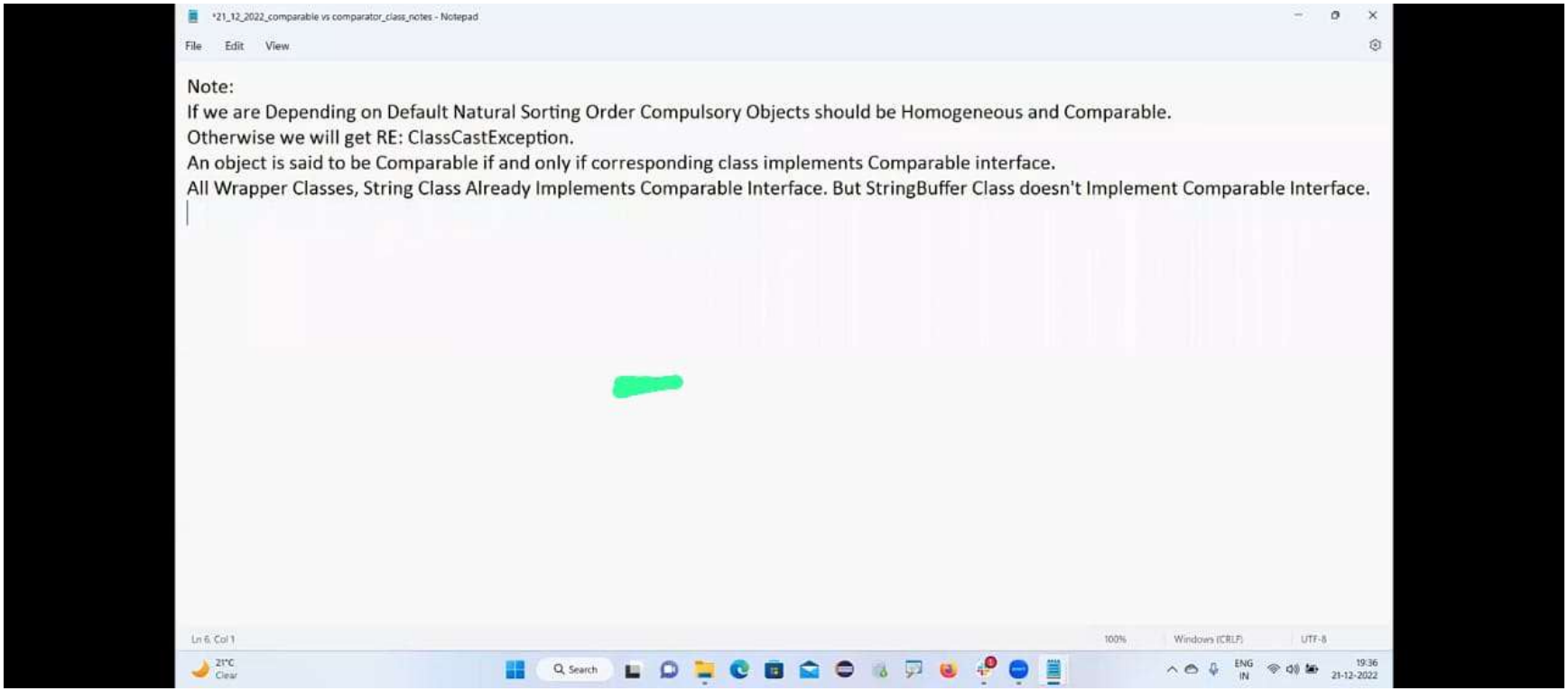
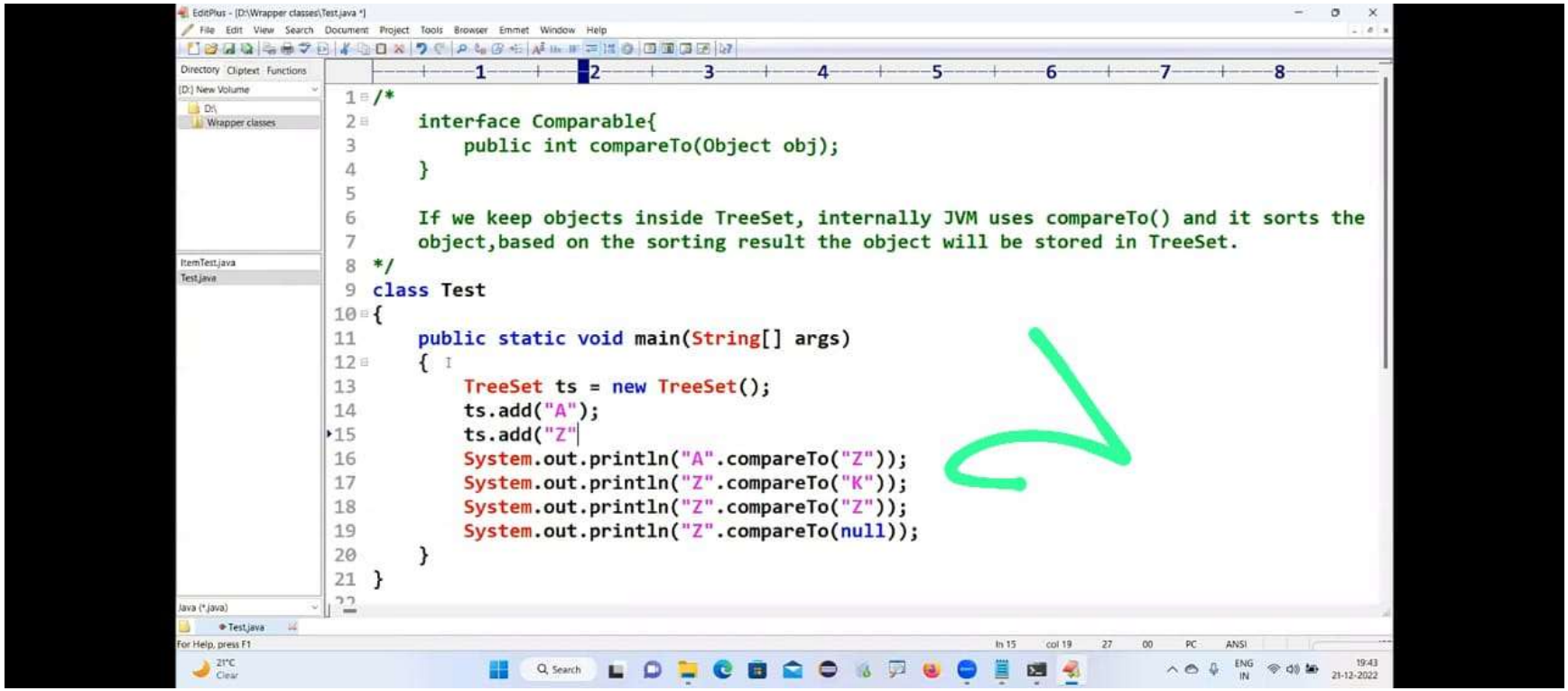
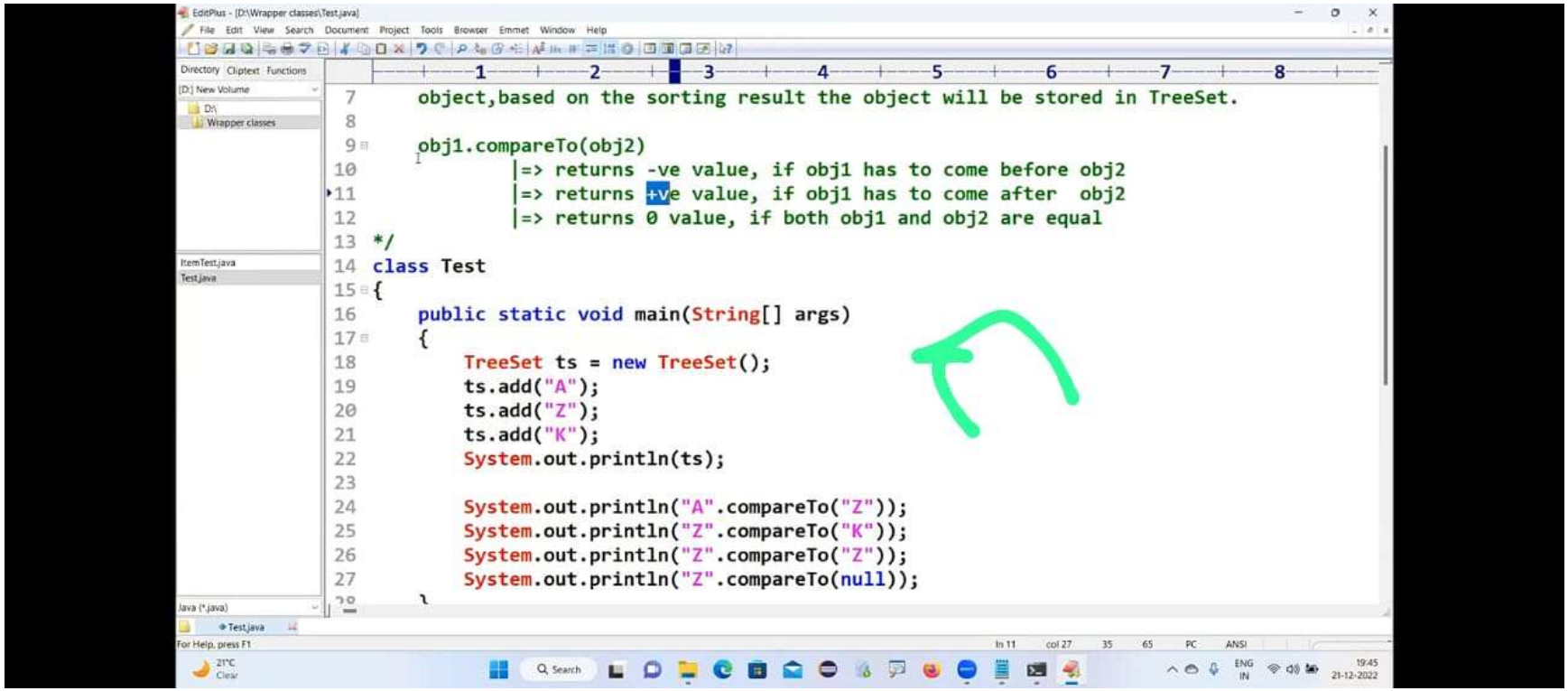
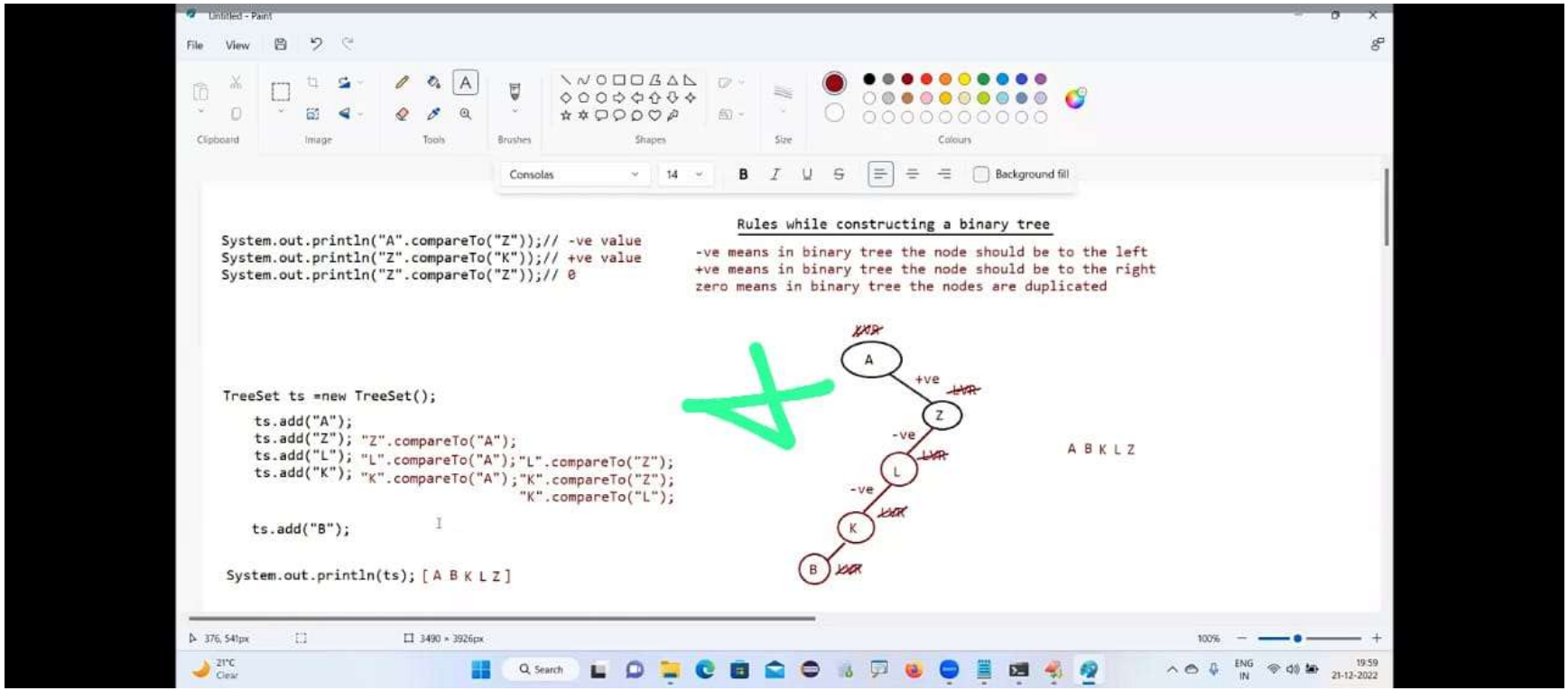
Comparable VS Comprator

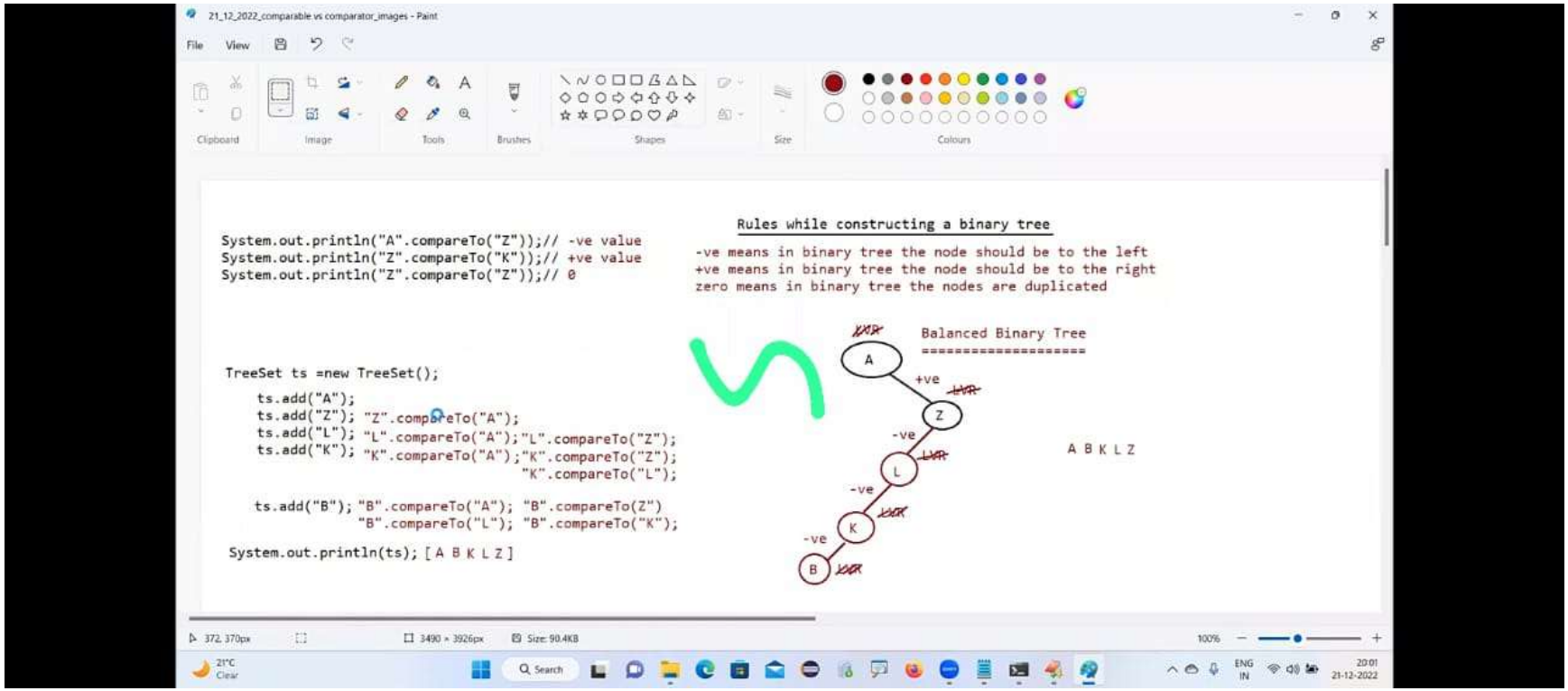


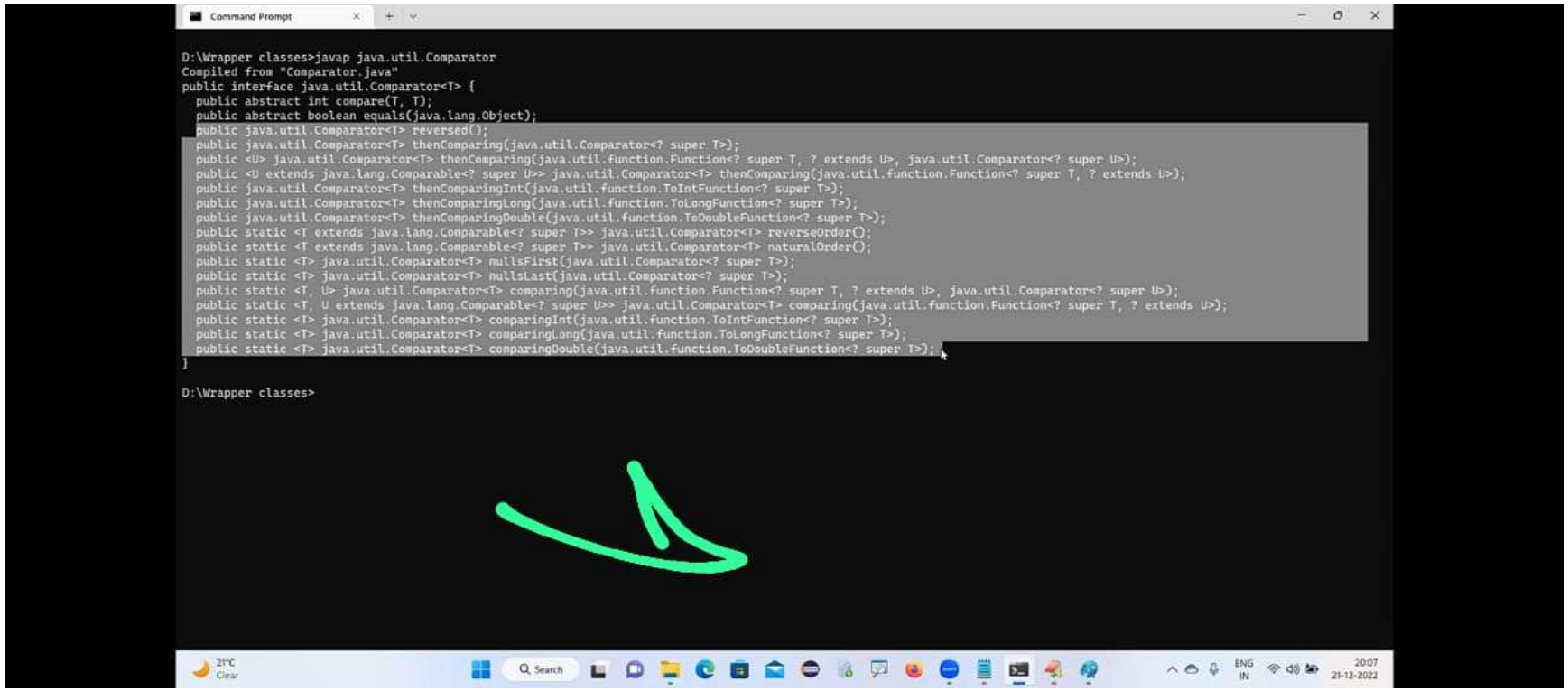



```
/*
    interface Comparable{
        public int compareTo(Object obj);
    }

    If we keep objects inside TreeSet, internally JVM uses compareTo() and it sorts the
    object,based on the sorting result the object will be stored in TreeSet.
*/
class Test
{
    public static void main(String[] args)
    {
        TreeSet ts = new TreeSet();
        ts.add("A");
        ts.add("Z"
        System.out.println("A".compareTo("Z"));
        System.out.println("Z".compareTo("K"));
        System.out.println("Z".compareTo("Z"));
        System.out.println("Z".compareTo(null));
    }
}
```


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
object,based on the sorting result the object will be stored in TreeSet.
obj1.compareTo(obj2)
|=> returns -ve value, if obj1 has to come before obj2
|=> returns +ve value, if obj1 has to come after obj2
|=> returns 0 value, if both obj1 and obj2 are equal
*/
class Test
{
public static void main(String[] args)
{
TreeSet ts = new TreeSet();
ts.add("A");
ts.add("Z");
ts.add("K");
System.out.println(ts);
System.out.println("A".compareTo("Z"));
System.out.println("Z".compareTo("K"));
System.out.println("Z".compareTo("Z"));
System.out.println("Z".compareTo(null));
}
Test.java
For Help, press F1
Search
21-12-2022


Untitled - Paint
File
View
Clipboard
Image
Tools
Brushes
Shapes
Size
Colours
Consolas
14
Background fill
System.out.println("A".compareTo("Z"));// -ve value
System.out.println("Z".compareTo("K"));// +ve value
System.out.println("Z".compareTo("Z"));// 0
Rules while constructing a binary tree
-ve means in binary tree the node should be to the left
+ve means in binary tree the node should be to the right
zero means in binary tree the nodes are duplicated
TreeSet ts =new TreeSet();
ts.add("A");
ts.add("Z"); "Z".compareTo("A");
ts.add("L"); "L".compareTo("A");"L".compareTo("Z");
ts.add("K"); "K".compareTo("A");"K".compareTo("Z");
"K".compareTo("L");
ts.add("B");
System.out.println(ts); [A B K L Z]
A
+ve
Z
-ve
L
-ve
K
B
A B K L Z
376, 541px
3490 × 3926px
100%
21°C
Clear
Search
ENG
IN
19:59
21-12-2022


21_12_2022_comparable vs comparator_images - Paint
File
View
Clipboard
Image
Tools
Brushes
Shapes
Size
Colours
System.out.println("A".compareTo("Z"));// -ve value
System.out.println("Z".compareTo("K"));// +ve value
System.out.println("Z".compareTo("Z"));// 0
Rules while constructing a binary tree
-ve means in binary tree the node should be to the left
+ve means in binary tree the node should be to the right
zero means in binary tree the nodes are duplicated
TreeSet ts =new TreeSet();
ts.add("A");
ts.add("Z"); "Z".compareTo("A");
ts.add("L"); "L".compareTo("A");"L".compareTo("Z");
ts.add("K"); "K".compareTo("A");"K".compareTo("Z");
"K".compareTo("L");
ts.add("B"); "B".compareTo("A"); "B".compareTo(Z")
"B".compareTo("L"); "B".compareTo("K");
System.out.println(ts); [A B K L Z]
Balanced Binary Tree
A
+ve
Z
-ve
L
-ve
K
-ve
B
A B K L Z
372, 370px
3490 × 3926px
Size: 90.4KB
100%
21°C
Clear
Search
ENG
IN
20:01
21-12-2022


