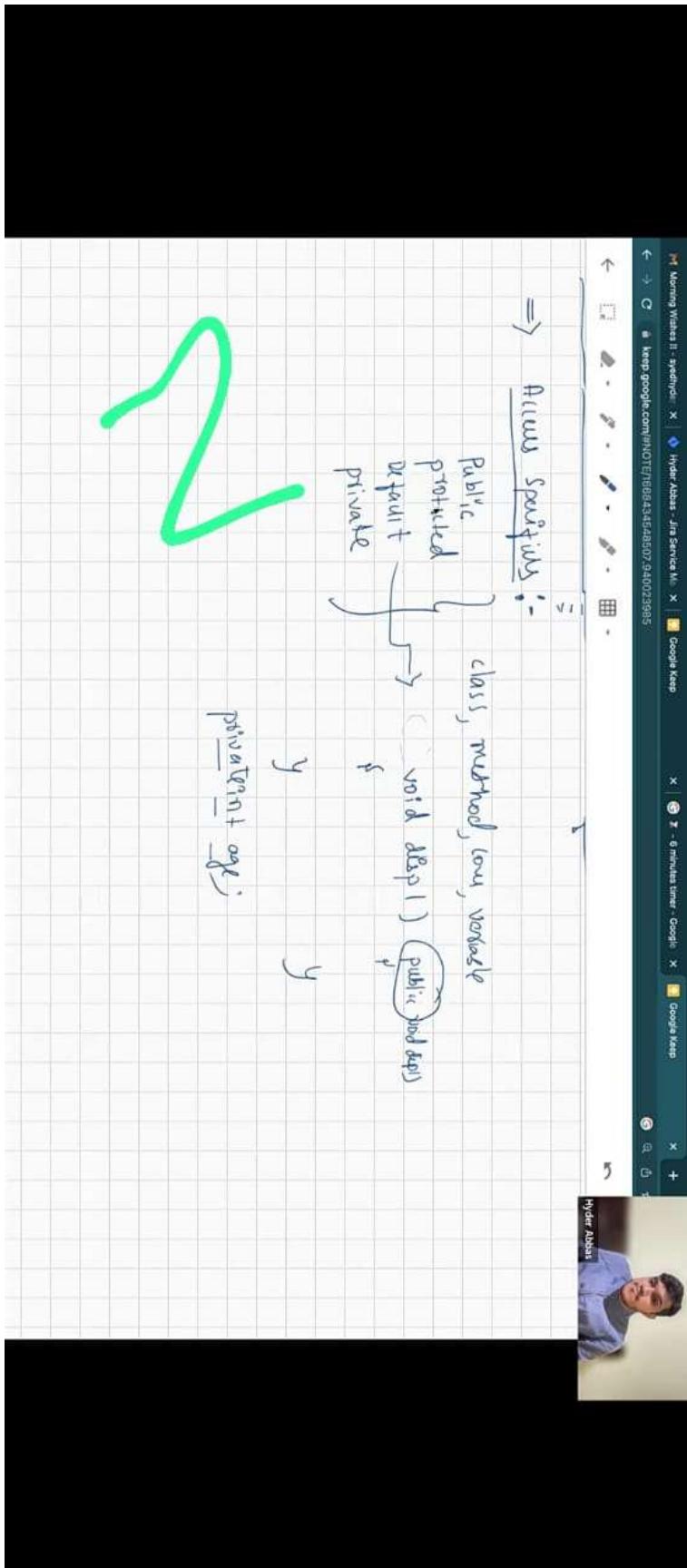
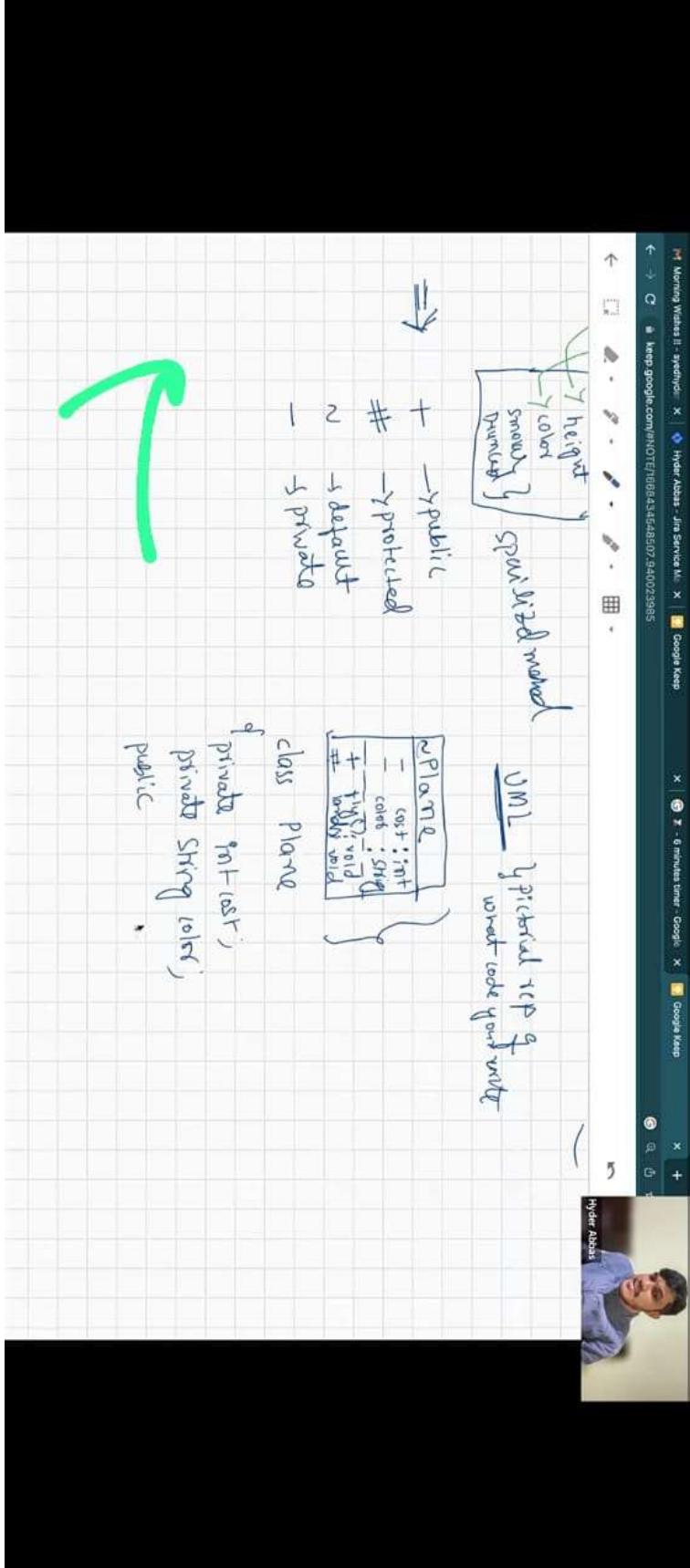
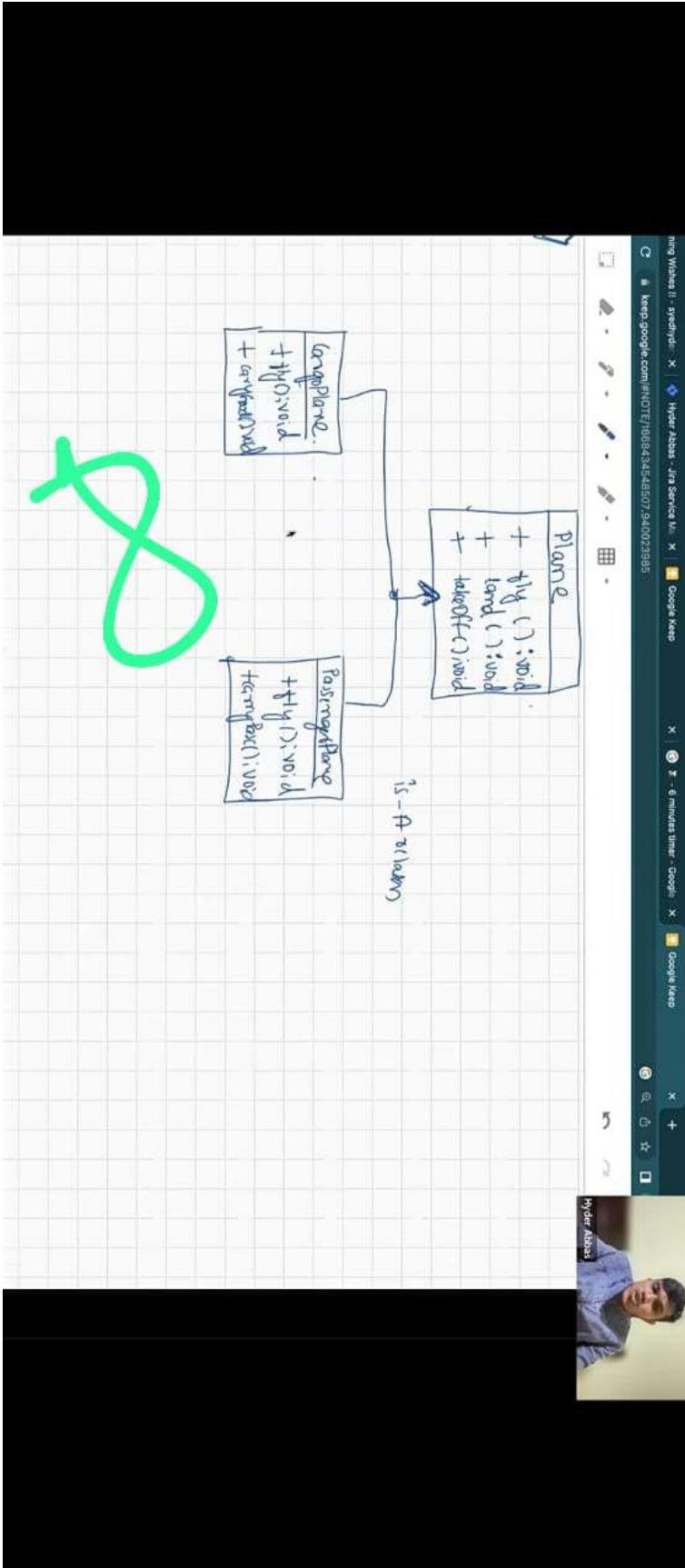


