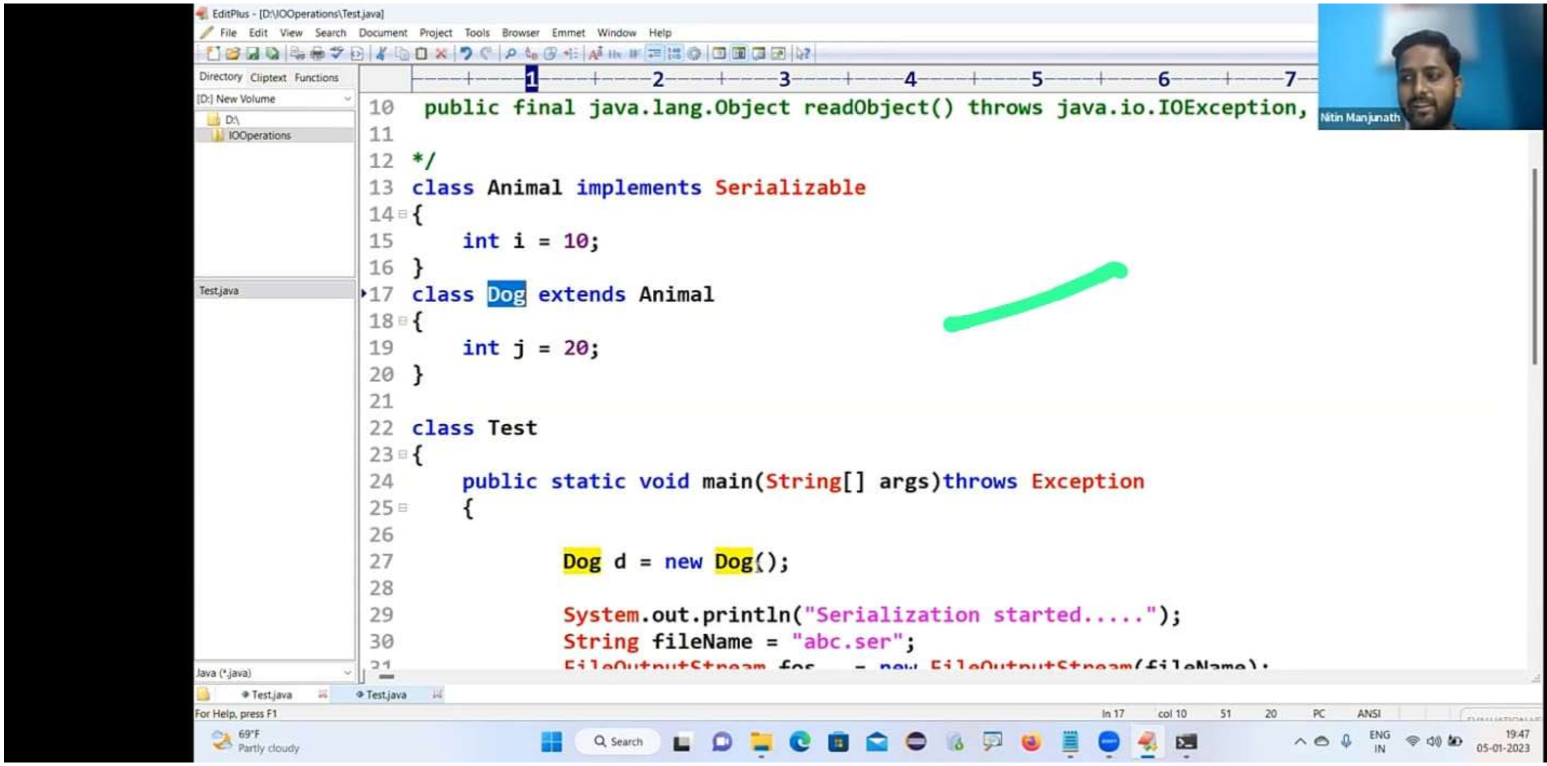
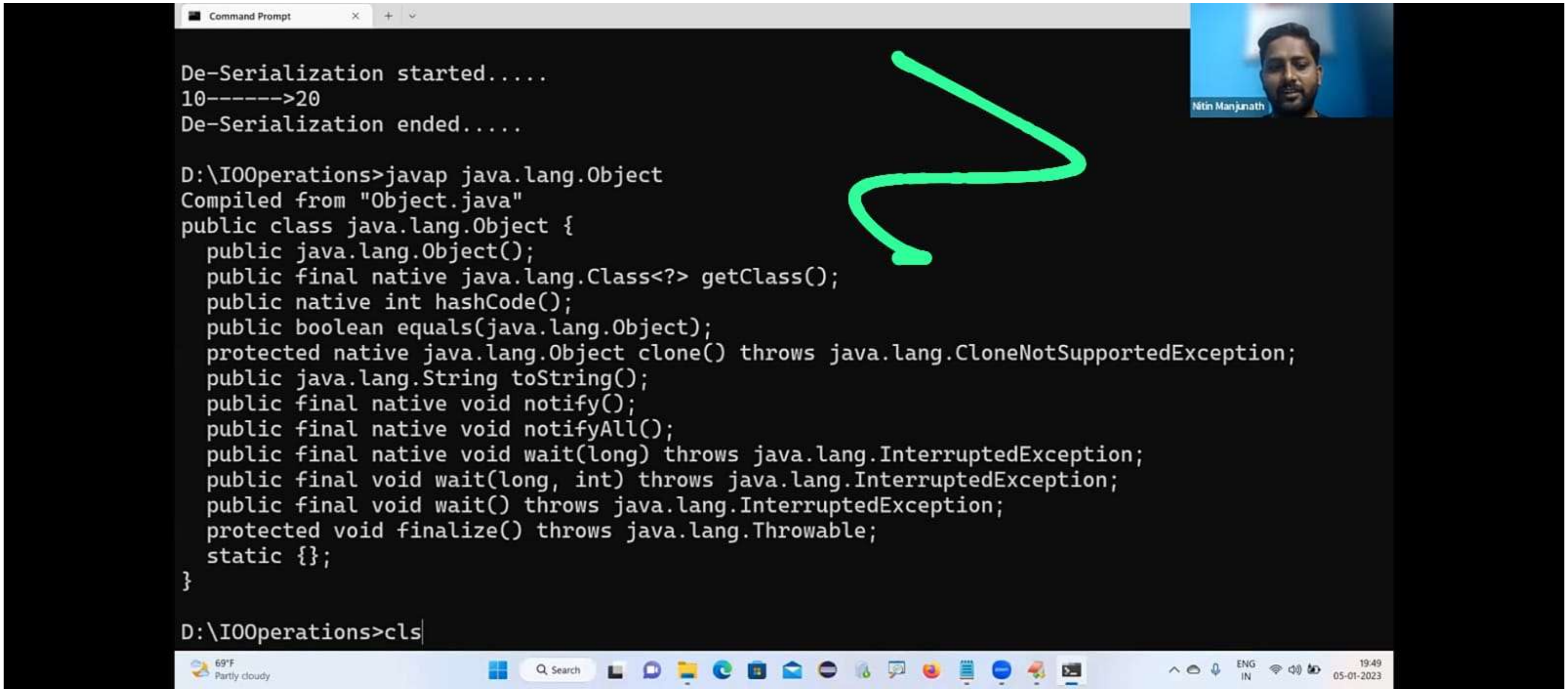
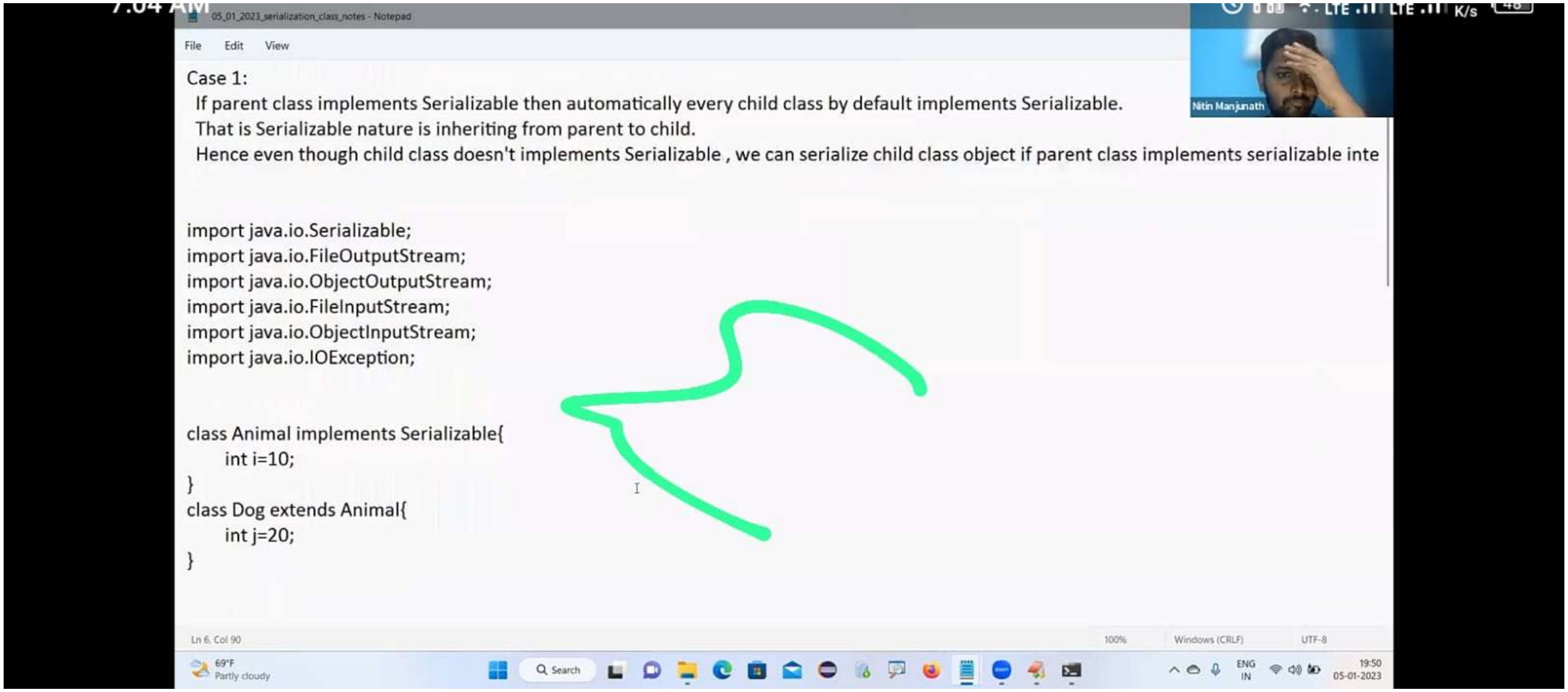
46Java_Serialization_Deserialization_Part2


EditPlus - [D:\IOOperations\Test.java]
File Edit View Search Document Project Tools Browser Emmet Window Help
Directory Cliptext Functions
[D:] New Volume
D:\
IOOperations
Test.java
Java (*.java)
Test.java
For Help, press F1
In 17 col 10 51 20 PC ANSI
69°F Partly cloudy
Search
ENG IN
19:47 05-01-2023
Nitin Manjunath

```java
 public final java.lang.Object readObject() throws java.io.IOException,

*/
class Animal implements Serializable
{
    int i = 10;
}
class Dog extends Animal
{
    int j = 20;
}

class Test
{
    public static void main(String[] args)throws Exception
    {

            Dog d = new Dog();

            System.out.println("Serialization started.....");
            String fileName = "abc.ser";
            FileOutputStream fos   = new FileOutputStream(fileName);
```



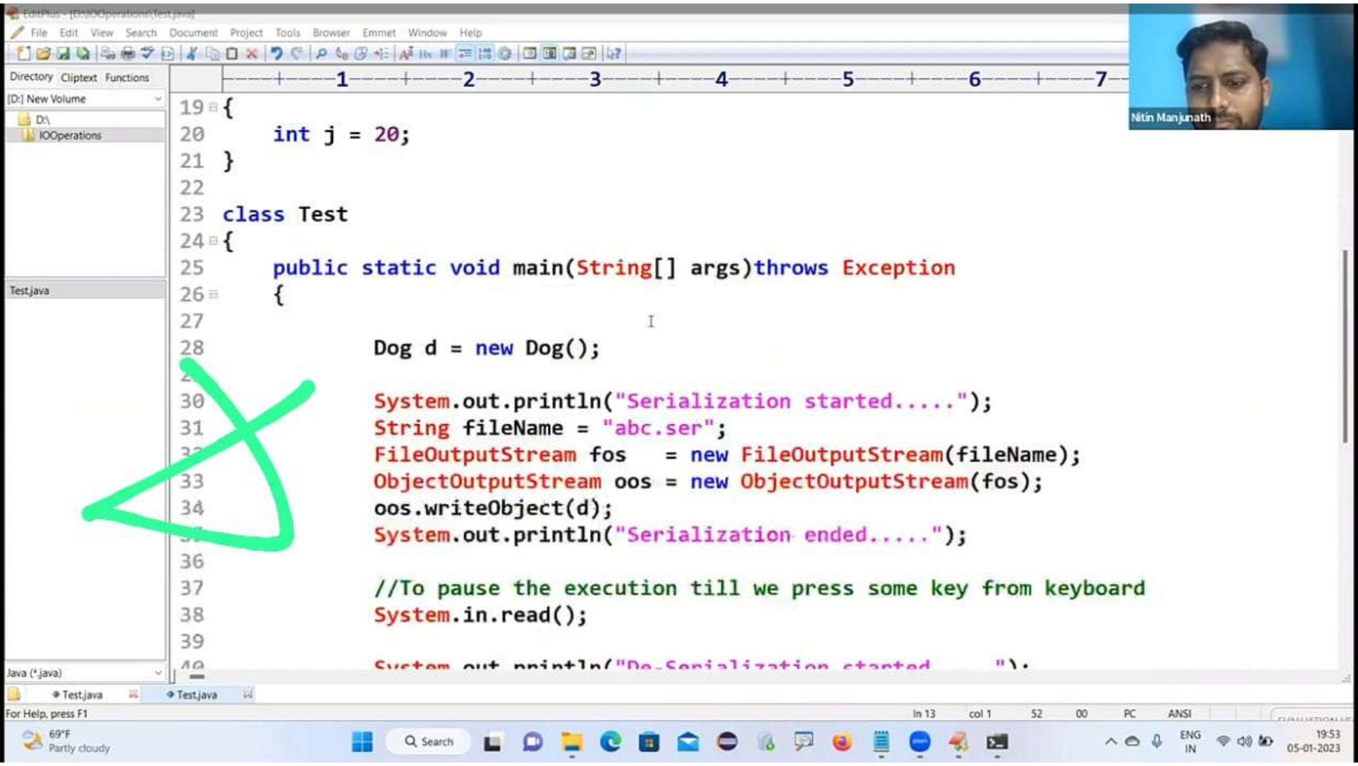
```
De-Serialization started.....
10------>20
De-Serialization ended.....

D:\IOOperations>javap java.lang.Object
Compiled from "Object.java"
public class java.lang.Object {
  public java.lang.Object();
  public final native java.lang.Class<?> getClass();
  public native int hashCode();
  public boolean equals(java.lang.Object);
  protected native java.lang.Object clone() throws java.lang.CloneNotSupportedException;
  public java.lang.String toString();
  public final native void notify();
  public final native void notifyAll();
  public final native void wait(long) throws java.lang.InterruptedException;
  public final void wait(long, int) throws java.lang.InterruptedException;
  public final void wait() throws java.lang.InterruptedException;
  protected void finalize() throws java.lang.Throwable;
  static {};
}

D:\IOOperations>cls
```



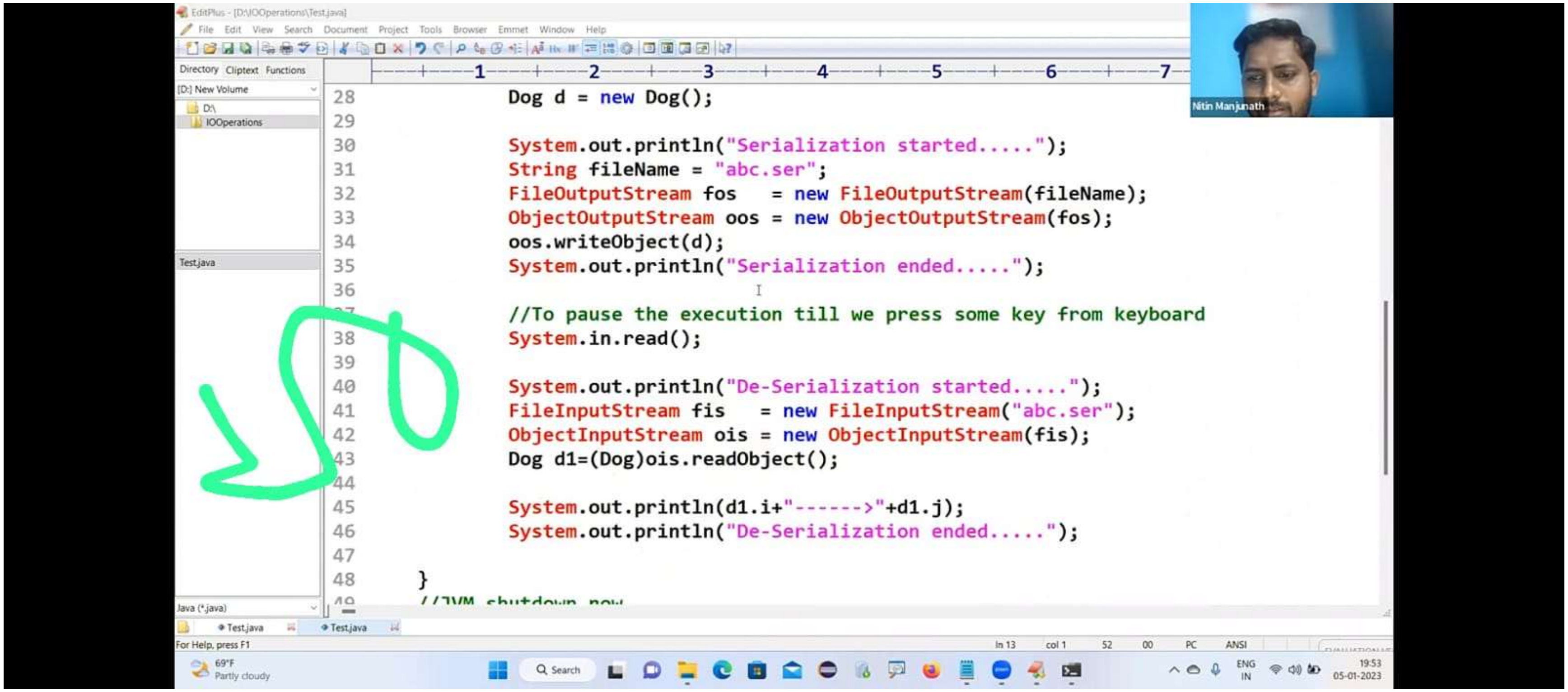
Case 1:

If parent class implements Serializable then automatically every child class by default implements Serializable.
That is Serializable nature is inheriting from parent to child.
Hence even though child class doesn't implements Serializable , we can serialize child class object if parent class implements serializable inte