Command Prompt
D:\Wrapper classes>javap java.util.Comparator
Compiled from "Comparator.java"
public interface java.util.Comparator<T> {
public abstract int compare(T, T);
public abstract boolean equals(java.lang.Object);
public java.util.Comparator<T> reversed();
public java.util.Comparator<T> thenComparing(java.util.Comparator<? super T>);
public <U> java.util.Comparator<T> thenComparing(java.util.function.Function<? super T, ? extends U>, java.util.Comparator<? super U>);
public <U extends java.lang.Comparable<? super U>> java.util.Comparator<T> thenComparing(java.util.function.Function<? super T, ? extends U>);
public java.util.Comparator<T> thenComparingInt(java.util.function.ToIntFunction<? super T>);
public java.util.Comparator<T> thenComparingLong(java.util.function.ToLongFunction<? super T>);
public java.util.Comparator<T> thenComparingDouble(java.util.function.ToDoubleFunction<? super T>);
public static <T extends java.lang.Comparable<? super T>> java.util.Comparator<T> reverseOrder();
public static <T extends java.lang.Comparable<? super T>> java.util.Comparator<T> naturalOrder();
public static <T> java.util.Comparator<T> nullsFirst(java.util.Comparator<? super T>);
public static <T> java.util.Comparator<T> nullsLast(java.util.Comparator<? super T>);
public static <T, U> java.util.Comparator<T> comparing(java.util.function.Function<? super T, ? extends U>, java.util.Comparator<? super U>);
public static <T, U extends java.lang.Comparable<? super U>> java.util.Comparator<T> comparing(java.util.function.Function<? super T, ? extends U>);
public static <T> java.util.Comparator<T> comparingInt(java.util.function.ToIntFunction<? super T>);
public static <T> java.util.Comparator<T> comparingLong(java.util.function.ToLongFunction<? super T>);
public static <T> java.util.Comparator<T> comparingDouble(java.util.function.ToDoubleFunction<? super T>);
}
D:\Wrapper classes>
21°C
Clear
Search
ENG
IN
20:07
21-12-2022

21_12_2022_comparable vs comparator_class_notes - Notepad

File Edit View

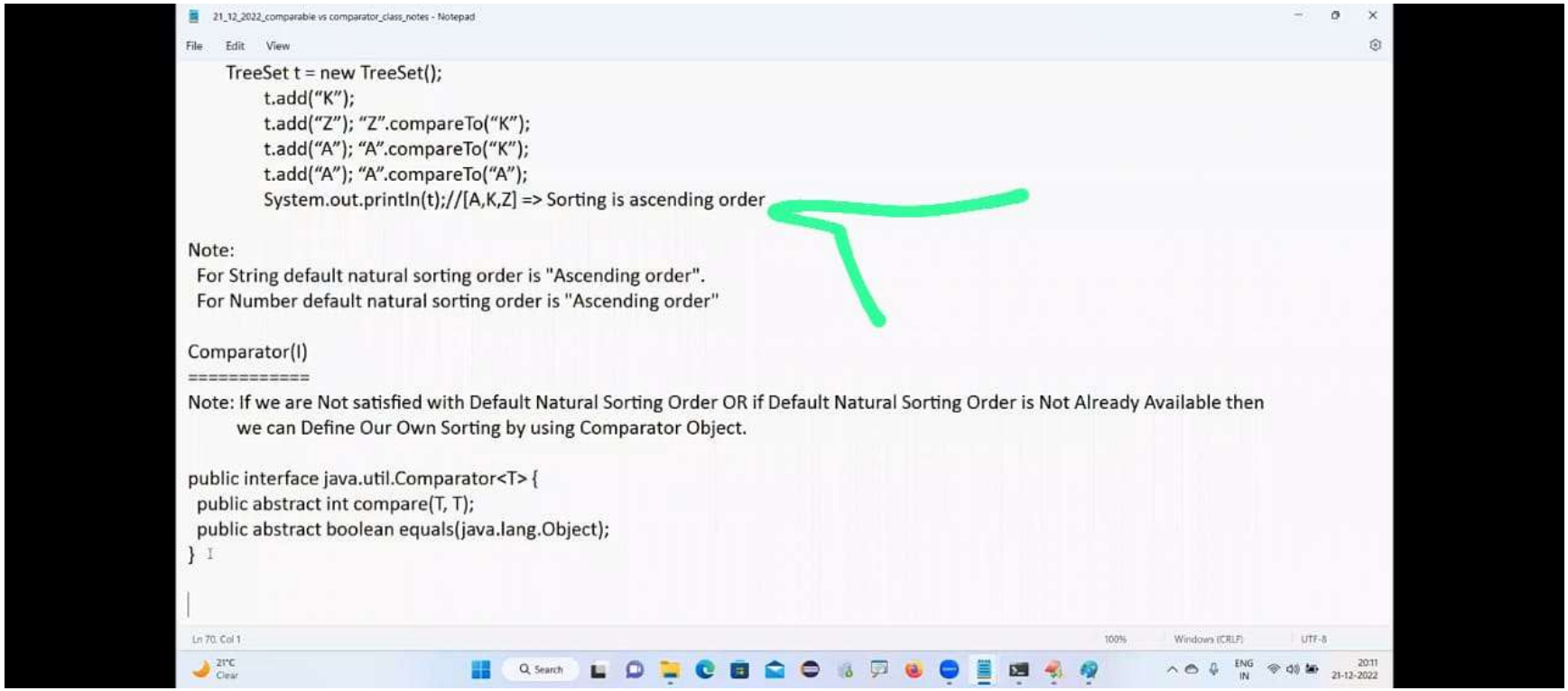
}

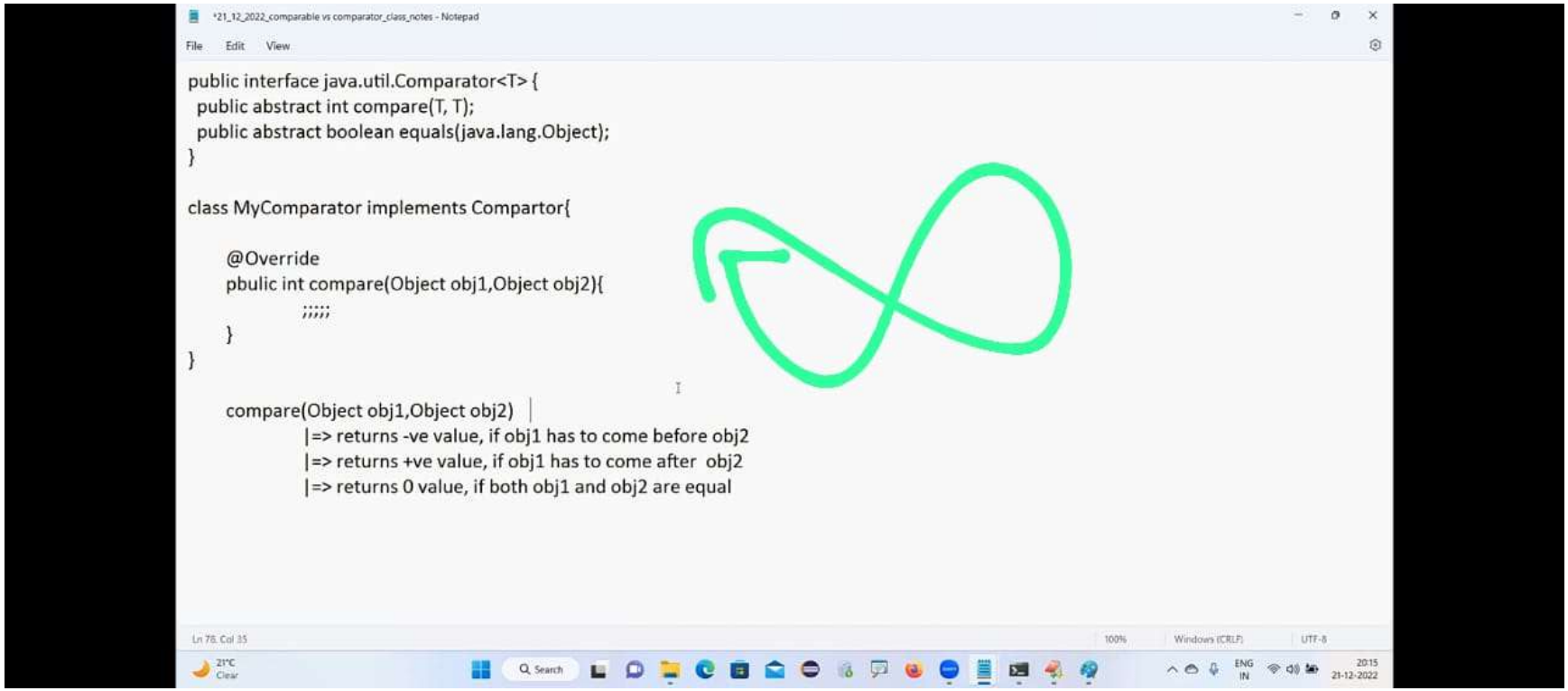
At Line 1 if we are Not Passing Comparator Object as an Argument then Internally JVM will Call compareTo(),
Which is Meant for Default Natural Sorting Order (Ascending Order).
In this Case the Output is [0, 5, 10, 15, 20].

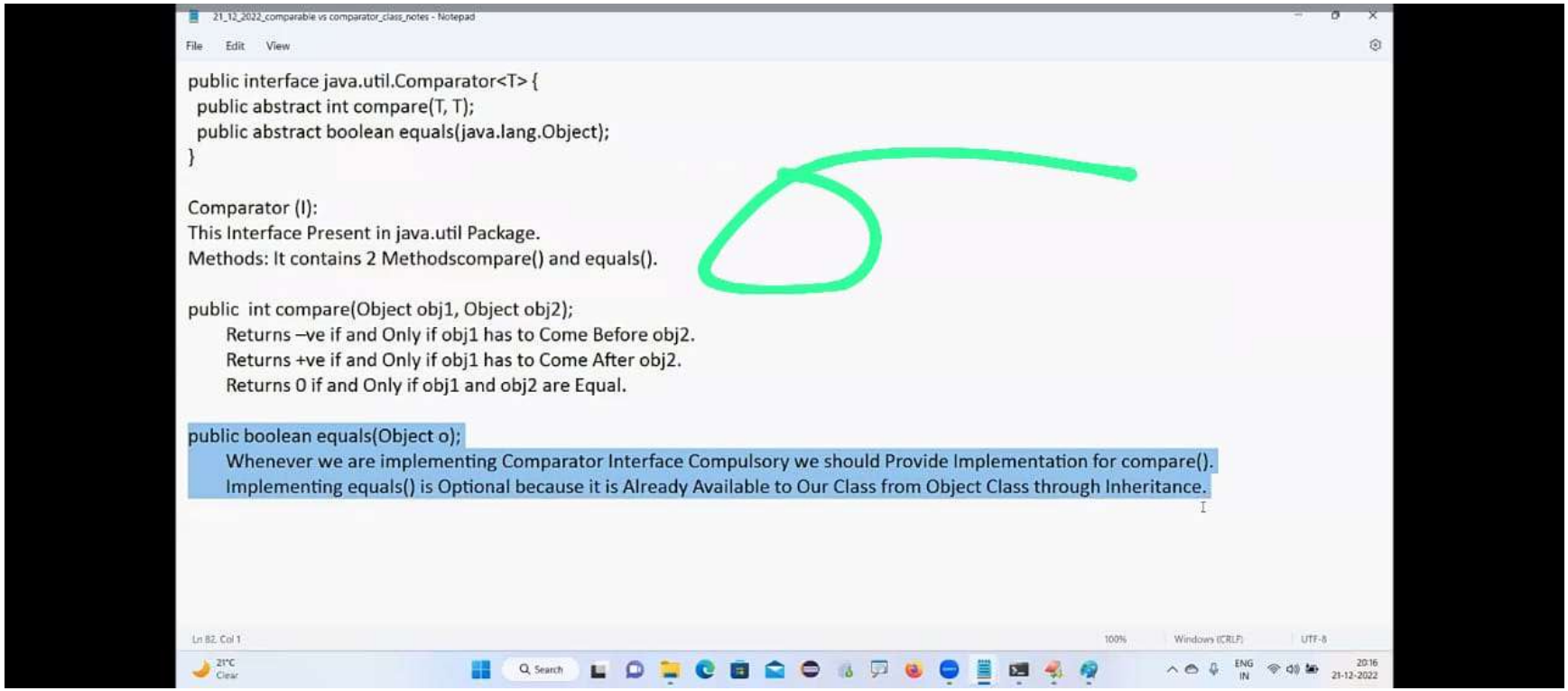
At Line 1 if we are Passing Comparator Object then JVM will Call compare() Instead of compareTo().
Which is Meant for Customized Sorting (can be Ascending /Descending Order).
In this Case the Ouput is [20, 15, 10, 5, 0]

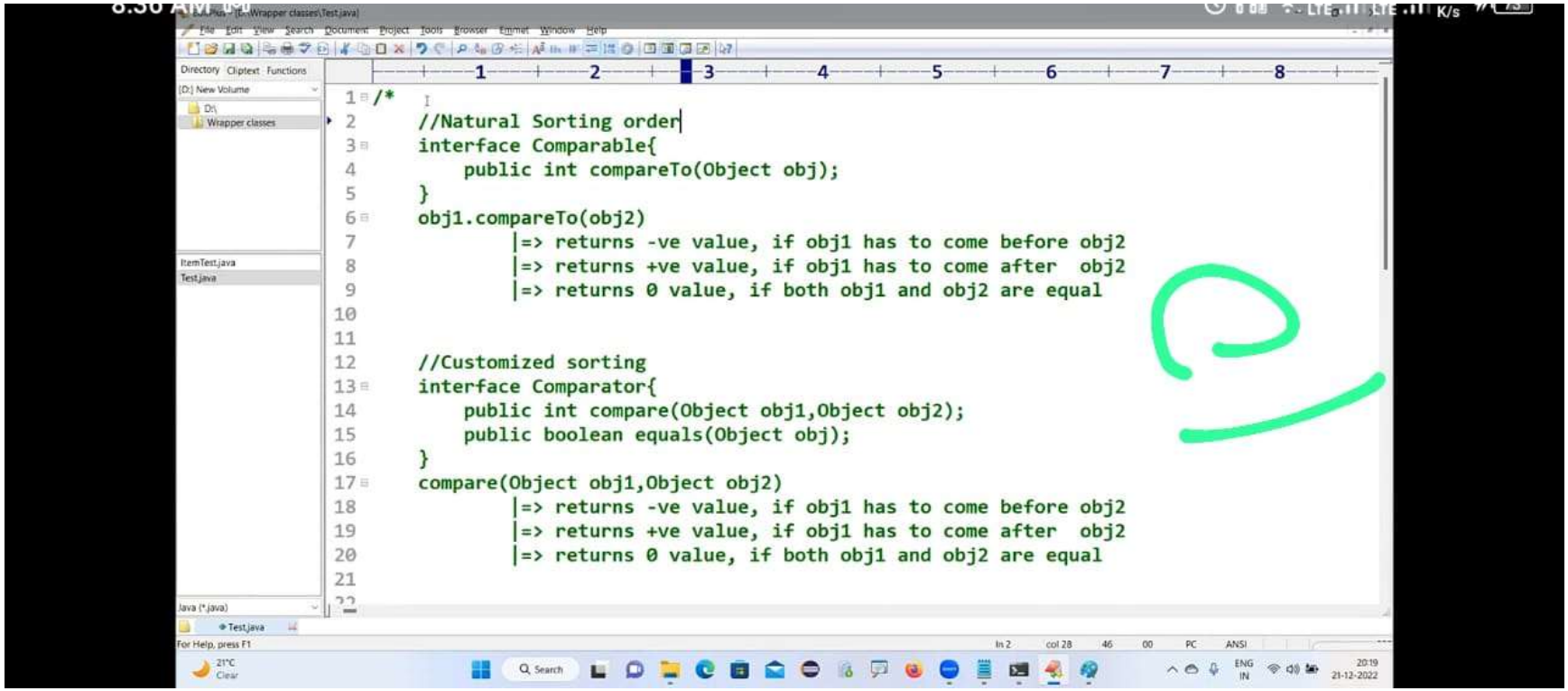
Ln 114, Col 45 100% Windows (CRLF) UTF-8

19°C Clear Search ENG IN 20:46 21-12-2022

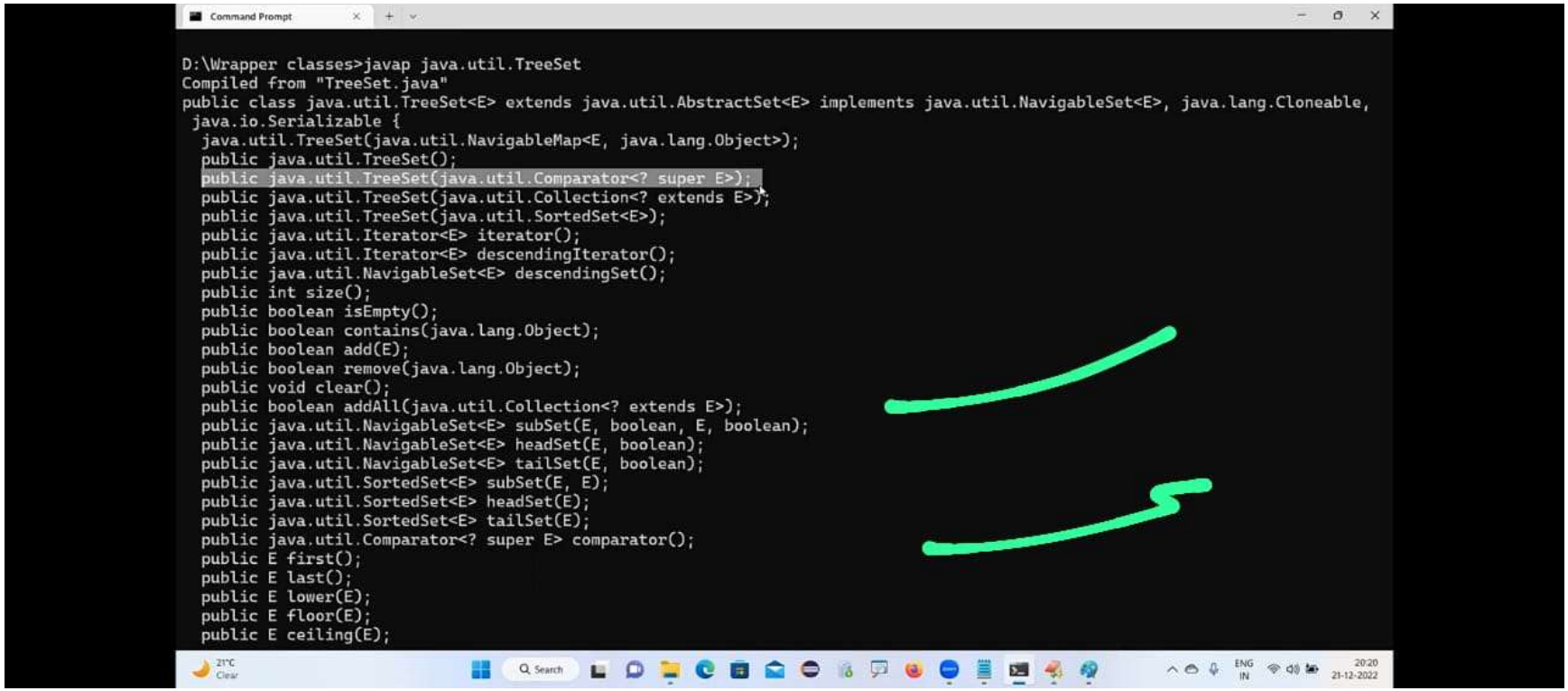

21_12_2022_comparable vs comparator_class_notes - Notepad
File Edit View
TreeSet t = new TreeSet();
t.add("K");
t.add("Z"); "Z".compareTo("K");
t.add("A"); "A".compareTo("K");
t.add("A"); "A".compareTo("A");
System.out.println(t);//[A,K,Z] => Sorting is ascending order
Note:
For String default natural sorting order is "Ascending order".
For Number default natural sorting order is "Ascending order"
Comparator(I)
==========
Note: If we are Not satisfied with Default Natural Sorting Order OR if Default Natural Sorting Order is Not Already Available then
we can Define Our Own Sorting by using Comparator Object.
public interface java.util.Comparator<T> {
public abstract int compare(T, T);
public abstract boolean equals(java.lang.Object);
}
Ln 70, Col 1
100%
Windows (CRLF)
UTF-8
21°C
Clear
Search
ENG
IN
20:11
21-12-2022

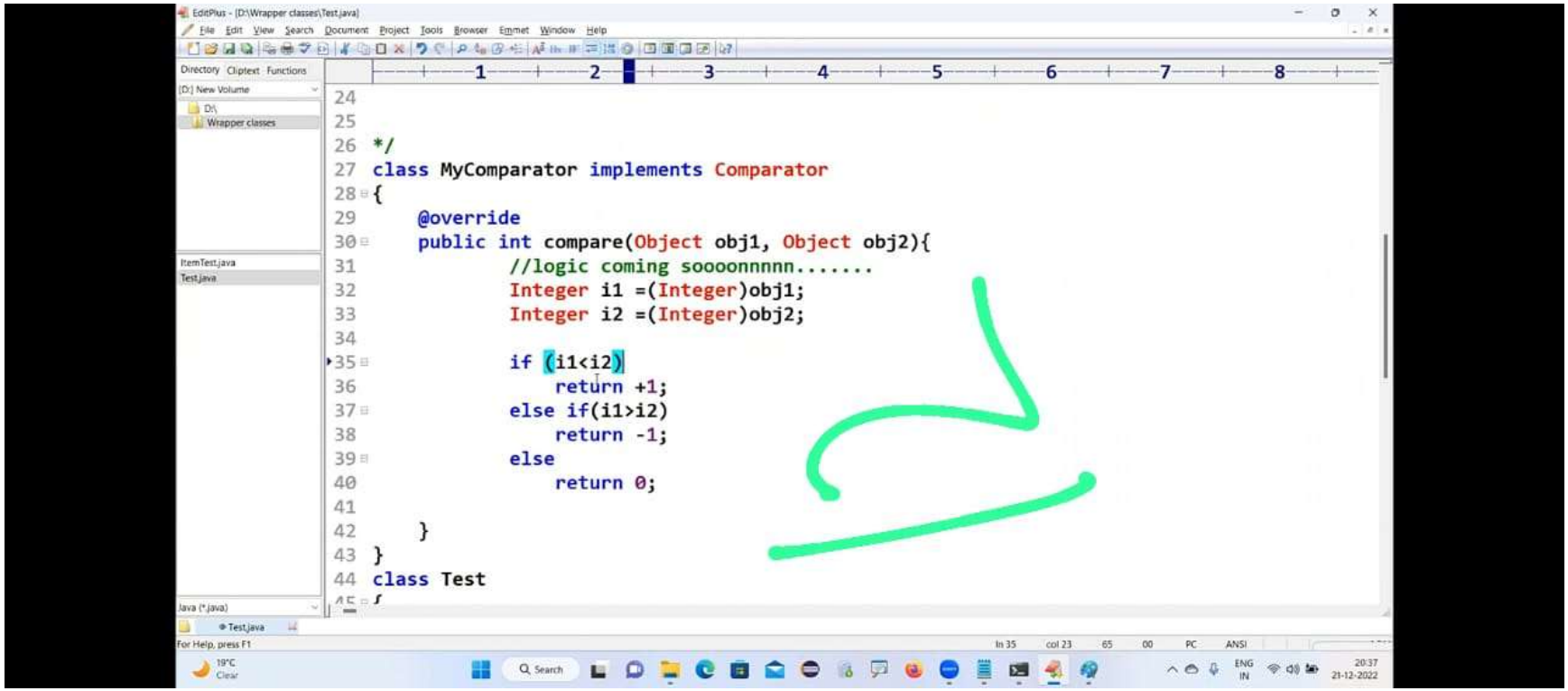

*21_12_2022_comparable vs comparator_class_notes - Notepad
File Edit View
public interface java.util.Comparator<T> {
public abstract int compare(T, T);
public abstract boolean equals(java.lang.Object);
}
class MyComparator implements Compartor{
@Override
pbulic int compare(Object obj1,Object obj2){
;;;;;
}
}
compare(Object obj1,Object obj2)
|=> returns -ve value, if obj1 has to come before obj2
|=> returns +ve value, if obj1 has to come after obj2
|=> returns 0 value, if both obj1 and obj2 are equal
Ln 76, Col 35
100%
Windows (CRLF)
UTF-8
21°C
Clear
Search
ENG
IN
20:15
21-12-2022