Java Inheritance







ming Wines II - smd919 - X | 🔍 Hyper Abbas - Jim Sarcina M. - X | Google Keep
 keep.google.com/nOTE/00083459807340023985





Java code showing inheritance:

```
1 package Engine X;
2
3 import javax.swing.*;
4 import java.awt.*;
5 import java.awt.event.*;
6 import java.awt.image.*;
7 import java.awt.Toolkit.*;
8 import java.awt.Container.*;
9 import java.awt.BorderLayout.*;
10 import java.awt.GridLayout.*;
11 import java.awt.GridBagLayout.*;
12 import java.awt.GridLayout.*;
13 import java.awt.GridLayout.*;
14 import java.awt.GridLayout.*;
15 import java.awt.GridLayout.*;
16 import java.awt.GridLayout.*;
17 import java.awt.GridLayout.*;
18 import java.awt.GridLayout.*;
19 import java.awt.GridLayout.*;
20 import java.awt.GridLayout.*;
21 import java.awt.GridLayout.*;
22 import java.awt.GridLayout.*;
23 import java.awt.GridLayout.*;
24 import java.awt.GridLayout.*;
25 import java.awt.GridLayout.*;
26 import java.awt.GridLayout.*;
27 }
```

The code defines a class `Plane` with methods `takeOff()` and `fly()`. It also defines `CargoPlane` and `PassengerPlane` classes that inherit from `Plane`.

Annotations in the code:

- `Hy der Abos` is a comment above the `System.out.println("Plane is flying");` line.
- `Bye the Abos` is a comment above the `System.out.println("Plane is landing");` line.

Video thumbnail:

1b

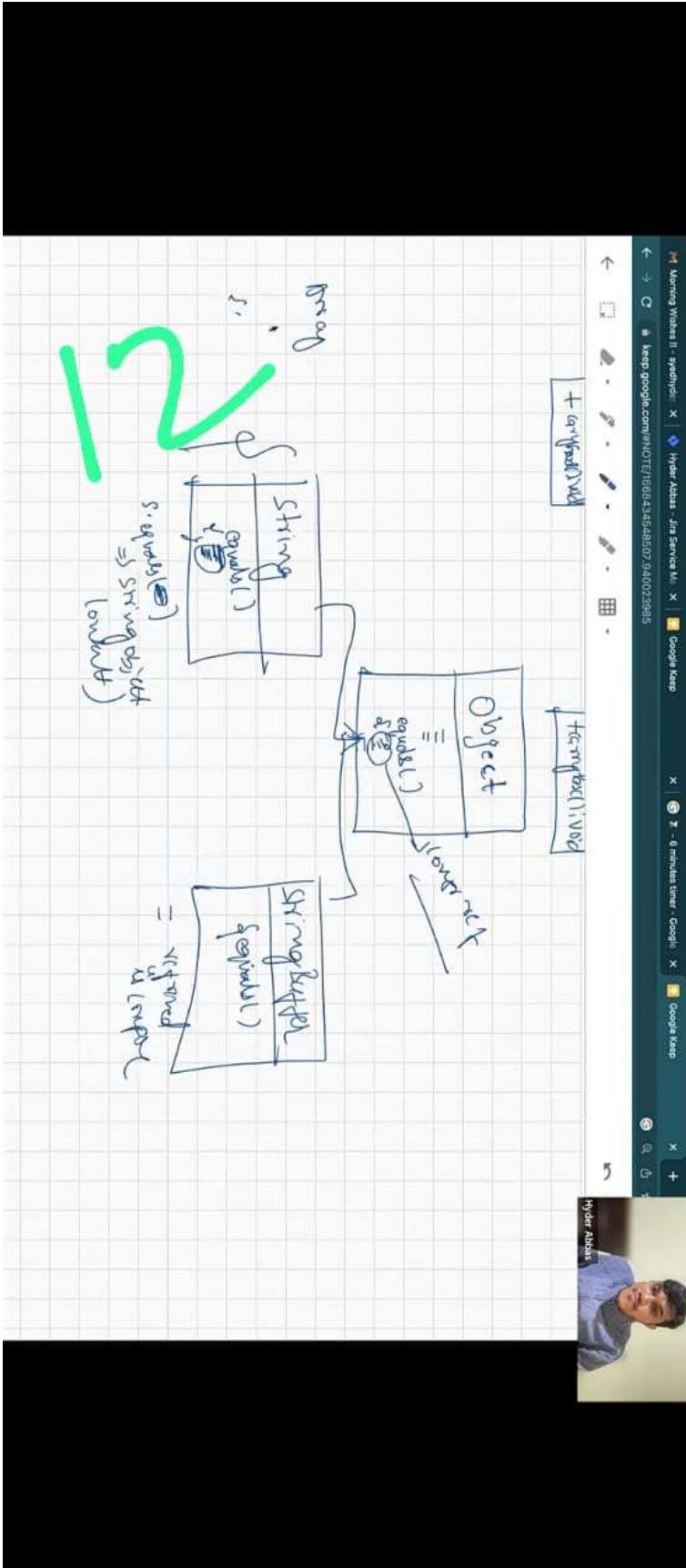
```
17 package Engine.X;
18 
19 class CargoPlane extends Plane
20 {
21     System.out.println("CargoPlane flies at lower height");
22 }
23 public void carryGoods()
24 {
25     System.out.println("cargoPlane carries goods");
26 }
27 class PassengerPlane extends Plane
28 {
29     public void fly()
30     {
31         System.out.println("PassengerPlane flies at medium height");
32     }
33     public void carryPassenger()
34     {
35         System.out.println("PassengerPlane carries passengers");
36     }
37 }
38 }
39 }
40 }
41 }
42 }
43 }
```

