Static scope rule : Another example

```
class StaticScope {
     public static void main(String args[]) {
          int x; // known to all code within main
          x = 10;

          if(x == 10) {          // start new scope
               int y = 20;       // known only to this block
               System.out.println("x and y: " + x + " " + y);
               x = y * 2;        // x and y both are known here.
          }
          y = 100;               // Error! y is now no more known here
          System.out.println("x is " + x); // x is still known here.
     }
}
```

Static scope rule

```
class Box{
    float x = 10.0;
    float y = 20.0;
    float w = 15.0;

    float area(){
        return(2*(x*y + x*w + y*w));
    }
}
```

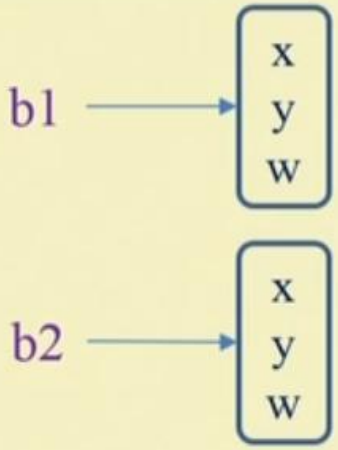
```
class Circle{
    float x = 0.0;
    float y = 0.0;
    float r = 5.0;

    float area(){
        return(((22/7)*r*r));
    }
}
```

```
class GeoClass{
    float x = 50;
    float y = 60;
    public static void main(String args[]){
    Box b = new Box();
    Circle c = new Circle();
    System.out.println("GeoClass Data: x = " + x);
    System.out.println("Box Data: x = " + b.x);
    System.out.println("Box Area: " + b.area);
    System.out.println("Circle Data: x = " + c.x);
    System.out.println("Circle Area: " + c.area);
    }
}
```

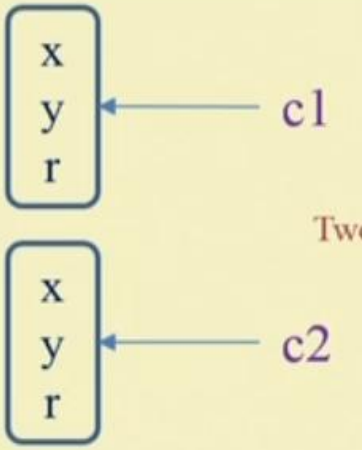
Static scope rule : Another example

```
class StaticScope {
     public static void main(String args[]) {
          int x; // known to all code within main
          x = 10;

          if(x == 10) {          // start new scope
               int y = 20;       // known only to this block
               System.out.println("x and y: " + x + " " + y);
               x = y * 2;        // x and y both are known here.
          }
          y = 100;               // Error! y is now no more known here
          System.out.println("x is " + x); // x is still known here.
     }
}
```

Instance variable versus Class variable



Two instances of class **Box**



Two instances of class **Circle**

Instance variable versus Class variable

- Java does not allow global variables.
- Every variable in Java must be declared inside a class.
- The keyword `static` is used to make a variable just like global variable.
- A variable declared with `static` keyword is called **class variable.**
- It acts like a global variable, that is, there is only one copy of the variable associated with the class.
  That is, one copy of the variable regardless of the number of instances of the class.

Static variable : An example

```
public class Circle{
   static int circlecount = 0; // class variable
   public double x,y,r; // instance variables
   public Circle(double x, double y, double r){
      this.x = x; this.y = y; this.r = r;
      circlecount++;
   }
   public Circle(double r){
      this(0.0,0.0,r);
      circlecount++;
   }
   public Circle(Circle c){
      this(c.x,c.y,c.r);
      circlecount++;
   }
   public Circle(){
      this(0.0,0.0,0.1);
       circlecount++;
   }
```

```
   public double circumference(){
       return (2*3.14159*r);
    }
    public double area(){
        return(3.14159*r*r);
    }
    public static void main(String args[ ]){
       Circle c1 = new Circle();
       Circle c2 = new Circle(5.0);
       Circle c3 = new Circle(c1);
       System.out.println("c1#" + c1.circlecount + "c2#" +
                          c2.circlecount + "c3#" + c3.circlecount);
    }
}
```

Declaring static method : An example

```
// A class method and instance method
public class Circle{
     public double x,y,r;
     // All constructors are here.
     // An instance method. Return the bigger of two circles.
     public Circle bigger(Circle c){
        if(c.r>r) return c;
        else return this;
     }
     // A class method: Return the bigger of two classes.
     public static Circle bigger (Circle a, Circle b) {
         if (a.r > b.r) return a;
         else return b:
     }

public static void main(String args[]){
         Circle a = new Circle (2.0);
         Circle b = new Circle (3.0);
         Circle c = a.bigger (b);        // Call of the instance method
         Circle d = Circle.bigger (a,b); // Call of the class method
    }
}
```

Declaring static method : An example

```
// A class method and instance method
public class Circle{
     public double x,y,r;
     // All constructors are here.
     // An instance method. Return the bigger of two circles.
     public Circle bigger(Circle c){
        if(c.r>r) return c;
        else return this;
     }
     // A class method: Return the bigger of two classes.
     public static Circle bigger (Circle a, Circle b) {
         if (a.r > b.r) return a;
         else return b:
     }

public static void main(String args[]){
         Circle a = new Circle (2.0);
         Circle b = new Circle (3.0);
         Circle c = a.bigger (b);        // Call of the instance method
         Circle d = Circle.bigger (a,b); // Call of the class method
    }
}
```

Nested class in Java

- In Java, a class can be defined inside a class. Let us look at the following example.

```
class Circle{
    static double x,y,r;
    Circle(double r){
        this.r = r;
    }
// Following is the nested class
   public static class Point{
      double x, y;
      void display(){
           System.out.println("(x,y): (" + this.x + "," + this.y + ")");
      }
      Point(double a, double b){
          this.x = a;
          this.y = b;
      }
  }
Continued to the next slide.....
```

Nested class in Java

- In Java, a class can be defined inside a class. Let us look at the following example.

```
class Circle{
    static double x,y,r;
    Circle(double r){
        this.r = r;
    }
// Following is the nested class
   public static class Point{
      double x, y;
      void display(){
           System.out.println("(x,y): (" + this.x + "," + this.y + ")");
      }
      Point(double a, double b){
          this.x = a;
          this.y = b;
      }
   }
Continued to the next slide.....
```

Nested class in Java

- In Java, a class can be defined inside a class. Let us look at the following example.

```
    public boolean isInside(Point p){
        double dx = p.x - x;
        double dy = p.y - y;
        double distance = Math.sqrt((dx*dx)+(dy*dy));
        if(distance < r) return true;
        else return false;
    }
    public static void main(String args[]){
            Circle a = new Circle (2.0);
            Point pa = new Point (1.0,2.0);
            pa.display();
            System.out.println("Is the points (1,2) inside the circle with radius 2 :"+a.isInside(pa));
            Circle b = new Circle (1.0);
            Point pb = new Point (3.0,3.0);
            System.out.println("Is the point (3,3) inside the circle with radius 1 :"+b.isInside(pb));
    }
}
```

Nested class in Java

- In Java, a class can be defined inside a class. Let us look at the following example.

```
class Circle{
    static double x,y,r;
    Circle(double r){
        this.r = r;
    }
// Following is the nested class
   public static class Point{
      double x, y;
      void display(){
           System.out.println("(x,y): (" + this.x + "," + this.y + ")");
      }
      Point(double a, double b){
          this.x = a;
          this.y = b;
      }
  }
Continued to the next slide.....
```

Recursion in Java : Calculation of *n*!

- Factorial calculation of *n*, an integer value is defines as follows.

$$n! = n \times (n\text{-}1) \times (n\text{-}2) \times \ldots 3 \times 2 \times 1$$

$$= 1 \times 2 \times 3 \times \ldots (n\text{-}2) \times (n\text{-}1) \times n$$

Note: $0! = 1$

```
public class SimpleFactorial{
    int n;

    public static void main(String[] args) {
       int facto = 1;
       n = Integer.parseInt(args[0]);
       if ((n == 0) || (n == 1) {
          System.out.println("Factorial of " + n + ": " + facto);
          return;
        }
        for(int i = 1; i <= n, i++)
            facto = facto * i;
        System.out.println("Factorial of " + n + ": " + facto);
        return;
   }
}
```

Recursion in Java : Calculation of *n*!

- Recursive definition of $n!$

$$n! = n \times (n\text{-}1) \times (n\text{-}2) \times \ldots 3 \times 2 \times 1$$

$$= n \times (n\text{-}1)! \qquad \text{with } 0! = 1$$

```
public class RecursiveFactorial{
  int n;
  int factorial(int n) {
      if (n == 0)
          return(1);
      else
          return(n*factorial(n-1));
   }

   public static void main(String[] args) {
       Recursive x = new RecursiveFactorial();
       x.n = Integer.parseInt(args[0]);
       System.out.println("Factorial of "+ n + ": " + x.factorial(x.n));
   }
}
```

Recursion in Java : Fibonacci sequence

- Following is a series of numbers called the Fibonacci sequence.

0 1 1 2 3 5 8 13 21

- Recursive definition of n-th Fibonacci number.

$$F_n = F_{n-1} + F_{n-2} \quad \text{with } F_0 = F_1 = 1$$

```
public class SimpleFibonacci{
   int n;

   public static void main(String[] args) {
       n = Integer.parseInt(args[0]);
       int fibo1 = 0, fibo2 = 1;
       System.out.print(fibo1 + " " + fibo2);
       while (n > 1) {
         fibo = fibo1 + fibo2;
         System.out.print(" " + fibo);
         fibo1 = fibo2; fibo2 = fibo; n++;
       }
   }
}
```

Recursion in Java : Fibonacci sequence

- Recursive definition of n-th Fibonacci number.

$$F_n = F_{n-1} + F_{n-2} \quad \text{with } F_0 = F_1 = 1$$

```
class RecursiveFibonacci {
    int n;
    int fibonacci(int n){
       if (n == 0)
           return 0;
       else if (n == 1)
              return 1;
           else
              return(fibonacci(n-1) + fibonacci(n-2));
     }
     public static void main(String args[]){
         Fibonacci x = new RecursiveFibonacci();
         x.n = Integer.parseInt(args[0]);
         for(int i = 0; i <= x.n; i++){
               System.out.println(x.fibonacci(i));
     }
   }
}
```

Recursion in Java : GCD calculation

Greatest common divisor (GCD) calculation is as follows.

GCD(35, 15) = 5
GCD(10, 50) = 10
GCD(11, 0) = 11
GCD(8, 8) = 8
GCD(1, 13) = 1

For any two integers *m* and *n* (such that $m < n$), the *GCD (m,n)* calculation can be defined recursively is as follows.

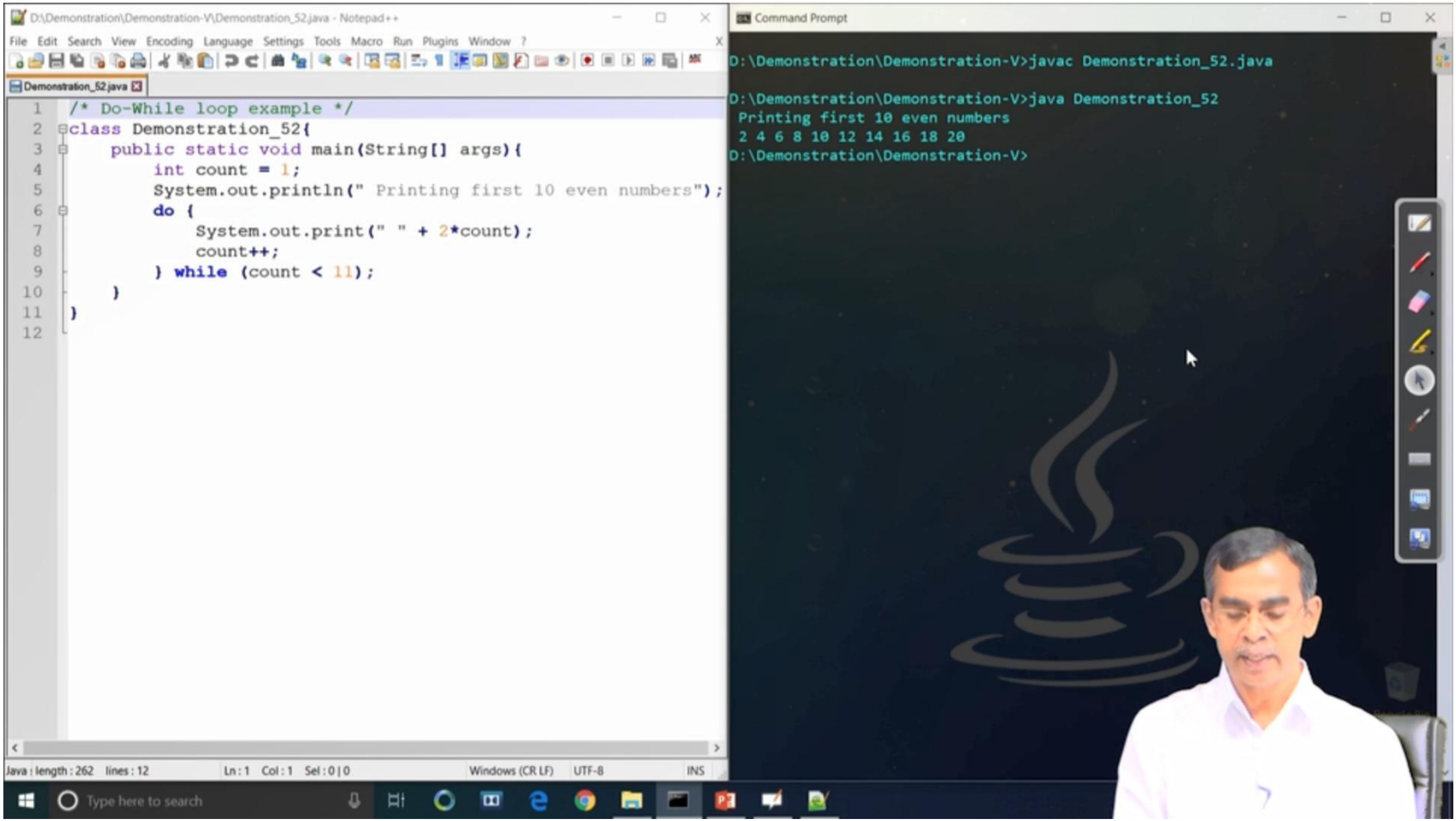
GCD(m, n) = GCD(n,m) if m>n;
GCD(m, n) = m, if m = 0;
GCD(m, n) = 1, if m = 1;
GCD(m, n) = m, if m = n;
GCD(m, n) = GCD(m, n%m);

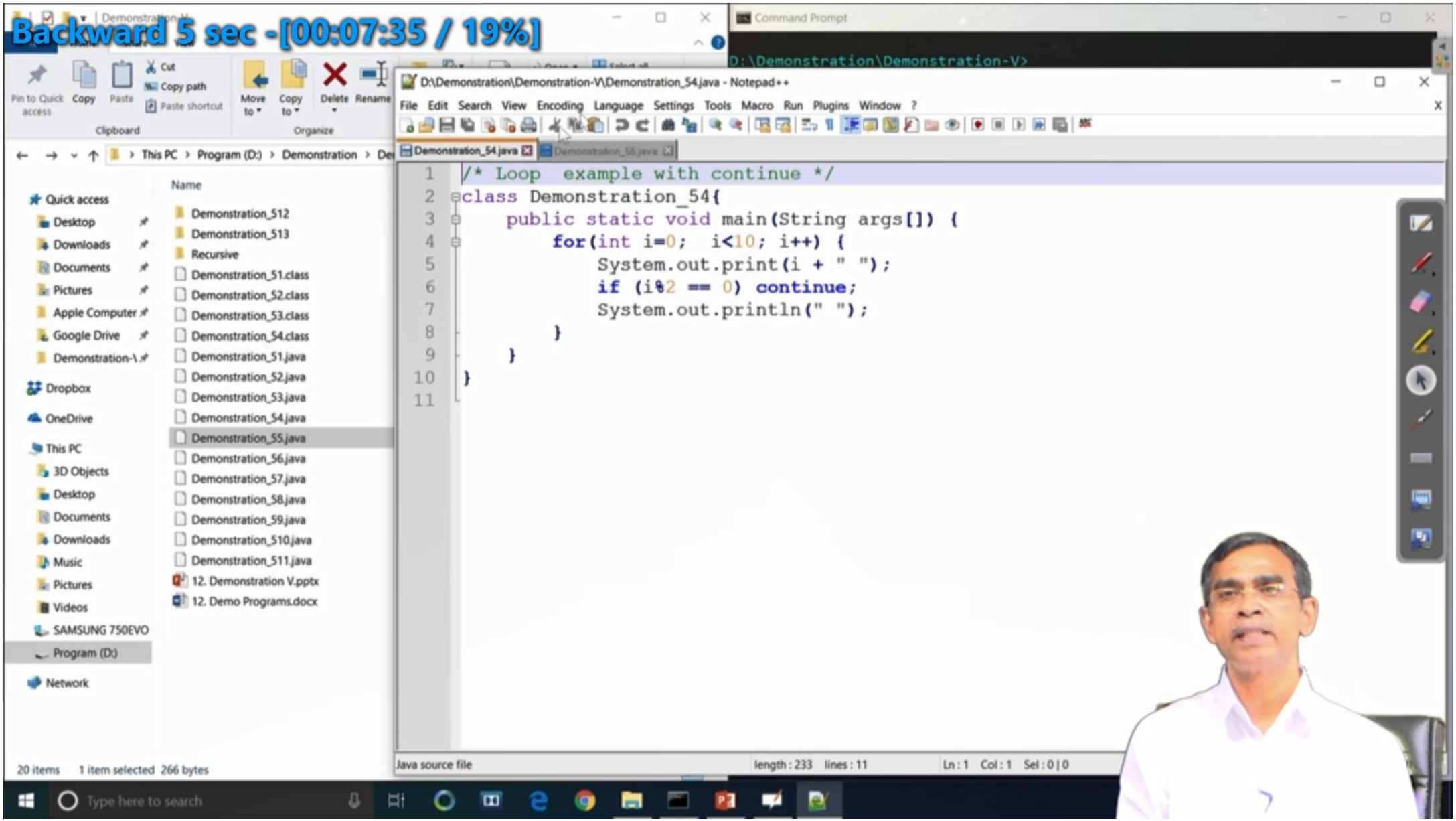
Recursion in Java : GCD calculation

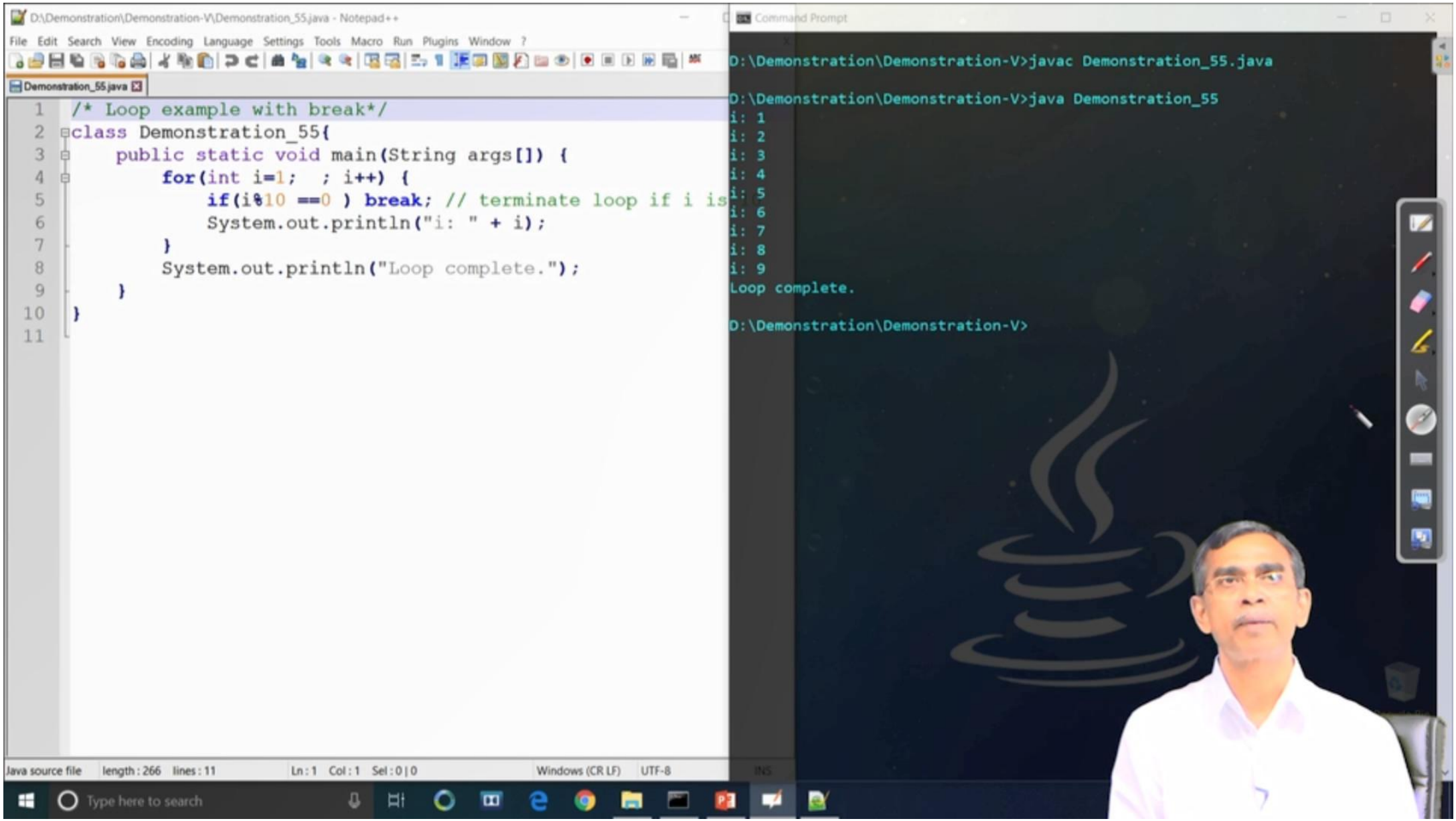
For any two integers *m* and *n* (such that $m < n$), the *GCD (m,n)* calculation can be defined recursively is as follows.

