when we perform read operation on ArrayList.

methodOne(ArrayList<?> al){
    al.add("navinreddy");//invalid
    al.add("iNeuron");//invalid
    al.add(new Integer(10));//invalid

    al.add(null);
}
```



```
    al.add("navinreddy");
    al.add("iNeuron");
    al.add(new Integer(10));//invalid
}
```

Within the method we can add only String type of objects and null to the List.

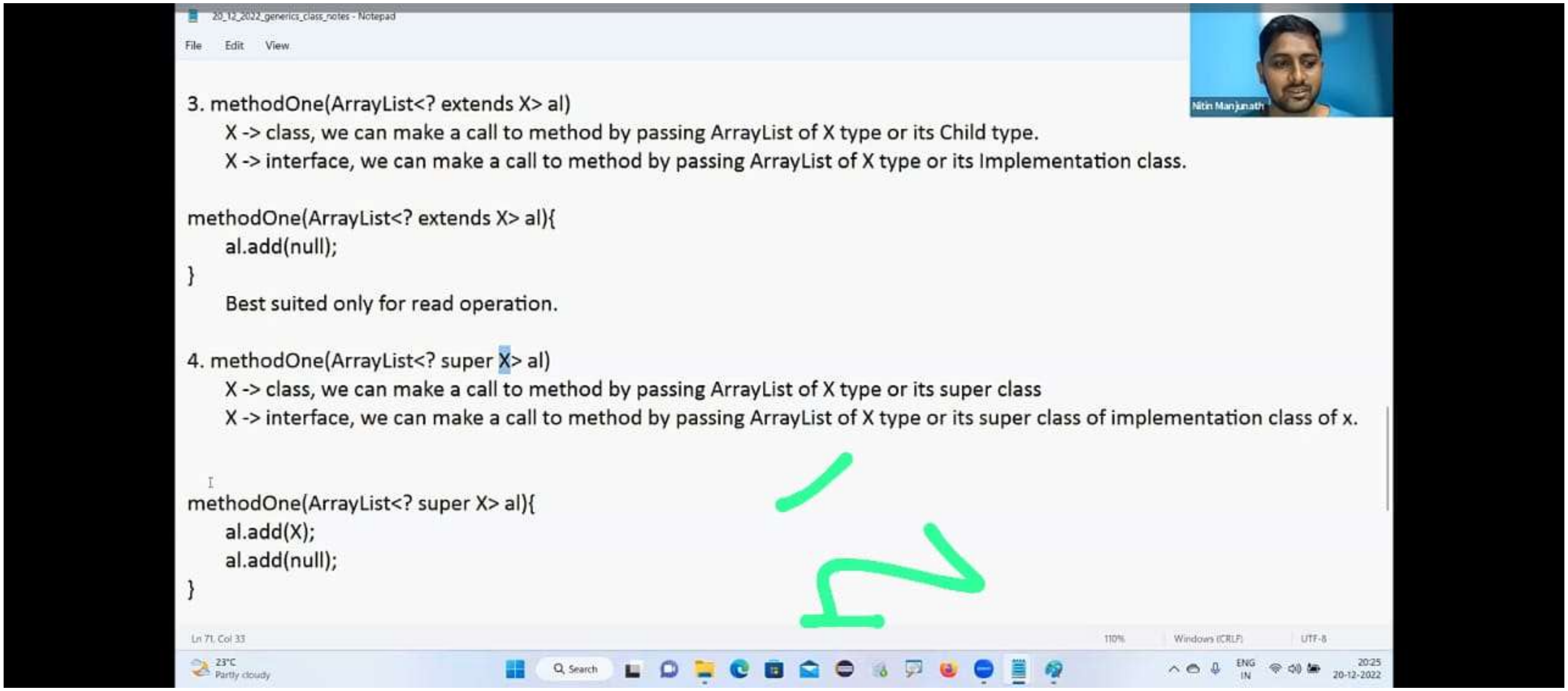
    methodOne(ArrayList<?> l):
        We can use this method for ArrayList of any type but within the method we can't add anything to the List except null.

Example:

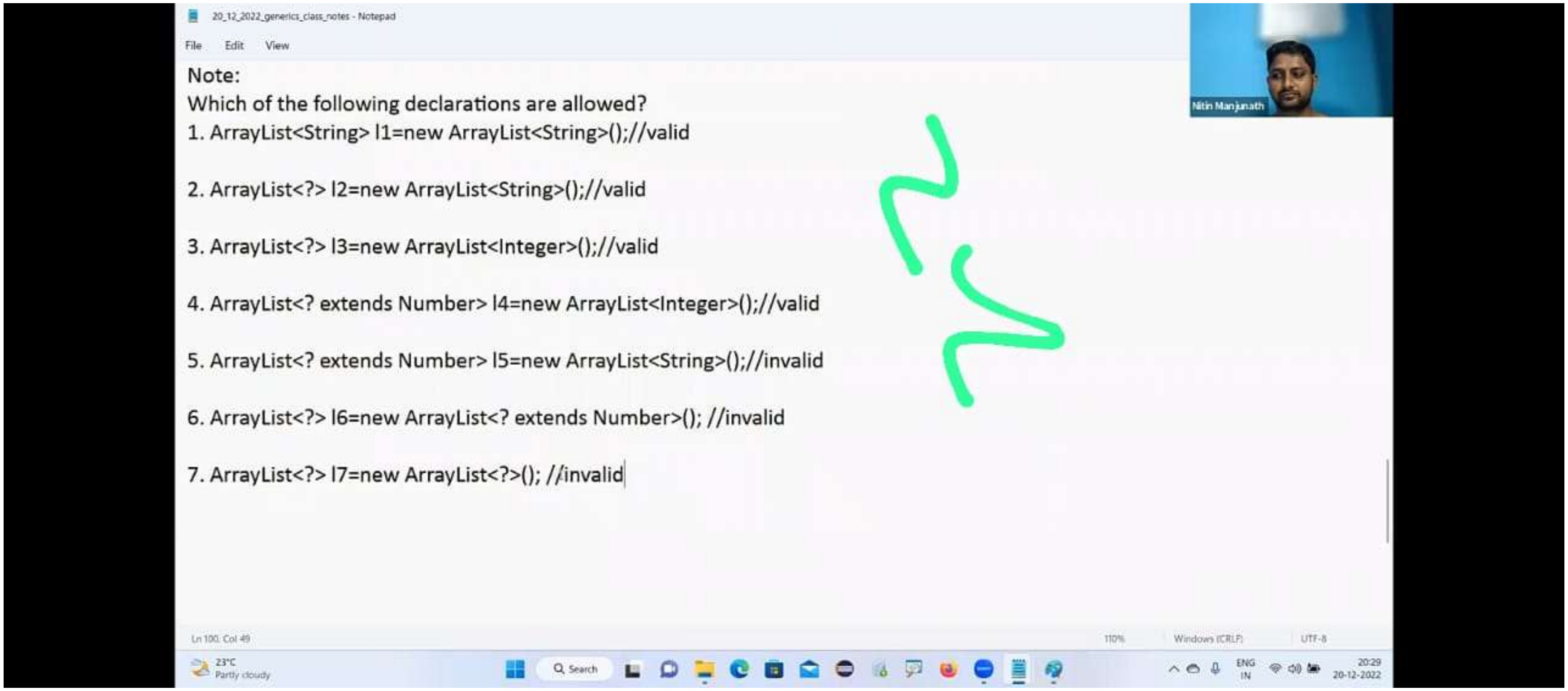
```
    l.add(null);//(valid)
    l.add("A");//(invalid)
    l.add(10);//(invalid)
```

This method is useful whenever we are performing only read operation.

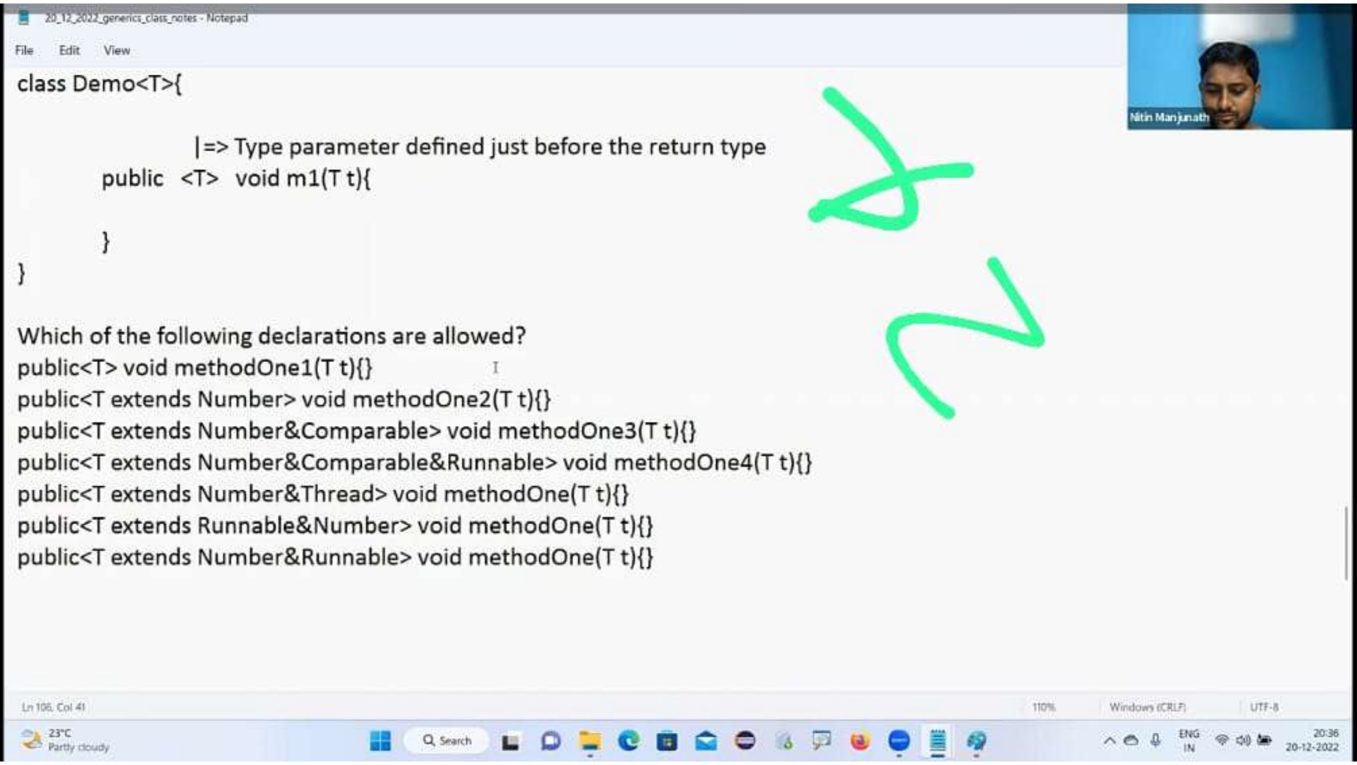
19



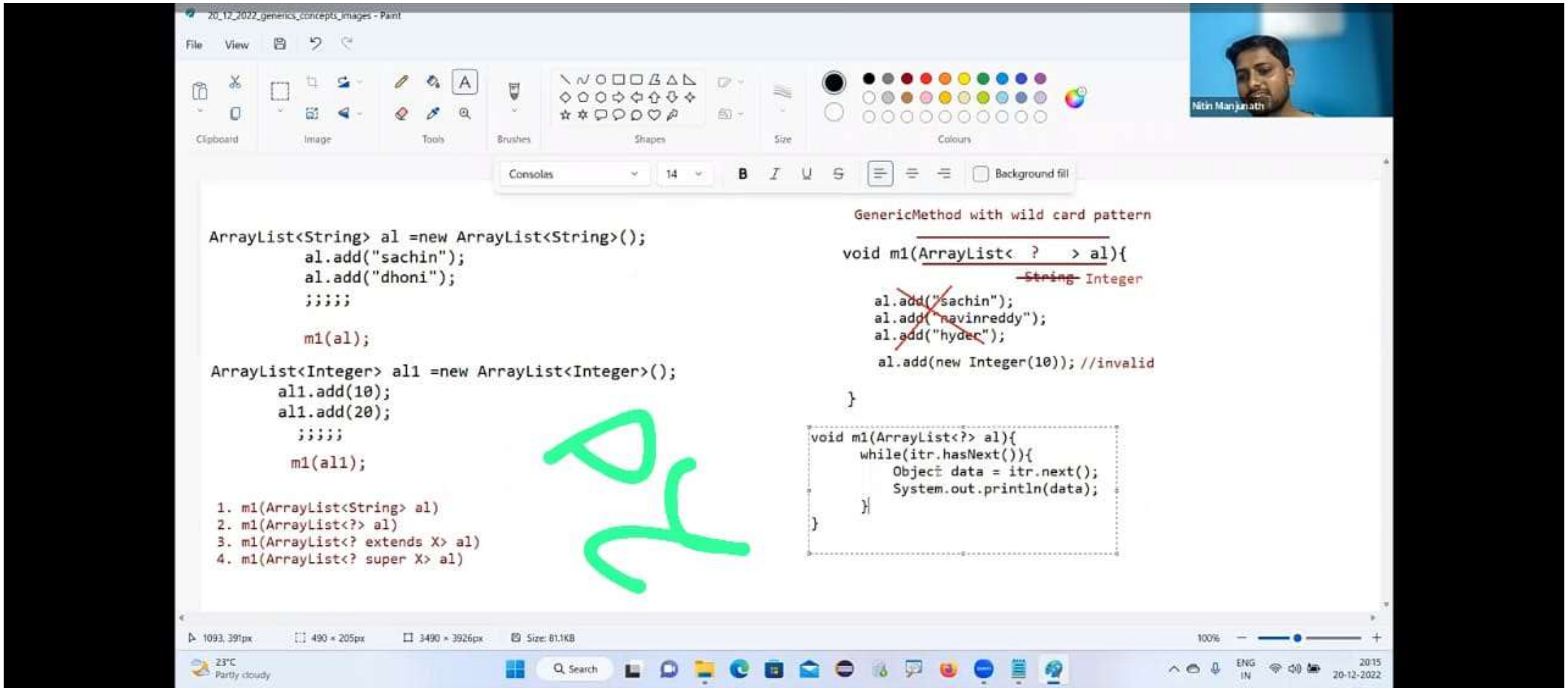
```
3. methodOne(ArrayList<? extends X> al)
    X -> class, we can make a call to method by passing ArrayList of X type or its Child type.
    X -> interface, we can make a call to method by passing ArrayList of X type or its Implementation class.

methodOne(ArrayList<? extends X> al){
    al.add(null);
}
    Best suited only for read operation.