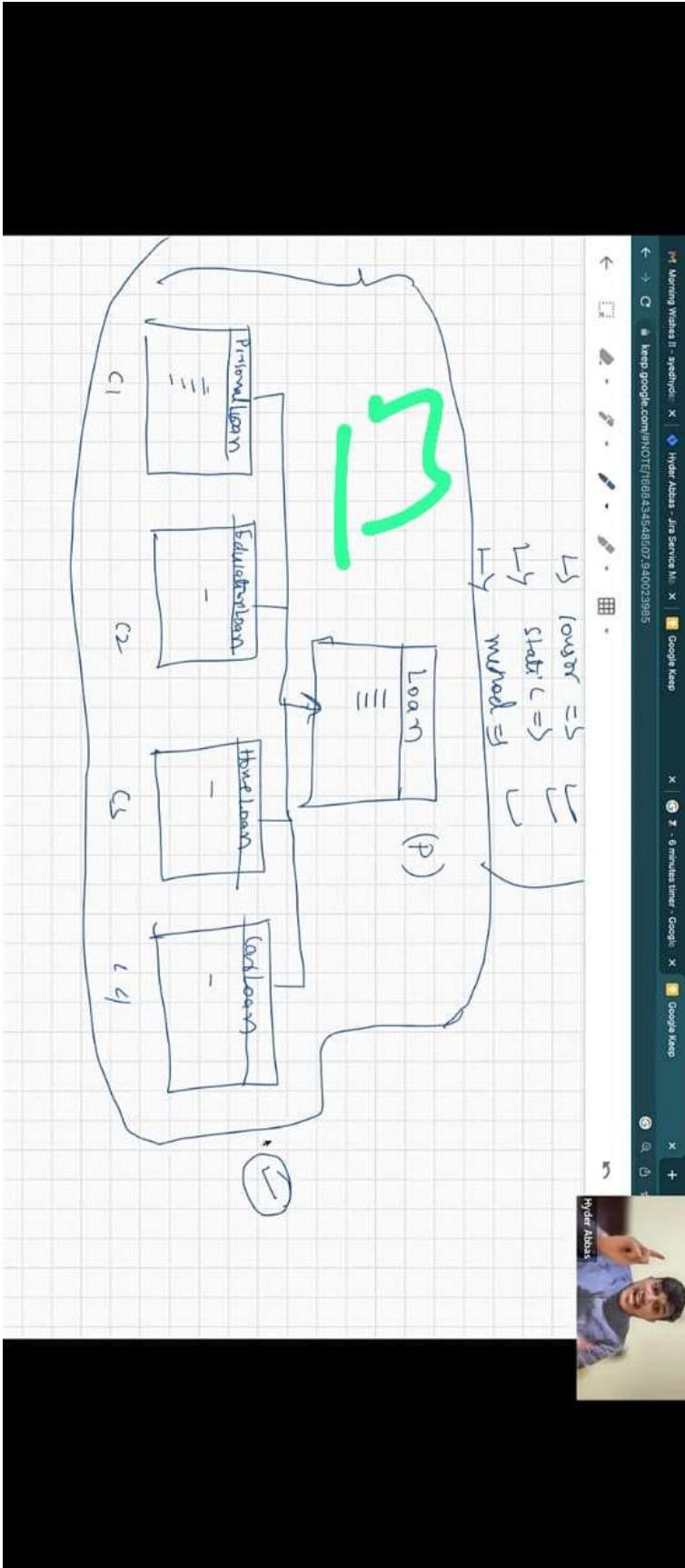


```
 36 }  
 37 }  
 38 }  
 39 }  
 40 }  
 41 }  
 42 }  
 43 }  
 44 }  
 45 }  
 46 }  
 47 }  
 48 }  
 49 }  
 50 }  
 51 }  
 52 }  
 53 }  
 54 }  
 55 }  
 56 }  
 57 }  
 58 }  
 59 }  
 60 }  
 61 }  
 62 }
```

The code is a Java program named `LaunchPlane.java`. It contains a `main` method that creates two plane objects: `CargoPlane cp` and `PassengerPlane pp`. It then calls methods on these objects: `takeOff()`, `carryGoods()`, `fly()`, and `landing()`. The output of the `System.out.println` statement is "PassengerPlane carries passengers".