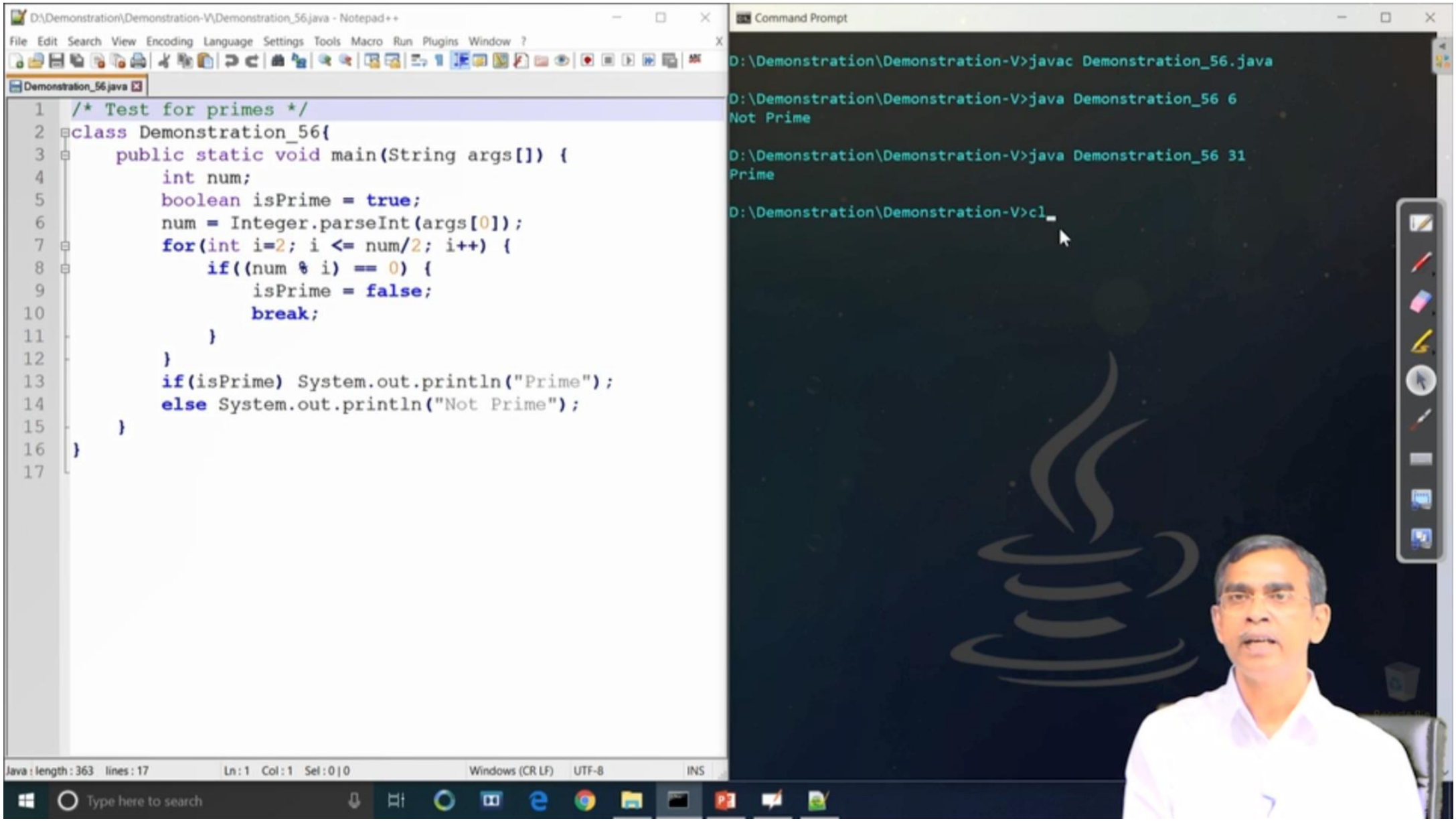
```
GCD(m, n) = GCD(n,m) if m>n;
GCD(m, n) = n, if m = 0;
GCD(m, n) = 1, if m = 1;
GCD(m, n) = m, If m = n;
GCD(m, n) = GCD(m, n%m);
```

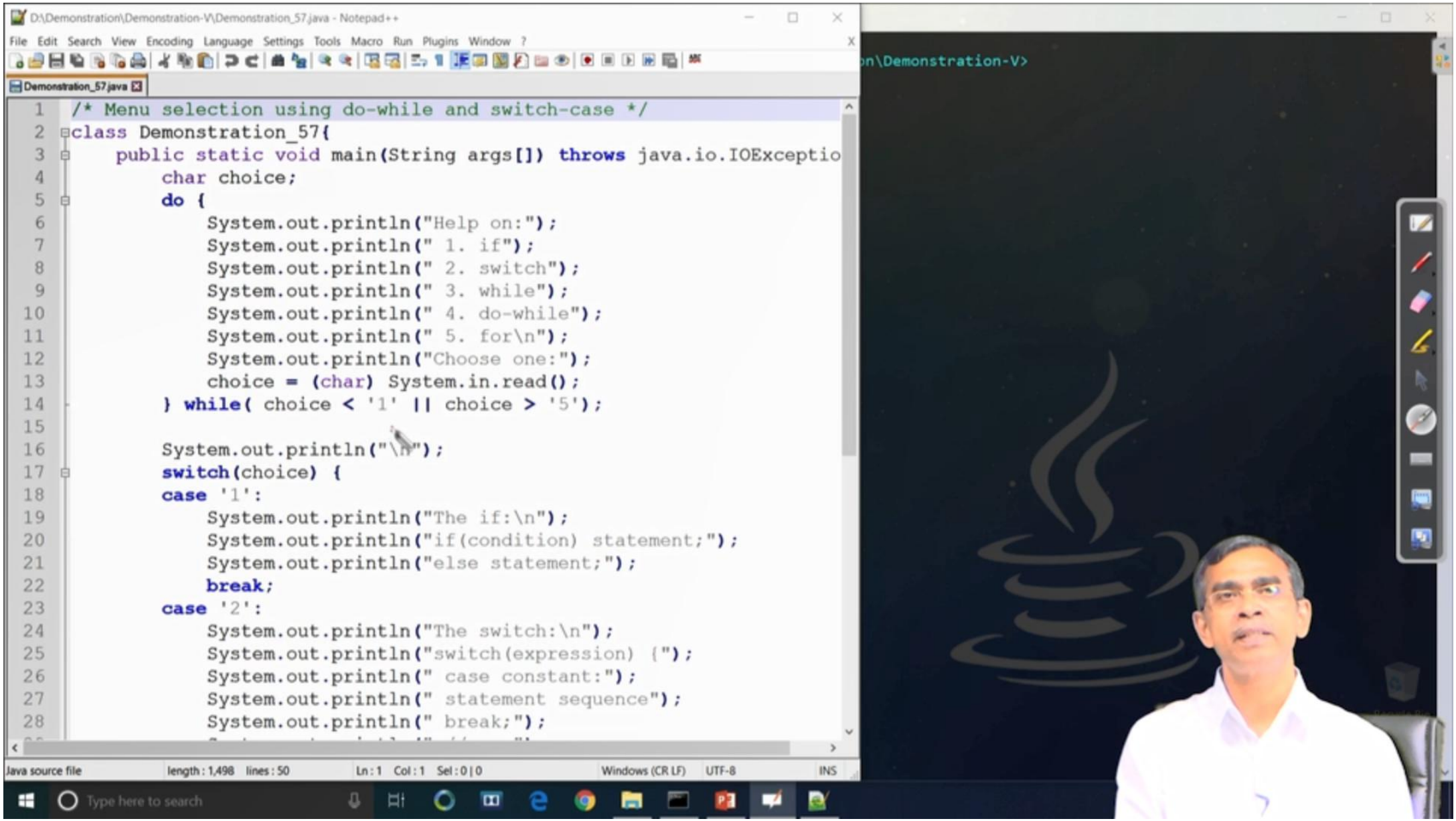
```java
public class RecursiveGCD {
    int m, n;
    int gcd(int m, int n){
        if(m>n) return gcd(n,m);
        if(m==n)return m;
        if(m==0)return n;
        if(m==1)return 1;
        return gcd(m,n%m);
    }

    public static void main(String[] args) {
        RecursiveGCD g = new RecursiveGCD();
        g.m = Integer.ParseInt(args[0]);
        g.n = Integer.ParseInt(args[1]);
        System.out.printf("GCD of %d and %d is %d.", g.m, g.n, g.gcd(g.m, g.n));
    }
}
```

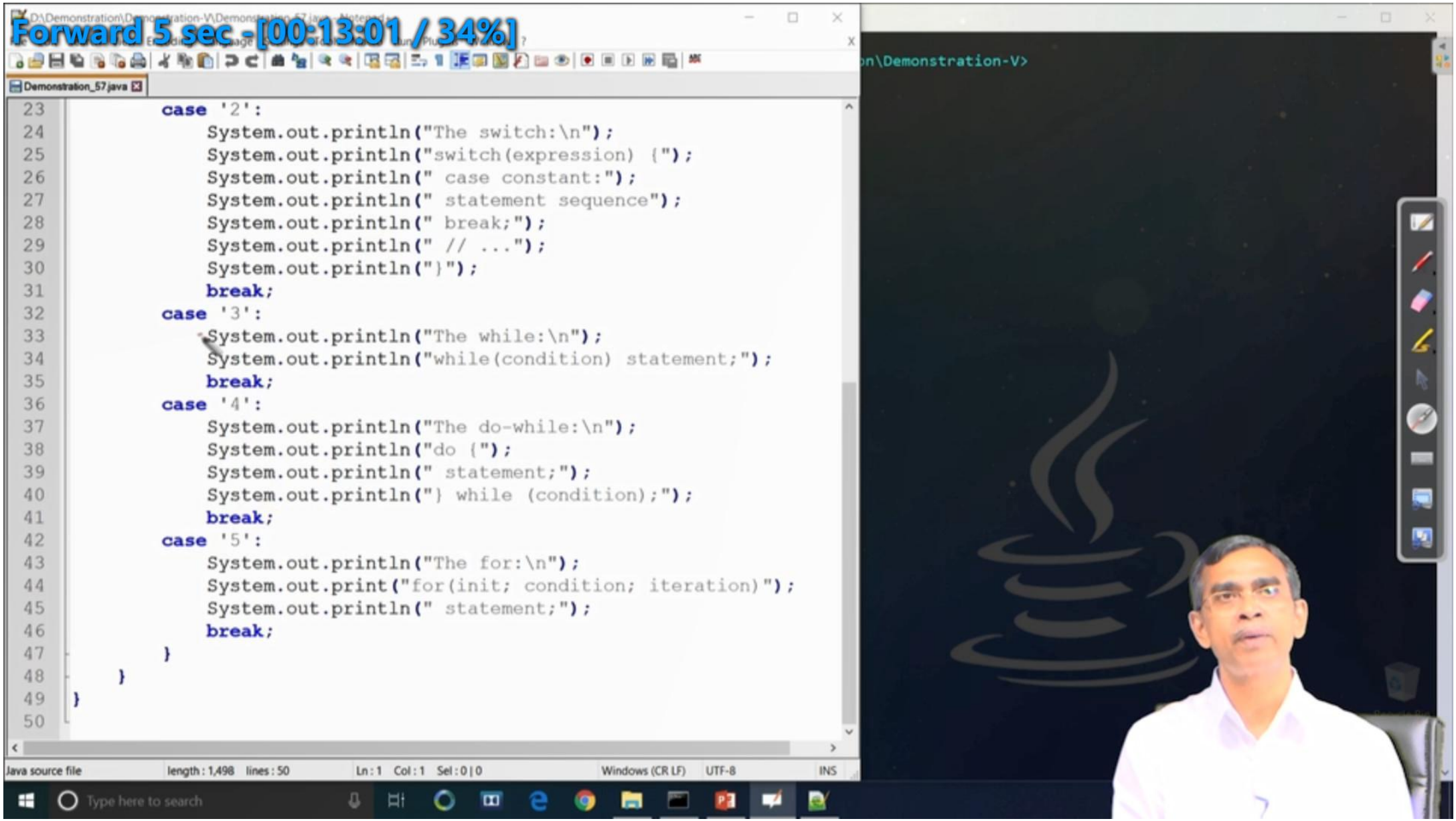

D:\Demonstration\Demonstration-V\Demonstration_52.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Demonstration_52.java
1 /* Do-While loop example */
2 class Demonstration_52{
3 public static void main(String[] args){
4 int count = 1;
5 System.out.println(" Printing first 10 even numbers");
6 do {
7 System.out.print(" " + 2*count);
8 count++;
9 } while (count < 11);
10 }
11 }
12
Java : length : 262 lines : 12 Ln : 1 Col : 1 Sel : 0 | 0 Windows (CR LF) UTF-8 INS
Command Prompt
D:\Demonstration\Demonstration-V>javac Demonstration_52.java
D:\Demonstration\Demonstration-V>java Demonstration_52
Printing first 10 even numbers
2 4 6 8 10 12 14 16 18 20
D:\Demonstration\Demonstration-V>
Type here to search

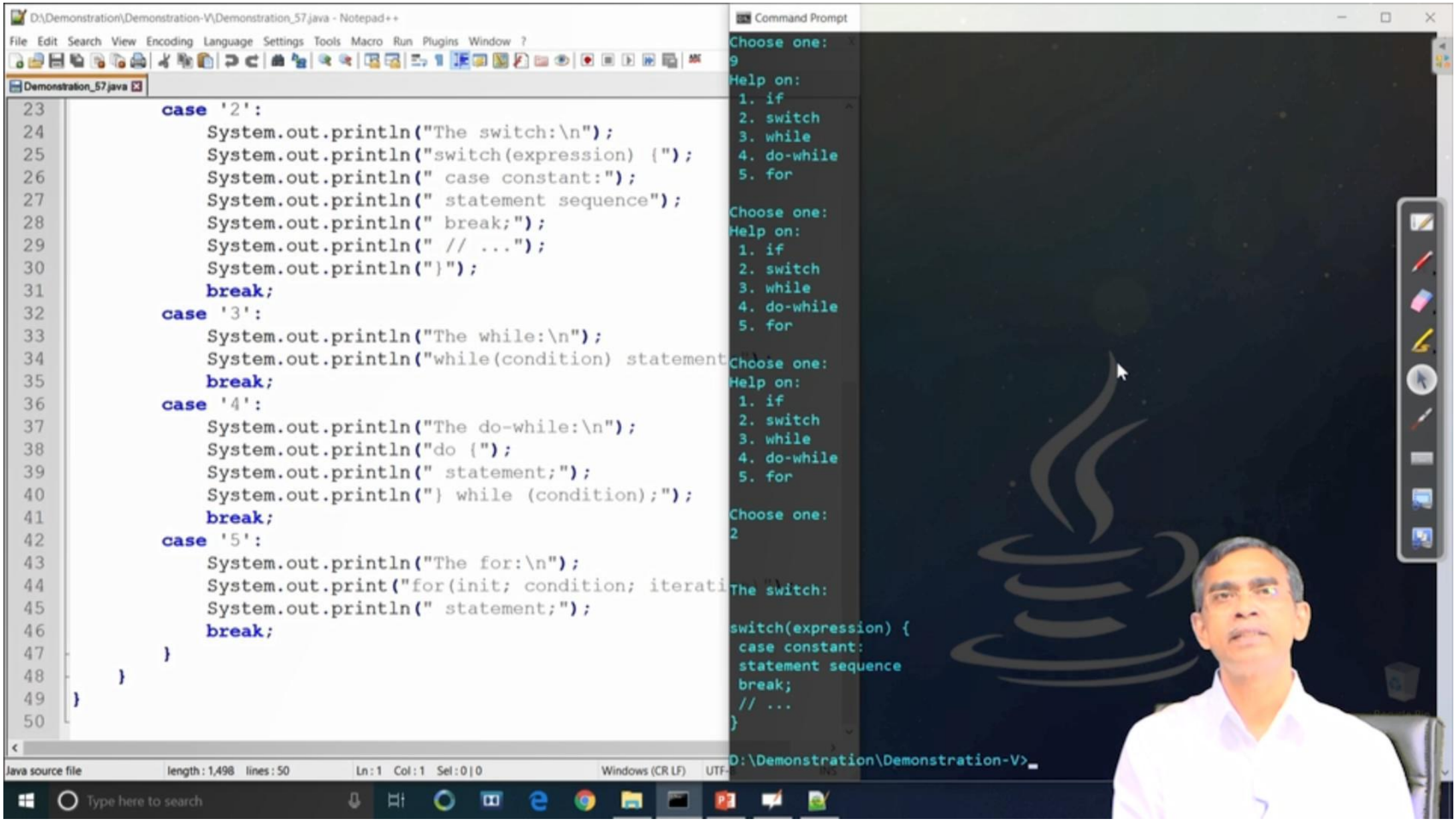

Backward 5 sec -[00:07:35 / 19%]
Command Prompt
D:\Demonstration\Demonstration-V>
D:\Demonstration\Demonstration-V\Demonstration_54.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Demonstration_54.java
Demonstration_55.java
/* Loop example with continue */
class Demonstration_54{
public static void main(String args[]) {
for(int i=0; i<10; i++) {
System.out.print(i + " ");
if (i%2 == 0) continue;
System.out.println(" ");
}
}
}
Java source file
length : 233 lines : 11
Ln : 1 Col : 1 Sel : 0 | 0
Name
Demonstration_512
Demonstration_513
Recursive
Demonstration_51.class
Demonstration_52.class
Demonstration_53.class
Demonstration_54.class
Demonstration_51.java
Demonstration_52.java
Demonstration_53.java
Demonstration_54.java
Demonstration_55.java
Demonstration_56.java
Demonstration_57.java
Demonstration_58.java
Demonstration_59.java
Demonstration_510.java
Demonstration_511.java
12. Demonstration V.pptx
12. Demo Programs.docx
20 items 1 item selected 266 bytes
Type here to search

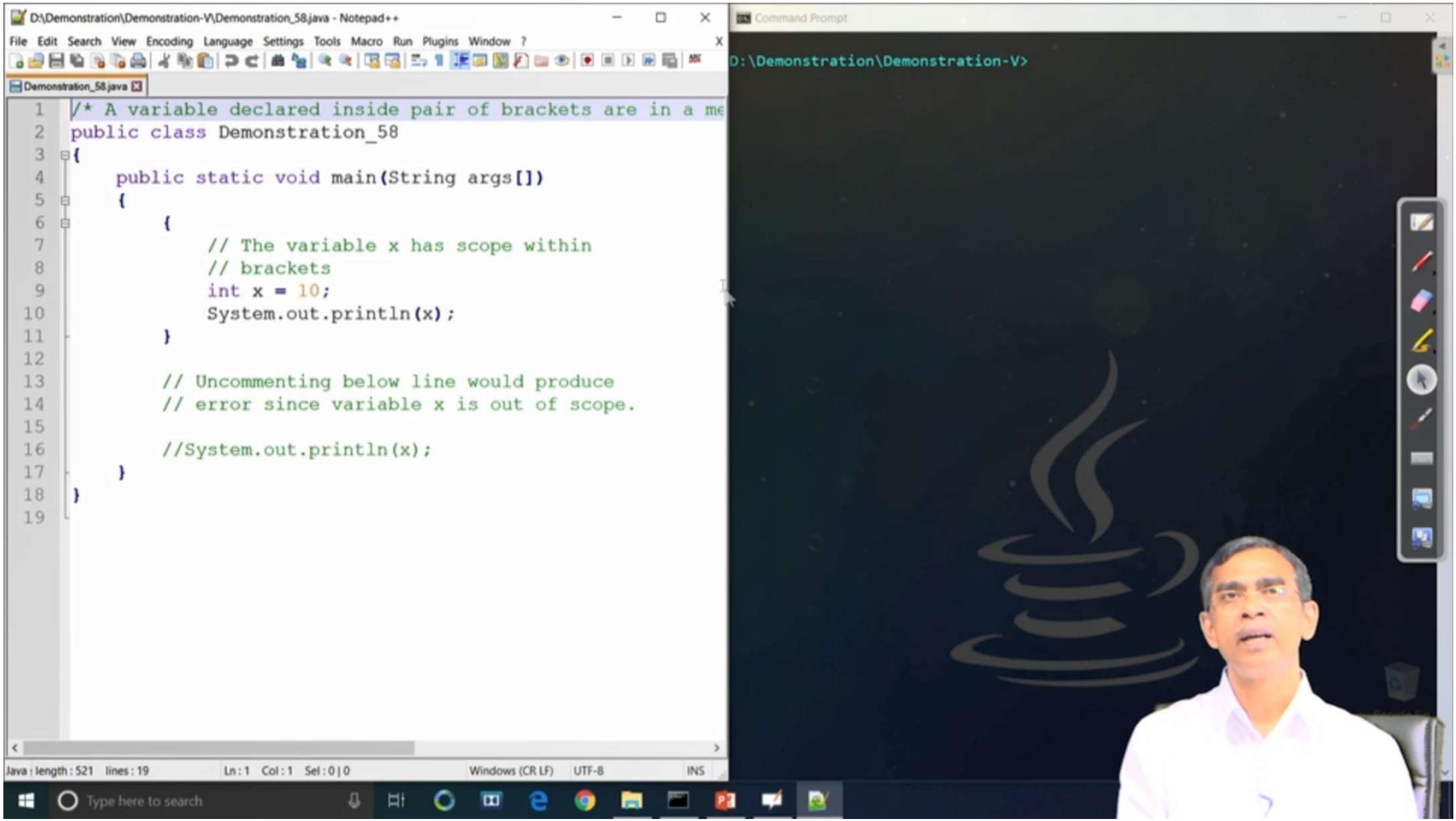

D:\Demonstration\Demonstration-V\Demonstration_55.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Demonstration_55.java
1 /* Loop example with break*/
2 class Demonstration_55{
3 public static void main(String args[]) {
4 for(int i=1; ; i++) {
5 if(i%10 ==0) break; // terminate loop if i is
6 System.out.println("i: " + i);
7 }
8 System.out.println("Loop complete.");
9 }
10 }
11
Java source file length : 266 lines : 11 Ln : 1 Col : 1 Sel : 0 | 0 Windows (CR LF) UTF-8 INS
Command Prompt
D:\Demonstration\Demonstration-V>javac Demonstration_55.java
D:\Demonstration\Demonstration-V>java Demonstration_55
i: 1
i: 2
i: 3
i: 4
i: 5
i: 6
i: 7
i: 8
i: 9
Loop complete.
D:\Demonstration\Demonstration-V>
Type here to search

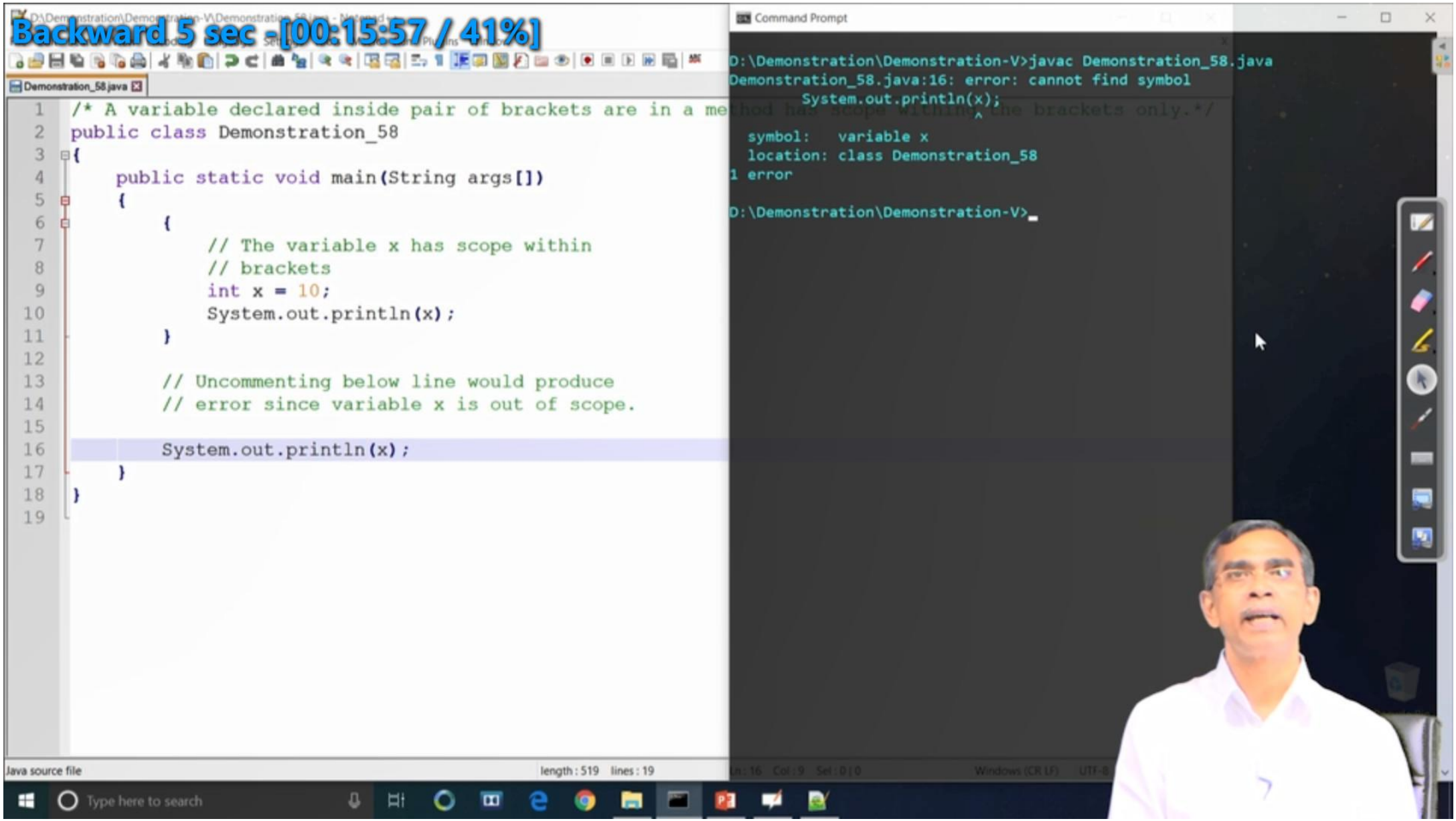

D:\Demonstration\Demonstration-V\Demonstration_56.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Demonstration_56.java
/* Test for primes */
class Demonstration_56{
public static void main(String args[]) {
int num;
boolean isPrime = true;
num = Integer.parseInt(args[0]);
for(int i=2; i <= num/2; i++) {
if((num % i) == 0) {
isPrime = false;
break;
}
}
if(isPrime) System.out.println("Prime");
else System.out.println("Not Prime");
}
}
Java : length : 363 lines : 17 Ln : 1 Col : 1 Sel : 0 | 0 Windows (CR LF) UTF-8 INS
Command Prompt
D:\Demonstration\Demonstration-V>javac Demonstration_56.java
D:\Demonstration\Demonstration-V>java Demonstration_56 6
Not Prime
D:\Demonstration\Demonstration-V>java Demonstration_56 31
Prime
D:\Demonstration\Demonstration-V>cl
Type here to search