4. methodOne(ArrayList<? super X> al)
    X -> class, we can make a call to method by passing ArrayList of X type or its super class
    X -> interface, we can make a call to method by passing ArrayList of X type or its super class of implementation class of x.


methodOne(ArrayList<? super X> al){
    al.add(X);
    al.add(null);
}
```



```
Note:
Which of the following declarations are allowed?
1. ArrayList<String> l1=new ArrayList<String>();//valid

2. ArrayList<?> l2=new ArrayList<String>();//valid

3. ArrayList<?> l3=new ArrayList<Integer>();//valid

4. ArrayList<? extends Number> l4=new ArrayList<Integer>();//valid

5. ArrayList<? extends Number> l5=new ArrayList<String>();//invalid

6. ArrayList<?> l6=new ArrayList<? extends Number>(); //invalid

7. ArrayList<?> l7=new ArrayList<?>(); //invalid
```



```
class Demo<T>{

                 |=> Type parameter defined just before the return type
        public  <T>  void m1(T t){

        }
}

Which of the following declarations are allowed?
public<T> void methodOne1(T t){}
public<T extends Number> void methodOne2(T t){}
public<T extends Number&Comparable> void methodOne3(T t){}
public<T extends Number&Comparable&Runnable> void methodOne4(T t){}
public<T extends Number&Thread> void methodOne(T t){}
public<T extends Runnable&Number> void methodOne(T t){}
public<T extends Number&Runnable> void methodOne(T t){}
```

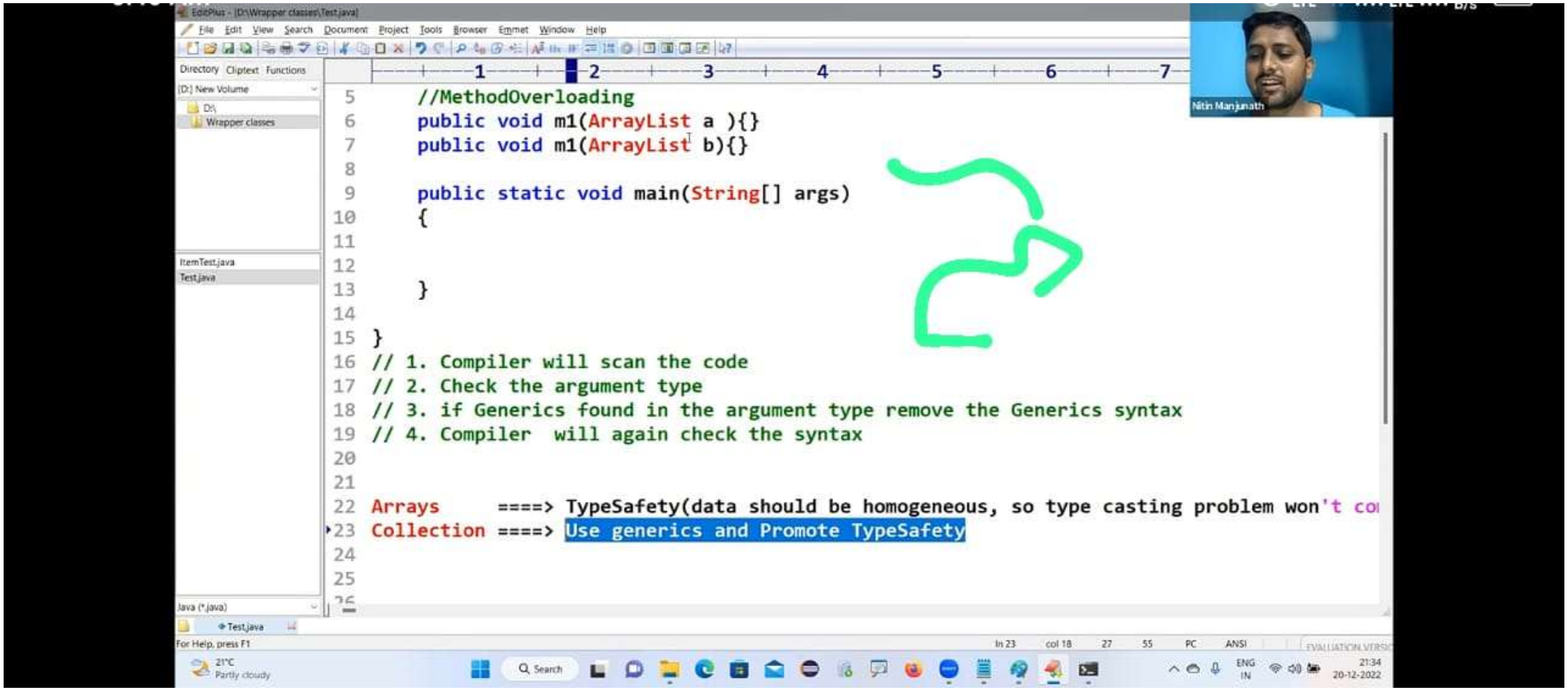


```
ArrayList<String> al =new ArrayList<String>();
          al.add("sachin");
          al.add("dhoni");
          ;;;;;

          m1(al);

ArrayList<Integer> al1 =new ArrayList<Integer>();
       al1.add(10);
       al1.add(20);
         ;;;;;
        m1(al1);

 1. m1(ArrayList<String> al)
 2. m1(ArrayList<?> al)
 3. m1(ArrayList<? extends X> al)
 4. m1(ArrayList<? super X> al)
```

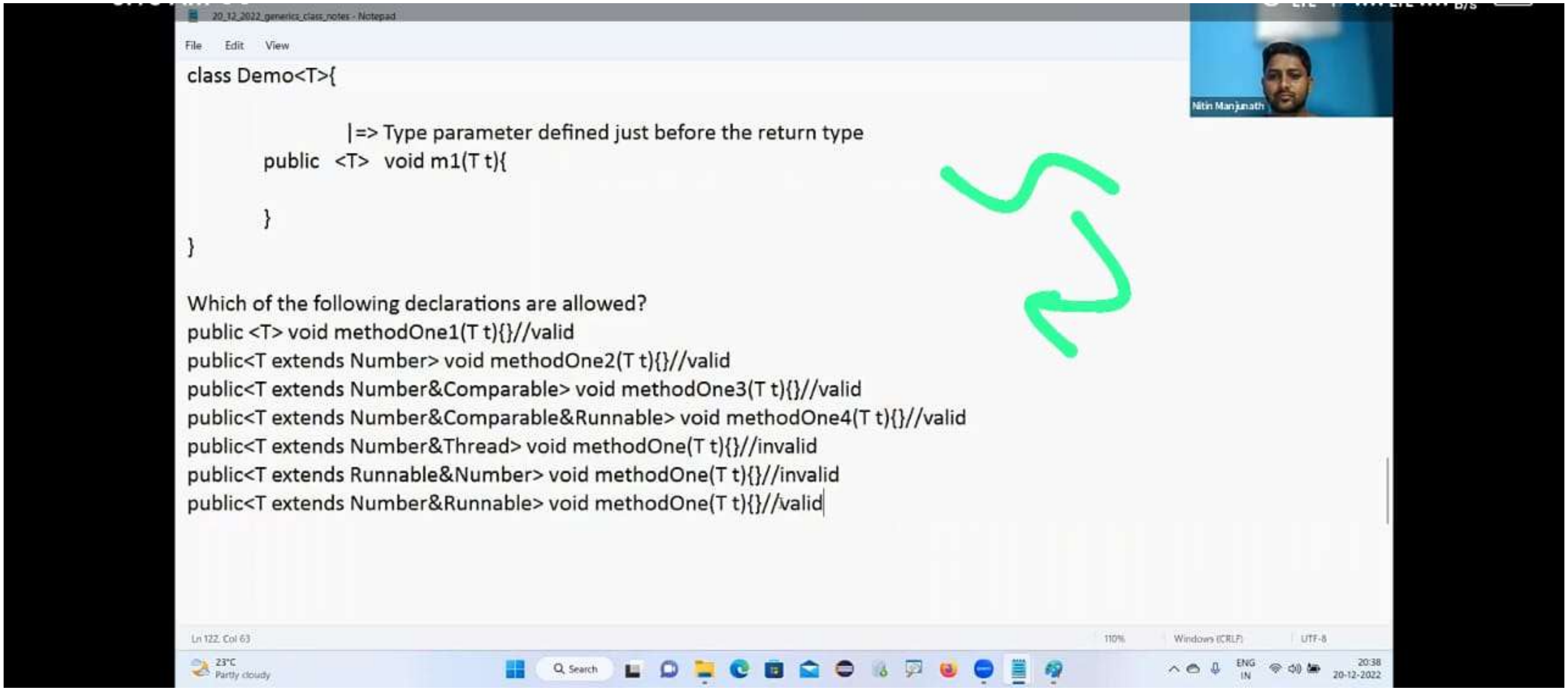
GenericMethod with wild card pattern

```
void m1(ArrayList<  ?    > al){
                  ~~String~~ Integer
    ~~al.add("sachin");~~
    ~~al.add("navinreddy");~~
    ~~al.add("hyder");~~
     al.add(new Integer(10)); //invalid

}
```

```
void m1(ArrayList<?> al){
      while(itr.hasNext()){
          Object data = itr.next();
          System.out.println(data);
      }
}
```

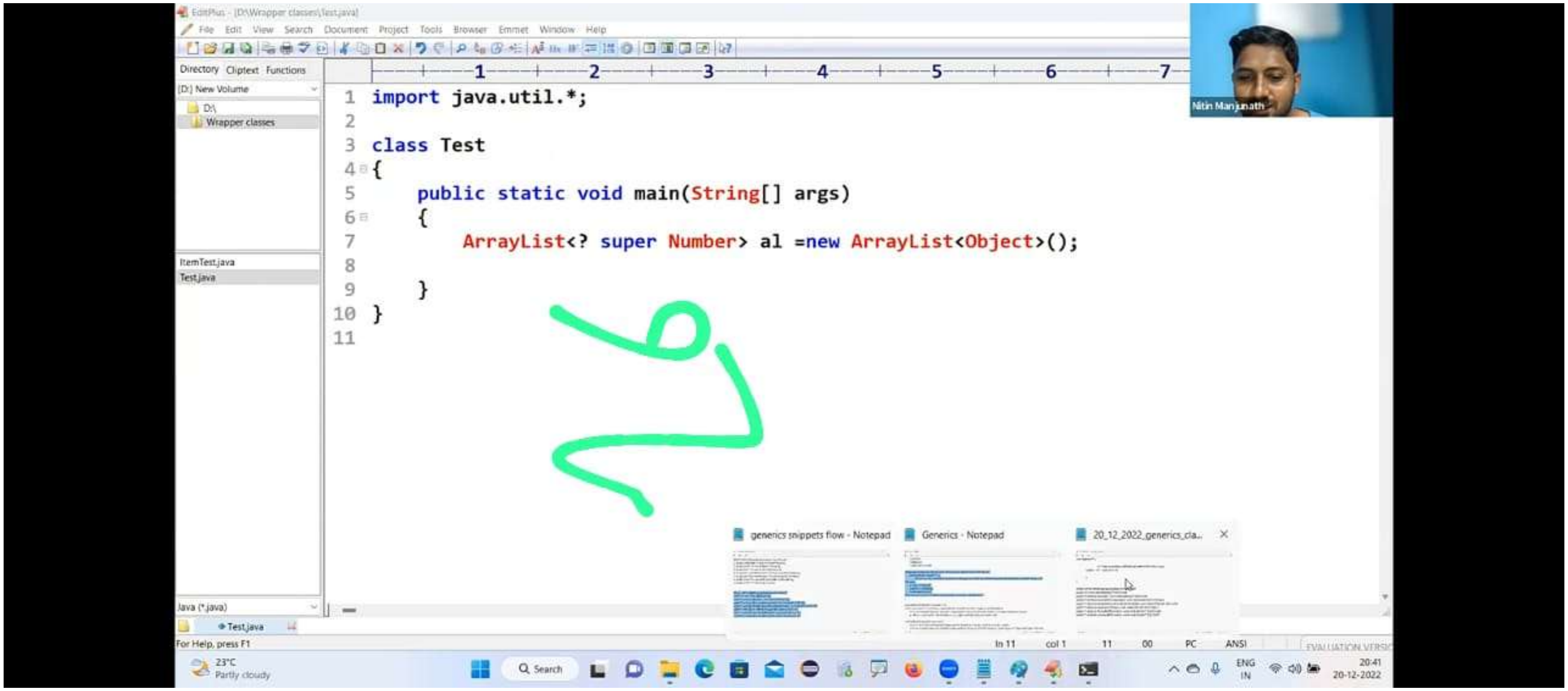

EditPlus - [D:\Wrapper classes\Test.java]
File Edit View Search Document Project Tools Browser Emmet Window Help
Directory Cliptext Functions
[D:] New Volume
D:\
Wrapper classes
ItemTest.java
Test.java
Java (*.java)
Nitin Manjunath
For Help, press F1
In 23 col 18 27 55 PC ANSI

```
    //MethodOverloading
    public void m1(ArrayList a ){}
    public void m1(ArrayList b){}

    public static void main(String[] args)
    {


    }

}
// 1. Compiler will scan the code
// 2. Check the argument type
// 3. if Generics found in the argument type remove the Generics syntax
// 4. Compiler  will again check the syntax


Arrays     ====> TypeSafety(data should be homogeneous, so type casting problem won't co
Collection ====> Use generics and Promote TypeSafety
```



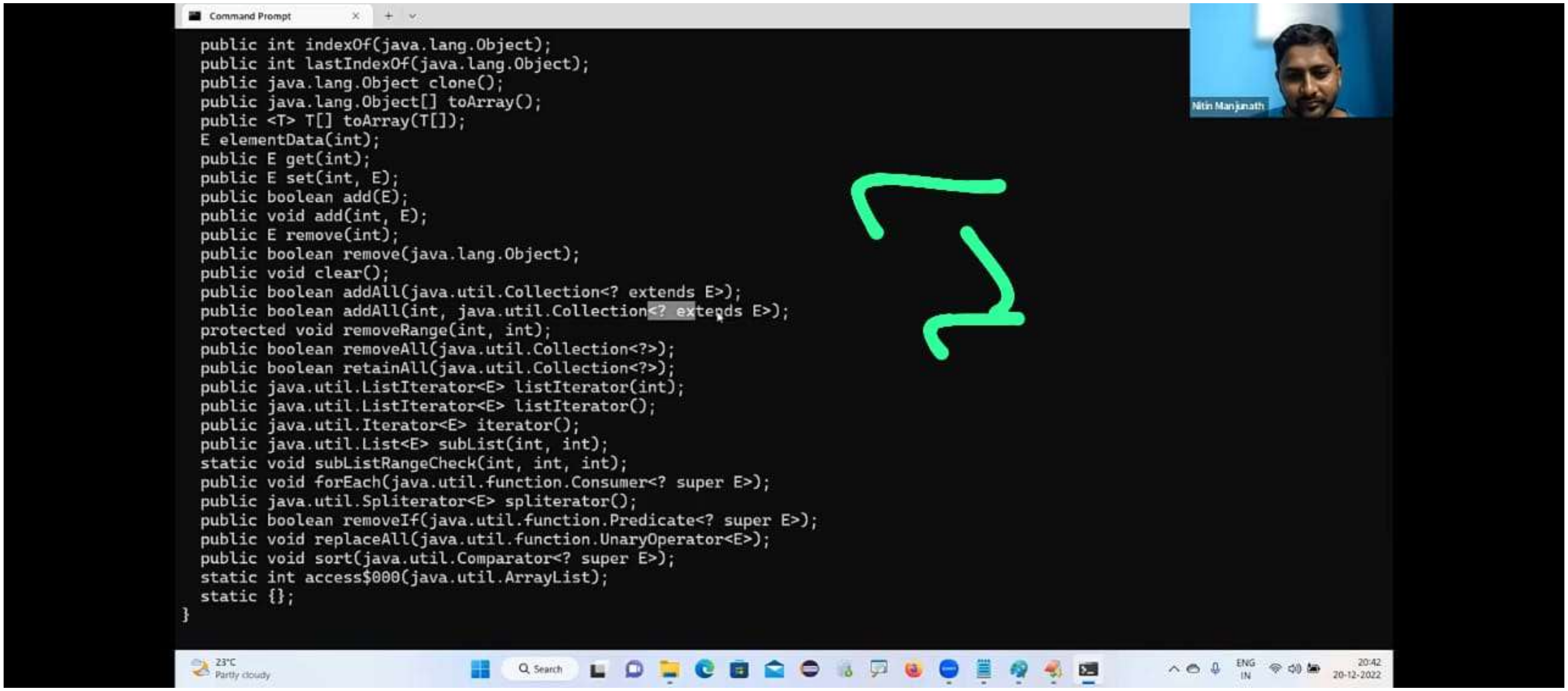
```
class Demo<T>{

                    |=> Type parameter defined just before the return type
          public   <T>   void m1(T t){