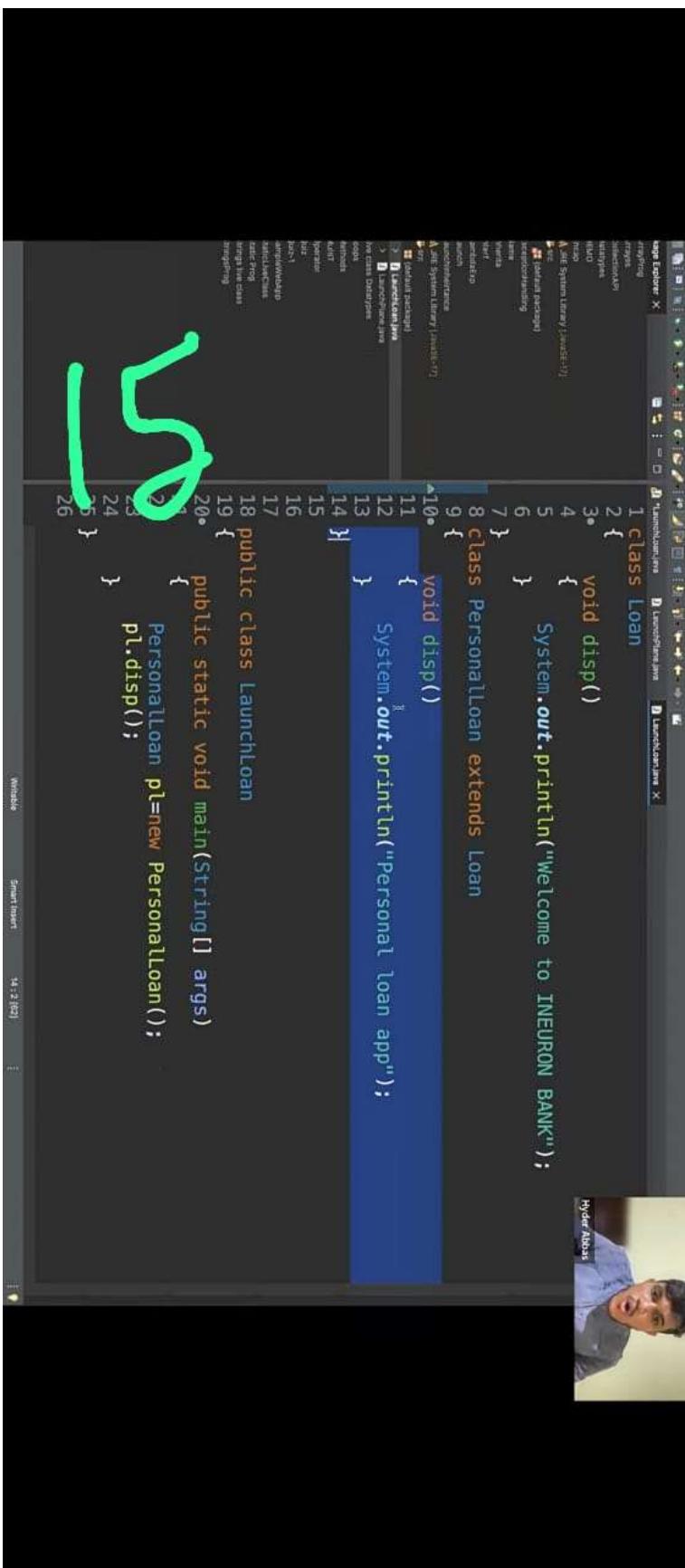




TELUSKO

	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	Yes	Yes	No

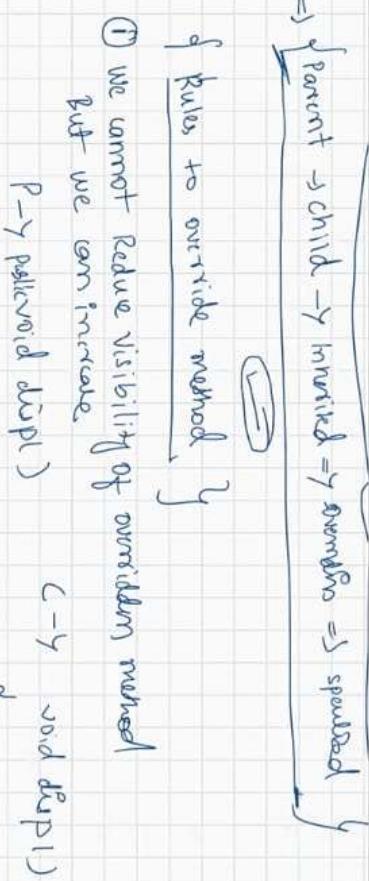
Subscribe
Telusko



```
Java Editor X  D NameLoans.java  D LemoopPhone.java  D LaunchLoans.java
 1 import
 2 import
 3 import
 4 import
 5 import
 6 import
 7 import
 8 class PersonalLoan extends Loan
 9 {
10     void disp()
11     {
12         System.out.println("Personal loan app");
13     }
14 }
15
16
17
18 public class LaunchLoan
19 {
20     public static void main(String[] args)
21     {
22         PersonalLoan pl=new PersonalLoan();
23         pl.disp();
24     }
25 }
26
```



16



.17

② Return type of overridden method must same
as that of overridden in parent.



```
1. package Engine;
2. import java.awt.*;
3. import javax.swing.*;
4. import java.awt.event.*;
5. import java.awt.image.*;
6. import java.io.*;
7. import java.util.*;
8. import java.awt.Toolkit;
9. import java.awt.BorderLayout;
10. import java.awt.Container;
11. import java.awt.GridLayout;
12. import java.awt.GridBagLayout;
13. import java.awt.GridBagConstraints;
14. import java.awt.Insets;
15. import java.awt.Window;
16. import java.awt.WindowEvent;
17. import java.awt.WindowListener;
18. import java.awt.WindowFocusListener;
19. import java.awt.WindowIconListener;
20. import java.awt.WindowListener;
21. import java.awt.WindowFocusListener;
22. import java.awt.WindowIconListener;
23. import java.awt.Window;
24. import java.awt.WindowEvent;
25. import java.awt.WindowIconEvent;
26. import java.awt.WindowFocusEvent;
27. import java.awt.WindowIconListener;
28. import java.awt.WindowFocusEvent;
29. import java.awt.WindowIconEvent;
30. import java.awt.WindowEvent;
31. import java.awt.WindowIconEvent;
32. import java.awt.WindowFocusEvent;
33. import java.awt.WindowIconListener;
34. import java.awt.WindowFocusEvent;
35. import java.awt.WindowIconEvent;
```

The code block contains several imports for Java's AWT (Abstract Window Toolkit) and Swing packages, including AWT's `Window`, `WindowEvent`, `WindowIconEvent`, `WindowIconListener`, `WindowFocusEvent`, and `WindowFocusListener`. It also includes imports for `Toolkit`, `BorderLayout`, `Container`, `GridLayout`, `GridBagLayout`, `GridBagConstraints`, and `Insets`.



19

- 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 superclass).



```
public class LaunchLoans {
    public class Loans {
        public Interest interest() {
            Interest it=new Interest();
            return it;
        }
    }
    public class PersonalLoan extends Loans {
        public InterestPersonalLoan interest() {
            InterestPersonalLoan ipl=new InterestPersonalLoan();
            return ipl;
        }
    }
}
```

The image shows a screenshot of a Java code editor. The code is a simple class definition for `LaunchLoans` containing two nested classes, `Loans` and `PersonalLoan`. Both classes implement a method `interest()` that returns an object of type `Interest`. The code is annotated with line numbers from 1 to 33. A large, hand-drawn green number '20' is drawn over the first 20 lines of the code. In the bottom right corner of the code area, there is a small portrait of a man with a mustache.

21

```
1 package com.lunchplan;
2
3 import java.util.List;
4
5 import com.lunchplan.domain.LunchPlan;
6
7
8 class InterestPersonalLoan extends Interest
9 {
10     class Loans
11     {
12         public Interest interest()
13         {
14             Interest it=new Interest();
15             return it;
16         }
17     }
18 }
19
20 class PersonalLoan extends Loans
21 {
22     public InterestPersonalLoan interest()
23     {
24         InterestPersonalLoan ipl=new InterestPersonalLoan();
25         return ipl;
26     }
27 }
```

22

- (3) Return type of overridden method and covariance can be different as not of parent if it is covariant return type (return type is -A returning).
- (4) \Rightarrow parameters of overridden method must be same as not of parent else it will be considered as spurious method considering method overriding.





```
1. package Engine;
2. 
3. import java.awt.*;
4. import javax.swing.*;
5. import java.awt.event.*;
6. import java.awt.image.*;
7. import java.util.*;
8. import java.io.*;
9. import java.net.*;
10. import java.awt.Container;
11. import java.awt.GridLayout;
12. import java.awt.BorderLayout;
13. import java.awt.GridBagLayout;
14. import java.awt.GridBagConstraints;
15. import java.awt.Insets;
16. import java.awt.Window;
17. import java.awt.WindowEvent;
18. import java.awt.WindowListener;
19. import java.awt.WindowFocusListener;
20. import java.awt.WindowIconListener;
21. import java.awt.WindowListener;
22. import java.awt.WindowFocusListener;
23. import java.awt.WindowIconListener;
24. import java.awt.WindowEvent;
25. import java.awt.Window;
26. import java.awt.Window;
27. import java.awt.Window;
28. import java.awt.Window;
29. import java.awt.Window;
30. import java.awt.Window;
31. import java.awt.Window;
32. import java.awt.Window;
33. import java.awt.Window;
34. import java.awt.Window;
35. import java.awt.Window;
36. import java.awt.Window;
```

```
10. }  
11. }  
12. }  
13. }  
14. }  
15. }  
16. }  
17. }  
18. }  
19. }  
20. }  
21. }  
22. }  
23. class Demo2 extends Demo1  
24. {  
25.     public void disp()// we can increase visibility  
26.     {  
27.         void disp2() we cannot reduce visibility  
28.     }  
29.     //  
30.     //  
31.     //  
32.     //  
33.     //  
34.     //  
35.     //  
36. }
```



10:41 PM

```
22 package Demo;
23
24 class Demo2 extends Demo1 {
25     public void disp() // we can increase visibility
26     {
27         // ...
28     }
29     // void disp2() we cannot reduce visibility
30     // ...
31     // ...
32     // ...
33     // public void add() return type cannot be changed
34     // ...
35     // System.out.println("Child");
36     // ...
37     // public int add(int a, int b)
38     // ...
39     // ...
40     // ...
41     // ...
42     public int add(int a)
43     {
44         return a;
45     }
46 }
```