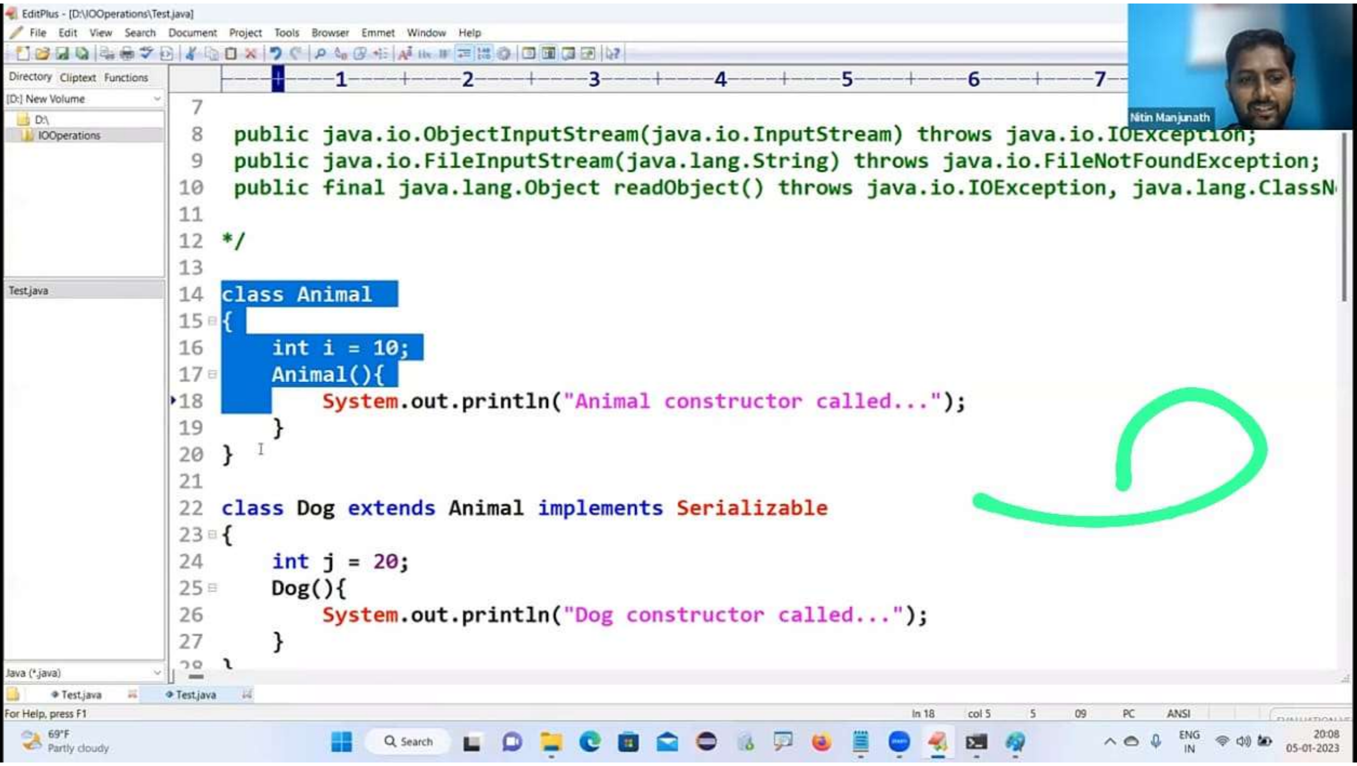
```
import java.io.Serializable;
import java.io.FileOutputStream;
import java.io.ObjectOutputStream;
import java.io.FileInputStream;
import java.io.ObjectInputStream;
import java.io.IOException;


class Animal implements Serializable{
      int i=10;
}
class Dog extends Animal{
      int j=20;
}
```


EditPlus - [D:\IOOperations\Test.java]
File Edit View Search Document Project Tools Browser Emmet Window Help
Directory Cliptext Functions
[D:] New Volume
D:\
IOOperations
Test.java
Java (*.java)
Test.java
Test.java
For Help, press F1
In 13 col 1 52 00 PC ANSI
69°F Partly cloudy
Search
ENG IN
19:53 05-01-2023
Nitin Manjunath

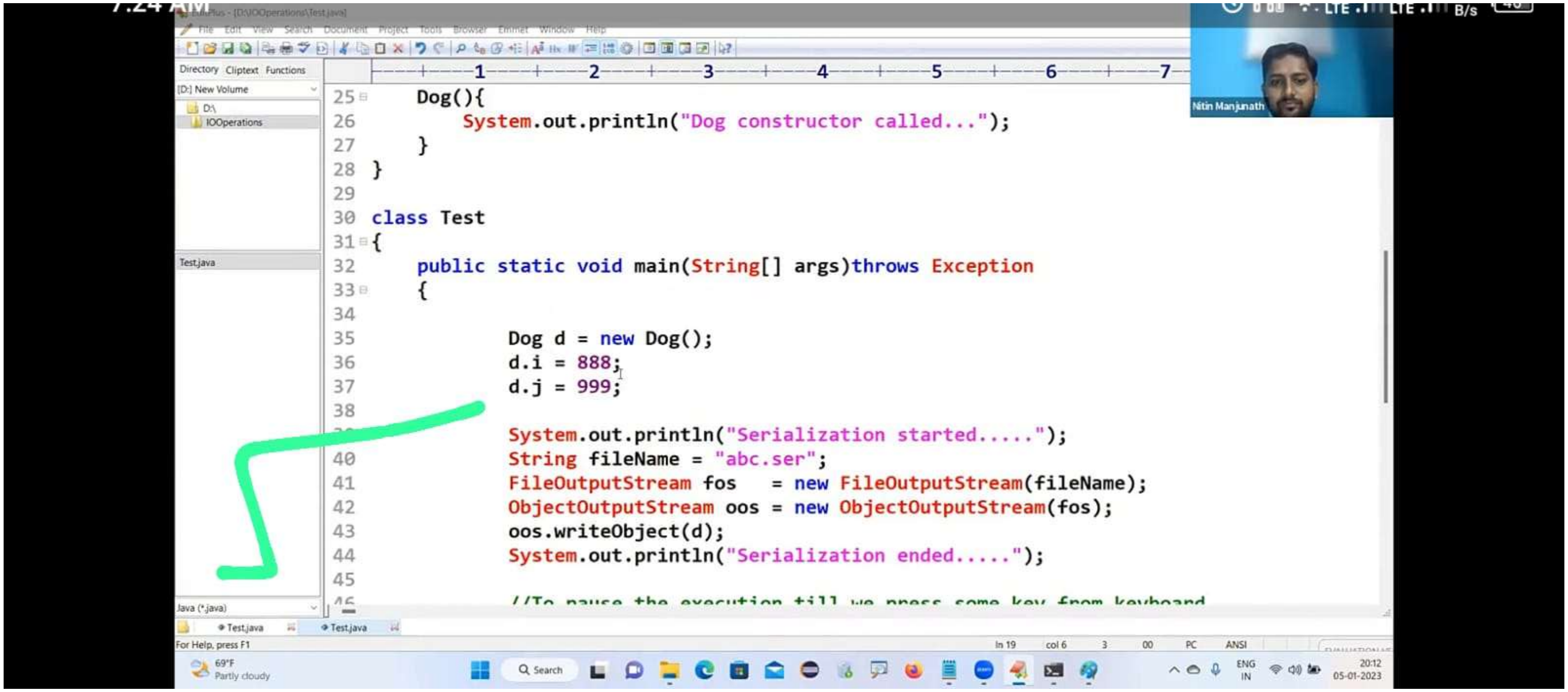
```
{
    int j = 20;
}

class Test
{
    public static void main(String[] args)throws Exception
    {

            Dog d = new Dog();

            System.out.println("Serialization started.....");
            String fileName = "abc.ser";
            FileOutputStream fos   = new FileOutputStream(fileName);
            ObjectOutputStream oos = new ObjectOutputStream(fos);
            oos.writeObject(d);
            System.out.println("Serialization ended.....");

            //To pause the execution till we press some key from keyboard
            System.in.read();

            System.out.println("De-Serialization started.....");
```


EditPlus - [D:\IOOperations\Test.java]
File Edit View Search Document Project Tools Browser Emmet Window Help
Directory Cliptext Functions
[D:] New Volume
D:\
IOOperations
Test.java
Java (*.java)
Test.java
For Help, press F1
In 13 col 1 52 00 PC ANSI
69°F Partly cloudy
Search
ENG IN
19:53 05-01-2023
Nitin Manjunath

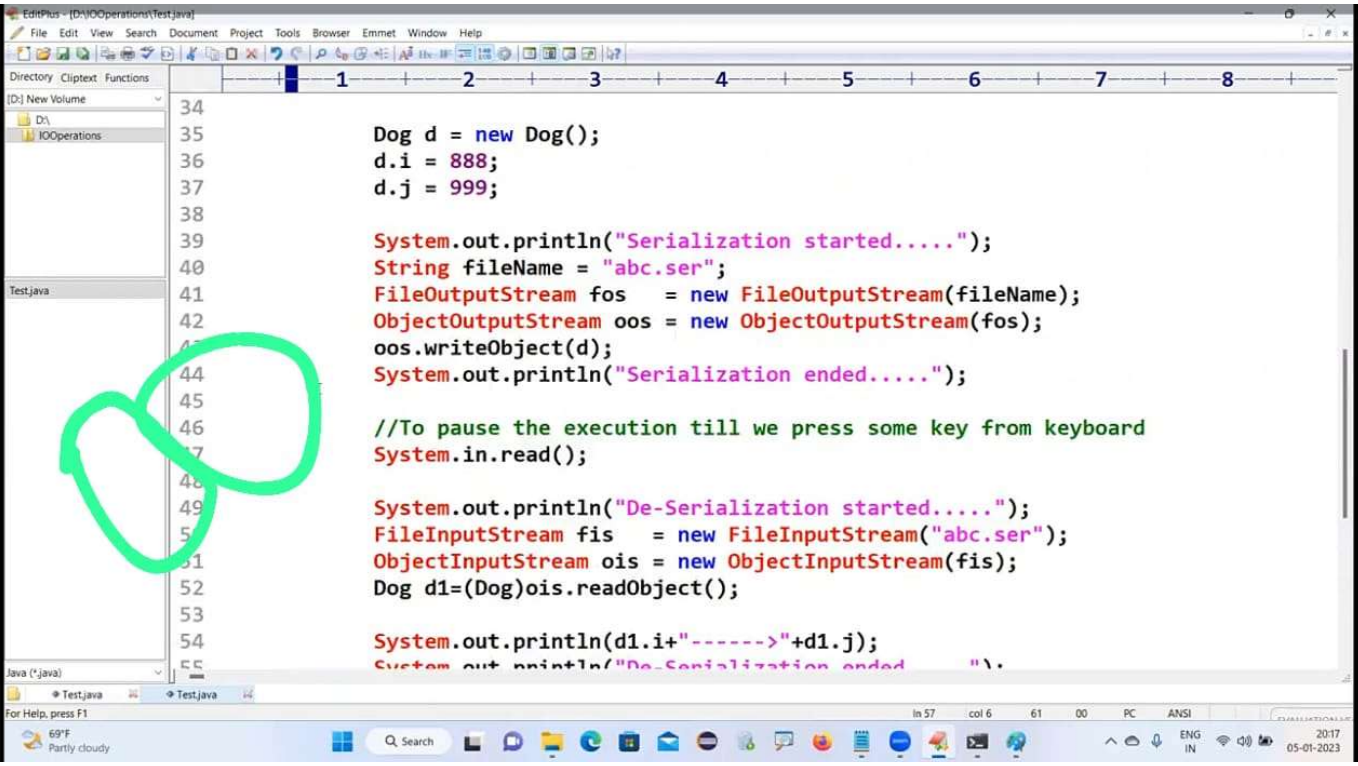
```java
            Dog d = new Dog();

            System.out.println("Serialization started.....");
            String fileName = "abc.ser";
            FileOutputStream fos   = new FileOutputStream(fileName);
            ObjectOutputStream oos = new ObjectOutputStream(fos);
            oos.writeObject(d);
            System.out.println("Serialization ended.....");

            //To pause the execution till we press some key from keyboard
            System.in.read();

            System.out.println("De-Serialization started.....");
            FileInputStream fis   = new FileInputStream("abc.ser");
            ObjectInputStream ois = new ObjectInputStream(fis);
            Dog d1=(Dog)ois.readObject();

            System.out.println(d1.i+"------>"+d1.j);
            System.out.println("De-Serialization ended.....");

        }
        //JVM shutdown now
```



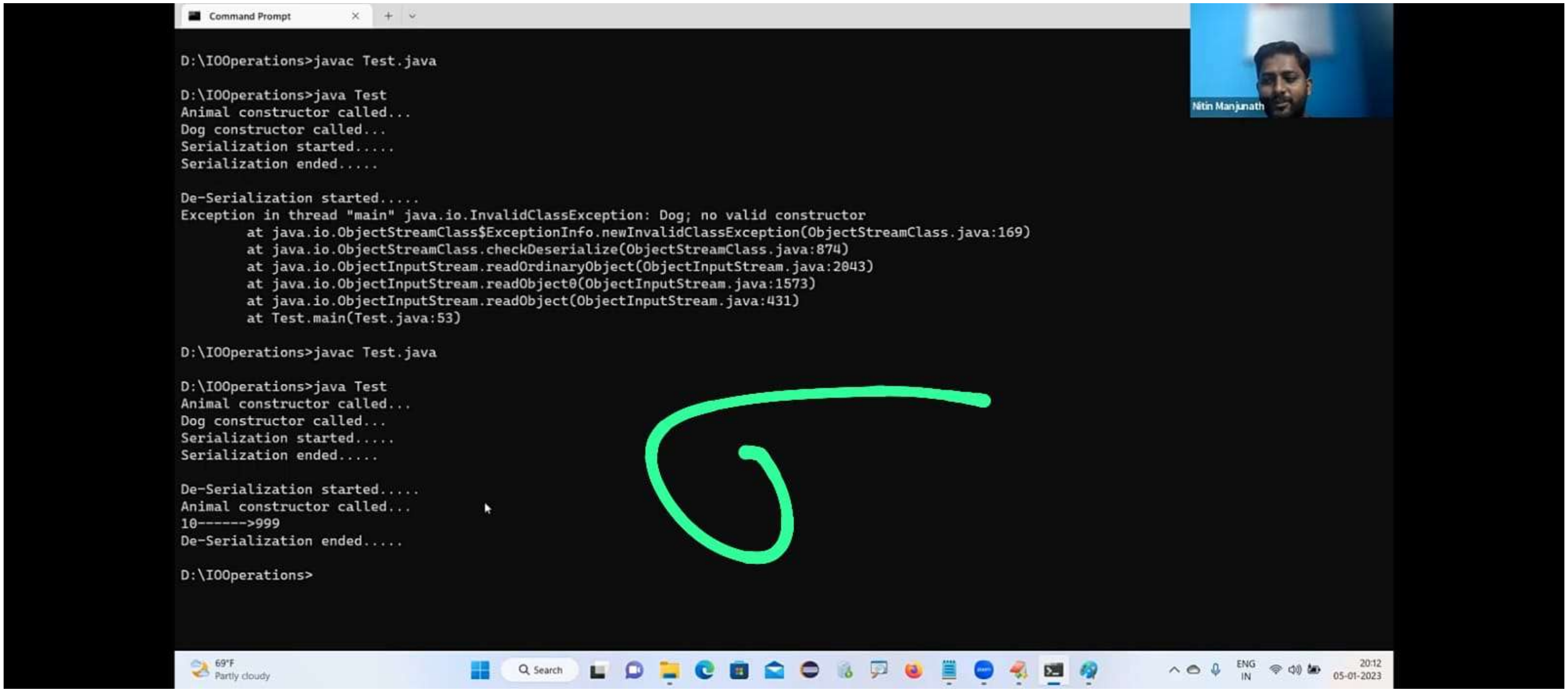
```
 public java.io.ObjectInputStream(java.io.InputStream) throws java.io.IOException;
 public java.io.FileInputStream(java.lang.String) throws java.io.FileNotFoundException;
 public final java.lang.Object readObject() throws java.io.IOException, java.lang.ClassN

*/

class Animal
{
    int i = 10;
    Animal(){
        System.out.println("Animal constructor called...");
    }
}

class Dog extends Animal implements Serializable
{
    int j = 20;
    Dog(){
        System.out.println("Dog constructor called...");
    }
}
```



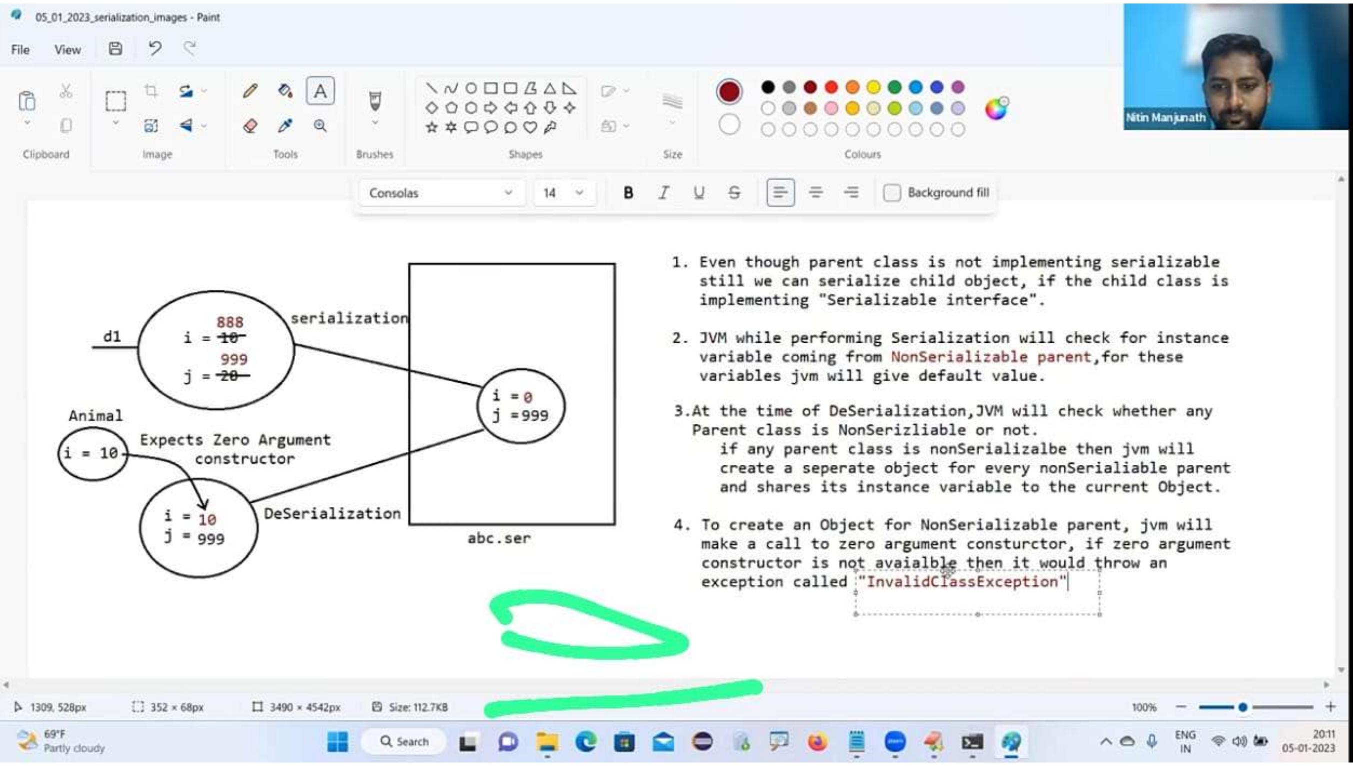
```
    Dog(){
        System.out.println("Dog constructor called...");
    }
}

class Test
{
    public static void main(String[] args)throws Exception
    {

            Dog d = new Dog();
            d.i = 888;
            d.j = 999;

            System.out.println("Serialization started.....");
            String fileName = "abc.ser";
            FileOutputStream fos   = new FileOutputStream(fileName);
            ObjectOutputStream oos = new ObjectOutputStream(fos);
            oos.writeObject(d);
            System.out.println("Serialization ended.....");

            //To pause the execution till we press some key from keyboard
```