          }
}

Which of the following declarations are allowed?
public <T> void methodOne1(T t){}//valid
public<T extends Number> void methodOne2(T t){}//valid
public<T extends Number&Comparable> void methodOne3(T t){}//valid
public<T extends Number&Comparable&Runnable> void methodOne4(T t){}//valid
public<T extends Number&Thread> void methodOne(T t){}//invalid
public<T extends Runnable&Number> void methodOne(T t){}//invalid
public<T extends Number&Runnable> void methodOne(T t){}//valid
```

25

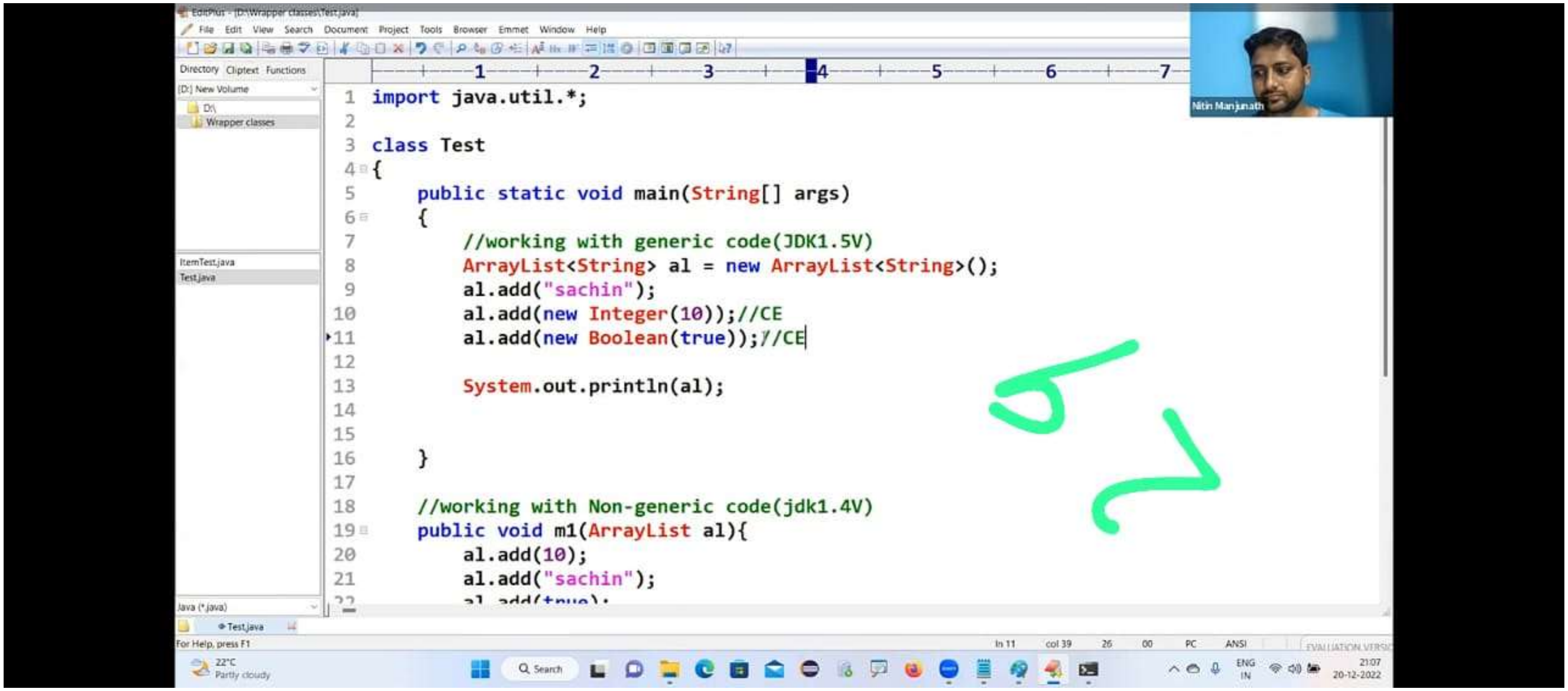

EditPlus - [D:\Wrapper classes\Test.java]
File Edit View Search Document Project Tools Browser Emmet Window Help
Directory Cliptext Functions
[D:] New Volume
D:\
Wrapper classes
ItemTest.java
Test.java
Java (*.java)
Test.java
For Help, press F1
generics snippets flow - Notepad
Generics - Notepad
20_12_2022_generics_cla...
ln 11 col 1 11 00 PC ANSI
Nitin Manjunath
23°C Partly cloudy
Search
ENG IN
20:41 20-12-2022

```java
import java.util.*;

class Test
{
    public static void main(String[] args)
    {
        ArrayList<? super Number> al =new ArrayList<Object>();

    }
}
```

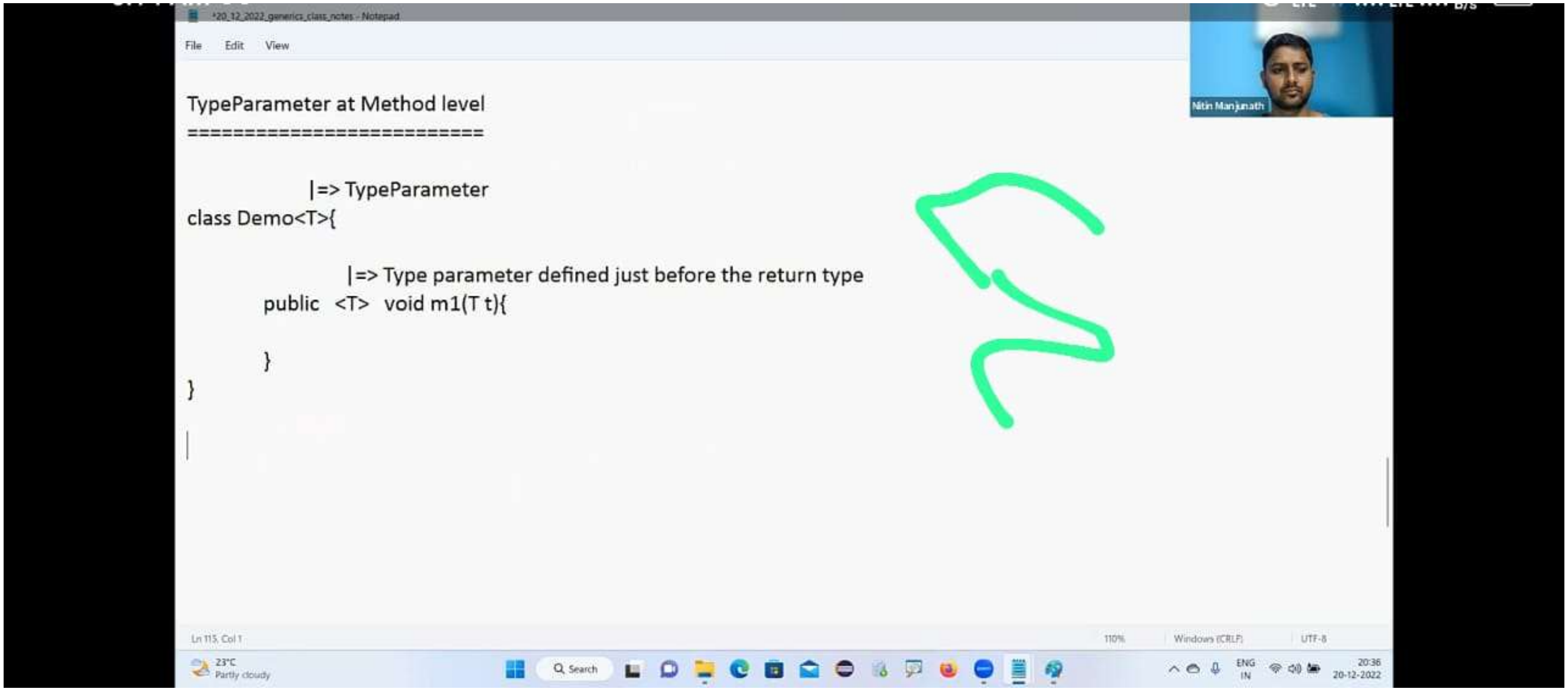
26



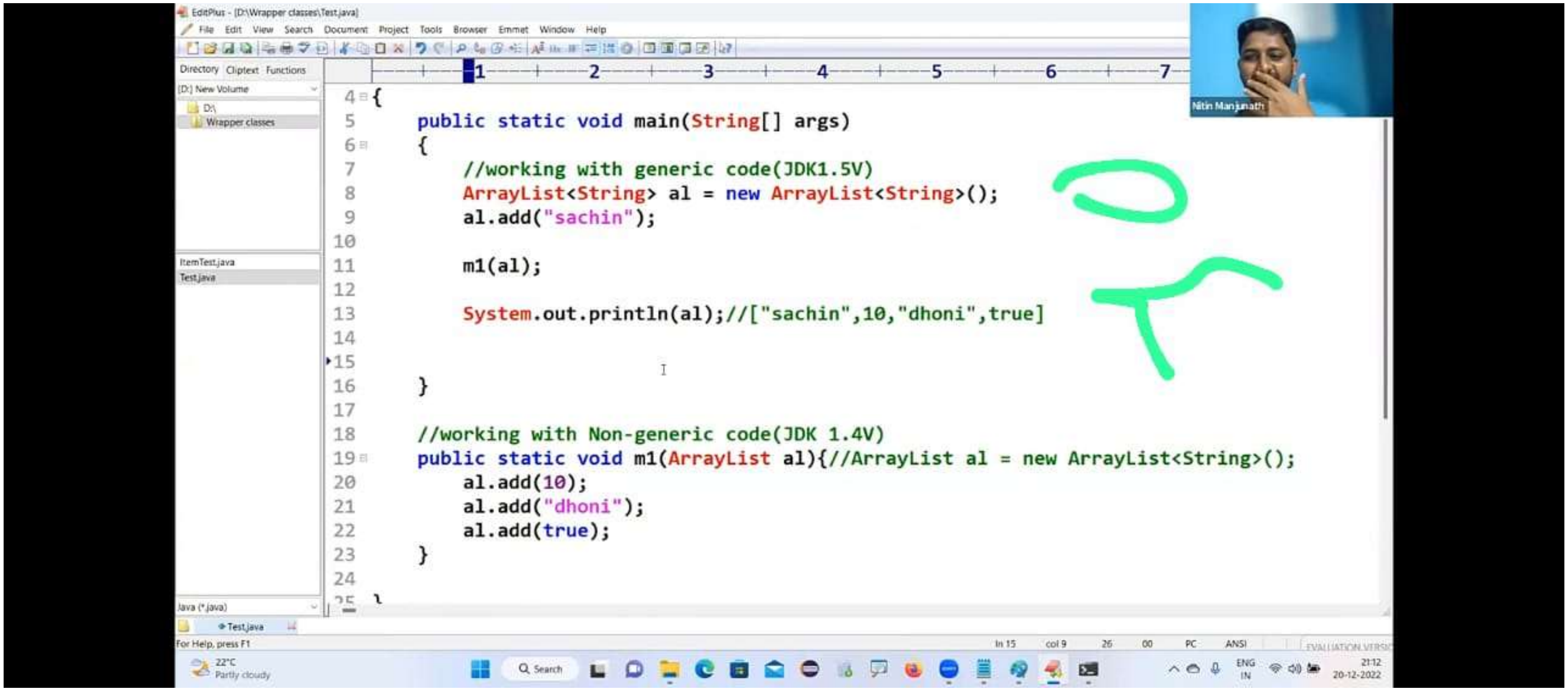
```
  public int indexOf(java.lang.Object);
  public int lastIndexOf(java.lang.Object);
  public java.lang.Object clone();
  public java.lang.Object[] toArray();
  public <T> T[] toArray(T[]);
  E elementData(int);
  public E get(int);
  public E set(int, E);
  public boolean add(E);
  public void add(int, E);
  public E remove(int);
  public boolean remove(java.lang.Object);
  public void clear();
  public boolean addAll(java.util.Collection<? extends E>);
  public boolean addAll(int, java.util.Collection<? extends E>);
  protected void removeRange(int, int);
  public boolean removeAll(java.util.Collection<?>);
  public boolean retainAll(java.util.Collection<?>);
  public java.util.ListIterator<E> listIterator(int);
  public java.util.ListIterator<E> listIterator();
  public java.util.Iterator<E> iterator();
  public java.util.List<E> subList(int, int);
  static void subListRangeCheck(int, int, int);
  public void forEach(java.util.function.Consumer<? super E>);
  public java.util.Spliterator<E> spliterator();
  public boolean removeIf(java.util.function.Predicate<? super E>);
  public void replaceAll(java.util.function.UnaryOperator<E>);
  public void sort(java.util.Comparator<? super E>);
  static int access$000(java.util.ArrayList);
  static {};
}
```


EditPlus - [D:\Wrapper classes\Test.java]
File Edit View Search Document Project Tools Browser Emmet Window Help
Directory Cliptext Functions
[D:] New Volume
D:\
Wrapper classes
ItemTest.java
Test.java
Java (*.java)
Test.java
For Help, press F1
In 11 col 39 26 00 PC ANSI
Nitin Manjunath
22°C Partly cloudy
Search
ENG IN
21:07
20-12-2022
29

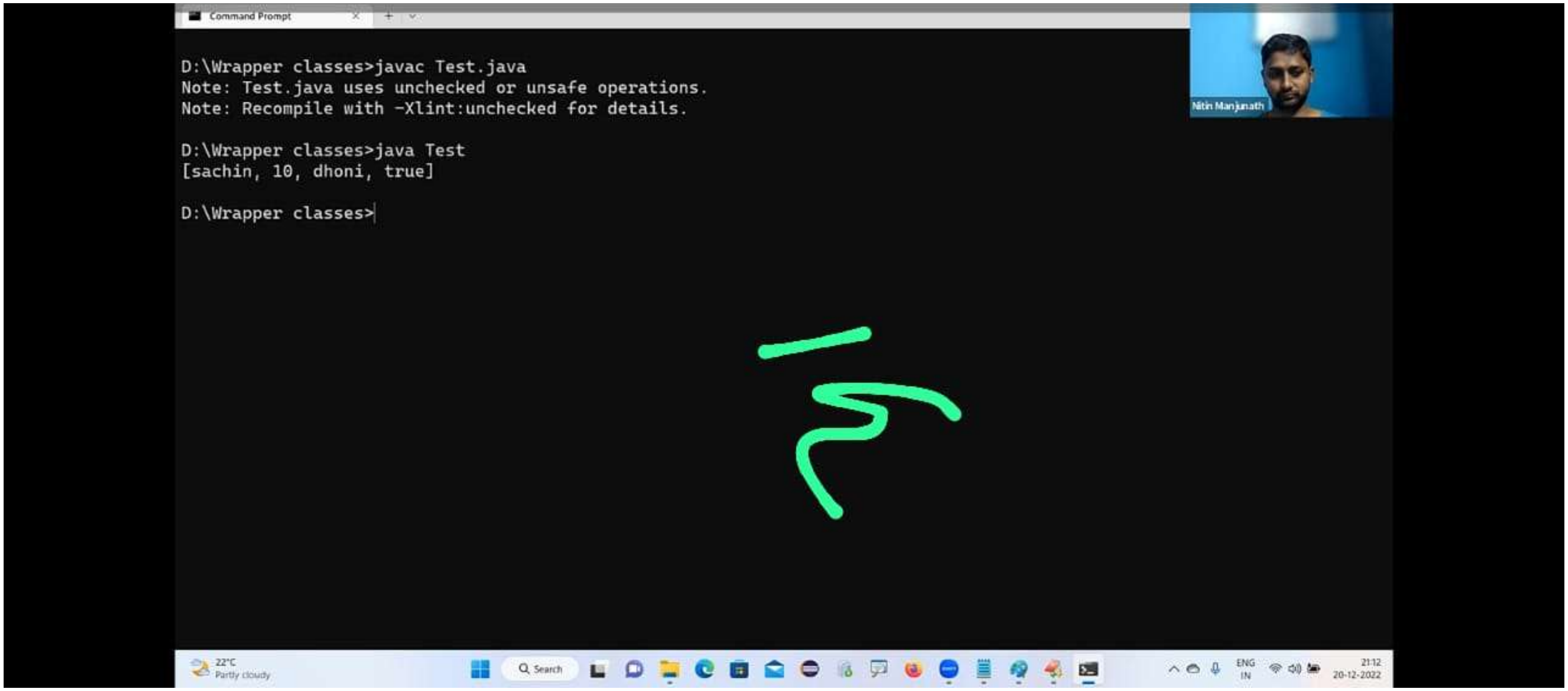
```java
import java.util.*;

class Test
{
    public static void main(String[] args)
    {
        //working with generic code(JDK1.5V)
        ArrayList<String> al = new ArrayList<String>();
        al.add("sachin");
        al.add(new Integer(10));//CE
        al.add(new Boolean(true));//CE