By der Abst



6 dB 6: LTE .11 Ks 64

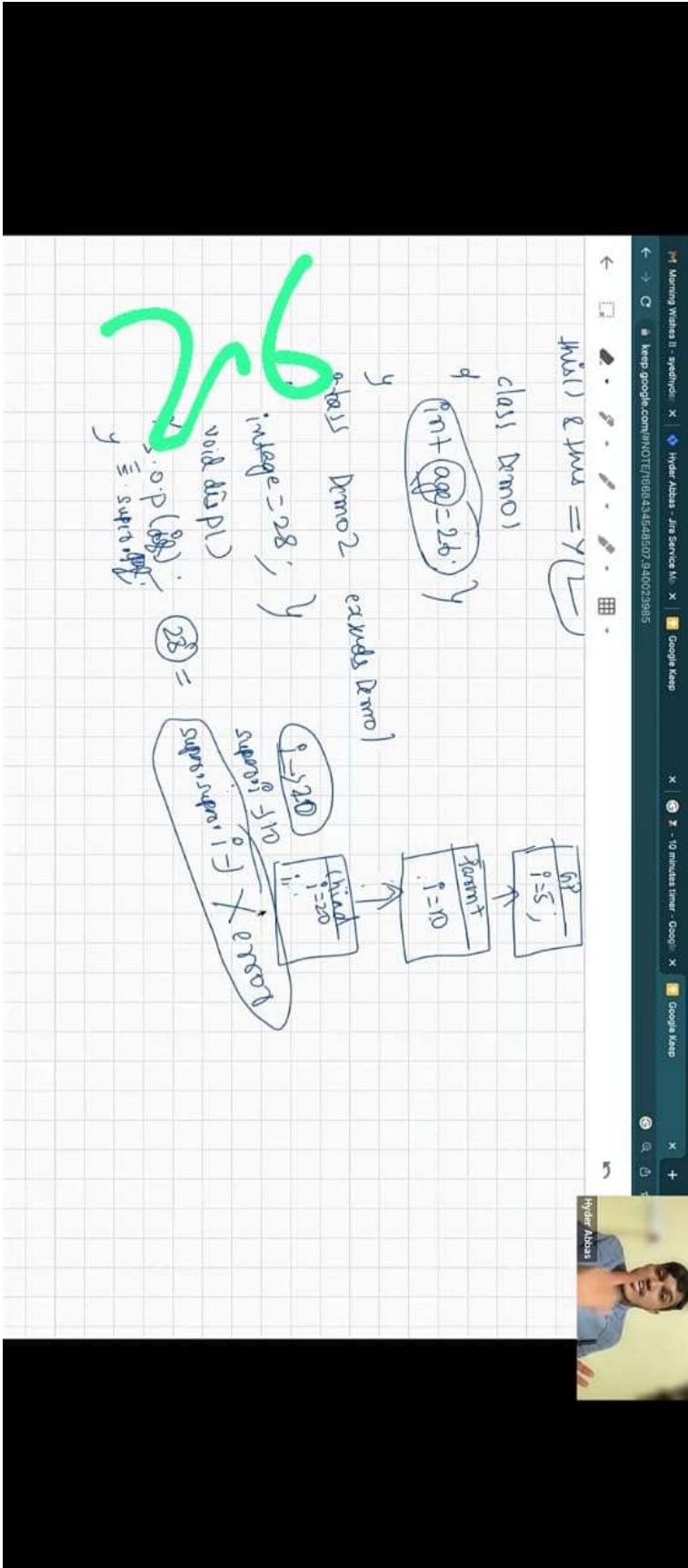
considering method overloading.

super() = inside \rightarrow constructor
↳ it will call parent class

standardized samples

$\Theta \equiv \text{dist} \{ \cdot \}$

3



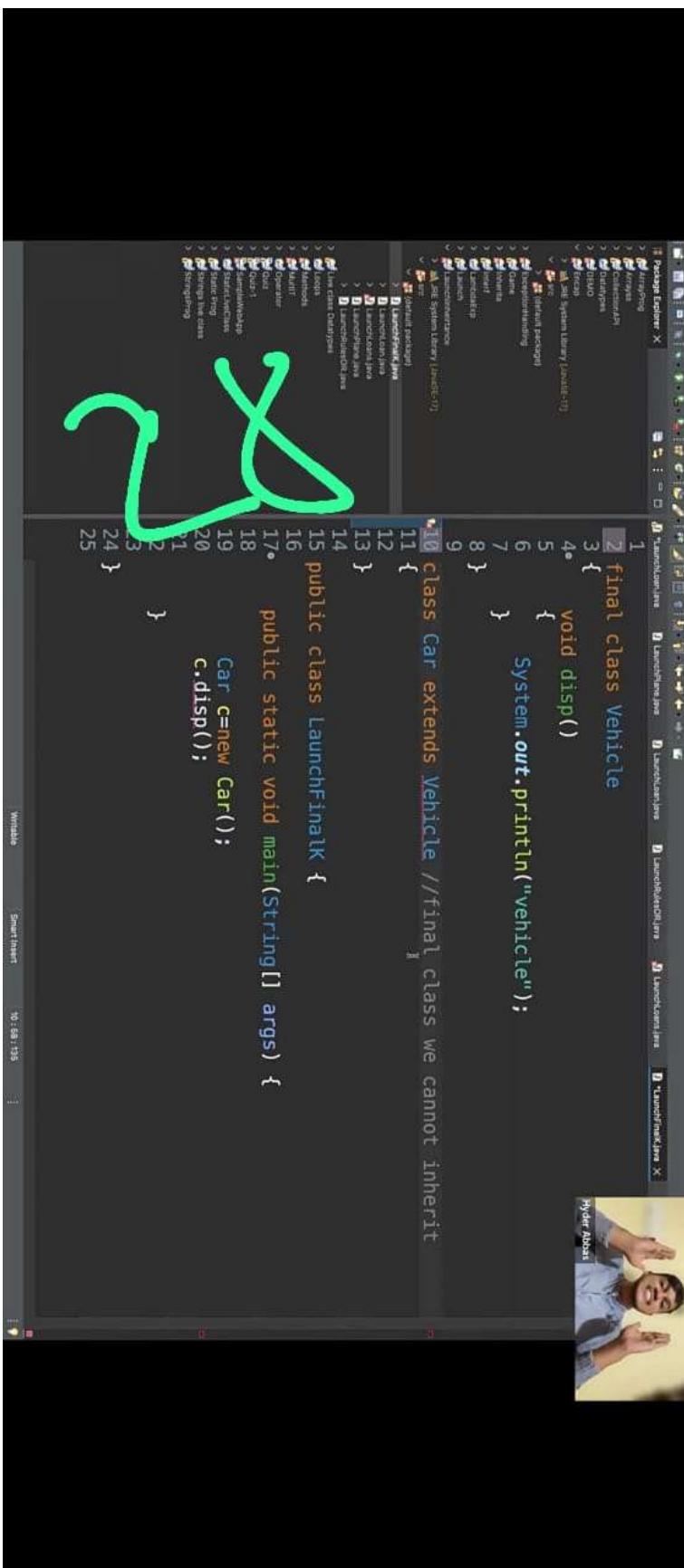
27

final keywords

final = class
final = method
final = variable

new
to
make
obj





```
1 package Engine;
2
3 import java.awt.*;
4 import javax.swing.*;
5 import java.awt.event.*;
6 import java.awt.image.*;
7 import java.awt.Toolkit;
8 import java.awt.Container;
9 import java.awt.BorderLayout;
10 import java.awt.GridLayout;
11 import java.awt.GridBagLayout;
12 import java.awt.GridLayout;
13 import java.awt.GridBagLayout;
14 import java.awt.GridLayout;
15 import java.awt.GridBagLayout;
16 import java.awt.GridLayout;
17 import java.awt.GridBagLayout;
18 import java.awt.GridLayout;
19 import java.awt.GridBagLayout;
20 import java.awt.GridLayout;
21 import java.awt.GridBagLayout;
22 import java.awt.GridLayout;
23 import java.awt.GridBagLayout;
24 import java.awt.GridLayout;
25 import java.awt.GridBagLayout;
```

```
1 package Engine;
2
3 final class Vehicle {
4     void disp() {
5         System.out.println("vehicle");
6     }
7 }
8
9 class Car extends Vehicle //final class we cannot inherit
10
11 {
12 }
13 }
14
15 public class LaunchFinalK {
16
17     public static void main(String[] args) {
18         Car c=new Car();
19         c.disp();
20     }
21 }
22
23
24
25 }
```



```
1 package Engine.X;
2
3 import java.awt.*;
4 import javax.swing.*;
5 import java.awt.event.*;
6 import java.awt.image.*;
7 import java.awt.Container;
8 import java.awt.BorderLayout;
9 import java.awt.GridLayout;
10 import java.awt.GridBagLayout;
11 import java.awt.GridLayout;
12 import java.awt.GridBagLayout;
13 // final class doesn't participate in inheritance
14 //-
15 class Vehicle {
16     final void disp() {
17         System.out.println("Vehicle");
18     }
19 }
20 }
21 }
22 class Car extends Vehicle {
23     void disp() {
24         System.out.println("Car");
25     }
26     //-
27     public class LaunchFinalX {
28         public void main(String[] args) {
29             public class LaunchFinalX {
30                 public static void main(String[] args) {
31                     Car c=new Car();
32                     c.disp();
33                 }
34             }
35         }
36     }
37 }
38 }
39 }
```