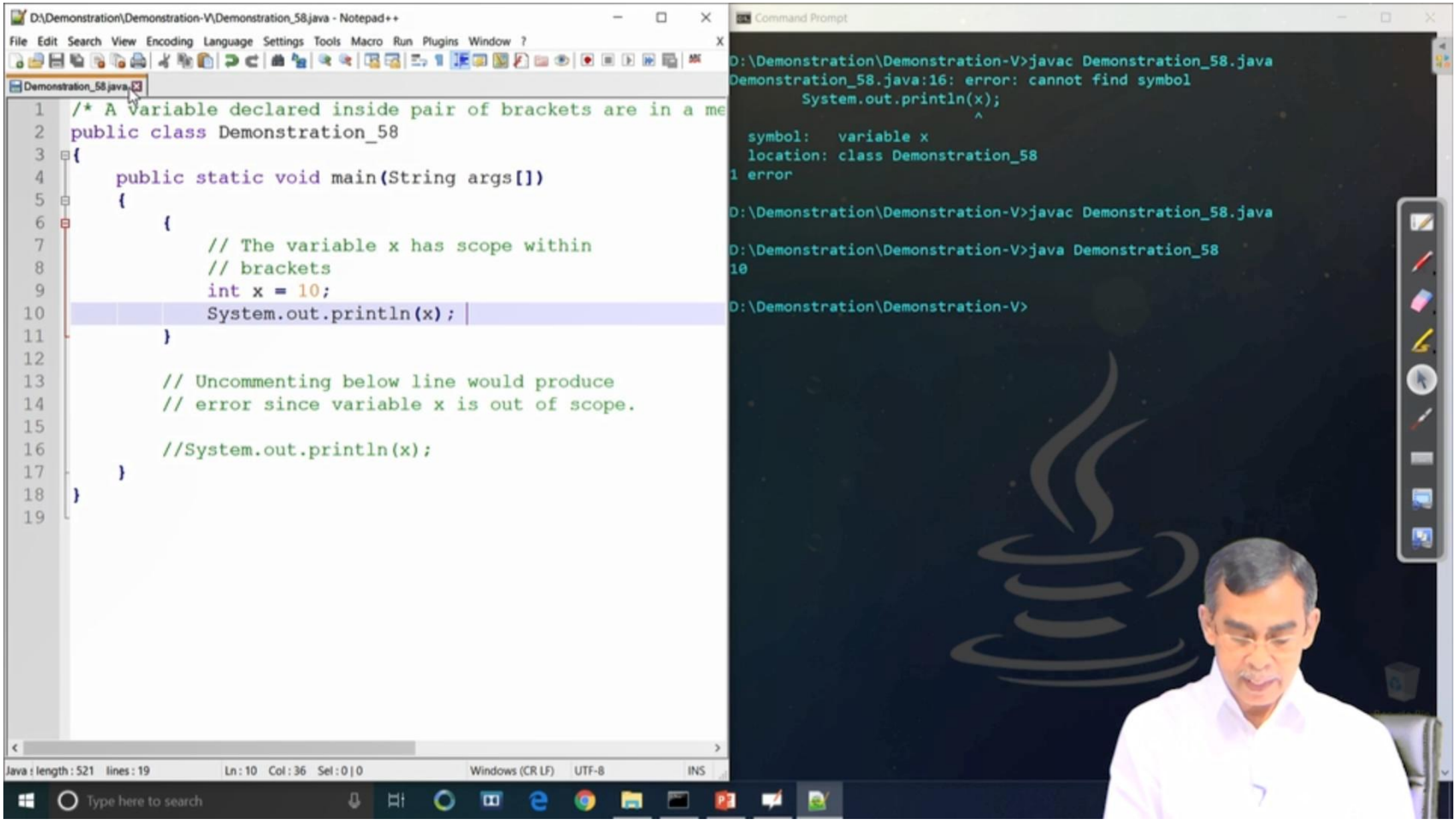

D:\Demonstration\Demonstration-V\Demonstration_57.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Demonstration_57.java
1 /* Menu selection using do-while and switch-case */
2 class Demonstration_57{
3 public static void main(String args[]) throws java.io.IOExceptio
4 char choice;
5 do {
6 System.out.println("Help on:");
7 System.out.println(" 1. if");
8 System.out.println(" 2. switch");
9 System.out.println(" 3. while");
10 System.out.println(" 4. do-while");
11 System.out.println(" 5. for\n");
12 System.out.println("Choose one:");
13 choice = (char) System.in.read();
14 } while(choice < '1' || choice > '5');
15
16 System.out.println("\n");
17 switch(choice) {
18 case '1':
19 System.out.println("The if:\n");
20 System.out.println("if(condition) statement;");
21 System.out.println("else statement;");
22 break;
23 case '2':
24 System.out.println("The switch:\n");
25 System.out.println("switch(expression) {");
26 System.out.println(" case constant:");
27 System.out.println(" statement sequence");
28 System.out.println(" break;");
Java source file length : 1,498 lines : 50 Ln : 1 Col : 1 Sel : 0 | 0 Windows (CR LF) UTF-8 INS
Type here to search
\Demonstration-V>

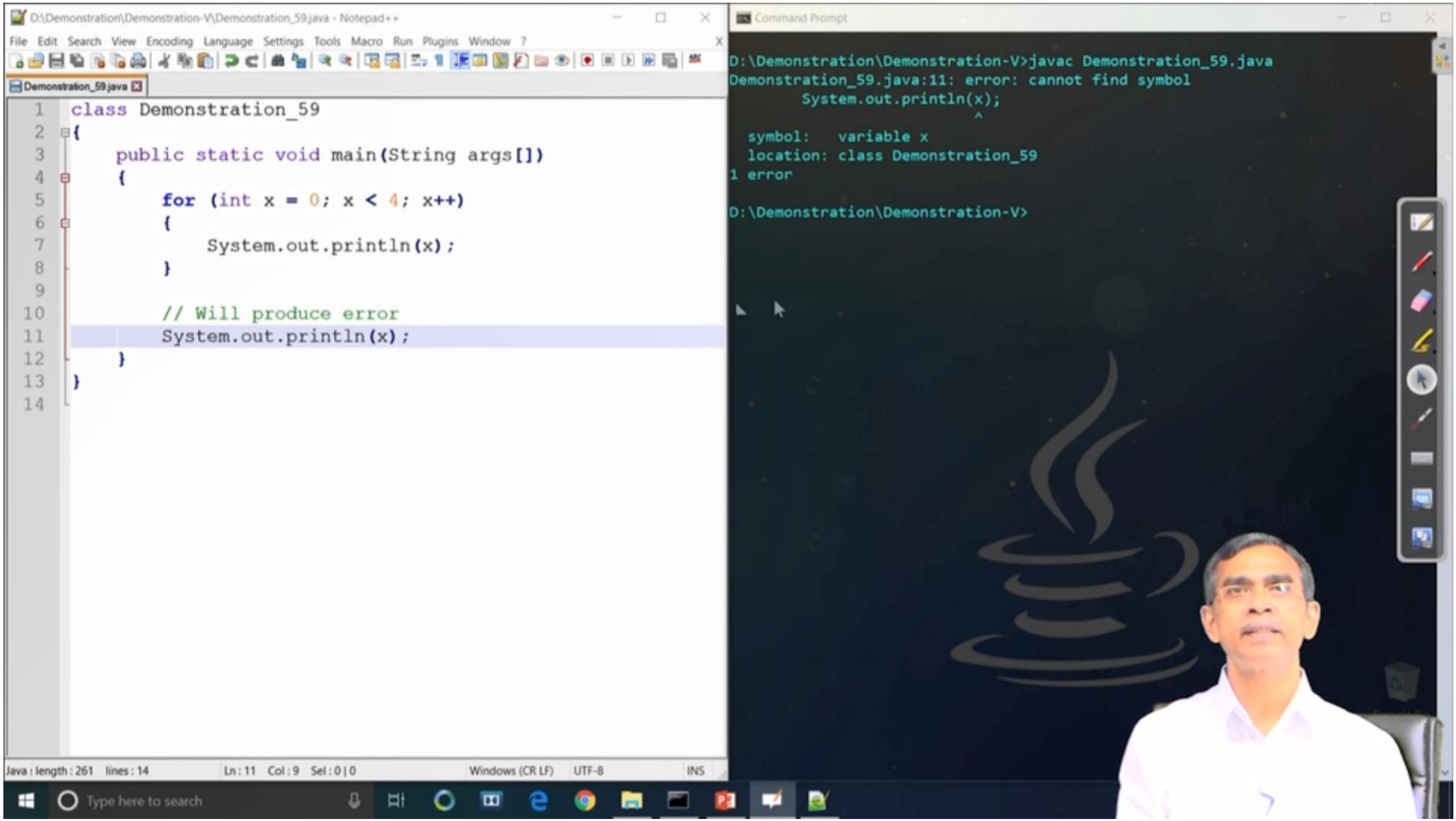

Forward 5 sec - [00:13:01 / 34%]
Demonstration_57.java
23 case '2':
24 System.out.println("The switch:\n");
25 System.out.println("switch(expression) {");
26 System.out.println(" case constant:");
27 System.out.println(" statement sequence");
28 System.out.println(" break;");
29 System.out.println(" // ...");
30 System.out.println("}");
31 break;
32 case '3':
33 System.out.println("The while:\n");
34 System.out.println("while(condition) statement;");
35 break;
36 case '4':
37 System.out.println("The do-while:\n");
38 System.out.println("do {");
39 System.out.println(" statement;");
40 System.out.println("} while (condition);");
41 break;
42 case '5':
43 System.out.println("The for:\n");
44 System.out.print("for(init; condition; iteration)");
45 System.out.println(" statement;");
46 break;
47 }
48 }
49 }
50
Java source file
length : 1,498 lines : 50
Ln : 1 Col : 1 Sel : 0 | 0
Windows (CR LF)
UTF-8
INS
Type here to search
on\Demonstration-V>

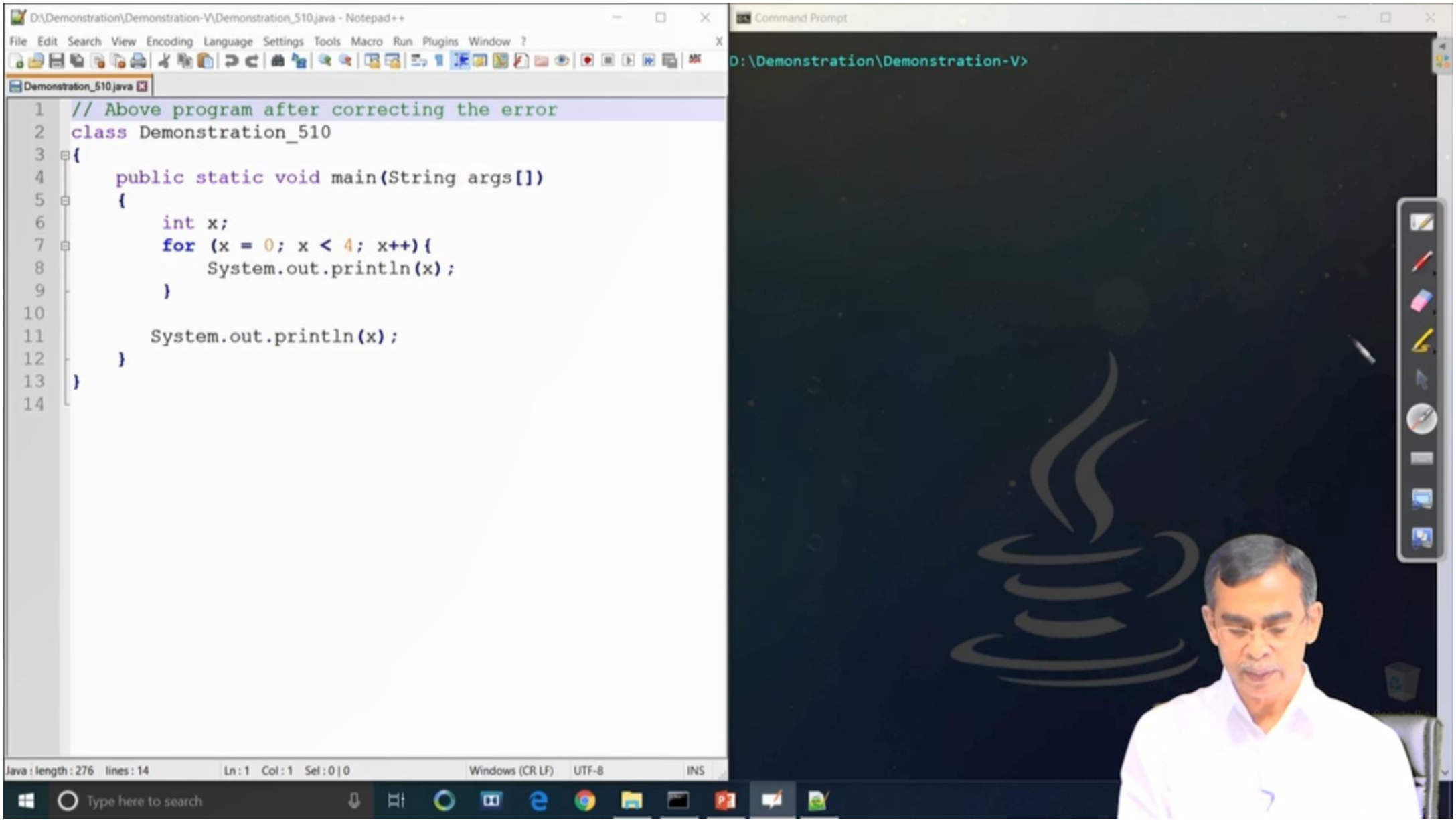

D:\Demonstration\Demonstration-V\Demonstration_57.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Demonstration_57.java
23 case '2':
24 System.out.println("The switch:\n");
25 System.out.println("switch(expression) {");
26 System.out.println(" case constant:");
27 System.out.println(" statement sequence");
28 System.out.println(" break;");
29 System.out.println(" // ...");
30 System.out.println("}");
31 break;
32 case '3':
33 System.out.println("The while:\n");
34 System.out.println("while(condition) statement
35 break;
36 case '4':
37 System.out.println("The do-while:\n");
38 System.out.println("do {");
39 System.out.println(" statement;");
40 System.out.println("} while (condition);");
41 break;
42 case '5':
43 System.out.println("The for:\n");
44 System.out.print("for(init; condition; iterati
45 System.out.println(" statement;");
46 break;
47 }
48 }
49 }
50
Java source file length : 1,498 lines : 50 Ln : 1 Col : 1 Sel : 0 | 0 Windows (CR LF)
Command Prompt
Choose one:
9
Help on:
1. if
2. switch
3. while
4. do-while
5. for
Choose one:
Help on:
1. if
2. switch
3. while
4. do-while
5. for
Choose one:
Help on:
1. if
2. switch
3. while
4. do-while
5. for
Choose one:
2
The switch:
switch(expression) {
case constant:
statement sequence
break;
// ...
}
D:\Demonstration\Demonstration-V>
Type here to search

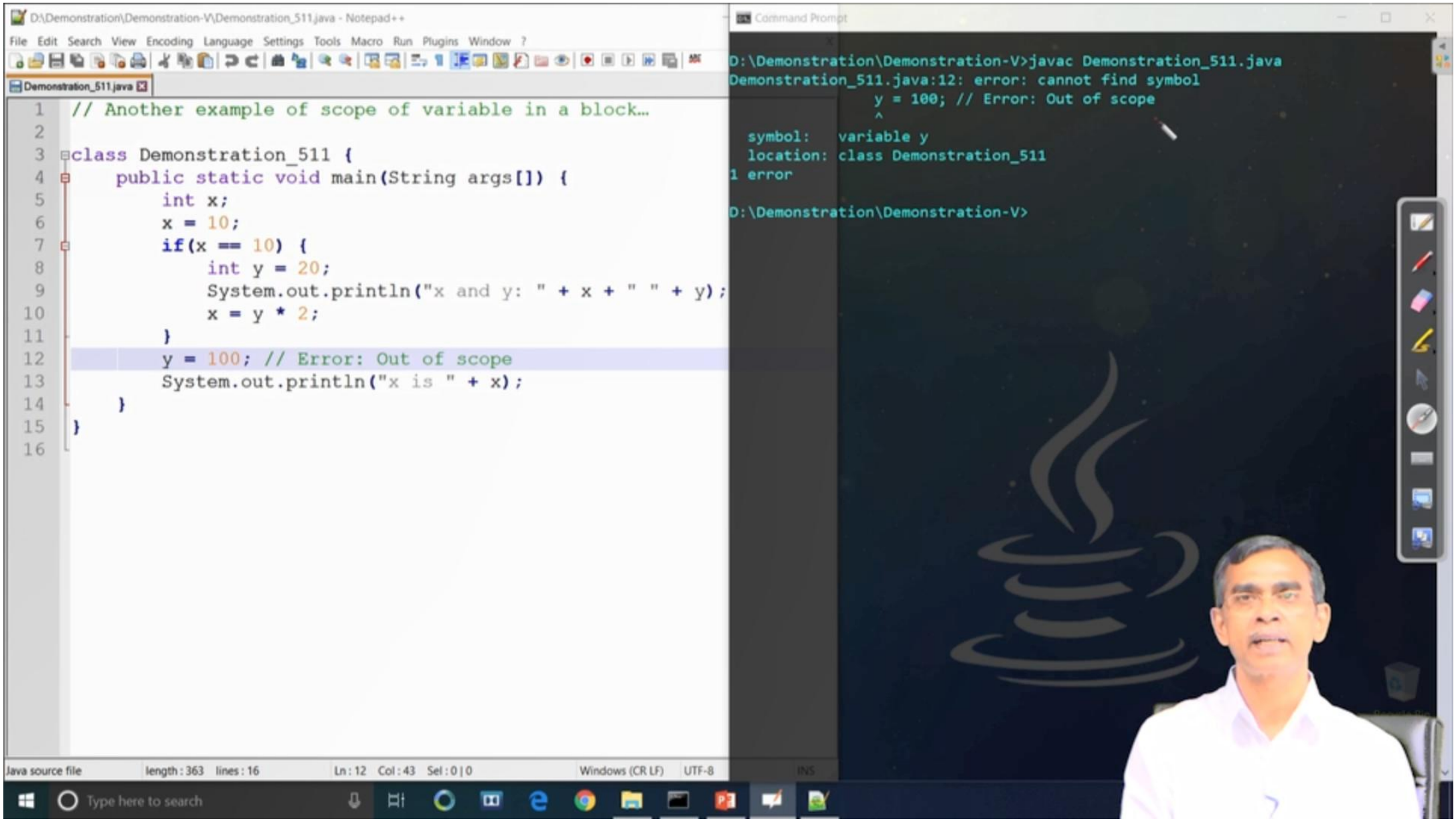

D:\Demonstration\Demonstration-V\Demonstration_58.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Demonstration_58.java
/* A variable declared inside pair of brackets are in a me
public class Demonstration_58
{
public static void main(String args[])
{
{
// The variable x has scope within
// brackets
int x = 10;
System.out.println(x);
}
// Uncommenting below line would produce
// error since variable x is out of scope.
//System.out.println(x);
}
}
Java : length : 521 lines : 19 Ln : 1 Col : 1 Sel : 0 | 0 Windows (CR LF) UTF-8 INS
Command Prompt
D:\Demonstration\Demonstration-V>
Type here to search

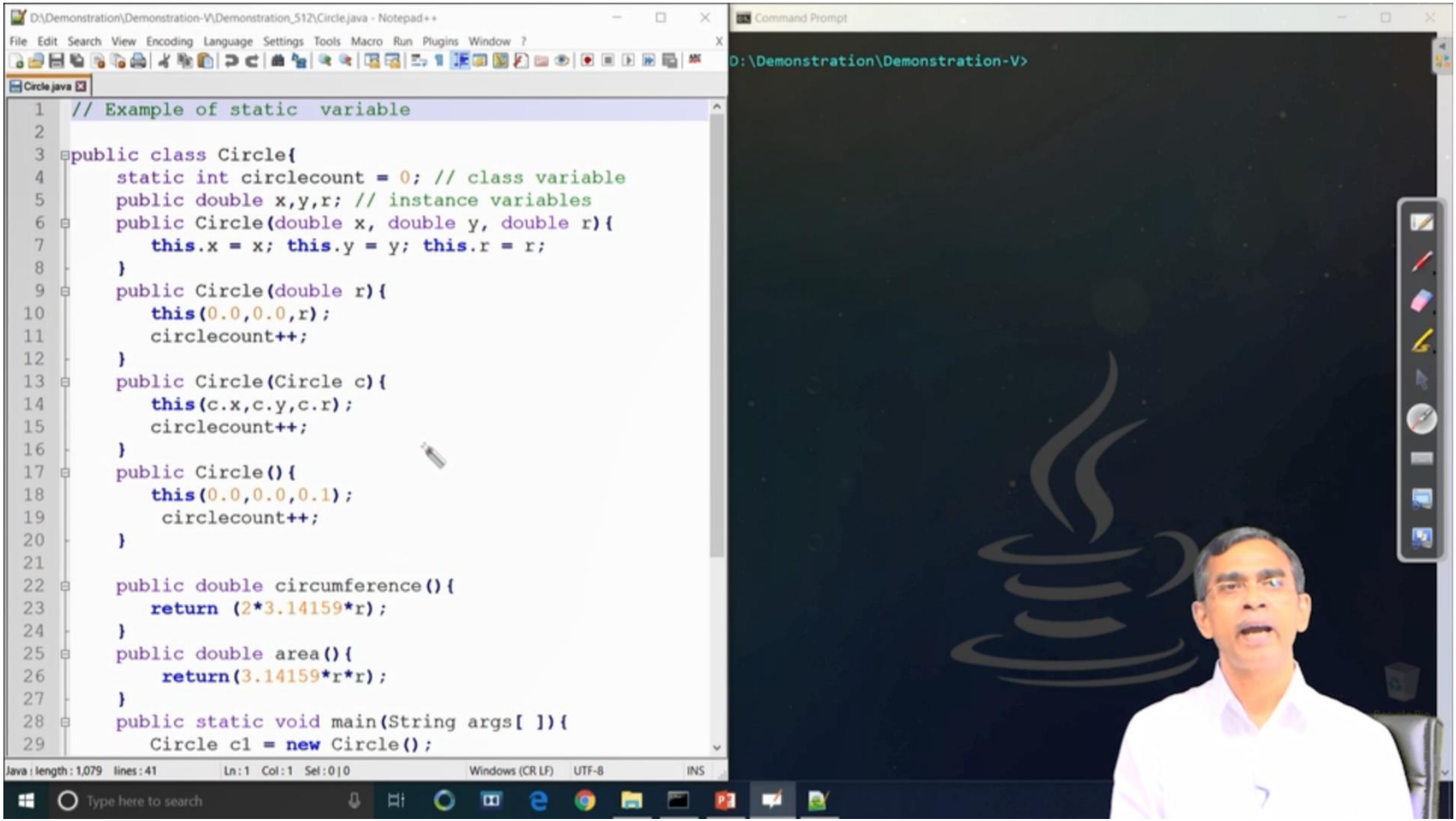

Backward 5 sec - [00:15:57 / 41%]
Demonstration_58.java
/* A variable declared inside pair of brackets are in a me
public class Demonstration_58
{
public static void main(String args[])
{
{
// The variable x has scope within
// brackets
int x = 10;
System.out.println(x);
}
// Uncommenting below line would produce
// error since variable x is out of scope.
System.out.println(x);
}
}
Java source file
length : 519 lines : 19
Command Prompt
D:\Demonstration\Demonstration-V>javac Demonstration_58.java
Demonstration_58.java:16: error: cannot find symbol
System.out.println(x);
^
symbol: variable x
location: class Demonstration_58
1 error
D:\Demonstration\Demonstration-V>
Type here to search

