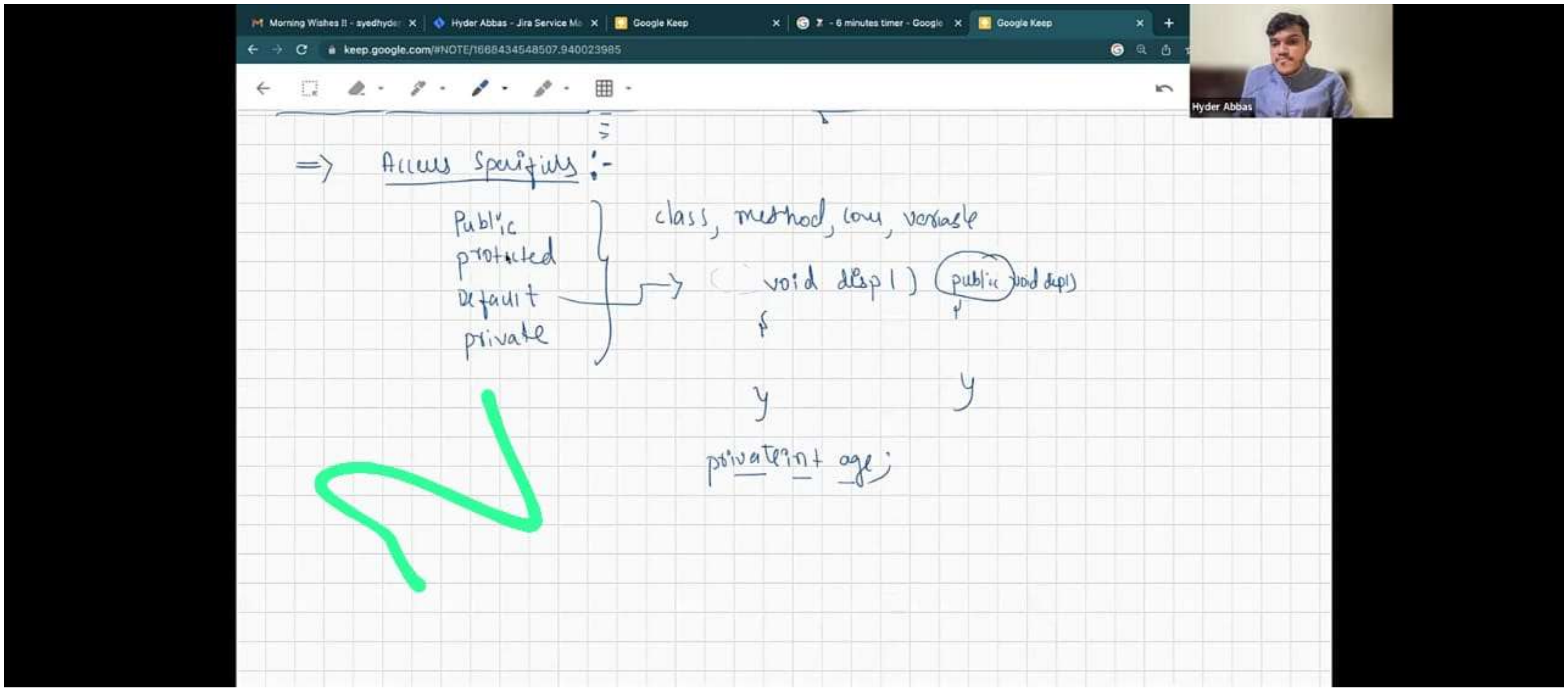
Java Inheritance

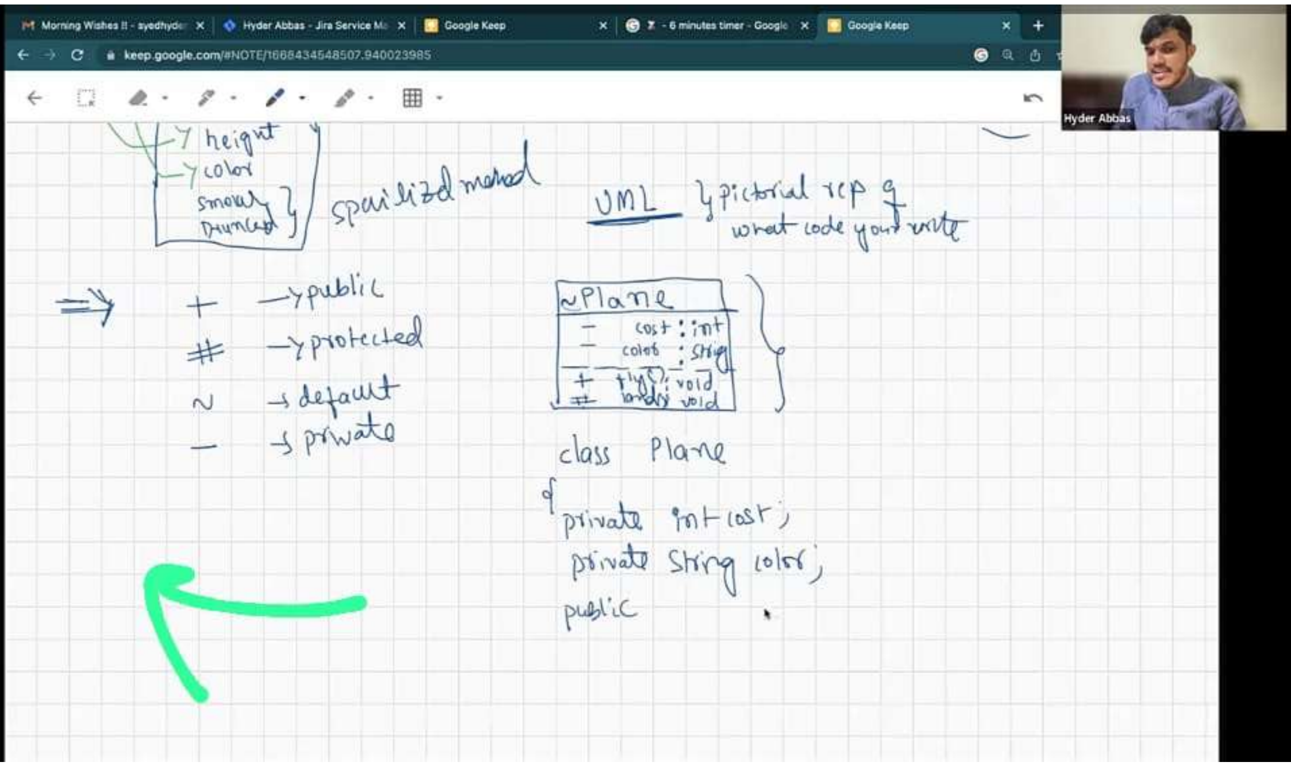


=> Access Specifiers :-

Public
protected
Default
private

class, method, cons, variable

```
void disp( )          public void disp( )
{                     {

}                     }

private int age;
```


Morning Wishes !! - syedhyder
Hyder Abbas - Jira Service M
Google Keep
6 minutes timer - Google
Google Keep
keep.google.com/#NOTE/1668434548507.940023985
Hyder Abbas

→ height
→ color
smoking }
Drunk } spailizd mehod

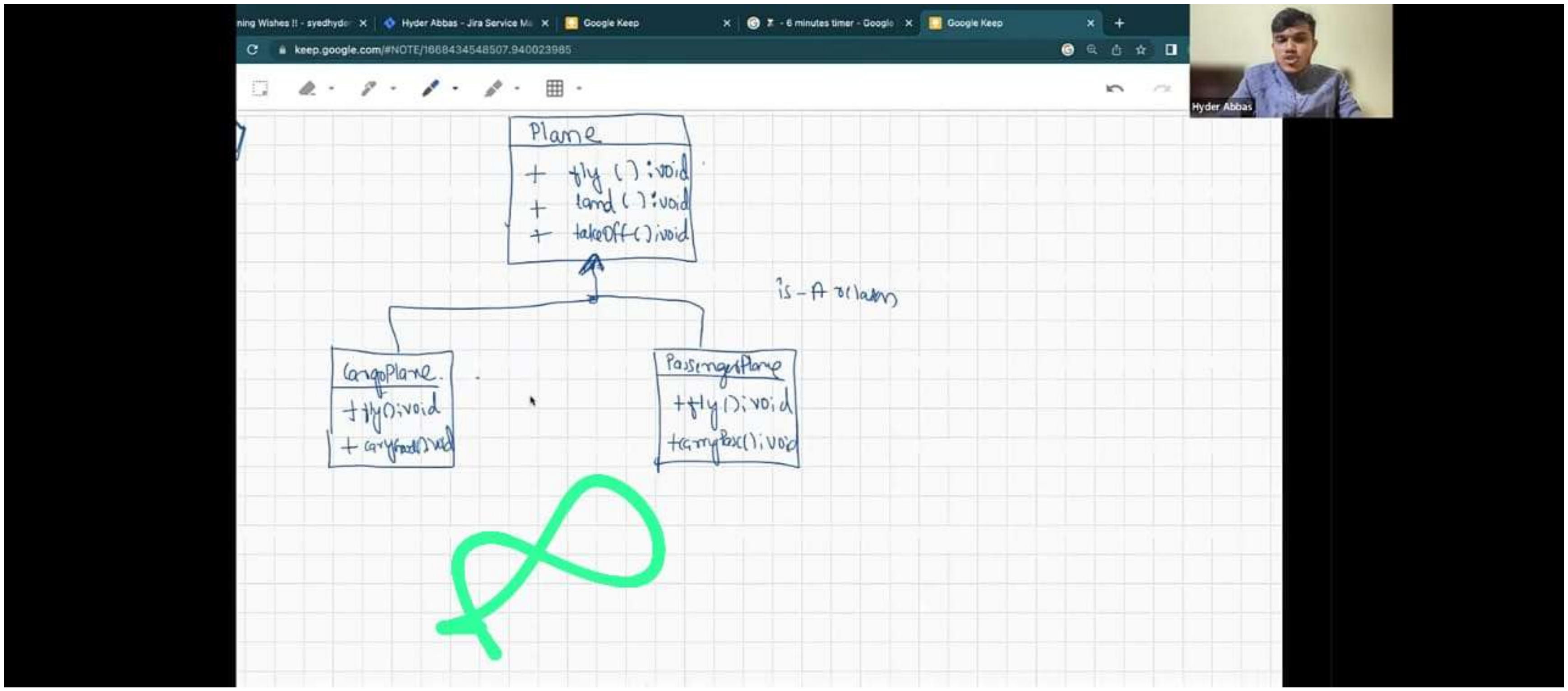
UML } pictorial rep of what code your write

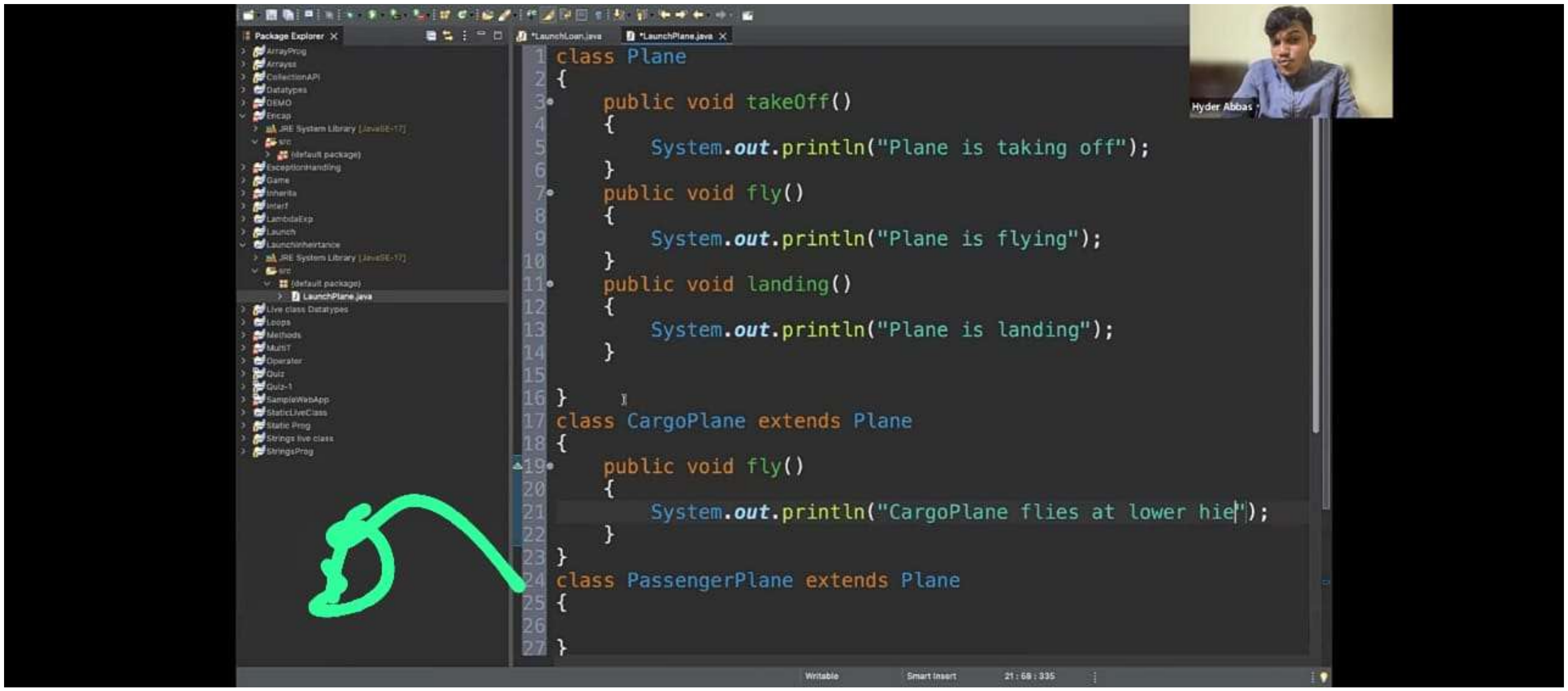
⟹ + → public
# → protected
~ → default
− → private

~Plane

−	cost	: int
−	color	: String
+	fly()	: void
\#	landing	void

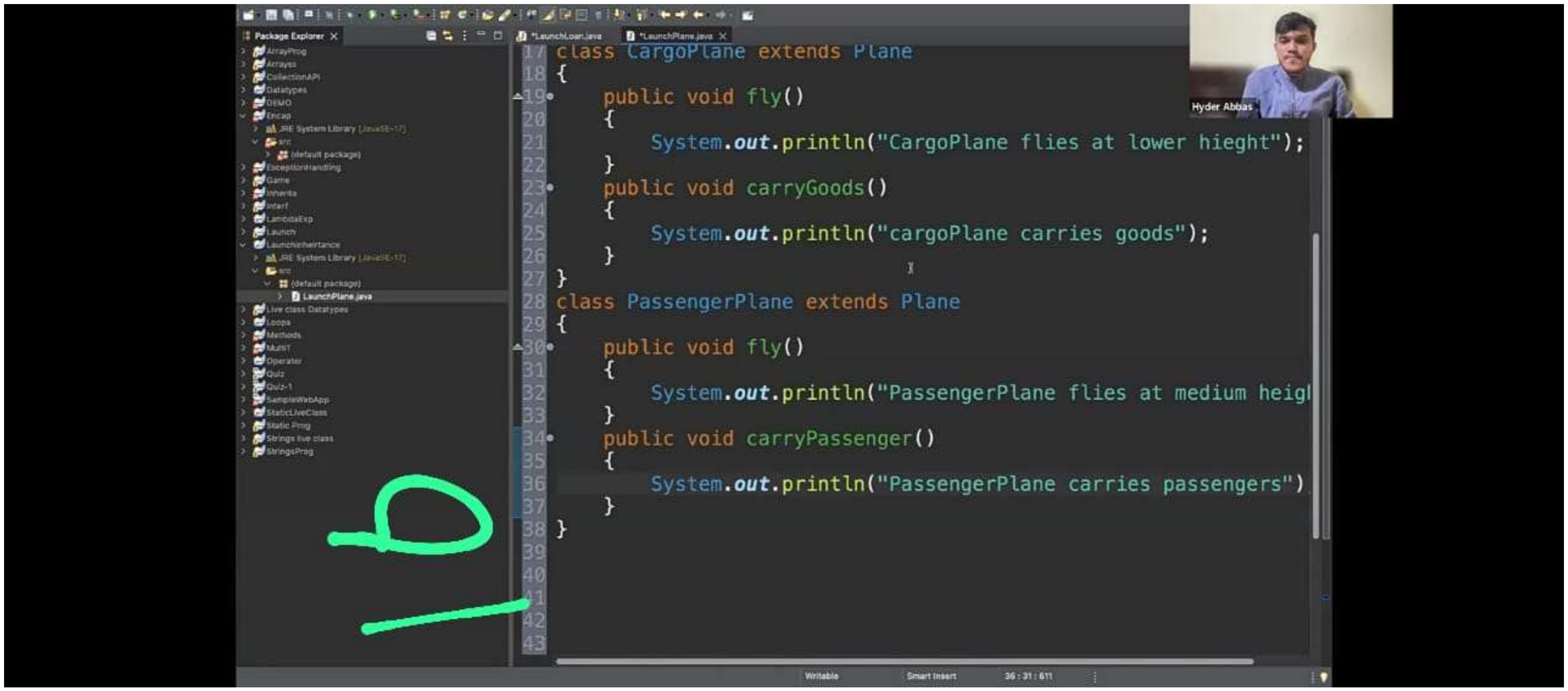
```
class Plane
{
 private int cost;
 private String color;
 public
```


Plane
+ fly ( ) : void
+ land ( ) : void
+ takeOff ( ) : void
is - A relation
CargoPlane
+ fly() : void
+ carryGoods() : void
PassengerPlane
+ fly() : void
+ carryPax() : void
Hyder Abbas