21_12_2022_comparable vs comparator_class_notes - Notepad
File Edit View
public interface java.util.Comparator<T> {
public abstract int compare(T, T);
public abstract boolean equals(java.lang.Object);
}
Comparator (I):
This Interface Present in java.util Package.
Methods: It contains 2 Methodscompare() and equals().
public int compare(Object obj1, Object obj2);
Returns –ve if and Only if obj1 has to Come Before obj2.
Returns +ve if and Only if obj1 has to Come After obj2.
Returns 0 if and Only if obj1 and obj2 are Equal.
public boolean equals(Object o);
Whenever we are implementing Comparator Interface Compulsory we should Provide Implementation for compare().
Implementing equals() is Optional because it is Already Available to Our Class from Object Class through Inheritance.
Ln 82, Col 1
100%
Windows (CRLF)
UTF-8
21°C
Clear
Search
ENG
IN
20:16
21-12-2022

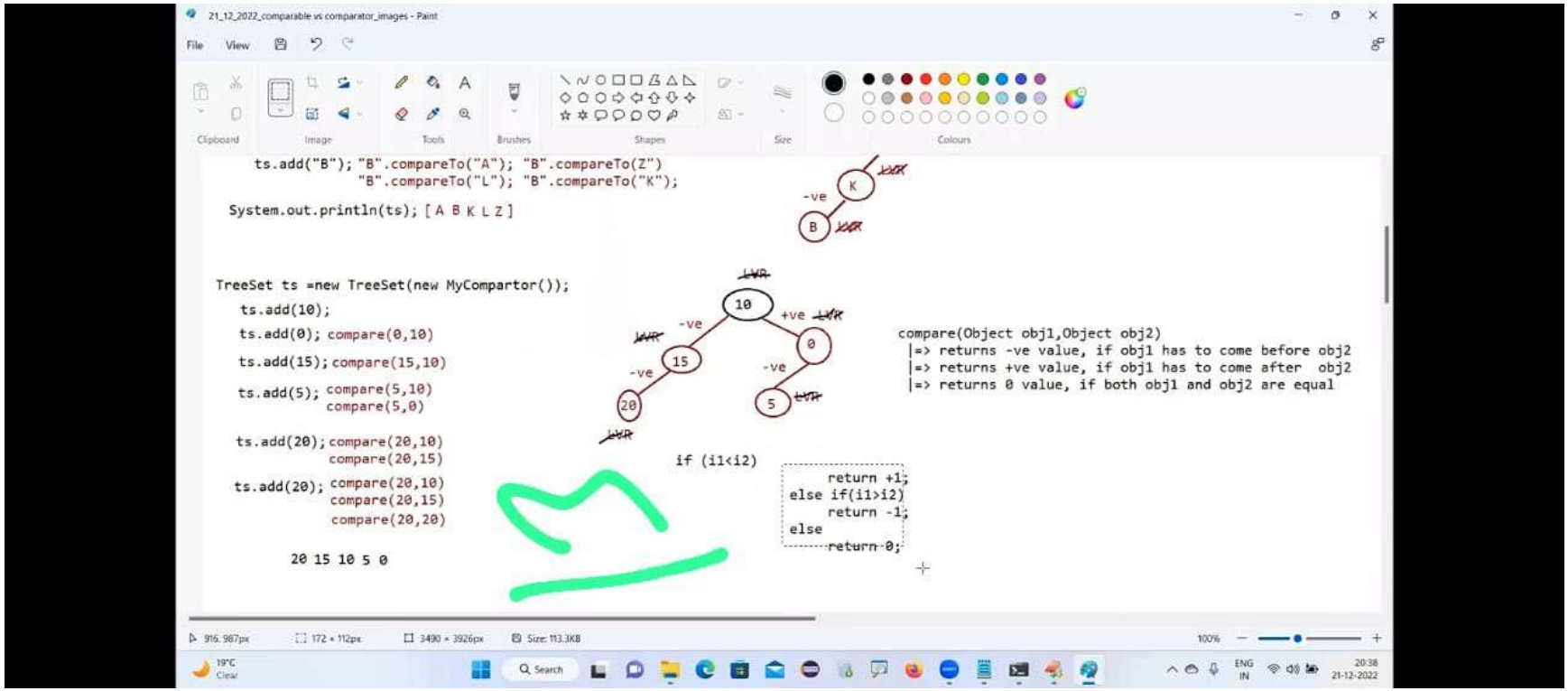


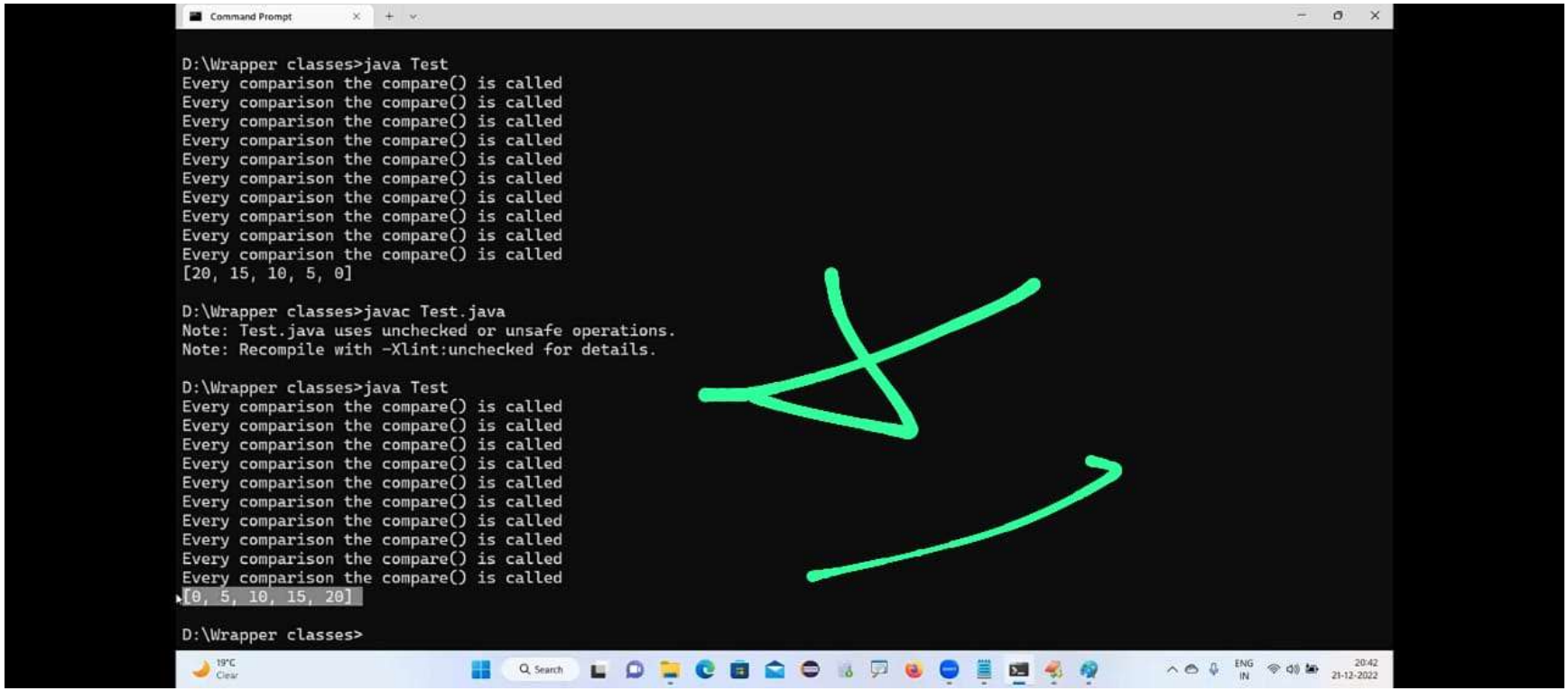

```
/*
    //Natural Sorting order
    interface Comparable{
        public int compareTo(Object obj);
    }
    obj1.compareTo(obj2)
            |=> returns -ve value, if obj1 has to come before obj2
            |=> returns +ve value, if obj1 has to come after  obj2
            |=> returns 0 value, if both obj1 and obj2 are equal


    //Customized sorting
    interface Comparator{
        public int compare(Object obj1,Object obj2);
        public boolean equals(Object obj);
    }
    compare(Object obj1,Object obj2)
            |=> returns -ve value, if obj1 has to come before obj2
            |=> returns +ve value, if obj1 has to come after  obj2
            |=> returns 0 value, if both obj1 and obj2 are equal
```

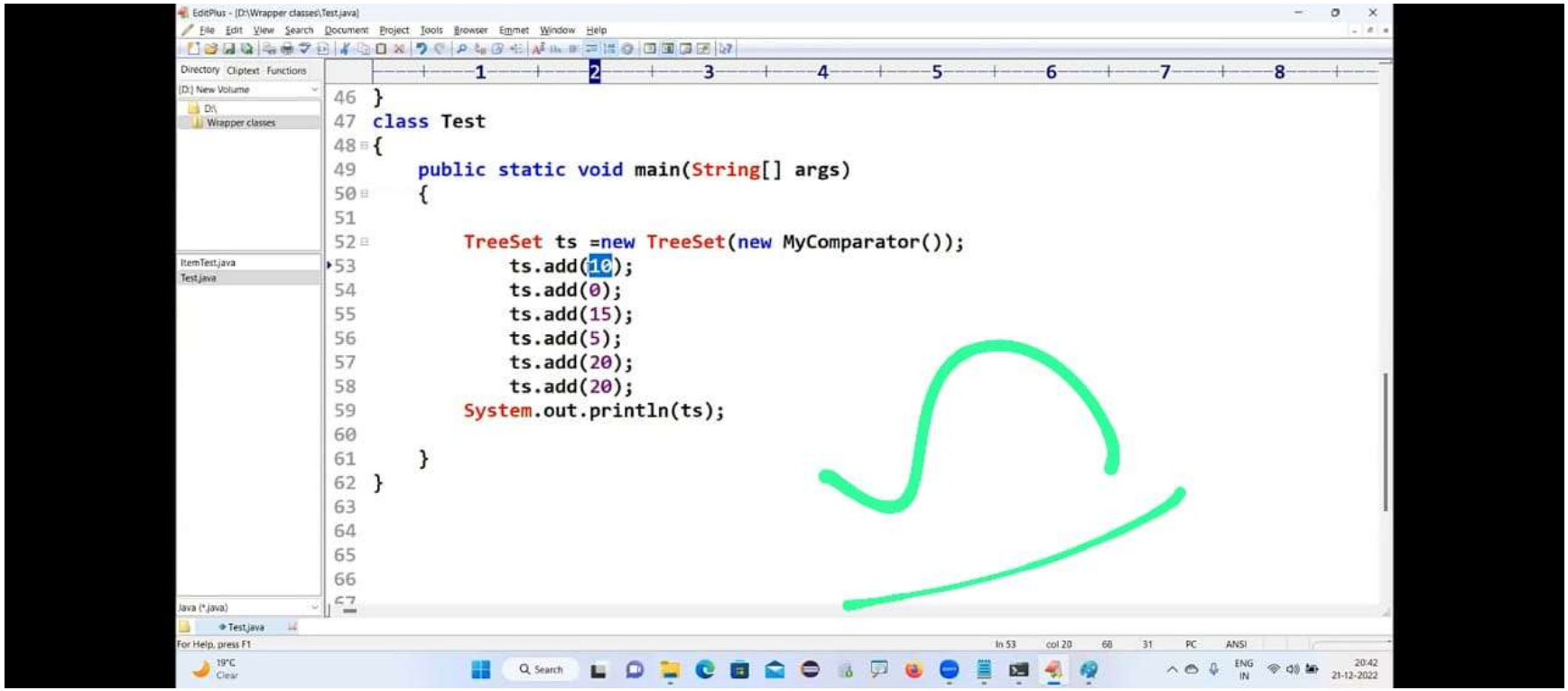

Command Prompt
D:\Wrapper classes>javap java.util.TreeSet
Compiled from "TreeSet.java"
public class java.util.TreeSet<E> extends java.util.AbstractSet<E> implements java.util.NavigableSet<E>, java.lang.Cloneable,
java.io.Serializable {
java.util.TreeSet(java.util.NavigableMap<E, java.lang.Object>);
public 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);
21°C
Clear
Search
ENG
IN
20:20
21-12-2022



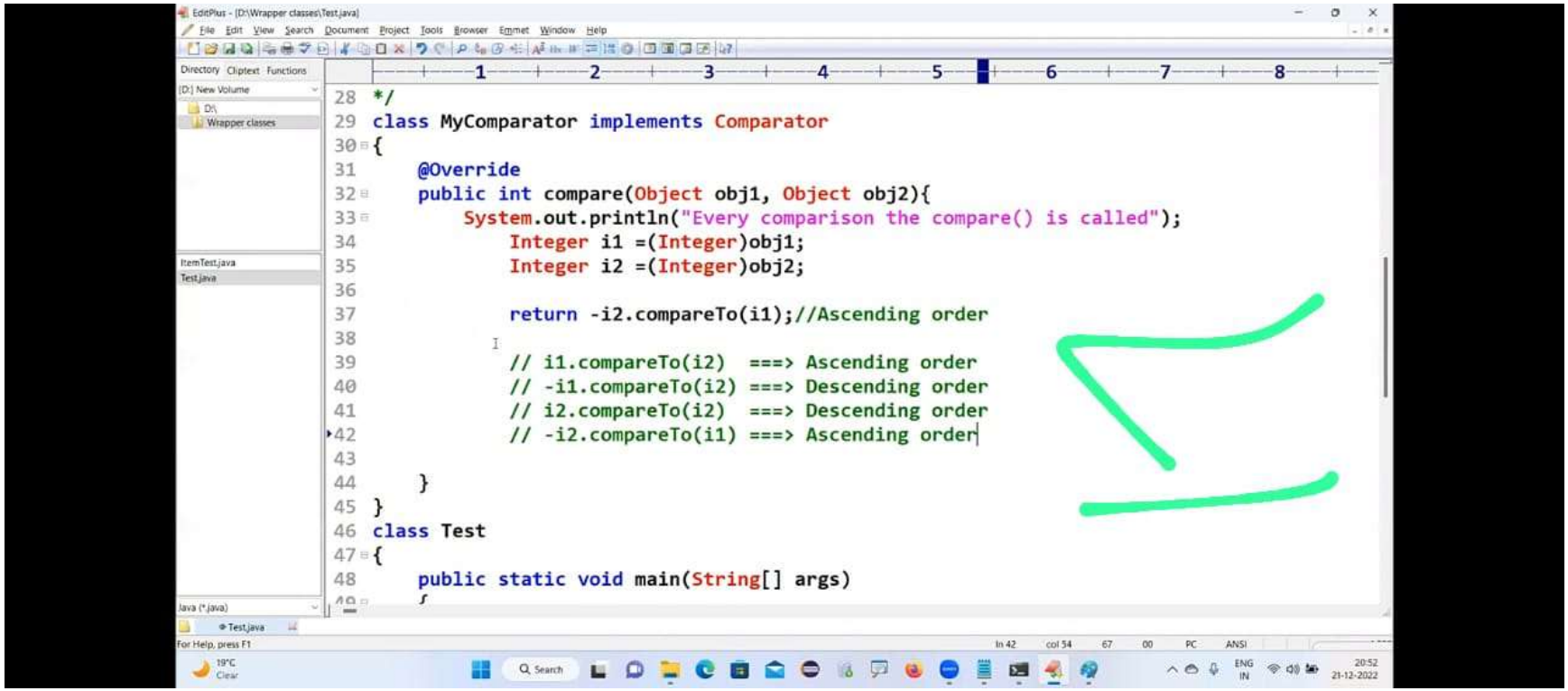

```
*/
class MyComparator implements Comparator
{
    @override
    public int compare(Object obj1, Object obj2){
            //logic coming soooonnnnn.......
            Integer i1 =(Integer)obj1;
            Integer i2 =(Integer)obj2;

            if (i1<i2)
                return +1;
            else if(i1>i2)
                return -1;
            else
                return 0;

    }
}
class Test
```

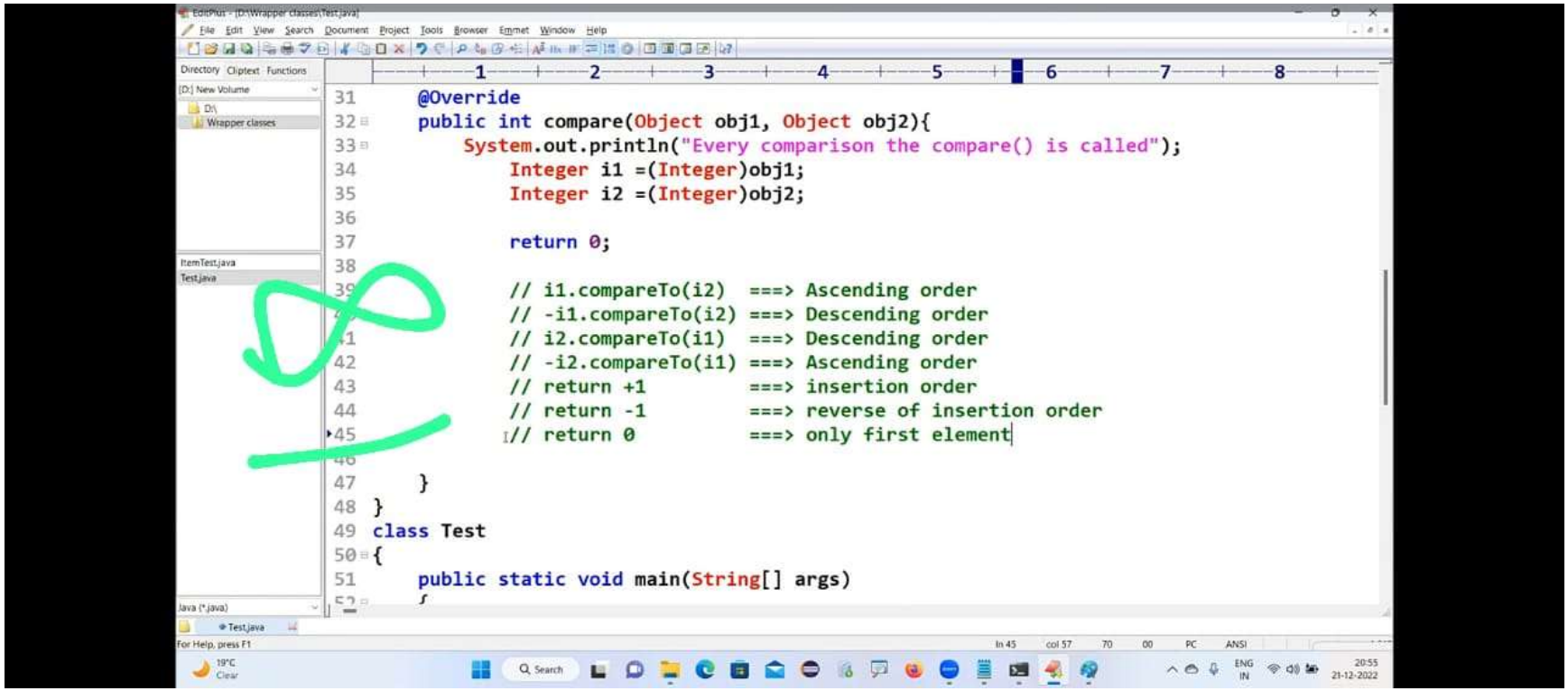

21_12_2022_comparable vs comparator_images - Paint
File
View
ts.add("B"); "B".compareTo("A"); "B".compareTo(Z")
"B".compareTo("L"); "B".compareTo("K");
System.out.println(ts); [A B K L Z]
K
-ve
B
TreeSet ts =new TreeSet(new MyCompartor());
ts.add(10);
ts.add(0); compare(0,10)
ts.add(15); compare(15,10)
ts.add(5); compare(5,10)
compare(5,0)
ts.add(20); compare(20,10)
compare(20,15)
ts.add(20); compare(20,10)
compare(20,15)
compare(20,20)
20 15 10 5 0
10
-ve
+ve
15
0
-ve
-ve
20
5
compare(Object obj1,Object obj2)
|=> returns -ve value, if obj1 has to come before obj2
|=> returns +ve value, if obj1 has to come after obj2
|=> returns 0 value, if both obj1 and obj2 are equal
if (i1<i2)
return +1;
else if(i1>i2)
return -1;
else
return 0;