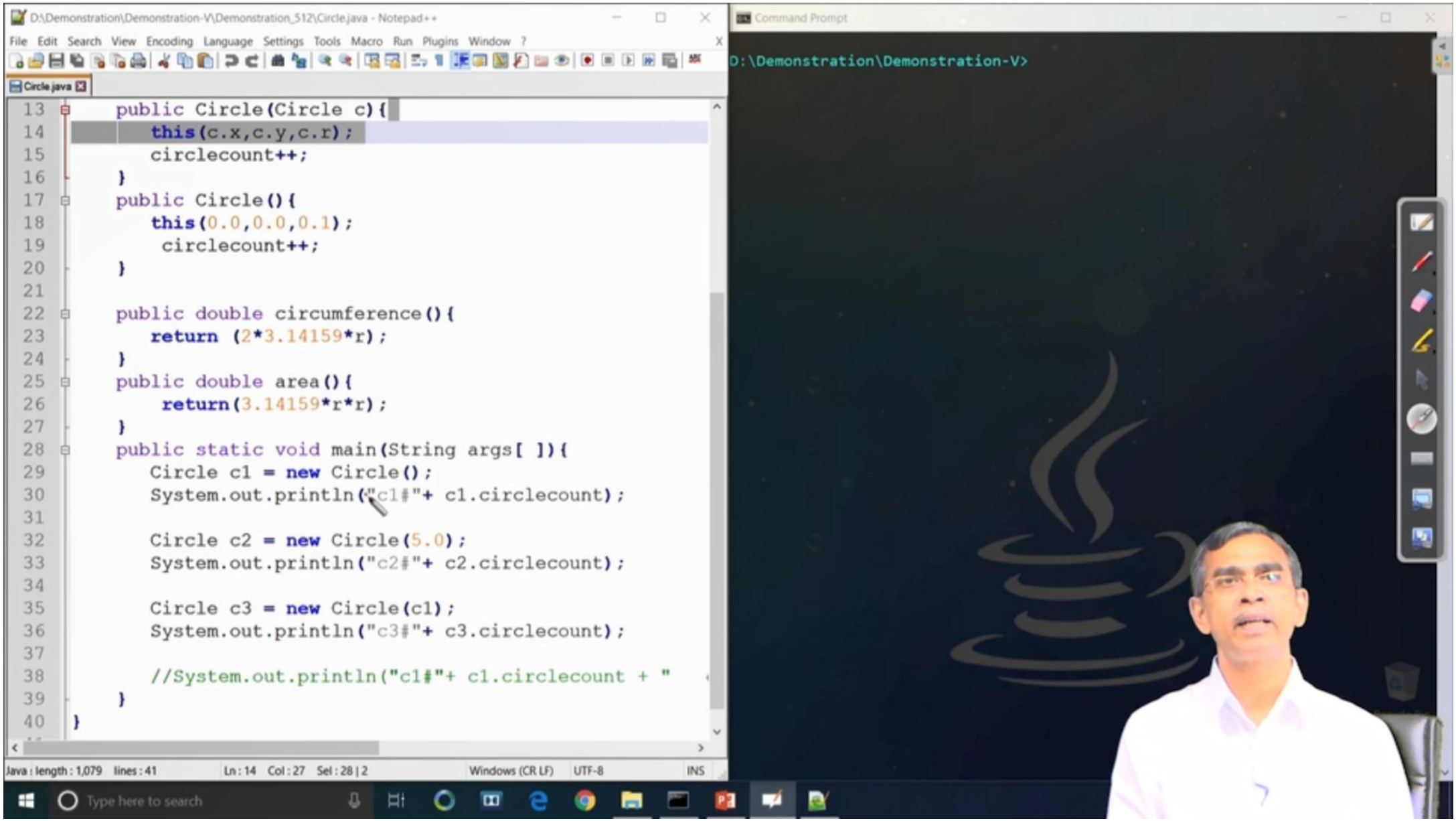

D:\Demonstration\Demonstration-V\Demonstration_58.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Demonstration_58.java
/* A variable declared inside pair of brackets are in a me
public class Demonstration_58
{
public static void main(String args[])
{
{
// The variable x has scope within
// brackets
int x = 10;
System.out.println(x);
}
// Uncommenting below line would produce
// error since variable x is out of scope.
//System.out.println(x);
}
}
Java : length : 521 lines : 19 Ln : 10 Col : 36 Sel : 0 | 0 Windows (CR LF) UTF-8 INS
Command Prompt
D:\Demonstration\Demonstration-V>javac Demonstration_58.java
Demonstration_58.java:16: error: cannot find symbol
System.out.println(x);
^
symbol: variable x
location: class Demonstration_58
1 error
D:\Demonstration\Demonstration-V>javac Demonstration_58.java
D:\Demonstration\Demonstration-V>java Demonstration_58
10
D:\Demonstration\Demonstration-V>
Type here to search

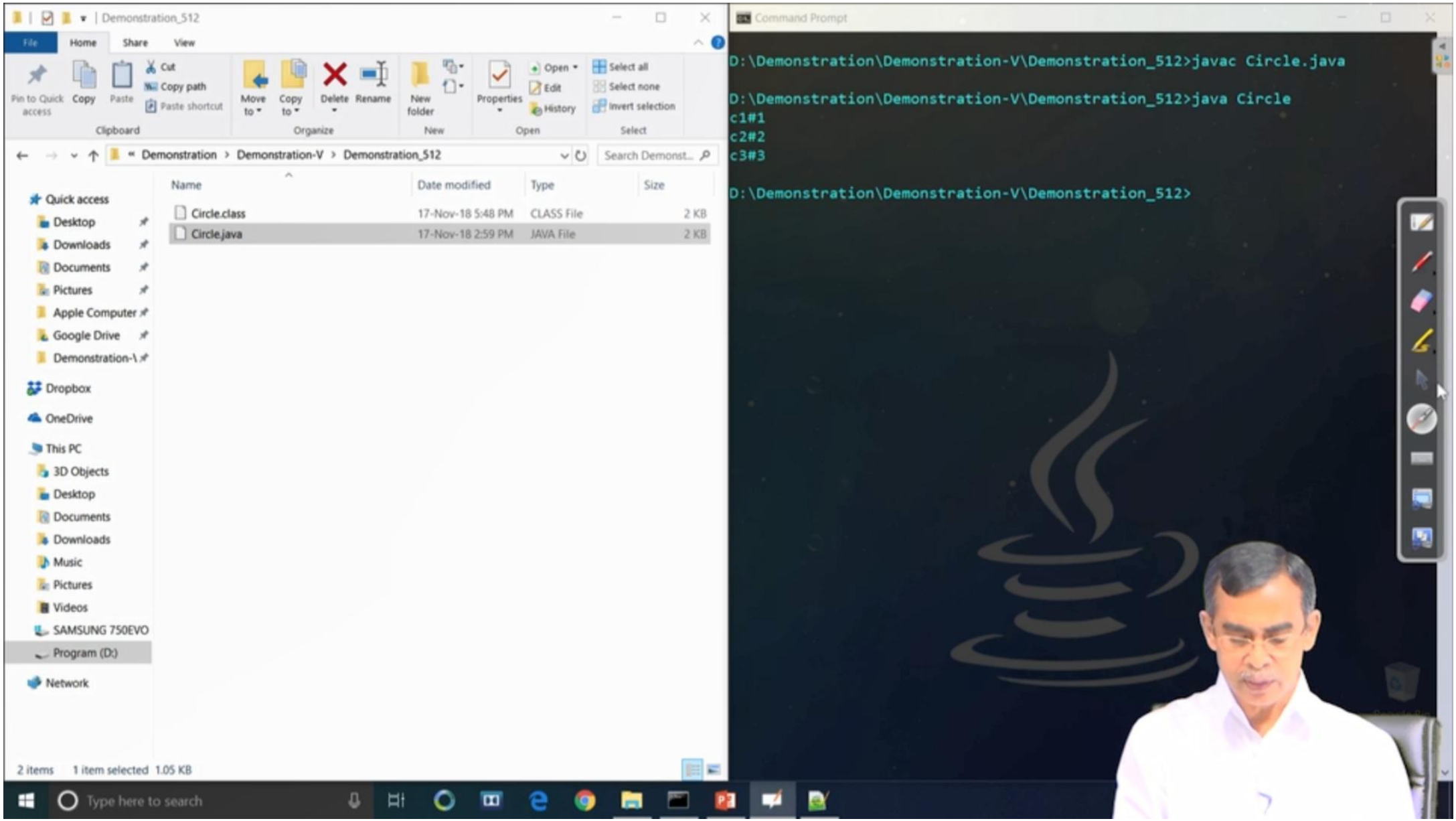

D:\Demonstration\Demonstration-V\Demonstration_59.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Demonstration_59.java
1 class Demonstration_59
2 {
3 public static void main(String args[])
4 {
5 for (int x = 0; x < 4; x++)
6 {
7 System.out.println(x);
8 }
9
10 // Will produce error
11 System.out.println(x);
12 }
13 }
14
Java : length : 261 lines : 14 Ln : 11 Col : 9 Sel : 0 | 0 Windows (CR LF) UTF-8 INS
Command Prompt
D:\Demonstration\Demonstration-V>javac Demonstration_59.java
Demonstration_59.java:11: error: cannot find symbol
System.out.println(x);
^
symbol: variable x
location: class Demonstration_59
1 error
D:\Demonstration\Demonstration-V>
Type here to search

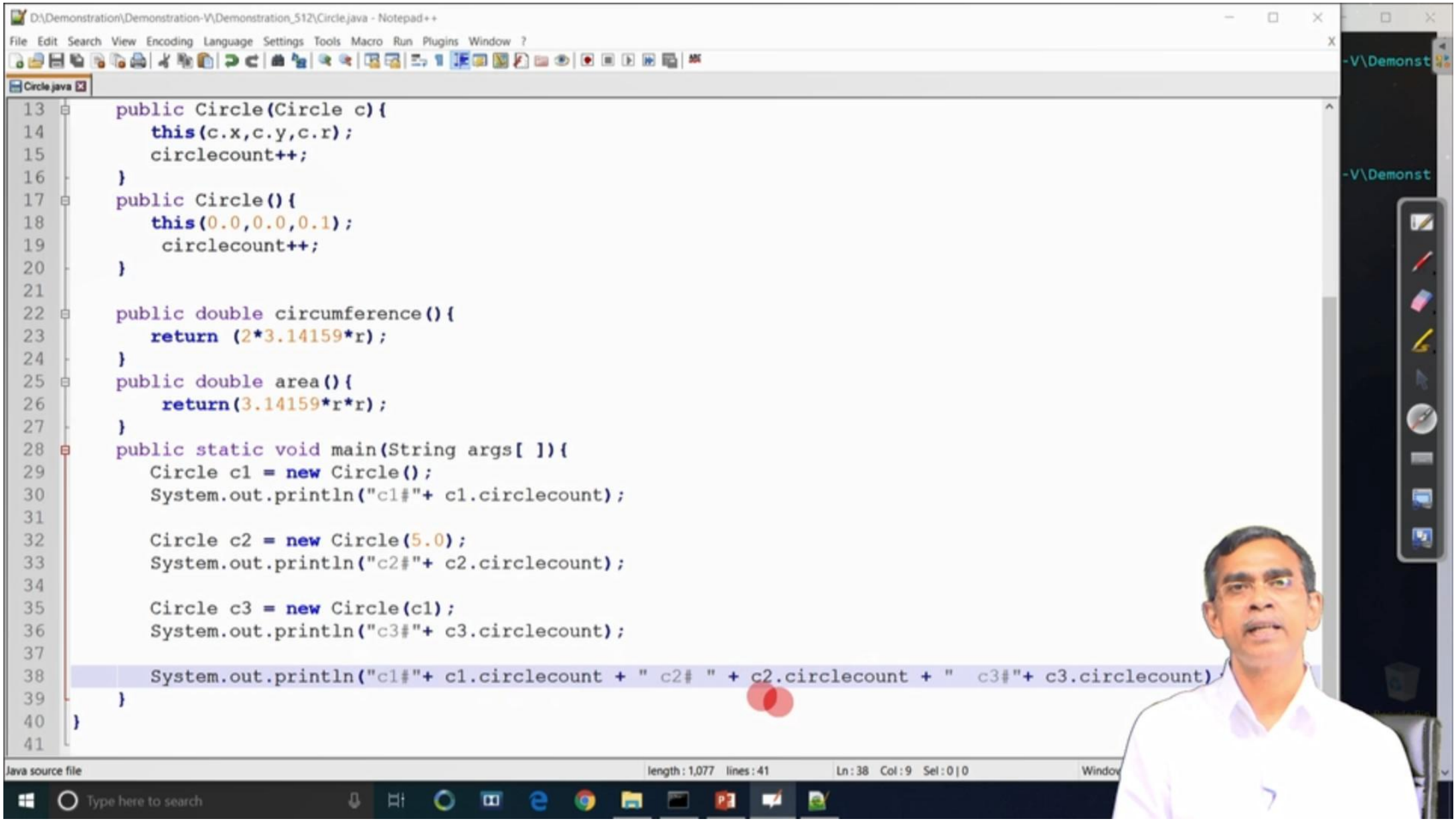

D:\Demonstration\Demonstration-V\Demonstration_510.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Demonstration_510.java
// Above program after correcting the error
class Demonstration_510
{
public static void main(String args[])
{
int x;
for (x = 0; x < 4; x++){
System.out.println(x);
}
System.out.println(x);
}
}
Java : length : 276 lines : 14
Ln : 1 Col : 1 Sel : 0 | 0
Windows (CR LF)
UTF-8
INS
Command Prompt
D:\Demonstration\Demonstration-V>
Type here to search

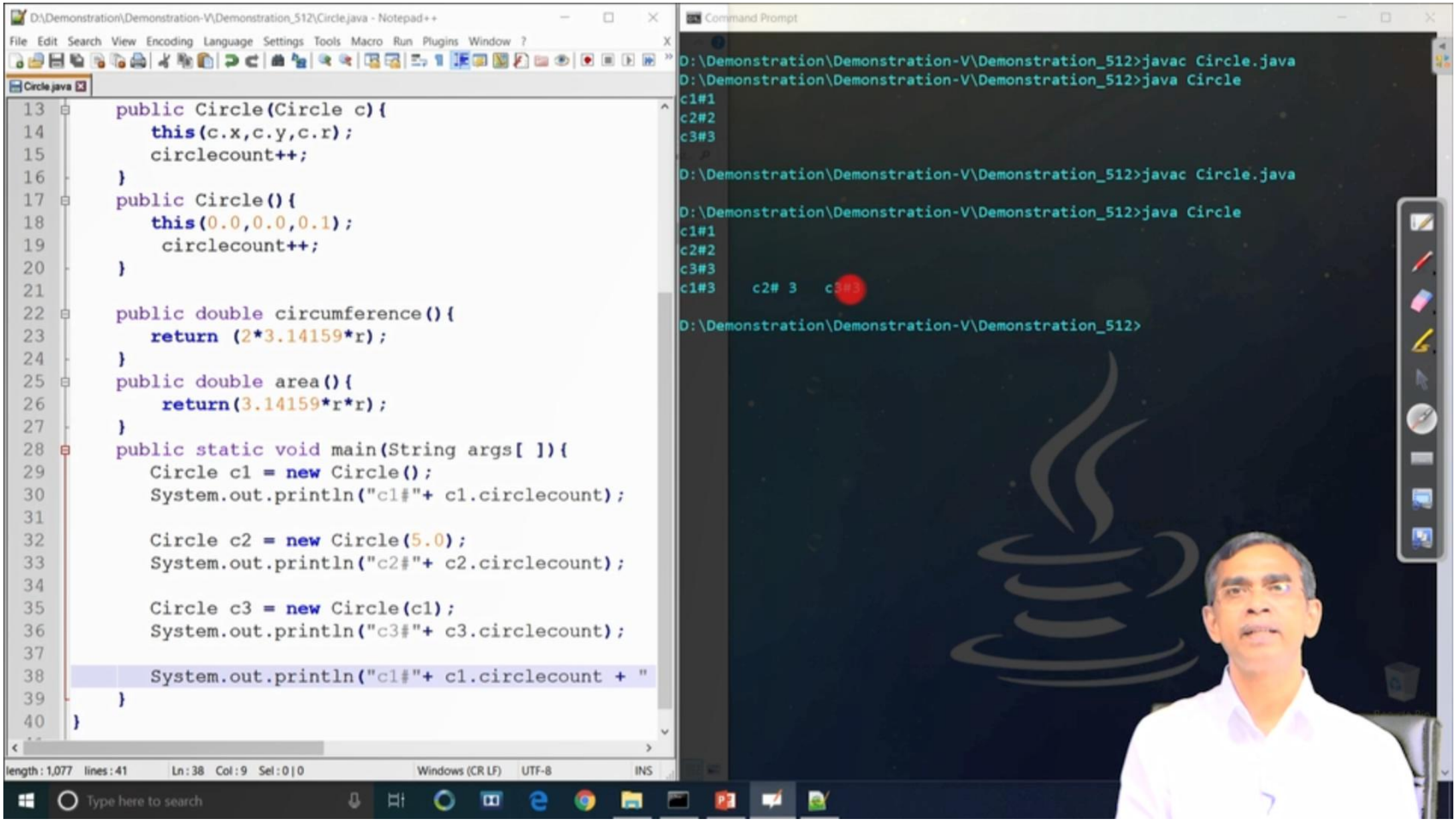

D:\Demonstration\Demonstration-V\Demonstration_511.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Demonstration_511.java
1 // Another example of scope of variable in a block…
2
3 class Demonstration_511 {
4 public static void main(String args[]) {
5 int x;
6 x = 10;
7 if(x == 10) {
8 int y = 20;
9 System.out.println("x and y: " + x + " " + y);
10 x = y * 2;
11 }
12 y = 100; // Error: Out of scope
13 System.out.println("x is " + x);
14 }
15 }
16
Java source file length : 363 lines : 16 Ln : 12 Col : 43 Sel : 0 | 0 Windows (CR LF) UTF-8 INS
Command Prompt
D:\Demonstration\Demonstration-V>javac Demonstration_511.java
Demonstration_511.java:12: error: cannot find symbol
y = 100; // Error: Out of scope
^
symbol: variable y
location: class Demonstration_511
1 error
D:\Demonstration\Demonstration-V>
Type here to search

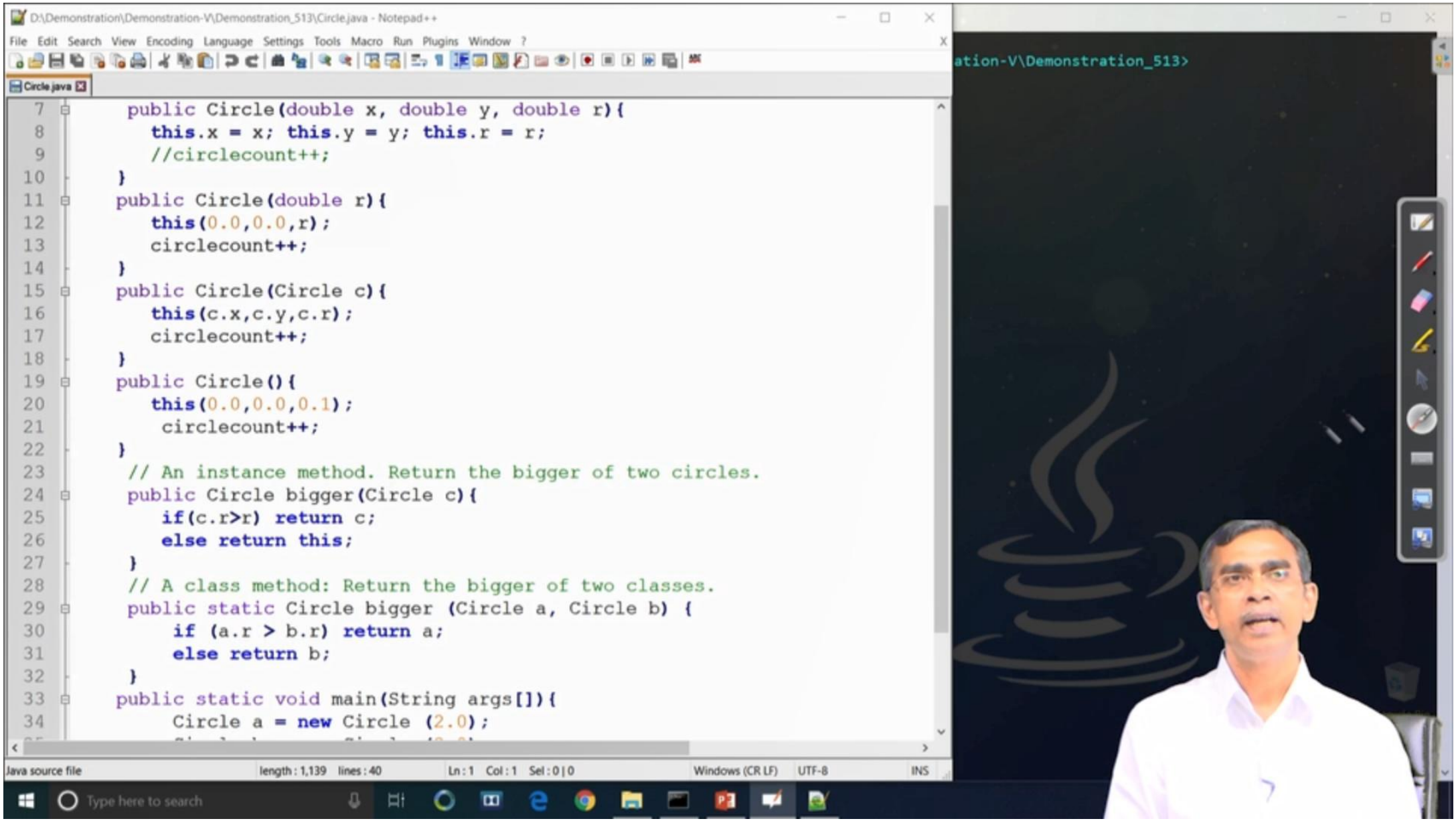

D:\Demonstration\Demonstration-V\Demonstration_512\Circle.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Circle.java
// Example of static variable
public class Circle{
static int circlecount = 0; // class variable
public double x,y,r; // instance variables
public Circle(double x, double y, double r){
this.x = x; this.y = y; this.r = r;
}
public Circle(double r){
this(0.0,0.0,r);
circlecount++;
}
public Circle(Circle c){
this(c.x,c.y,c.r);
circlecount++;
}
public Circle(){
this(0.0,0.0,0.1);
circlecount++;
}
public double circumference(){
return (2*3.14159*r);
}
public double area(){
return(3.14159*r*r);
}
public static void main(String args[ ]){
Circle c1 = new Circle();
Java : length : 1,079 lines : 41
Ln : 1 Col : 1 Sel : 0 | 0
Windows (CR LF)
UTF-8
INS
Command Prompt
D:\Demonstration\Demonstration-V>
Type here to search

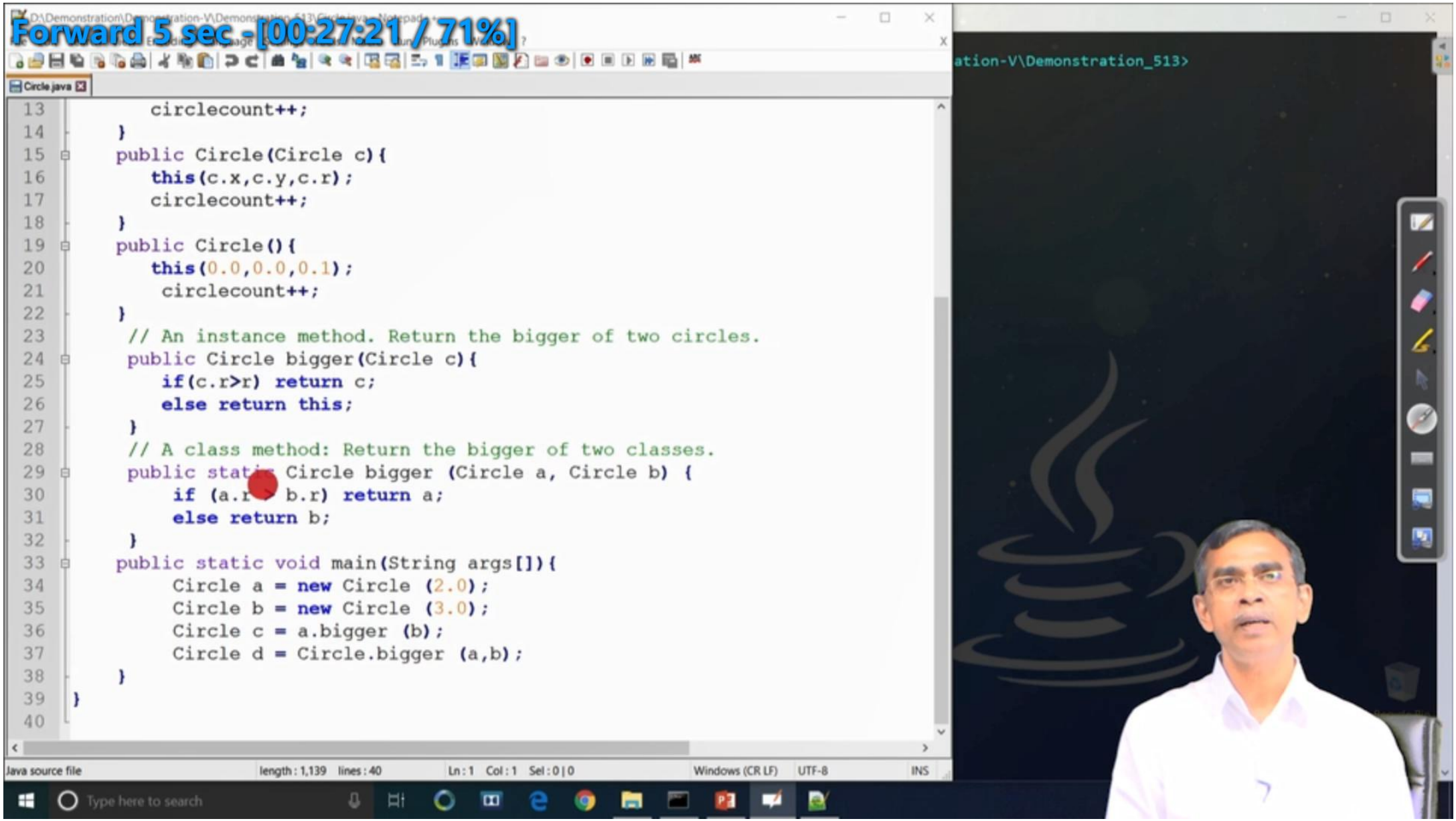

D:\Demonstration\Demonstration-V\Demonstration_512\Circle.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Circle.java
13 public Circle(Circle c){
14 this(c.x,c.y,c.r);
15 circlecount++;
16 }
17 public Circle(){
18 this(0.0,0.0,0.1);
19 circlecount++;
20 }
21
22 public double circumference(){
23 return (2*3.14159*r);
24 }
25 public double area(){
26 return(3.14159*r*r);
27 }
28 public static void main(String args[ ]){
29 Circle c1 = new Circle();
30 System.out.println("c1#"+ c1.circlecount);
31
32 Circle c2 = new Circle(5.0);
33 System.out.println("c2#"+ c2.circlecount);
34
35 Circle c3 = new Circle(c1);
36 System.out.println("c3#"+ c3.circlecount);
37
38 //System.out.println("c1#"+ c1.circlecount + "
39 }
40 }
Java : length : 1,079 lines : 41
Ln : 14 Col : 27 Sel : 28 | 2
Windows (CR LF)
UTF-8
INS
Command Prompt
D:\Demonstration\Demonstration-V>
Type here to search