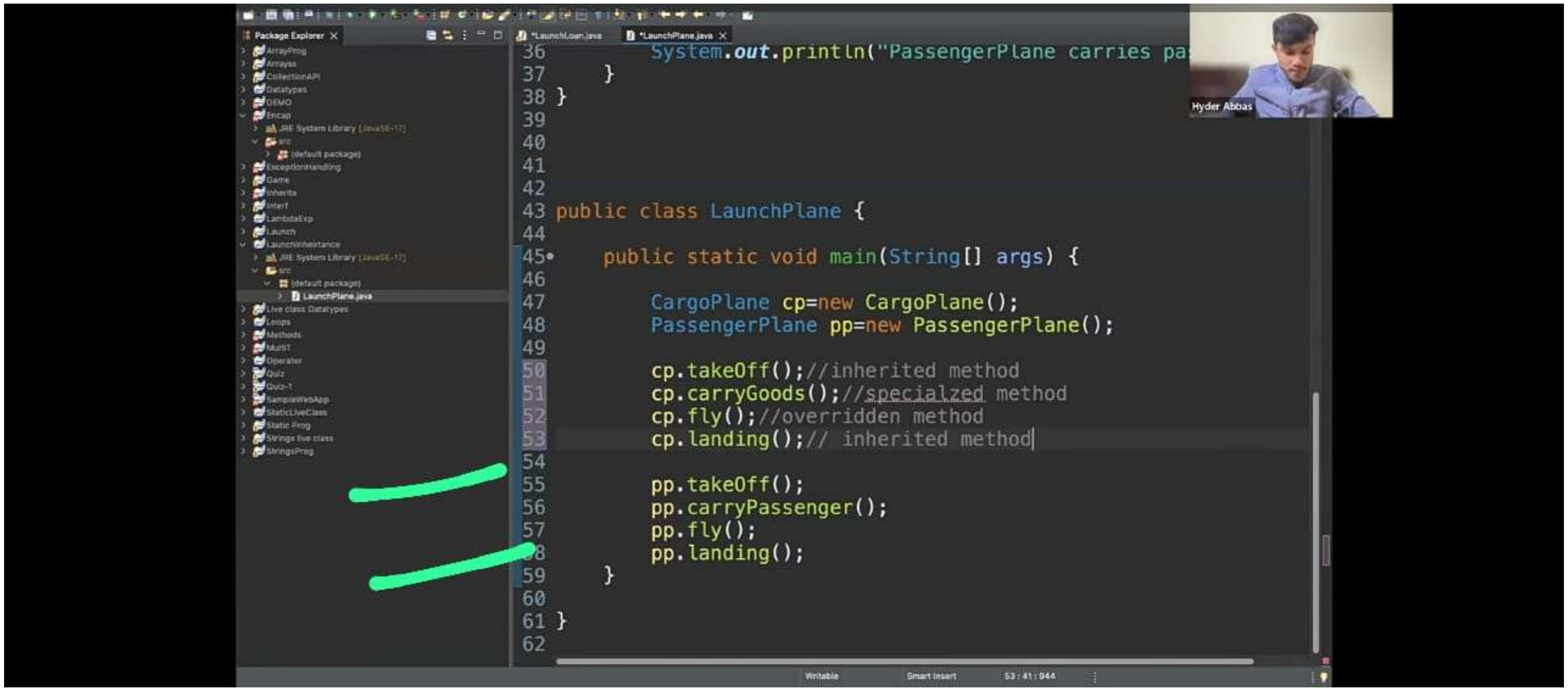


```java
class Plane
{
    public void takeOff()
    {
        System.out.println("Plane is taking off");
    }
    public void fly()
    {
        System.out.println("Plane is flying");
    }
    public void landing()
    {
        System.out.println("Plane is landing");
    }

}
class CargoPlane extends Plane
{
    public void fly()
    {
        System.out.println("CargoPlane flies at lower hie");
    }
}
class PassengerPlane extends Plane
{

}
```

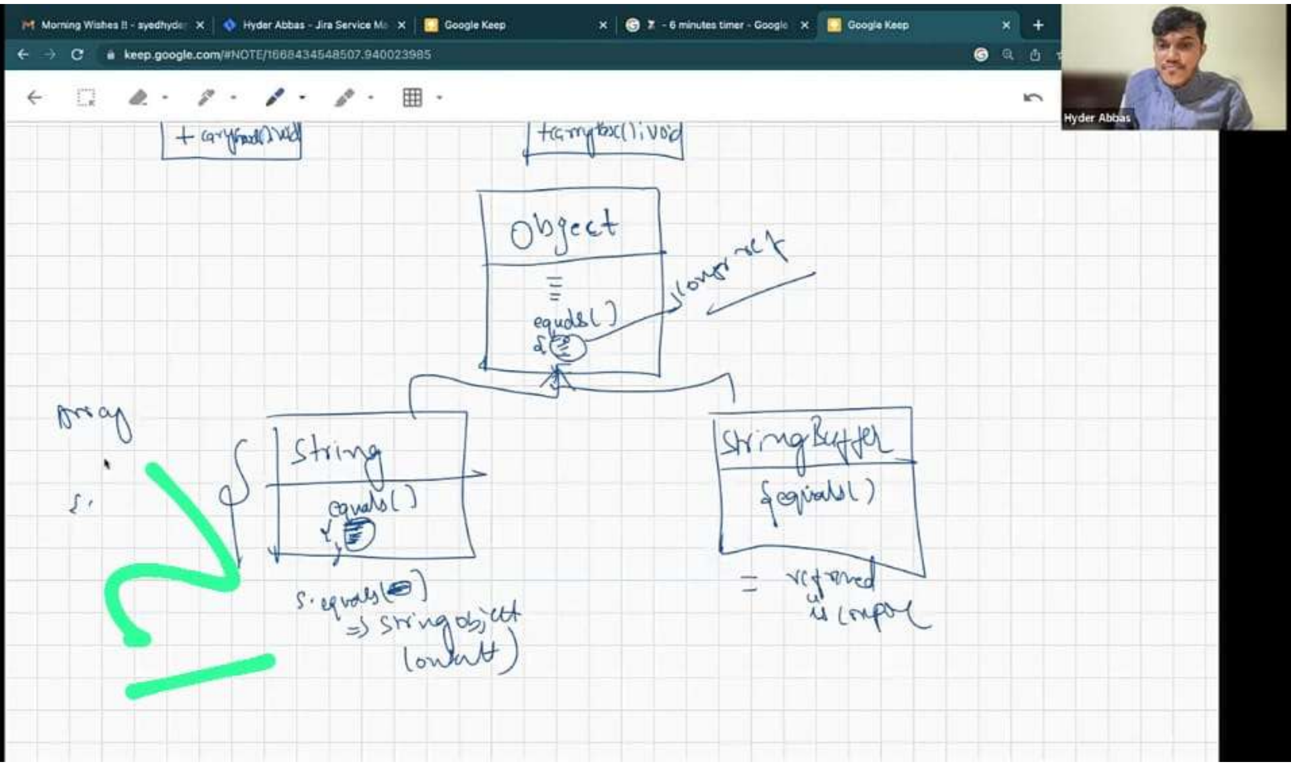
16

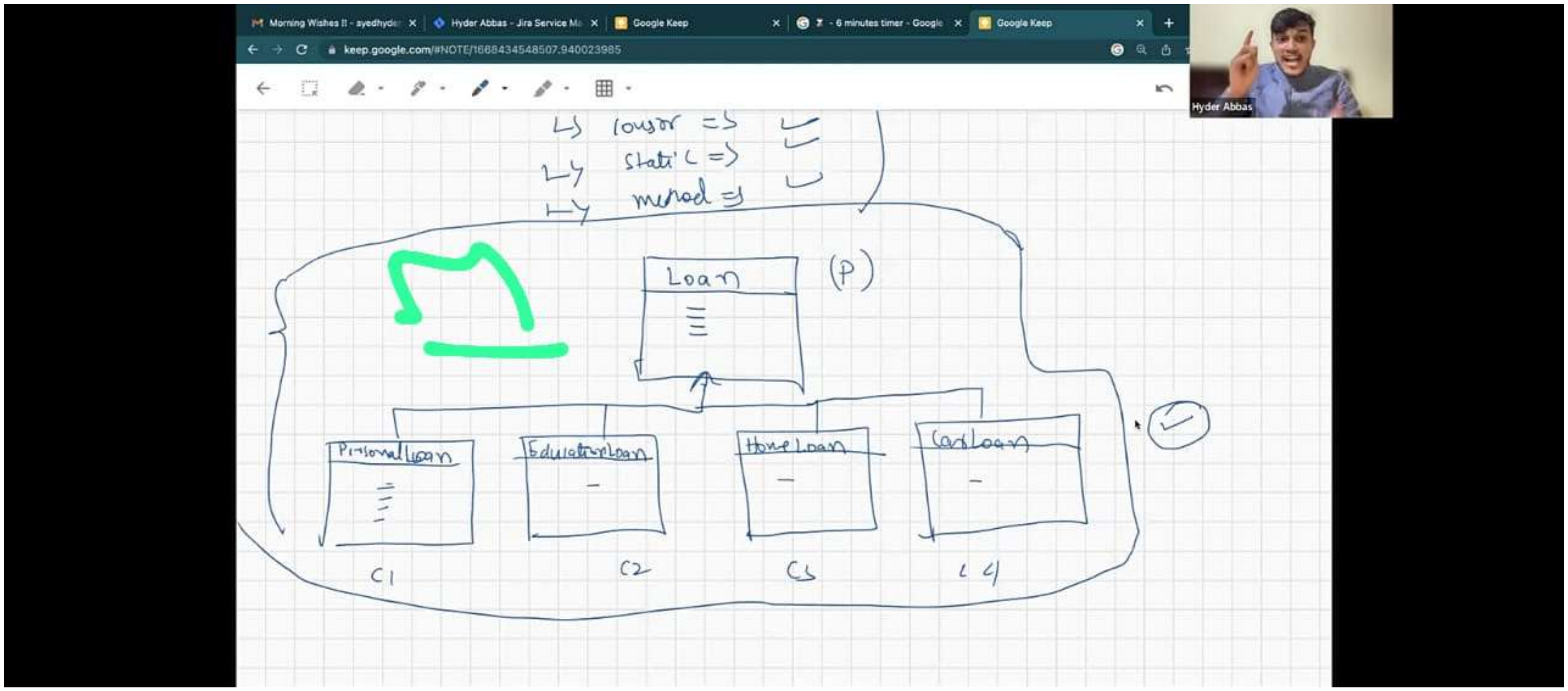


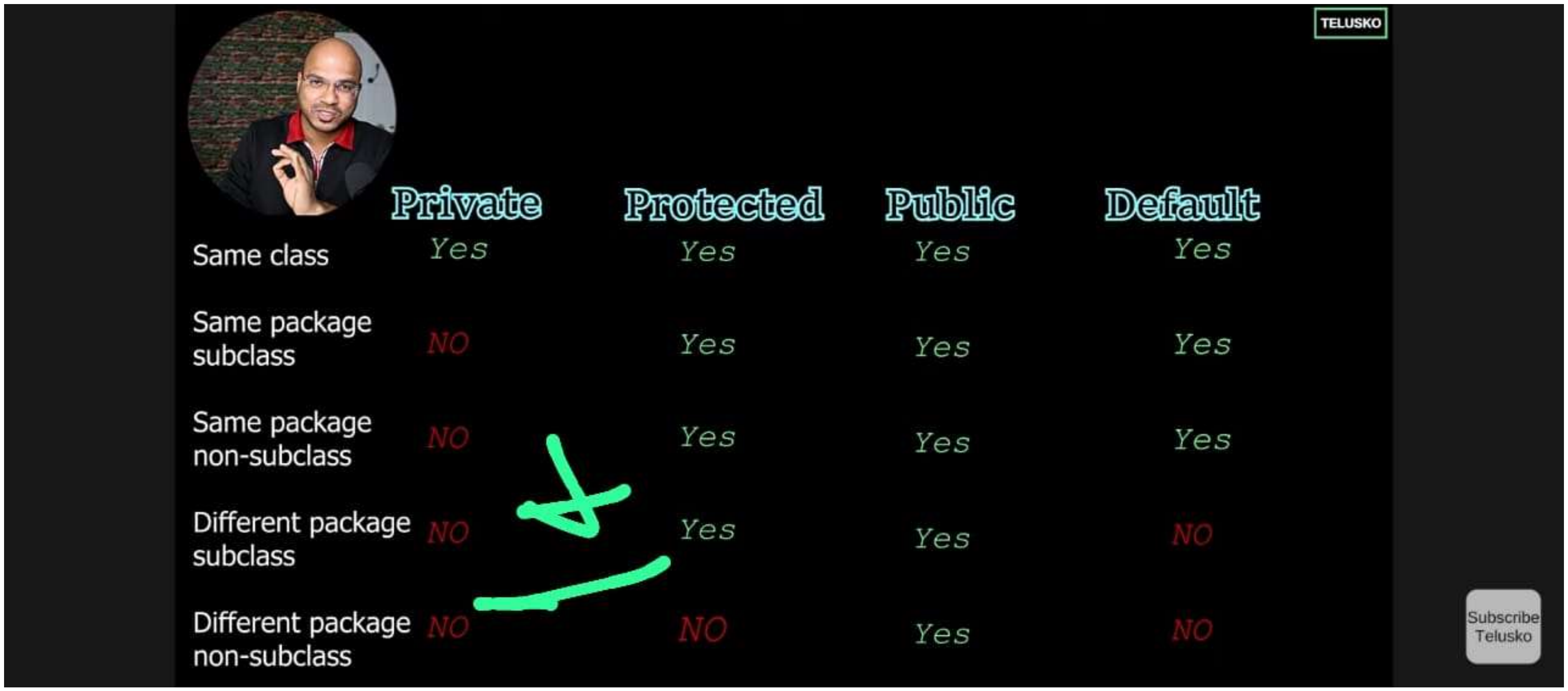
```java
class CargoPlane extends Plane
{
    public void fly()
    {
        System.out.println("CargoPlane flies at lower hieght");
    }
    public void carryGoods()
    {
        System.out.println("cargoPlane carries goods");
    }
}
class PassengerPlane extends Plane
{
    public void fly()
    {
        System.out.println("PassengerPlane flies at medium heigl
    }
    public void carryPassenger()
    {
        System.out.println("PassengerPlane carries passengers")
    }
}
```



```java
        System.out.println("PassengerPlane carries pa
    }
}



public class LaunchPlane {

    public static void main(String[] args) {

        CargoPlane cp=new CargoPlane();
        PassengerPlane pp=new PassengerPlane();

        cp.takeOff();//inherited method
        cp.carryGoods();//specialzed method
        cp.fly();//overridden method
        cp.landing();// inherited method

        pp.takeOff();
        pp.carryPassenger();
        pp.fly();
        pp.landing();
    }

}
```

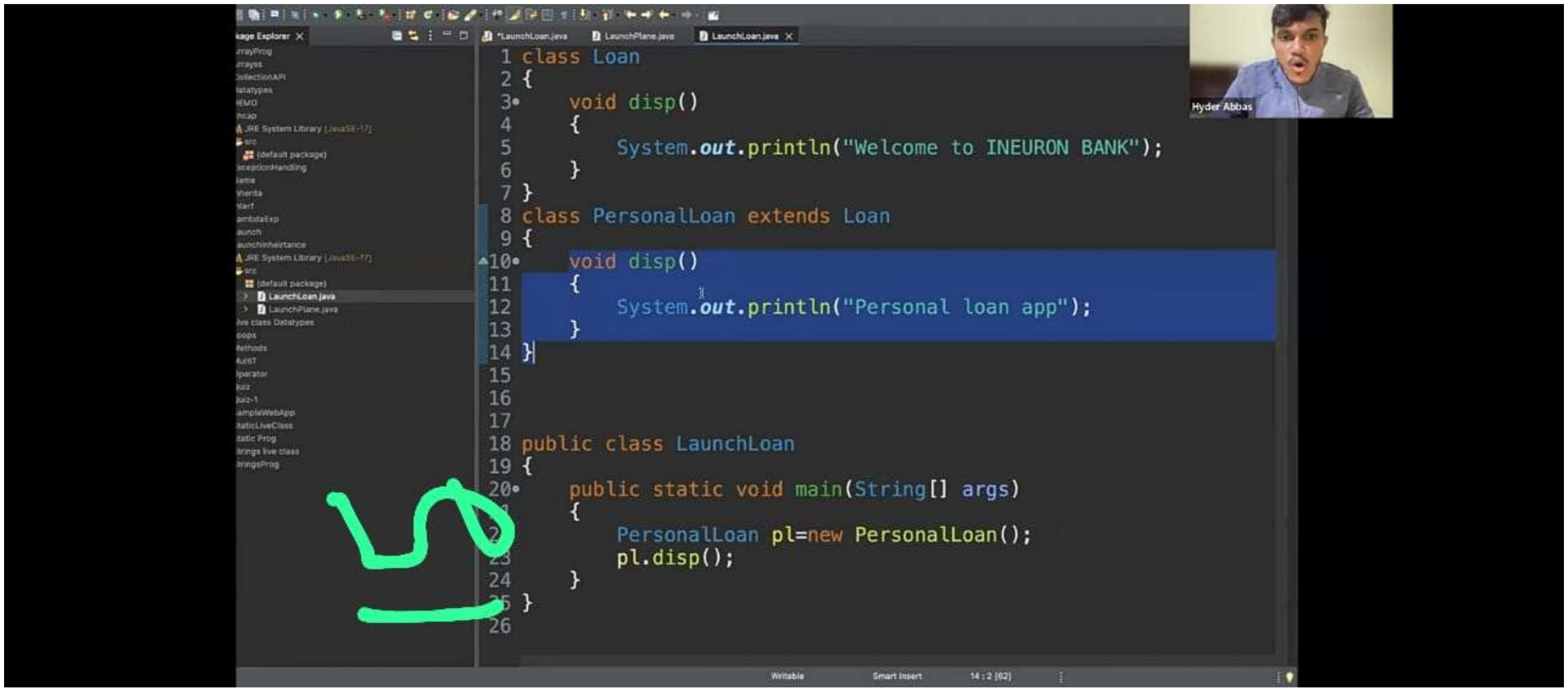

Morning Wishes !! - syedhyder
Hyder Abbas - Jira Service Ma
Google Keep
6 minutes timer - Google
Google Keep
keep.google.com/#NOTE/1668434548507.940023985
Hyder Abbas
Object
equals()
String
equals()
s.equals()
=> String object
StringBuffer
equals()
12


Morning Wishes !! - syedhyder
Hyder Abbas - Jira Service Ma
Google Keep
6 minutes timer - Google
Google Keep
keep.google.com/#NOTE/1668434548507.940023985
Hyder Abbas
lousor =>
static =>
method =>
Loan
(P)
Personal Loan
EducationLoan
Home Loan
CarLoan
C1
C2
C3
C4