EditPlus - [D:\IOOperations\Test.java]
File Edit View Search Document Project Tools Browser Emmet Window Help
Directory Cliptext Functions
[D:] New Volume
D:\
IOOperations
Test.java
Java (*.java)
Test.java
For Help, press F1
ln 57 col 6 61 00 PC ANSI
69°F Partly cloudy
Search
ENG IN
20:17 05-01-2023

```java

        Dog d = new Dog();
        d.i = 888;
        d.j = 999;

        System.out.println("Serialization started.....");
        String fileName = "abc.ser";
        FileOutputStream fos   = new FileOutputStream(fileName);
        ObjectOutputStream oos = new ObjectOutputStream(fos);
        oos.writeObject(d);
        System.out.println("Serialization ended.....");

        //To pause the execution till we press some key from keyboard
        System.in.read();

        System.out.println("De-Serialization started.....");
        FileInputStream fis   = new FileInputStream("abc.ser");
        ObjectInputStream ois = new ObjectInputStream(fis);
        Dog d1=(Dog)ois.readObject();

        System.out.println(d1.i+"------>"+d1.j);
        System.out.println("De-Serialization ended.....");
```



```
D:\IOOperations>javac Test.java

D:\IOOperations>java Test
Animal constructor called...
Dog constructor called...
Serialization started.....
Serialization ended.....

De-Serialization started.....
Exception in thread "main" java.io.InvalidClassException: Dog; no valid constructor
        at java.io.ObjectStreamClass$ExceptionInfo.newInvalidClassException(ObjectStreamClass.java:169)
        at java.io.ObjectStreamClass.checkDeserialize(ObjectStreamClass.java:874)
        at java.io.ObjectInputStream.readOrdinaryObject(ObjectInputStream.java:2043)
        at java.io.ObjectInputStream.readObject0(ObjectInputStream.java:1573)
        at java.io.ObjectInputStream.readObject(ObjectInputStream.java:431)
        at Test.main(Test.java:53)

D:\IOOperations>javac Test.java

D:\IOOperations>java Test
Animal constructor called...
Dog constructor called...
Serialization started.....
Serialization ended.....

De-Serialization started.....
Animal constructor called...
10------>999
De-Serialization ended.....

D:\IOOperations>
```

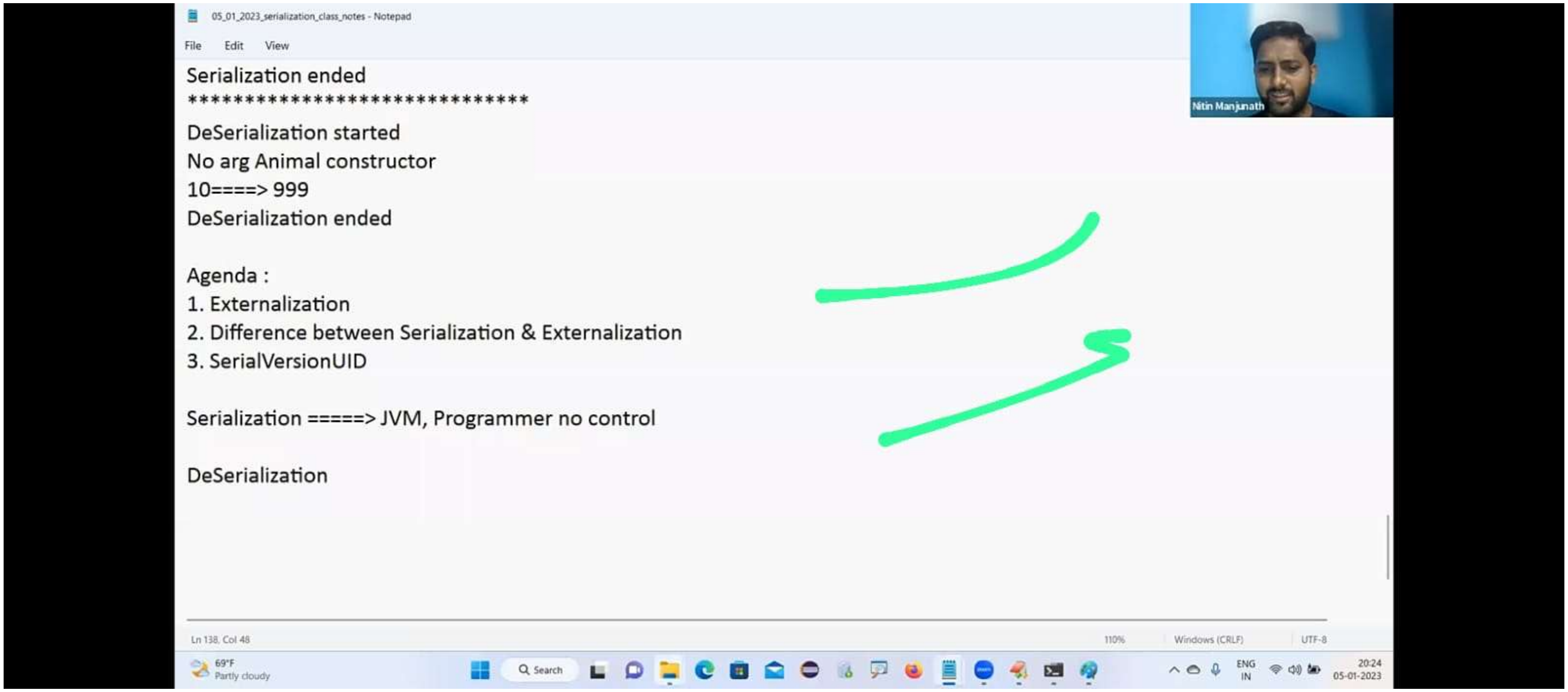


1. Even though parent class is not implementing serializable still we can serialize child object, if the child class is implementing "Serializable interface".

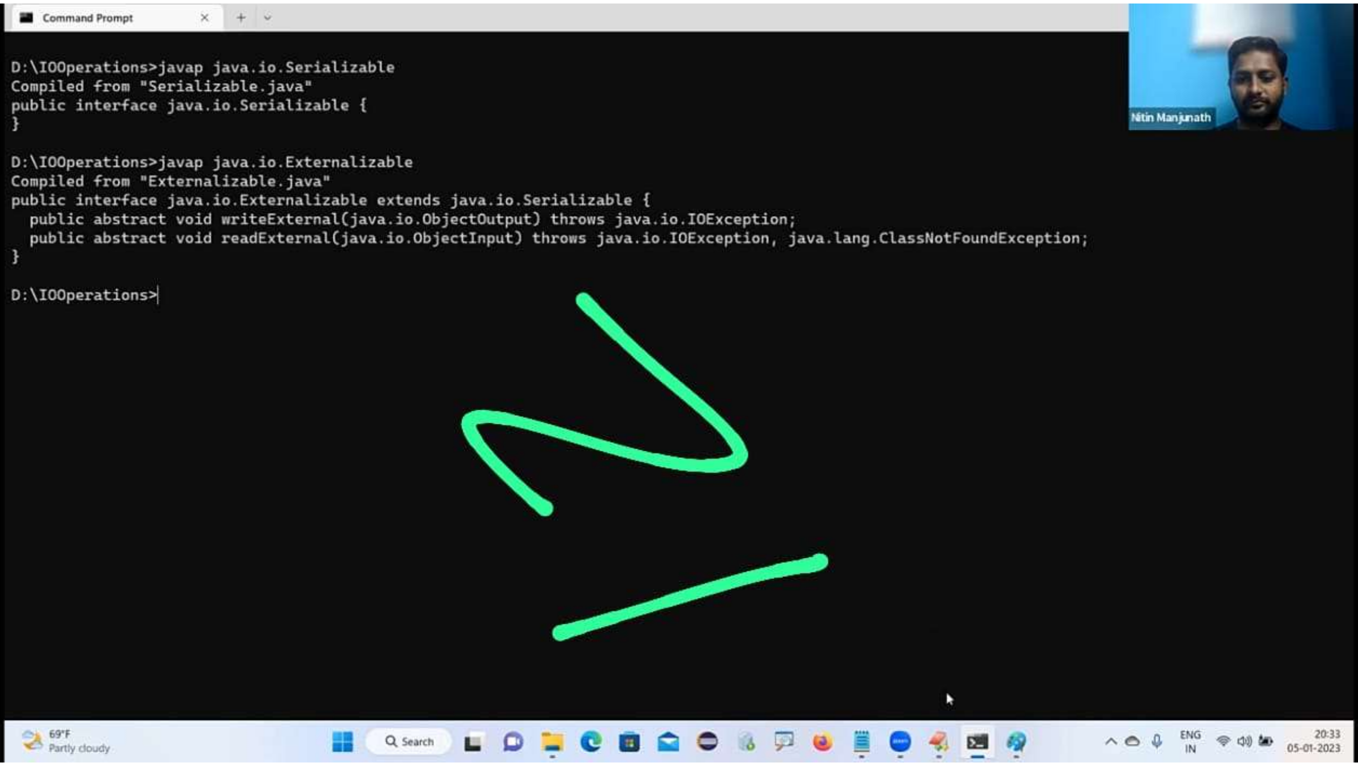
2. JVM while performing Serialization will check for instance variable coming from NonSerializable parent,for these variables jvm will give default value.

3.At the time of DeSerialization,JVM will check whether any Parent class is NonSerizliable or not.
if any parent class is nonSerializalbe then jvm will create a seperate object for every nonSerialiable parent and shares its instance variable to the current Object.

4. To create an Object for NonSerializable parent, jvm will make a call to zero argument consturctor, if zero argument constructor is not avaialble then it would throw an exception called "InvalidClassException"

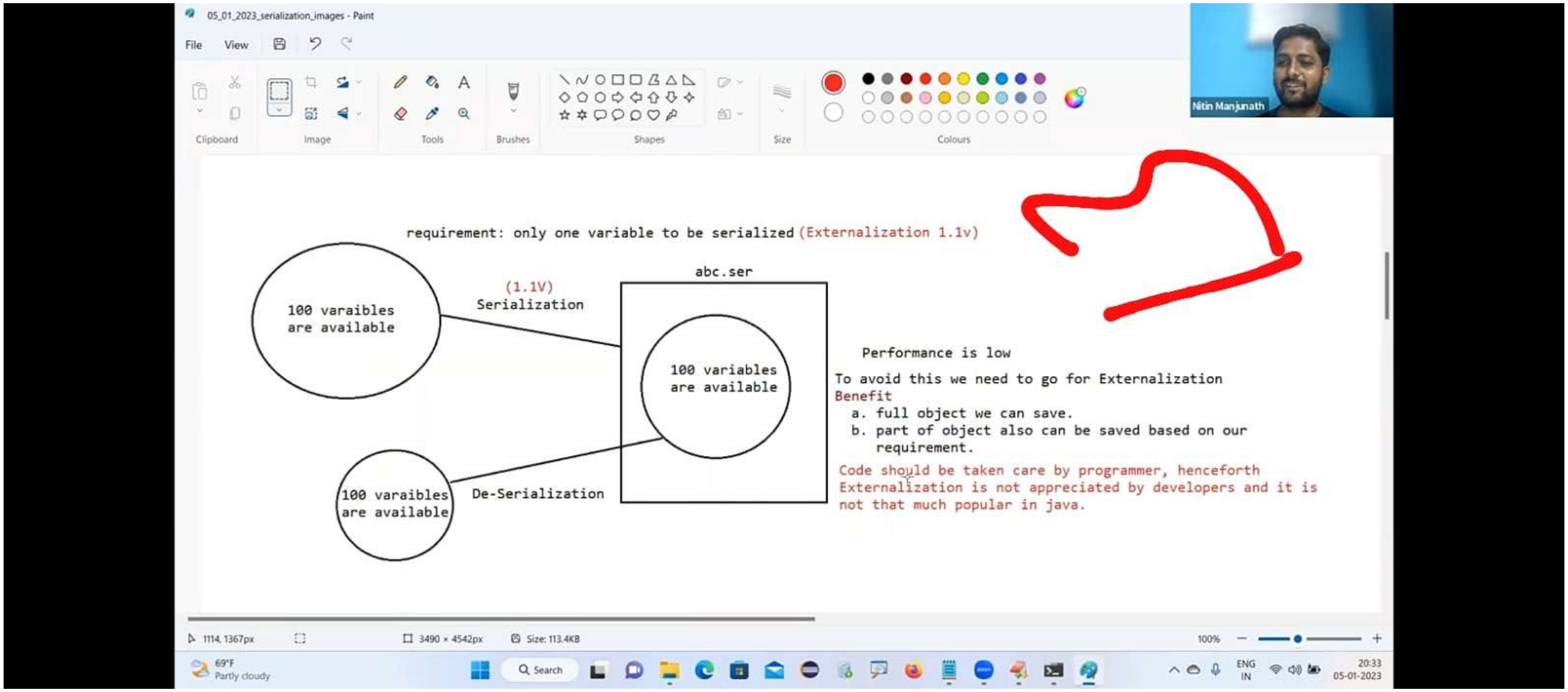


```
Serialization ended
******************************
DeSerialization started
No arg Animal constructor
10====> 999
DeSerialization ended

Agenda :
1. Externalization
2. Difference between Serialization & Externalization
3. SerialVersionUID

Serialization =====> JVM, Programmer no control

DeSerialization
```



```
D:\IOOperations>javap java.io.Serializable
Compiled from "Serializable.java"
public interface java.io.Serializable {
}

D:\IOOperations>javap java.io.Externalizable
Compiled from "Externalizable.java"
public interface java.io.Externalizable extends java.io.Serializable {
  public abstract void writeExternal(java.io.ObjectOutput) throws java.io.IOException;
  public abstract void readExternal(java.io.ObjectInput) throws java.io.IOException, java.lang.ClassNotFoundException;
}

D:\IOOperations>
```

12



05_01_2023_serialization_images - Paint
File
View
Clipboard
Image
Tools
Brushes
Shapes
Size
Colours
Nitin Manjunath

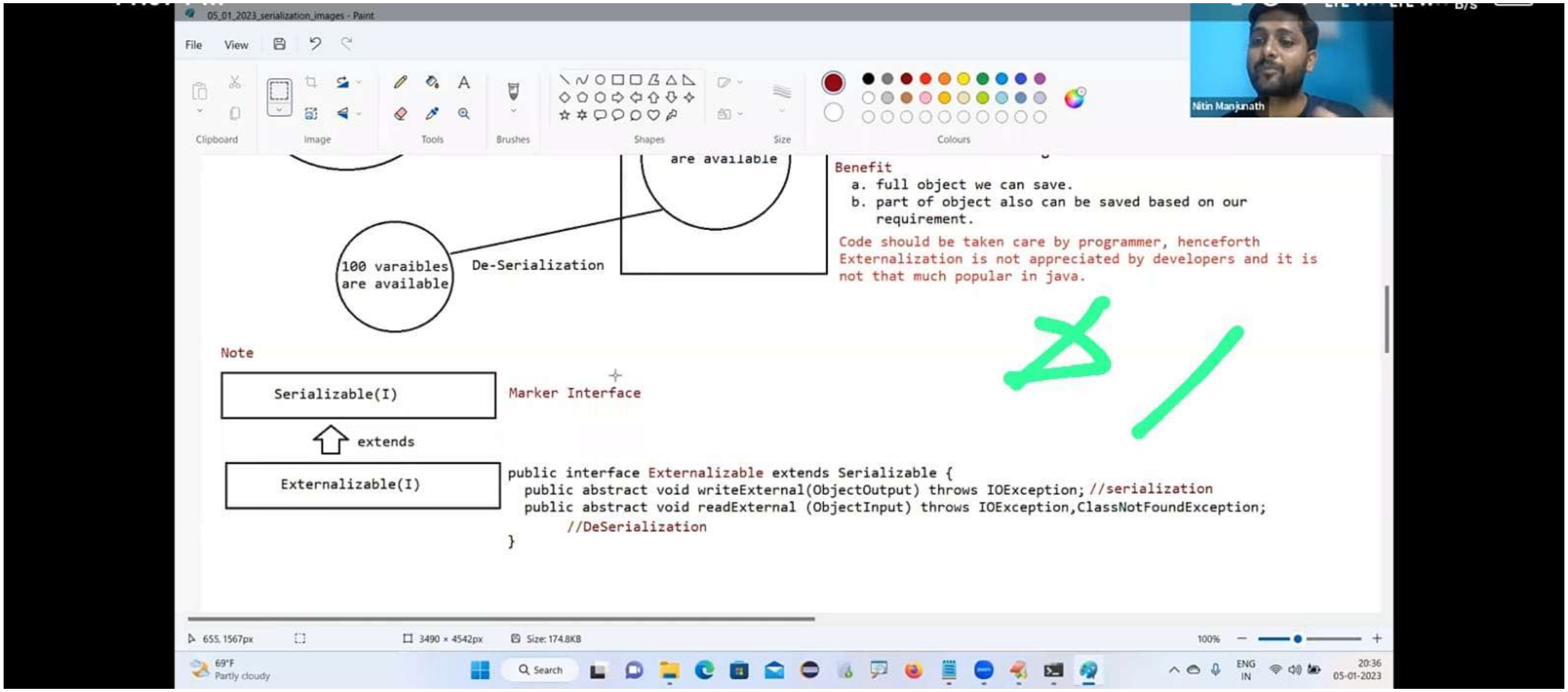
requirement: only one variable to be serialized (Externalization 1.1v)

100 varaibles are available
(1.1V)
Serialization
abc.ser
100 variables are available
100 varaibles are available
De-Serialization

Performance is low

To avoid this we need to go for Externalization
Benefit
  a. full object we can save.
  b. part of object also can be saved based on our requirement.

Code should be taken care by programmer, henceforth Externalization is not appreciated by developers and it is not that much popular in java.



Benefit
  a. full object we can save.
  b. part of object also can be saved based on our requirement.

Code should be taken care by programmer, henceforth Externalization is not appreciated by developers and it is not that much popular in java.