Wimble

Smart Home

29.7.2006

By der Abbau

The screenshot shows a Java development environment with a code editor and a video player.

Code Editor:

```
10.3.1 - W100
11 Package Explorer X
12 Project Navigator
13 Properties
14 Javadoc
15 LaunchTime.java
16 LaunchFinal.java
17 LaunchMain.java
18 LaunchVehicle.java
19 LaunchFinal.java
20 LaunchFinal.java
21 LaunchFinal.java
22 LaunchFinal.java
23 LaunchFinal.java
24 LaunchFinal.java
25 LaunchFinal.java
26 LaunchFinal.java
27 LaunchFinal.java
28 LaunchFinal.java
29 LaunchFinal.java
30 LaunchFinal.java
31 LaunchFinal.java
32 class LaunchFinal {
33
34
35 public static void main(String[] args) {
36     Car c=new Car();
37     c.disp();
38
39
40 }
```

Annotations:

- A large green hand-drawn heart is drawn over the code editor area.
- A small red arrow points from line 20 to the line 20 annotation.
- An annotation at line 20 states: "final int i=10; acts as constant we cannot change the value".
- An annotation at line 27 states: "oid disp() final will get inherited but we cannot override".

Video Player:

The video player shows a man in a blue shirt, identified as "Hyder Abbas". The video progress bar indicates it is at 0:00 of a 1:11 duration.

14.11.2022 Snippets_Discussion - Notepad

File Edit View

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
F. 8

Answer: G

14.11.2022 20°C Partly cloudy

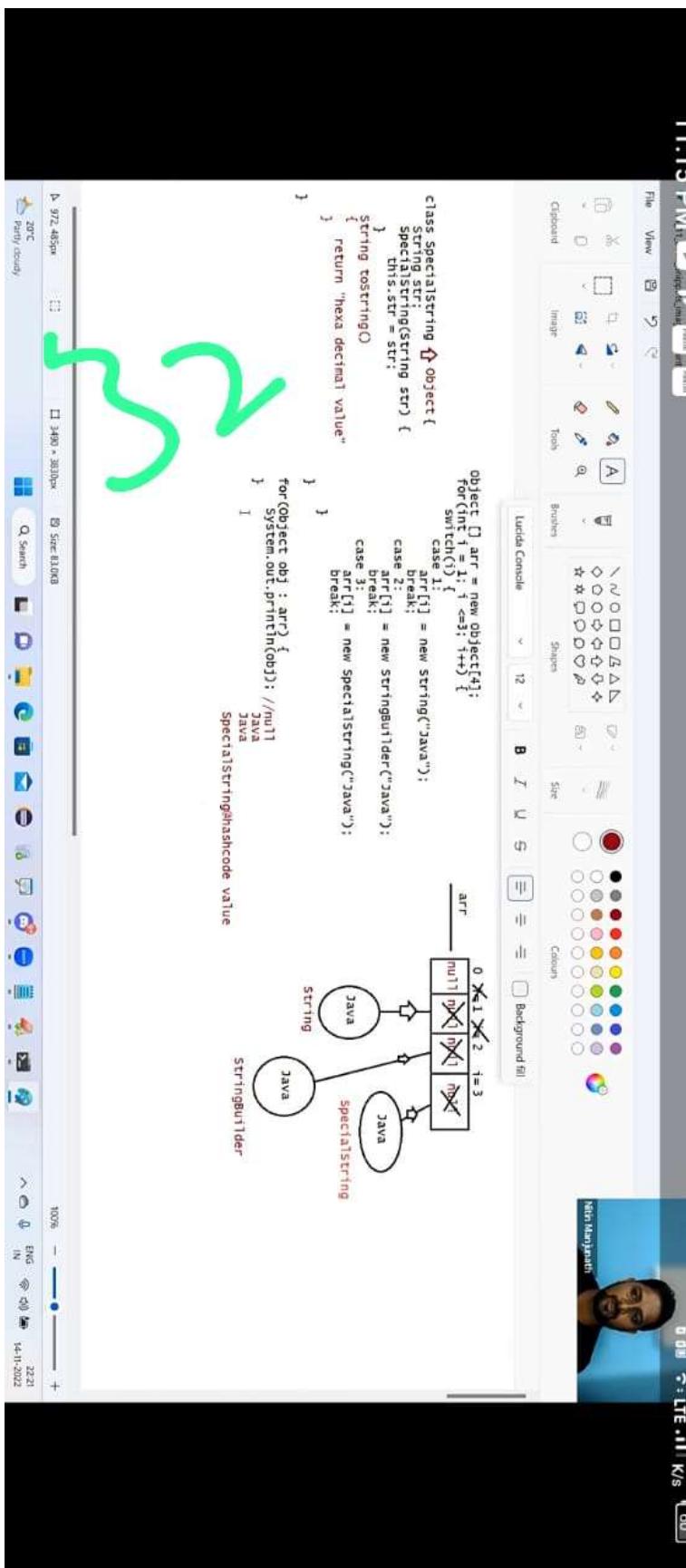
Search

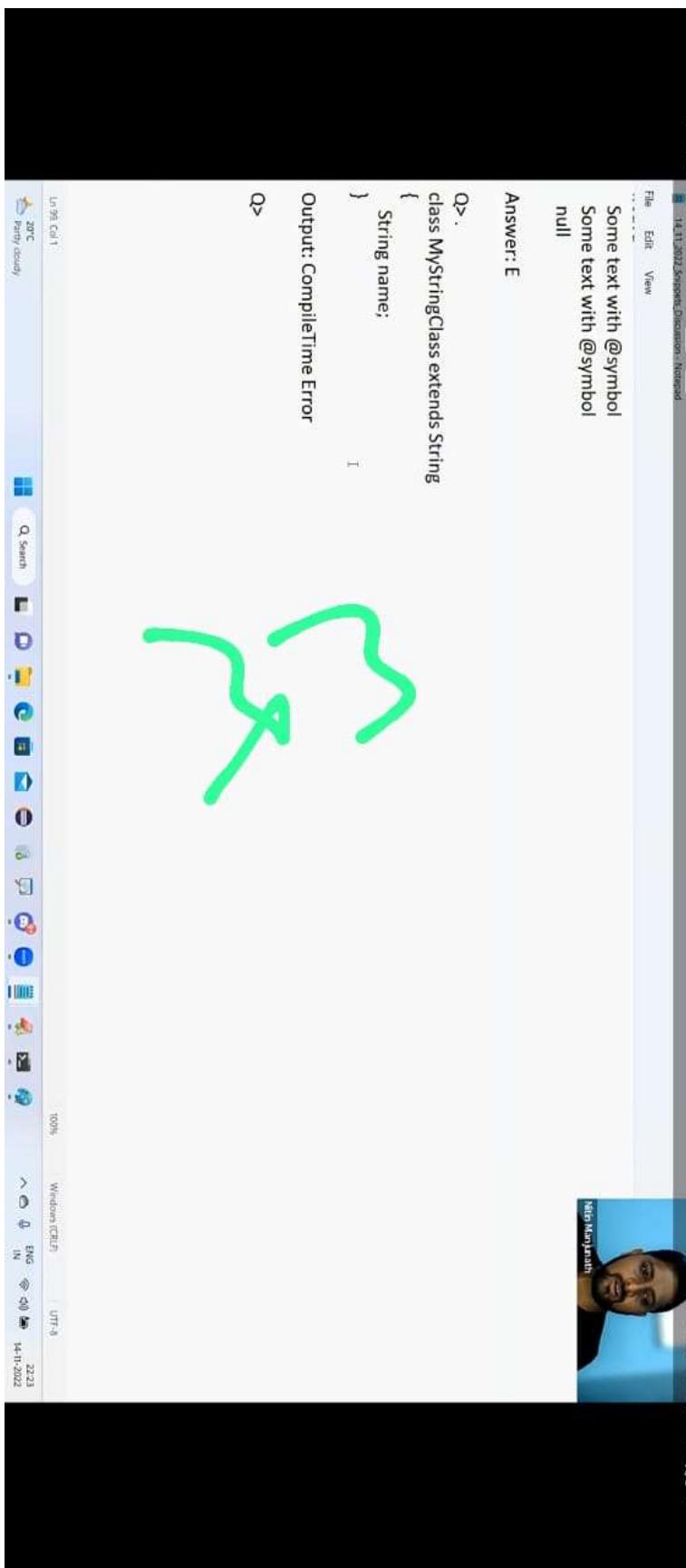
Windows (CEP7)