        System.out.println(al);


    }

    //working with Non-generic code(jdk1.4V)
    public void m1(ArrayList al){
        al.add(10);
        al.add("sachin");
        al.add(true);
```



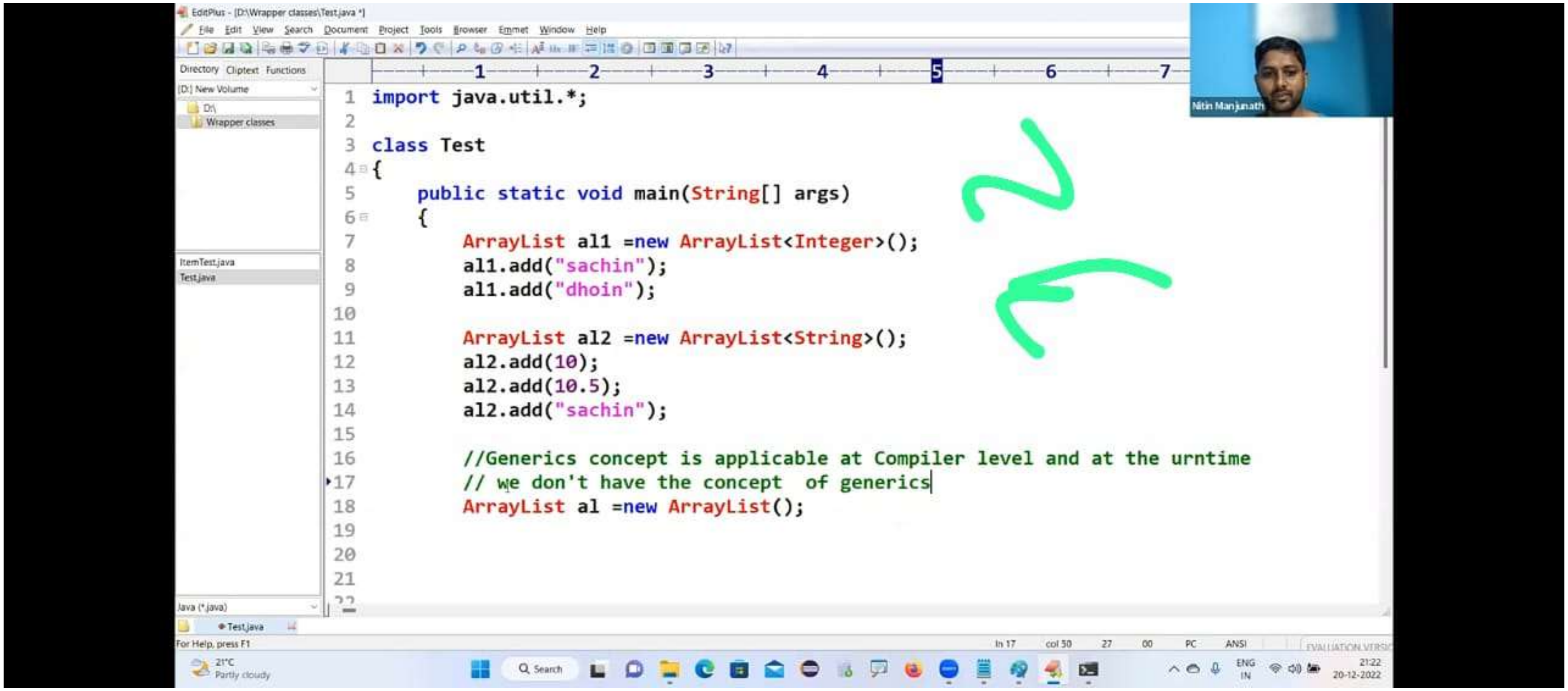
```
TypeParameter at Method level
===========================

                 |=> TypeParameter
class Demo<T>{

                 |=> Type parameter defined just before the return type
        public   <T>   void m1(T t){

        }
}
```


EditPlus - [D:\Wrapper classes\Test.java]
File Edit View Search Document Project Tools Browser Emmet Window Help
Directory Cliptext Functions
[D:] New Volume
D:\
Wrapper classes
ItemTest.java
Test.java
Java (*.java)
Test.java
For Help, press F1
In 15 col 9 26 00 PC ANSI
Nitin Manjunath
22°C Partly cloudy
Search
ENG IN
21:12 20-12-2022

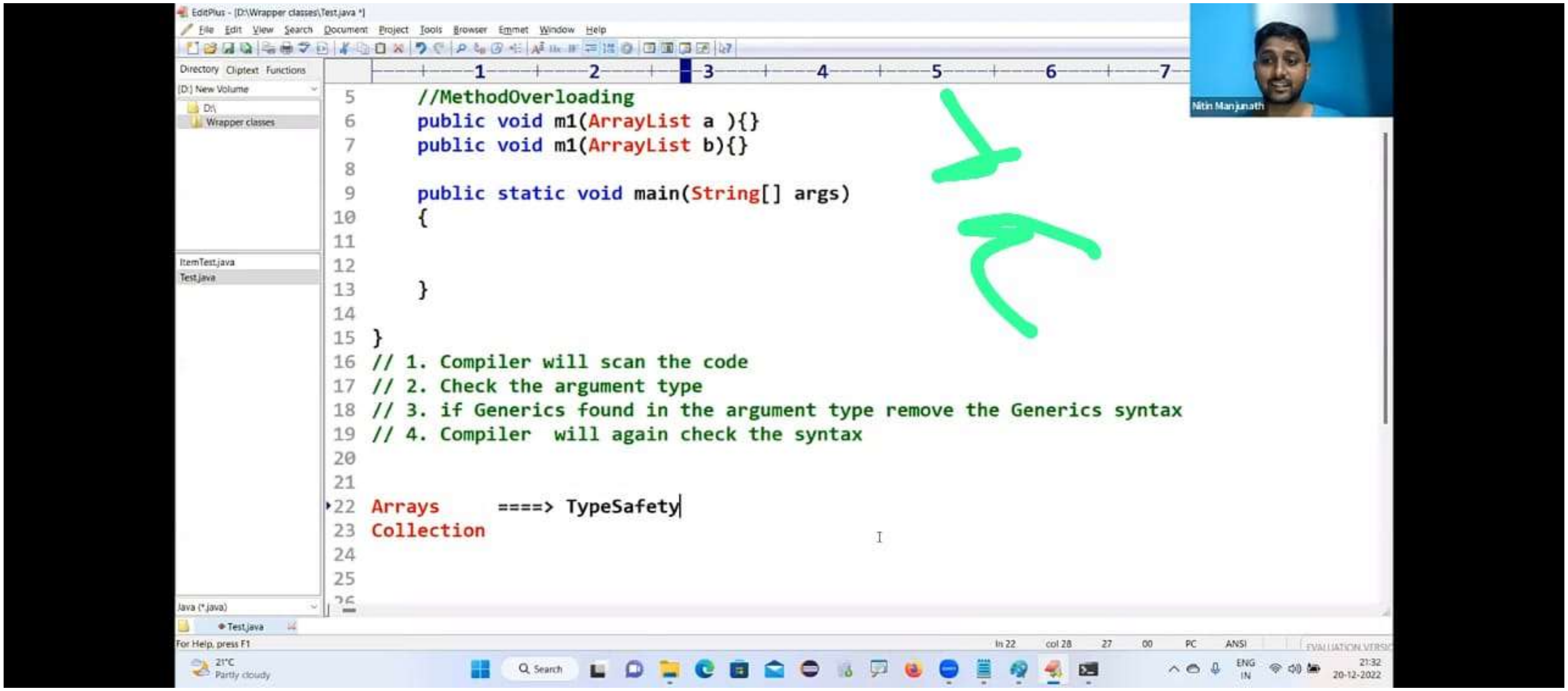
```java
{
    public static void main(String[] args)
    {
        //working with generic code(JDK1.5V)
        ArrayList<String> al = new ArrayList<String>();
        al.add("sachin");

        m1(al);

        System.out.println(al);//["sachin",10,"dhoni",true]


    }

    //working with Non-generic code(JDK 1.4V)
    public static void m1(ArrayList al){//ArrayList al = new ArrayList<String>();
        al.add(10);
        al.add("dhoni");
        al.add(true);
    }

}
```



```
D:\Wrapper classes>javac Test.java
Note: Test.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

D:\Wrapper classes>java Test
[sachin, 10, dhoni, true]

D:\Wrapper classes>
```

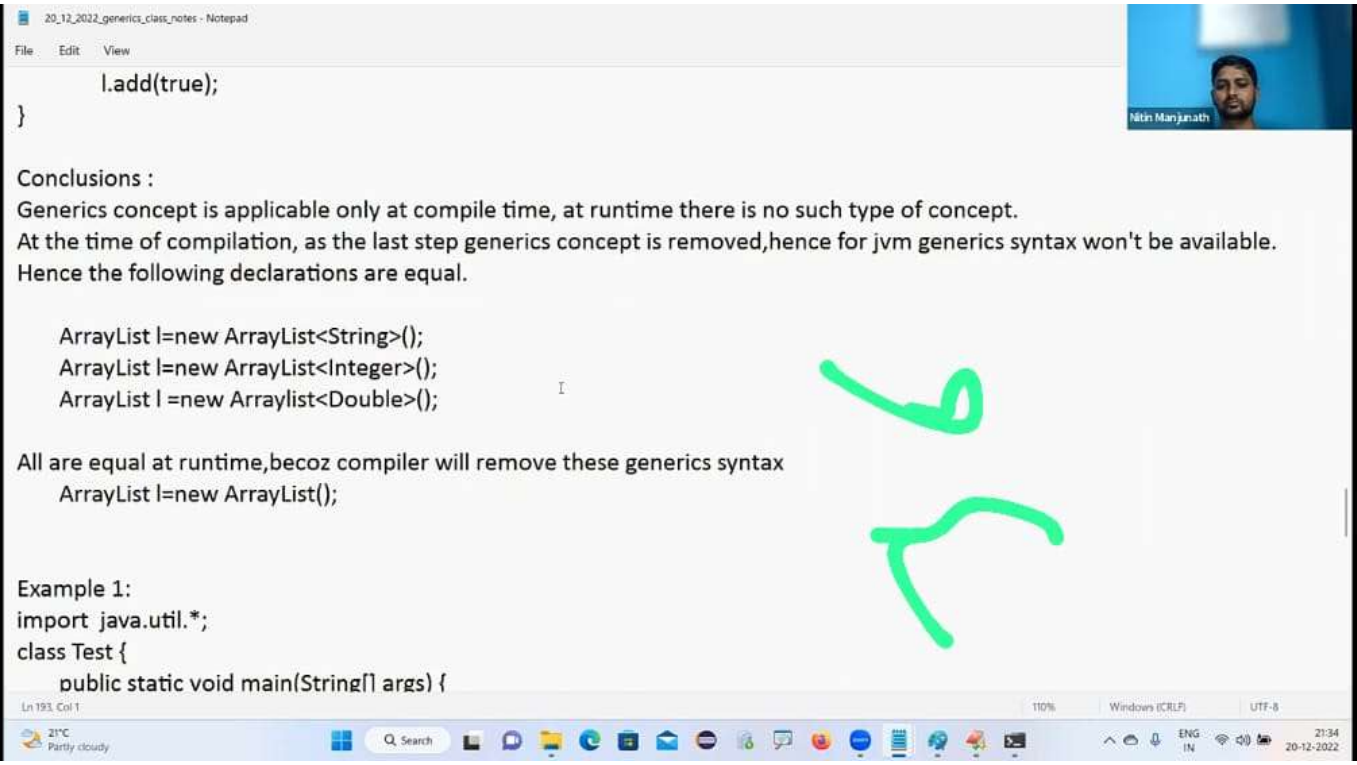

EditPlus - [D:\Wrapper classes\Test.java *]
File Edit View Search Document Project Tools Browser Emmet Window Help
Directory Cliptext Functions
[D:] New Volume
D:\
Wrapper classes
ItemTest.java
Test.java
Java (*.java)
Test.java
For Help, press F1
In 17 col 50 27 00 PC ANSI
Nitin Manjunath
21°C Partly cloudy
Search
ENG IN
21:22 20-12-2022

```java
import java.util.*;

class Test
{
    public static void main(String[] args)
    {
        ArrayList al1 =new ArrayList<Integer>();
        al1.add("sachin");
        al1.add("dhoin");

        ArrayList al2 =new ArrayList<String>();
        al2.add(10);
        al2.add(10.5);
        al2.add("sachin");

        //Generics concept is applicable at Compiler level and at the urntime
        // we don't have the concept  of generics
        ArrayList al =new ArrayList();
```



```
    //MethodOverloading
    public void m1(ArrayList a ){}
    public void m1(ArrayList b){}

    public static void main(String[] args)
    {


    }

}
// 1. Compiler will scan the code
// 2. Check the argument type
// 3. if Generics found in the argument type remove the Generics syntax
// 4. Compiler  will again check the syntax


Arrays      ====> TypeSafety
Collection
```

EditPlus - [D:\Wrapper classes\Test.java *]
File Edit View Search Document Project Tools Browser Emmet Window Help
Directory Cliptext Functions
[D:] New Volume
D:\
Wrapper classes
ItemTest.java
Test.java
Java (*.java)
Test.java
For Help, press F1
ln 22 col 28 27 00 PC ANSI
Nitin Manjunath
21°C Partly cloudy
Search
ENG IN
21:32 20-12-2022



```
        l.add(true);
}
```

Conclusions :

Generics concept is applicable only at compile time, at runtime there is no such type of concept.
At the time of compilation, as the last step generics concept is removed,hence for jvm generics syntax won't be available.
Hence the following declarations are equal.