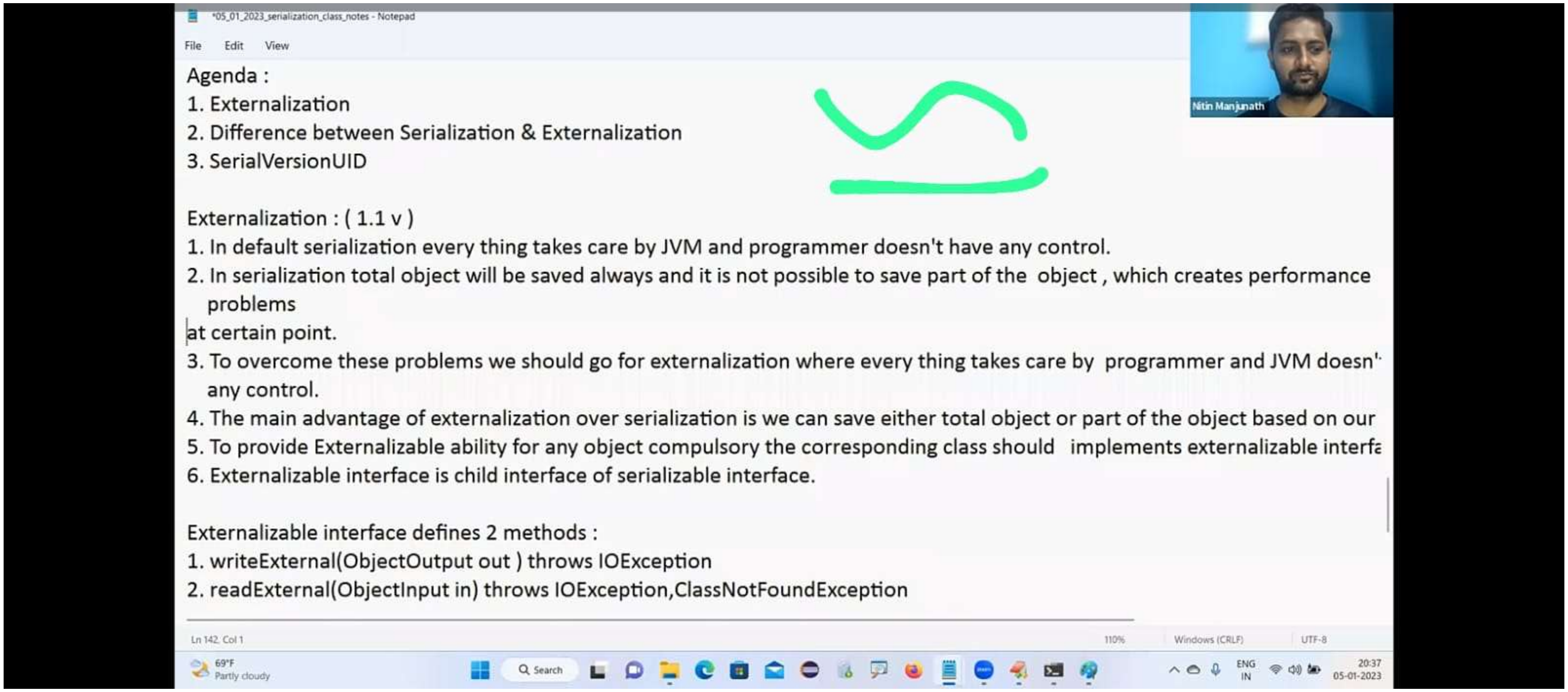
Note

Serializable(I) Marker Interface

extends

Externalizable(I)

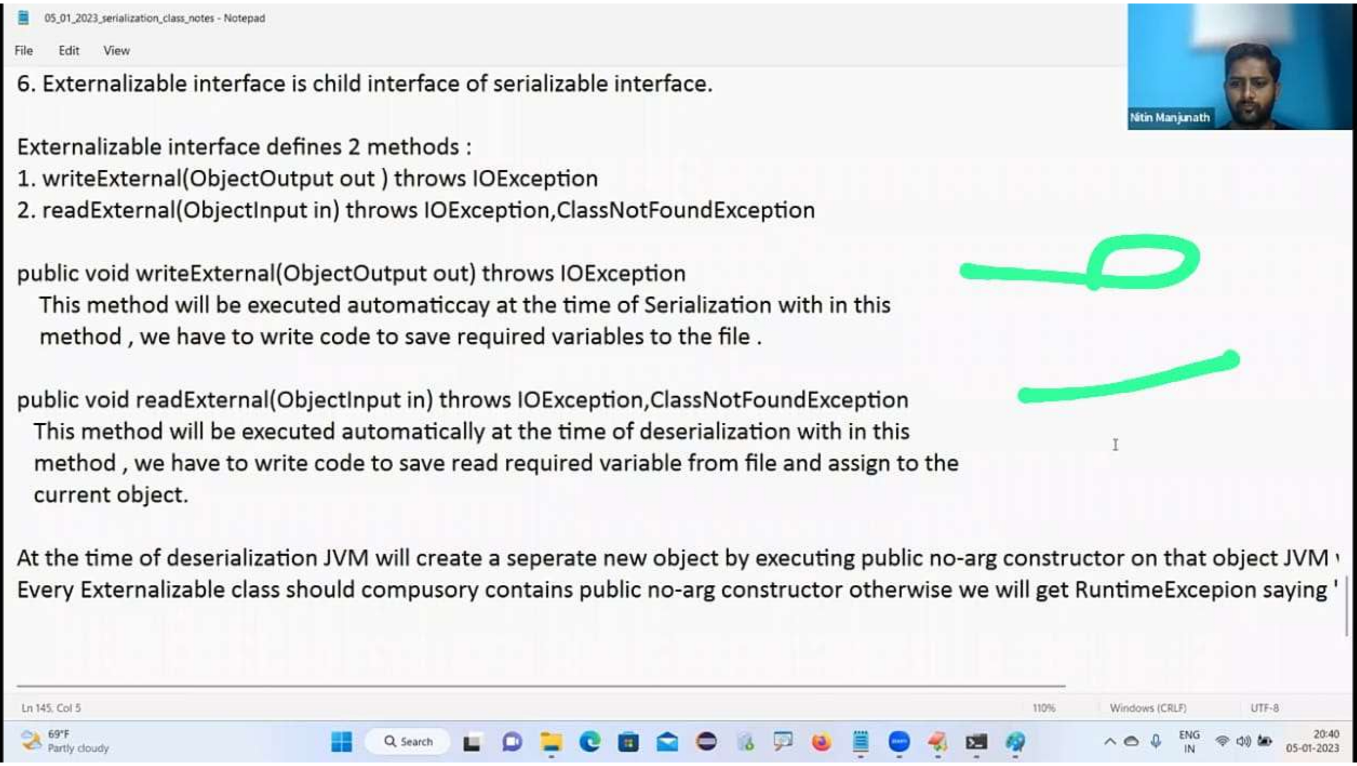
```
public interface Externalizable extends Serializable {
  public abstract void writeExternal(ObjectOutput) throws IOException; //serialization
  public abstract void readExternal (ObjectInput) throws IOException,ClassNotFoundException;
        //DeSerialization
}
```



Agenda :
1. Externalization
2. Difference between Serialization & Externalization
3. SerialVersionUID

Externalization : (1.1 v)
1. In default serialization every thing takes care by JVM and programmer doesn't have any control.
2. In serialization total object will be saved always and it is not possible to save part of the object , which creates performance problems
at certain point.
3. To overcome these problems we should go for externalization where every thing takes care by programmer and JVM doesn'
any control.
4. The main advantage of externalization over serialization is we can save either total object or part of the object based on our
5. To provide Externalizable ability for any object compulsory the corresponding class should implements externalizable interfa
6. Externalizable interface is child interface of serializable interface.

Externalizable interface defines 2 methods :
1. writeExternal(ObjectOutput out) throws IOException
2. readExternal(ObjectInput in) throws IOException,ClassNotFoundException



6. Externalizable interface is child interface of serializable interface.

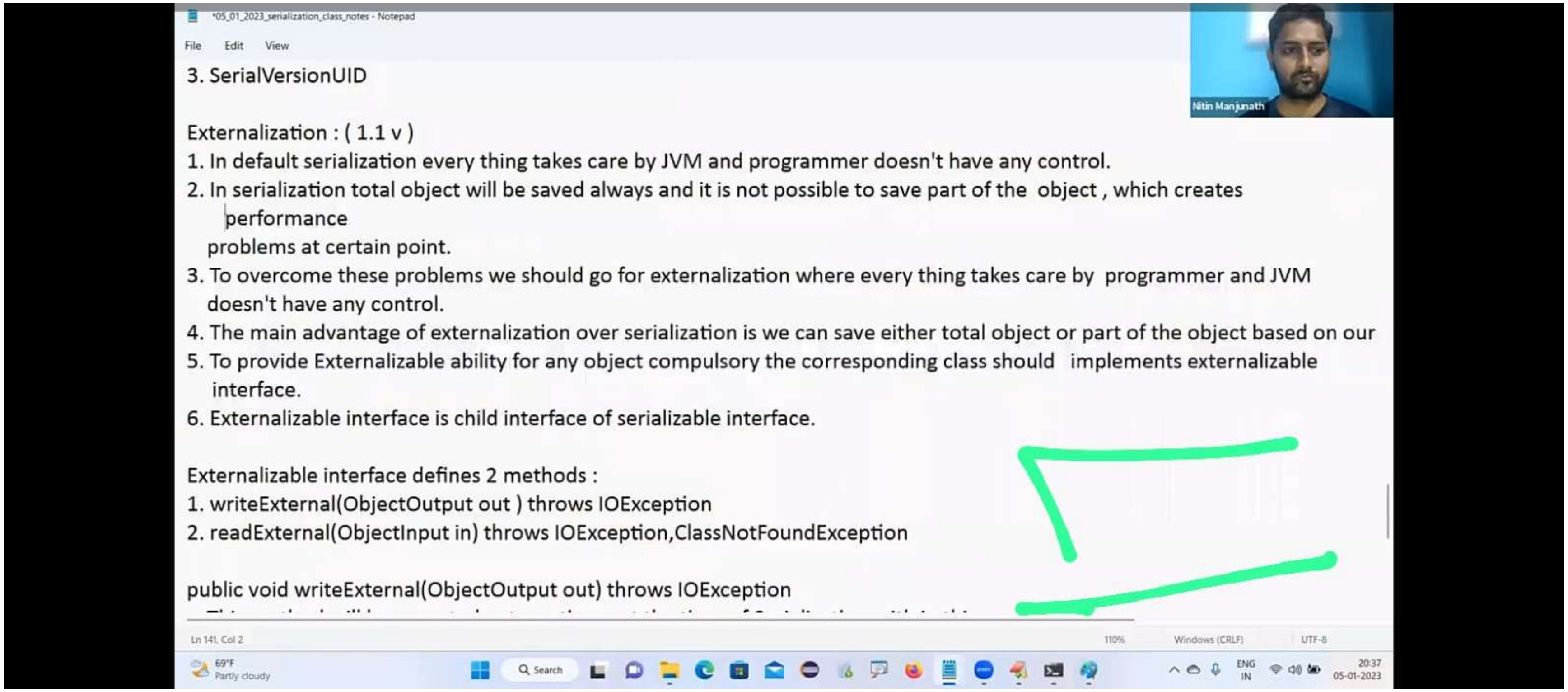
Externalizable interface defines 2 methods :
1. writeExternal(ObjectOutput out) throws IOException
2. readExternal(ObjectInput in) throws IOException,ClassNotFoundException

public void writeExternal(ObjectOutput out) throws IOException
This method will be executed automaticcay at the time of Serialization with in this method , we have to write code to save required variables to the file .

public void readExternal(ObjectInput in) throws IOException,ClassNotFoundException
This method will be executed automatically at the time of deserialization with in this method , we have to write code to save read required variable from file and assign to the current object.

At the time of deserialization JVM will create a seperate new object by executing public no-arg constructor on that object JVM
Every Externalizable class should compusory contains public no-arg constructor otherwise we will get RuntimeExcepion saying '

16



3. SerialVersionUID

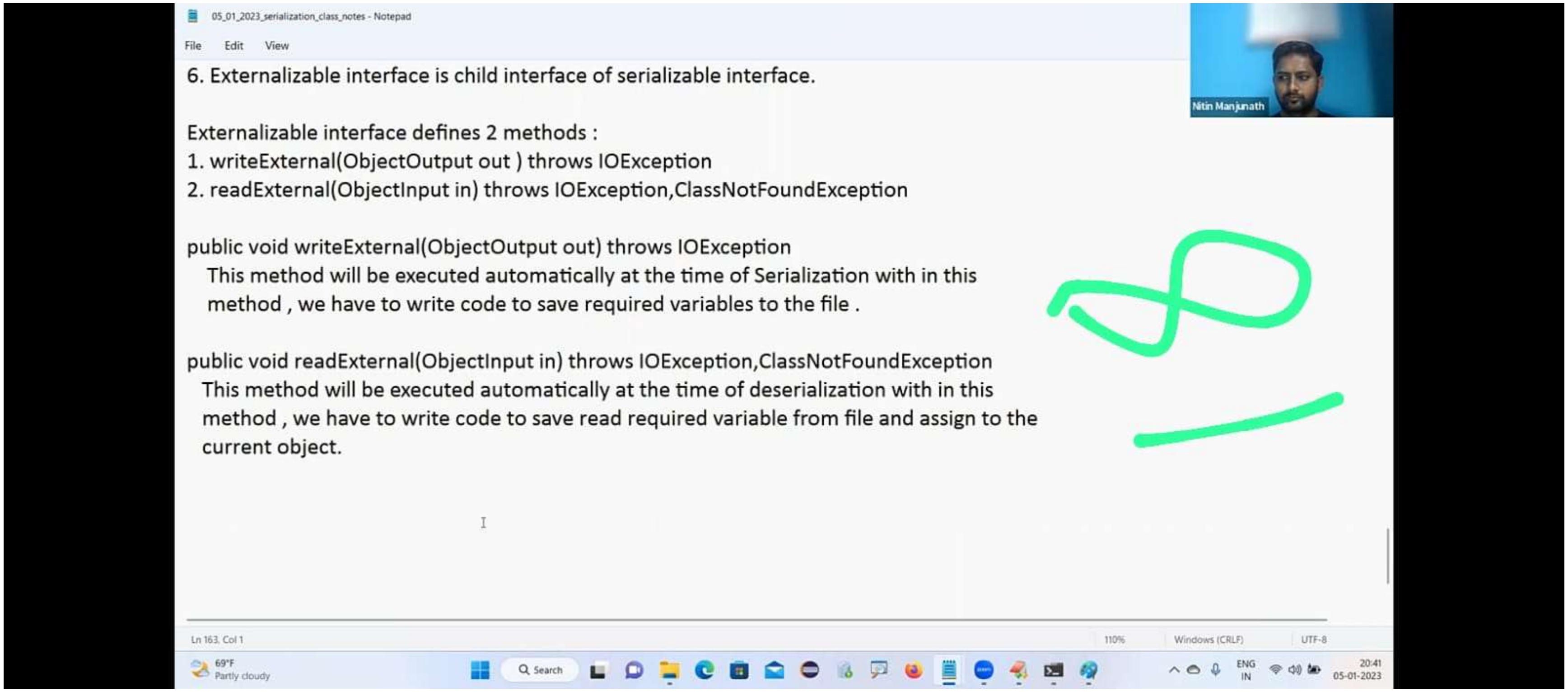
Externalization : (1.1 v)

1. In default serialization every thing takes care by JVM and programmer doesn't have any control.
2. In serialization total object will be saved always and it is not possible to save part of the object , which creates performance problems at certain point.
3. To overcome these problems we should go for externalization where every thing takes care by programmer and JVM doesn't have any control.
4. The main advantage of externalization over serialization is we can save either total object or part of the object based on our
5. To provide Externalizable ability for any object compulsory the corresponding class should implements externalizable interface.
6. Externalizable interface is child interface of serializable interface.

Externalizable interface defines 2 methods :

1. writeExternal(ObjectOutput out) throws IOException
2. readExternal(ObjectInput in) throws IOException,ClassNotFoundException

public void writeExternal(ObjectOutput out) throws IOException

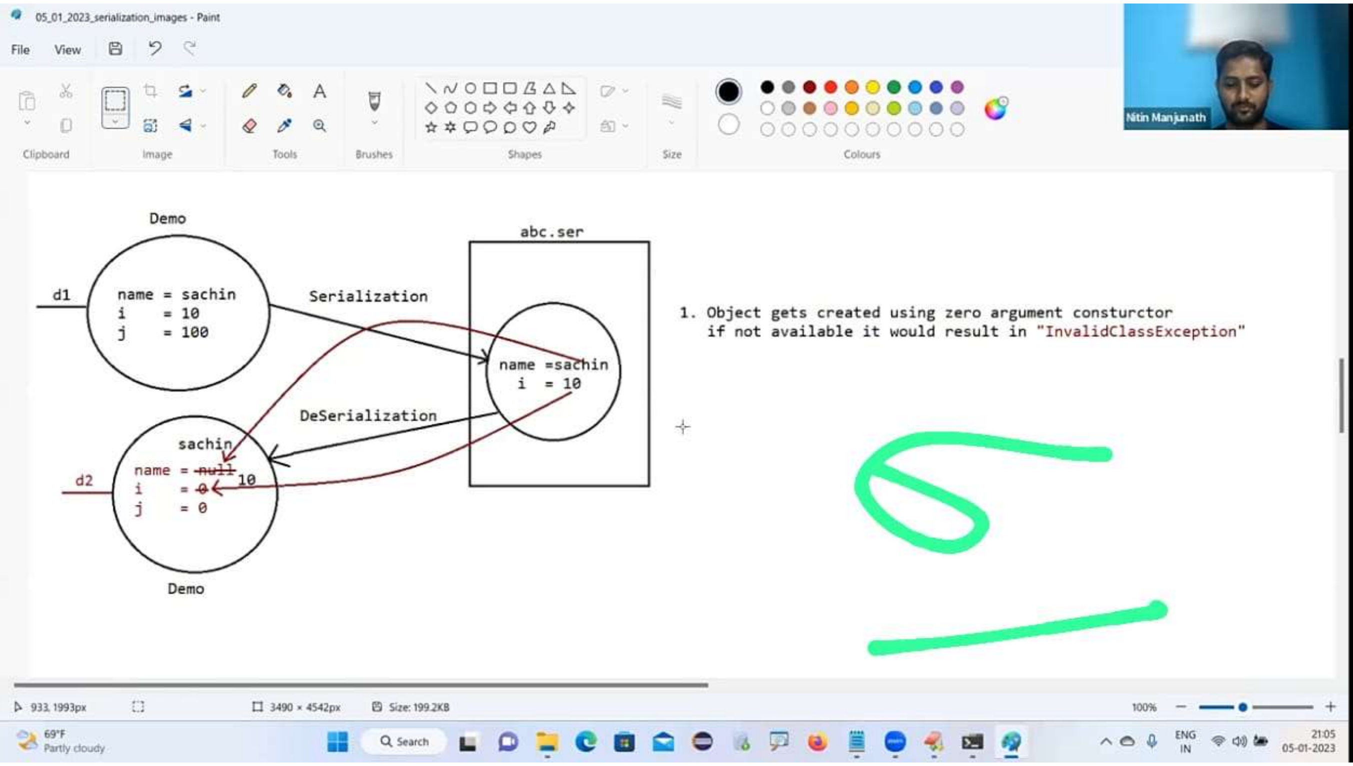


6. Externalizable interface is child interface of serializable interface.

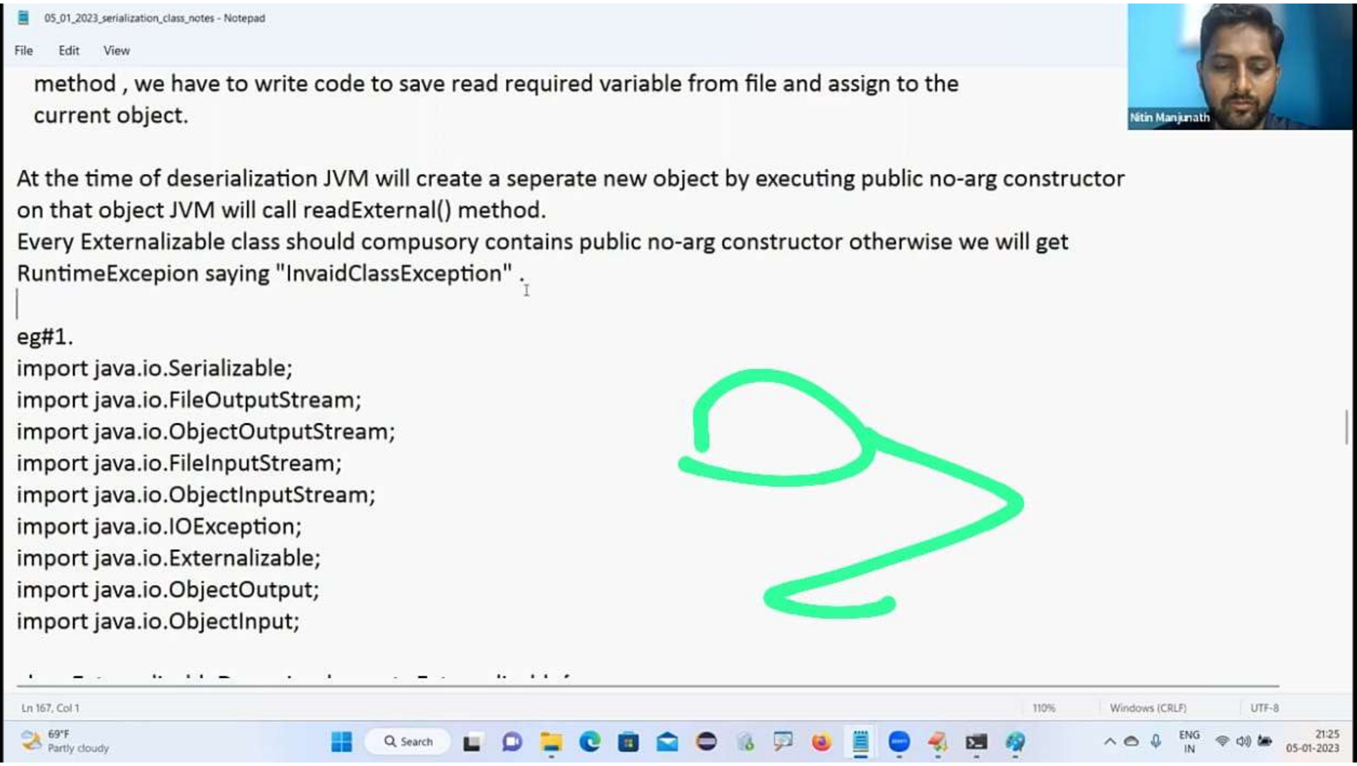
Externalizable interface defines 2 methods :
1. writeExternal(ObjectOutput out) throws IOException
2. readExternal(ObjectInput in) throws IOException,ClassNotFoundException

public void writeExternal(ObjectOutput out) throws IOException
  This method will be executed automatically at the time of Serialization with in this
  method , we have to write code to save required variables to the file .

public void readExternal(ObjectInput in) throws IOException,ClassNotFoundException
  This method will be executed automatically at the time of deserialization with in this
  method , we have to write code to save read required variable from file and assign to the
  current object.