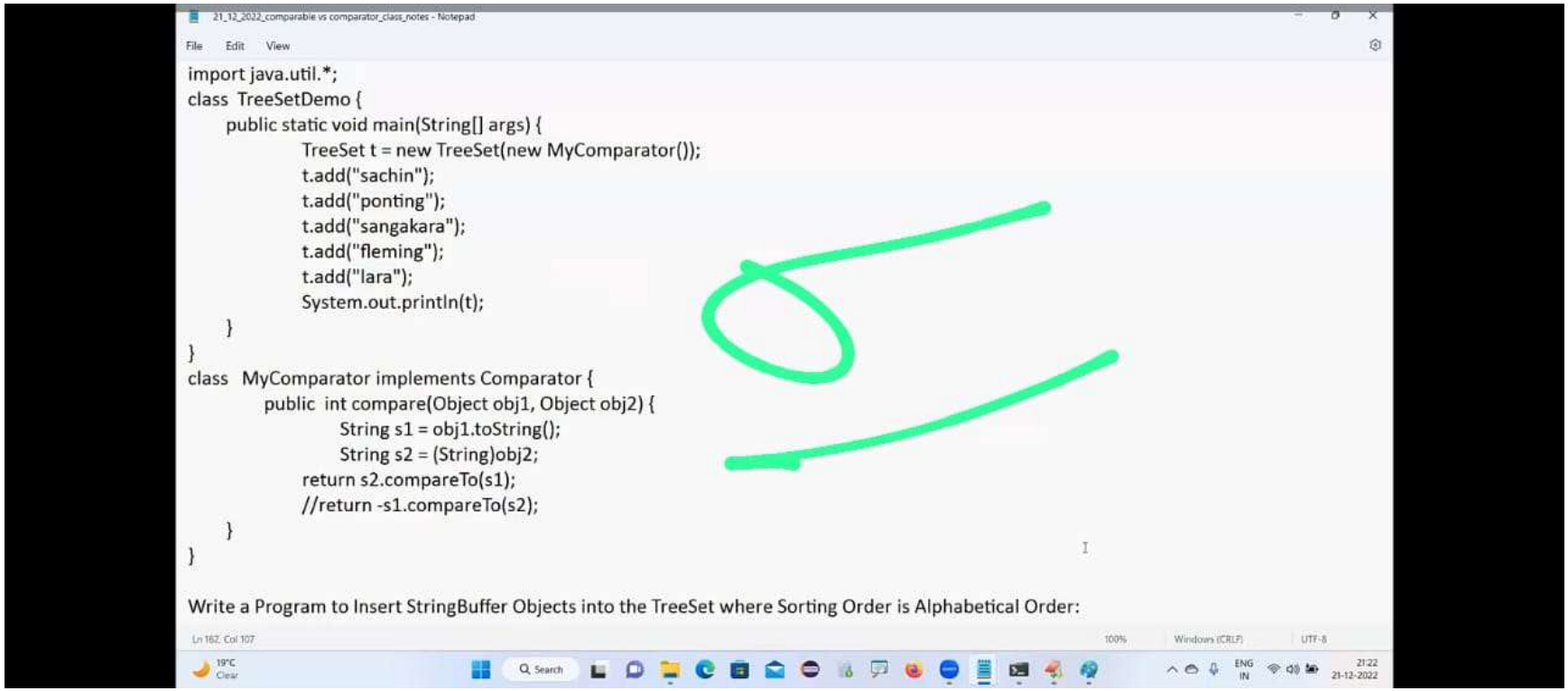

Command Prompt
D:\Wrapper classes>java Test
Every comparison the compare() is called
Every comparison the compare() is called
Every comparison the compare() is called
Every comparison the compare() is called
Every comparison the compare() is called
Every comparison the compare() is called
Every comparison the compare() is called
Every comparison the compare() is called
Every comparison the compare() is called
Every comparison the compare() is called
[20, 15, 10, 5, 0]
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
Every comparison the compare() is called
Every comparison the compare() is called
Every comparison the compare() is called
Every comparison the compare() is called
Every comparison the compare() is called
Every comparison the compare() is called
Every comparison the compare() is called
Every comparison the compare() is called
Every comparison the compare() is called
Every comparison the compare() is called
[0, 5, 10, 15, 20]
D:\Wrapper classes>
19°C
Clear
Search
ENG
IN
20:42
21-12-2022

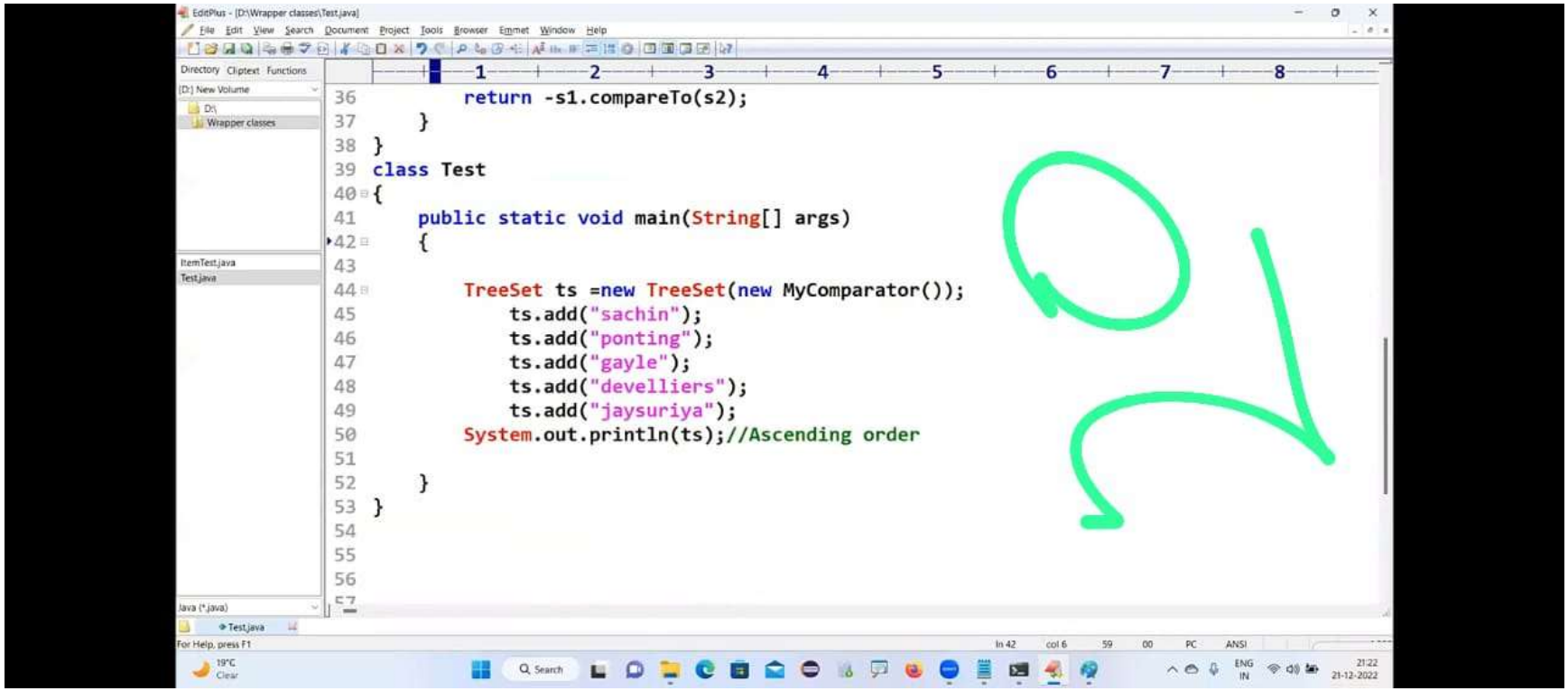



```
}
class Test
{
    public static void main(String[] args)
    {

        TreeSet ts =new TreeSet(new MyComparator());
            ts.add(10);
            ts.add(0);
            ts.add(15);
            ts.add(5);
            ts.add(20);
            ts.add(20);
        System.out.println(ts);

    }
}
```

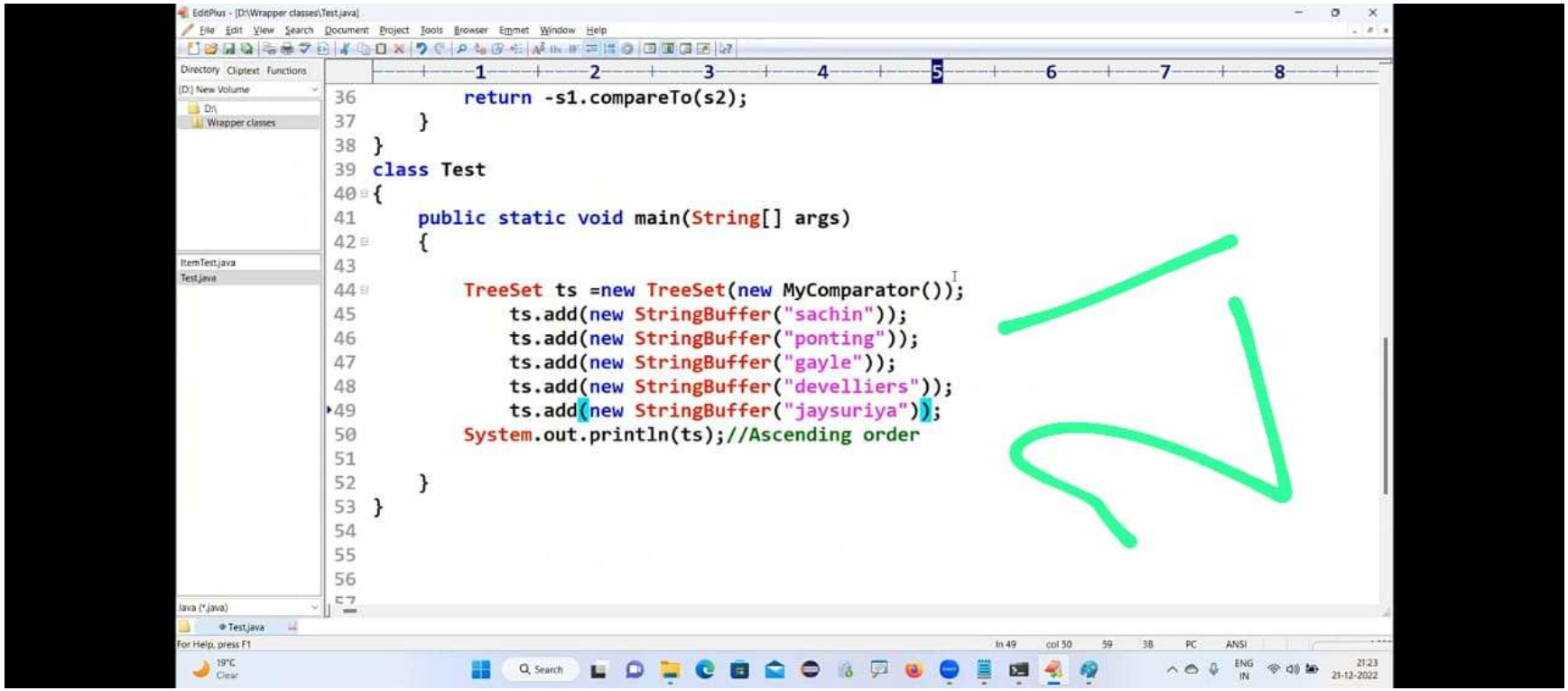

EditPlus - [D:\Wrapper classes\Test.java]
File Edit View Search Document Project Tools Browser Emmet Window Help
Directory Cliptext Functions
[D:] New Volume
D:\
Wrapper classes
ItemTest.java
Test.java
*/
class MyComparator implements Comparator
{
@Override
public int compare(Object obj1, Object obj2){
System.out.println("Every comparison the compare() is called");
Integer i1 =(Integer)obj1;
Integer i2 =(Integer)obj2;
return -i2.compareTo(i1);//Ascending order
// i1.compareTo(i2) ===> Ascending order
// -i1.compareTo(i2) ===> Descending order
// i2.compareTo(i2) ===> Descending order
// -i2.compareTo(i1) ===> Ascending order
}
}
class Test
{
public static void main(String[] args)
Test.java
For Help, press F1
ln 42 col 54 67 00 PC ANSI
19°C Clear
Search
ENG IN
20:52 21-12-2022

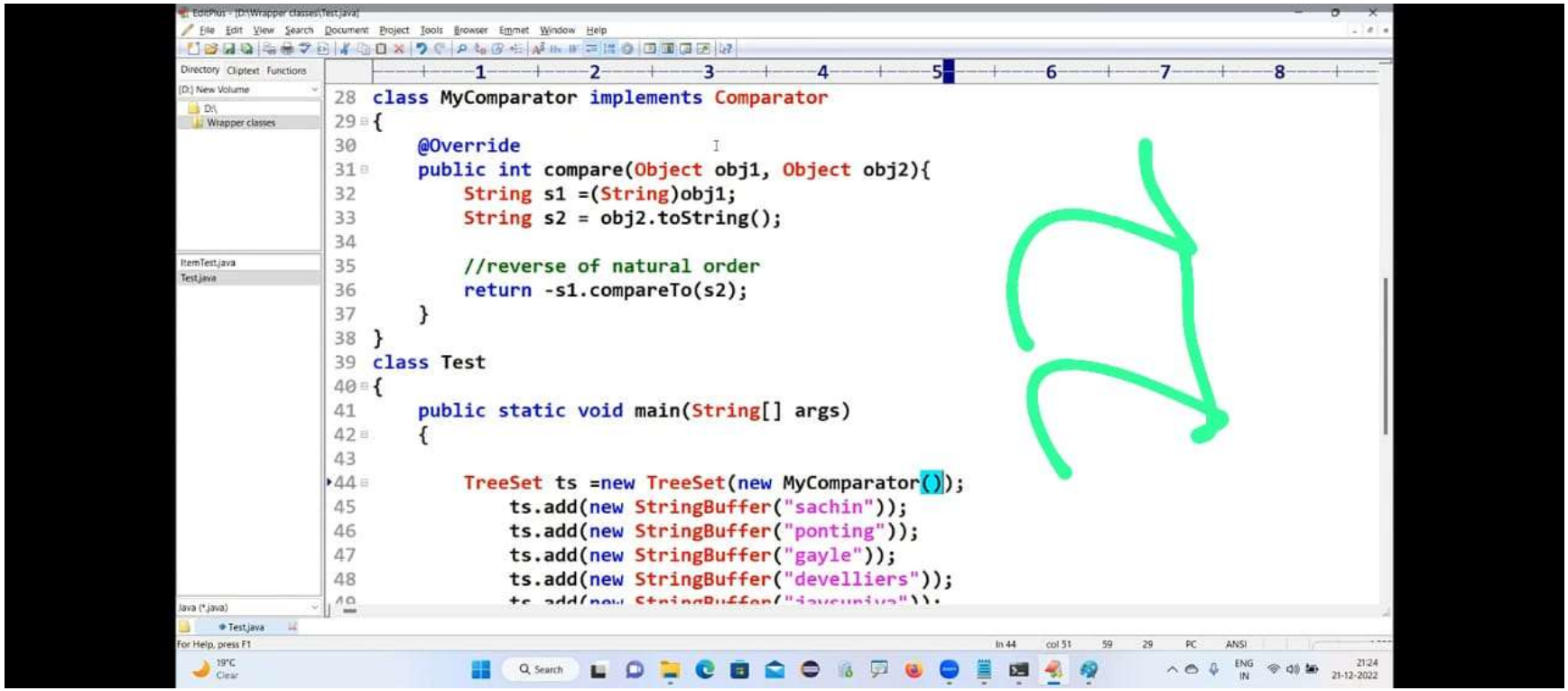

@Override
public int compare(Object obj1, Object obj2){
System.out.println("Every comparison the compare() is called");
Integer i1 =(Integer)obj1;
Integer i2 =(Integer)obj2;
return 0;
// i1.compareTo(i2) ===> Ascending order
// -i1.compareTo(i2) ===> Descending order
// i2.compareTo(i1) ===> Descending order
// -i2.compareTo(i1) ===> Ascending order
// return +1 ===> insertion order
// return -1 ===> reverse of insertion order
// return 0 ===> only first element
}
}
class Test
{
public static void main(String[] args)


21_12_2022_comparable vs comparator_class_notes - Notepad
File Edit View
import java.util.*;
class TreeSetDemo {
public static void main(String[] args) {
TreeSet t = new TreeSet(new MyComparator());
t.add("sachin");
t.add("ponting");
t.add("sangakara");
t.add("fleming");
t.add("lara");
System.out.println(t);
}
}
class MyComparator implements Comparator {
public int compare(Object obj1, Object obj2) {
String s1 = obj1.toString();
String s2 = (String)obj2;
return s2.compareTo(s1);
//return -s1.compareTo(s2);
}
}
Write a Program to Insert StringBuffer Objects into the TreeSet where Sorting Order is Alphabetical Order:
Ln 162, Col 107
100%
Windows (CRLF)
UTF-8
19°C
Clear
Search
ENG
IN
21:22
21-12-2022

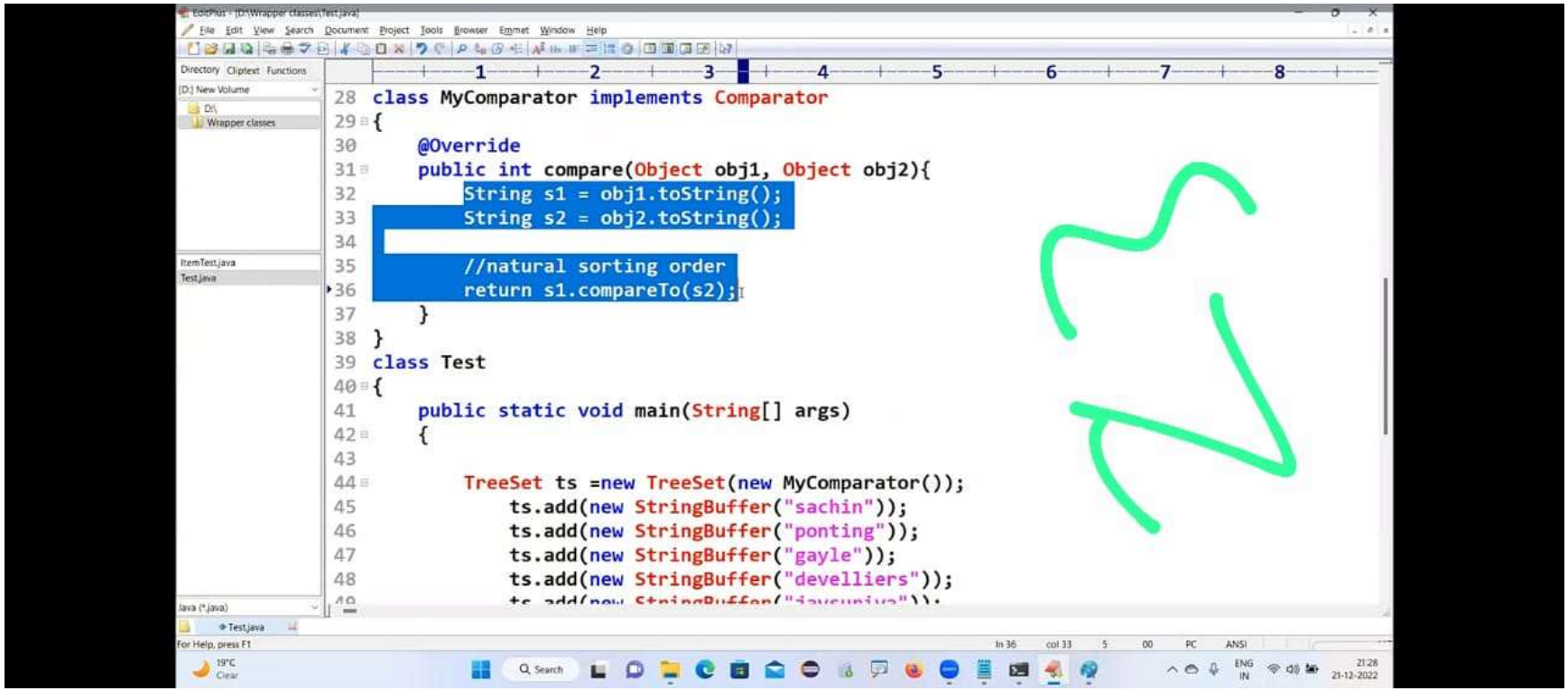


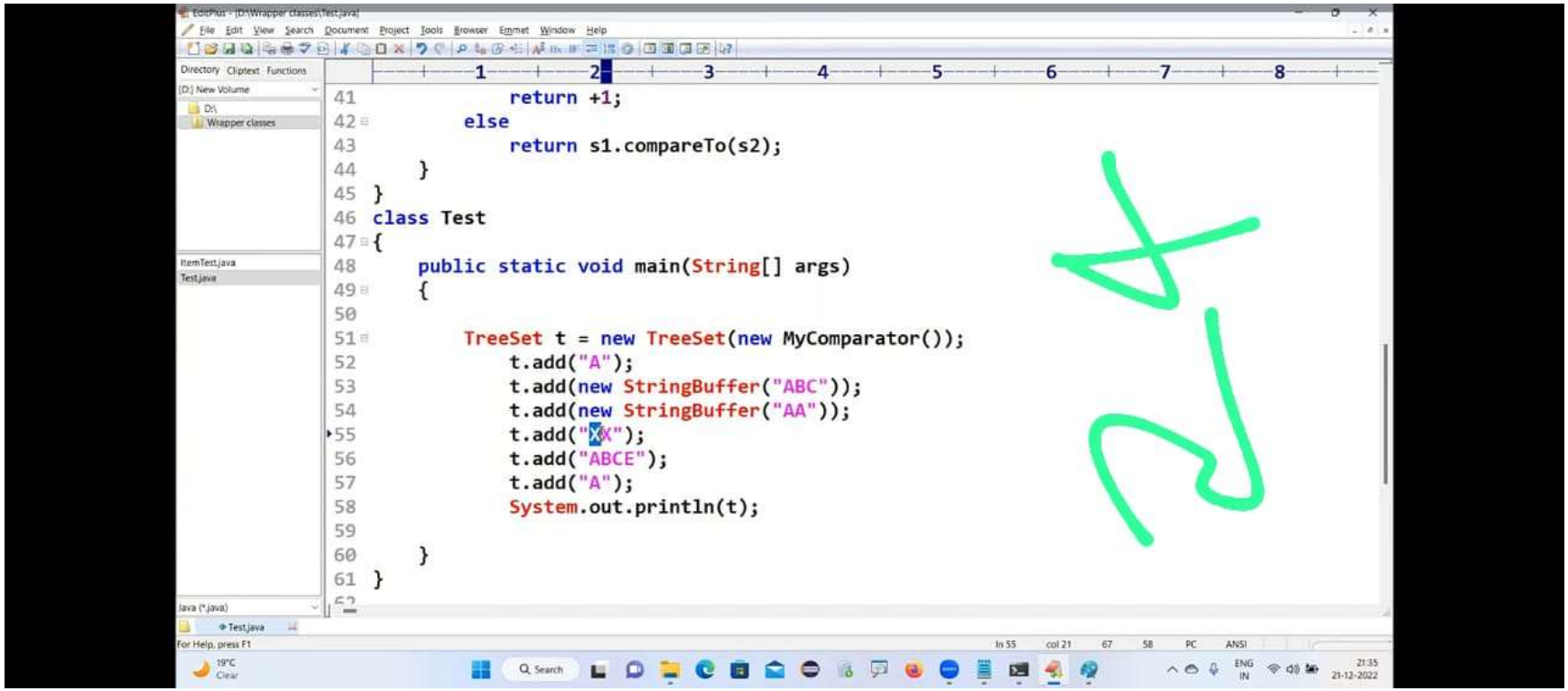

```
36          return -s1.compareTo(s2);
37      }
38  }
39  class Test
40  {
41      public static void main(String[] args)
42      {
43
44          TreeSet ts =new TreeSet(new MyComparator());
45              ts.add("sachin");
46              ts.add("ponting");
47              ts.add("gayle");
48              ts.add("develliers");
49              ts.add("jaysuriya");
50          System.out.println(ts);//Ascending order
51
52      }
53  }
54
55
56
```