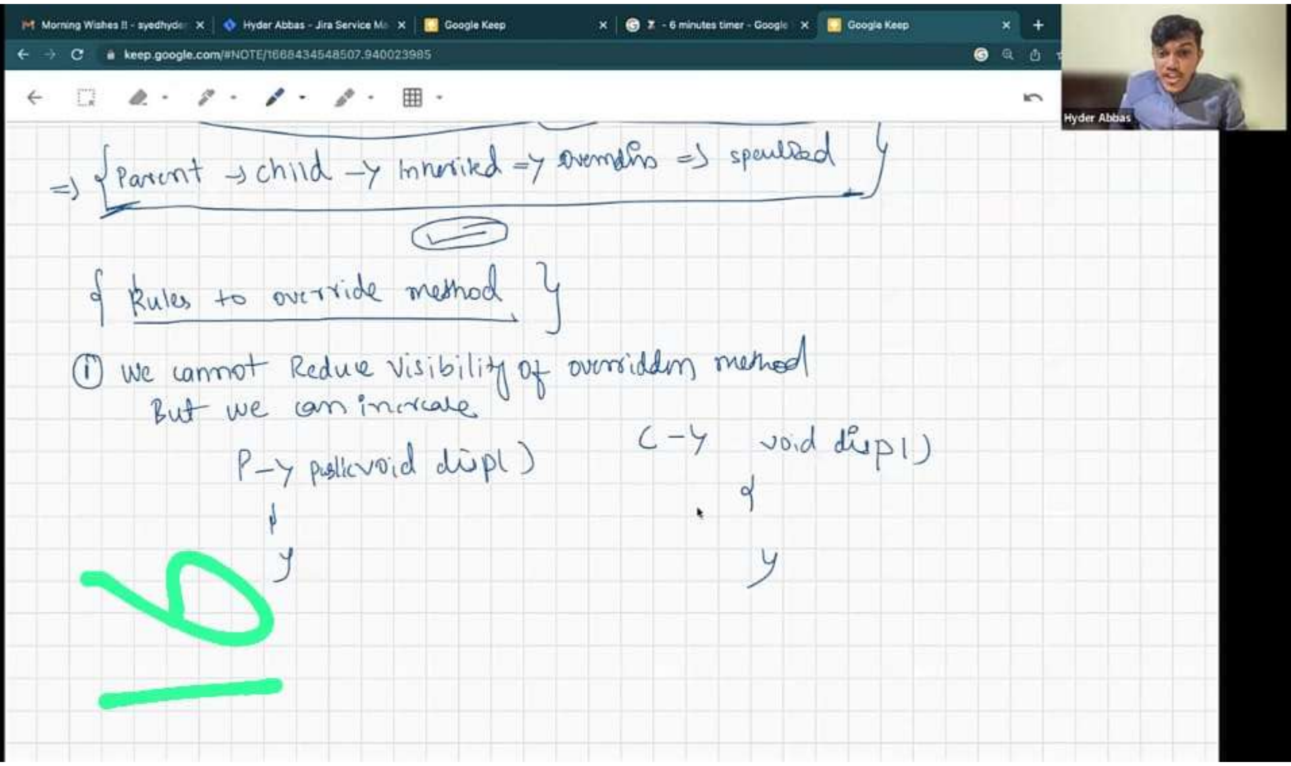


	Private	Protected	Public	Default
Same class	Yes	Yes	Yes	Yes
Same package subclass	NO	Yes	Yes	Yes
Same package non-subclass	NO	Yes	Yes	Yes
Different package subclass	NO	Yes	Yes	NO
Different package non-subclass	NO	NO	Yes	NO

15



```java
class Loan
{
    void disp()
    {
        System.out.println("Welcome to INEURON BANK");
    }
}
class PersonalLoan extends Loan
{
    void disp()
    {
        System.out.println("Personal loan app");
    }
}



public class LaunchLoan
{
    public static void main(String[] args)
    {
        PersonalLoan pl=new PersonalLoan();
        pl.disp();
    }
}
```


=) { Parent → child → Inherited => overriding => specialized }

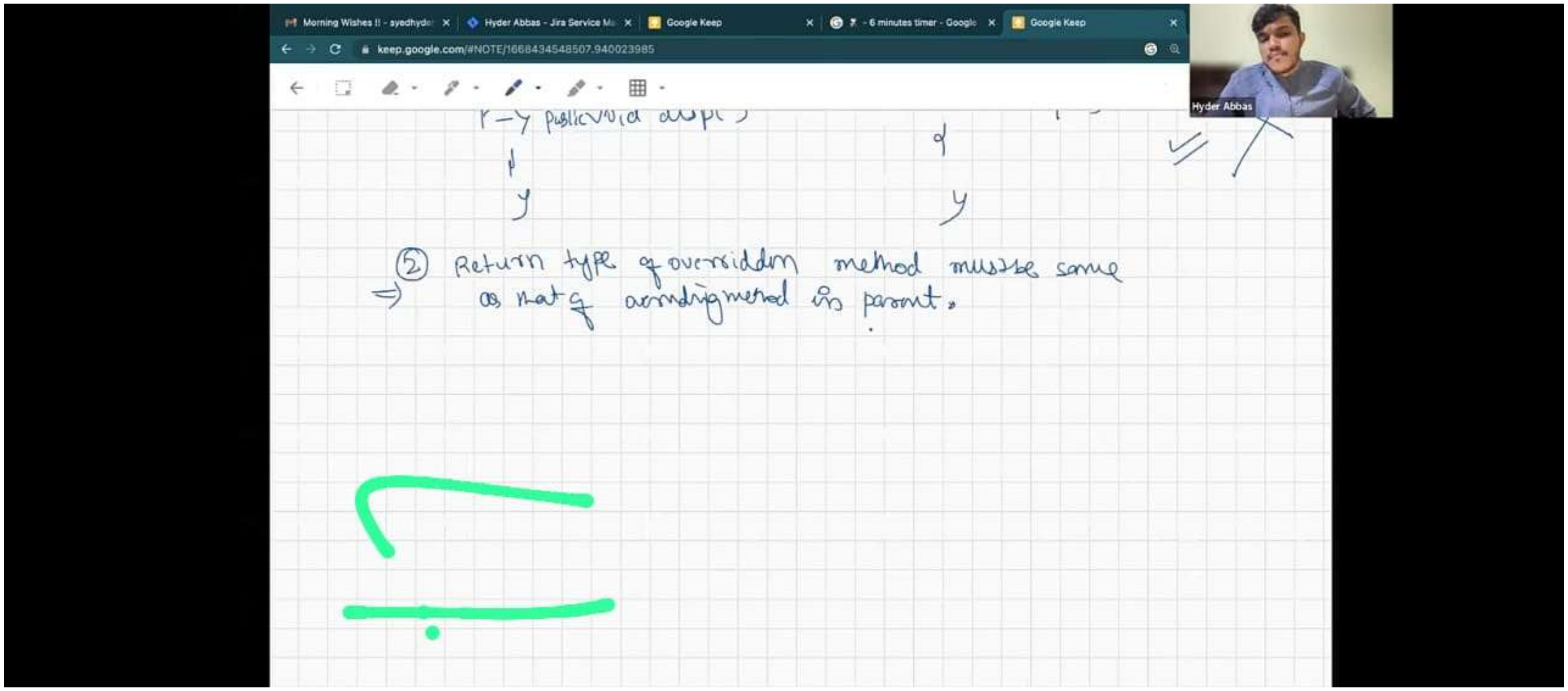
{ Rules to override method }

(i) We cannot Reduce visibility of overridden method
But we can increase

P → public void disp( )
{
}

C → void disp( )
{
}

16



r-> public void disp( )

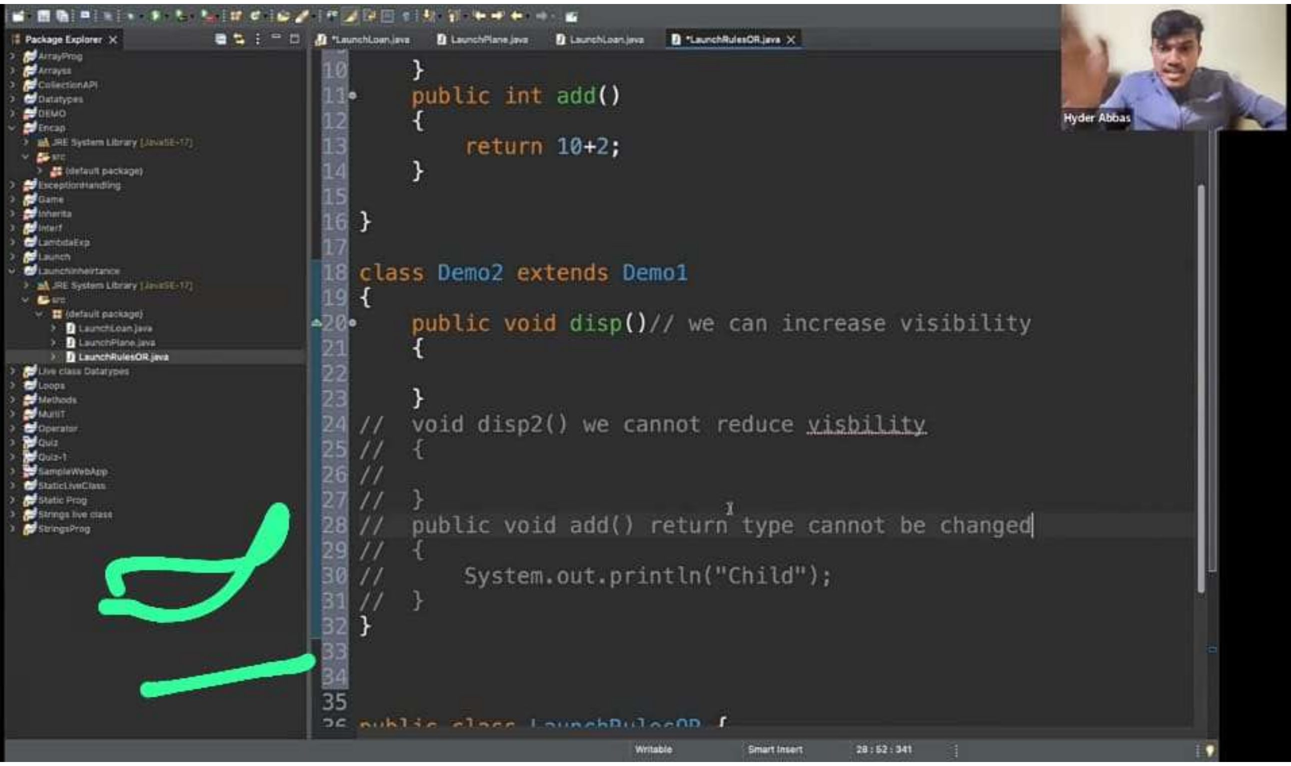
{

}

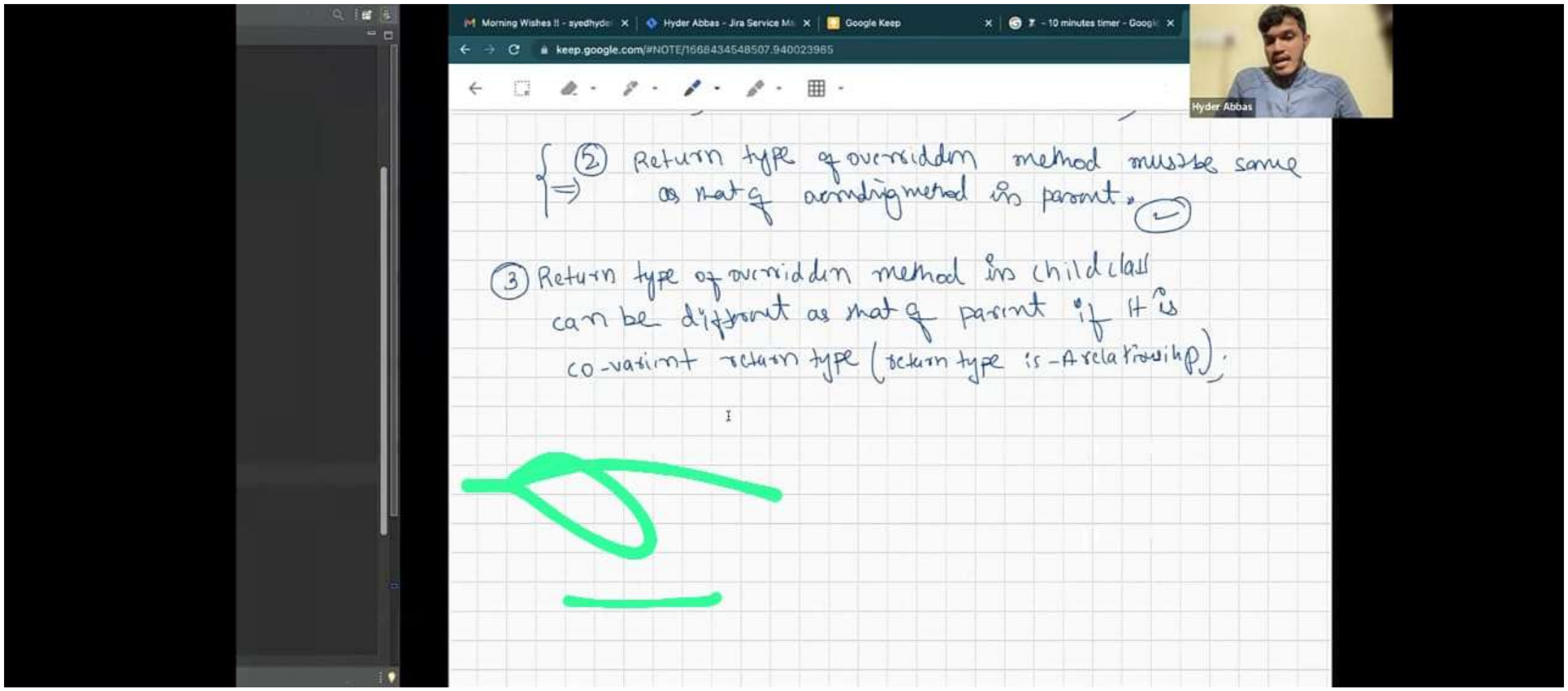
{

}

=> ② Return type of overridden method must be same as that of overriding method in parent.

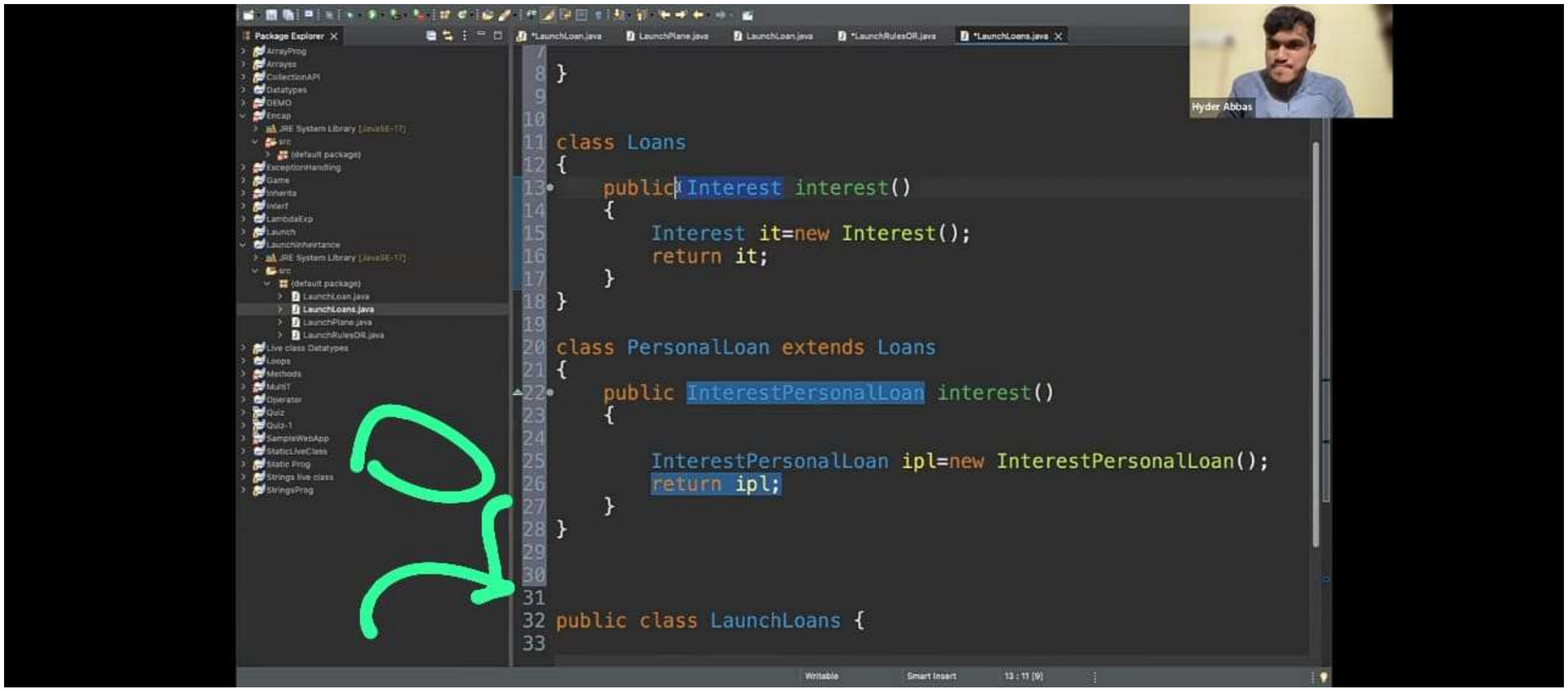


```java
    }
    public int add()
    {
        return 10+2;
    }

}

class Demo2 extends Demo1
{
    public void disp()// we can increase visibility
    {

    }
//  void disp2() we cannot reduce visbility
//  {
//
//  }
//  public void add() return type cannot be changed
//  {
//      System.out.println("Child");
//  }
}


public class LaunchRulesOR {
```



⟹ (2) Return type of overridden method must be same as that of overriding method in parent.

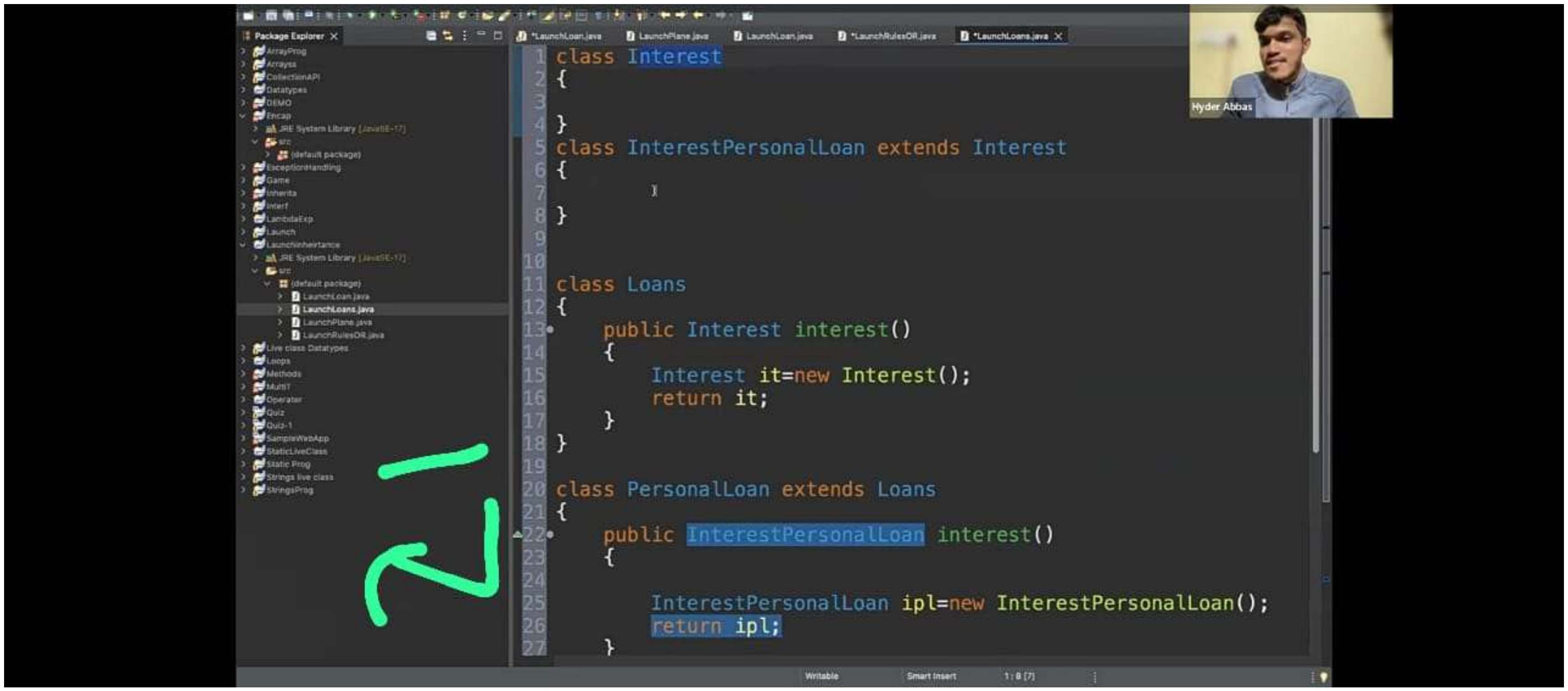
(3) Return type of overridden method in child class can be different as that of parent if it is co-variant return type (return type is-A relationship).