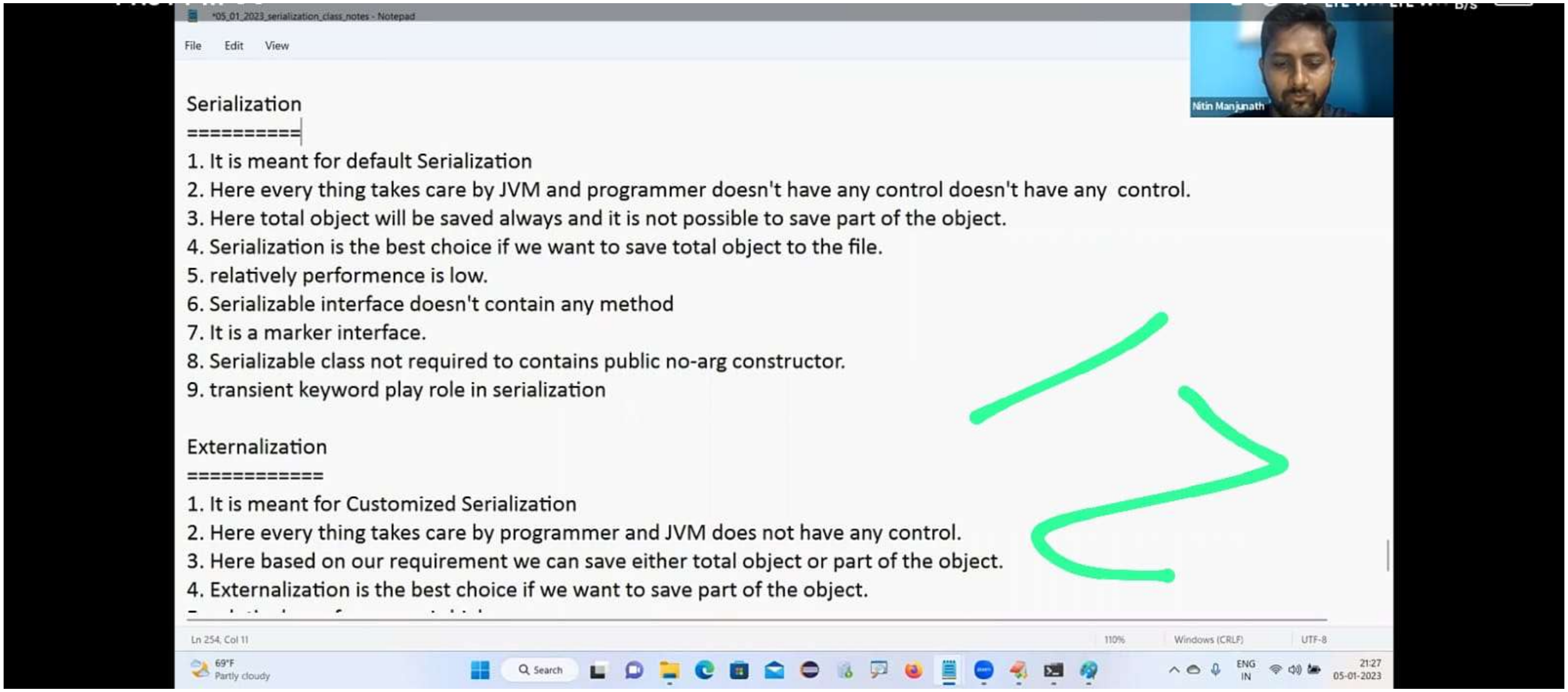
1. Object gets created using zero argument consturctor
   if not available it would result in "InvalidClassException"



method , we have to write code to save read required variable from file and assign to the current object.

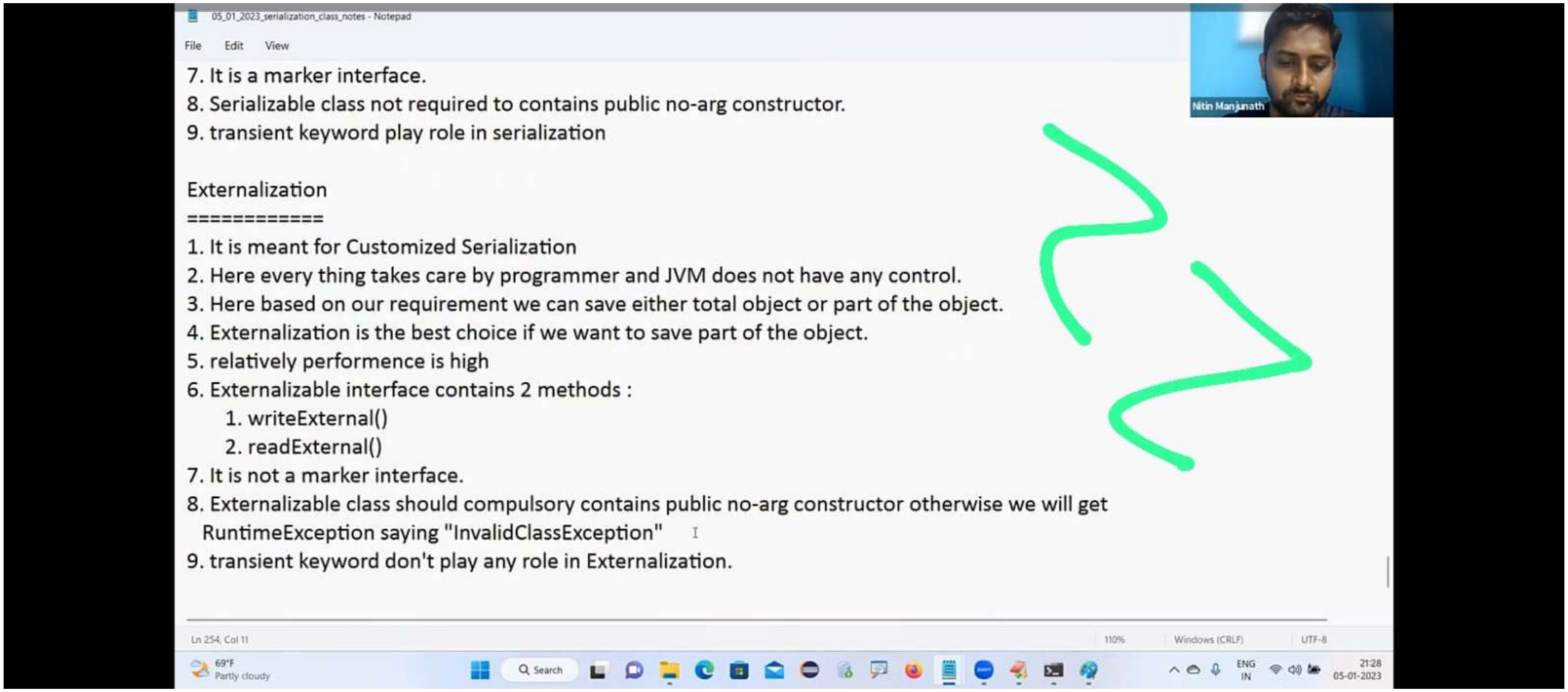
At the time of deserialization JVM will create a seperate new object by executing public no-arg constructor on that object JVM will call readExternal() method.
Every Externalizable class should compusory contains public no-arg constructor otherwise we will get RuntimeExcepion saying "InvaidClassException" .

```
eg#1.
import java.io.Serializable;
import java.io.FileOutputStream;
import java.io.ObjectOutputStream;
import java.io.FileInputStream;
import java.io.ObjectInputStream;
import java.io.IOException;
import java.io.Externalizable;
import java.io.ObjectOutput;
import java.io.ObjectInput;
```



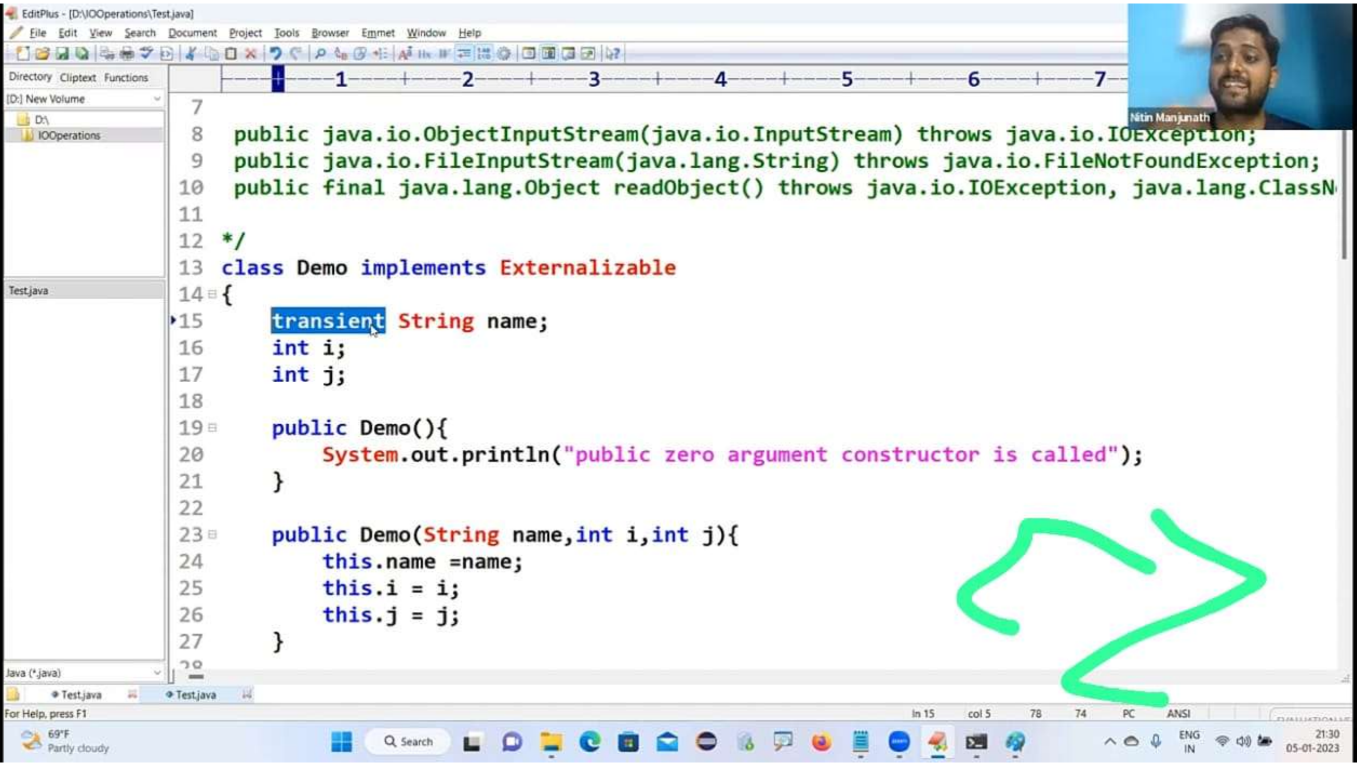
Serialization
==========
1. It is meant for default Serialization
2. Here every thing takes care by JVM and programmer doesn't have any control doesn't have any control.
3. Here total object will be saved always and it is not possible to save part of the object.
4. Serialization is the best choice if we want to save total object to the file.
5. relatively performence is low.
6. Serializable interface doesn't contain any method
7. It is a marker interface.
8. Serializable class not required to contains public no-arg constructor.
9. transient keyword play role in serialization

Externalization
============
1. It is meant for Customized Serialization
2. Here every thing takes care by programmer and JVM does not have any control.
3. Here based on our requirement we can save either total object or part of the object.
4. Externalization is the best choice if we want to save part of the object.



7. It is a marker interface.
8. Serializable class not required to contains public no-arg constructor.
9. transient keyword play role in serialization

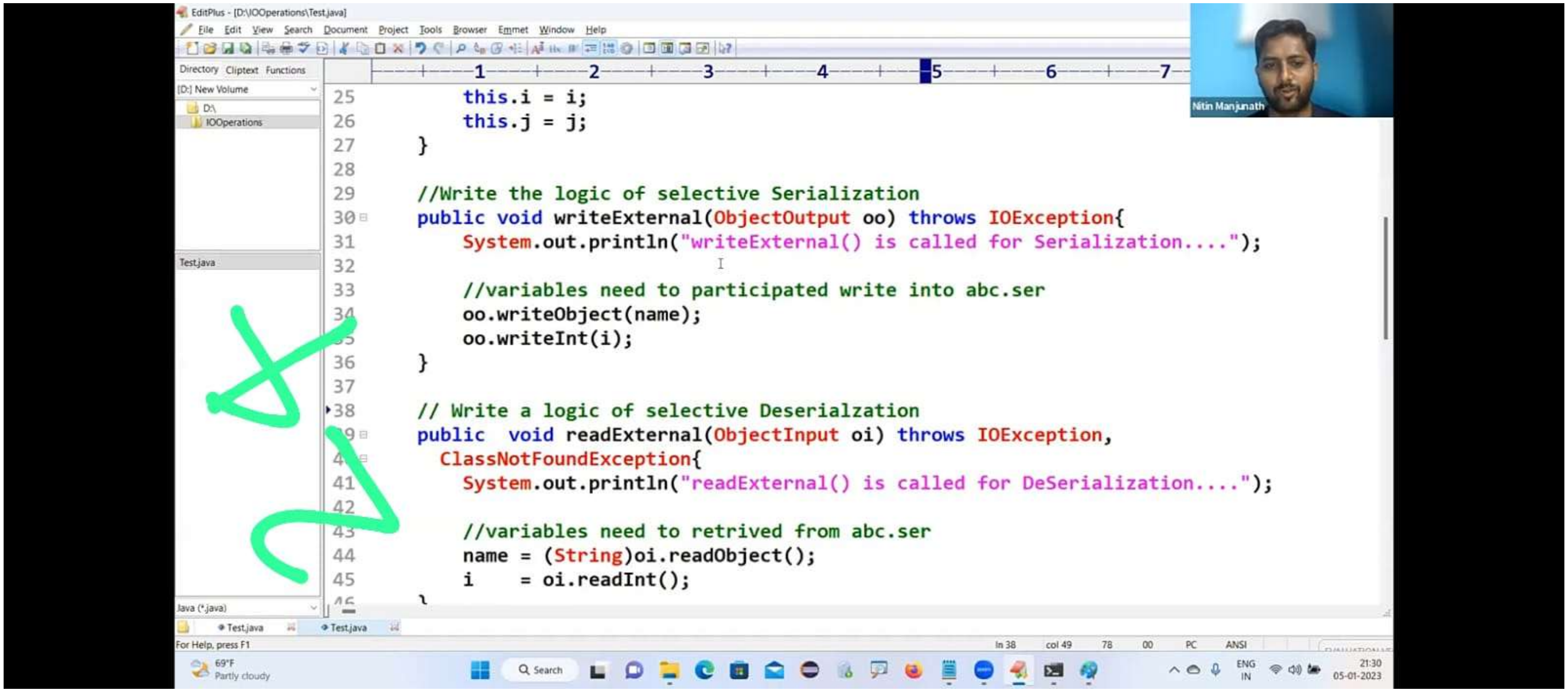
Externalization
============
1. It is meant for Customized Serialization
2. Here every thing takes care by programmer and JVM does not have any control.
3. Here based on our requirement we can save either total object or part of the object.
4. Externalization is the best choice if we want to save part of the object.
5. relatively performence is high
6. Externalizable interface contains 2 methods :
    1. writeExternal()
    2. readExternal()
7. It is not a marker interface.
8. Externalizable class should compulsory contains public no-arg constructor otherwise we will get RuntimeException saying "InvalidClassException"
9. transient keyword don't play any role in Externalization.