EditPlus - [D:\Wrapper classes\Test.java]
File Edit View Search Document Project Tools Browser Emmet Window Help
Directory Cliptext Functions
D:\
Wrapper classes
ItemTest.java
Test.java
return -s1.compareTo(s2);
}
}
class Test
{
public static void main(String[] args)
{
TreeSet ts =new TreeSet(new MyComparator());
ts.add(new StringBuffer("sachin"));
ts.add(new StringBuffer("ponting"));
ts.add(new StringBuffer("gayle"));
ts.add(new StringBuffer("develliers"));
ts.add(new StringBuffer("jaysuriya"));
System.out.println(ts);//Ascending order
}
}
For Help, press F1
ln 49 col 50 59 3B PC ANSI
19°C Clear
Search
ENG IN
21:23 21-12-2022

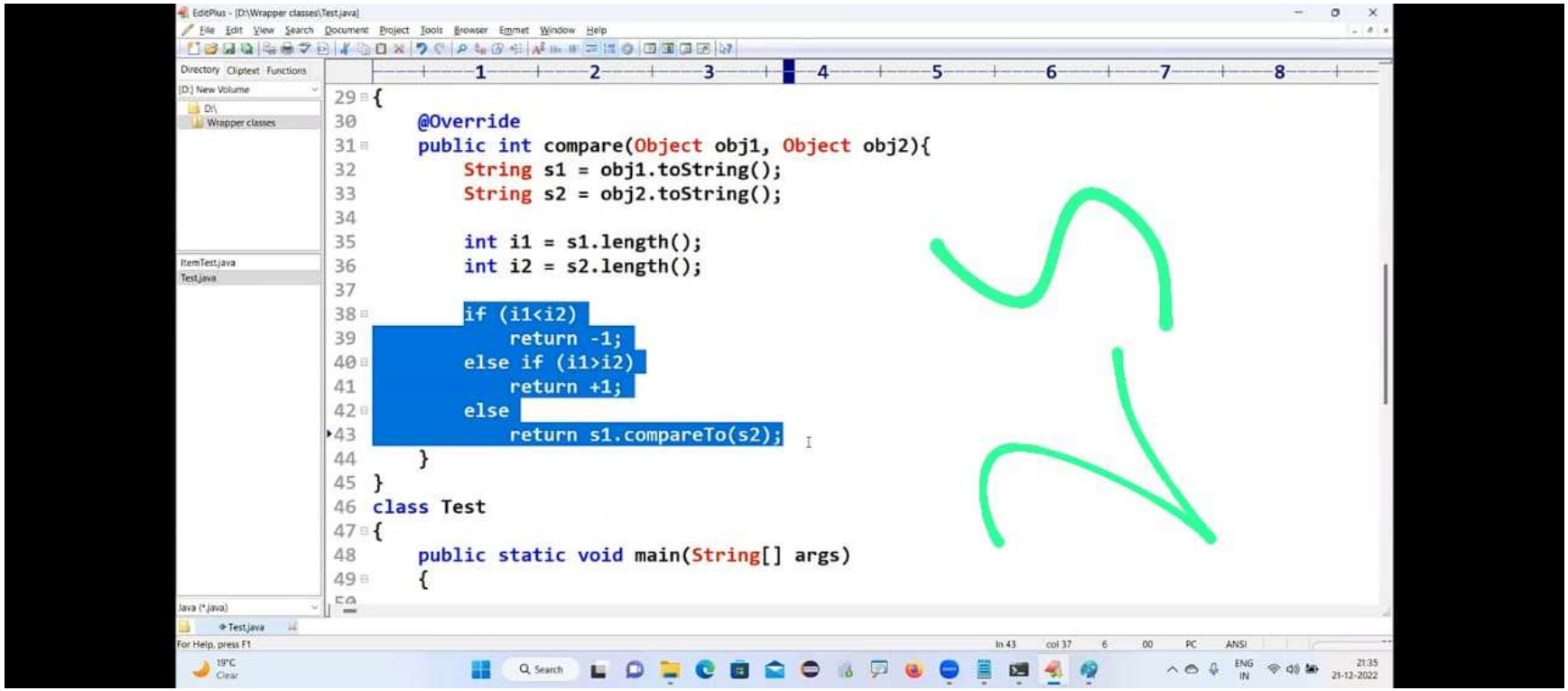



```
28 class MyComparator implements Comparator
29 {
30     @Override
31     public int compare(Object obj1, Object obj2){
32         String s1 =(String)obj1;
33         String s2 = obj2.toString();
34
35         //reverse of natural order
36         return -s1.compareTo(s2);
37     }
38 }
39 class Test
40 {
41     public static void main(String[] args)
42     {
43
44         TreeSet ts =new TreeSet(new MyComparator());
45             ts.add(new StringBuffer("sachin"));
46             ts.add(new StringBuffer("ponting"));
47             ts.add(new StringBuffer("gayle"));
48             ts.add(new StringBuffer("develliers"));
```

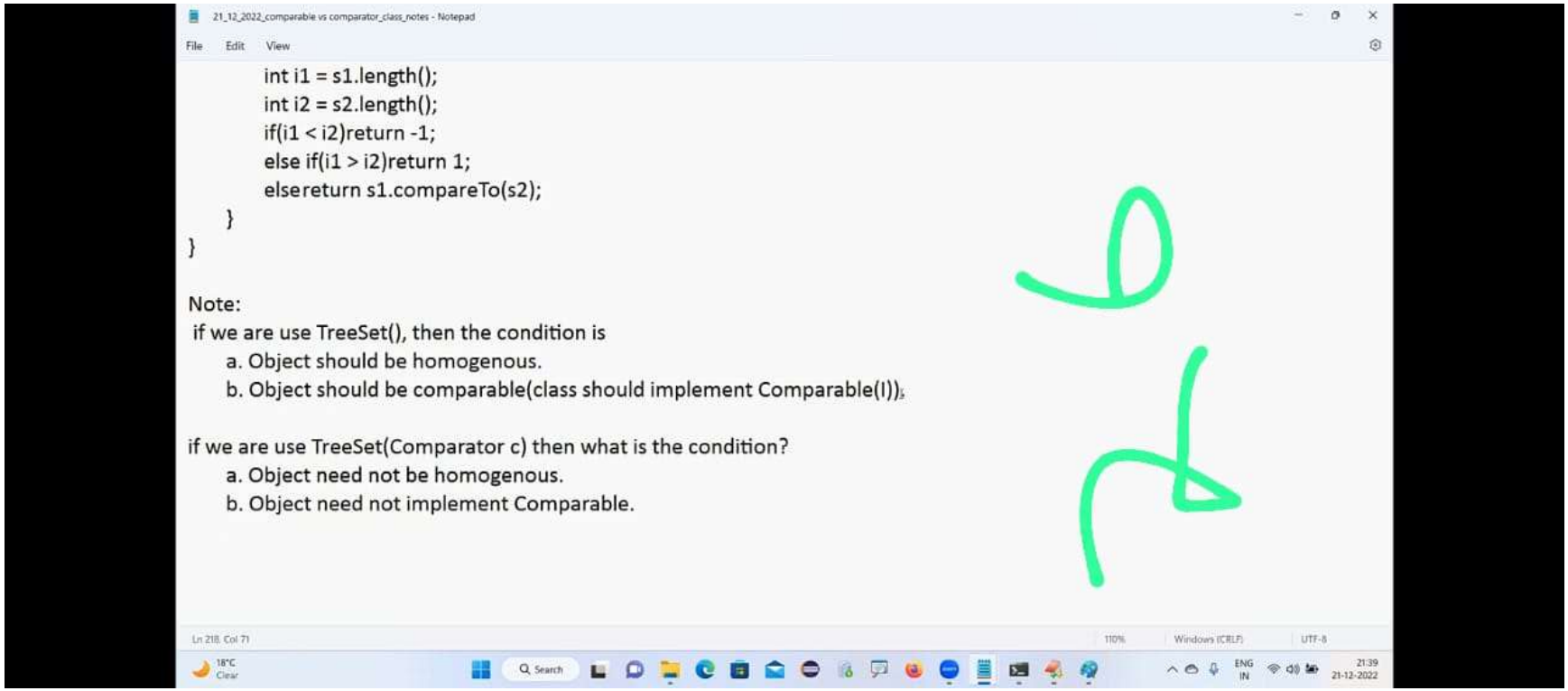

class MyComparator implements Comparator
{
@Override
public int compare(Object obj1, Object obj2){
String s1 = obj1.toString();
String s2 = obj2.toString();
//natural sorting order
return s1.compareTo(s2);
}
}
class Test
{
public static void main(String[] args)
{
TreeSet ts =new TreeSet(new MyComparator());
ts.add(new StringBuffer("sachin"));
ts.add(new StringBuffer("ponting"));
ts.add(new StringBuffer("gayle"));
ts.add(new StringBuffer("develliers"));

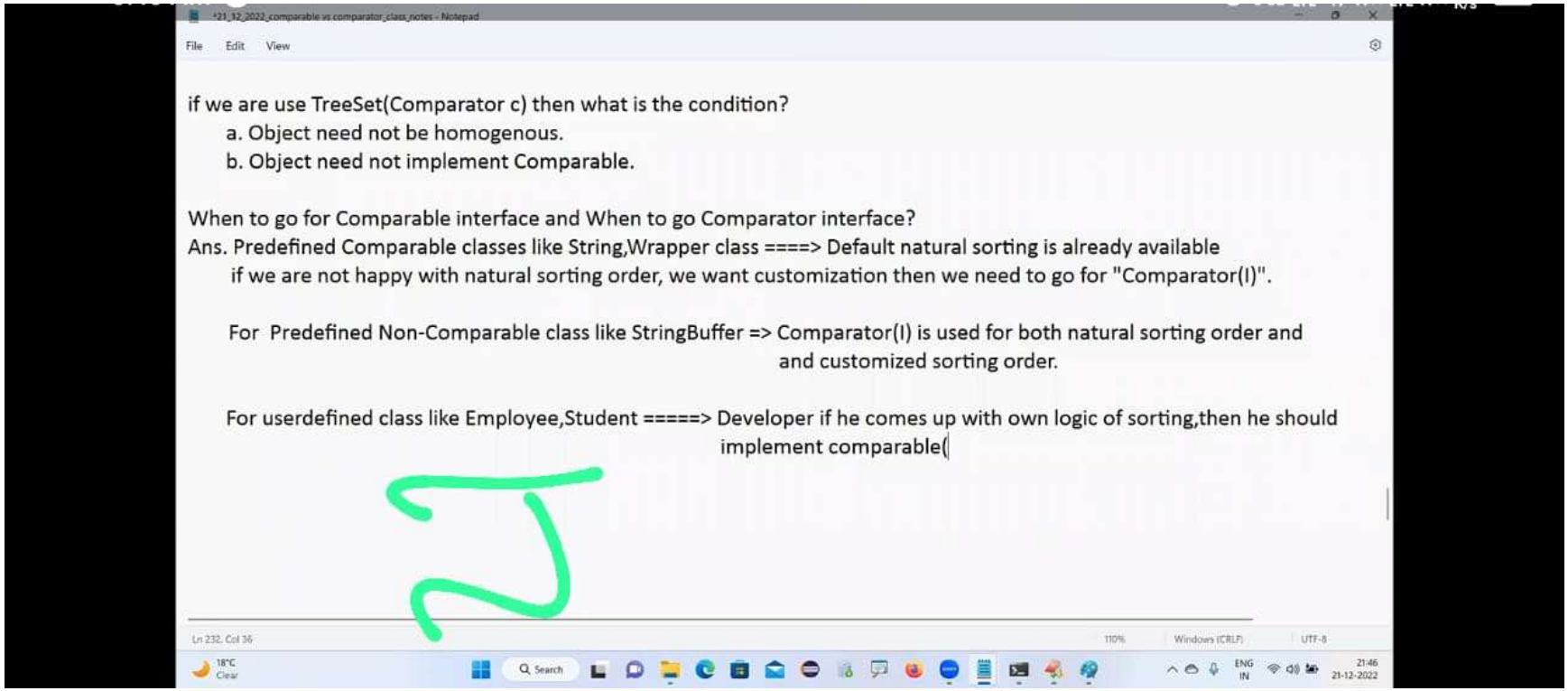



```
            return +1;
        else
            return s1.compareTo(s2);
    }
}
class Test
{
    public static void main(String[] args)
    {

        TreeSet t = new TreeSet(new MyComparator());
            t.add("A");
            t.add(new StringBuffer("ABC"));
            t.add(new StringBuffer("AA"));
            t.add("XX");
            t.add("ABCE");
            t.add("A");
            System.out.println(t);

    }
}
```




```
{
    @Override
    public int compare(Object obj1, Object obj2){
        String s1 = obj1.toString();
        String s2 = obj2.toString();

        int i1 = s1.length();
        int i2 = s2.length();

        if (i1<i2)
            return -1;
        else if (i1>i2)
            return +1;
        else
            return s1.compareTo(s2);
    }
}
class Test
{
    public static void main(String[] args)
    {
```


21_12_2022_comparable vs comparator_class_notes - Notepad
File Edit View
int i1 = s1.length();
int i2 = s2.length();
if(i1 < i2)return -1;
else if(i1 > i2)return 1;
elsereturn s1.compareTo(s2);
}
}
Note:
if we are use TreeSet(), then the condition is
a. Object should be homogenous.
b. Object should be comparable(class should implement Comparable(I)).
if we are use TreeSet(Comparator c) then what is the condition?
a. Object need not be homogenous.
b. Object need not implement Comparable.
Ln 218, Col 71
110%
Windows (CRLF)
UTF-8
18°C
Clear
Search
ENG
IN
21:39
21-12-2022


*21_12_2022_comparable vs comparator_class_notes - Notepad
File Edit View
if we are use TreeSet(Comparator c) then what is the condition?
a. Object need not be homogenous.
b. Object need not implement Comparable.
When to go for Comparable interface and When to go Comparator interface?
Ans. Predefined Comparable classes like String,Wrapper class ====> Default natural sorting is already available
if we are not happy with natural sorting order, we want customization then we need to go for "Comparator(I)".
For Predefined Non-Comparable class like StringBuffer => Comparator(I) is used for both natural sorting order and
and customized sorting order.
For userdefined class like Employee,Student =====> Developer if he comes up with own logic of sorting,then he should
implement comparable(
Ln 232, Col 36
110%
Windows (CRLF)
UTF-8
18°C
Clear
Search
ENG
IN
21:46
21-12-2022

21_12_2022_comparable vs comparator_class_notes - Notepad

File Edit View

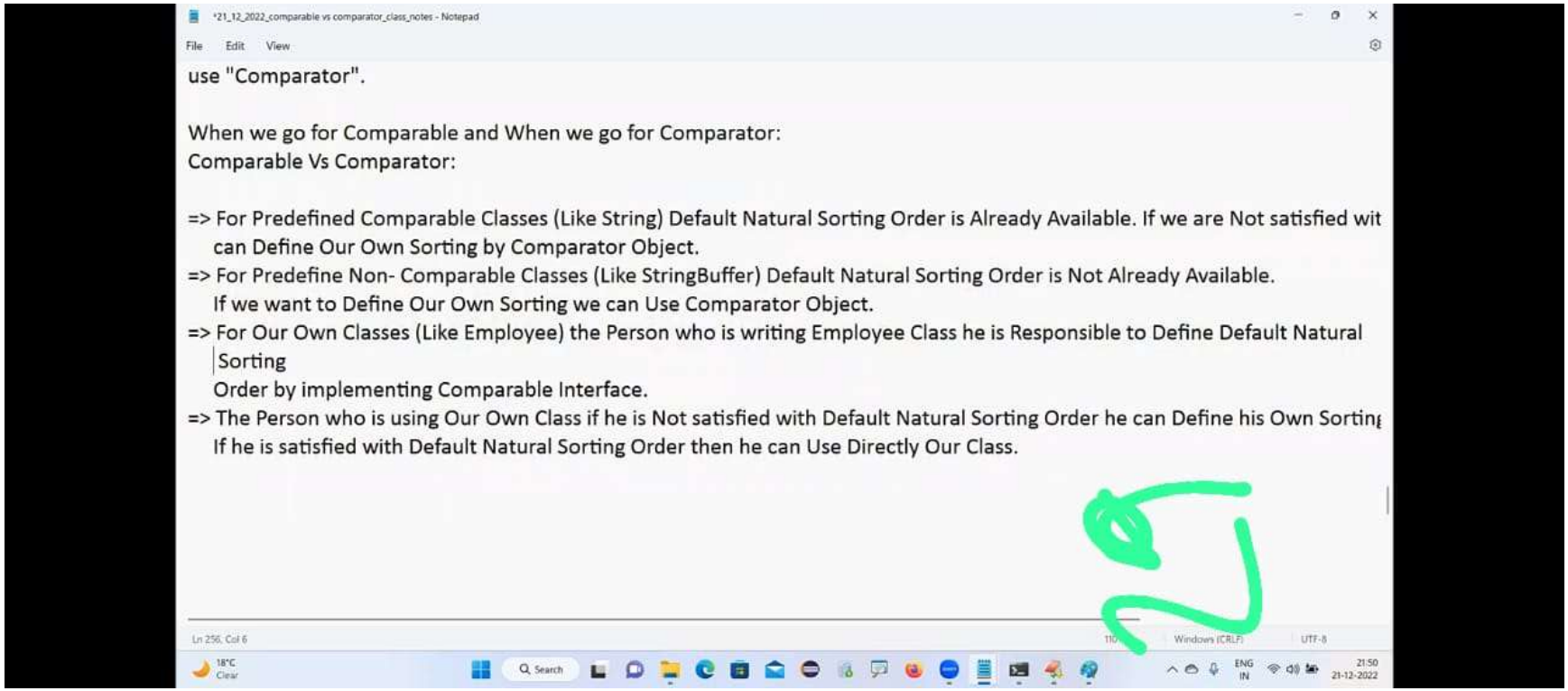
For Predefined Non-Comparable class like StringBuffer => Comparator(I) is used for both natural sorting order and and customized sorting order.

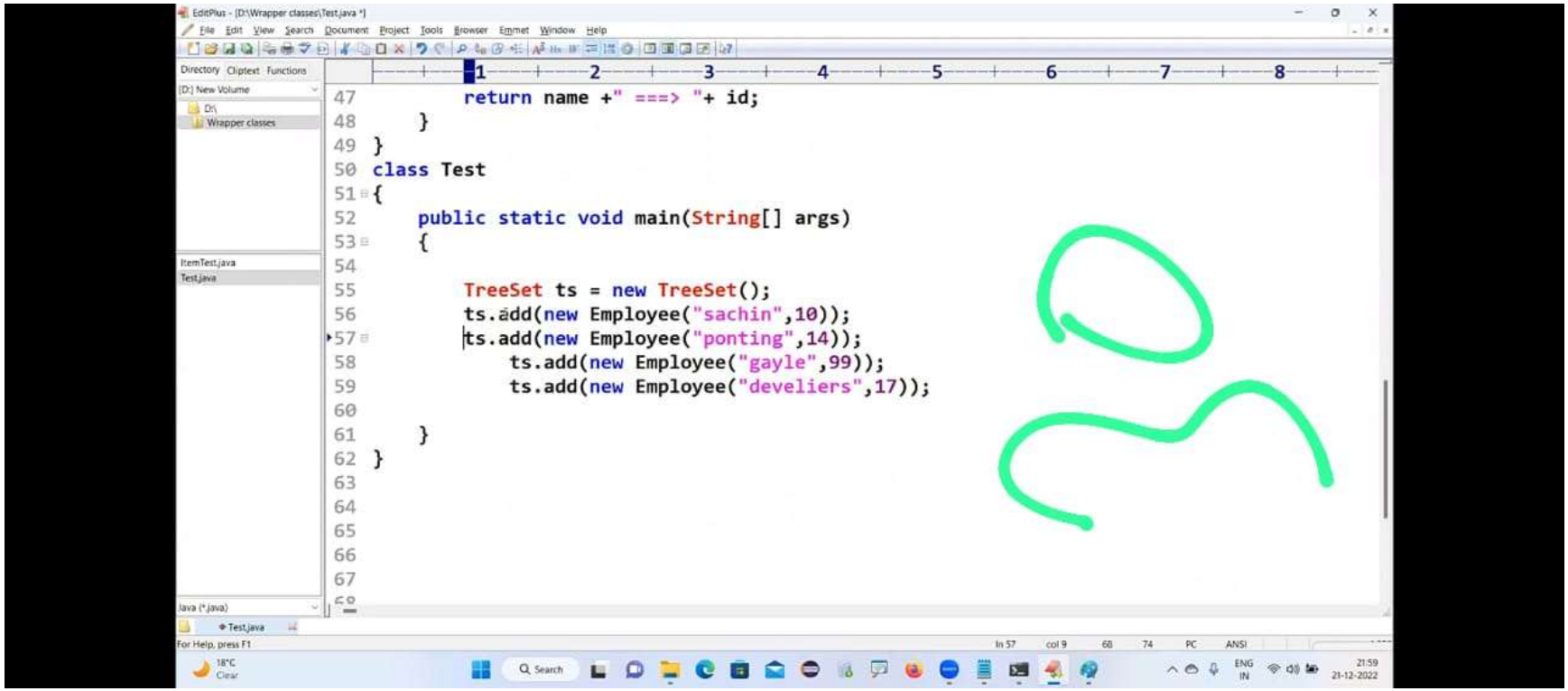
For userdefined class like Employee,Student =====> Developer if he comes up with own logic of sorting,then he should implement Comparable(I) and give it as a ready made logic.

```
class Employee implements Comparable
{
    int id;
    String name;
    int age;

    public int compareTo(Object obj){
        //sorting is done based on "id"
        ;;;;;
    }
}
```

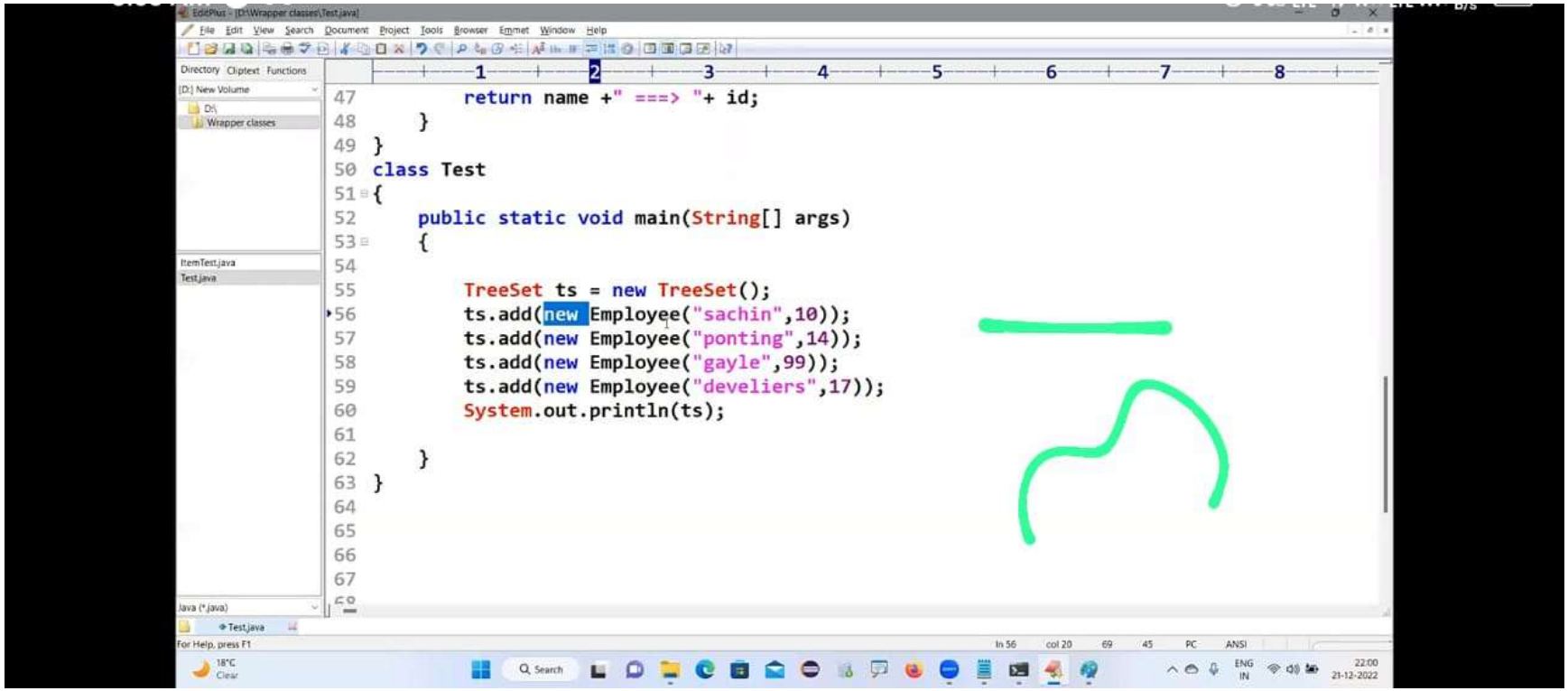
Ln 237, Col 13 110% Windows (CRLF) UTF-8