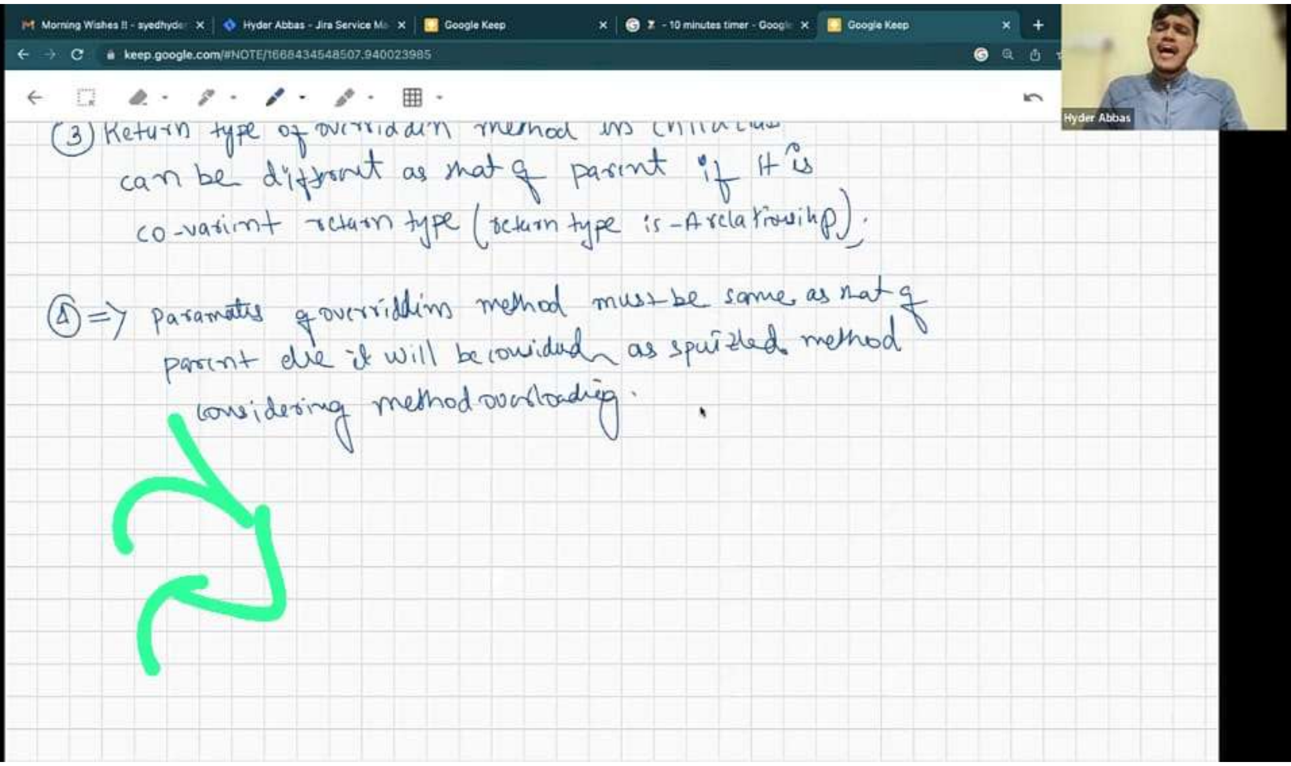
20

```java
}


class Loans
{
    public Interest interest()
    {
        Interest it=new Interest();
        return it;
    }
}

class PersonalLoan extends Loans
{
    public InterestPersonalLoan interest()
    {

        InterestPersonalLoan ipl=new InterestPersonalLoan();
        return ipl;
    }
}



public class LaunchLoans {

```

21

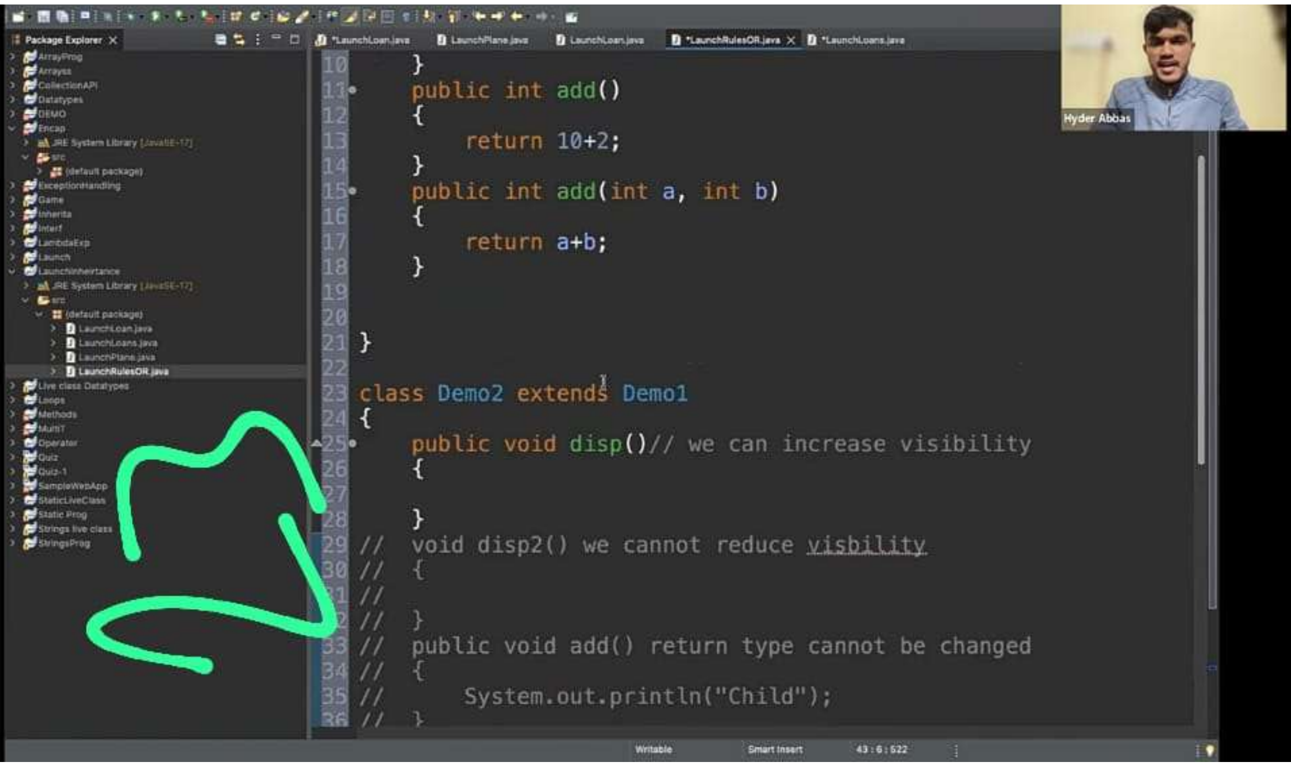


```java
class Interest
{

}
class InterestPersonalLoan extends Interest
{

}


class Loans
{
    public Interest interest()
    {
        Interest it=new Interest();
        return it;
    }
}

class PersonalLoan extends Loans
{
    public InterestPersonalLoan interest()
    {

        InterestPersonalLoan ipl=new InterestPersonalLoan();
        return ipl;
    }
```


Morning Wishes !! - syedhyder
Hyder Abbas - Jira Service Ma
Google Keep
10 minutes timer - Google
Google Keep
keep.google.com/#NOTE/1668434548507.940023985
Hyder Abbas

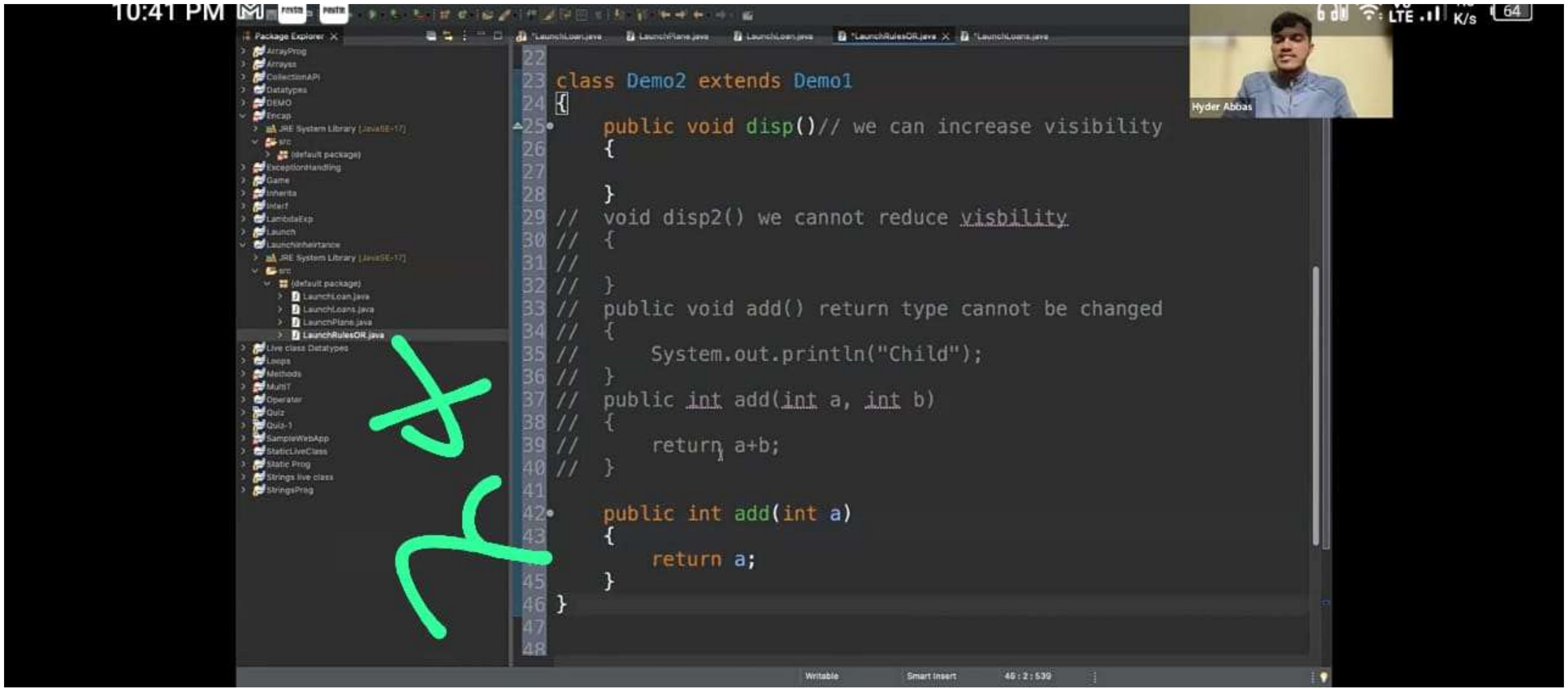
(3) Return type of overridden method in child class can be different as that of parent if it is co-variant return type (return type is-A relationship).

(4) => Parameters of overridding method must be same as that of parent else it will be considered as specialized method considering method overloading.



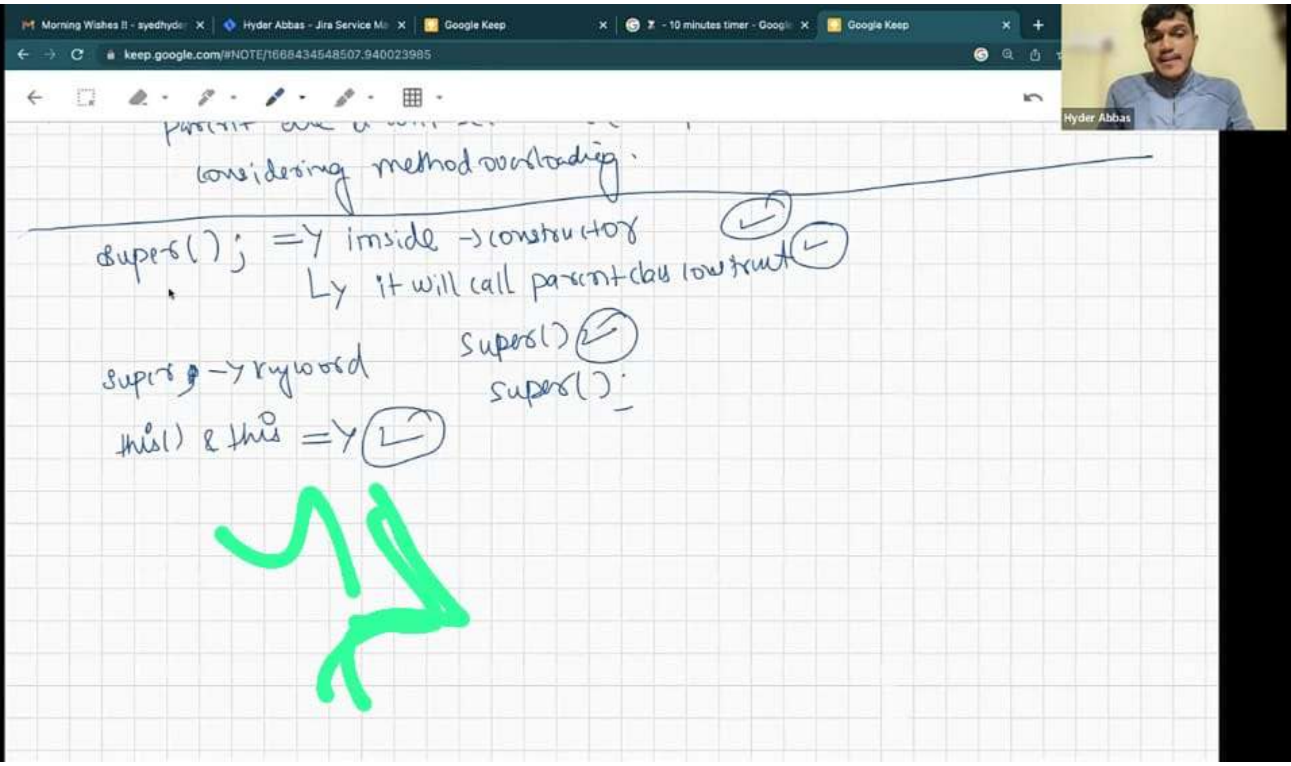
23

```
    }
    public int add()
    {
        return 10+2;
    }
    public int add(int a, int b)
    {
        return a+b;
    }


}

class Demo2 extends Demo1
{
    public void disp()// we can increase visibility
    {

    }
//  void disp2() we cannot reduce visbility
//  {
//
//  }
//  public void add() return type cannot be changed
//  {
//      System.out.println("Child");
//  }
```



24

```
class Demo2 extends Demo1
{
    public void disp()// we can increase visibility
    {

    }
//  void disp2() we cannot reduce visbility
//  {
//
//  }
//  public void add() return type cannot be changed
//  {
//      System.out.println("Child");
//  }
//  public int add(int a, int b)
//  {
//      return a+b;
//  }

    public int add(int a)
    {
        return a;
    }
}
```

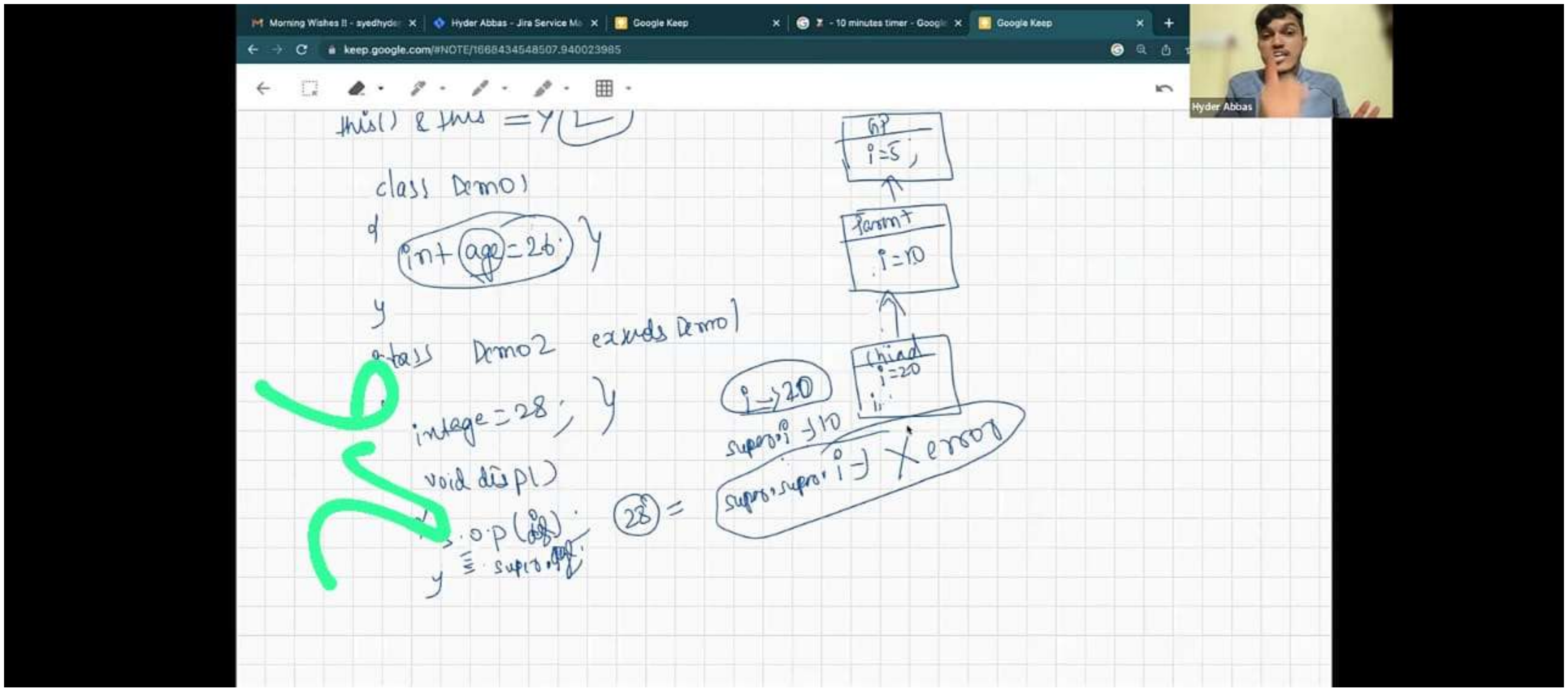


considering method overloading.