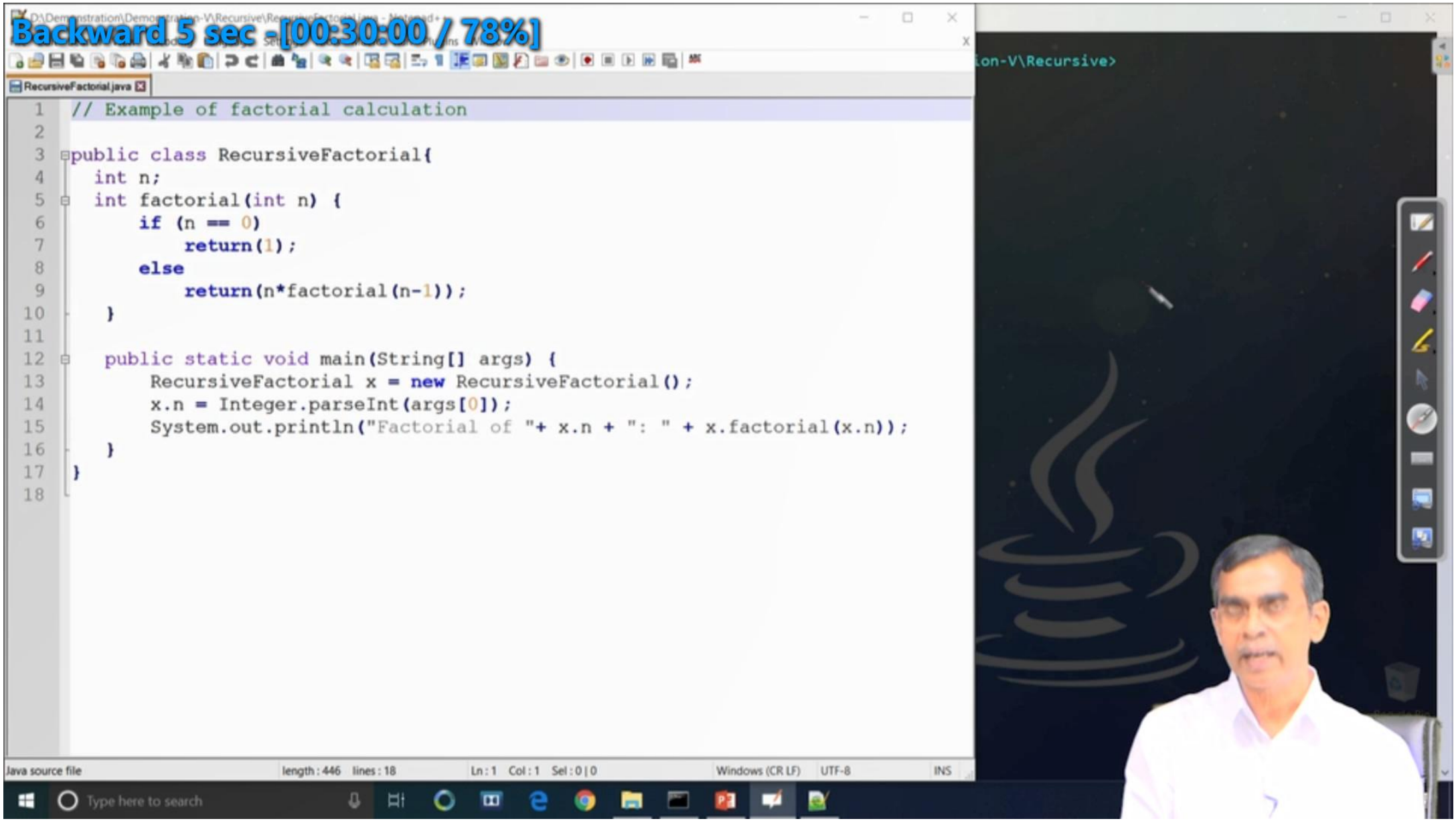

Demonstration_512
File
Home
Share
View
Pin to Quick access
Copy
Paste
Cut
Copy path
Paste shortcut
Clipboard
Move to
Copy to
Delete
Rename
Organize
New folder
New
Properties
Open
Edit
History
Open
Select all
Select none
Invert selection
Select
Demonstration › Demonstration-V › Demonstration_512
Search Demonst...
Quick access
Desktop
Downloads
Documents
Pictures
Apple Computer
Google Drive
Demonstration-\
Dropbox
OneDrive
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
SAMSUNG 750EVO
Program (D:)
Network
Name
Date modified
Type
Size
Circle.class
17-Nov-18 5:48 PM
CLASS File
2 KB
Circle.java
17-Nov-18 2:59 PM
JAVA File
2 KB
2 items
1 item selected 1.05 KB
Type here to search
Command Prompt
D:\Demonstration\Demonstration-V\Demonstration_512>javac Circle.java
D:\Demonstration\Demonstration-V\Demonstration_512>java Circle
c1#1
c2#2
c3#3
D:\Demonstration\Demonstration-V\Demonstration_512>

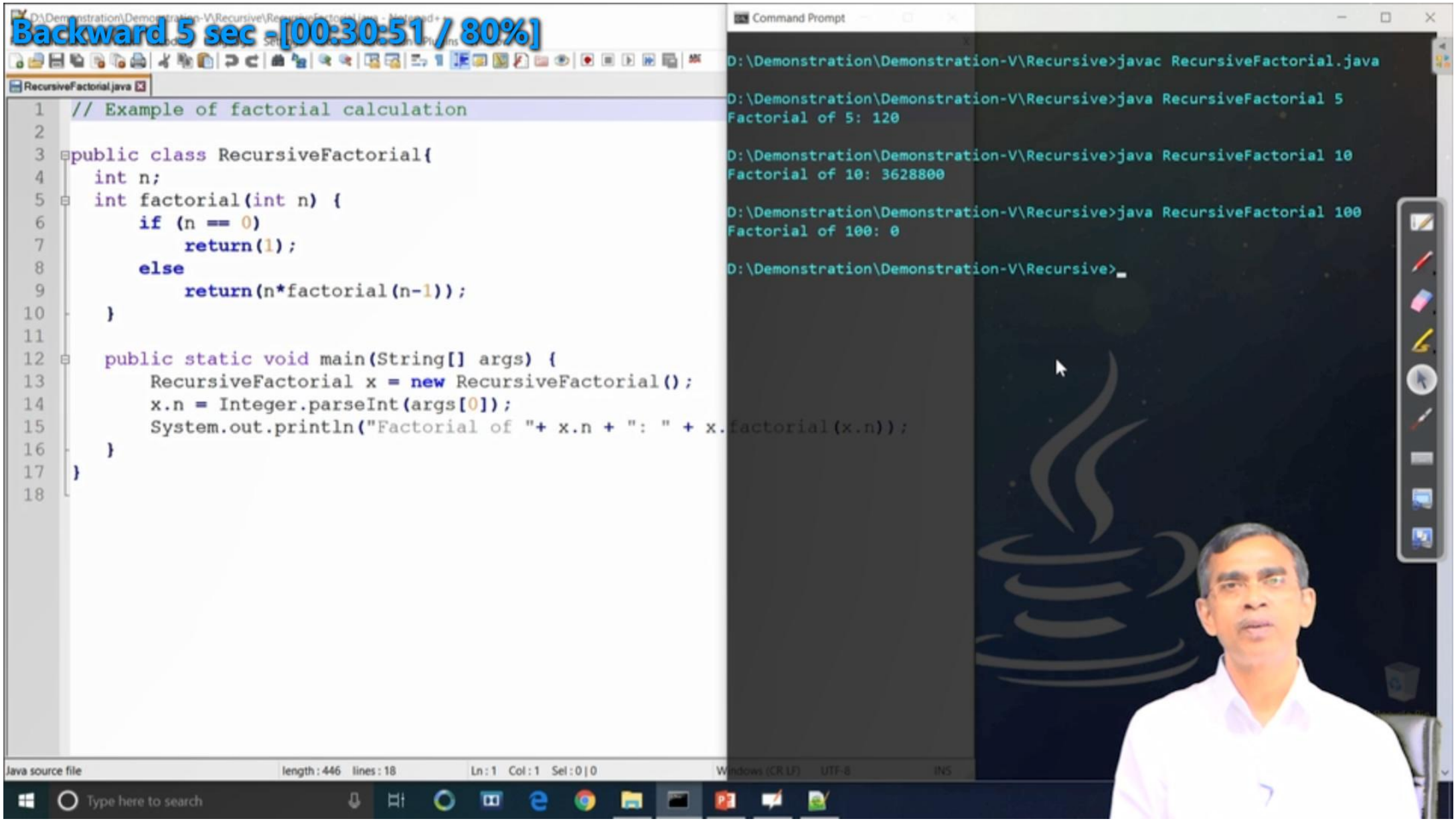

D:\Demonstration\Demonstration-V\Demonstration_512\Circle.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Circle.java
13 public Circle(Circle c){
14 this(c.x,c.y,c.r);
15 circlecount++;
16 }
17 public Circle(){
18 this(0.0,0.0,0.1);
19 circlecount++;
20 }
21
22 public double circumference(){
23 return (2*3.14159*r);
24 }
25 public double area(){
26 return(3.14159*r*r);
27 }
28 public static void main(String args[ ]){
29 Circle c1 = new Circle();
30 System.out.println("c1#"+ c1.circlecount);
31
32 Circle c2 = new Circle(5.0);
33 System.out.println("c2#"+ c2.circlecount);
34
35 Circle c3 = new Circle(c1);
36 System.out.println("c3#"+ c3.circlecount);
37
38 System.out.println("c1#"+ c1.circlecount + " c2# " + c2.circlecount + " c3#"+ c3.circlecount);
39 }
40 }
41
Java source file
length : 1,077 lines : 41
Ln : 38 Col : 9 Sel : 0 | 0
Type here to search

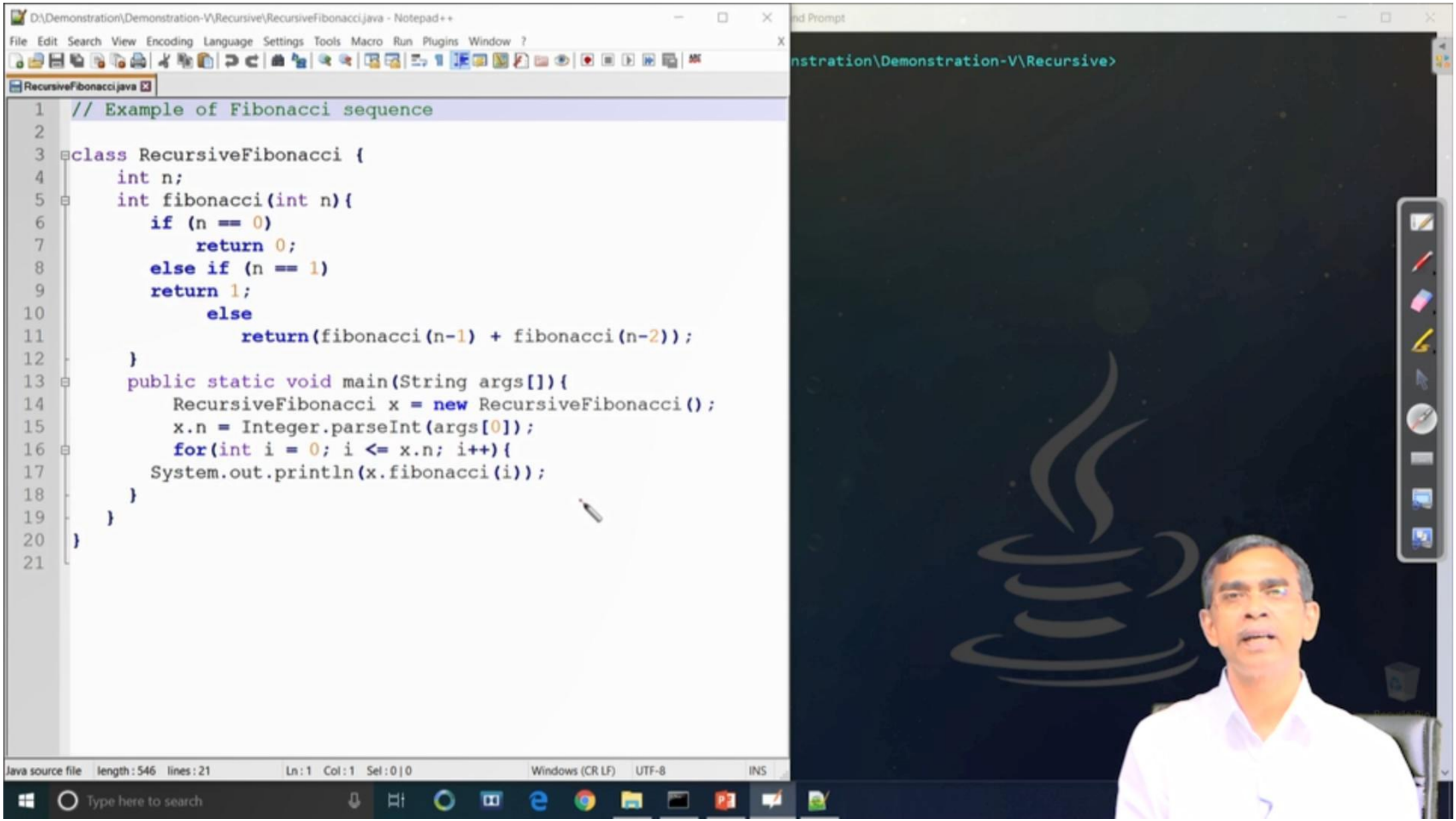

D:\Demonstration\Demonstration-V\Demonstration_512\Circle.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Circle.java
13 public Circle(Circle c){
14 this(c.x,c.y,c.r);
15 circlecount++;
16 }
17 public Circle(){
18 this(0.0,0.0,0.1);
19 circlecount++;
20 }
21
22 public double circumference(){
23 return (2*3.14159*r);
24 }
25 public double area(){
26 return(3.14159*r*r);
27 }
28 public static void main(String args[ ]){
29 Circle c1 = new Circle();
30 System.out.println("c1#"+ c1.circlecount);
31
32 Circle c2 = new Circle(5.0);
33 System.out.println("c2#"+ c2.circlecount);
34
35 Circle c3 = new Circle(c1);
36 System.out.println("c3#"+ c3.circlecount);
37
38 System.out.println("c1#"+ c1.circlecount + "
39 }
40 }
length : 1,077 lines : 41 Ln : 38 Col : 9 Sel : 0 | 0 Windows (CR LF) UTF-8 INS
Command Prompt
D:\Demonstration\Demonstration-V\Demonstration_512>javac Circle.java
D:\Demonstration\Demonstration-V\Demonstration_512>java Circle
c1#1
c2#2
c3#3
D:\Demonstration\Demonstration-V\Demonstration_512>javac Circle.java
D:\Demonstration\Demonstration-V\Demonstration_512>java Circle
c1#1
c2#2
c3#3
c1#3 c2# 3 c3#3
D:\Demonstration\Demonstration-V\Demonstration_512>
Type here to search

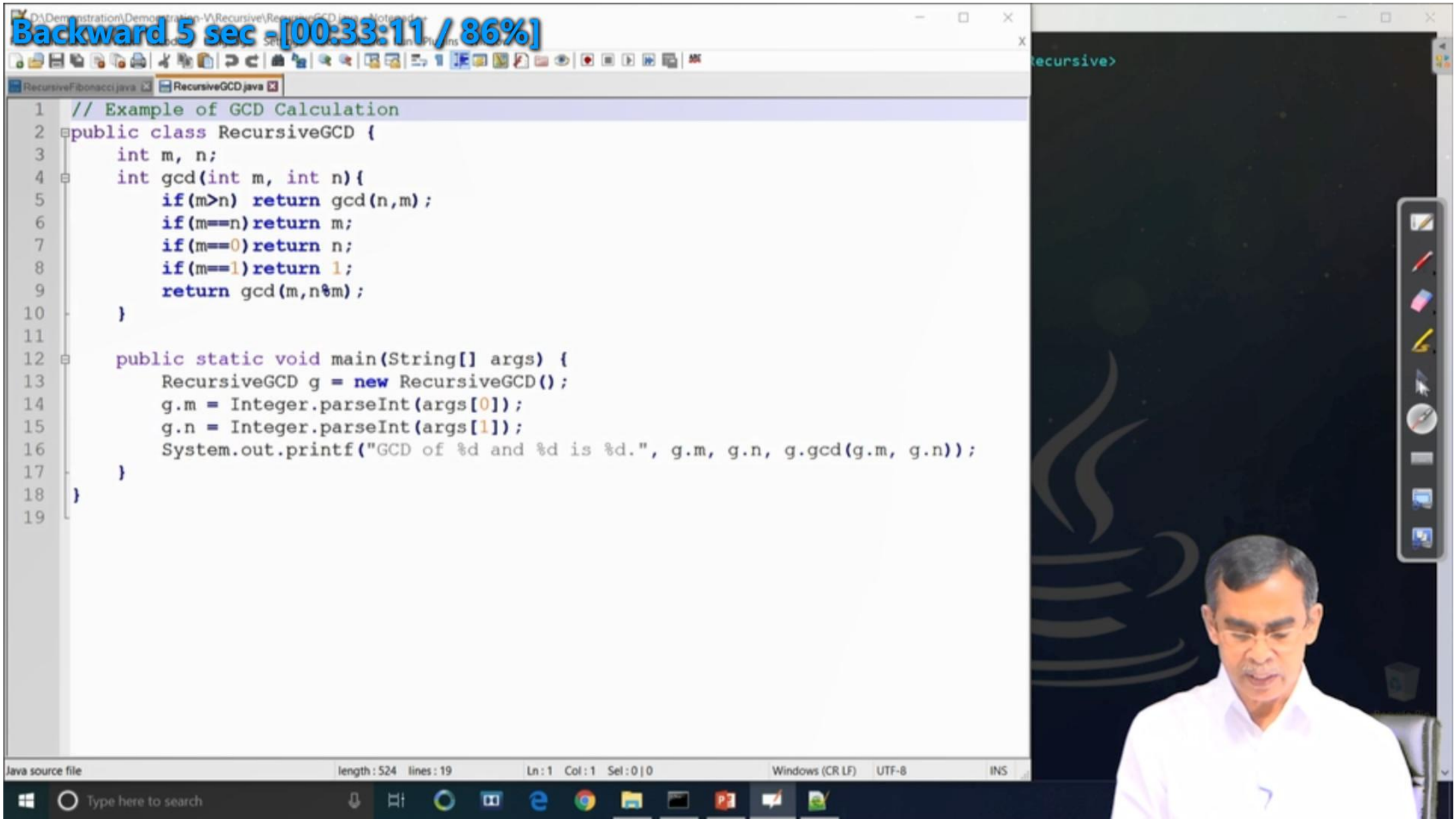

D:\Demonstration\Demonstration-V\Demonstration_513\Circle.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Circle.java
public Circle(double x, double y, double r){
this.x = x; this.y = y; this.r = r;
//circlecount++;
}
public Circle(double r){
this(0.0,0.0,r);
circlecount++;
}
public Circle(Circle c){
this(c.x,c.y,c.r);
circlecount++;
}
public Circle(){
this(0.0,0.0,0.1);
circlecount++;
}
// An instance method. Return the bigger of two circles.
public Circle bigger(Circle c){
if(c.r>r) return c;
else return this;
}
// A class method: Return the bigger of two classes.
public static Circle bigger (Circle a, Circle b) {
if (a.r > b.r) return a;
else return b;
}
public static void main(String args[]){
Circle a = new Circle (2.0);
Java source file
length : 1,139 lines : 40
Ln : 1 Col : 1 Sel : 0 | 0
Windows (CR LF)
UTF-8
INS
ation-V\Demonstration_513>
Type here to search

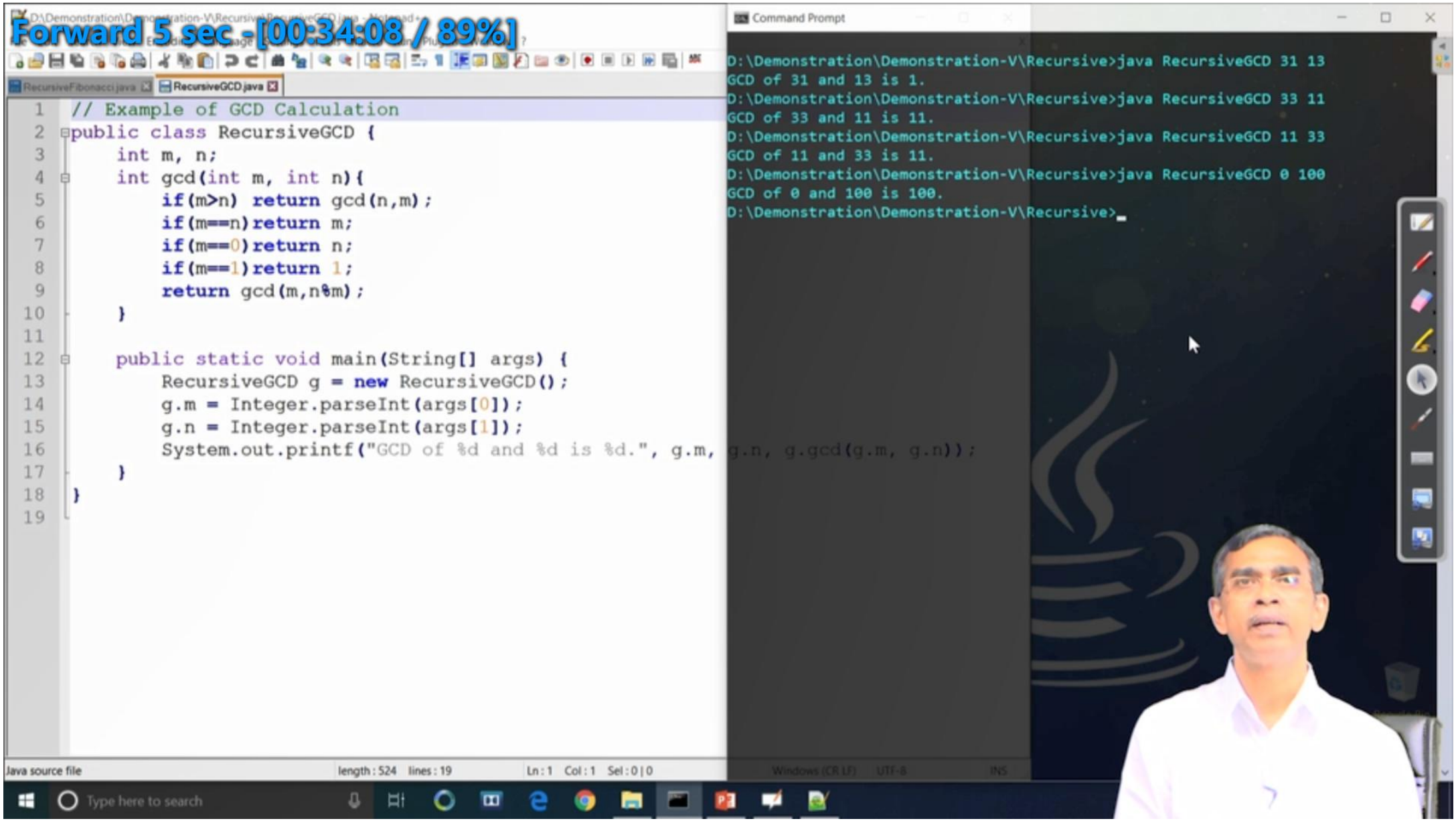

Forward 5 sec - [00:27:21 / 71%]
Circle.java
13 circlecount++;
14 }
15 public Circle(Circle c){
16 this(c.x,c.y,c.r);
17 circlecount++;
18 }
19 public Circle(){
20 this(0.0,0.0,0.1);
21 circlecount++;
22 }
23 // An instance method. Return the bigger of two circles.
24 public Circle bigger(Circle c){
25 if(c.r>r) return c;
26 else return this;
27 }
28 // A class method: Return the bigger of two classes.
29 public static Circle bigger (Circle a, Circle b) {
30 if (a.r > b.r) return a;
31 else return b;
32 }
33 public static void main(String args[]){
34 Circle a = new Circle (2.0);
35 Circle b = new Circle (3.0);
36 Circle c = a.bigger (b);
37 Circle d = Circle.bigger (a,b);
38 }
39 }
40
Java source file
length : 1,139 lines : 40
Ln : 1 Col : 1 Sel : 0 | 0
Windows (CR LF)
UTF-8
INS
ation-V\Demonstration_513>
Type here to search