18°C Clear Search ENG IN 21:49 21-12-2022

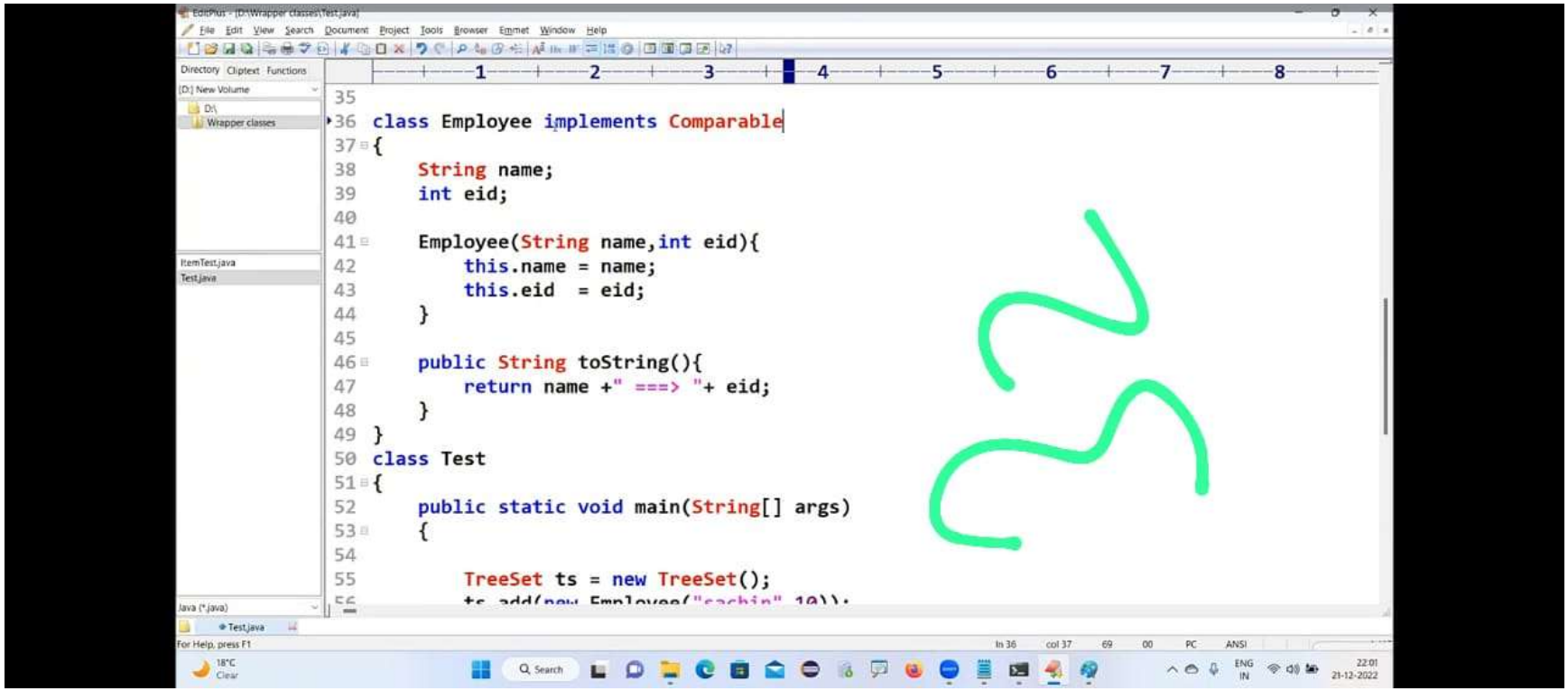

*21_12_2022_comparable vs comparator_class_notes - Notepad
File Edit View
use "Comparator".
When we go for Comparable and When we go for Comparator:
Comparable Vs Comparator:
=> For Predefined Comparable Classes (Like String) Default Natural Sorting Order is Already Available. If we are Not satisfied wit
can Define Our Own Sorting by Comparator Object.
=> For Predefine Non- Comparable Classes (Like StringBuffer) Default Natural Sorting Order is Not Already Available.
If we want to Define Our Own Sorting we can Use Comparator Object.
=> For Our Own Classes (Like Employee) the Person who is writing Employee Class he is Responsible to Define Default Natural
Sorting
Order by implementing Comparable Interface.
=> The Person who is using Our Own Class if he is Not satisfied with Default Natural Sorting Order he can Define his Own Sortin
If he is satisfied with Default Natural Sorting Order then he can Use Directly Our Class.
Ln 256, Col 6
Windows (CRLF)
UTF-8
18°C
Clear
Search
ENG
IN
21:50
21-12-2022

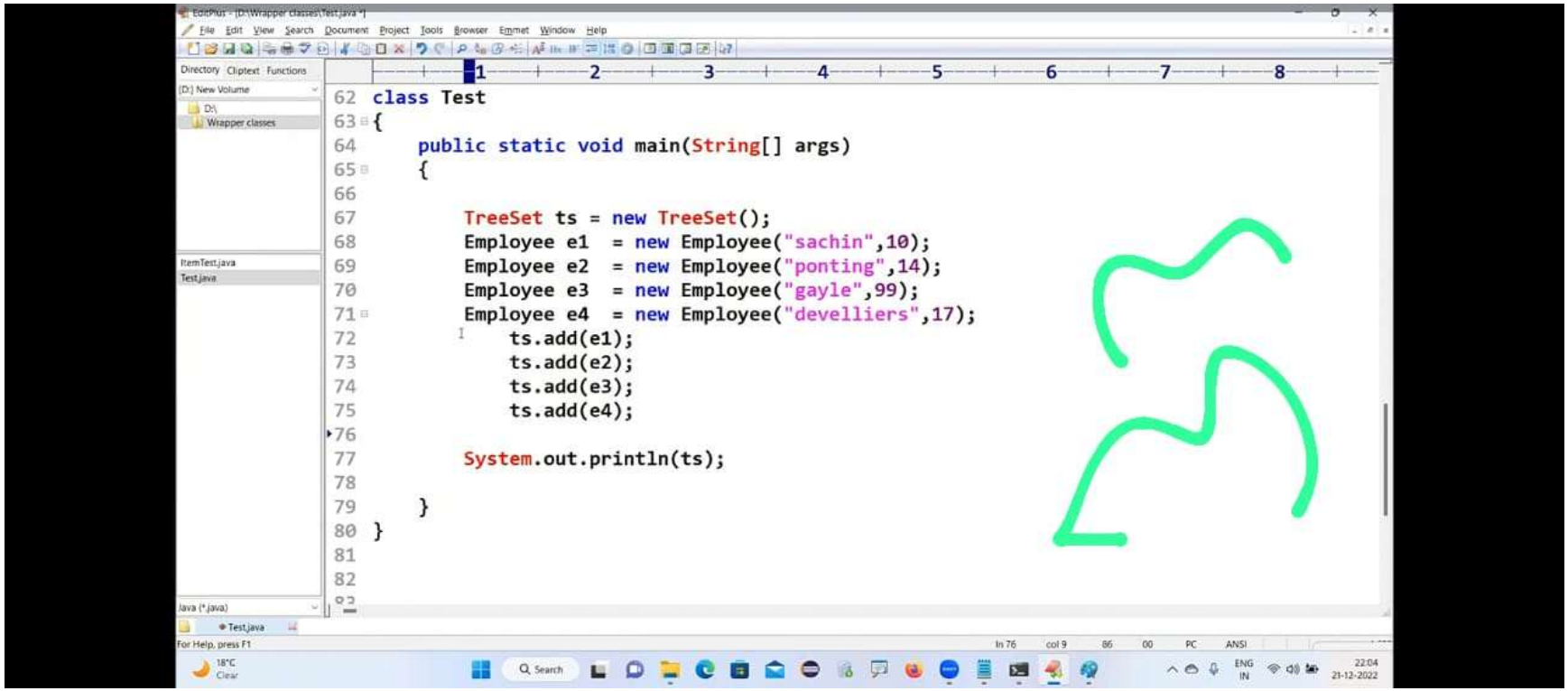


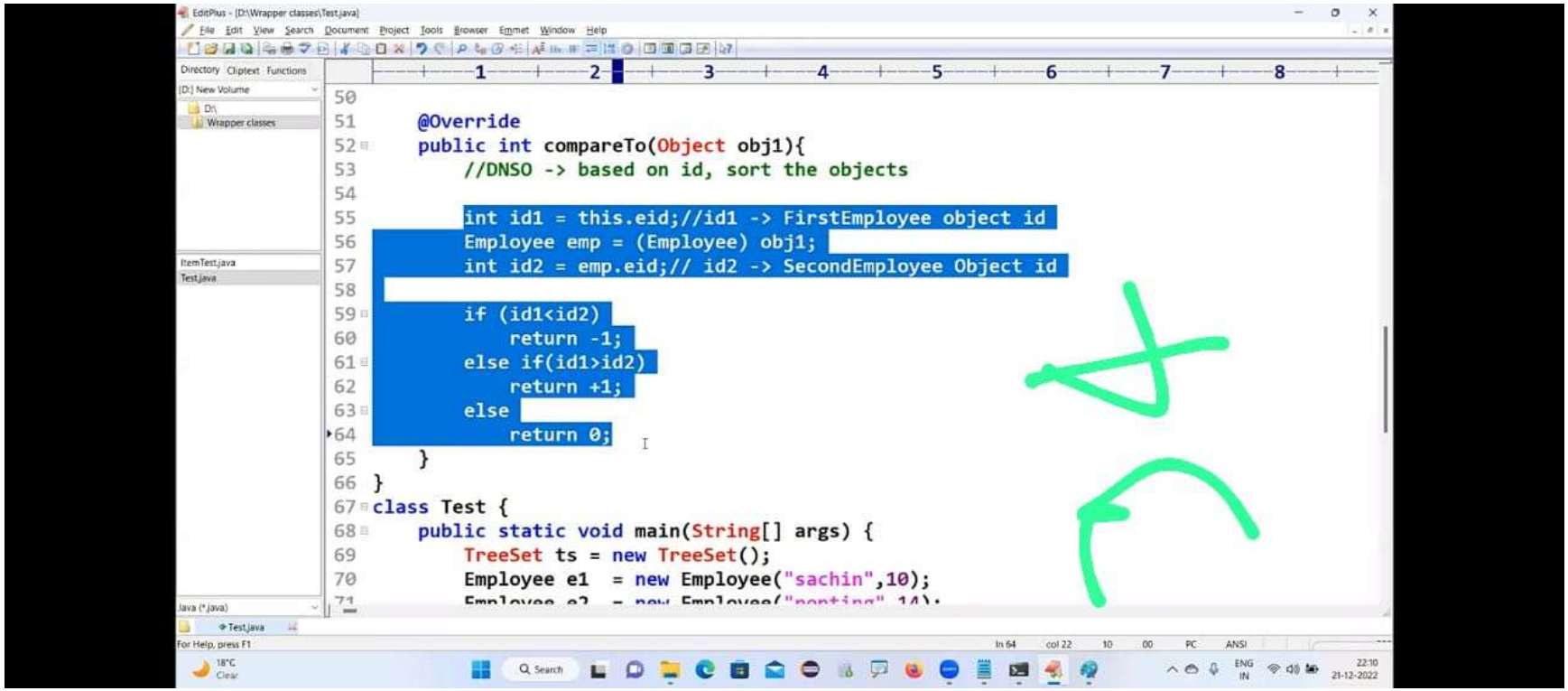

```
        return name +" ===> "+ id;
    }
}
class Test
{
    public static void main(String[] args)
    {

        TreeSet ts = new TreeSet();
        ts.add(new Employee("sachin",10));
        ts.add(new Employee("ponting",14));
            ts.add(new Employee("gayle",99));
            ts.add(new Employee("develiers",17));

    }
}
```



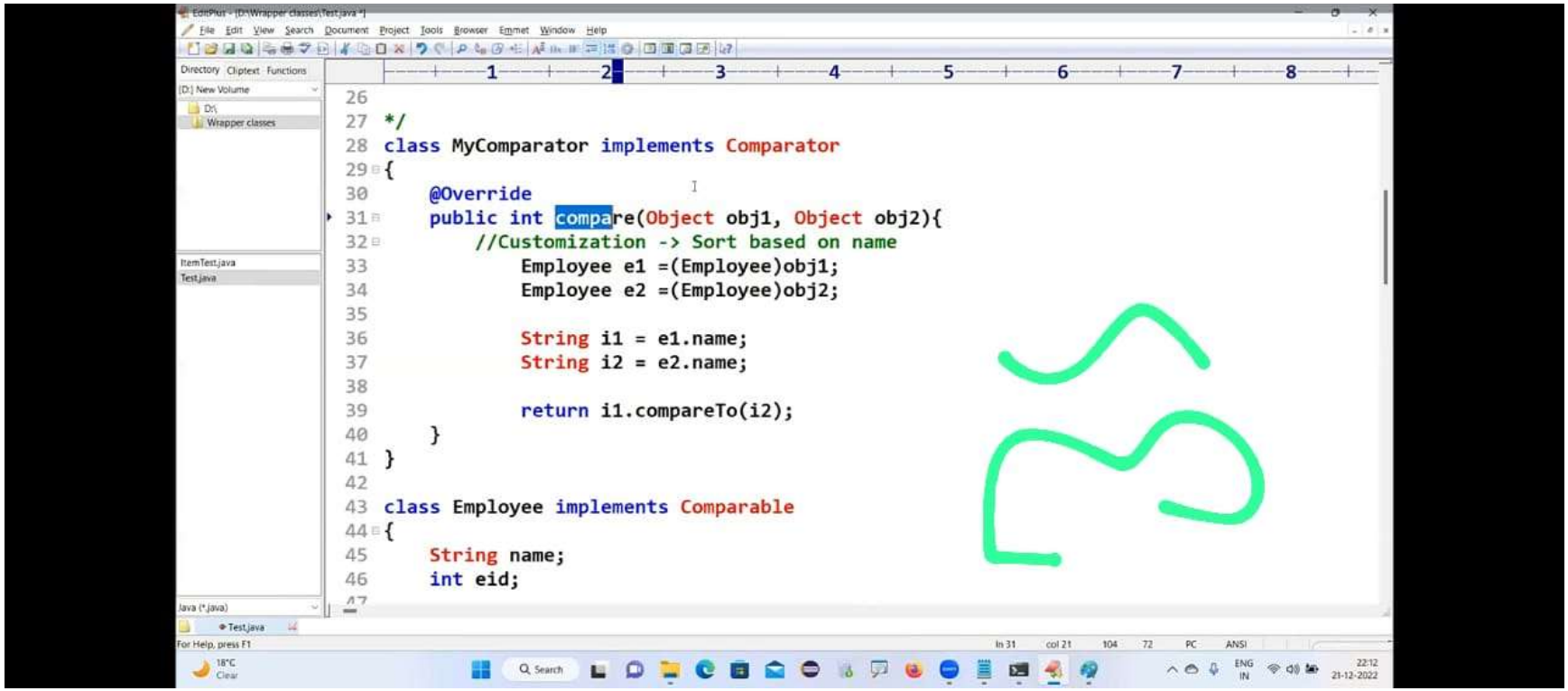

```
        return name +" ===> "+ id;
    }
}
class Test
{
    public static void main(String[] args)
    {

        TreeSet ts = new TreeSet();
        ts.add(new Employee("sachin",10));
        ts.add(new Employee("ponting",14));
        ts.add(new Employee("gayle",99));
        ts.add(new Employee("develiers",17));
        System.out.println(ts);

    }
}
```

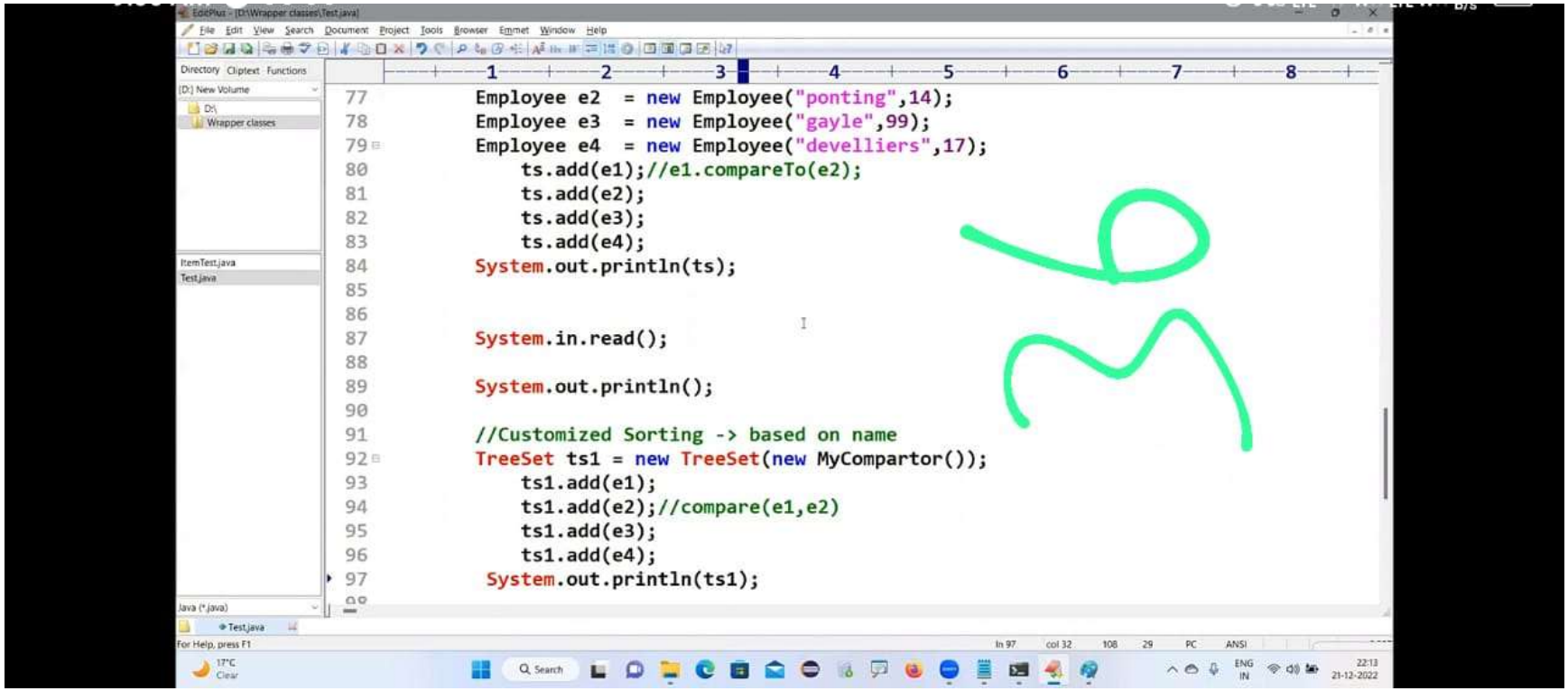

class Employee implements Comparable
{
String name;
int eid;
Employee(String name,int eid){
this.name = name;
this.eid = eid;
}
public String toString(){
return name +" ===> "+ eid;
}
}
class Test
{
public static void main(String[] args)
{
TreeSet ts = new TreeSet();


class Test
{
public static void main(String[] args)
{
TreeSet ts = new TreeSet();
Employee e1 = new Employee("sachin",10);
Employee e2 = new Employee("ponting",14);
Employee e3 = new Employee("gayle",99);
Employee e4 = new Employee("develliers",17);
ts.add(e1);
ts.add(e2);
ts.add(e3);
ts.add(e4);
System.out.println(ts);
}
}