super(); =⟩ inside → constructor ✓
  ↳ it will call parent class constructor ✓

super → keyword   super() ✓
                     super();

this() & this =⟩ ✓

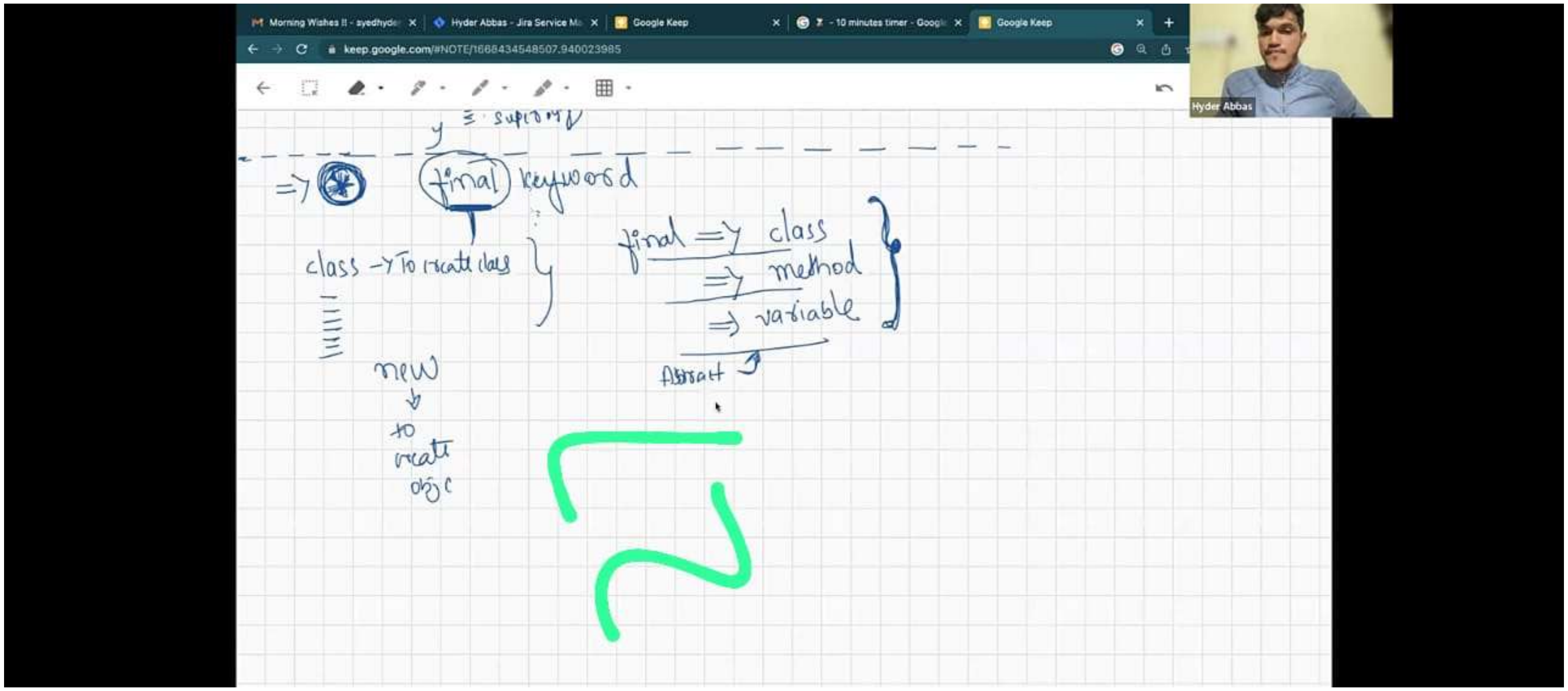


this() & this = ()

class Demo1
{
int age = 26;
}

class Demo2 extends Demo1
{
int age = 28;
void disp()
{
S.o.p(age);
S.o.p(super.age);
}
}

28 =


Morning Wishes !! - syedhyder | Hyder Abbas - Jira Service Ma | Google Keep | 10 minutes timer - Google | Google Keep

keep.google.com/#NOTE/1668434548507.940023985

Hyder Abbas

≡ · suprony

⇒ final keyword

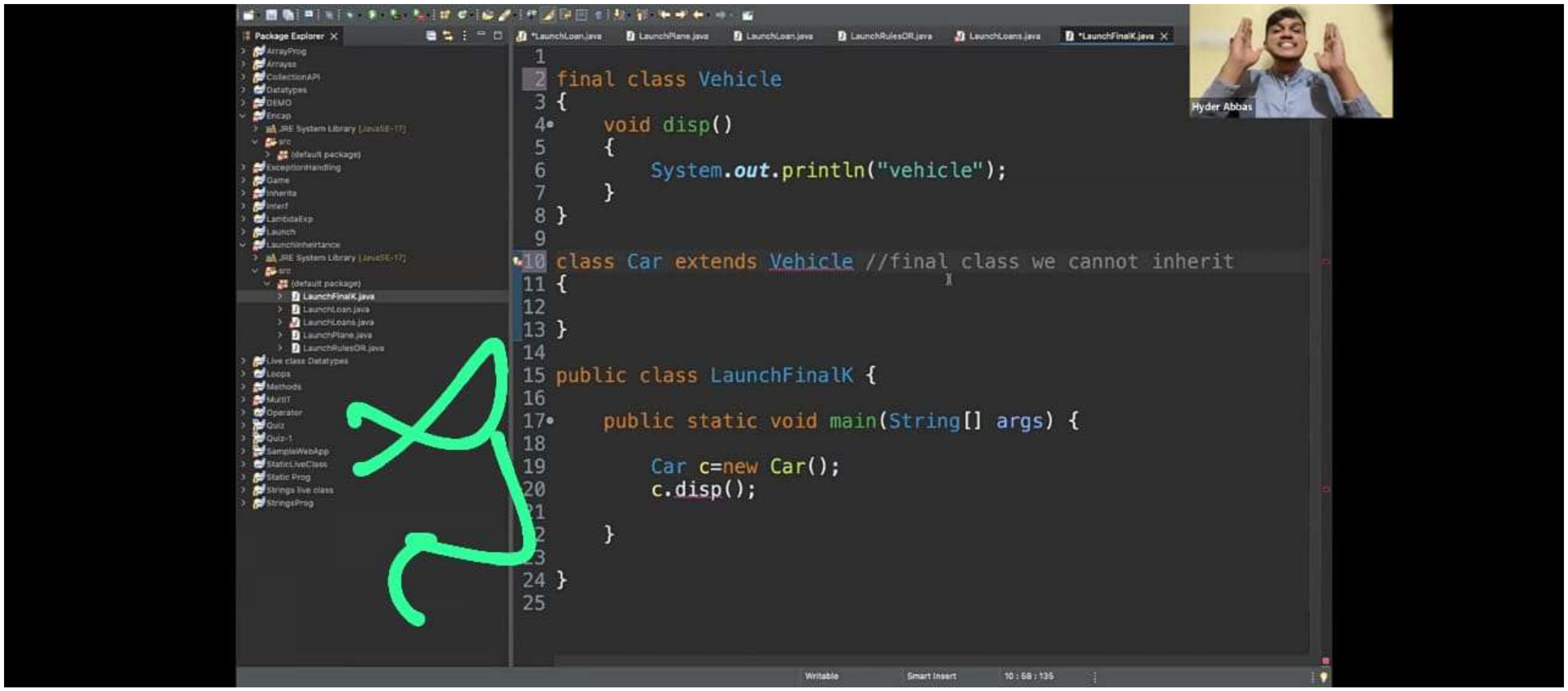
class → To create class

new → to create objc

final ⇒ class
⇒ method
⇒ variable

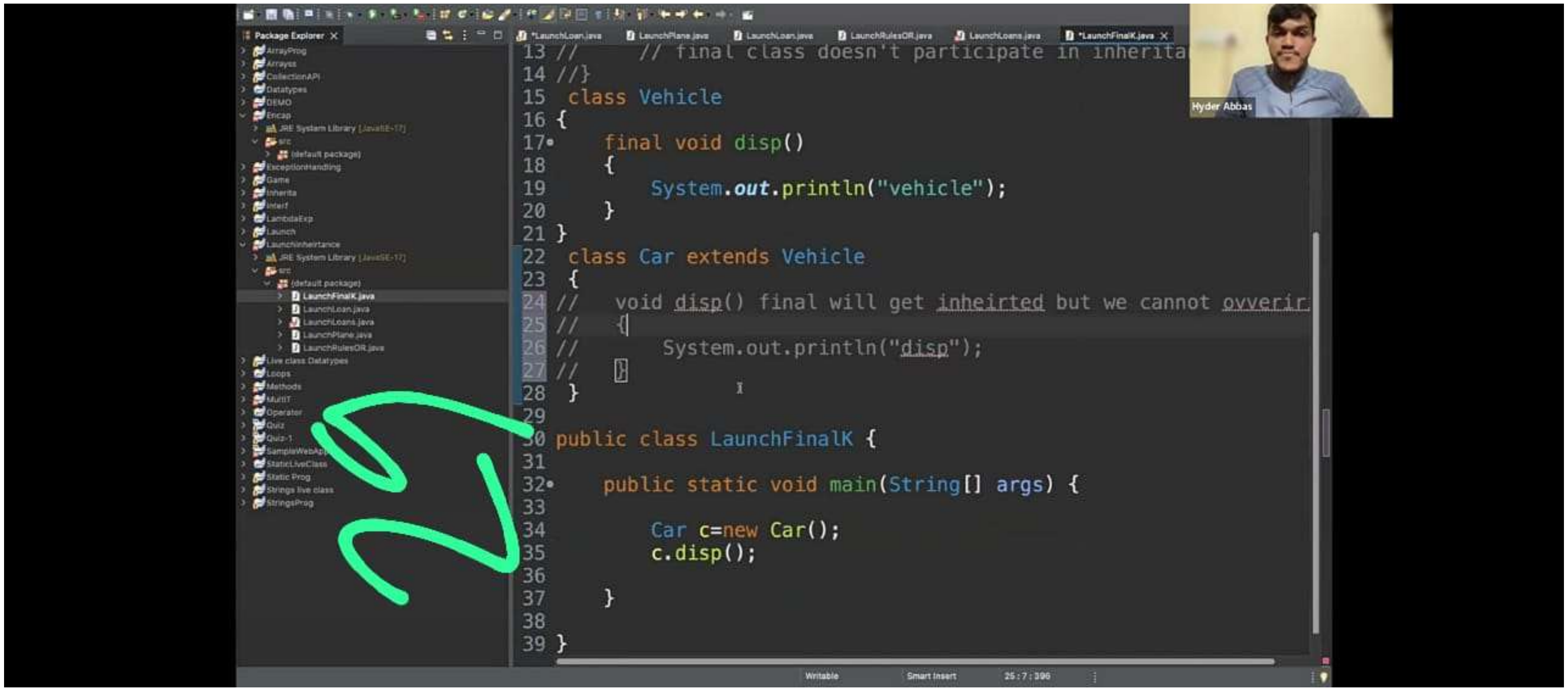
Abstract

27



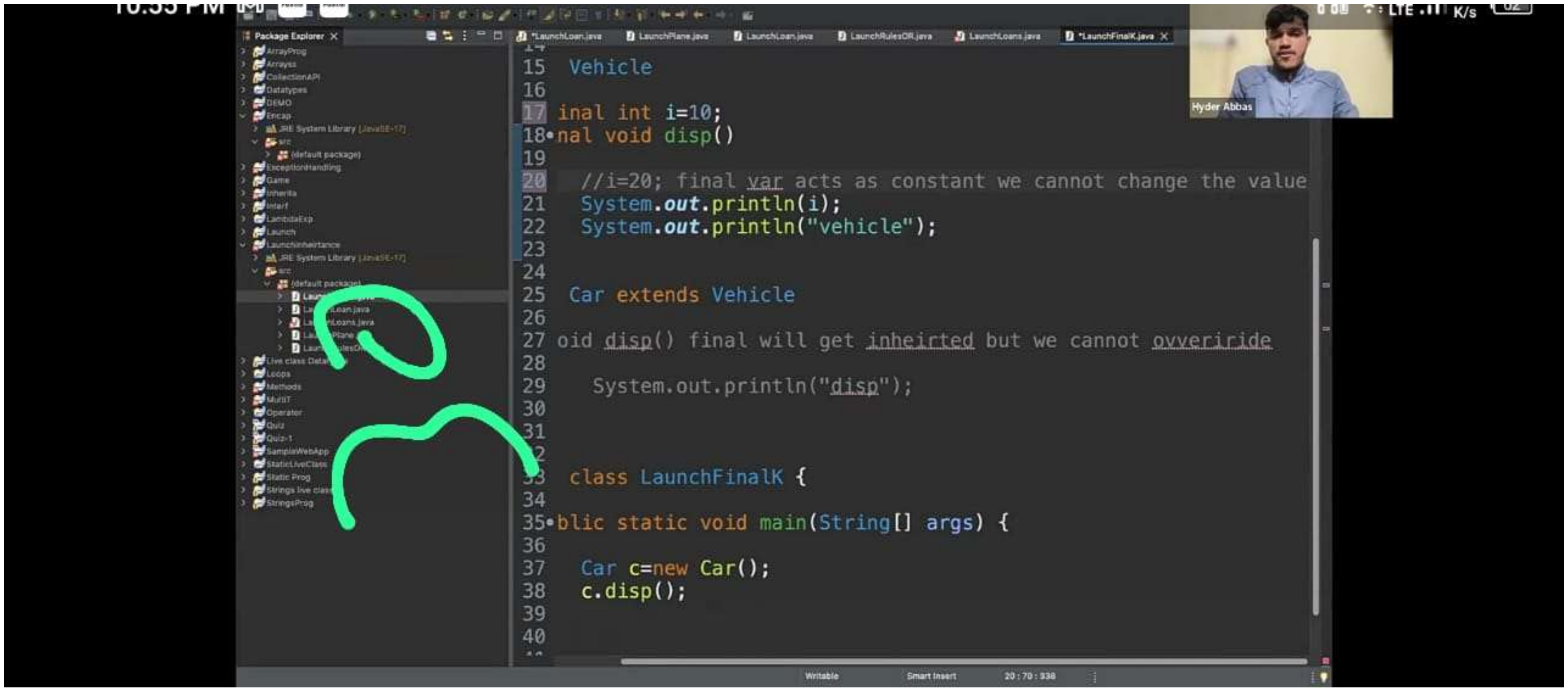
```java
final class Vehicle
{
    void disp()
    {
        System.out.println("vehicle");
    }
}

class Car extends Vehicle //final class we cannot inherit
{

}

public class LaunchFinalK {

    public static void main(String[] args) {

        Car c=new Car();
        c.disp();

    }

}
```

28

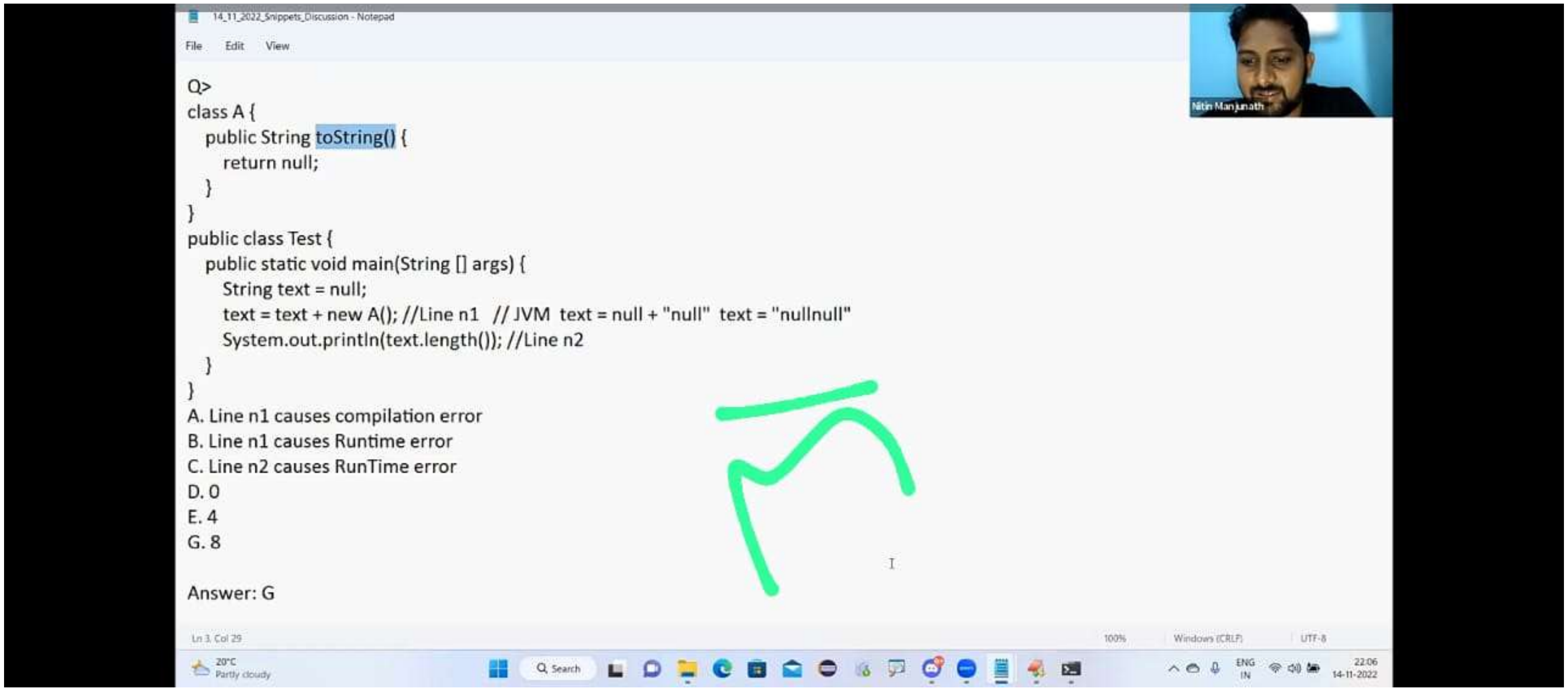


29

```java
//      // final class doesn't participate in inherita
//}
class Vehicle
{
    final void disp()
    {
        System.out.println("vehicle");
    }
}
class Car extends Vehicle
{
//    void disp() final will get inheirted but we cannot ovverir
//    {
//        System.out.println("disp");
//    }
}

public class LaunchFinalK {

    public static void main(String[] args) {

        Car c=new Car();
        c.disp();

    }

}
```