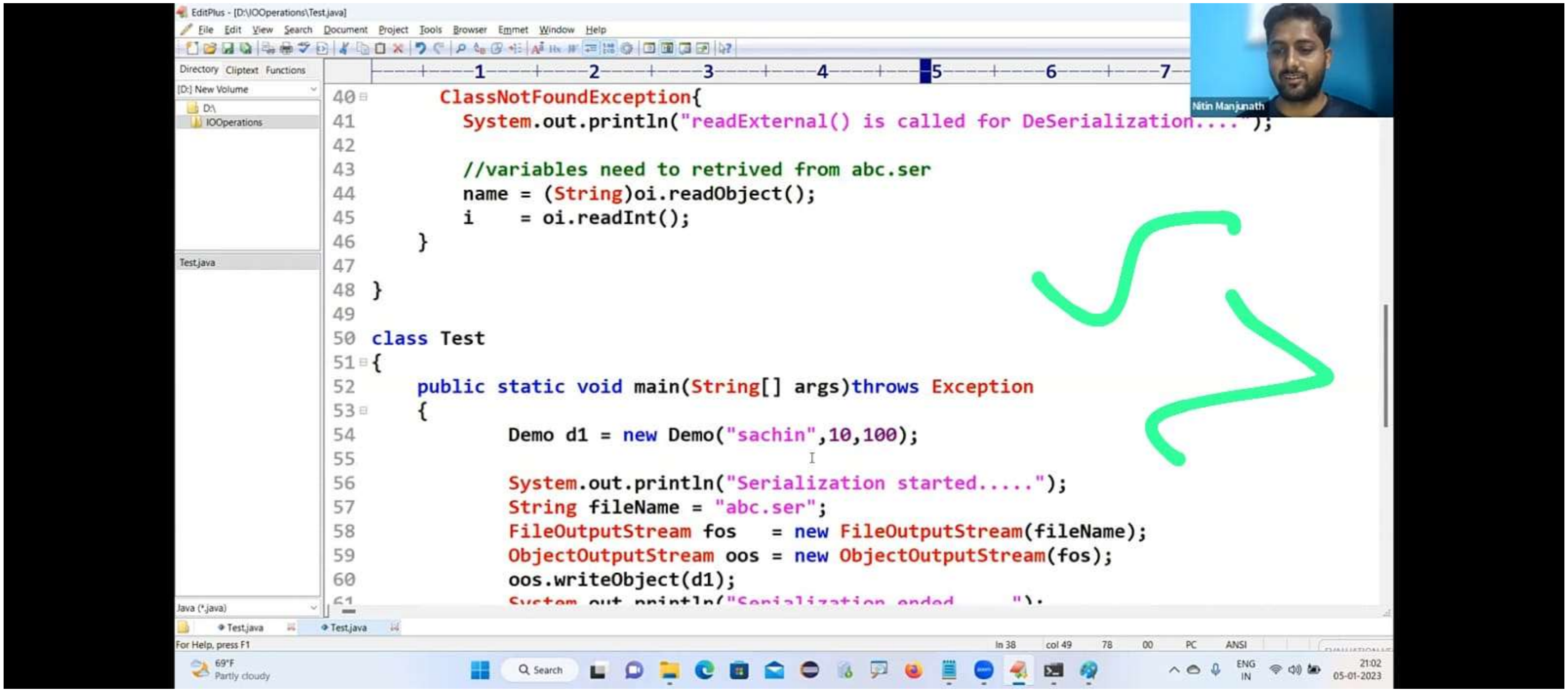
EditPlus - [D:\IOOperations\Test.java]
File Edit View Search Document Project Tools Browser Emmet Window Help
Directory Cliptext Functions
[D:] New Volume
D:\
IOOperations
Test.java
Java (*.java)
Nitin Manjunath
For Help, press F1
In 15 col 5 78 74 PC ANSI
69°F
Partly cloudy
Search
ENG
IN
21:30
05-01-2023

```
 public java.io.ObjectInputStream(java.io.InputStream) throws java.io.IOException;
 public java.io.FileInputStream(java.lang.String) throws java.io.FileNotFoundException;
 public final java.lang.Object readObject() throws java.io.IOException, java.lang.ClassN

*/
class Demo implements Externalizable
{
    transient String name;
    int i;
    int j;

    public Demo(){
        System.out.println("public zero argument constructor is called");
    }

    public Demo(String name,int i,int j){
        this.name =name;
        this.i = i;
        this.j = j;
    }
```

24

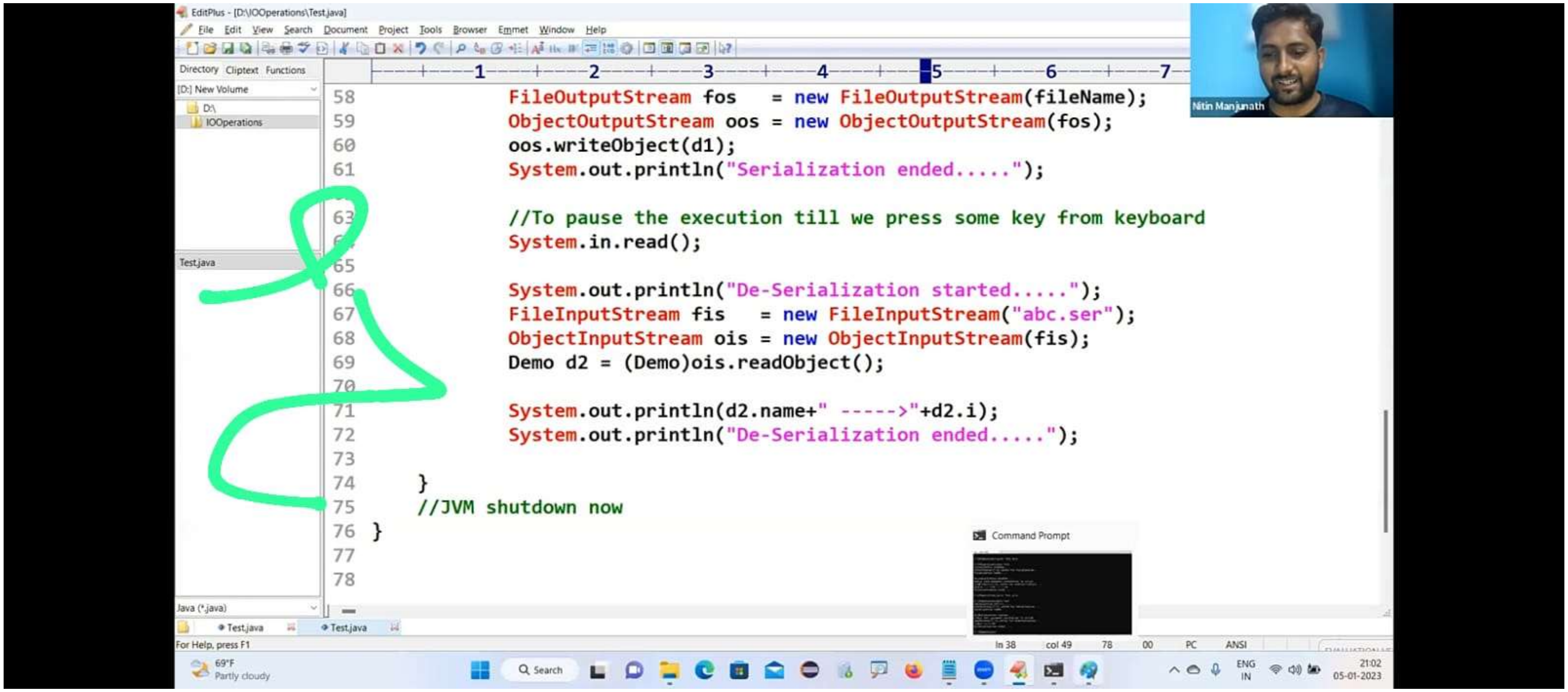

EditPlus - [D:\IOOperations\Test.java]
File Edit View Search Document Project Tools Browser Emmet Window Help
Directory Cliptext Functions
[D:] New Volume
D:\
IOOperations
Test.java
Java (*.java)
Nitin Manjunath
For Help, press F1
ln 38 col 49 78 00 PC ANSI
69°F Partly cloudy
21:30 05-01-2023

```java
        this.i = i;
        this.j = j;
    }

    //Write the logic of selective Serialization
    public void writeExternal(ObjectOutput oo) throws IOException{
        System.out.println("writeExternal() is called for Serialization....");

        //variables need to participated write into abc.ser
        oo.writeObject(name);
        oo.writeInt(i);
    }

    // Write a logic of selective Deserialzation
    public  void readExternal(ObjectInput oi) throws IOException,
      ClassNotFoundException{
        System.out.println("readExternal() is called for DeSerialization....");

        //variables need to retrived from abc.ser
        name = (String)oi.readObject();
        i    = oi.readInt();
```

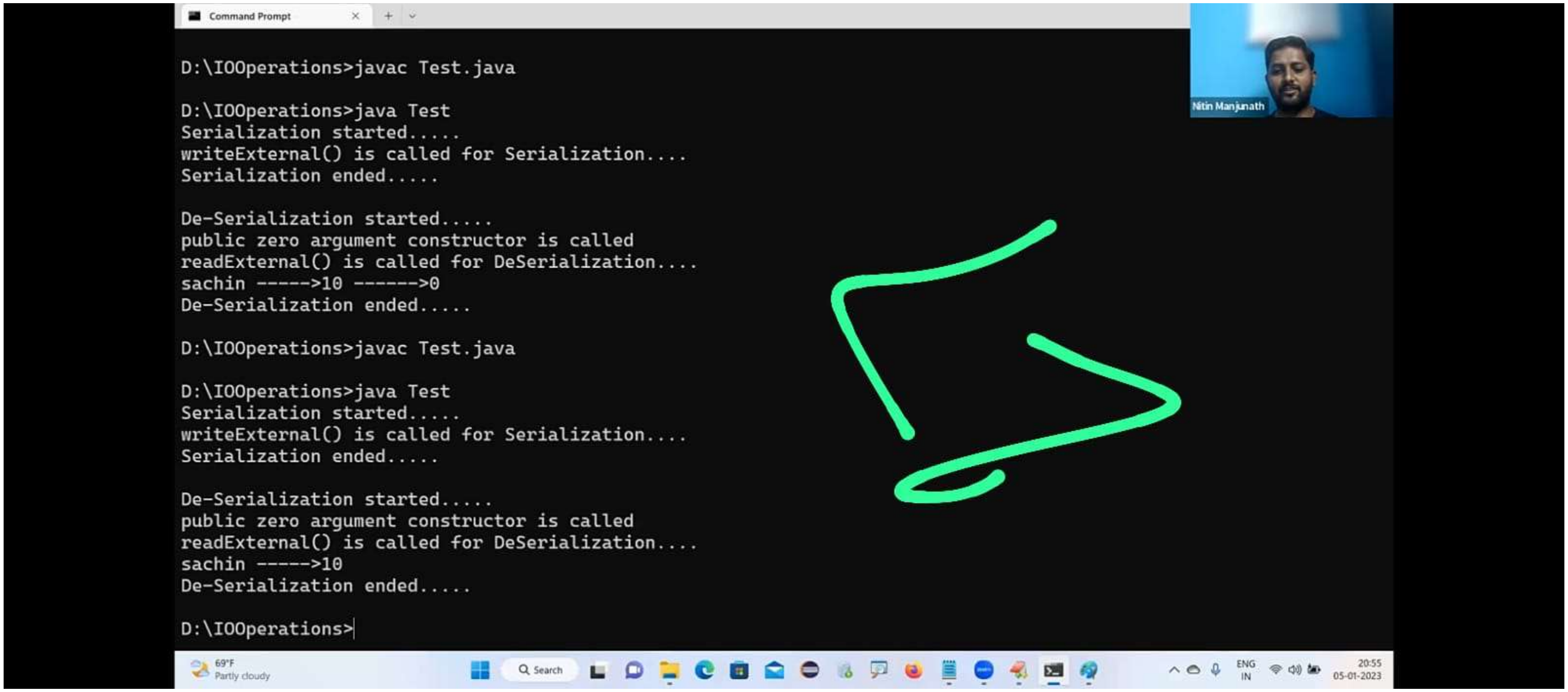


EditPlus - [D:\IOOperations\Test.java]
File Edit View Search Document Project Tools Browser Emmet Window Help
Directory Cliptext Functions
[D:] New Volume
D:\
IOOperations
Test.java
Java (*.java)
Test.java
For Help, press F1
In 38 col 49 78 00 PC ANSI
Nitin Manjunath
69°F Partly cloudy
21:02 05-01-2023
25

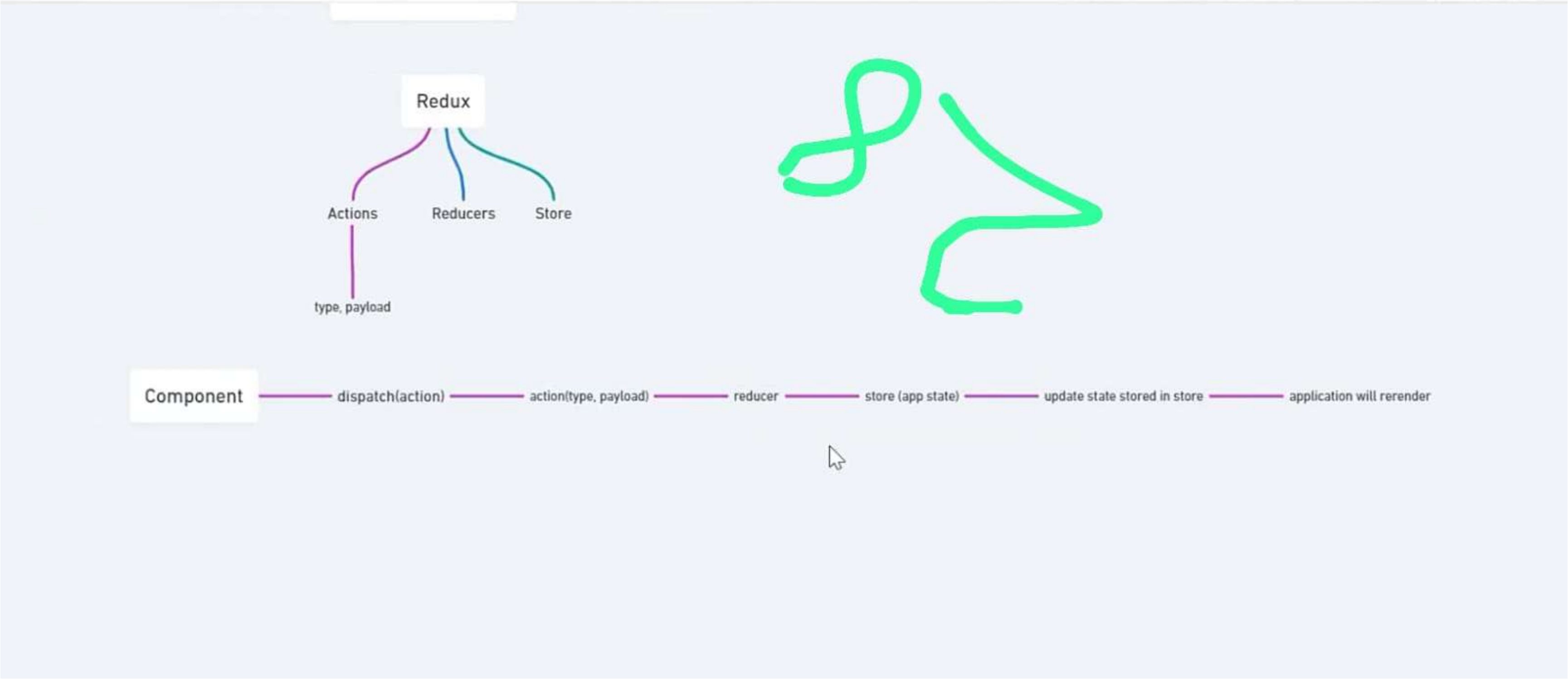
```java
        ClassNotFoundException{
          System.out.println("readExternal() is called for DeSerialization....");

          //variables need to retrived from abc.ser
          name = (String)oi.readObject();
          i    = oi.readInt();
      }

}

class Test
{
    public static void main(String[] args)throws Exception
    {
            Demo d1 = new Demo("sachin",10,100);

            System.out.println("Serialization started.....");
            String fileName = "abc.ser";
            FileOutputStream fos   = new FileOutputStream(fileName);
            ObjectOutputStream oos = new ObjectOutputStream(fos);
            oos.writeObject(d1);
            System.out.println("Serialization ended.....");
```



```java
            FileOutputStream fos   = new FileOutputStream(fileName);
            ObjectOutputStream oos = new ObjectOutputStream(fos);
            oos.writeObject(d1);
            System.out.println("Serialization ended.....");

            //To pause the execution till we press some key from keyboard
            System.in.read();

            System.out.println("De-Serialization started.....");
            FileInputStream fis   = new FileInputStream("abc.ser");
            ObjectInputStream ois = new ObjectInputStream(fis);
            Demo d2 = (Demo)ois.readObject();

            System.out.println(d2.name+" ----->"+d2.i);
            System.out.println("De-Serialization ended.....");

        }
        //JVM shutdown now
    }
```



```
D:\IOOperations>javac Test.java

D:\IOOperations>java Test
Serialization started.....
writeExternal() is called for Serialization....
Serialization ended.....

De-Serialization started.....
public zero argument constructor is called
readExternal() is called for DeSerialization....
sachin ----->10 ------>0
De-Serialization ended.....

D:\IOOperations>javac Test.java

D:\IOOperations>java Test
Serialization started.....
writeExternal() is called for Serialization....
Serialization ended.....

De-Serialization started.....
public zero argument constructor is called
readExternal() is called for DeSerialization....
sachin ----->10
De-Serialization ended.....

D:\IOOperations>
```



Redux

Actions

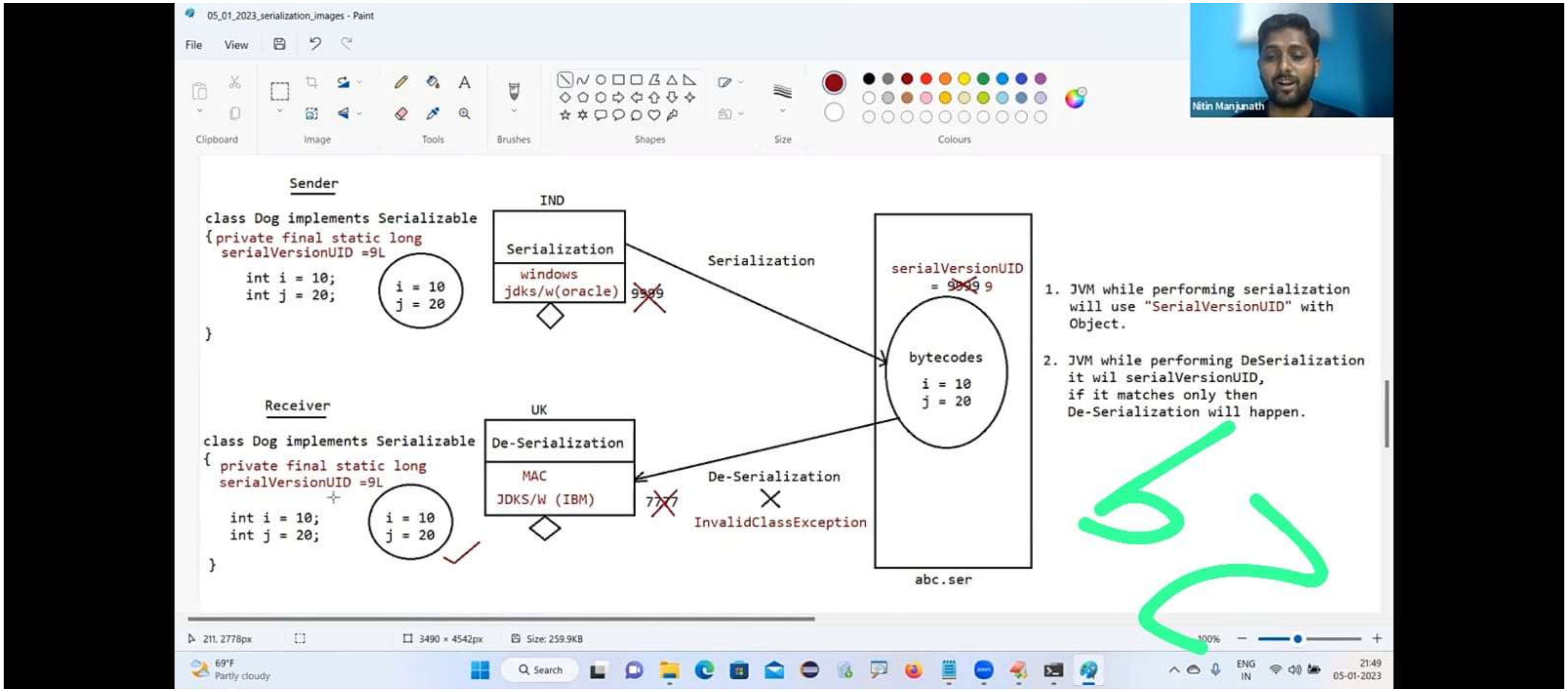
Reducers

Store

type, payload

Component — dispatch(action) — action(type, payload) — reducer — store (app state) — update state stored in store — application will rerender