```
    ArrayList l=new ArrayList<String>();
    ArrayList l=new ArrayList<Integer>();
    ArrayList l =new Arraylist<Double>();
```

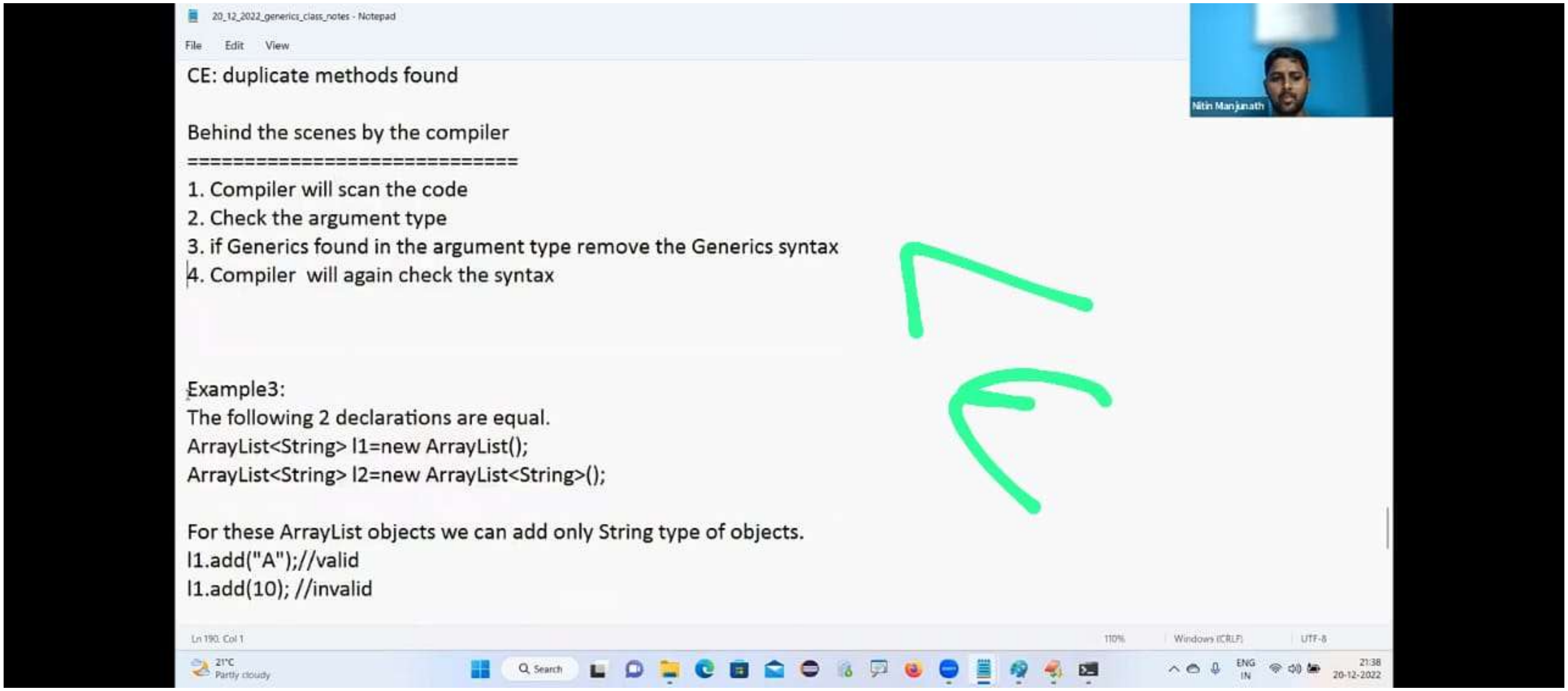
All are equal at runtime,becoz compiler will remove these generics syntax

```
    ArrayList l=new ArrayList();
```

Example 1:

```
import  java.util.*;
class Test {
    public static void main(String[] args) {
```

36



CE: duplicate methods found

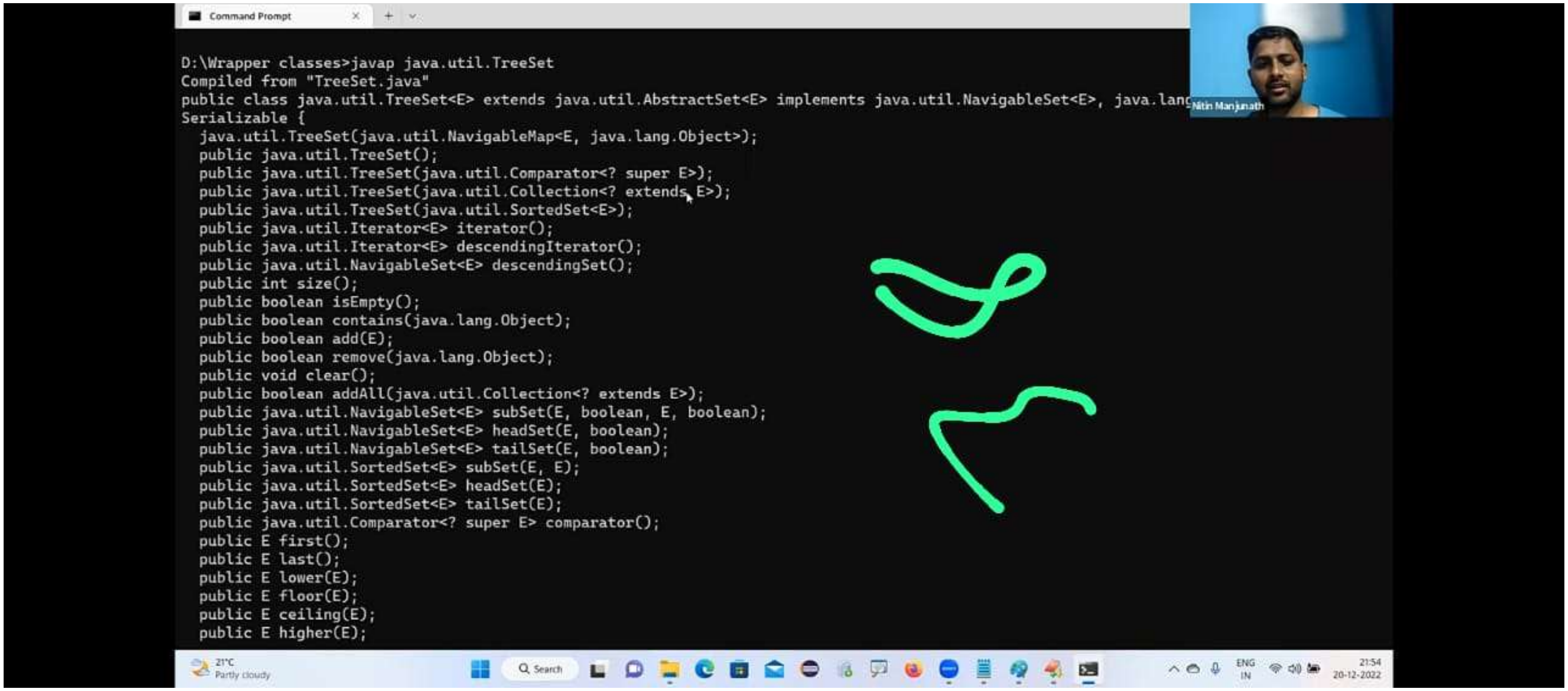
Behind the scenes by the compiler
=============================
1. Compiler will scan the code
2. Check the argument type
3. if Generics found in the argument type remove the Generics syntax
4. Compiler will again check the syntax

Example3:
The following 2 declarations are equal.

```
ArrayList<String> l1=new ArrayList();
ArrayList<String> l2=new ArrayList<String>();
```

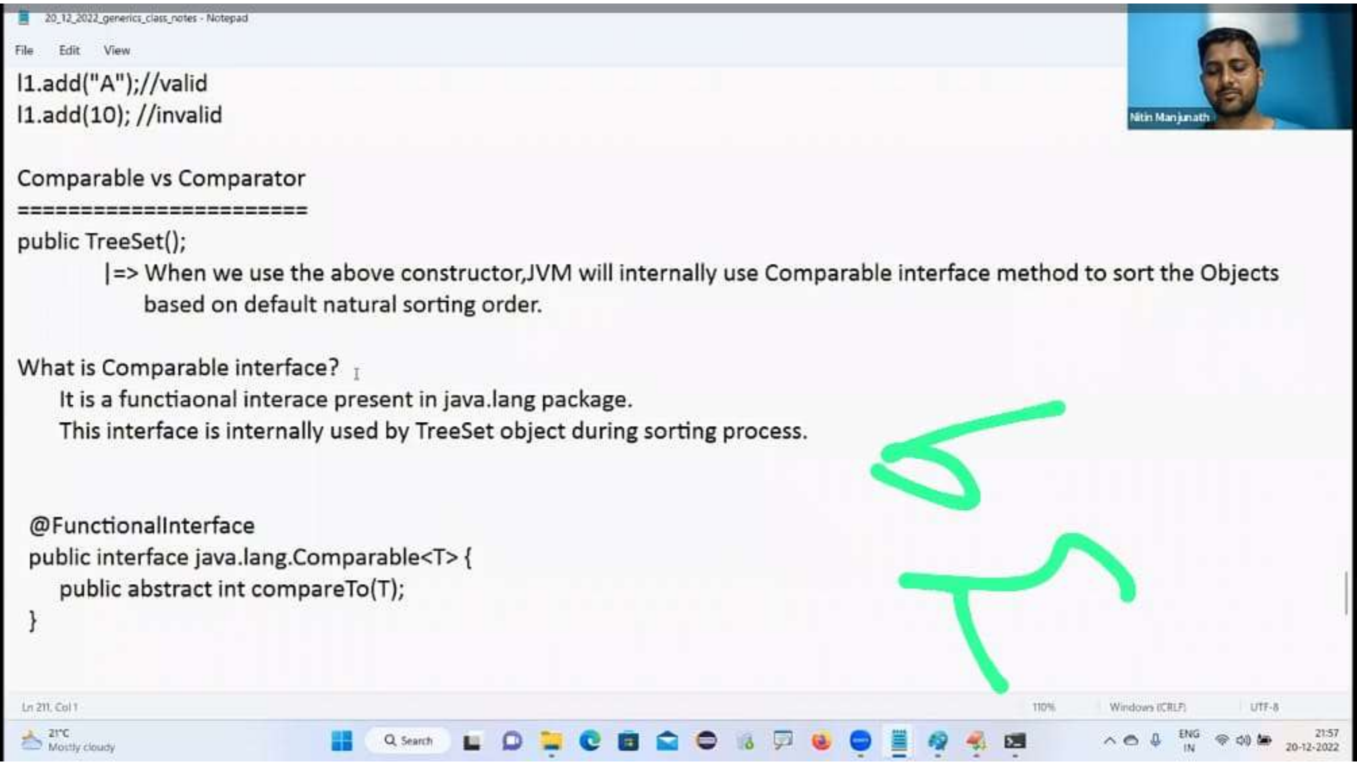
For these ArrayList objects we can add only String type of objects.

```
l1.add("A");//valid
l1.add(10); //invalid
```



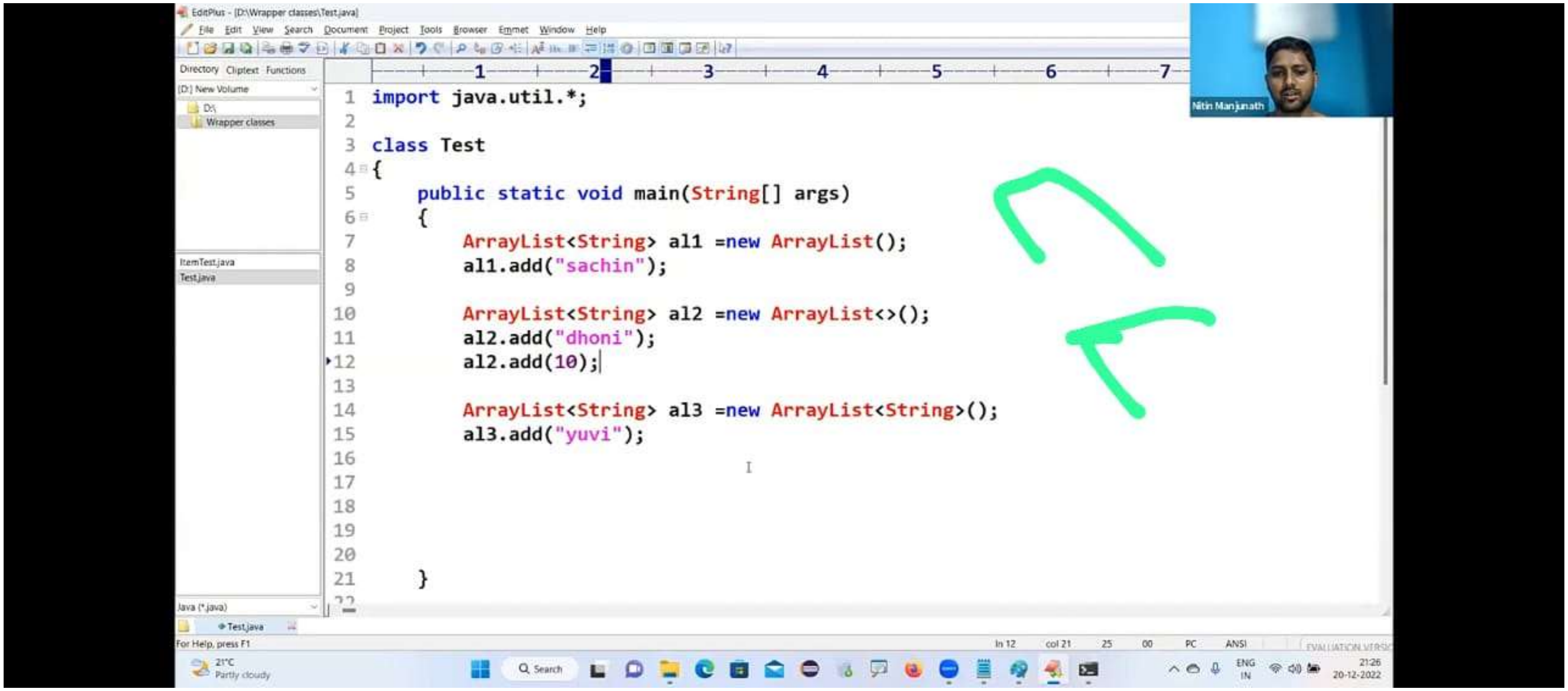
```
D:\Wrapper classes>javap java.util.TreeSet
Compiled from "TreeSet.java"
public class java.util.TreeSet<E> extends java.util.AbstractSet<E> implements java.util.NavigableSet<E>, java.lang
Serializable {
  java.util.TreeSet(java.util.NavigableMap<E, java.lang.Object>);
  public java.util.TreeSet();
  public java.util.TreeSet(java.util.Comparator<? super E>);
  public java.util.TreeSet(java.util.Collection<? extends E>);
  public java.util.TreeSet(java.util.SortedSet<E>);
  public java.util.Iterator<E> iterator();
  public java.util.Iterator<E> descendingIterator();
  public java.util.NavigableSet<E> descendingSet();
  public int size();
  public boolean isEmpty();
  public boolean contains(java.lang.Object);
  public boolean add(E);
  public boolean remove(java.lang.Object);
  public void clear();
  public boolean addAll(java.util.Collection<? extends E>);
  public java.util.NavigableSet<E> subSet(E, boolean, E, boolean);
  public java.util.NavigableSet<E> headSet(E, boolean);
  public java.util.NavigableSet<E> tailSet(E, boolean);
  public java.util.SortedSet<E> subSet(E, E);
  public java.util.SortedSet<E> headSet(E);
  public java.util.SortedSet<E> tailSet(E);
  public java.util.Comparator<? super E> comparator();
  public E first();
  public E last();
  public E lower(E);
  public E floor(E);
  public E ceiling(E);
  public E higher(E);
```

38

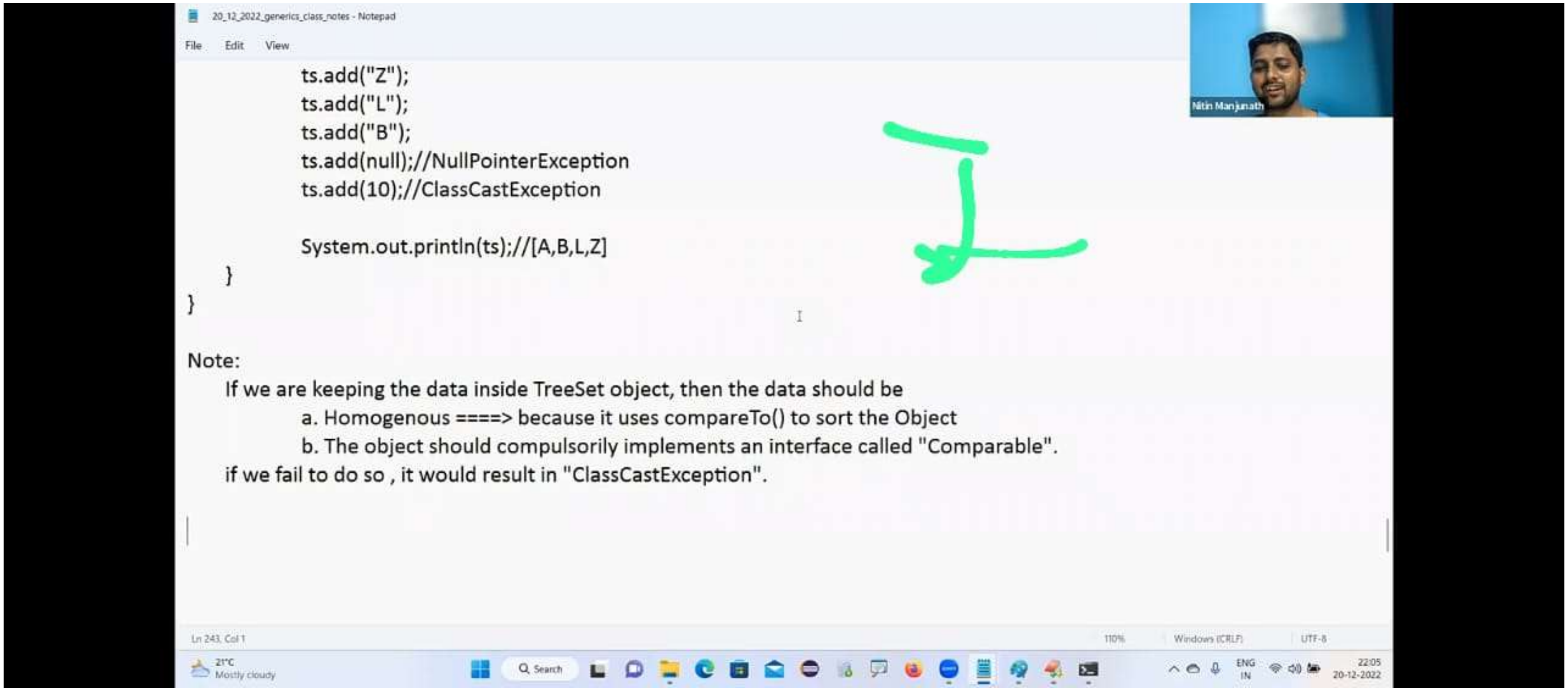


```
l1.add("A");//valid
l1.add(10); //invalid

Comparable vs Comparator
=======================
public TreeSet();
        |=> When we use the above constructor,JVM will internally use Comparable interface method to sort the Objects
           based on default natural sorting order.