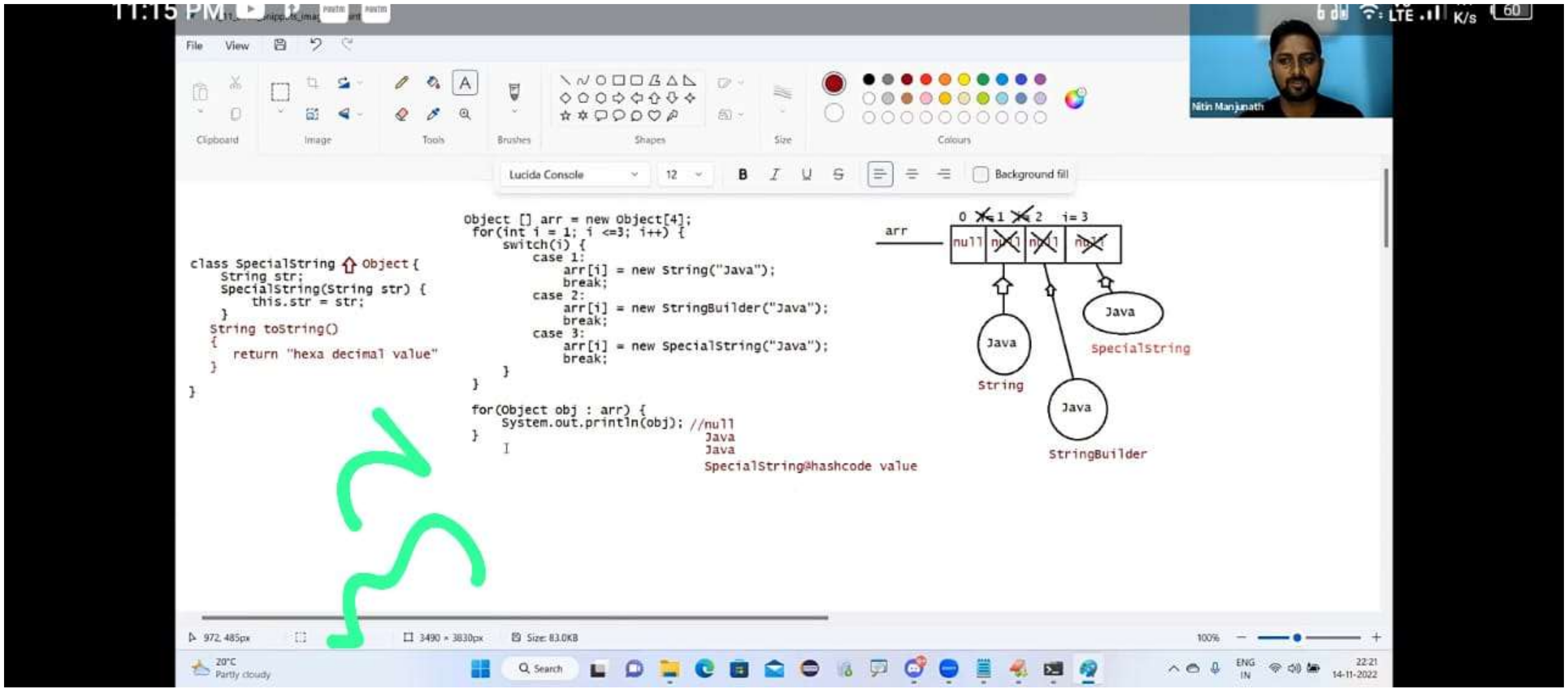


```
 Vehicle

inal int i=10;
nal void disp()

  //i=20; final var acts as constant we cannot change the value
  System.out.println(i);
  System.out.println("vehicle");


 Car extends Vehicle

oid disp() final will get inheirted but we cannot ovveriride

   System.out.println("disp");



 class LaunchFinalK {

blic static void main(String[] args) {

  Car c=new Car();
  c.disp();
```



Q>
```
class A {
  public String toString() {
    return null;
  }
}
public class Test {
  public static void main(String [] args) {
    String text = null;
    text = text + new A(); //Line n1   // JVM  text = null + "null"  text = "nullnull"
    System.out.println(text.length()); //Line n2
  }
}
```
A. Line n1 causes compilation error
B. Line n1 causes Runtime error
C. Line n2 causes RunTime error
D. 0
E. 4
G. 8

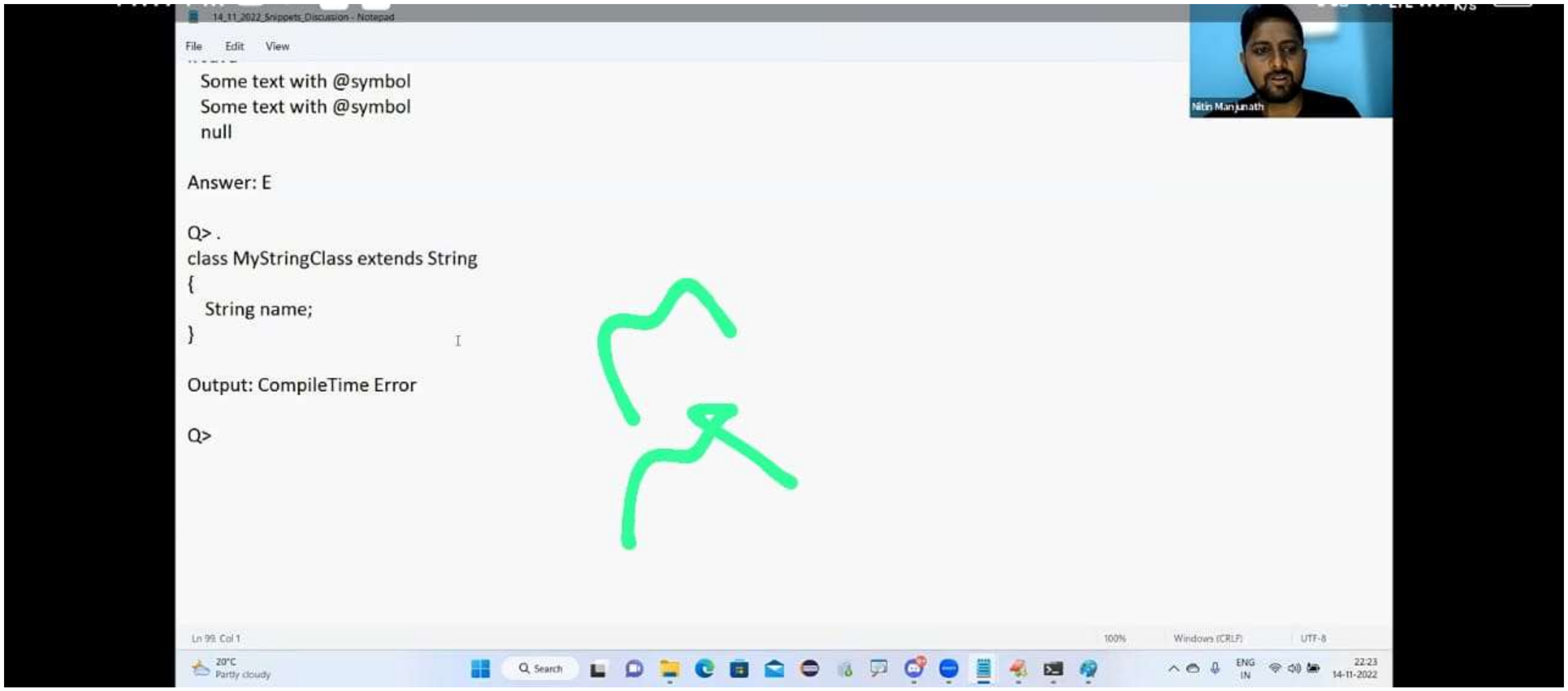
Answer: G



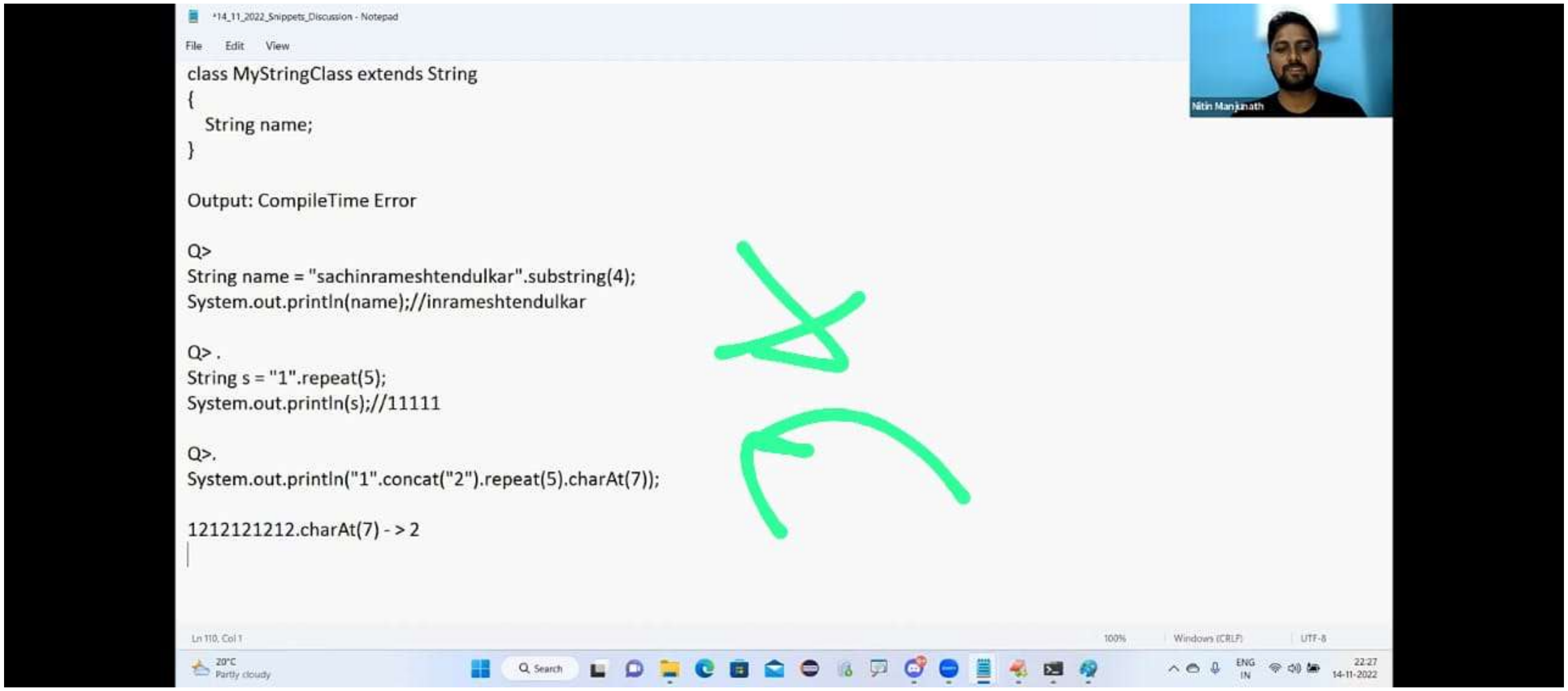
```
class SpecialString ⇧ Object {
    String str;
    SpecialString(String str) {
        this.str = str;
    }
  String toString()
  {
     return "hexa decimal value"
  }
}
```

```
Object [] arr = new Object[4];
 for(int i = 1; i <=3; i++) {
     switch(i) {
         case 1:
             arr[i] = new String("Java");
             break;
         case 2:
             arr[i] = new StringBuilder("Java");
             break;
         case 3:
             arr[i] = new SpecialString("Java");
             break;
     }
 }

 for(Object obj : arr) {
     System.out.println(obj); //null
 }                              Java
                                Java
                                SpecialString@hashcode value
```

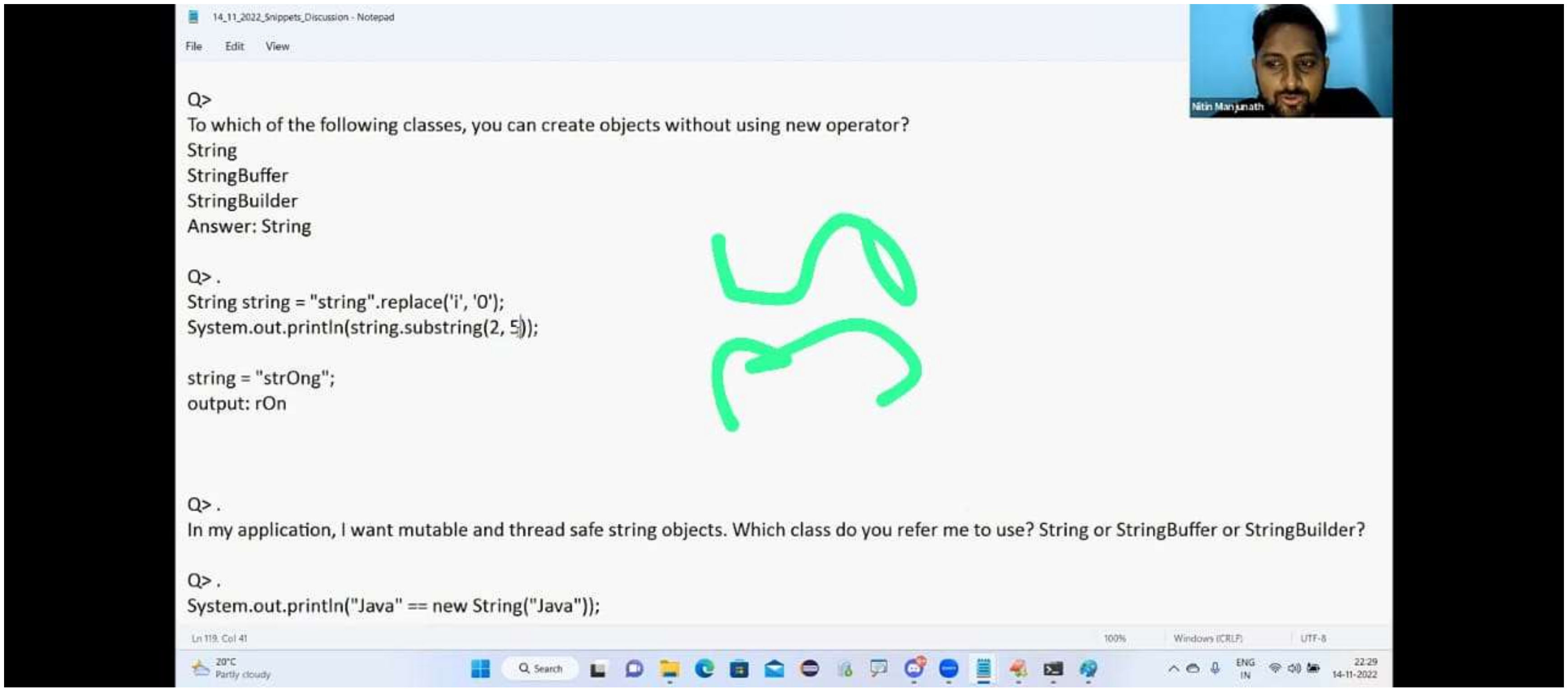
32



```
  Some text with @symbol
  Some text with @symbol
  null

Answer: E

Q> .
class MyStringClass extends String
{
   String name;
}

Output: CompileTime Error

Q>
```



```
class MyStringClass extends String
{
  String name;
}

Output: CompileTime Error

Q>
String name = "sachinrameshtendulkar".substring(4);
System.out.println(name);//inrameshtendulkar

Q> .
String s = "1".repeat(5);
System.out.println(s);//11111

Q>.
System.out.println("1".concat("2").repeat(5).charAt(7));

1212121212.charAt(7) - > 2
```



14_11_2022_Snippets_Discussion - Notepad

File Edit View

Q>
To which of the following classes, you can create objects without using new operator?
String
StringBuffer
StringBuilder
Answer: String

Q> .

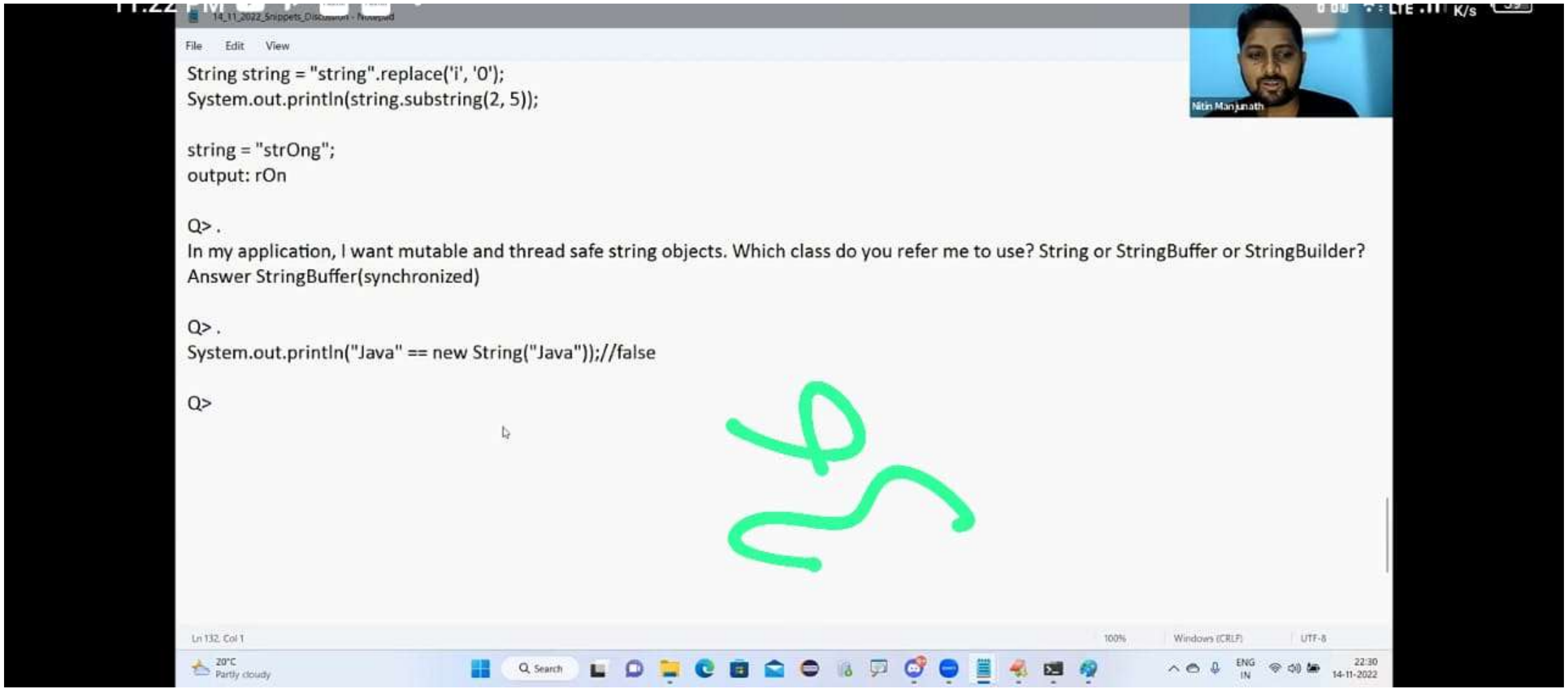
```
String string = "string".replace('i', '0');
System.out.println(string.substring(2, 5));

string = "strOng";
output: rOn
```

35

Q> .
In my application, I want mutable and thread safe string objects. Which class do you refer me to use? String or StringBuffer or StringBuilder?