28



Sender

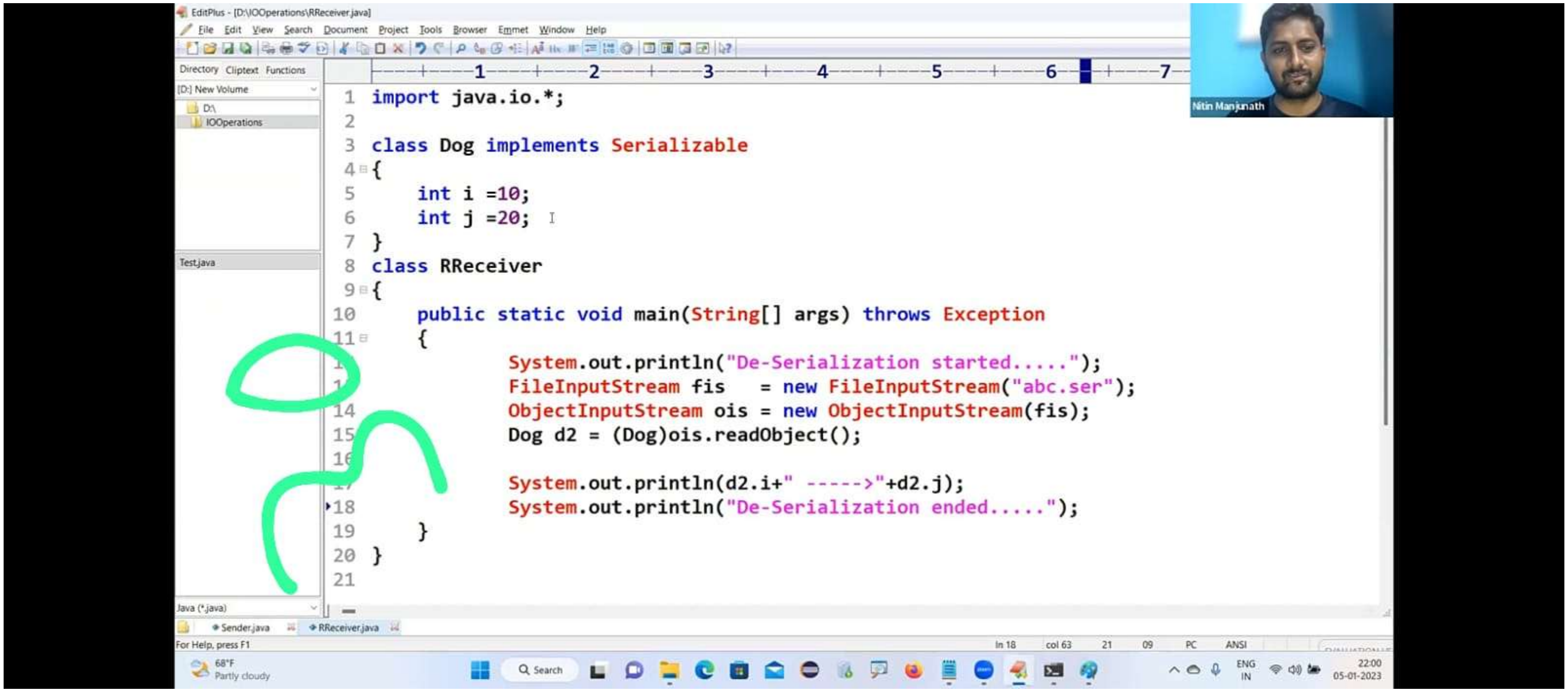
```
class Dog implements Serializable
{private final static long
  serialVersionUID =9L
    int i = 10;
    int j = 20;
}
```

Receiver

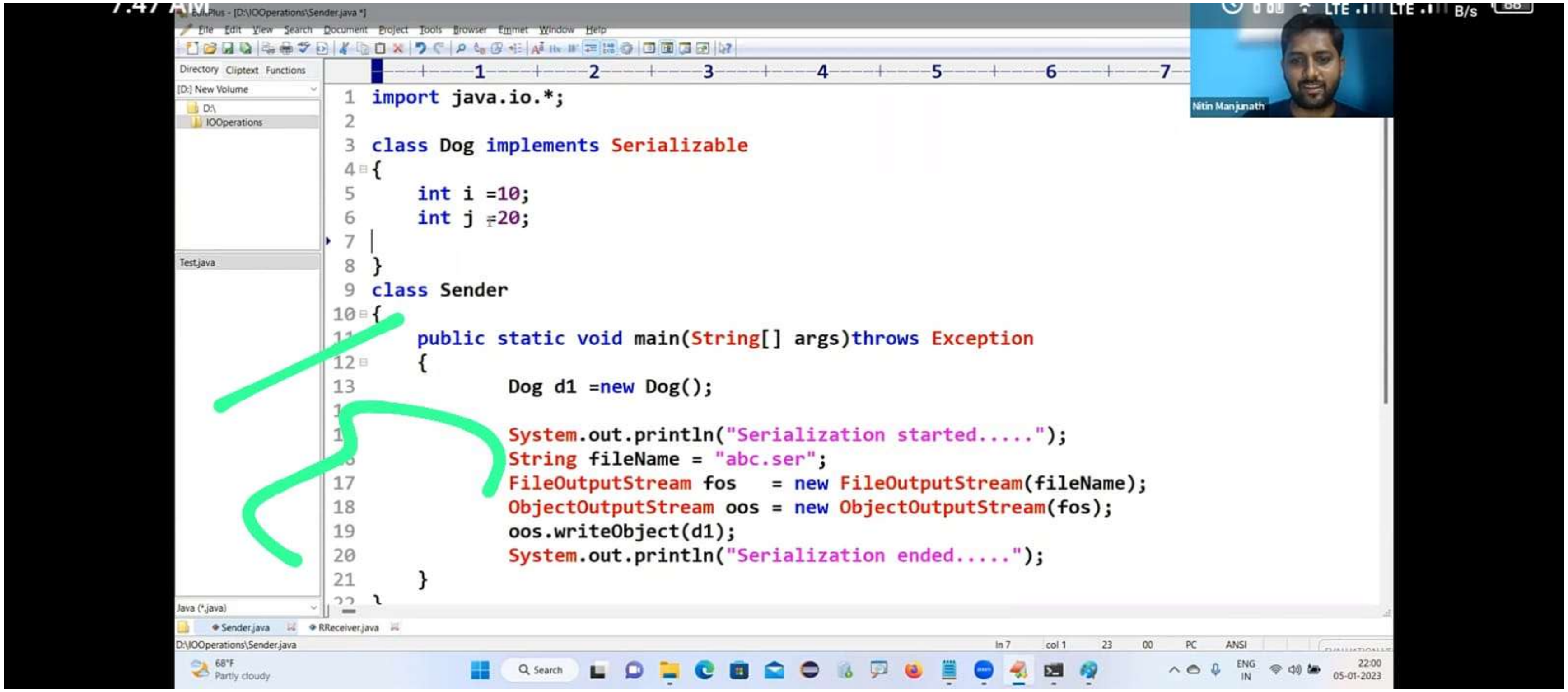
```
class Dog implements Serializable
{ private final static long
  serialVersionUID =9L
    int i = 10;
    int j = 20;
}
```

1. JVM while performing serialization will use "SerialVersionUID" with Object.

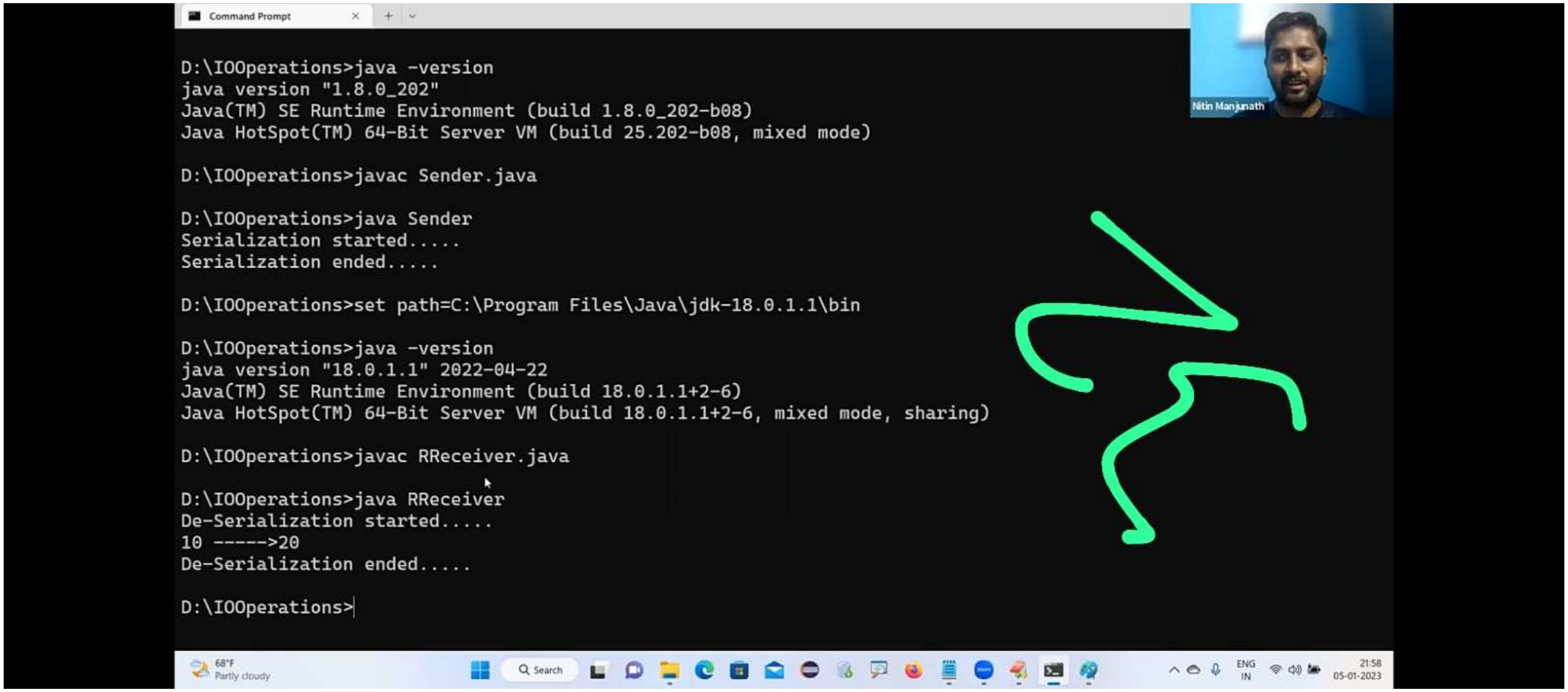
2. JVM while performing DeSerialization it wil serialVersionUID, if it matches only then De-Serialization will happen.


EditPlus - [D:\IOOperations\RReceiver.java]
File Edit View Search Document Project Tools Browser Emmet Window Help
Directory Cliptext Functions
[D:] New Volume
D:\
IOOperations
Test.java
Java (*.java)
Sender.java
RReceiver.java
For Help, press F1
In 18 col 63 21 09 PC ANSI
68°F Partly cloudy
Search
ENG IN
22:00 05-01-2023
Nitin Manjunath

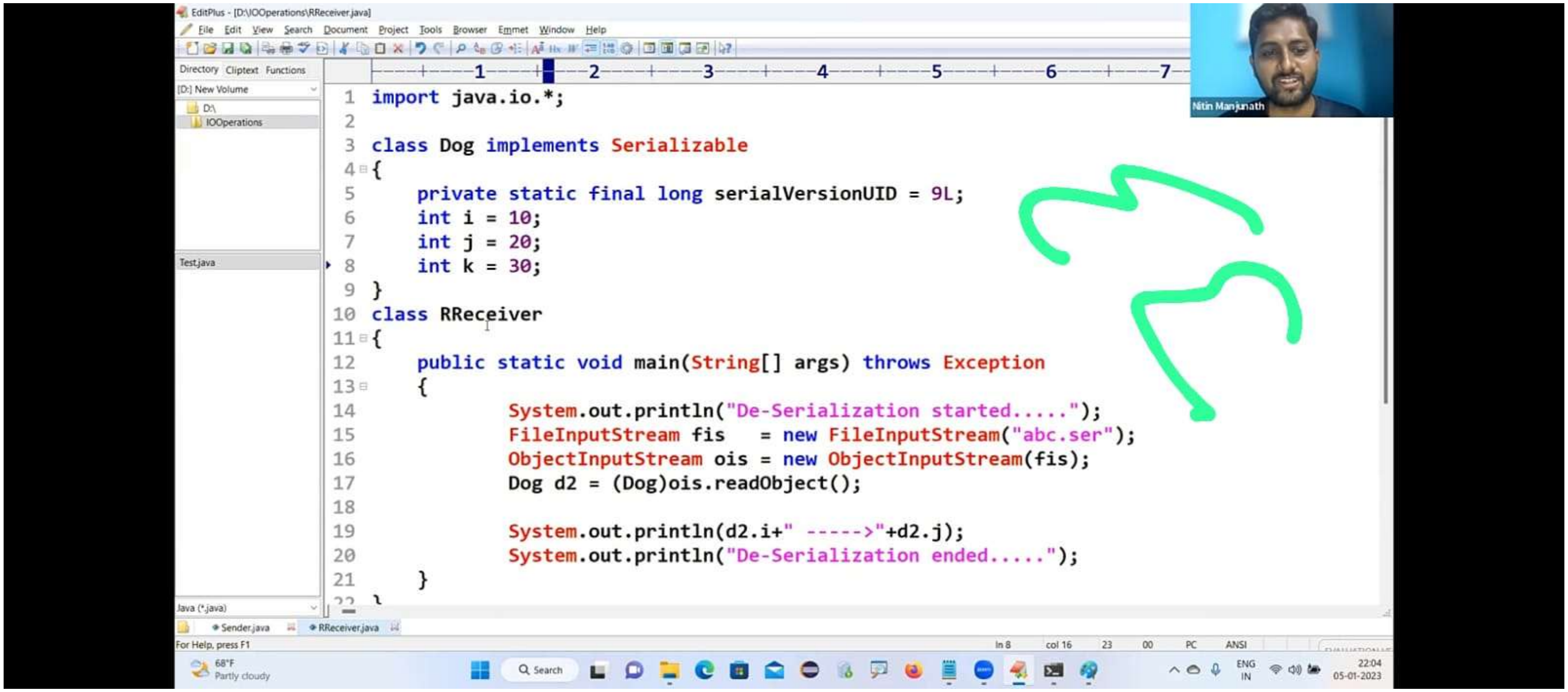
```java
import java.io.*;

class Dog implements Serializable
{
    int i =10;
    int j =20;
}
class RReceiver
{
    public static void main(String[] args) throws Exception
    {
            System.out.println("De-Serialization started.....");
            FileInputStream fis   = new FileInputStream("abc.ser");
            ObjectInputStream ois = new ObjectInputStream(fis);
            Dog d2 = (Dog)ois.readObject();

            System.out.println(d2.i+" ----->"+d2.j);
            System.out.println("De-Serialization ended.....");
    }
}
```


EditPlus - [D:\IOOperations\Sender.java *]
File Edit View Search Document Project Tools Browser Emmet Window Help
Directory Cliptext Functions
[D:] New Volume
D:\
IOOperations
Test.java
Java (*.java)
Sender.java
RReceiver.java
D:\IOOperations\Sender.java
In 7 col 1 23 00 PC ANSI
Nitin Manjunath
68°F Partly cloudy
Search
ENG IN
22:00 05-01-2023

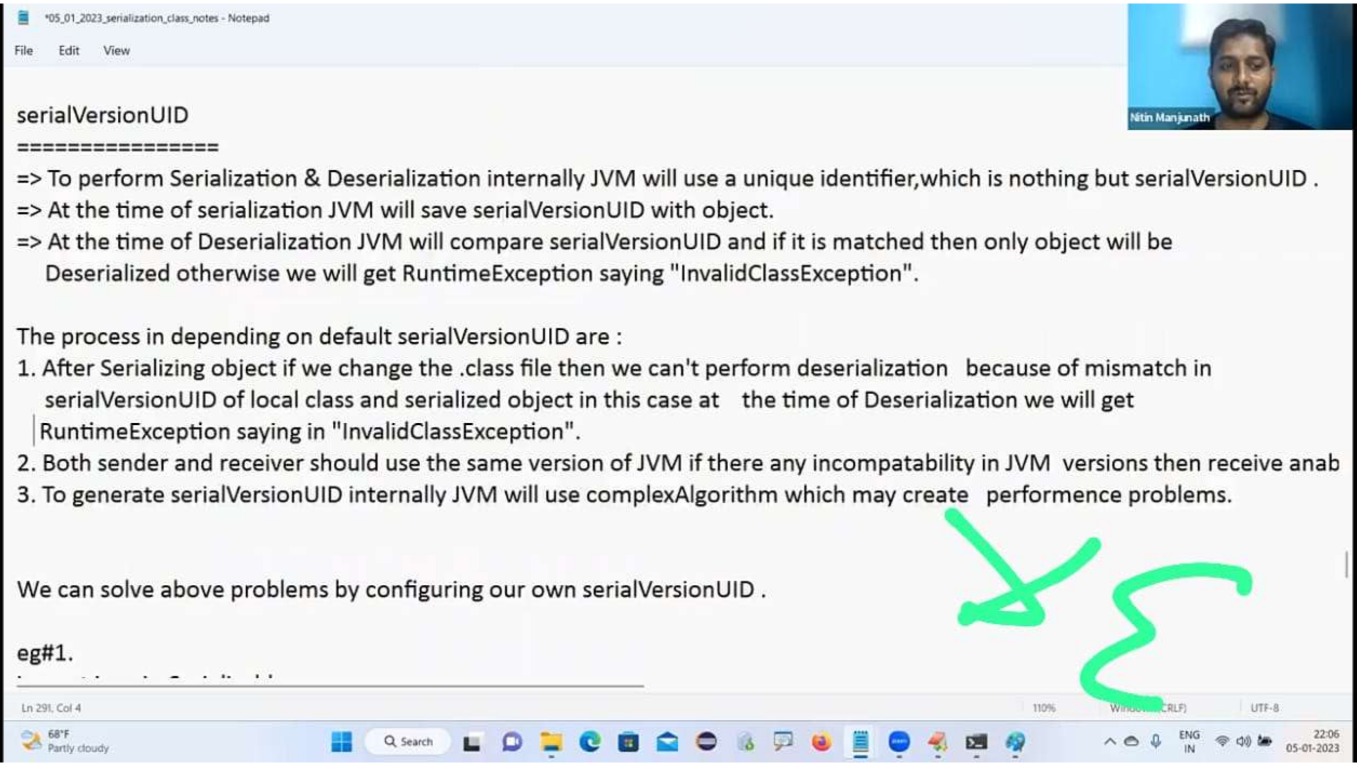
```java
import java.io.*;

class Dog implements Serializable
{
    int i =10;
    int j =20;

}
class Sender
{
    public static void main(String[] args)throws Exception
    {
            Dog d1 =new Dog();

            System.out.println("Serialization started.....");
            String fileName = "abc.ser";
            FileOutputStream fos    = new FileOutputStream(fileName);
            ObjectOutputStream oos = new ObjectOutputStream(fos);
            oos.writeObject(d1);
            System.out.println("Serialization ended.....");
    }
}
```



```
D:\IOOperations>java -version
java version "1.8.0_202"
Java(TM) SE Runtime Environment (build 1.8.0_202-b08)
Java HotSpot(TM) 64-Bit Server VM (build 25.202-b08, mixed mode)

D:\IOOperations>javac Sender.java

D:\IOOperations>java Sender
Serialization started.....
Serialization ended.....

D:\IOOperations>set path=C:\Program Files\Java\jdk-18.0.1.1\bin

D:\IOOperations>java -version
java version "18.0.1.1" 2022-04-22
Java(TM) SE Runtime Environment (build 18.0.1.1+2-6)
Java HotSpot(TM) 64-Bit Server VM (build 18.0.1.1+2-6, mixed mode, sharing)

D:\IOOperations>javac RReceiver.java

D:\IOOperations>java RReceiver
De-Serialization started.....
10 ----->20
De-Serialization ended.....

D:\IOOperations>
```