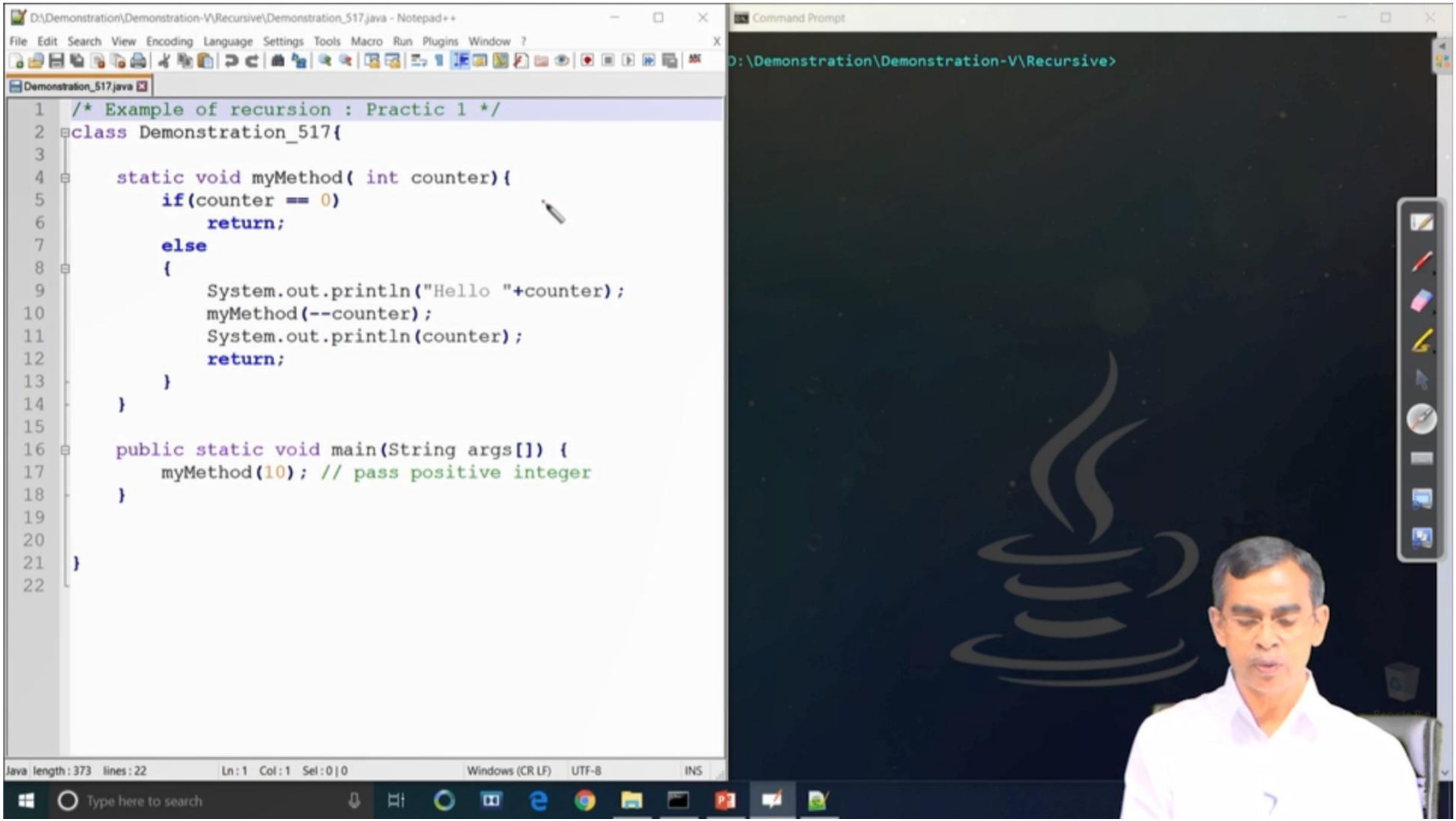

Backward 5 sec - [00:30:00 / 78%]
RecursiveFactorial.java
// Example of factorial calculation
public class RecursiveFactorial{
int n;
int factorial(int n) {
if (n == 0)
return(1);
else
return(n*factorial(n-1));
}
public static void main(String[] args) {
RecursiveFactorial x = new RecursiveFactorial();
x.n = Integer.parseInt(args[0]);
System.out.println("Factorial of "+ x.n + ": " + x.factorial(x.n));
}
}
Java source file
length : 446 lines : 18
Ln : 1 Col : 1 Sel : 0 | 0
Windows (CR LF)
UTF-8
INS
Type here to search

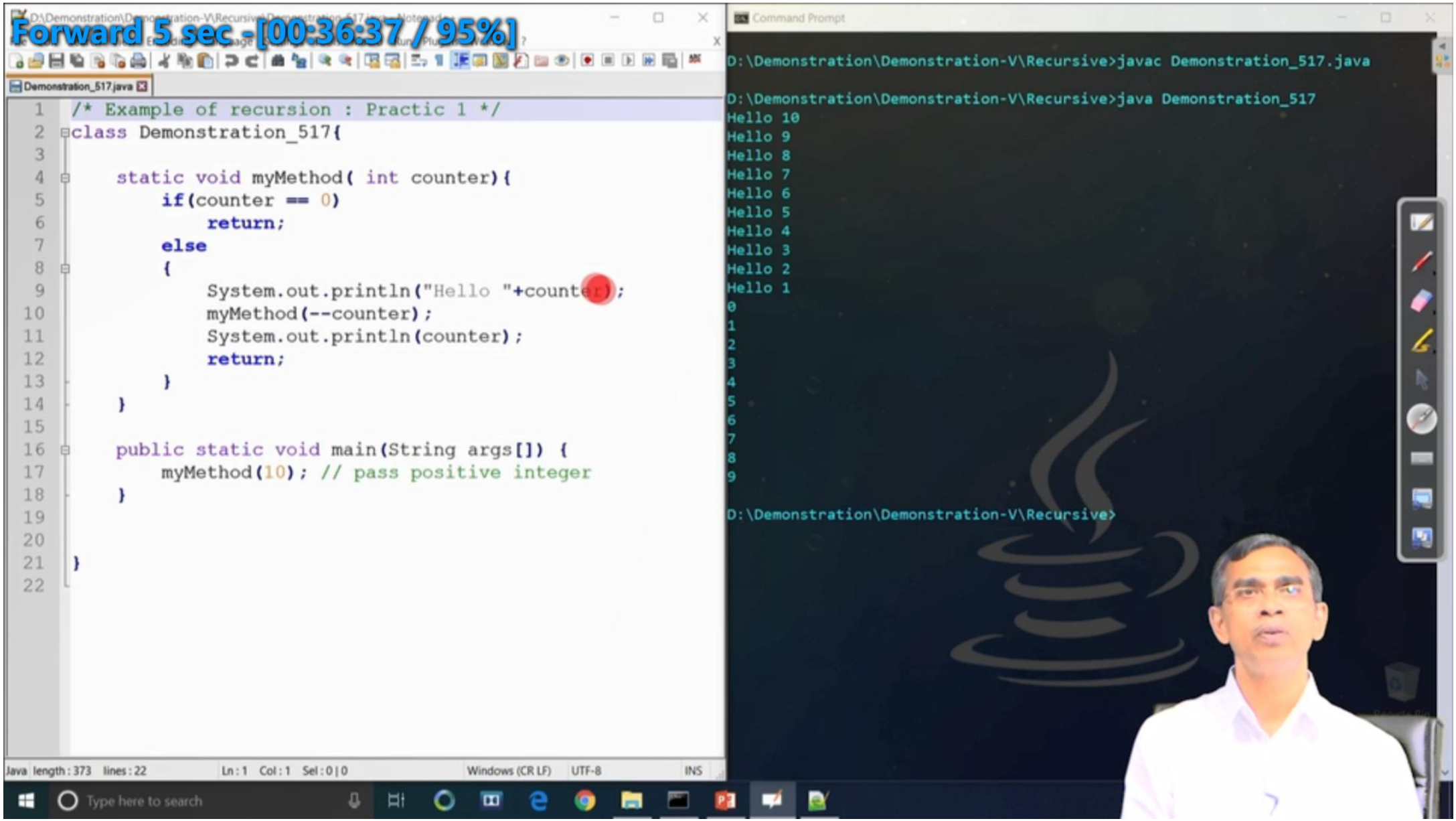

Backward 5 sec - [00:30:51 / 80%]
RecursiveFactorial.java
// Example of factorial calculation
public class RecursiveFactorial{
int n;
int factorial(int n) {
if (n == 0)
return(1);
else
return(n*factorial(n-1));
}
public static void main(String[] args) {
RecursiveFactorial x = new RecursiveFactorial();
x.n = Integer.parseInt(args[0]);
System.out.println("Factorial of "+ x.n + ": " + x.factorial(x.n));
}
}
Java source file
length : 446 lines : 18
Ln : 1 Col : 1 Sel : 0 | 0
Windows (CR LF) UTF-8 INS
Command Prompt
D:\Demonstration\Demonstration-V\Recursive>javac RecursiveFactorial.java
D:\Demonstration\Demonstration-V\Recursive>java RecursiveFactorial 5
Factorial of 5: 120
D:\Demonstration\Demonstration-V\Recursive>java RecursiveFactorial 10
Factorial of 10: 3628800
D:\Demonstration\Demonstration-V\Recursive>java RecursiveFactorial 100
Factorial of 100: 0
D:\Demonstration\Demonstration-V\Recursive>
Type here to search

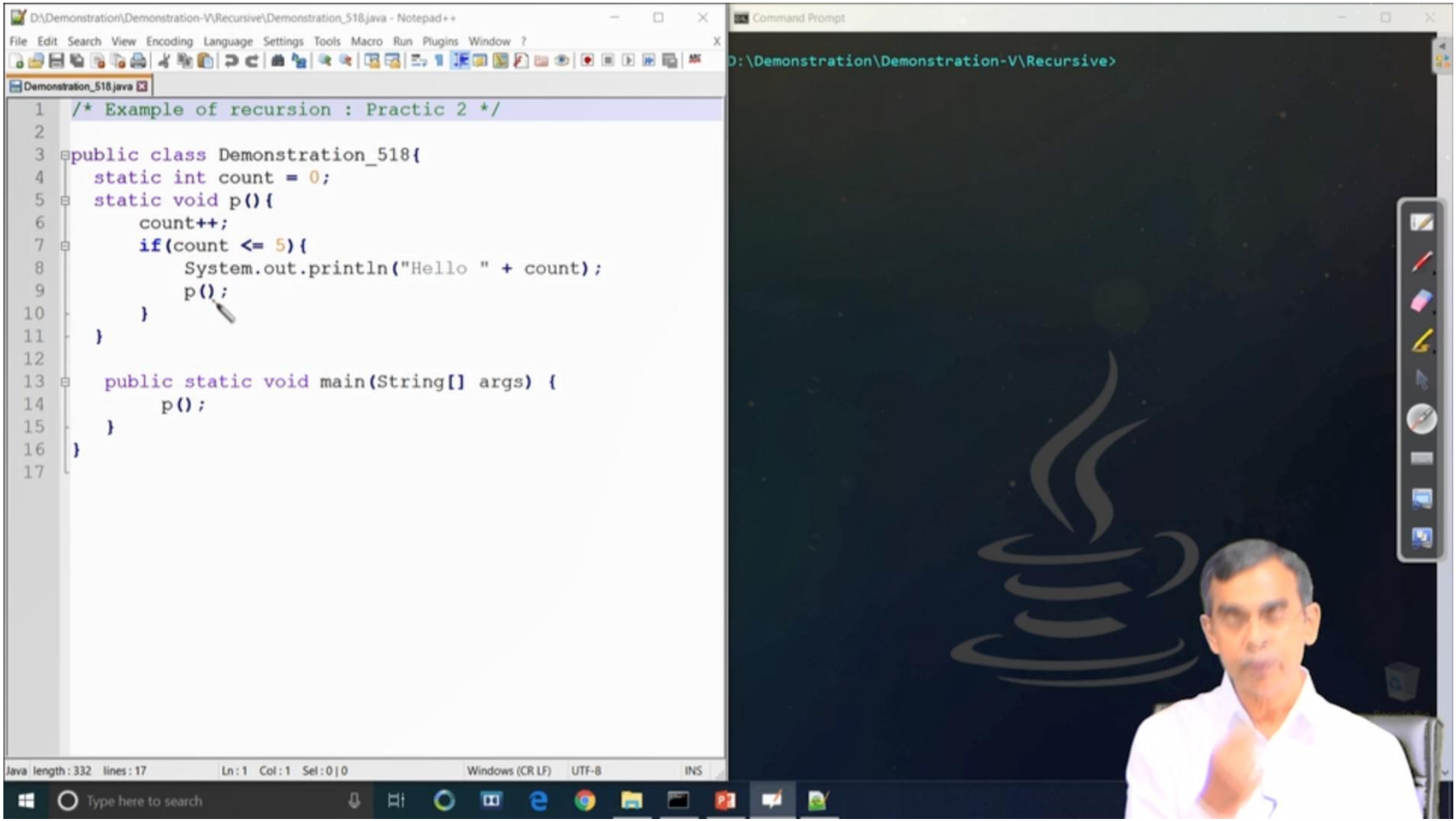

D:\Demonstration\Demonstration-V\Recursive\RecursiveFibonacci.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
RecursiveFibonacci.java
// Example of Fibonacci sequence
class RecursiveFibonacci {
int n;
int fibonacci(int n){
if (n == 0)
return 0;
else if (n == 1)
return 1;
else
return(fibonacci(n-1) + fibonacci(n-2));
}
public static void main(String args[]){
RecursiveFibonacci x = new RecursiveFibonacci();
x.n = Integer.parseInt(args[0]);
for(int i = 0; i <= x.n; i++){
System.out.println(x.fibonacci(i));
}
}
}
Java source file length : 546 lines : 21 Ln : 1 Col : 1 Sel : 0 | 0 Windows (CR LF) UTF-8 INS
nstration\Demonstration-V\Recursive>

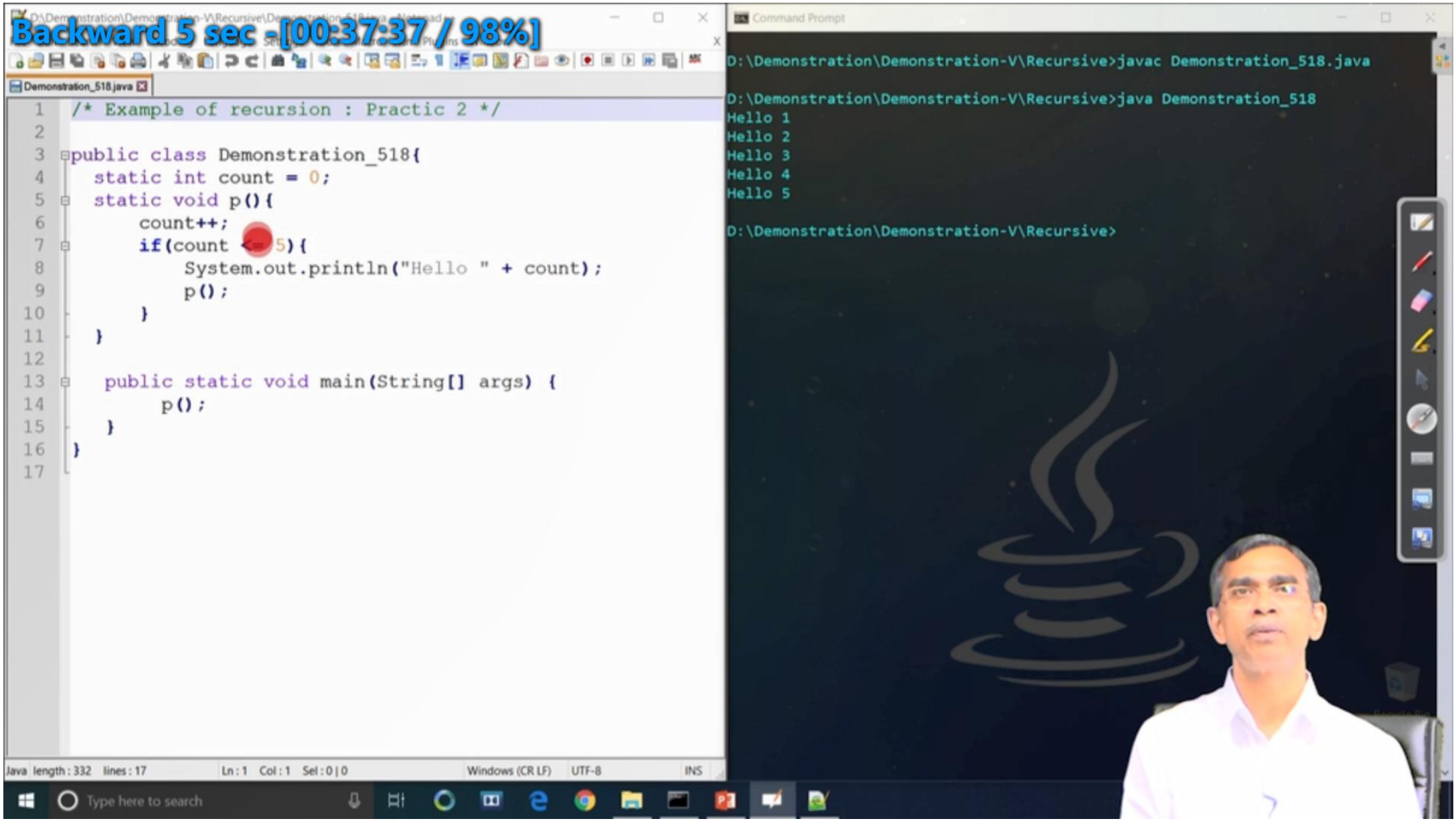

Backward 5 sec - [00:33:11 / 86%]
RecursiveFibonacci.java
RecursiveGCD.java
// Example of GCD Calculation
public class RecursiveGCD {
int m, n;
int gcd(int m, int n){
if(m>n) return gcd(n,m);
if(m==n)return m;
if(m==0)return n;
if(m==1)return 1;
return gcd(m,n%m);
}
public static void main(String[] args) {
RecursiveGCD g = new RecursiveGCD();
g.m = Integer.parseInt(args[0]);
g.n = Integer.parseInt(args[1]);
System.out.printf("GCD of %d and %d is %d.", g.m, g.n, g.gcd(g.m, g.n));
}
}
Java source file
length : 524 lines : 19
Ln : 1 Col : 1 Sel : 0 | 0
Windows (CR LF)
UTF-8
INS
Type here to search


Forward 5 sec - [00:34:08 / 89%]
RecursiveFibonacci.java
RecursiveGCD.java
// Example of GCD Calculation
public class RecursiveGCD {
int m, n;
int gcd(int m, int n){
if(m>n) return gcd(n,m);
if(m==n)return m;
if(m==0)return n;
if(m==1)return 1;
return gcd(m,n%m);
}
public static void main(String[] args) {
RecursiveGCD g = new RecursiveGCD();
g.m = Integer.parseInt(args[0]);
g.n = Integer.parseInt(args[1]);
System.out.printf("GCD of %d and %d is %d.", g.m, g.n, g.gcd(g.m, g.n));
}
}
Java source file
length : 524 lines : 19
Ln : 1 Col : 1 Sel : 0 | 0
Command Prompt
D:\Demonstration\Demonstration-V\Recursive>java RecursiveGCD 31 13
GCD of 31 and 13 is 1.
D:\Demonstration\Demonstration-V\Recursive>java RecursiveGCD 33 11
GCD of 33 and 11 is 11.
D:\Demonstration\Demonstration-V\Recursive>java RecursiveGCD 11 33
GCD of 11 and 33 is 11.
D:\Demonstration\Demonstration-V\Recursive>java RecursiveGCD 0 100
GCD of 0 and 100 is 100.
D:\Demonstration\Demonstration-V\Recursive>
Windows (CR LF) UTF-8 INS
Type here to search


D:\Demonstration\Demonstration-V\Recursive\Demonstration_517.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Demonstration_517.java
/* Example of recursion : Practic 1 */
class Demonstration_517{
static void myMethod(int counter){
if(counter == 0)
return;
else
{
System.out.println("Hello "+counter);
myMethod(--counter);
System.out.println(counter);
return;
}
}
public static void main(String args[]) {
myMethod(10); // pass positive integer
}
}
Java length : 373 lines : 22 Ln : 1 Col : 1 Sel : 0 | 0 Windows (CR LF) UTF-8 INS
Command Prompt
D:\Demonstration\Demonstration-V\Recursive>
Type here to search

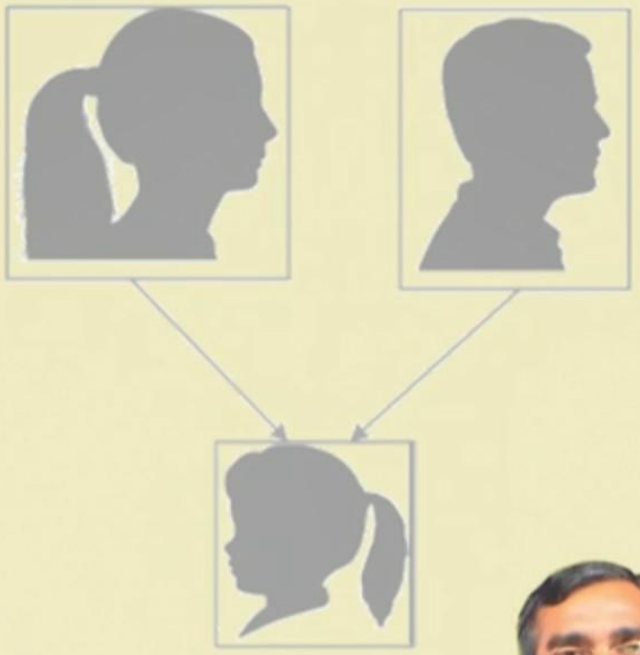

Forward 5 sec - [00:36:37 / 95%]
Demonstration_517.java
/* Example of recursion : Practic 1 */
class Demonstration_517{
static void myMethod(int counter){
if(counter == 0)
return;
else
{
System.out.println("Hello "+counter);
myMethod(--counter);
System.out.println(counter);
return;
}
}
public static void main(String args[]) {
myMethod(10); // pass positive integer
}
}
Java length : 373 lines : 22
Ln : 1 Col : 1 Sel : 0 | 0
Windows (CR LF)
UTF-8
INS
Command Prompt
D:\Demonstration\Demonstration-V\Recursive>javac Demonstration_517.java
D:\Demonstration\Demonstration-V\Recursive>java Demonstration_517
Hello 10
Hello 9
Hello 8
Hello 7
Hello 6
Hello 5
Hello 4
Hello 3
Hello 2
Hello 1
0
1
2
3
4
5
6
7
8
9
D:\Demonstration\Demonstration-V\Recursive>
Type here to search


D:\Demonstration\Demonstration-V\Recursive\Demonstration_518.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Demonstration_518.java
/* Example of recursion : Practic 2 */
public class Demonstration_518{
static int count = 0;
static void p(){
count++;
if(count <= 5){
System.out.println("Hello " + count);
p();
}
}
public static void main(String[] args) {
p();
}
}
Java length : 332 lines : 17 Ln : 1 Col : 1 Sel : 0 | 0 Windows (CR LF) UTF-8 INS
Command Prompt
D:\Demonstration\Demonstration-V\Recursive>
Type here to search


Backward 5 sec - [00:37:37 / 98%]
Demonstration_518.java
/* Example of recursion : Practic 2 */
public class Demonstration_518{
static int count = 0;
static void p(){
count++;
if(count <= 5){
System.out.println("Hello " + count);
p();
}
}
public static void main(String[] args) {
p();
}
}
Java length : 332 lines : 17 Ln : 1 Col : 1 Sel : 0 | 0 Windows (CR LF) UTF-8 INS
Command Prompt
D:\Demonstration\Demonstration-V\Recursive>javac Demonstration_518.java
D:\Demonstration\Demonstration-V\Recursive>java Demonstration_518
Hello 1
Hello 2
Hello 3
Hello 4
Hello 5
D:\Demonstration\Demonstration-V\Recursive>
Type here to search

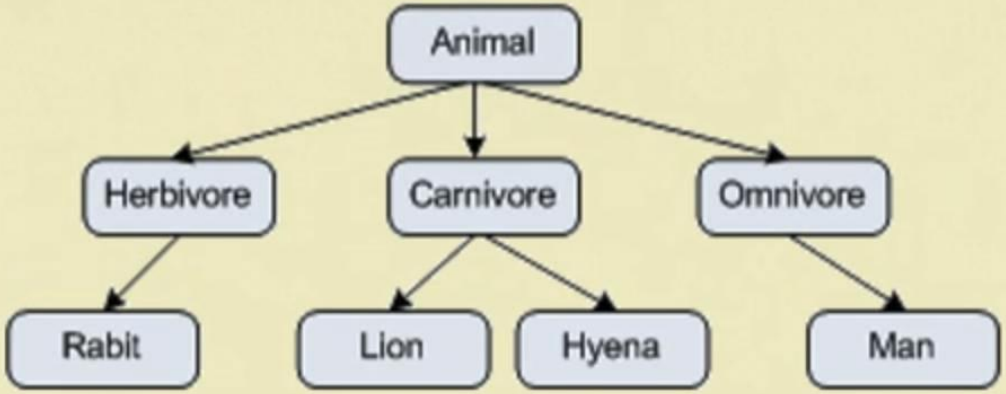
Concept of inheritance

Single inheritance

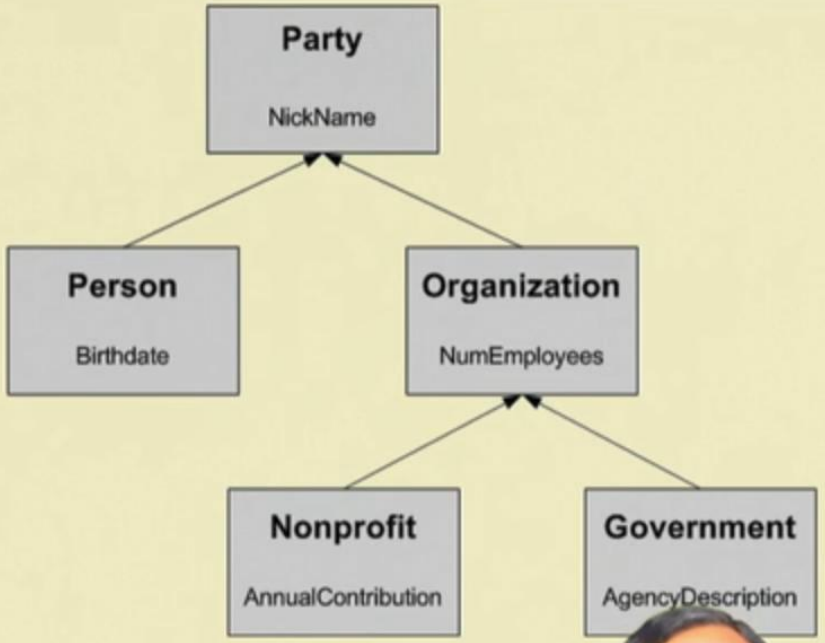


Multiple inheritanc

Concept of inheritance



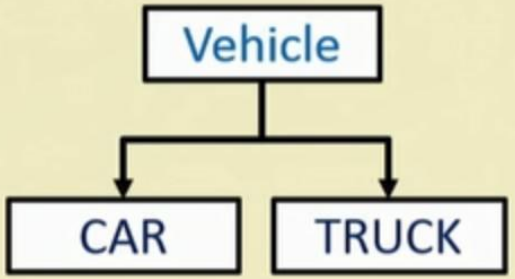
Single multi-level inheritance



Class hierarchy

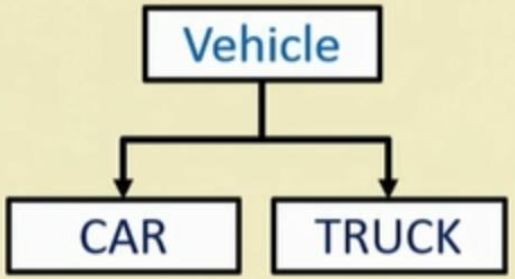
Inheritance in Java

- Inheritance is one of the cornerstone of object-oriented programming because it allows the creation of hierarchical classification.
- Using inheritance, one can create a general class that include some common set of items.
- This class then can be used to create more specific classes which has all the items from the base class, in addition to some items of its own.