What is Comparable interface?
    It is a functiaonal interace present in java.lang package.
    This interface is internally used by TreeSet object during sorting process.


 @FunctionalInterface
 public interface java.lang.Comparable<T> {
     public abstract int compareTo(T);
 }
```

59


EditPlus - [D:\Wrapper classes\Test.java]
File Edit View Search Document Project Tools Browser Emmet Window Help
Directory Cliptext Functions
[D:] New Volume
D:\
Wrapper classes
ItemTest.java
Test.java
Java (*.java)
Test.java
For Help, press F1
In 12 col 21 25 00 PC ANSI
Nitin Manjunath
21°C Partly cloudy
Search
ENG IN
21:26 20-12-2022

```
import java.util.*;

class Test
{
    public static void main(String[] args)
    {
        ArrayList<String> al1 =new ArrayList();
        al1.add("sachin");

        ArrayList<String> al2 =new ArrayList<>();
        al2.add("dhoni");
        al2.add(10);

        ArrayList<String> al3 =new ArrayList<String>();
        al3.add("yuvi");





    }
```



```
            ts.add("Z");
            ts.add("L");
            ts.add("B");
            ts.add(null);//NullPointerException
            ts.add(10);//ClassCastException

            System.out.println(ts);//[A,B,L,Z]
    }
}
```

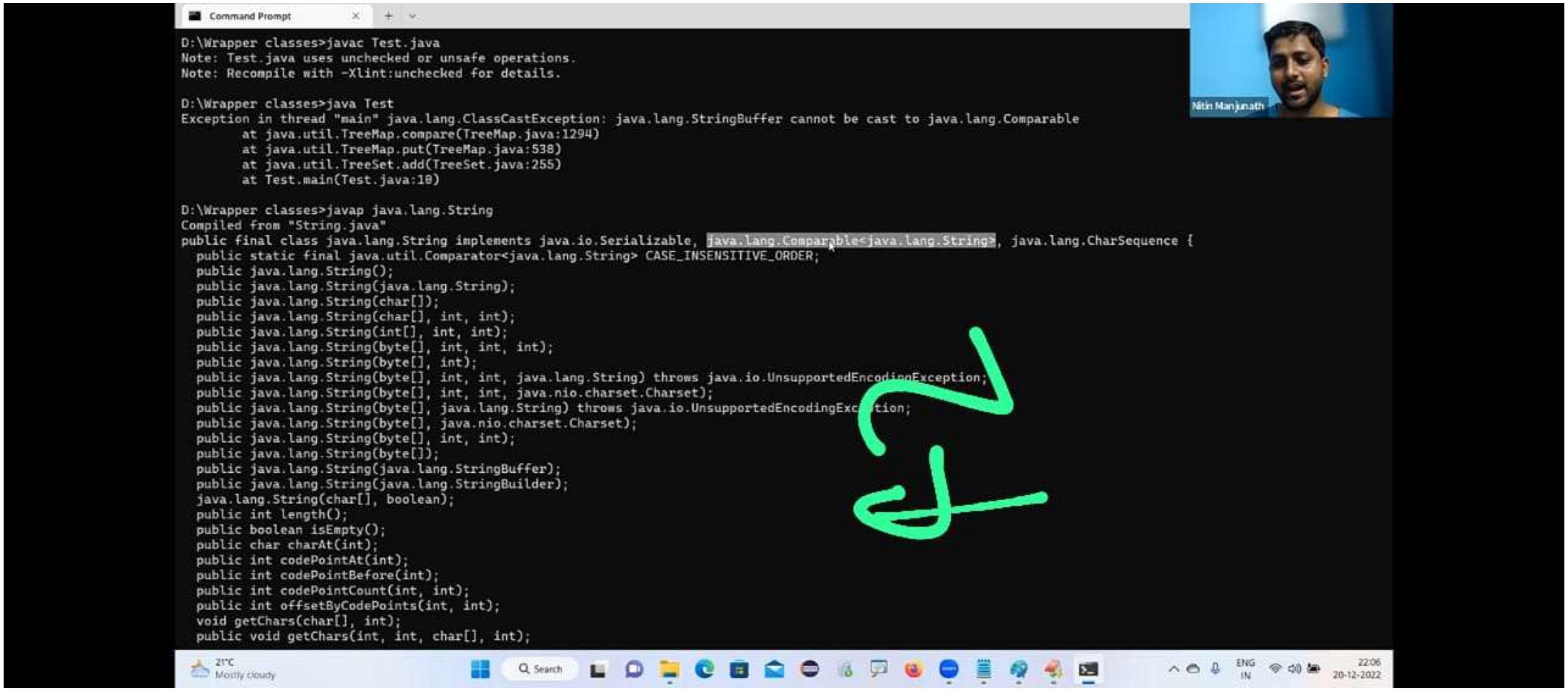
Note:

If we are keeping the data inside TreeSet object, then the data should be

a. Homogenous ====> because it uses compareTo() to sort the Object

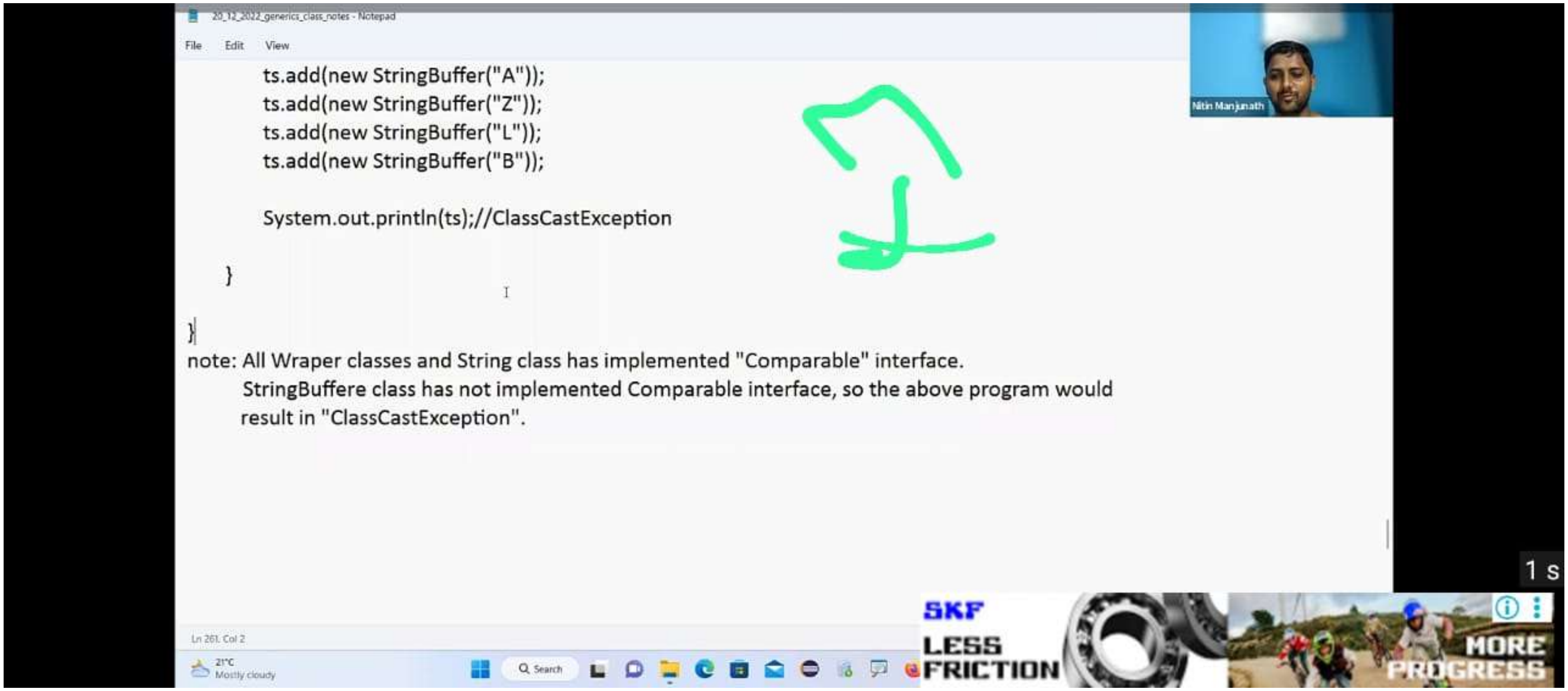
b. The object should compulsorily implements an interface called "Comparable".

if we fail to do so , it would result in "ClassCastException".



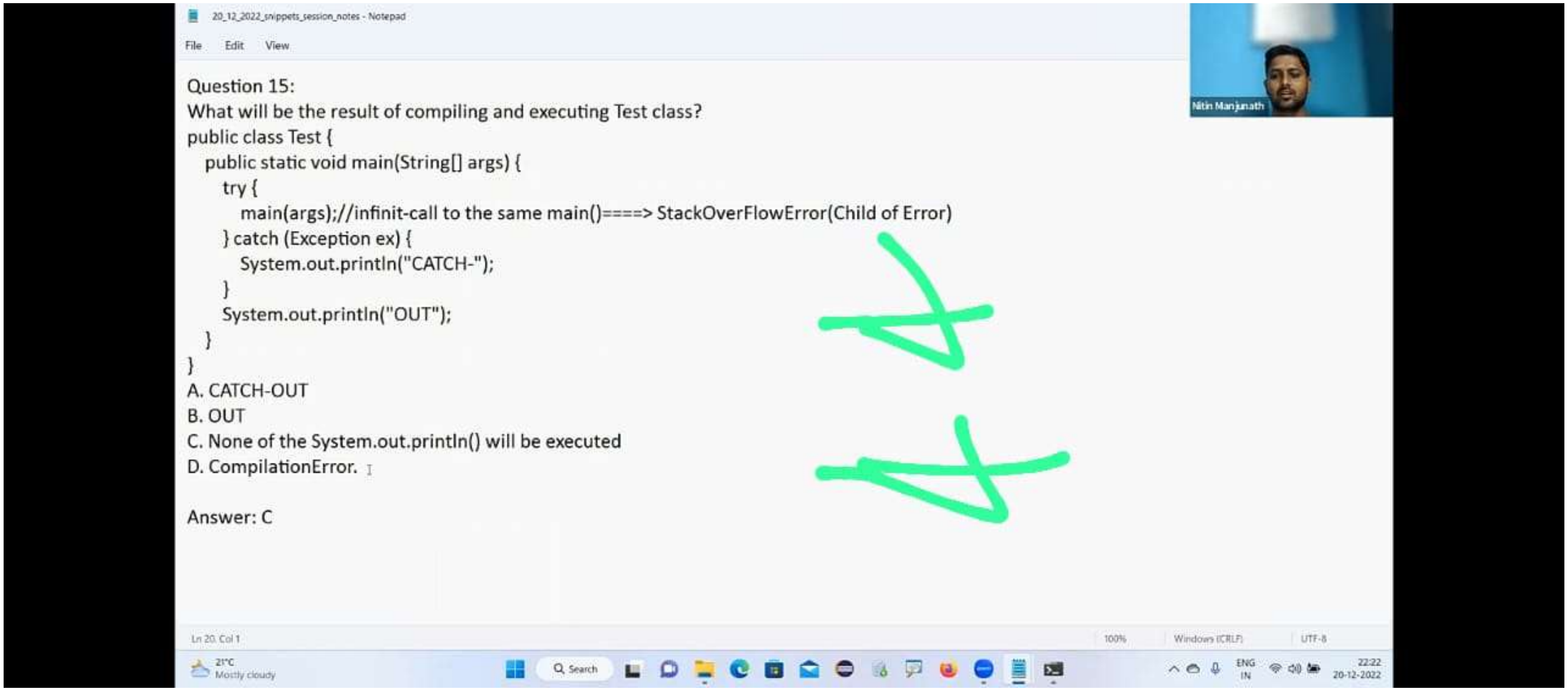
```
D:\Wrapper classes>javac Test.java
Note: Test.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

D:\Wrapper classes>java Test
Exception in thread "main" java.lang.ClassCastException: java.lang.StringBuffer cannot be cast to java.lang.Comparable
        at java.util.TreeMap.compare(TreeMap.java:1294)
        at java.util.TreeMap.put(TreeMap.java:538)
        at java.util.TreeSet.add(TreeSet.java:255)
        at Test.main(Test.java:10)

D:\Wrapper classes>javap java.lang.String
Compiled from "String.java"
public final class java.lang.String implements java.io.Serializable, java.lang.Comparable<java.lang.String>, java.lang.CharSequence {
  public static final java.util.Comparator<java.lang.String> CASE_INSENSITIVE_ORDER;
  public java.lang.String();
  public java.lang.String(java.lang.String);
  public java.lang.String(char[]);
  public java.lang.String(char[], int, int);
  public java.lang.String(int[], int, int);
  public java.lang.String(byte[], int, int, int);
  public java.lang.String(byte[], int);
  public java.lang.String(byte[], int, int, java.lang.String) throws java.io.UnsupportedEncodingException;
  public java.lang.String(byte[], int, int, java.nio.charset.Charset);
  public java.lang.String(byte[], java.lang.String) throws java.io.UnsupportedEncodingException;
  public java.lang.String(byte[], java.nio.charset.Charset);
  public java.lang.String(byte[], int, int);
  public java.lang.String(byte[]);
  public java.lang.String(java.lang.StringBuffer);
  public java.lang.String(java.lang.StringBuilder);
  java.lang.String(char[], boolean);
  public int length();
  public boolean isEmpty();
  public char charAt(int);
  public int codePointAt(int);
  public int codePointBefore(int);
  public int codePointCount(int, int);
  public int offsetByCodePoints(int, int);
  void getChars(char[], int);
  public void getChars(int, int, char[], int);
```

f2



```
        ts.add(new StringBuffer("A"));
        ts.add(new StringBuffer("Z"));
        ts.add(new StringBuffer("L"));
        ts.add(new StringBuffer("B"));