EditPlus - [D:\IOOperations\RReceiver.java]
File Edit View Search Document Project Tools Browser Emmet Window Help
Directory Cliptext Functions
[D:] New Volume
D:\
IOOperations
Test.java
Java (*.java)
Sender.java
RReceiver.java
For Help, press F1
In 8 col 16 23 00 PC ANSI
68°F Partly cloudy
Search
ENG IN
22:04 05-01-2023
Nitin Manjunath

```java
import java.io.*;

class Dog implements Serializable
{
    private static final long serialVersionUID = 9L;
    int i = 10;
    int j = 20;
    int k = 30;
}
class RReceiver
{
    public static void main(String[] args) throws Exception
    {
            System.out.println("De-Serialization started.....");
            FileInputStream fis   = new FileInputStream("abc.ser");
            ObjectInputStream ois = new ObjectInputStream(fis);
            Dog d2 = (Dog)ois.readObject();

            System.out.println(d2.i+" ----->"+d2.j);
            System.out.println("De-Serialization ended.....");
    }
}
```

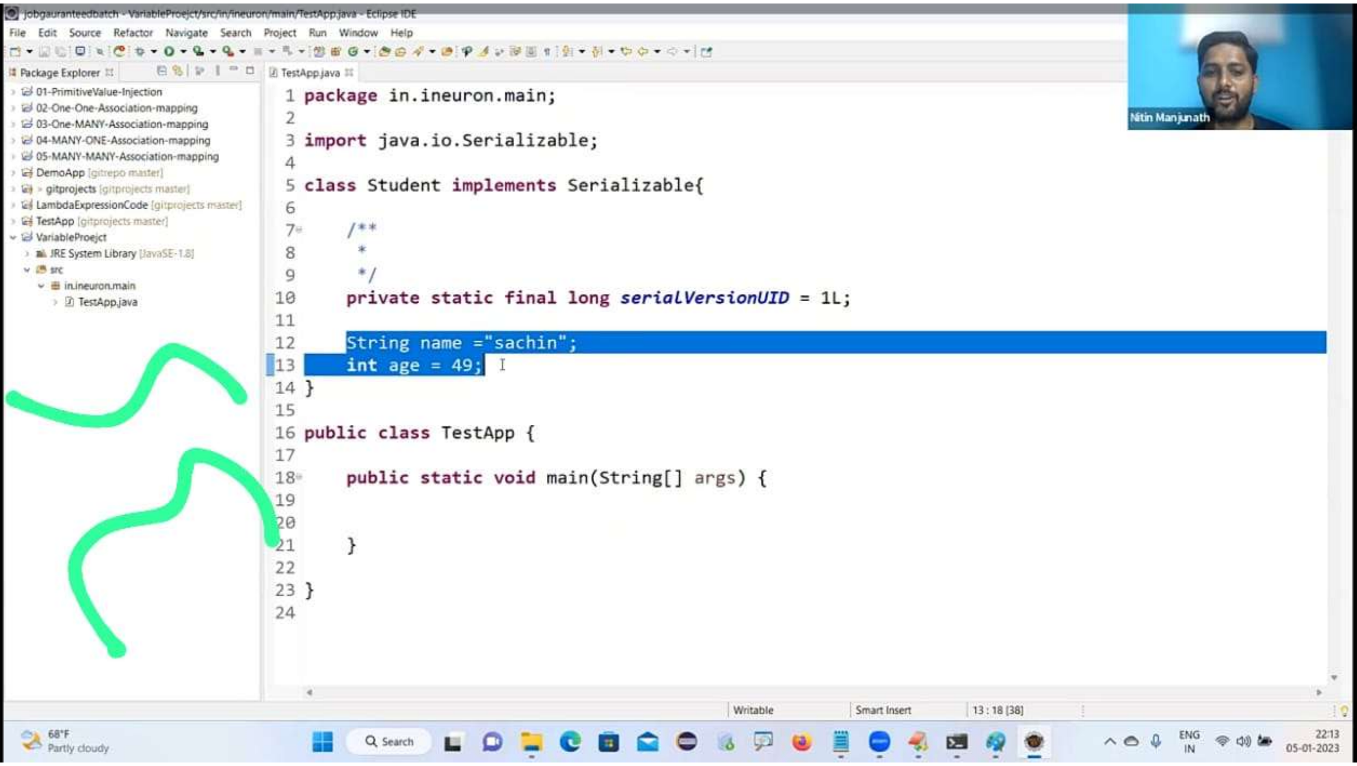


serialVersionUID
================
=> To perform Serialization & Deserialization internally JVM will use a unique identifier,which is nothing but serialVersionUID .
=> At the time of serialization JVM will save serialVersionUID with object.
=> At the time of Deserialization JVM will compare serialVersionUID and if it is matched then only object will be
   Deserialized otherwise we will get RuntimeException saying "InvalidClassException".

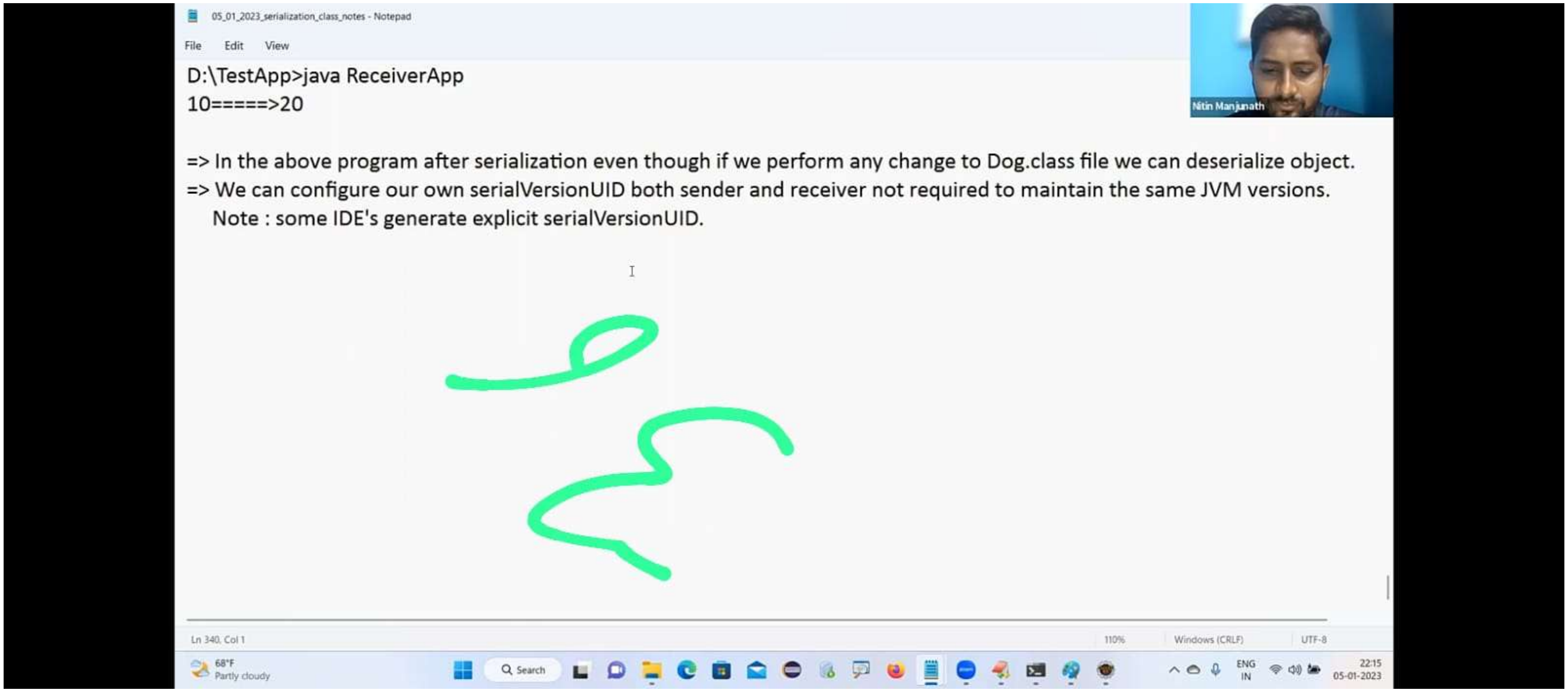
The process in depending on default serialVersionUID are :
1. After Serializing object if we change the .class file then we can't perform deserialization   because of mismatch in
   serialVersionUID of local class and serialized object in this case at   the time of Deserialization we will get
  RuntimeException saying in "InvalidClassException".
2. Both sender and receiver should use the same version of JVM if there any incompatability in JVM versions then receive anab
3. To generate serialVersionUID internally JVM will use complexAlgorithm which may create   performence problems.

We can solve above problems by configuring our own serialVersionUID .

eg#1.

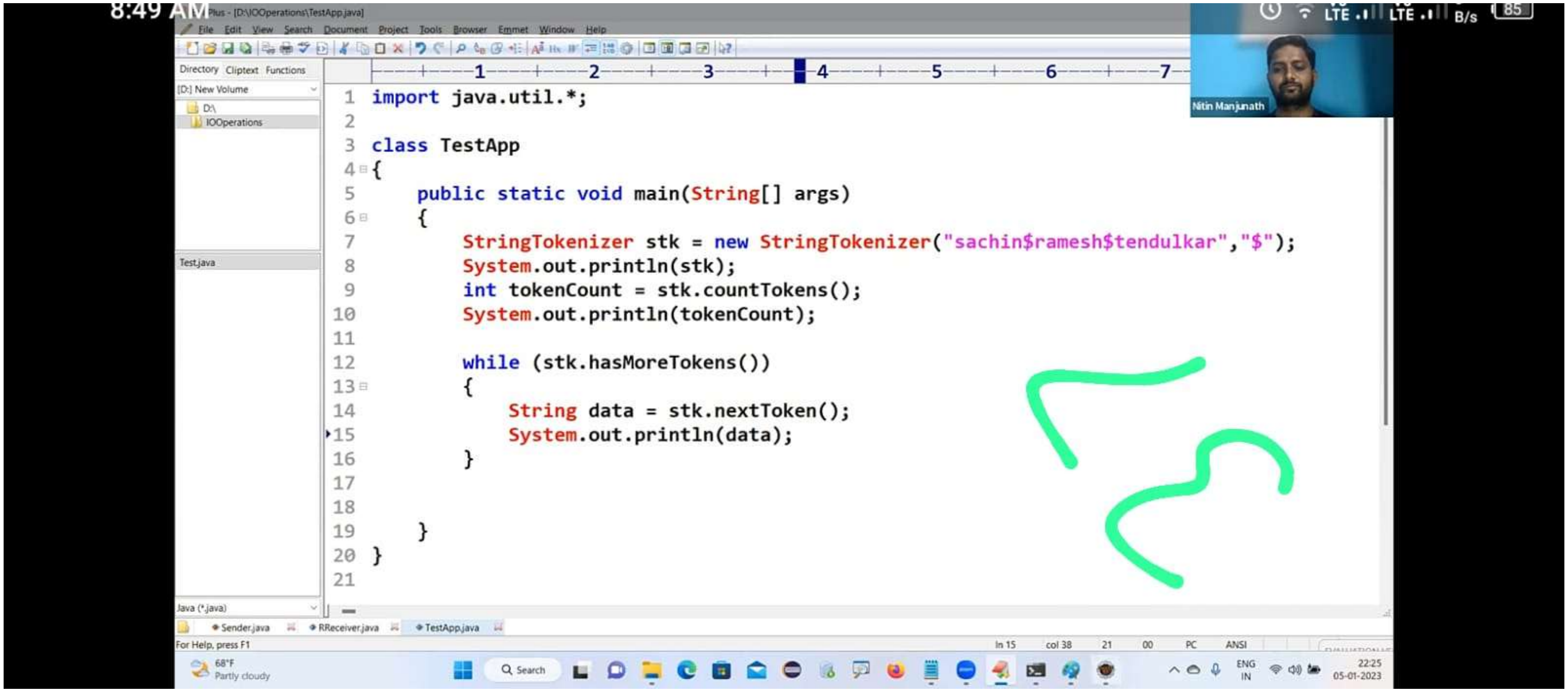

jobgauranteedbatch - VariableProejct/src/in/ineuron/main/TestApp.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
01-PrimitiveValue-Injection
02-One-One-Association-mapping
03-One-MANY-Association-mapping
04-MANY-ONE-Association-mapping
05-MANY-MANY-Association-mapping
DemoApp [gitrepo master]
> gitprojects [gitprojects master]
LambdaExpressionCode [gitprojects master]
TestApp [gitprojects master]
VariableProejct
JRE System Library [JavaSE-1.8]
src
in.ineuron.main
TestApp.java
TestApp.java
Nitin Manjunath
Writable
Smart Insert
13 : 18 [38]
68°F
Partly cloudy
Search
ENG
IN
22:13
05-01-2023
35

```java
package in.ineuron.main;

import java.io.Serializable;

class Student implements Serializable{

	/**
	 * 
	 */
	private static final long serialVersionUID = 1L;

	String name ="sachin";
	int age = 49;
}

public class TestApp {

	public static void main(String[] args) {


	}

}

```

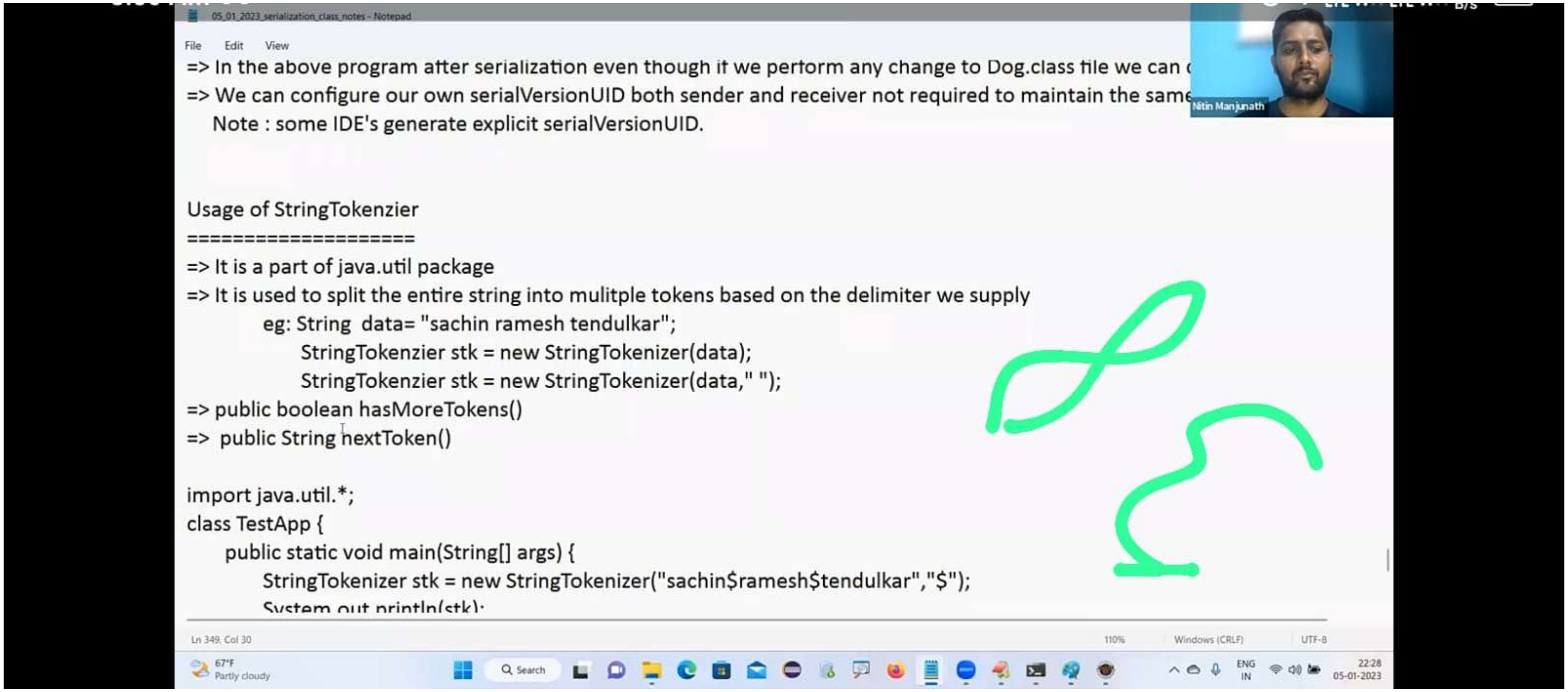


```
D:\TestApp>java ReceiverApp
10=====>20
```

=> In the above program after serialization even though if we perform any change to Dog.class file we can deserialize object.
=> We can configure our own serialVersionUID both sender and receiver not required to maintain the same JVM versions.
Note : some IDE's generate explicit serialVersionUID.



```java
import java.util.*;

class TestApp
{
    public static void main(String[] args)
    {
        StringTokenizer stk = new StringTokenizer("sachin$ramesh$tendulkar","$");
        System.out.println(stk);
        int tokenCount = stk.countTokens();
        System.out.println(tokenCount);

        while (stk.hasMoreTokens())
        {
            String data = stk.nextToken();
            System.out.println(data);
        }


    }
}
```



=> In the above program after serialization even though if we perform any change to Dog.class file we can c
=> We can configure our own serialVersionUID both sender and receiver not required to maintain the same
   Note : some IDE's generate explicit serialVersionUID.

```
Usage of StringTokenzier
====================
=> It is a part of java.util package
=> It is used to split the entire string into mulitple tokens based on the delimiter we supply
        eg: String  data= "sachin ramesh tendulkar";
            StringTokenzier stk = new StringTokenizer(data);
            StringTokenzier stk = new StringTokenizer(data," ");
=> public boolean hasMoreTokens()
=>  public String nextToken()

import java.util.*;
class TestApp {
    public static void main(String[] args) {
        StringTokenizer stk = new StringTokenizer("sachin$ramesh$tendulkar","$");
        System.out.println(stk);
```