UTF-8

14.11.2022 22:06







*14.11.2022-Snippet-Discussion - Nonprod

File Edit View

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

Output: CompileTime Error

Q>

```
String name = "sachinrameshendar".substring(4);
System.out.println(name); //Inrameshendar
```

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
```

LN 110, Col 1

20°C Party cloudy

Q. Search

100% Windows (CET) UTF-8

ENG ⇄ DE 🇩🇪 22:27 14.11.2022

3X

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

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"));
```

14.11.2022 20°C, Gurgaon, Partly cloudy

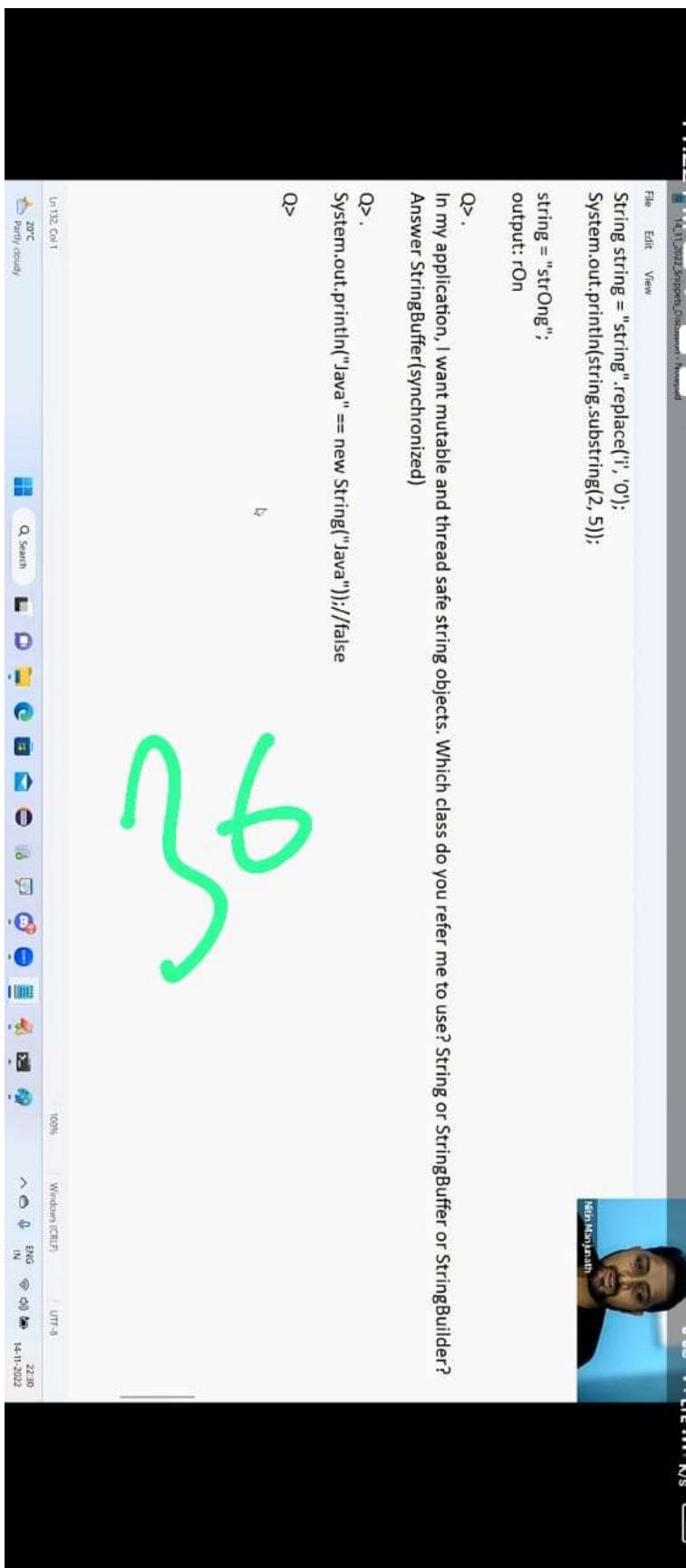
Q_Switch

14.11.2022 22:29

Windows (CEP7) | UTF-8

Nitin Manjrekar

35



14.11.2022-Swiper-Discussion - Normal

File Edit View

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

O>.

```
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>.

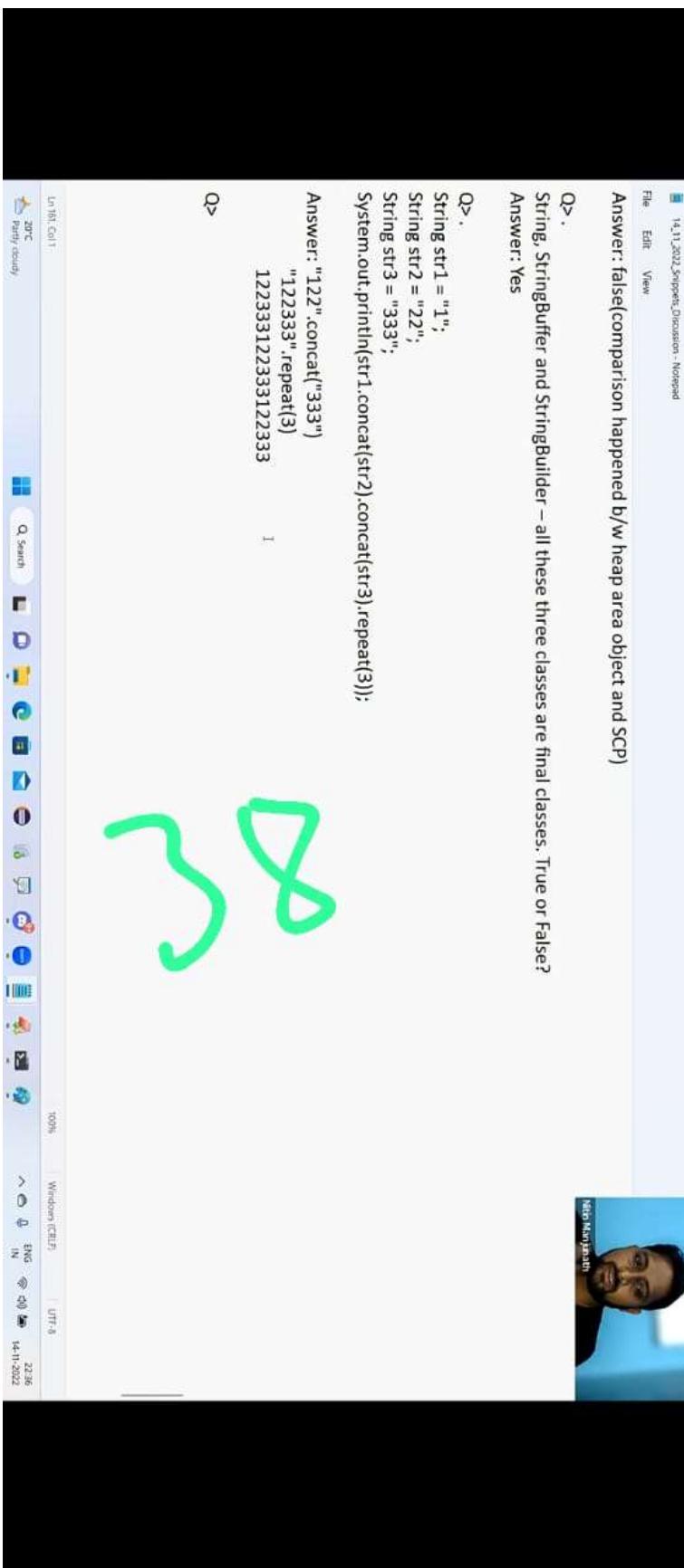
String, StringBuffer and StringBuilder – all these three classes are final classes. True or False?

Q>.