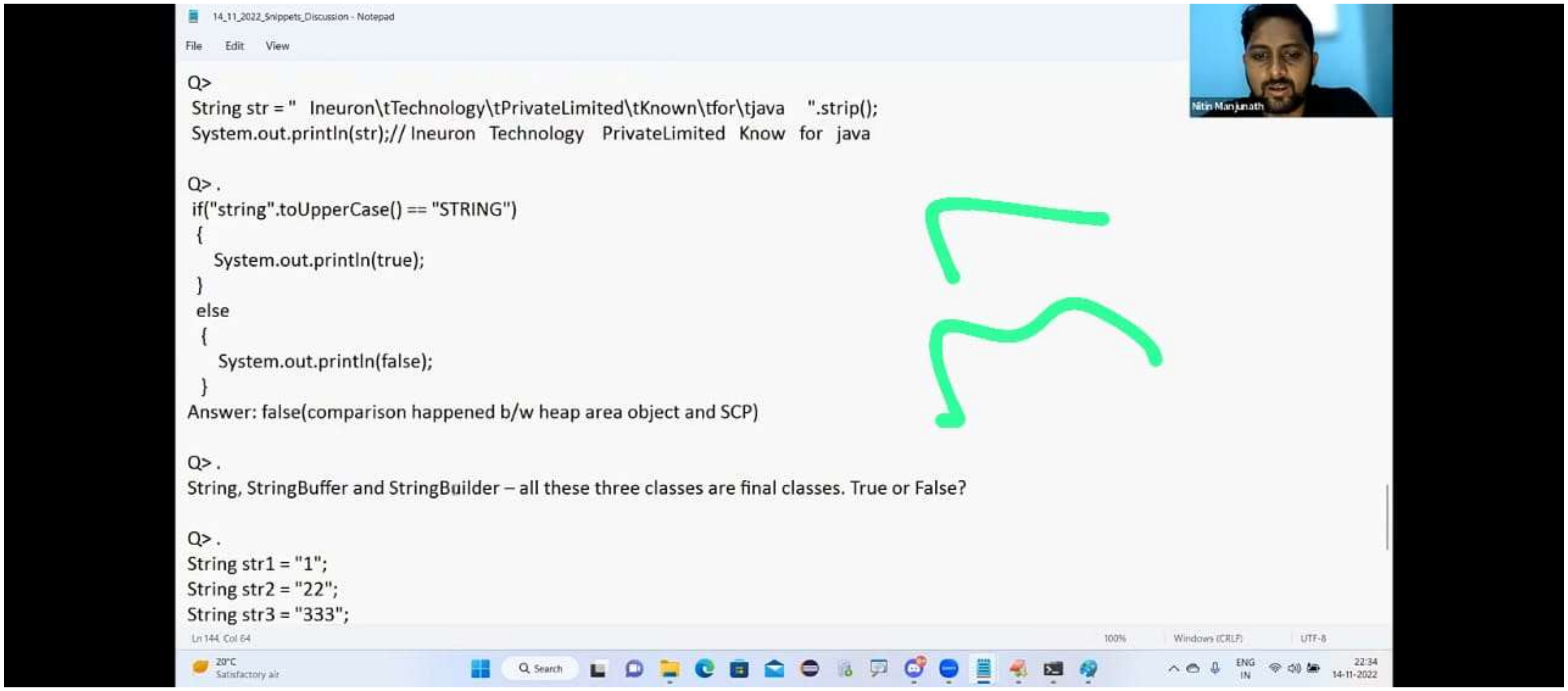
Q> .

```
System.out.println("Java" == new String("Java"));
```



```
String string = "string".replace('i', 'O');
System.out.println(string.substring(2, 5));

string = "strOng";
output: rOn

Q> .
In my application, I want mutable and thread safe string objects. Which class do you refer me to use? String or StringBuffer or StringBuilder?
Answer StringBuffer(synchronized)

Q> .
System.out.println("Java" == new String("Java"));//false

Q>
```

36



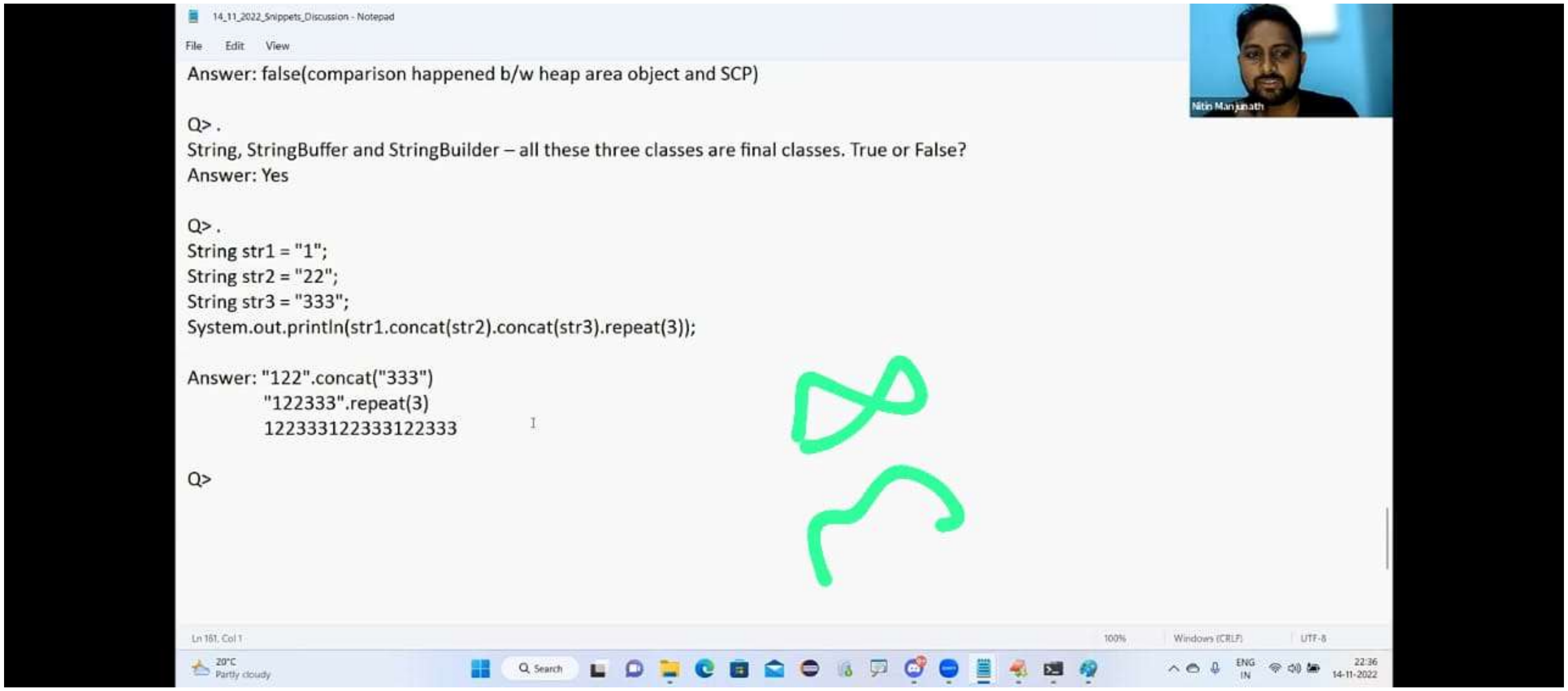
Q>
```
String str = "  Ineuron\tTechnology\tPrivateLimited\tKnown\tfor\tjava   ".strip();
System.out.println(str);// Ineuron  Technology  PrivateLimited  Know  for  java
```

Q> .
```
if("string".toUpperCase() == "STRING")
{
   System.out.println(true);
}
else
{
   System.out.println(false);
}
```
Answer: false(comparison happened b/w heap area object and SCP)

Q> .
String, StringBuffer and StringBuilder – all these three classes are final classes. True or False?

Q> .
```
String str1 = "1";
String str2 = "22";
String str3 = "333";
```

37



Answer: false(comparison happened b/w heap area object and SCP)

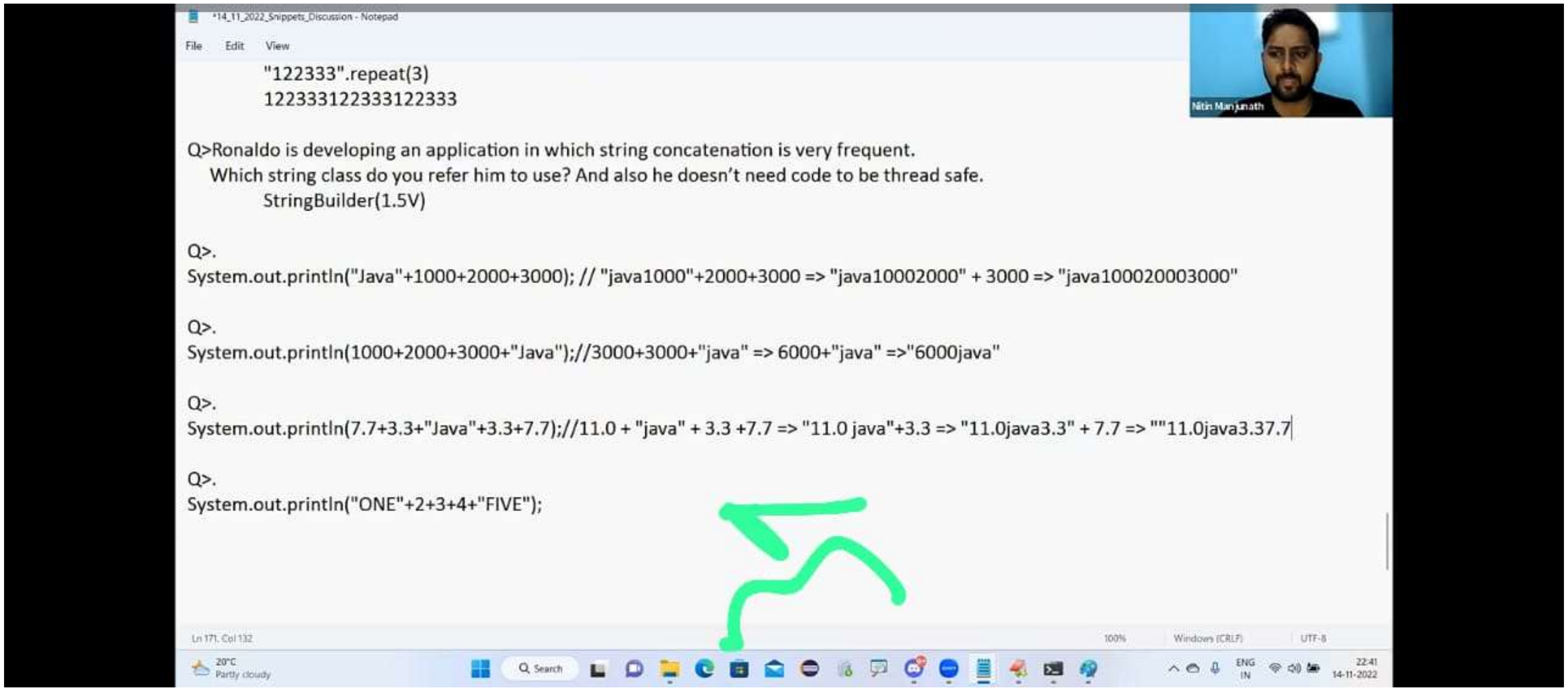
Q> .
String, StringBuffer and StringBuilder – all these three classes are final classes. True or False?
Answer: Yes

Q> .
```
String str1 = "1";
String str2 = "22";
String str3 = "333";
System.out.println(str1.concat(str2).concat(str3).repeat(3));
```

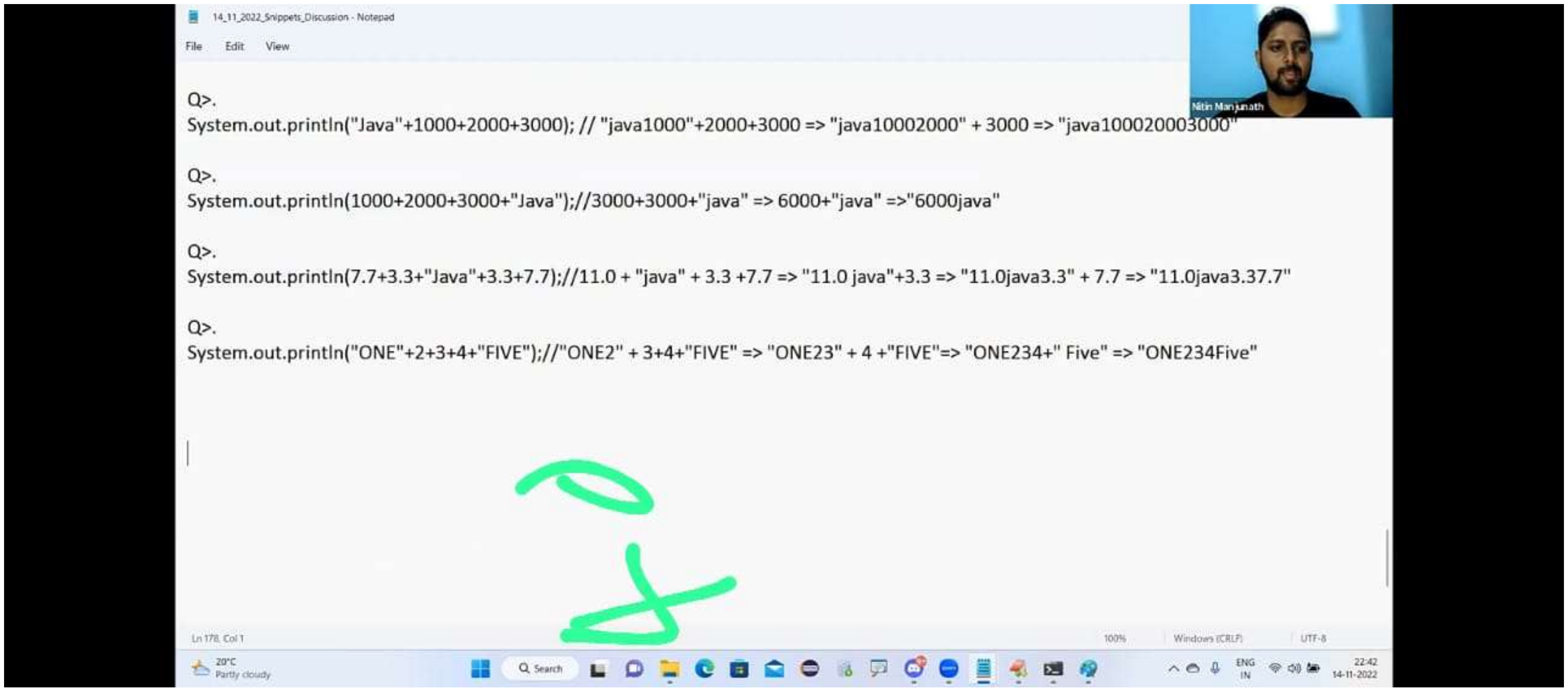
```
Answer: "122".concat("333")
          "122333".repeat(3)
          122333122333122333
```

Q>

38

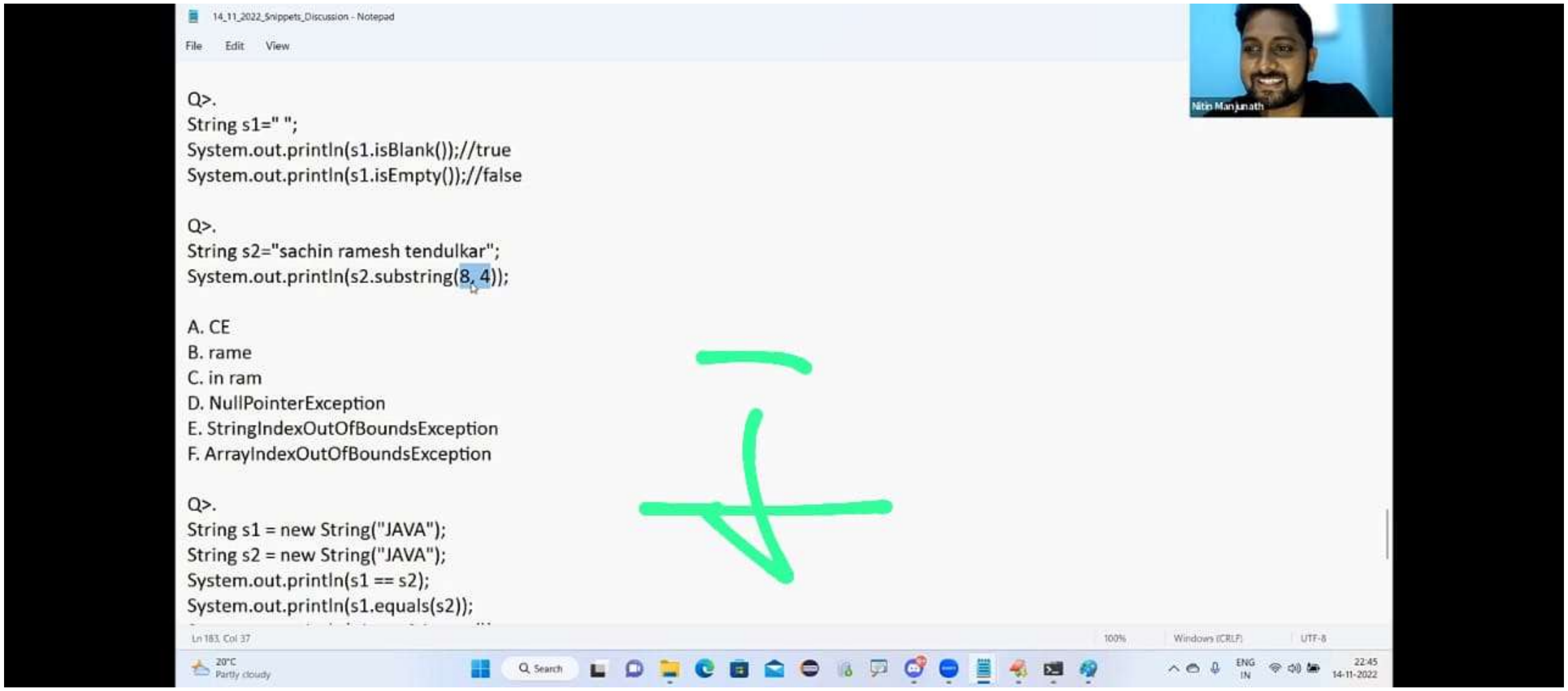


```
*14_11_2022_Snippets_Discussion - Notepad
File   Edit   View
        "122333".repeat(3)
        122333122333122333

Q>Ronaldo is developing an application in which string concatenation is very frequent.
  Which string class do you refer him to use? And also he doesn't need code to be thread safe.
        StringBuilder(1.5V)

Q>.
System.out.println("Java"+1000+2000+3000); // "java1000"+2000+3000 => "java10002000" + 3000 => "java100020003000"

Q>.
System.out.println(1000+2000+3000+"Java");//3000+3000+"java" => 6000+"java" =>"6000java"

Q>.
System.out.println(7.7+3.3+"Java"+3.3+7.7);//11.0 + "java" + 3.3 +7.7 => "11.0 java"+3.3 => "11.0java3.3" + 7.7 => ""11.0java3.37.7

Q>.
System.out.println("ONE"+2+3+4+"FIVE");

Ln 171, Col 132                                   100%    Windows (CRLF)    UTF-8
```