        System.out.println(ts);//ClassCastException

    }

}
```

note: All Wraper classes and String class has implemented "Comparable" interface.
StringBuffere class has not implemented Comparable interface, so the above program would result in "ClassCastException".



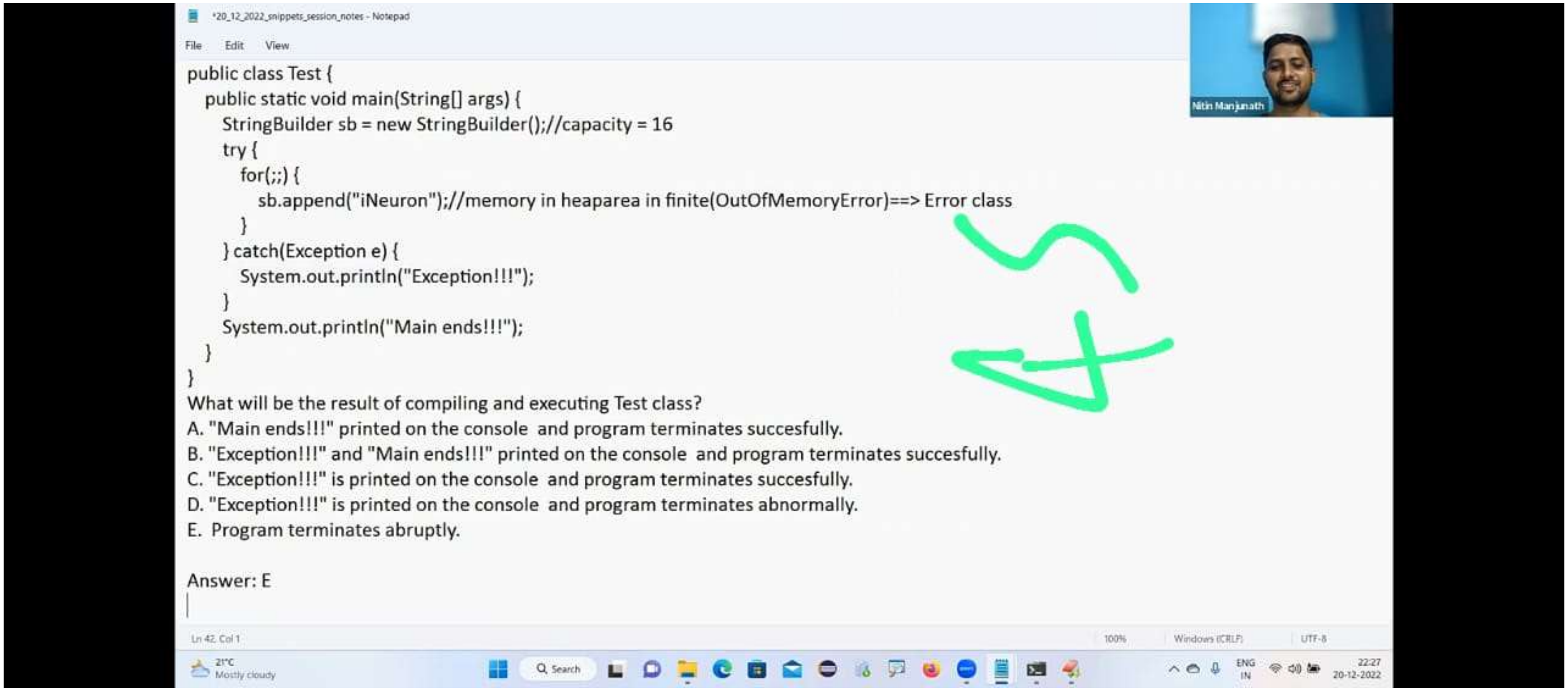
Question 15:
What will be the result of compiling and executing Test class?

```
public class Test {
  public static void main(String[] args) {
    try {
      main(args);//infinit-call to the same main()====> StackOverFlowError(Child of Error)
    } catch (Exception ex) {
      System.out.println("CATCH-");
    }
    System.out.println("OUT");
  }
}
```

A. CATCH-OUT
B. OUT
C. None of the System.out.println() will be executed
D. CompilationError.

Answer: C

44



```
public class Test {
  public static void main(String[] args) {
    StringBuilder sb = new StringBuilder();//capacity = 16
    try {
      for(;;) {
        sb.append("iNeuron");//memory in heaparea in finite(OutOfMemoryError)==> Error class
      }
    } catch(Exception e) {
      System.out.println("Exception!!!");
    }
    System.out.println("Main ends!!!");
  }
}
```

What will be the result of compiling and executing Test class?

A. "Main ends!!!" printed on the console and program terminates succesfully.

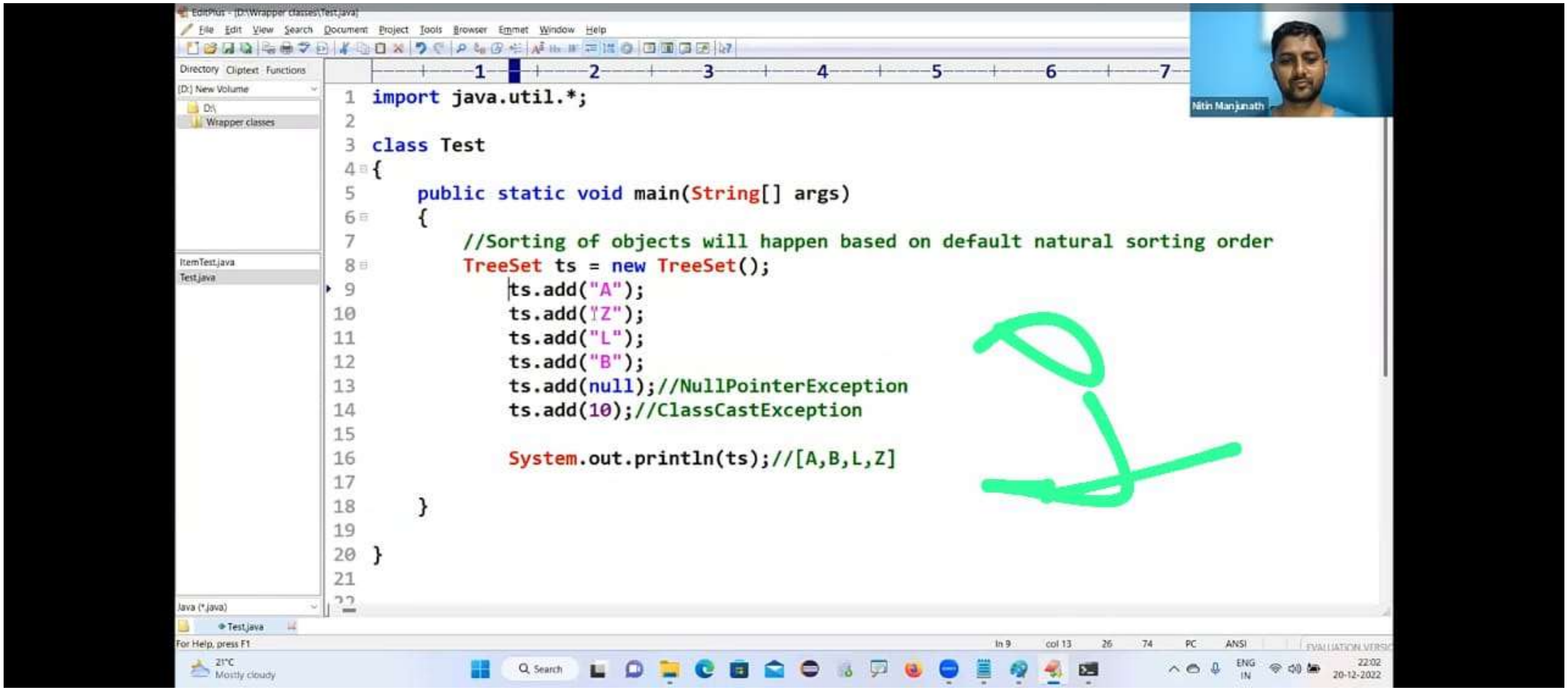
B. "Exception!!!" and "Main ends!!!" printed on the console and program terminates succesfully.

C. "Exception!!!" is printed on the console and program terminates succesfully.

D. "Exception!!!" is printed on the console and program terminates abnormally.

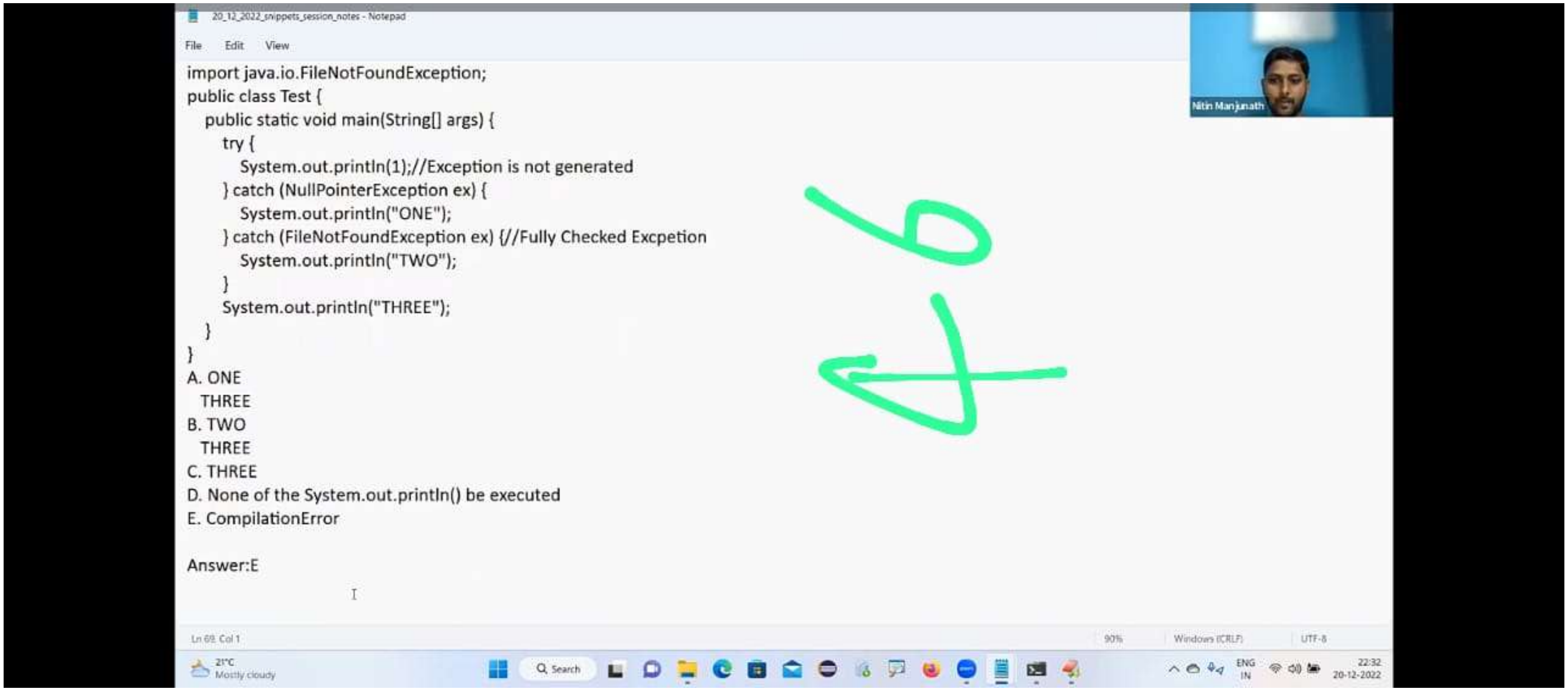
E. Program terminates abruptly.

Answer: E


EditPlus - [D:\Wrapper classes\Test.java]
File Edit View Search Document Project Tools Browser Emmet Window Help
Directory Cliptext Functions
[D:] New Volume
D:\
Wrapper classes
ItemTest.java
Test.java
Java (*.java)
Test.java
For Help, press F1
In 9 col 13 26 74 PC ANSI
Nitin Manjunath
21°C
Mostly cloudy
Search
ENG
IN
22:02
20-12-2022

```
import java.util.*;

class Test
{
    public static void main(String[] args)
    {
        //Sorting of objects will happen based on default natural sorting order
        TreeSet ts = new TreeSet();
            ts.add("A");
            ts.add("Z");
            ts.add("L");
            ts.add("B");
            ts.add(null);//NullPointerException
            ts.add(10);//ClassCastException

            System.out.println(ts);//[A,B,L,Z]