```
String str1 = "1";  
String str2 = "22";  
String str3 = "333";
```

100% Windows (C:\Users\Nitin Manjrekar) UTF-8

37



14.11.2022 Snippet Discussion - Notepad

File Edit View

```
"12333".repeat(3)
123331233312333
```

Nitin Marupindi

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.+3.3+"Java"+3.3+7.7);//11.0 +"java" + 3.3 +7.7 => "11.0.java" +3.3 => "11.0.java3.3" + 7.7 => ""11.0.java3.3.7"
Q>
System.out.println("ONE"+2+3+4+"FIVE");
```

117, Cn 112

20°C Party cloudy

Q. Search

100% Windows (UTF-8) UTF-8

ENG ENG IN IN 14.11.2022 22:41

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

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

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

4 -

20°C Party cloudy

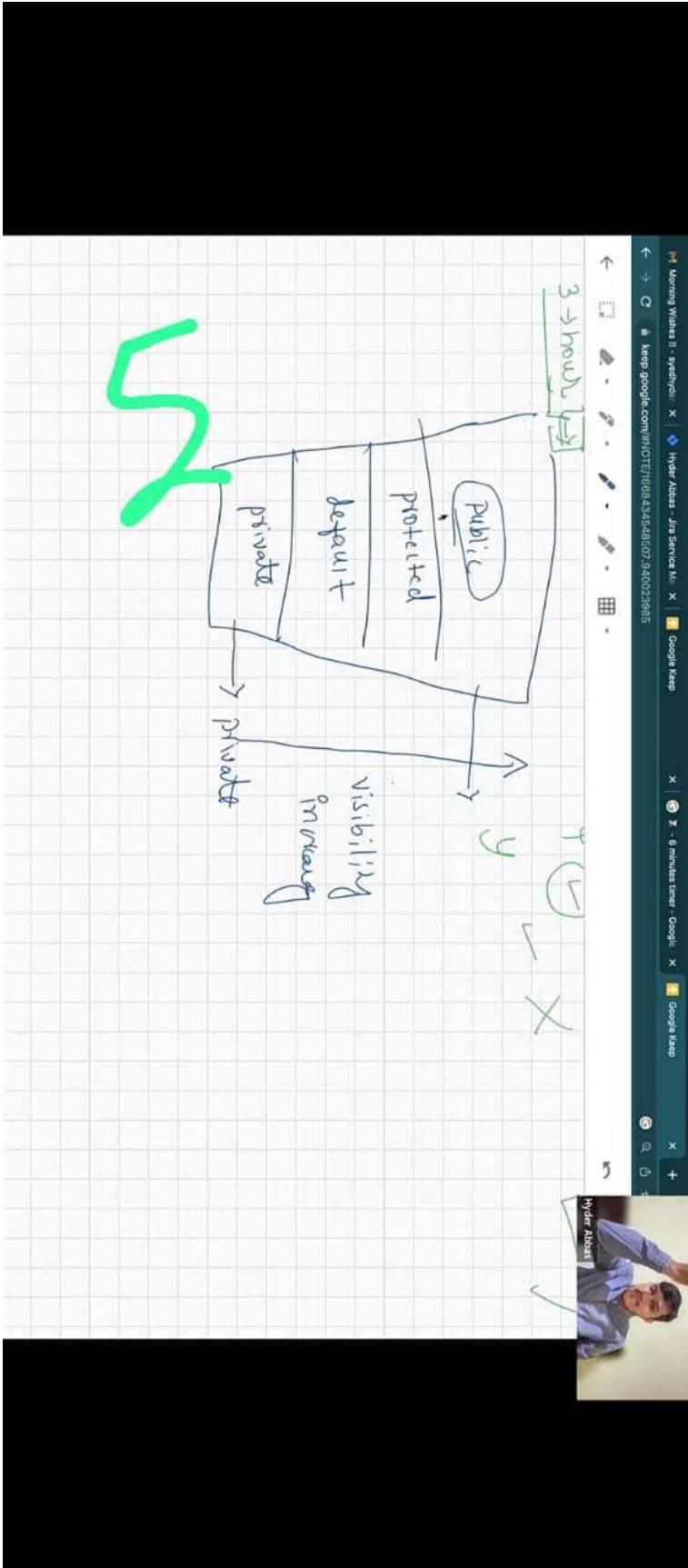
Q. Search

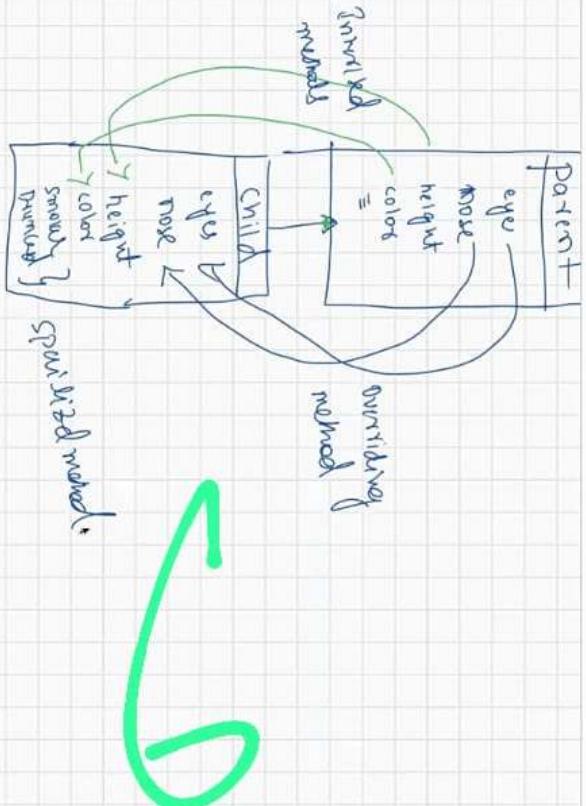


In 183. Col 37 100% Windows (CEP) | UTF-8 ENG ⇄ 22:45 IN 14.11.2022



File Edit View
14.11.2022 Snippets_Discussion - Notepad





Project \Rightarrow Multiple functioning patients per age:

ορθογωνίοι

Addition = multiple
multiple

Subtraction = multiple

↳ multiplication \Rightarrow multiple

11

In one J.

Partage

(f)

100

ANSWER

1

In one Java project

↳ Packag

→ Paket

→ Paulsen

11