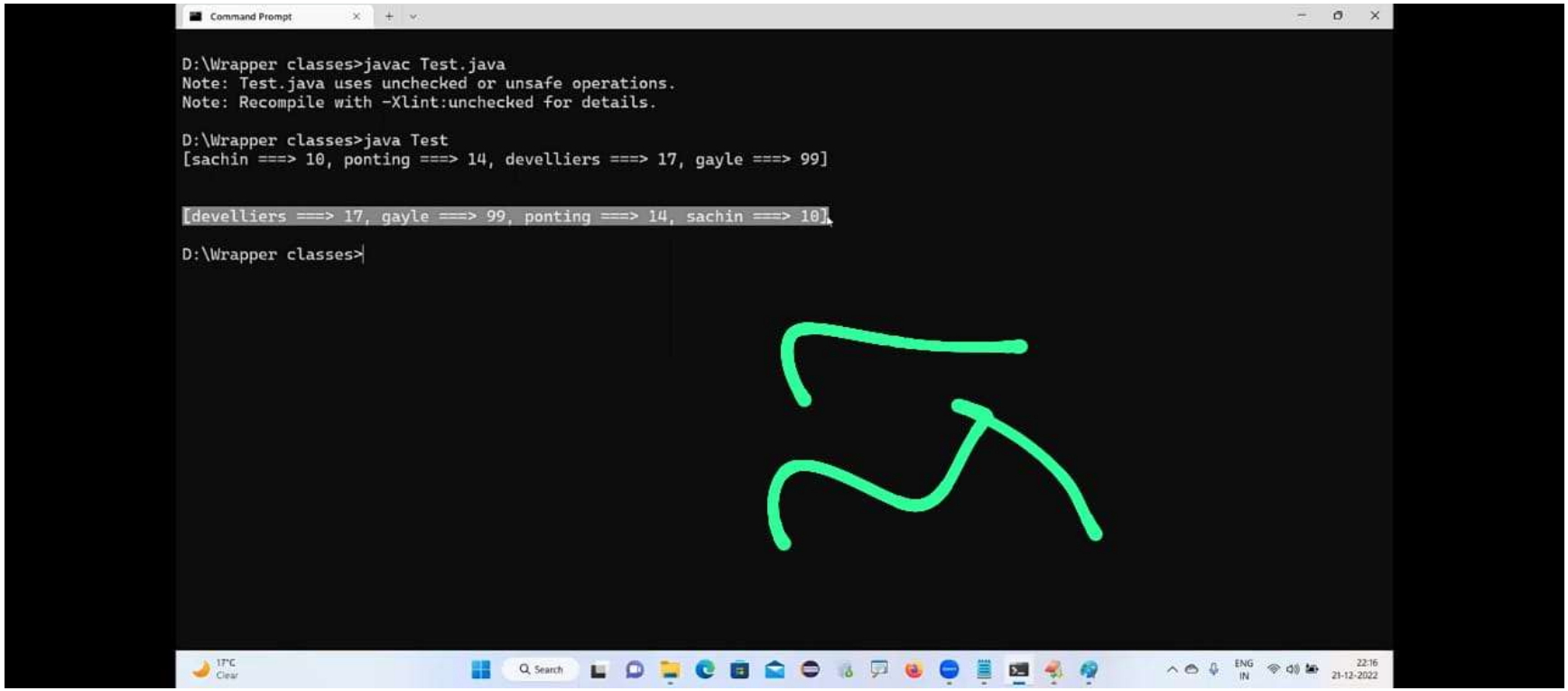


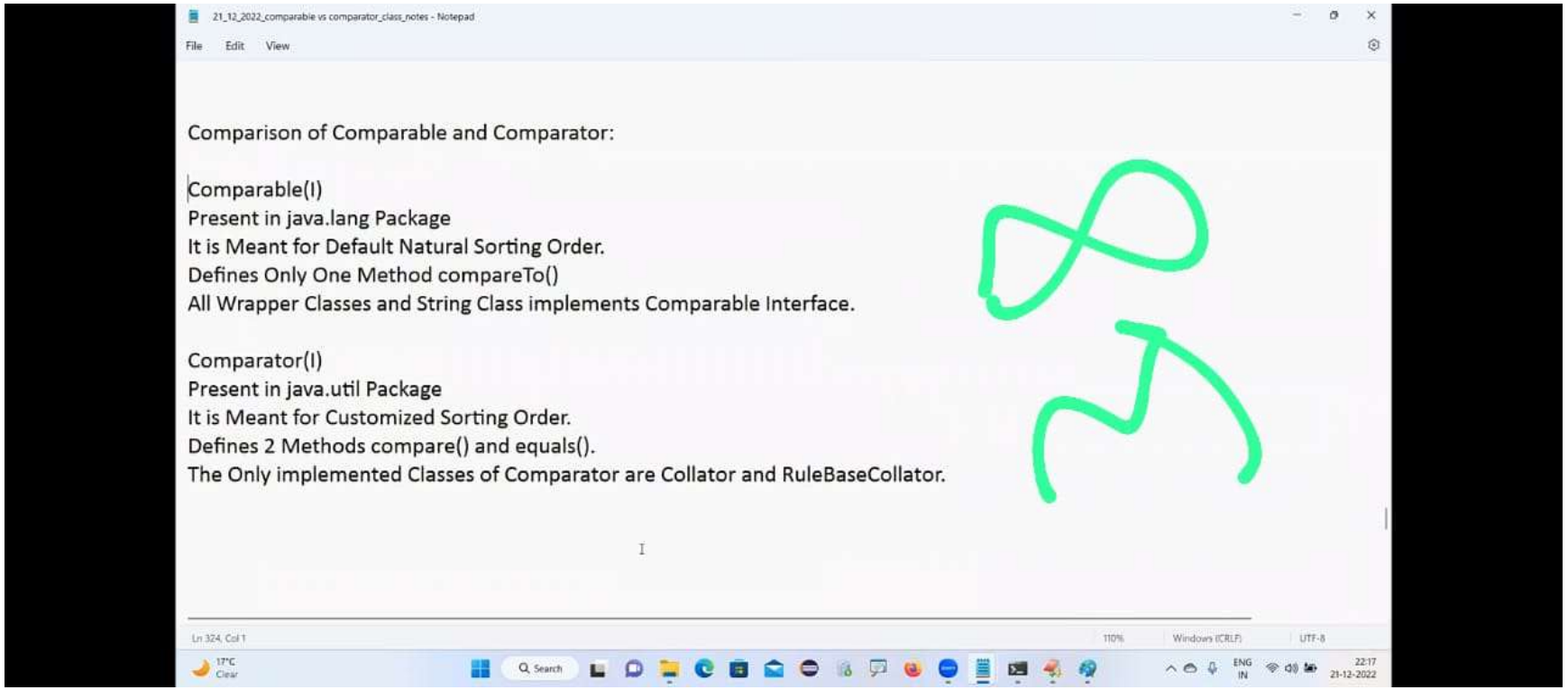

```
50
51      @Override
52      public int compareTo(Object obj1){
53          //DNSO -> based on id, sort the objects
54
55          int id1 = this.eid;//id1 -> FirstEmployee object id
56          Employee emp = (Employee) obj1;
57          int id2 = emp.eid;// id2 -> SecondEmployee Object id
58
59          if (id1<id2)
60              return -1;
61          else if(id1>id2)
62              return +1;
63          else
64              return 0;
65      }
66  }
67  class Test {
68      public static void main(String[] args) {
69          TreeSet ts = new TreeSet();
70          Employee e1  = new Employee("sachin",10);
71          Employee e2  = new Employee("ponting",14);
```

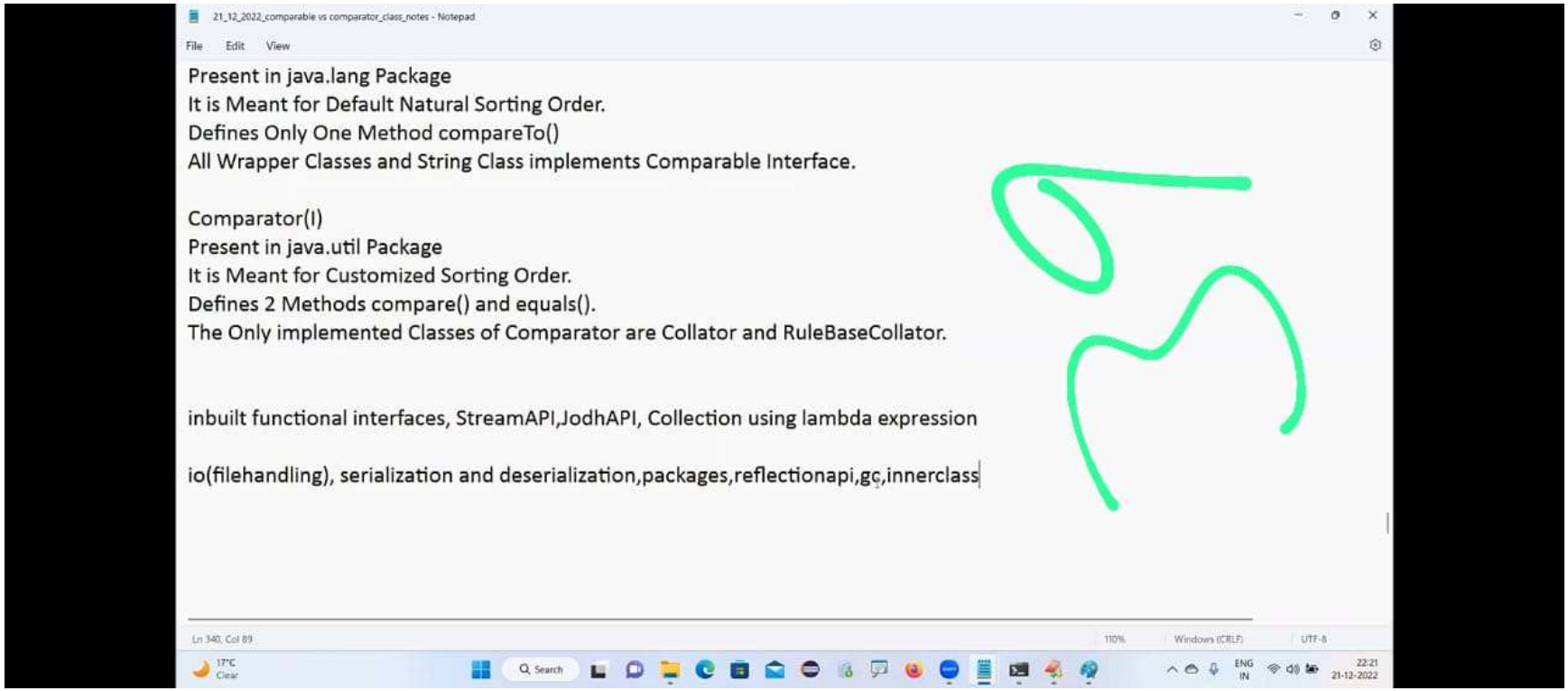

*/
class MyComparator implements Comparator
{
@Override
public int compare(Object obj1, Object obj2){
//Customization -> Sort based on name
Employee e1 =(Employee)obj1;
Employee e2 =(Employee)obj2;
String i1 = e1.name;
String i2 = e2.name;
return i1.compareTo(i2);
}
}
class Employee implements Comparable
{
String name;
int eid;




```
77          Employee e2  = new Employee("ponting",14);
78          Employee e3  = new Employee("gayle",99);
79          Employee e4  = new Employee("develliers",17);
80              ts.add(e1);//e1.compareTo(e2);
81              ts.add(e2);
82              ts.add(e3);
83              ts.add(e4);
84          System.out.println(ts);
85
86
87          System.in.read();
88
89          System.out.println();
90
91          //Customized Sorting -> based on name
92          TreeSet ts1 = new TreeSet(new MyCompartor());
93              ts1.add(e1);
94              ts1.add(e2);//compare(e1,e2)
95              ts1.add(e3);
96              ts1.add(e4);
97           System.out.println(ts1);
```


Command Prompt
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, ponting ===> 14, devellers ===> 17, gayle ===> 99]
[devellers ===> 17, gayle ===> 99, ponting ===> 14, sachin ===> 10]
D:\Wrapper classes>
17°C
Clear
Search
ENG
IN
22:16
21-12-2022


21_12_2022_comparable vs comparator_class_notes - Notepad
File Edit View
Comparison of Comparable and Comparator:
Comparable(I)
Present in java.lang Package
It is Meant for Default Natural Sorting Order.
Defines Only One Method compareTo()
All Wrapper Classes and String Class implements Comparable Interface.
Comparator(I)
Present in java.util Package
It is Meant for Customized Sorting Order.
Defines 2 Methods compare() and equals().
The Only implemented Classes of Comparator are Collator and RuleBaseCollator.
Ln 324, Col 1
110%
Windows (CRLF)
UTF-8
17°C
Clear
Search
ENG
IN
22:17
21-12-2022


21_12_2022_comparable vs comparator_class_notes - Notepad
File Edit View
Present in java.lang Package
It is Meant for Default Natural Sorting Order.
Defines Only One Method compareTo()
All Wrapper Classes and String Class implements Comparable Interface.
Comparator(I)
Present in java.util Package
It is Meant for Customized Sorting Order.
Defines 2 Methods compare() and equals().
The Only implemented Classes of Comparator are Collator and RuleBaseCollator.
inbuilt functional interfaces, StreamAPI,JodhAPI, Collection using lambda expression
io(filehandling), serialization and deserialization,packages,reflectionapi,gc,innerclass
Ln 340, Col 89
110%
Windows (CRLF)
UTF-8
17°C
Clear
Search
ENG
IN
22:21
21-12-2022

21_12_2022_snippets_session_notes - Notepad

File Edit View

Question
Consider below code:

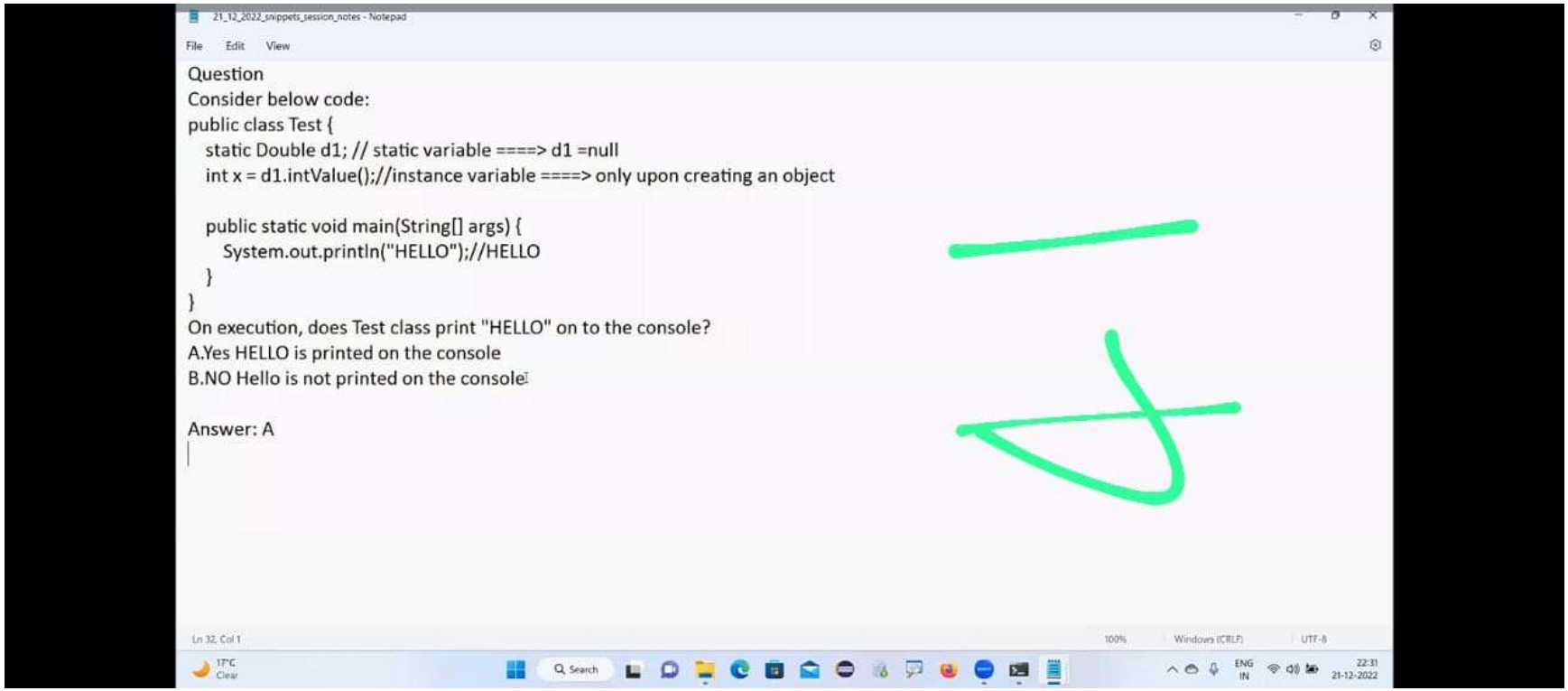
```
public class Test {
   static Double d1;  // d1 =null
   static int x = d1.intValue();// null.intValue() ---> NullPointerException

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

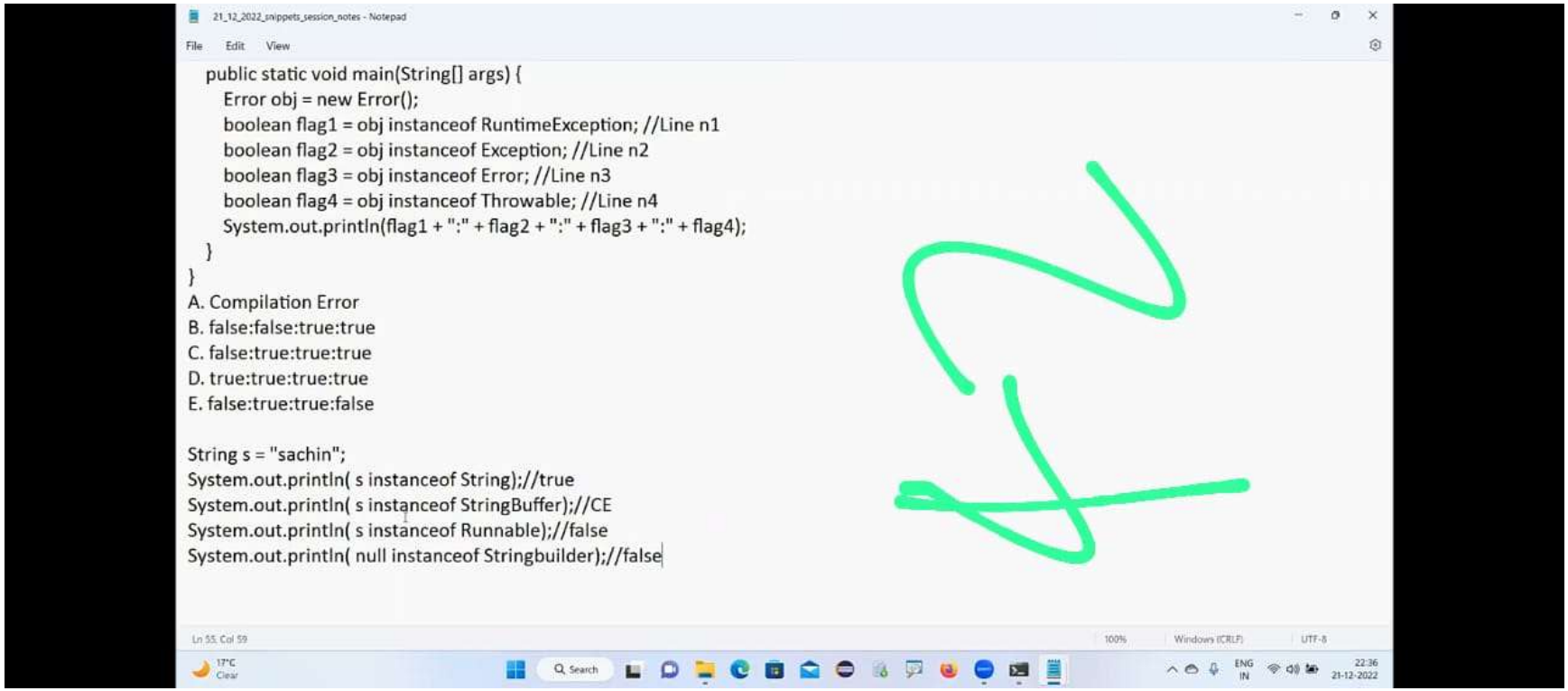
On execution, does Test class print "HELLO" on to the console?
A.Yes HELLO is printed on the console
B.NO Hello is not printed on the console

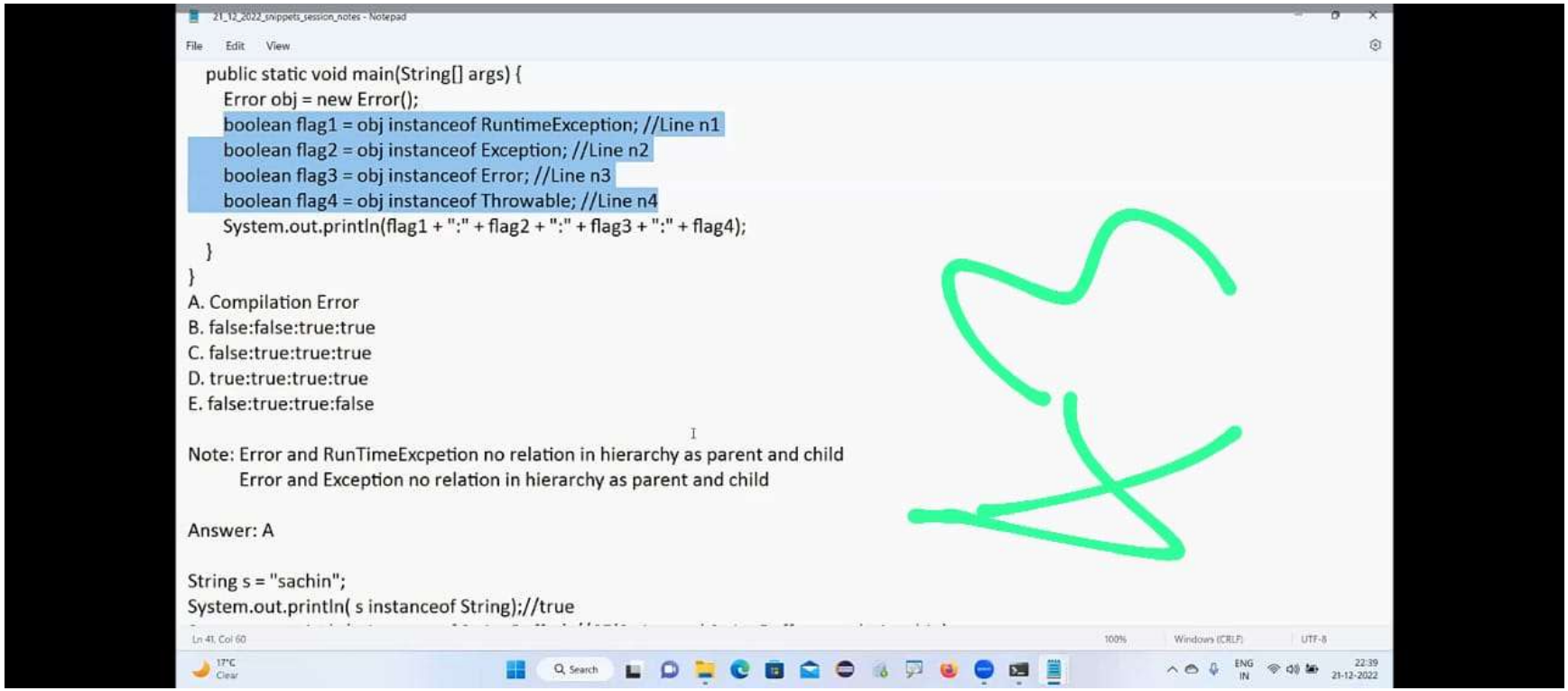
Answer: B

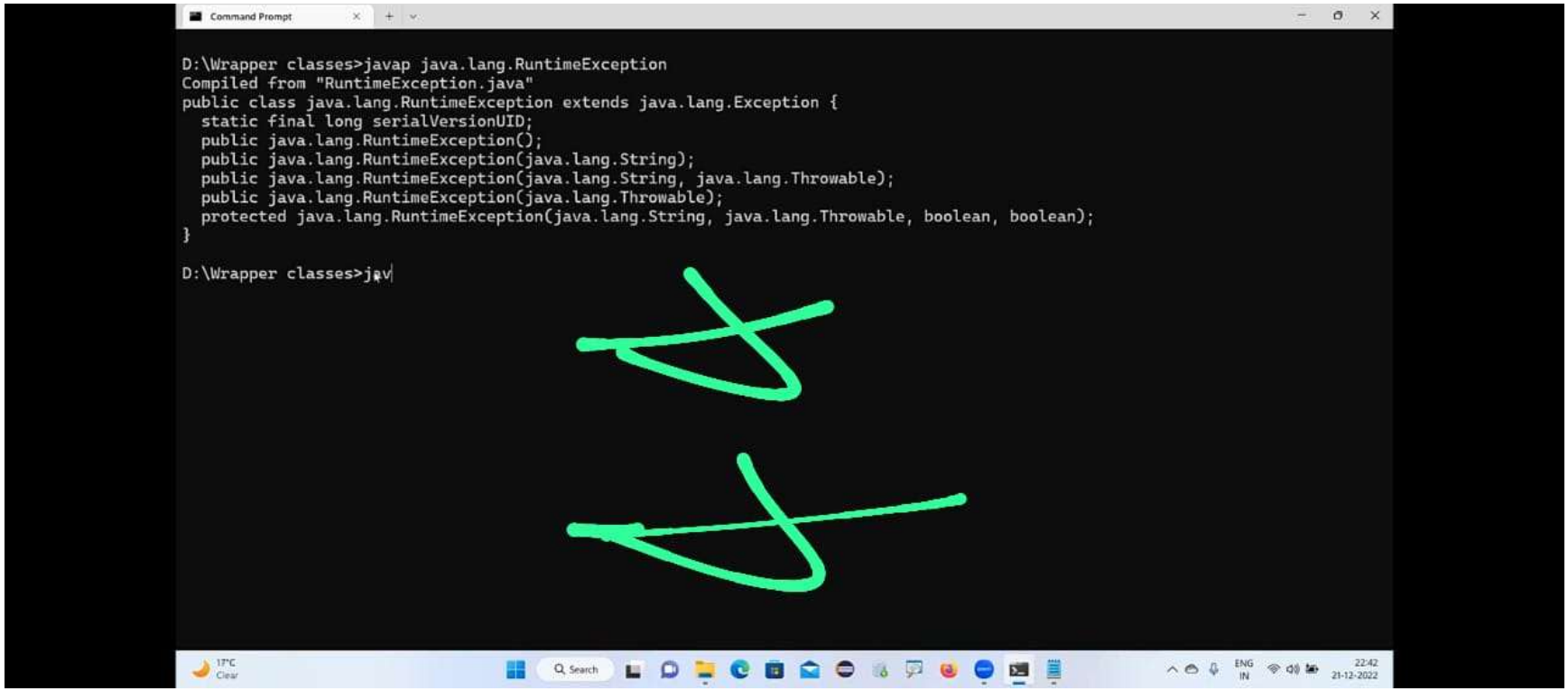
Ln 17, Col 1 100% Windows (CRLF) UTF-8

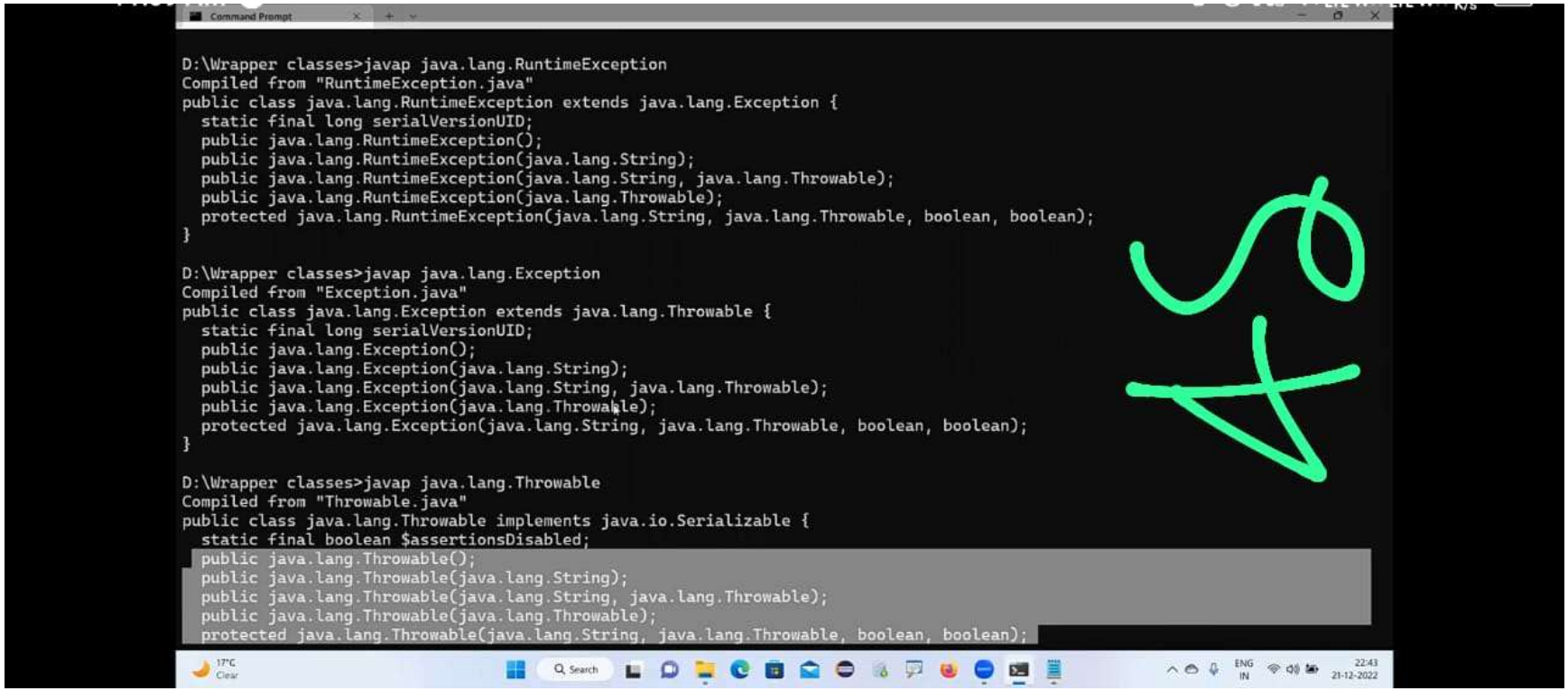
17°C Clear Search ENG IN 22:28 21-12-2022

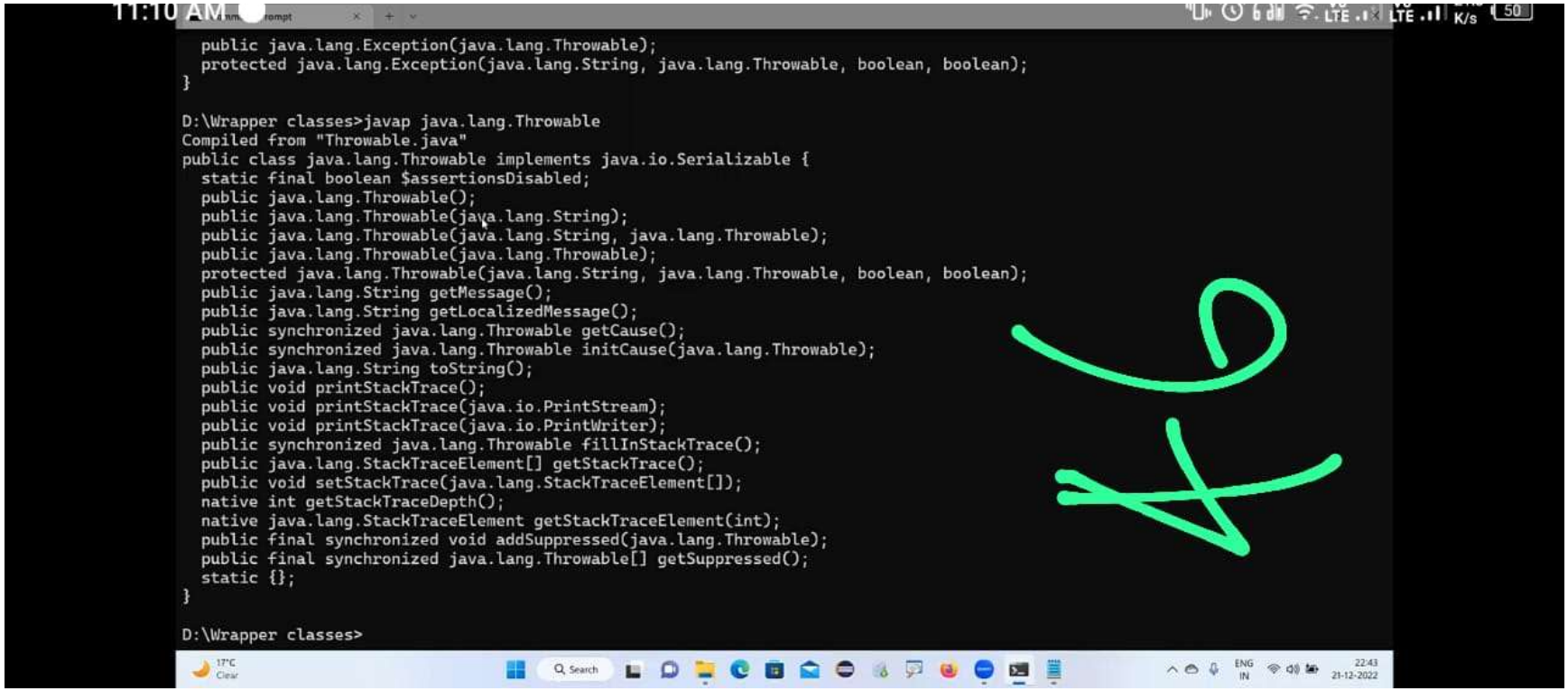

21_12_2022_snippets_session_notes - Notepad
File Edit View
Question
Consider below code:
public class Test {
static Double d1; // static variable ====> d1 =null
int x = d1.intValue();//instance variable ====> only upon creating an object
public static void main(String[] args) {
System.out.println("HELLO");//HELLO
}
}
On execution, does Test class print "HELLO" on to the console?
A.Yes HELLO is printed on the console
B.NO Hello is not printed on the console
Answer: A
Ln 32, Col 1
100%
Windows (CRLF)
UTF-8
17°C
Clear
Search
ENG
IN
22:31
21-12-2022

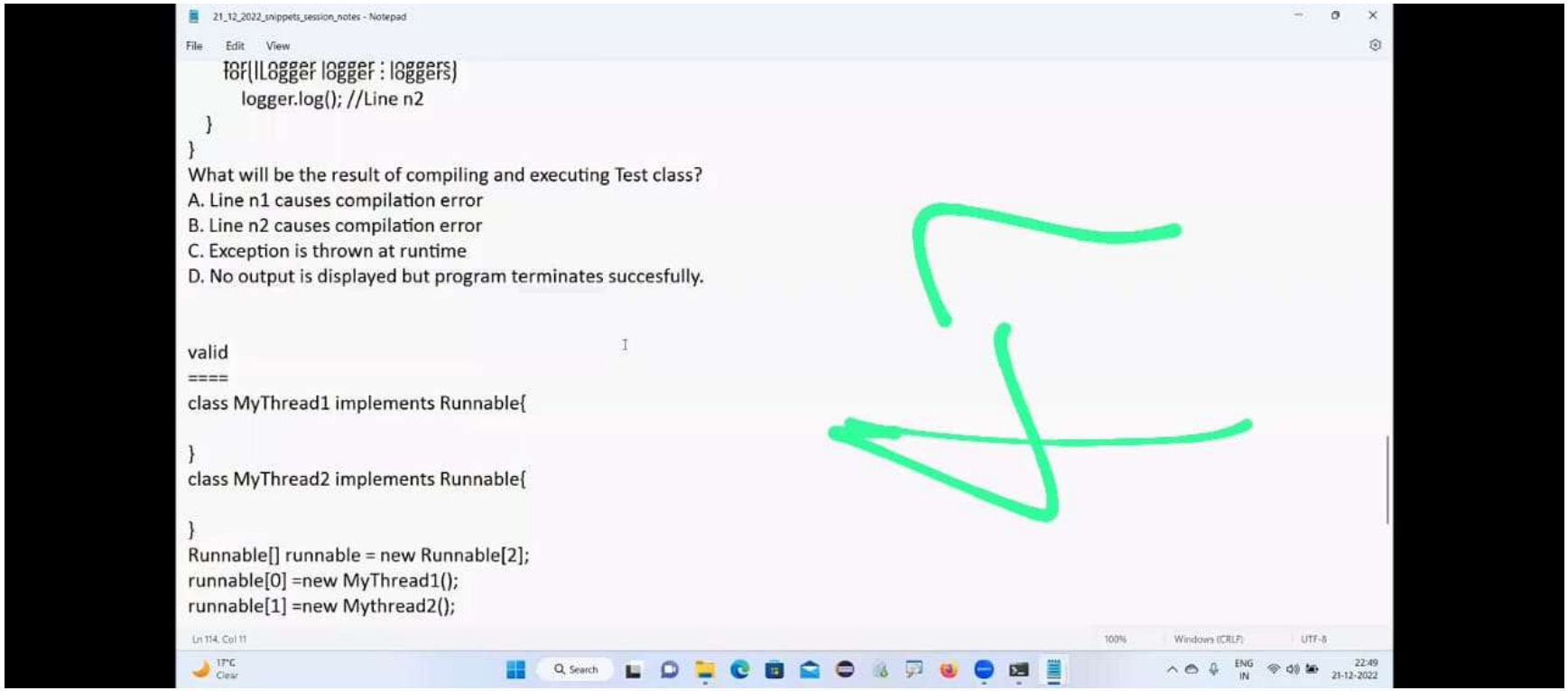

21_12_2022_snippets_session_notes - Notepad
File Edit View
public static void main(String[] args) {
Error obj = new Error();
boolean flag1 = obj instanceof RuntimeException; //Line n1
boolean flag2 = obj instanceof Exception; //Line n2
boolean flag3 = obj instanceof Error; //Line n3
boolean flag4 = obj instanceof Throwable; //Line n4
System.out.println(flag1 + ":" + flag2 + ":" + flag3 + ":" + flag4);
}
}
A. Compilation Error
B. false:false:true:true
C. false:true:true:true
D. true:true:true:true
E. false:true:true:false
String s = "sachin";
System.out.println(s instanceof String);//true
System.out.println(s instanceof StringBuffer);//CE
System.out.println(s instanceof Runnable);//false
System.out.println(null instanceof Stringbuilder);//false
Ln 55, Col 59
100%
Windows (CRLF)
UTF-8
17°C
Clear
Search
ENG
IN
22:36
21-12-2022

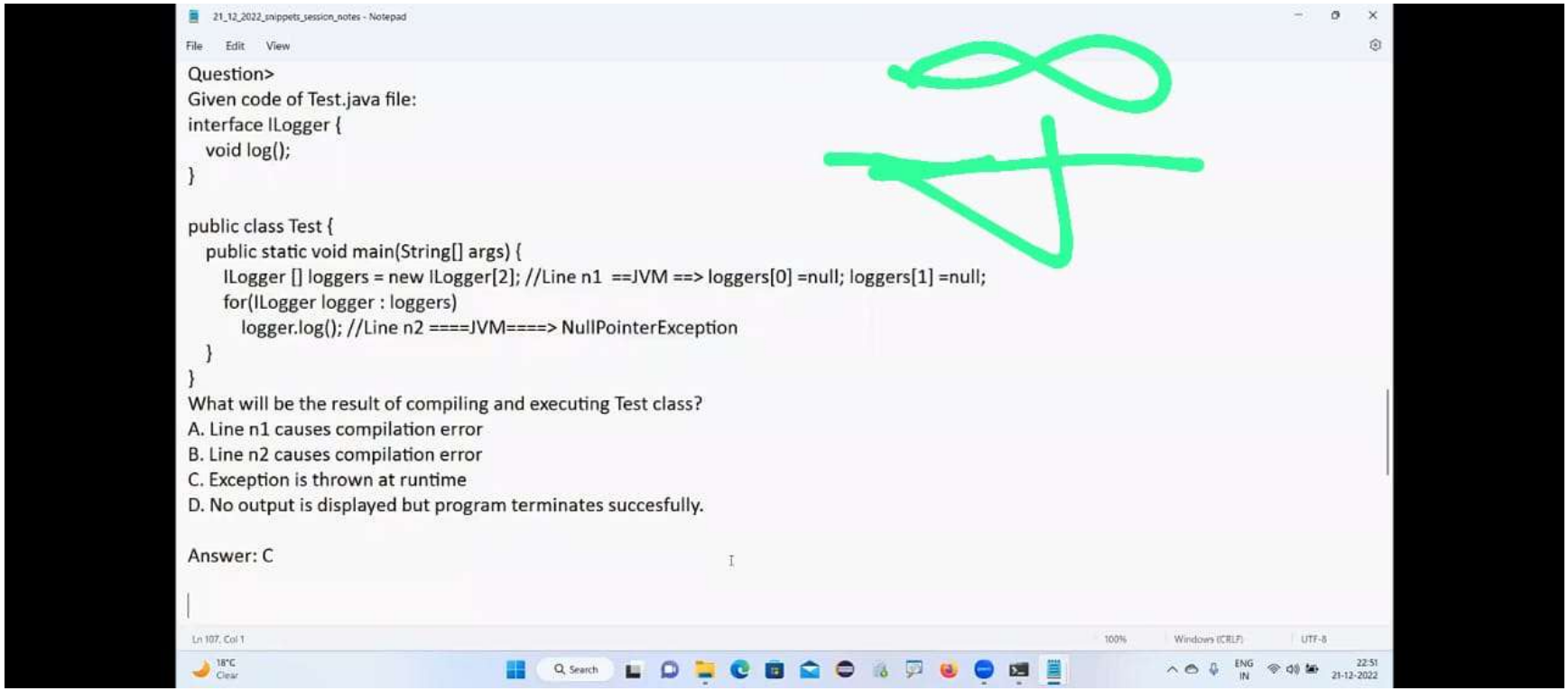

21_12_2022_snippets_session_notes - Notepad
File Edit View
public static void main(String[] args) {
Error obj = new Error();
boolean flag1 = obj instanceof RuntimeException; //Line n1
boolean flag2 = obj instanceof Exception; //Line n2
boolean flag3 = obj instanceof Error; //Line n3
boolean flag4 = obj instanceof Throwable; //Line n4
System.out.println(flag1 + ":" + flag2 + ":" + flag3 + ":" + flag4);
}
}
A. Compilation Error
B. false:false:true:true
C. false:true:true:true
D. true:true:true:true
E. false:true:true:false
Note: Error and RunTimeExcpetion no relation in hierarchy as parent and child
Error and Exception no relation in hierarchy as parent and child
Answer: A
String s = "sachin";
System.out.println(s instanceof String);//true
Ln 41, Col 60
100%
Windows (CRLF)
UTF-8
17°C
Clear
Search
ENG
IN
22:39
21-12-2022


Command Prompt
D:\Wrapper classes>javap java.lang.RuntimeException
Compiled from "RuntimeException.java"
public class java.lang.RuntimeException extends java.lang.Exception {
static final long serialVersionUID;
public java.lang.RuntimeException();
public java.lang.RuntimeException(java.lang.String);
public java.lang.RuntimeException(java.lang.String, java.lang.Throwable);
public java.lang.RuntimeException(java.lang.Throwable);
protected java.lang.RuntimeException(java.lang.String, java.lang.Throwable, boolean, boolean);
}
D:\Wrapper classes>jav
17°C
Clear
Search
ENG
IN
22:42
21-12-2022


Command Prompt
D:\Wrapper classes>javap java.lang.RuntimeException
Compiled from "RuntimeException.java"
public class java.lang.RuntimeException extends java.lang.Exception {
static final long serialVersionUID;
public java.lang.RuntimeException();
public java.lang.RuntimeException(java.lang.String);
public java.lang.RuntimeException(java.lang.String, java.lang.Throwable);
public java.lang.RuntimeException(java.lang.Throwable);
protected java.lang.RuntimeException(java.lang.String, java.lang.Throwable, boolean, boolean);
}
D:\Wrapper classes>javap java.lang.Exception
Compiled from "Exception.java"
public class java.lang.Exception extends java.lang.Throwable {
static final long serialVersionUID;
public java.lang.Exception();
public java.lang.Exception(java.lang.String);
public java.lang.Exception(java.lang.String, java.lang.Throwable);
public java.lang.Exception(java.lang.Throwable);
protected java.lang.Exception(java.lang.String, java.lang.Throwable, boolean, boolean);
}
D:\Wrapper classes>javap java.lang.Throwable
Compiled from "Throwable.java"
public class java.lang.Throwable implements java.io.Serializable {
static final boolean $assertionsDisabled;
public java.lang.Throwable();
public java.lang.Throwable(java.lang.String);
public java.lang.Throwable(java.lang.String, java.lang.Throwable);
public java.lang.Throwable(java.lang.Throwable);
protected java.lang.Throwable(java.lang.String, java.lang.Throwable, boolean, boolean);
17°C
Clear
Search
ENG
IN
22:43
21-12-2022


11:10 AM
public java.lang.Exception(java.lang.Throwable);
protected java.lang.Exception(java.lang.String, java.lang.Throwable, boolean, boolean);
}
D:\Wrapper classes>javap java.lang.Throwable
Compiled from "Throwable.java"
public class java.lang.Throwable implements java.io.Serializable {
static final boolean $assertionsDisabled;
public java.lang.Throwable();
public java.lang.Throwable(java.lang.String);
public java.lang.Throwable(java.lang.String, java.lang.Throwable);
public java.lang.Throwable(java.lang.Throwable);
protected java.lang.Throwable(java.lang.String, java.lang.Throwable, boolean, boolean);
public java.lang.String getMessage();
public java.lang.String getLocalizedMessage();
public synchronized java.lang.Throwable getCause();
public synchronized java.lang.Throwable initCause(java.lang.Throwable);
public java.lang.String toString();
public void printStackTrace();
public void printStackTrace(java.io.PrintStream);
public void printStackTrace(java.io.PrintWriter);
public synchronized java.lang.Throwable fillInStackTrace();
public java.lang.StackTraceElement[] getStackTrace();
public void setStackTrace(java.lang.StackTraceElement[]);
native int getStackTraceDepth();
native java.lang.StackTraceElement getStackTraceElement(int);
public final synchronized void addSuppressed(java.lang.Throwable);
public final synchronized java.lang.Throwable[] getSuppressed();
static {};
}
D:\Wrapper classes>
Search
ENG
IN
22:43
21-12-2022


21_12_2022_snippets_session_notes - Notepad
File Edit View
for(ILogger logger : loggers)
logger.log(); //Line n2
}
}
What will be the result of compiling and executing Test class?
A. Line n1 causes compilation error
B. Line n2 causes compilation error
C. Exception is thrown at runtime
D. No output is displayed but program terminates succesfully.
valid
====
class MyThread1 implements Runnable{
}
class MyThread2 implements Runnable{
}
Runnable[] runnable = new Runnable[2];
runnable[0] =new MyThread1();
runnable[1] =new Mythread2();
Ln 114, Col 11
100%
Windows (CRLF)
UTF-8
17°C
Clear
Search
ENG
IN
22:49
21-12-2022


21_12_2022_snippets_session_notes - Notepad
File Edit View
Question>
Given code of Test.java file:
interface ILogger {
void log();
}
public class Test {
public static void main(String[] args) {
ILogger [] loggers = new ILogger[2]; //Line n1 ==JVM ==> loggers[0] =null; loggers[1] =null;
for(ILogger logger : loggers)
logger.log(); //Line n2 ====JVM====> NullPointerException
}
}
What will be the result of compiling and executing Test class?
A. Line n1 causes compilation error
B. Line n2 causes compilation error
C. Exception is thrown at runtime
D. No output is displayed but program terminates succesfully.
Answer: C
Ln 107, Col 1
100%
Windows (CRLF)
UTF-8
18°C
Clear
Search
ENG
IN
22:51
21-12-2022