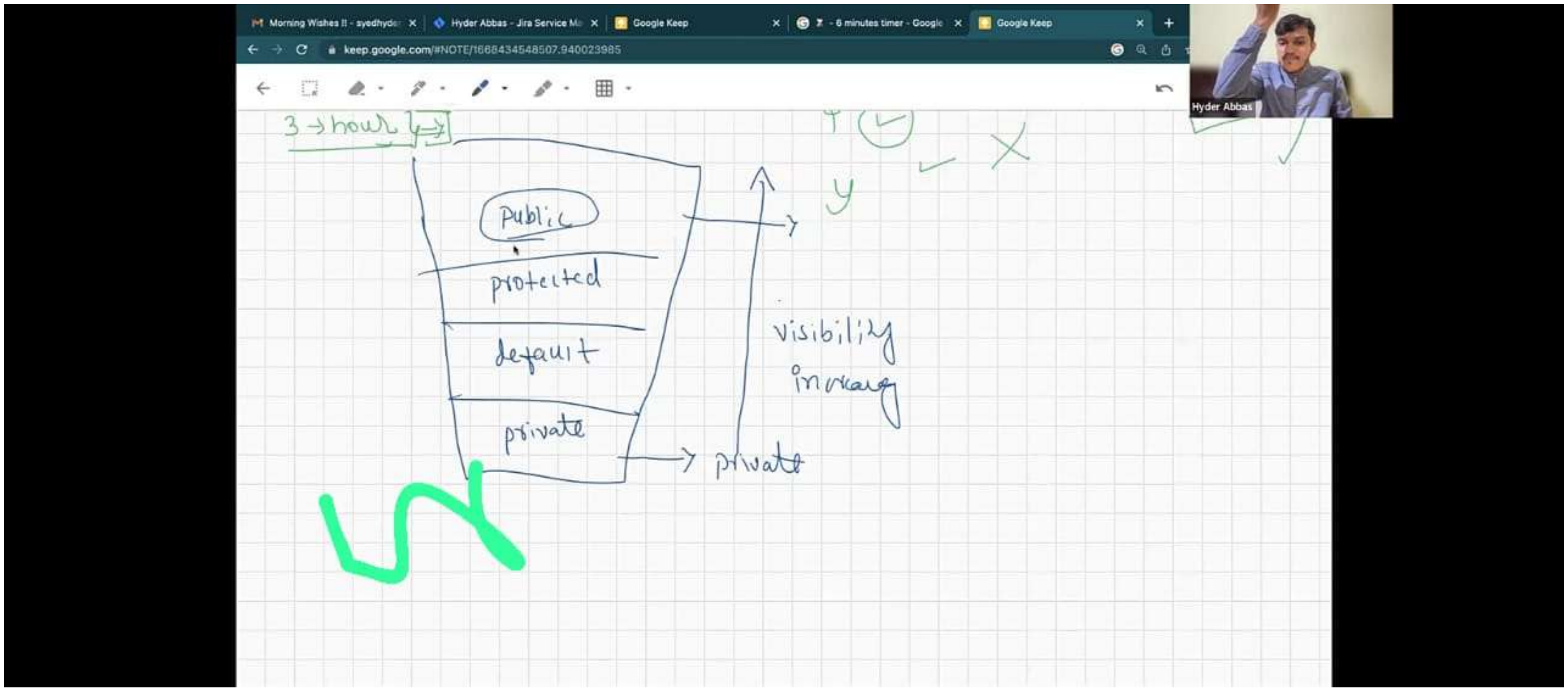
```
Q>.
System.out.println("Java"+1000+2000+3000); // "java1000"+2000+3000 => "java10002000" + 3000 => "java100020003000"

Q>.
System.out.println(1000+2000+3000+"Java");//3000+3000+"java" => 6000+"java" =>"6000java"

Q>.
System.out.println(7.7+3.3+"Java"+3.3+7.7);//11.0 + "java" + 3.3 +7.7 => "11.0 java"+3.3 => "11.0java3.3" + 7.7 => "11.0java3.37.7"

Q>.
System.out.println("ONE"+2+3+4+"FIVE");//"ONE2" + 3+4+"FIVE" => "ONE23" + 4 +"FIVE"=> "ONE234+" Five" => "ONE234Five"
```

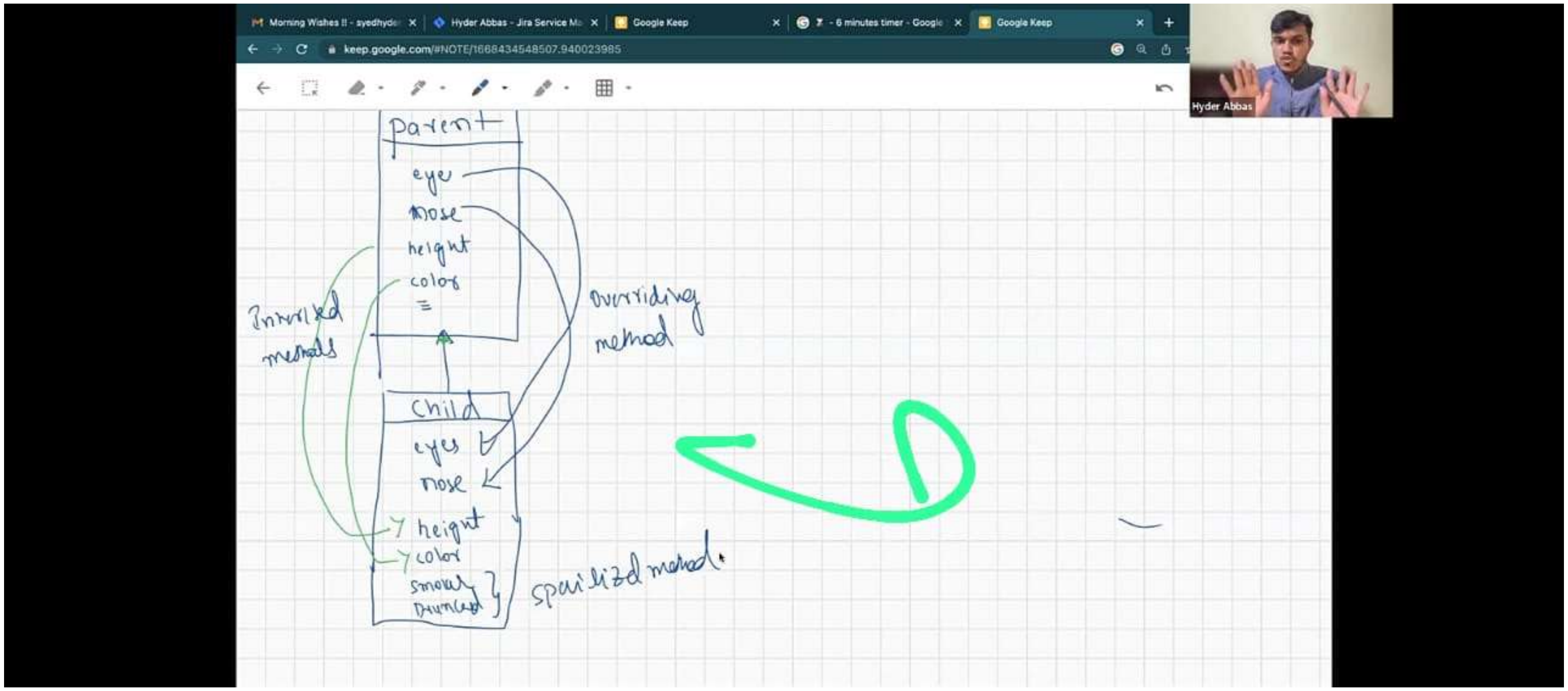
40

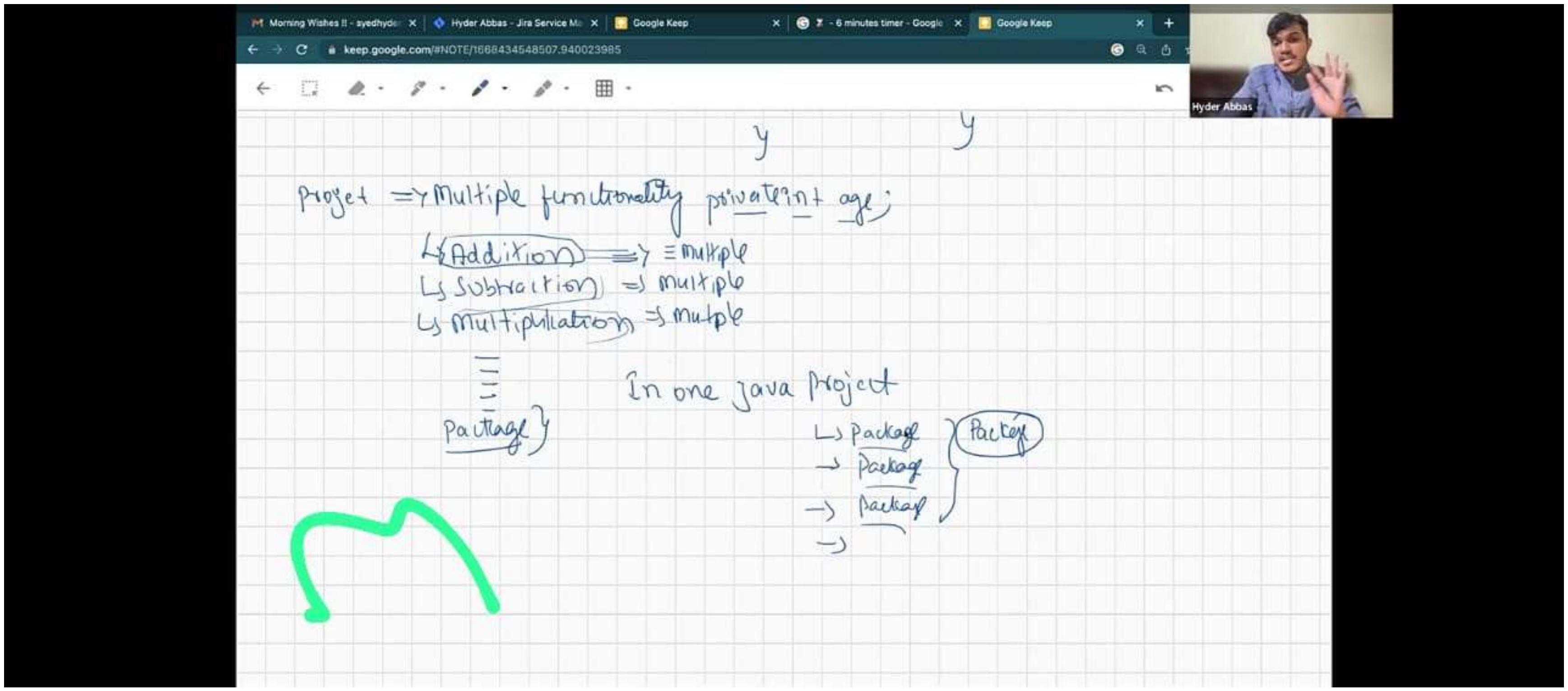


```
Q>.
String s1=" ";
System.out.println(s1.isBlank());//true
System.out.println(s1.isEmpty());//false

Q>.
String s2="sachin ramesh tendulkar";
System.out.println(s2.substring(8, 4));

A. CE
B. rame
C. in ram
D. NullPointerException
E. StringIndexOutOfBoundsException
F. ArrayIndexOutOfBoundsException

Q>.
String s1 = new String("JAVA");
String s2 = new String("JAVA");
System.out.println(s1 == s2);
System.out.println(s1.equals(s2));
```

4 1


3 → hour
public
protected
default
private
private
visibility
increasing
y
Hyder Abbas


Parent
eye
nose
height
color
Inherited methods
Overriding method
Child
eyes
nose
height
color
smoking
Drinking
specialized method



y y

Project => Multiple functionality private int age;

↳ Addition ==> ≡ multiple

↳ Subtraction => multiple

↳ Multiplication => multple

Package }

In one Java Project

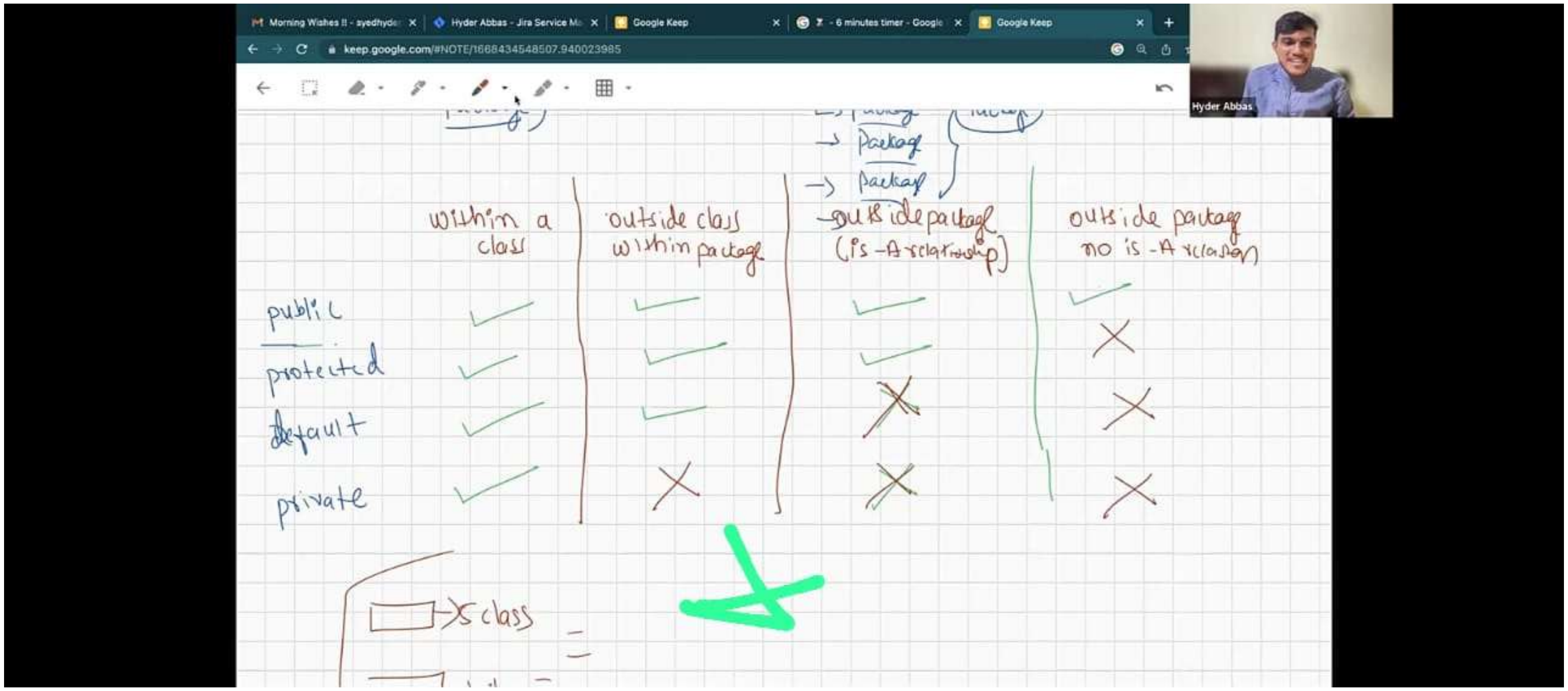
↳ Package

→ Package

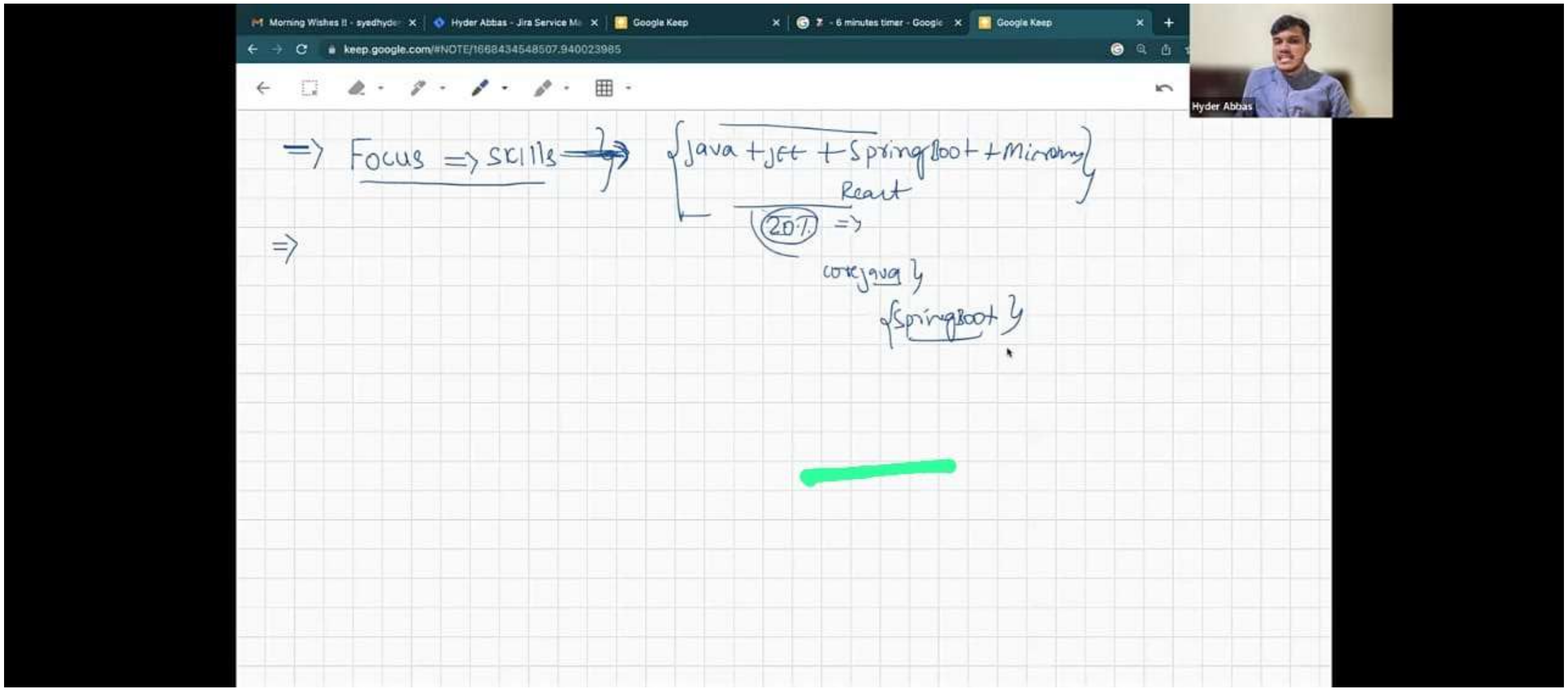
→ Package

→

Package



	within a class	outside class within package	outside package (is-A relationship)	outside package no is-A relation
public	✓	✓	✓	✓
protected	✓	✓	✓	✗
default	✓	✓	✗	✗
private	✓	✗	✗	✗



=> Focus => skills → {Java + JEE + Springboot + Microservices / React}

(20%) =>

corejava }

{Springboot }

=>