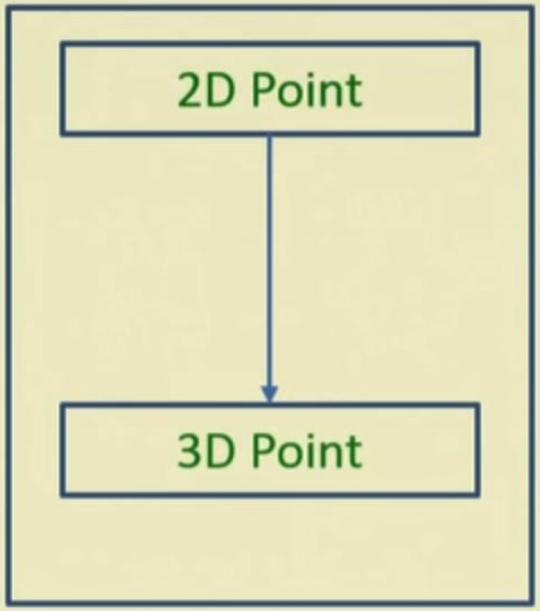


Inheritance in Java

- Inheritance is one of the cornerstone of object-oriented programming because it allows the creation of hierarchical classification.
- Using inheritance, one can create a general class that include some common set of items.
- This class then can be used to create more specific classes which has all the items from the base class, in addition to some items of its own.

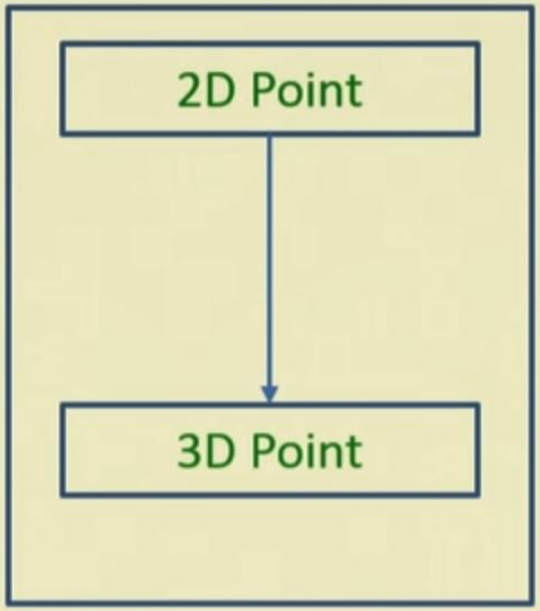


Example of a simple Inheritance




```
class simpleSingleInheritance{
    public static void main(String arge[]){
        Point2D new P1();
        Point3D new P2();
        P1.x = 10;
        P1.y = 20;
        System.out.println("Point2D P1 is" + P1.display());
        // Initializing Point3D
        P2.x = 5;
        P2.y = 6;
        P2.z = 15
        System.out.println("Point3D P2 is" + P2.display());
    }
}
```

Example of a simple Inheritance

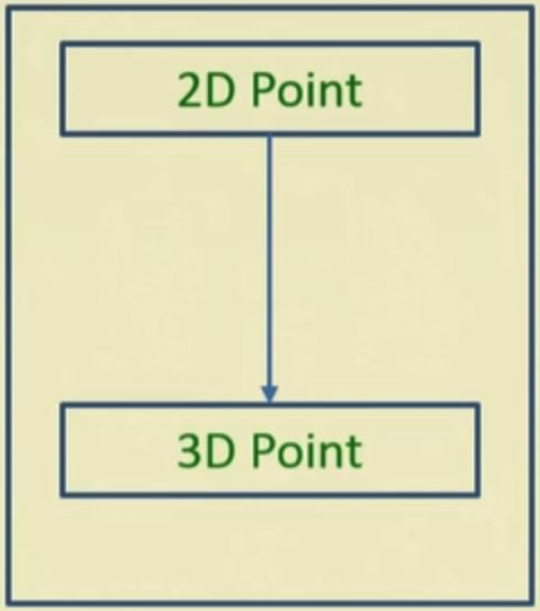



```
class Point2D{
            int x;
            int y;
            void display(){
                        System.out.println ("x="+x+"y="+y);
            }
}
```

```
class Point3D extends Point3D{
            int z;
            void display(){
                        System.out.println("x="+x+"y="+y+"z="+z);
            }
}
```

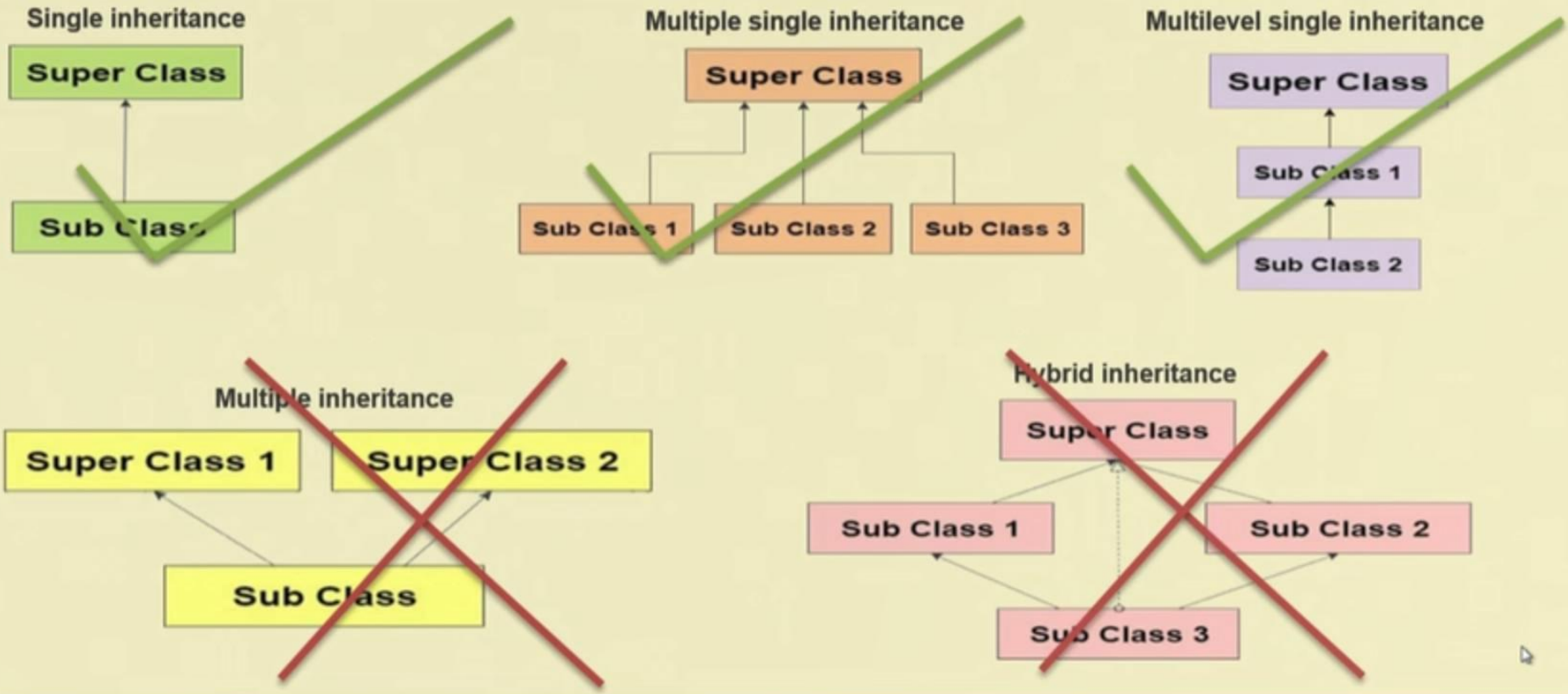
Pause

Example of a simple Inheritance

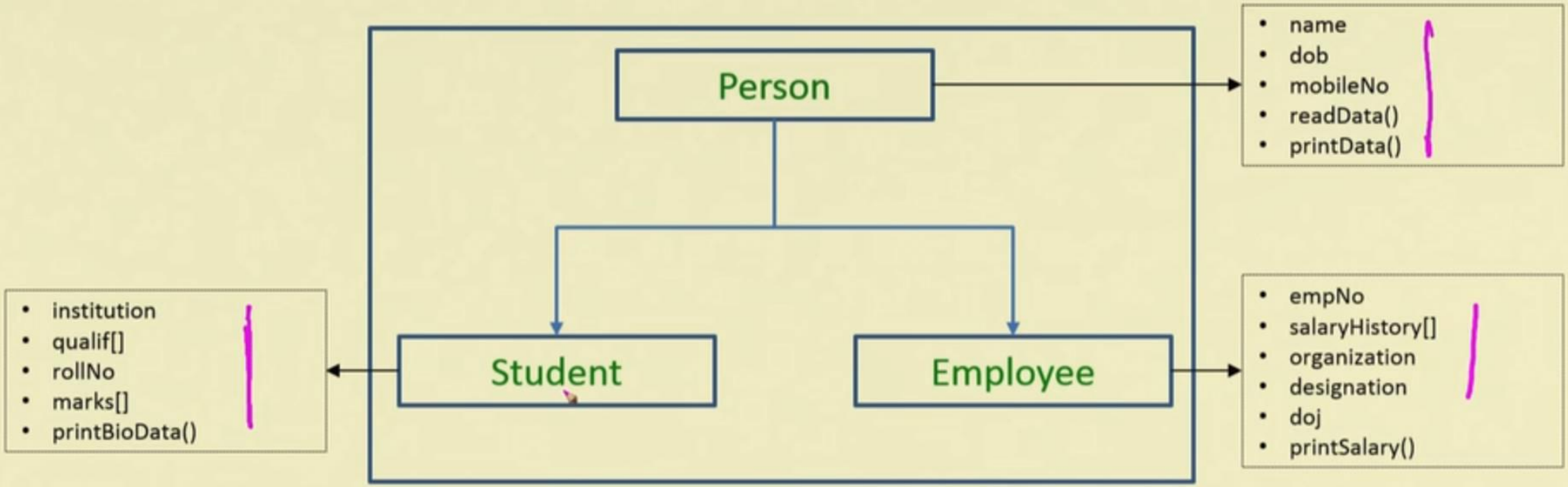



```
class simpleSingleInheritance{
     public static void main(String arge[]){
          Point2D new P1();
          Point3D new P2();
          P1.x = 10;
          P1.y = 20;
          System.out.println("Point2D P1 is" + P1.display());
          // Initializing Point3D
          P2.x = 5;
          P2.y = 6;
          P3.z = 15;
          System.out.println("Point3D P2 is" + P2.display());
     }
}
```

Inheritance types



Single inheritance : An example



Play

Single Inheritance : Person class

```
class Person{
        String name;
        Date dob;
        int mobileNo;
        void readData(String n, Date d, int m){
            name = n;
            dob = d;
            mobileNo = m;
        }
        void printData(){
            System.out.println("Name : "+ name);
            dob.printDate();
            System.out.println("Mobile : "+ mobileNo);
        }
}
```

Single inheritance : Student class

```
class Person{
    String name;
    Date dob;
    int mobileNo;
    void readData(String n, Date d, int m){
        name = n;
        dob = d;
        mobileNo = m;
    }
    void printData(){
        System.out.println("Name : "+ name);
        dob.printDate();
        System.out.println("Mobile : "+ mobileNo);
    }
}
```

```
class Student extends Person{
        String institution;
        int[] qualif = new int[20];
        int rollNo;
        int[] marks = new int[5];

        void printBioData(){
            printData();
            System.out.println("Institution : "+ institution);
            System.out.println("Roll : "+ rollNo);
            for(int q=0; q<qualif.length;q++){
               System.out.println("Marks "+q+": "+ qualif[q]);
            }
            for(int m=0; m<marks.length;m++){
               System.out.print("Result "+m+": "+marks[m]);
            }
        }
}
```

Single inheritance : Student class

```
class Person{
   String name;
   Date dob;
   int mobileNo;
   void readData(String n, Date d, int m){
       name = n;
      dob = d;
      mobileNo = m;
   }
   void printData(){
        System.out.println("Name : "+ name);
        dob.printDate();
        System.out.println("Mobile : "+ mobileNo);
   }
}
```

```
class Student extends Person{
        String institution;
        int[] qualif = new int[20];
        int rollNo;
        int[] marks = new int[5];

        void printBioData(){
            printData();
            System.out.println("Institution : "+ institution);
            System.out.println("Roll : "+ rollNo);
            for(int q=0; q<qualif.length;q++){
               System.out.println("Marks "+q+": "+ qualif[q]);
            }
            for(int m=0; m<marks.length;m++){
               System.out.print("Result "+m+": "+marks[m]);
            }
        }
}
```

Single Inheritance – employee

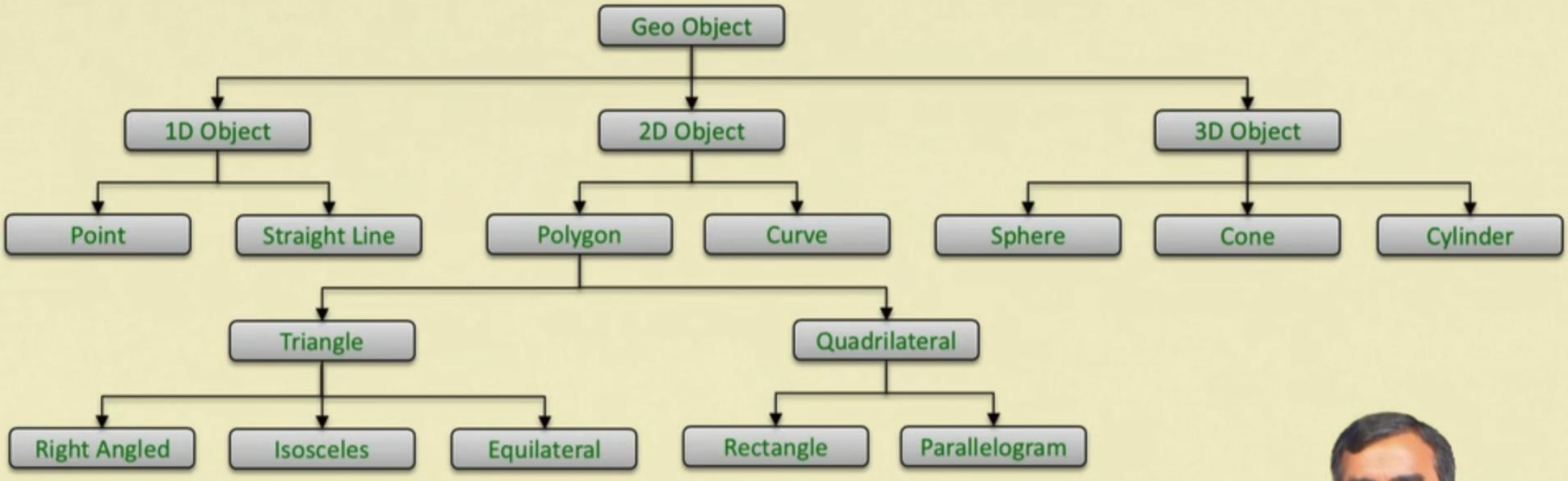
```
class Person{
    String name;
    Date dob;
    int mobileNo;
    void readData(String n, Date d, int m){
        name = n;
       dob = d;
       mobileNo = m;
    }
    void printData(){
         System.out.println("Name : "+ name);
         dob.printDate();
         System.out.println("Mobile : "+ mobileNo);
    }
}
```

```
class Employee extends Person{
        int empNo;
        int[] salaryHistory = new int[12];
        String organization;
        String designation;
        Date doj;
        void printSalary(){
            for(int s=0; s<salaryHistory.length;s++){
               System.out.println("Salary "+s+": "+salaryHistory[s]);
            }
        }
}
```

Single Inheritance : An example

```
class inheritanceDemo1{
      public static void main(String args[]){
          Person p = new Person();
          //Code with the objects p…
          Student s = new Student [100];
          //Code with the objects s…
          Employee e = new Employee[50];
          //Code with the objects e…

      }
}
```

Multilevel inheritance : An example



Method overriding concept

Usage of Java Method Overriding

- Method overriding is used to provide the specific implementation of a method which is already provided by its superclass.
- Method overriding is used for runtime polymorphism

Rules for Java Method Overriding

- The method must have the same name as in the parent class
- The method must have the same parameter as in the parent class.
- There must be an IS-A relationship (inheritance).

Method overriding : An example

```
class Point2D{
        int x;
        int y;
        Point2D(int a, int b){
            x = a;
            y = b;
        }
        void display(){
             System.out.println("x = "+x+"y = "+y);
        }
}
```

```
class Point3D extends Point3D{
        int z;
        Point3D(int c){
            z = c;
        }
        void display(){
            System.out.println("x="+x+"y="+y+"z="+z);
        }
}
```

```
class MethodOverridingTest{
     public static void main(String args[]){
          Point2D p = new Point2D(3.0, -4.0);
          p.display(); // Refers to the method in Point2D

          Point3D q = new Point3D(0.0);
          q.display(); // Refers to the method in Point3D

          Point2D x =(Point2D) q; // Cast q to an instance of class Point2D
          x.display();
          }
}
```


Method overriding : An example

```
class Point2D{
        int x;
        int y;
        Point2D(int a, int b){
            x = a;
            y = b;
        }
        void display(){
             System.out.println("x = "+x+"y = "+y);
        }
}
```

```
class Point3D extends Point3D{
        int z;
        Point3D(int c){
            z = c;
        }
        void display(){
            System.out.println("x="+x+"y="+y+"z="+z);
        }
}
```

```
class MethodOverridingTest{
     public static void main(String args[]){
          Point2D p = new Point2D(3.0, -4.0);
          p.display(); // Refers to the method in Point2D

          Point3D q = new Point3D(0.0);
          q.display(); // Refers to the method in Point3D

          Point2D x =(Point2D) q; // Cast q to an instance of class Point2D
          x.display();
          }
}
```

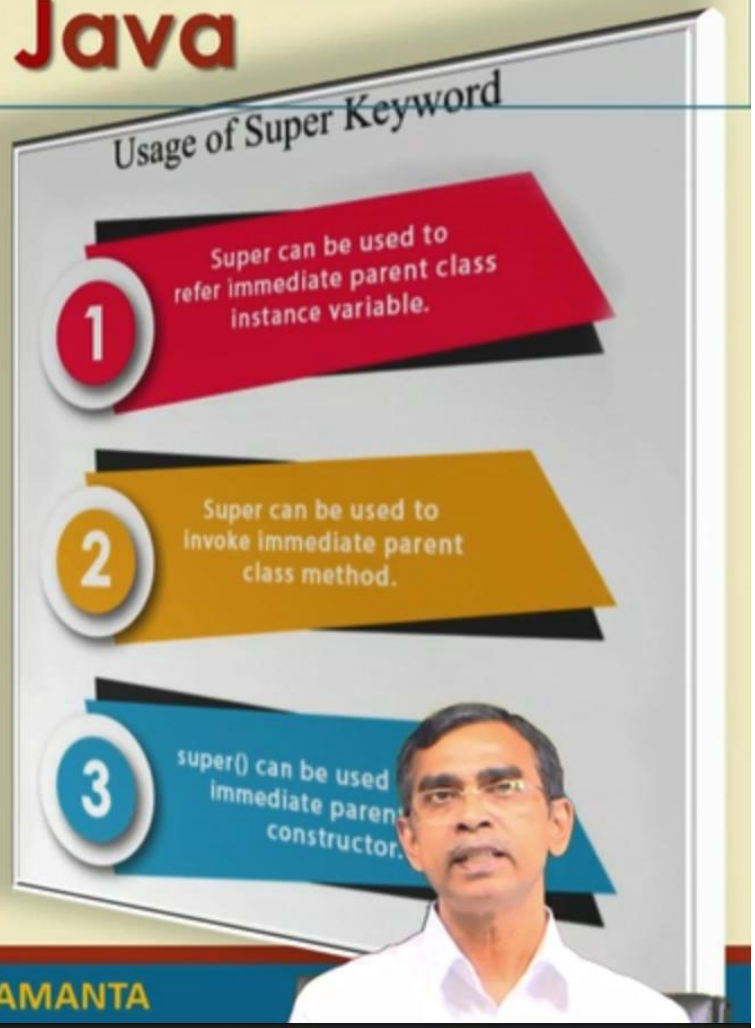
Note

- A sub class object can reference a super class variable or method if it is not overridden.
- A super class object cannot reference a variable or method which is explicit to the sub class object.

super Keyword concept in Java

The **super** keyword in Java is a reference variable which is used to refer immediate parent class members.

Whenever you create an instance of a sub class, an instance of its parent class is created implicitly, which is referred by **super** keyword.



super : Referring parent class instance variable

```
class Animal{
   String color="white";
}
class Dog extends Animal{
   String color = "black";
   void printColor(){
      System.out.println(color);
      System.out.println(super.color);
   }
}
class TestSuper1{
   public static void main(String args[]){
      Dog d = new Dog();
      d.printColor();
   }
}
```

super : Referring parent class instance variable

```
class Animal{
   String color="white";
}
class Dog extends Animal{
   String color = "black";
   void printColor(){
      System.out.println(color);
      System.out.println(super.color);
   }
}
class TestSuper1{
   public static void main(String args[]){
      Dog d = new Dog();
      d.printColor();
   }
}
```

super : Referring parent class instance variable

```
class Animal{
   String color="white";
}
class Dog extends Animal{
   String color = "black";
   void printColor(){
      System.out.println(color);
      System.out.println(super.color);
   }
}
class TestSuper1{
   public static void main(String args[]){
      Dog d = new Dog();
      d.printColor();
   }
}
```

black
white

super : Referring parent class instance variable

```
class Animal{
   String color="white";
}
class Dog extends Animal{
   String color = "black";
   void printColor(){
      System.out.println(color);
      System.out.println(super.color);
   }
}
class TestSuper1{
   public static void main(String args[]){
      Dog d = new Dog();
      d.printColor();
   }
}
```

```
black
white
```

Animal and **Dog** both classes have a common property color. If you print the color property, it will print the color of the current class by default. To access the parent property, you should use `super` keyword.

super : Invoking parent class method

```
class Animal{
   void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
   void eat(){System.out.println("eating bread...");}
   void bark(){System.out.println("barking...");}
   void work(){
           super.eat();
           bark();
           eat();}
   }
class TestSuper2{
   public static void main(String args[]){
      Dog d = new Dog();
      d.work();
   }
}
```

super : Invoking parent class method

```
class Animal{
   void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
   void eat(){System.out.println("eating bread...");}
   void bark(){System.out.println("barking...");}
   void work(){
           super.eat();
           bark();
           eat();}
   }
class TestSuper2{
   public static void main(String args[]){
      Dog d = new Dog();
      d.work();
   }
}
```

```
eating...
barking...
Eating bread...
```

super : Invoking parent class method

```
class Animal{
   void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
   void eat(){System.out.println("eating bread...");}
   void bark(){System.out.println("barking...");}
   void work(){
           super.eat();
           bark();
           eat();}
   }
class TestSuper2{
   public static void main(String args[]){
      Dog d = new Dog();
      d.work();
   }
}
```

Animal and Dog both the classes have eat() method. If you call eat() method from Dog class, it will call the eat() method of Dog class by default because priority is given to local.