    }

}
```

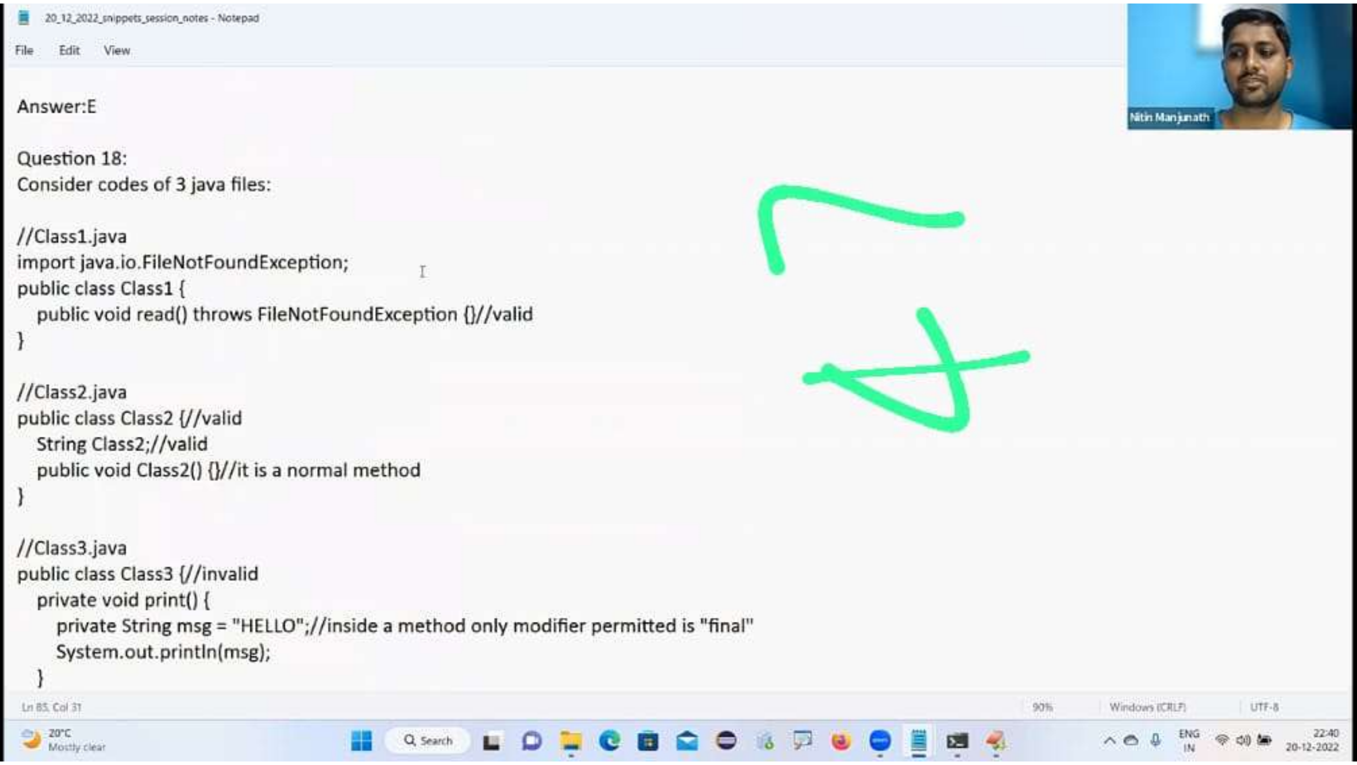
46



```
import java.io.FileNotFoundException;
public class Test {
  public static void main(String[] args) {
    try {
      System.out.println(1);//Exception is not generated
    } catch (NullPointerException ex) {
      System.out.println("ONE");
    } catch (FileNotFoundException ex) {//Fully Checked Excpetion
      System.out.println("TWO");
    }
    System.out.println("THREE");
  }
}
```

A. ONE
  THREE
B. TWO
  THREE
C. THREE
D. None of the System.out.println() be executed
E. CompilationError

Answer:E

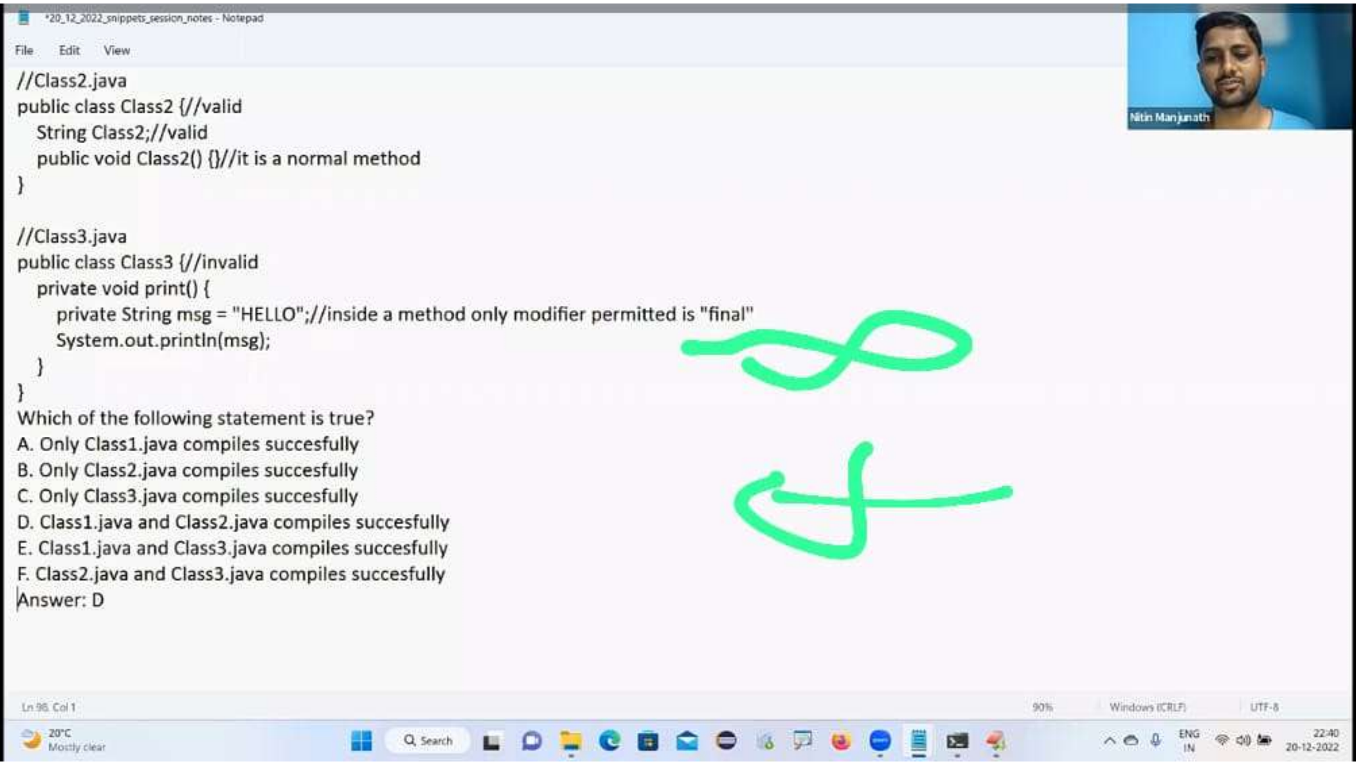


Answer:E

Question 18:
Consider codes of 3 java files:

```
//Class1.java
import java.io.FileNotFoundException;
public class Class1 {
   public void read() throws FileNotFoundException {}//valid
}

//Class2.java
public class Class2 {//valid
   String Class2;//valid
   public void Class2() {}//it is a normal method
}

//Class3.java
public class Class3 {//invalid
   private void print() {
      private String msg = "HELLO";//inside a method only modifier permitted is "final"
      System.out.println(msg);
   }
```

47



```
//Class2.java
public class Class2 {//valid
   String Class2;//valid
   public void Class2() {}//it is a normal method
}

//Class3.java
public class Class3 {//invalid
   private void print() {
      private String msg = "HELLO";//inside a method only modifier permitted is "final"
      System.out.println(msg);
   }
}
```

Which of the following statement is true?

A. Only Class1.java compiles succesfully

B. Only Class2.java compiles succesfully

C. Only Class3.java compiles succesfully

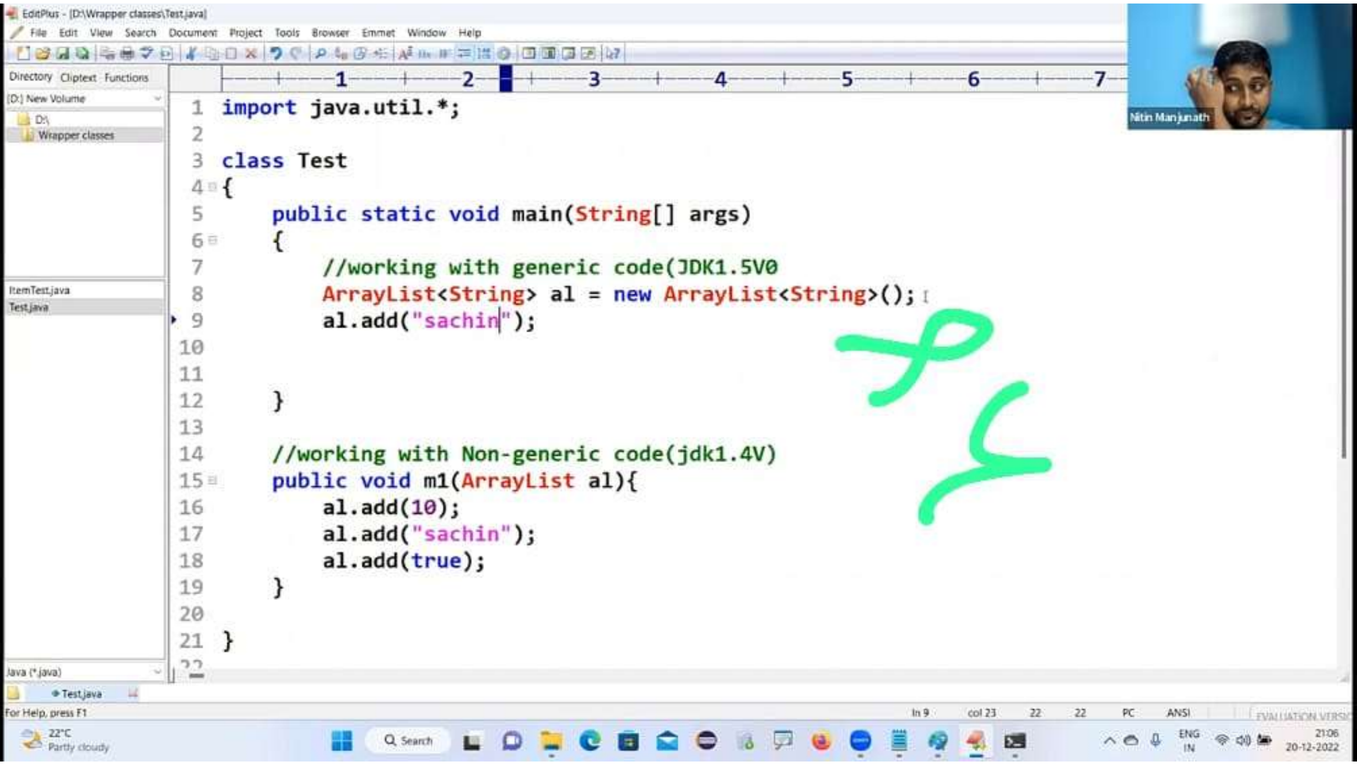
D. Class1.java and Class2.java compiles succesfully

E. Class1.java and Class3.java compiles succesfully

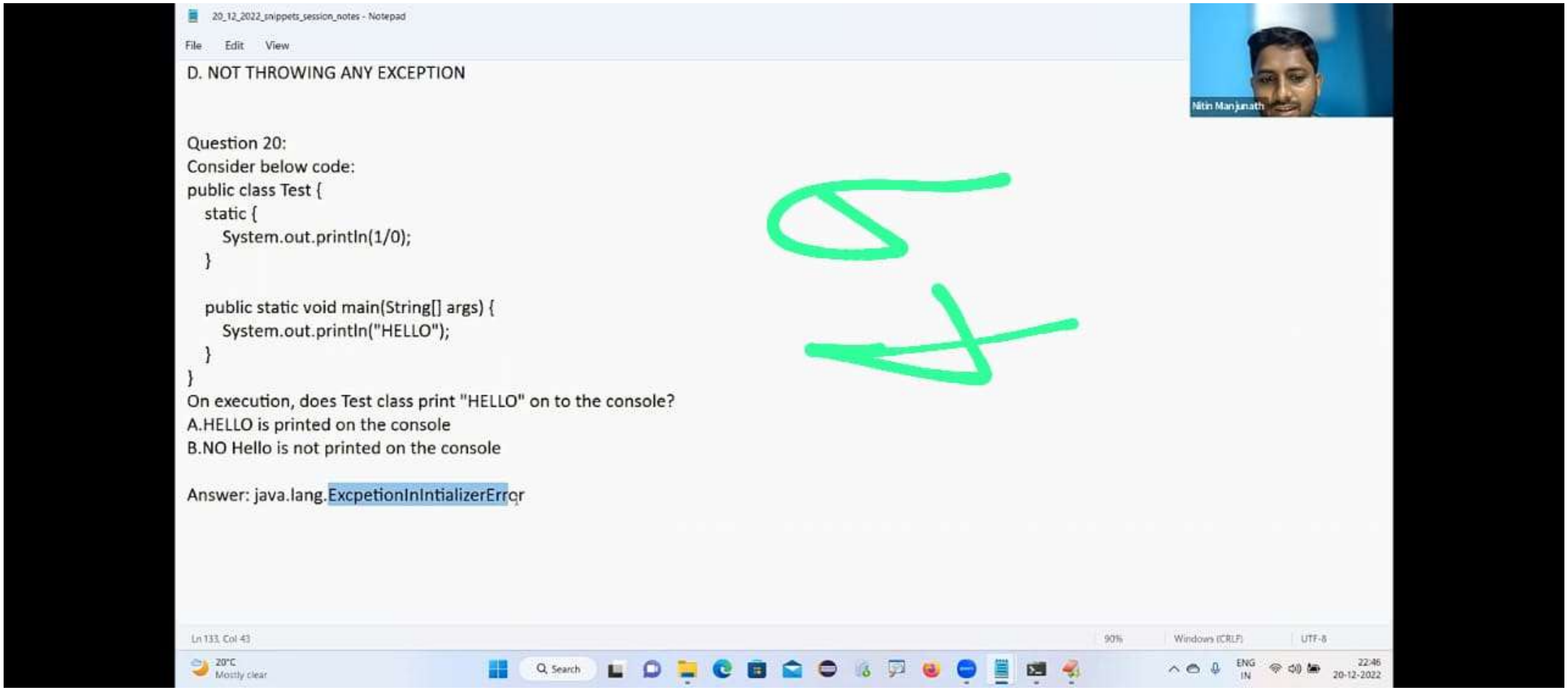
F. Class2.java and Class3.java compiles succesfully

Answer: D

48


EditPlus - [D:\Wrapper classes\Test.java]
File Edit View Search Document Project Tools Browser Emmet Window Help
Directory Cliptext Functions
[D:] New Volume
D:\
Wrapper classes
ItemTest.java
Test.java
Java (*.java)
Test.java
For Help, press F1
Nitin Manjunath

```java
import java.util.*;

class Test
{
    public static void main(String[] args)
    {
        //working with generic code(JDK1.5V0
        ArrayList<String> al = new ArrayList<String>();
        al.add("sachin");


    }

    //working with Non-generic code(jdk1.4V)
    public void m1(ArrayList al){
        al.add(10);
        al.add("sachin");
        al.add(true);
    }

}
```



D. NOT THROWING ANY EXCEPTION

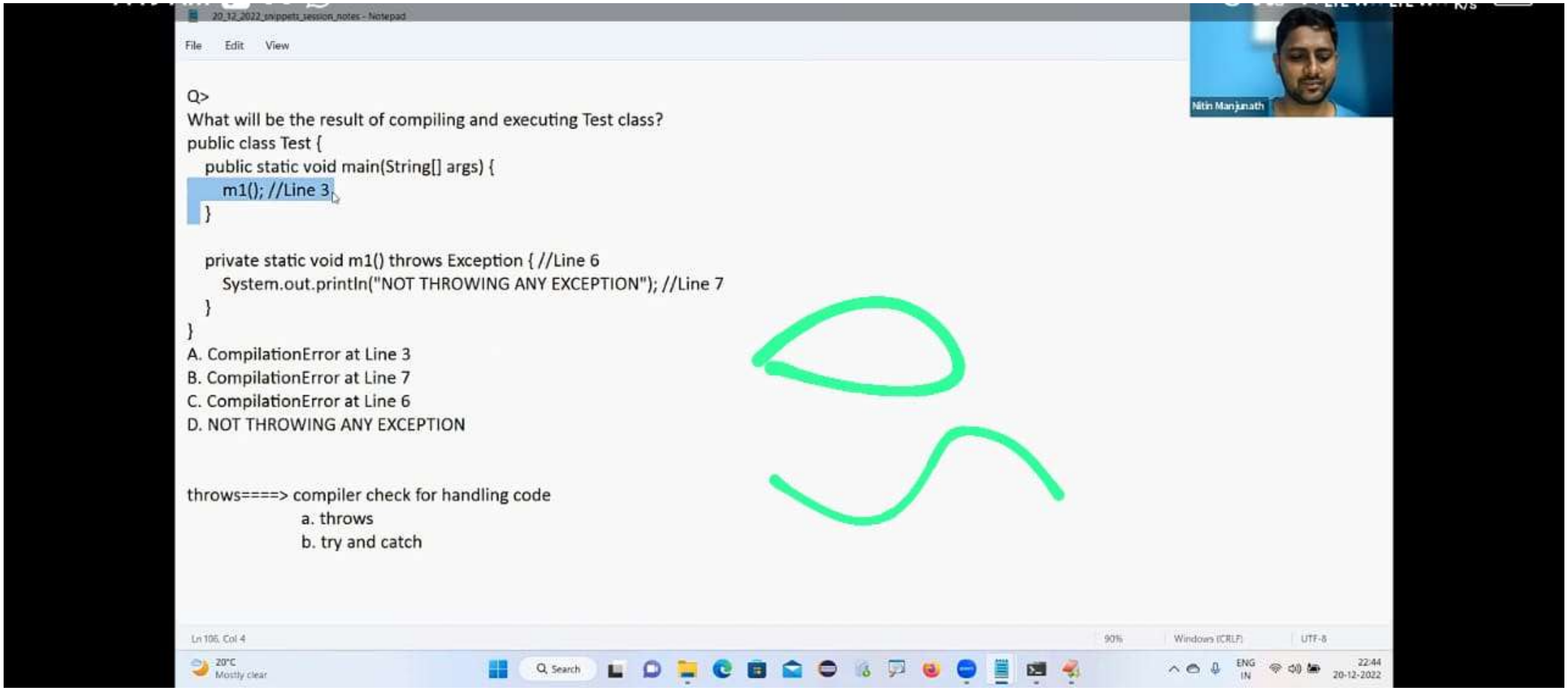
Question 20:
Consider below code:

```
public class Test {
   static {
      System.out.println(1/0);
   }

   public static void main(String[] args) {
      System.out.println("HELLO");
   }
}
```

On execution, does Test class print "HELLO" on to the console?
A.HELLO is printed on the console
B.NO Hello is not printed on the console

Answer: java.lang.ExcpetionInIntializerError

49



Q>
What will be the result of compiling and executing Test class?

```
public class Test {
   public static void main(String[] args) {
      m1(); //Line 3
   }

   private static void m1() throws Exception { //Line 6
      System.out.println("NOT THROWING ANY EXCEPTION"); //Line 7
   }
}
```

A. CompilationError at Line 3
B. CompilationError at Line 7
C. CompilationError at Line 6
D. NOT THROWING ANY EXCEPTION

throws====> compiler check for handling code
a. throws
b. try and catch