To call the parent class method, you need to use super keyword.

```
eating...
barking...
Eating bread...
```

super : Invoking parent class constructor

```
class Animal{
  Animal(){System.out.println("animal is created");}
}
class Dog extends Animal{
    Dog(){
       super();
       System.out.println("dog is created");
    }
}

class TestSuper3{
   public static void main(String args[]){
      Dog d = new Dog();
   }
}
```

super : Invoking parent class constructor

```
class Animal{
  Animal(){System.out.println("animal is created");}
}
class Dog extends Animal{
    Dog(){
       super();
       System.out.println("dog is created");
    }
}

class TestSuper3{
   public static void main(String args[]){
      Dog d = new Dog();
   }
}
```

super : Invoking parent class constructor

```
class Animal{
  Animal(){System.out.println("animal is created");}
}
class Dog extends Animal{
    Dog(){
       super();
       System.out.println("dog is created");
    }
}

class TestSuper3{
   public static void main(String args[]){
      Dog d = new Dog();
   }
}
```

animal is created
dog is created


super : Invoking parent class constructor

```
class Animal{
  Animal(){System.out.println("animal is created");}
}
class Dog extends Animal{
    Dog(){
       super();
       System.out.println("dog is created");
    }
}

class TestSuper3{
   public static void main(String args[]){
      Dog d = new Dog();
   }
}
```

animal is created
dog is created

The super keyword can also be used to invoke the overloaded parent class constructor, if arguments are there, then they should be specified accordingly.

super : Invoking parent class constructor

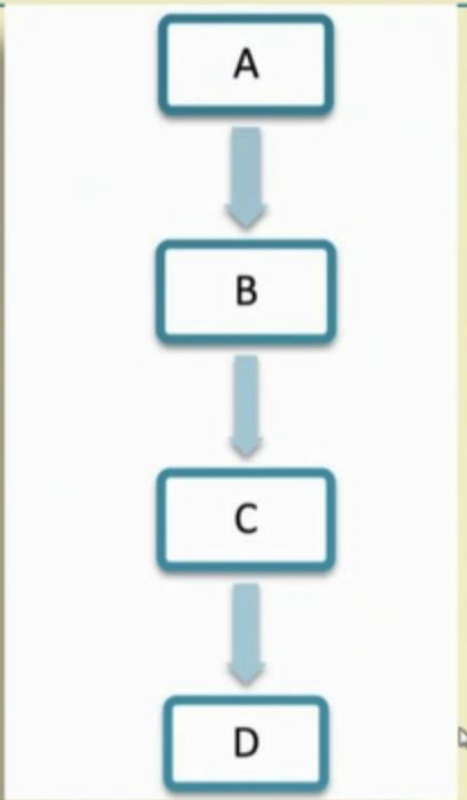
```
class Point2D{
   double x, y;
   Point2D(){x = 0.0; y = 0.0} //Default initialization
   Point2D(double x, double y){thix.x = x; this.y = y;}
}
class Point3D extends Point2D{
   double z;
   Point3D(){super(); z = 0.0} //Default initialization
   Point3D(double x, double y, double z){
   super(x, y);
   this.z = z; }
}
class TestSuper4{
   public static void main(String args[]){
        Point3D p = new Point3D(2.0, 3.0, 4.0);
   }
}
```

If there is a number of overloading constructors in the super class, then you have to define the super constructors matching with each constructor.

Dynamic method dispatch concept

Dynamic method dispatch is a process in which a call to an overridden method is resolved at runtime rather than compile-time. Also, it is called Runtime polymorphism.

In this process, an overridden method is called through the reference variable of a super class. The determination of the method to be called is based on the object being referred to by the reference variable.



super - refer parent class instance variable

```
class Bike{
    void run(){System.out.println("running");}
}
class Splendor extends Bike{
   void run(){System.out.println("running safely with 60km");
   }

   public static void main(String args[]){
        Splendor b1 = new Splendor();
        b1.run();
        Bike b2 = new Bike();
        b2.run();
        Bike b3 = new Splendor();//Up casting
        b3.run();
   }
}
```

running safely with 60k
Running
running safely with 6

Dynamic binding in Java: Example

```
class A {
        void callMe (  ) {
            System.out.println ( "I am from A ") ;
        }
}

class B  extends A {
        void callMe (  ) {
           System.out.println ( "I am from B ");
        }
}

class Who {
        public void static main (String args [  ] ) {
           A  a = new B ( ) ;
           a.callMe();
           B b = new B( );
           b.callMe();
        }
}
```

Dynamic binding in Java: Example

```
class A {
        void callMe (  ) {
            System.out.println ( "I am from A ") ;
        }
}

class B  extends A {
        void callMe (  ) {
           System.out.println ( "I am from B ");
        }
}

class Who {
        public void static main (String args [  ] ) {
           A  a = new B ( ) ;
           a.callMe();
           B b = new B( );
           b.callMe();
        }
}
```

I am from B
I am from B

KMPlayer

Lecture 13 _ Inheritance.mp4

Abstract class in Java

NPTEL ONLINE

DEBASIS SAMANTA

00:29:56

00:33:22

Abstract class in Java : Example

```
abstract class Bike{
    abstract void run();
}

 class Honda extends Bike{
     void run(){
         System.out.println("Running safely");
     }

    public static void main(String args[]){
        Bike obj = new Honda();
        obj.run();
    }
 }
```

Abstract concept

Points to remember

- An abstract class must be declared with an **`abstract`** keyword.
- It can have abstract and non-abstract methods.
- It cannot be instantiated.
- It can have constructors and static methods also.
- It can have final methods which will force the sub class not to change the body of the method.

Pause

Abstract class in Java : Example

```
abstract class Bike{
    abstract void run();
}

 class Honda extends Bike{
     void run(){
         System.out.println("Running safely");
     }

    public static void main(String args[]){
        Bike obj = new Honda();
        obj.run();
    }
 }
```

Abstract class in Java : Example

```
abstract class Bike{
    abstract void run();
}

 class Honda extends Bike{
     void run(){
         System.out.println("Running safely");
     }

    public static void main(String args[]){
        Bike obj = new Honda();
        obj.run();
    }
 }
```

Here, `Bike` is an abstract class that contains only one abstract method `run()`.

Its implementation is provided by the `Honda` class.

Note:
An abstract method should be defined in it sub class.

Running safely

Abstract class in Java : Example

```
abstract class Bike{
    abstract void run();
}

 class Honda extends Bike{
     void run(){
         System.out.println("Running safely");
     }

    public static void main(String args[]){
        Bike obj = new Honda();
        obj.run();
    }
 }
```

Here, `Bike` is an abstract class that contains only one abstract method `run()`.

Its implementation is provided by the `Honda` class.

Note:
An abstract method should be defined in it sub class.

Running safely

final keyword concept

The **final** keyword in Java is used to restrict the access of an item from its super class to a sub class. The Java `final` keyword can be used in many context.

- Variable : a variable cannot be accessed in sub class
- Method : a method cannot called from a sub class object
- Class : a class cannot be sub classed.

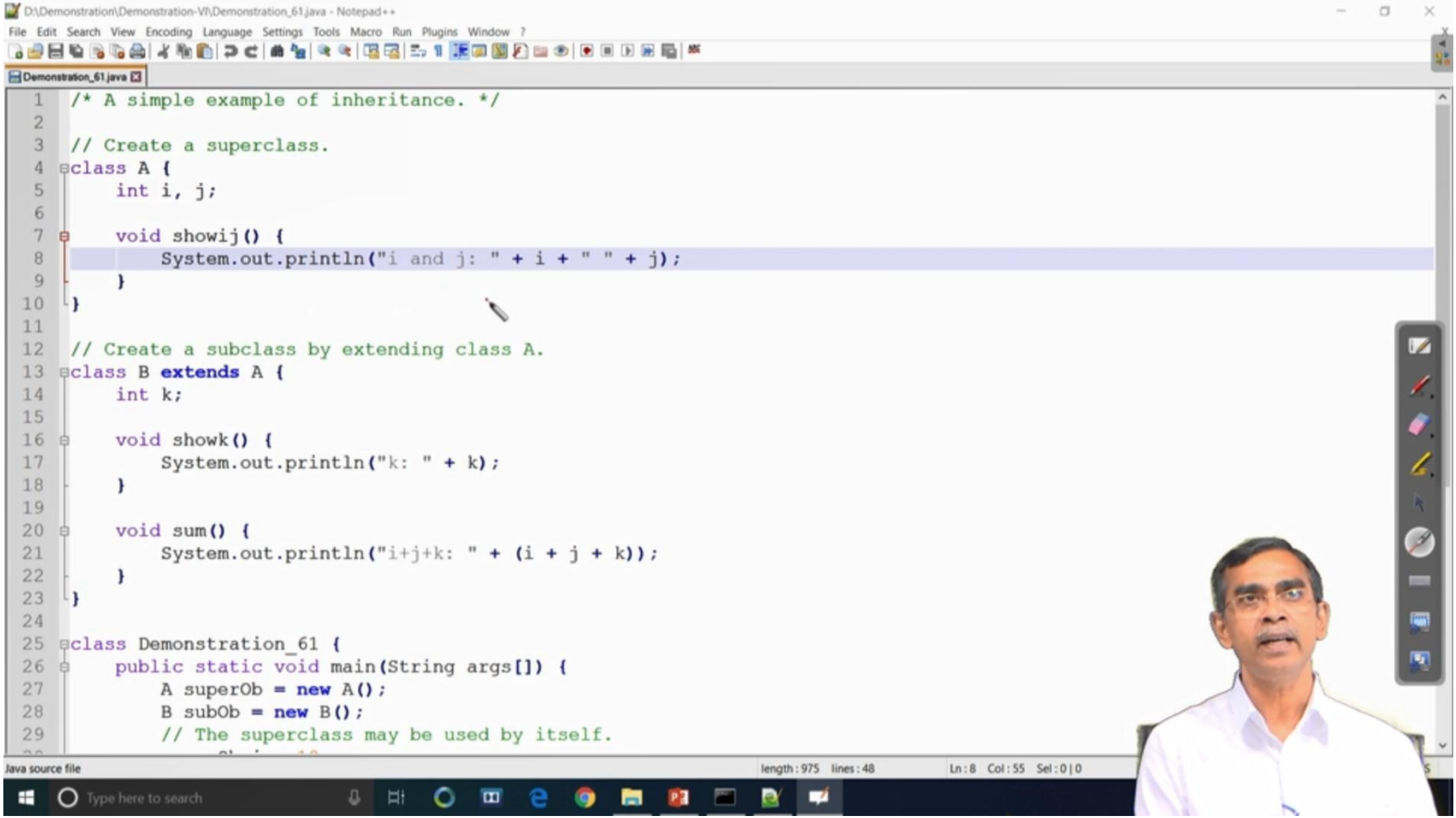
Note:
If you make any class as `final`, you cannot extend it.

Hey, I'm final!
You can not
change my value,
You cannot
override me
you can not
inherit

Final class in inheritance : An Example

```
final class Bike{}

class Honda1 extends Bike{
  void run(){
    System.out.println("Running safely with 100kmph");
  }

   public static void main(String args[]){
      Honda1 honda = new Honda1();
      honda.run();
   }
}
```

Extending a class which is declared as **final** will cause compile time error.


D:\Demonstration\Demonstration-VI\Demonstration_61.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Demonstration_61.java
/* A simple example of inheritance. */
// Create a superclass.
class A {
int i, j;
void showij() {
System.out.println("i and j: " + i + " " + j);
}
}
// Create a subclass by extending class A.
class B extends A {
int k;
void showk() {
System.out.println("k: " + k);
}
void sum() {
System.out.println("i+j+k: " + (i + j + k));
}
}
class Demonstration_61 {
public static void main(String args[]) {
A superOb = new A();
B subOb = new B();
// The superclass may be used by itself.
Java source file
length : 975 lines : 48
Ln : 8 Col : 55 Sel : 0 | 0
Type here to search

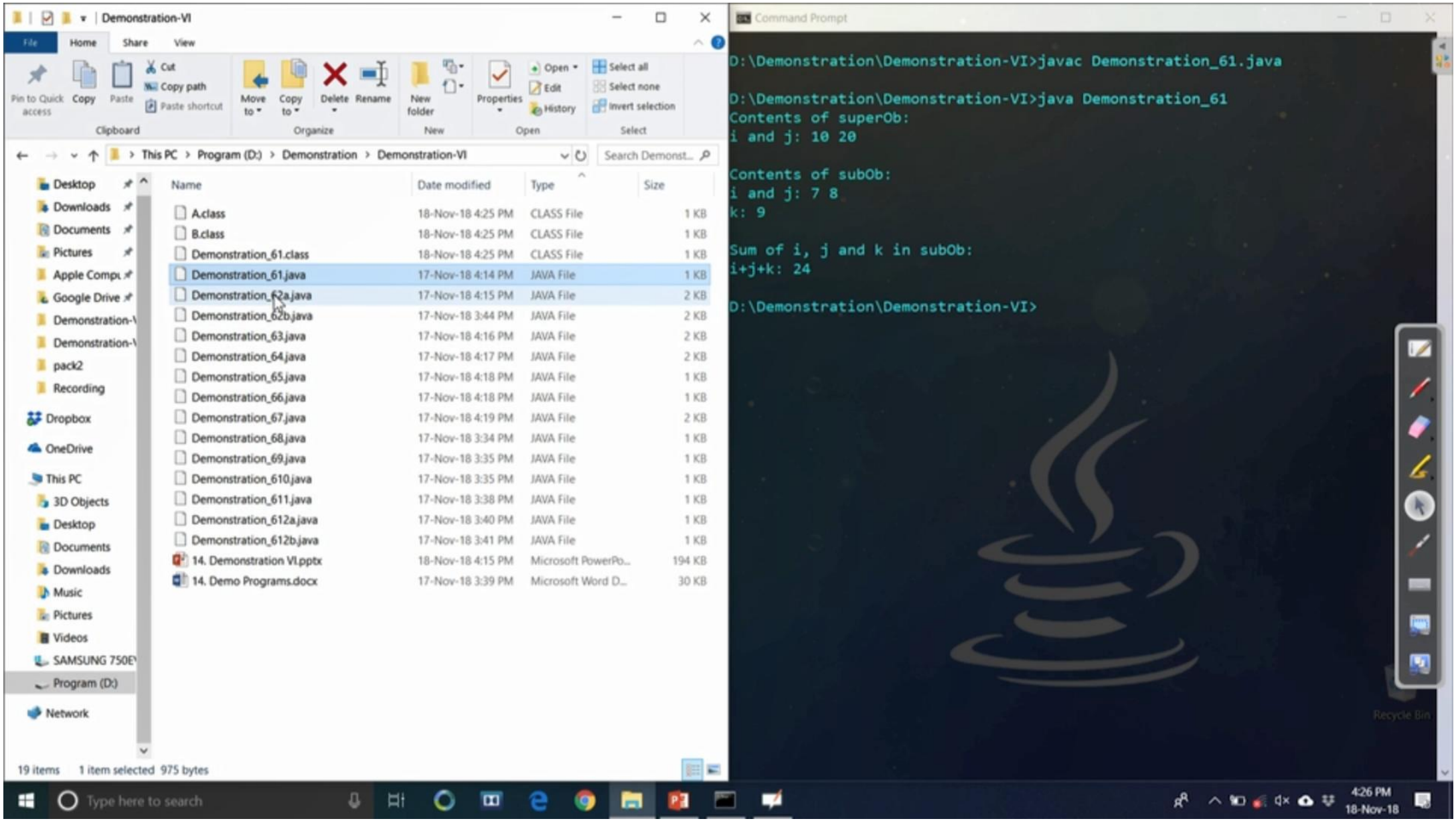
D:\Demonstration\Demonstration-VI\Demonstration_61.java - Notepad++

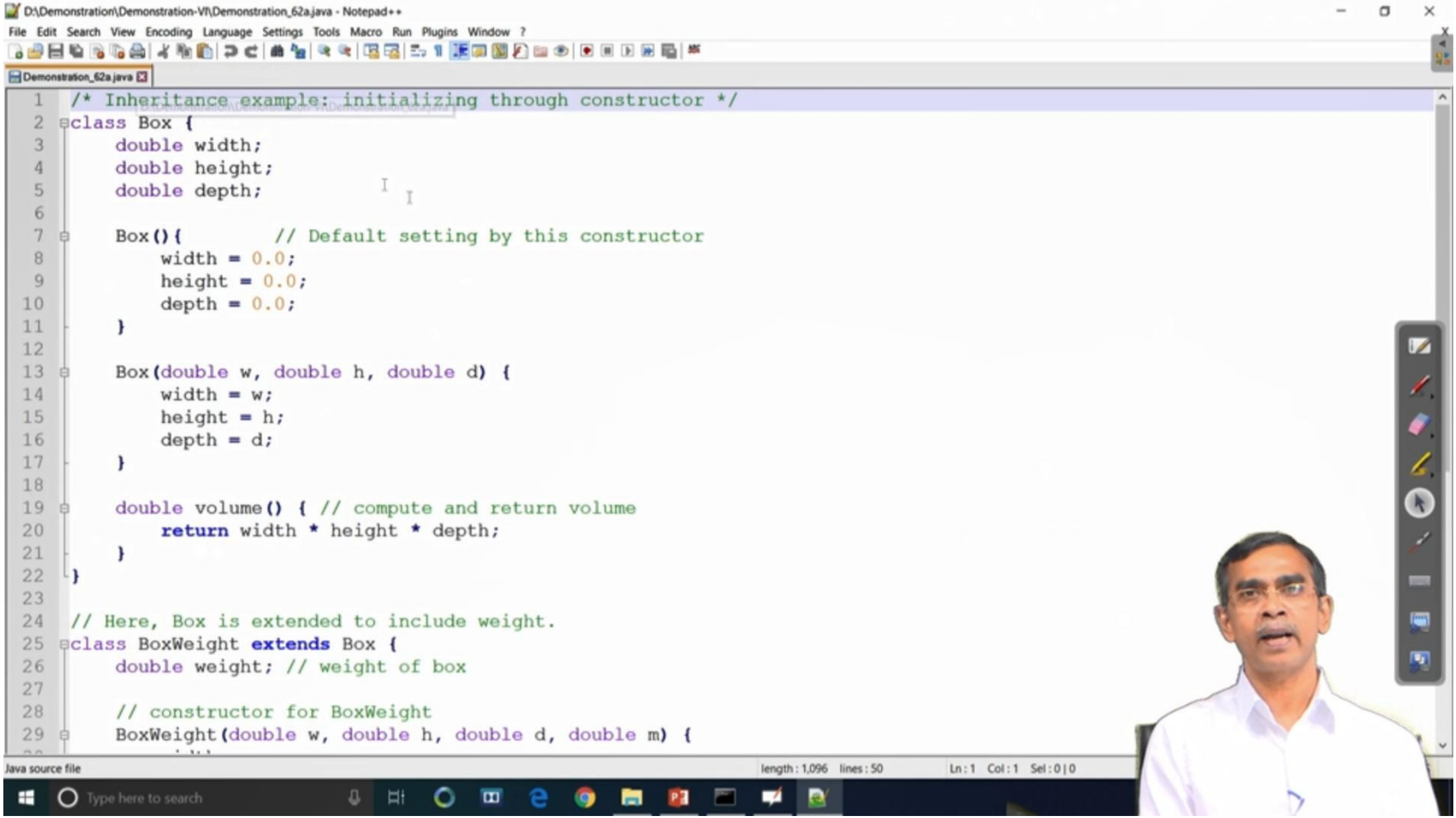
Demonstration_61.java

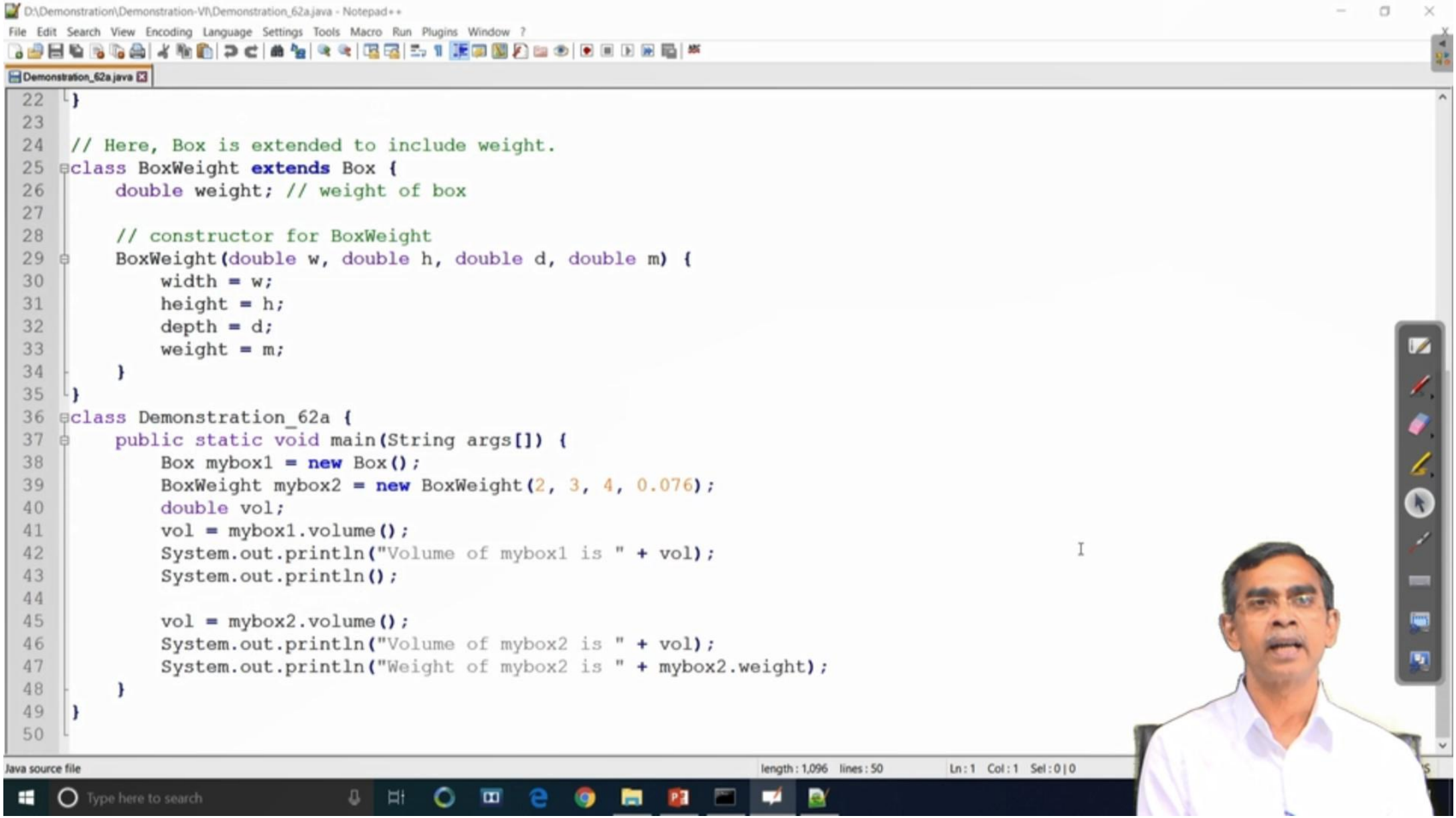
```
20      void sum() {
21          System.out.println("i+j+k: " + (i + j + k));
22      }
23  }
24
25  class Demonstration_61 {
26      public static void main(String args[]) {
27          A superOb = new A();
28          B subOb = new B();
29          // The superclass may be used by itself.
30          superOb.i = 10;
31          superOb.j = 20;
32          System.out.println("Contents of superOb: ");
33          superOb.showij();
34          System.out.println();
35
36          /* The subclass has access to all public members of its superclass. */
37          subOb.i = 7;
38          subOb.j = 8;
39          subOb.k = 9;
40          System.out.println("Contents of subOb: ");
41          subOb.showij();
42          subOb.showk();
43          System.out.println();
44          System.out.println("Sum of i, j and k in subOb: ");
45          subOb.sum();
46      }
47  }
48
```

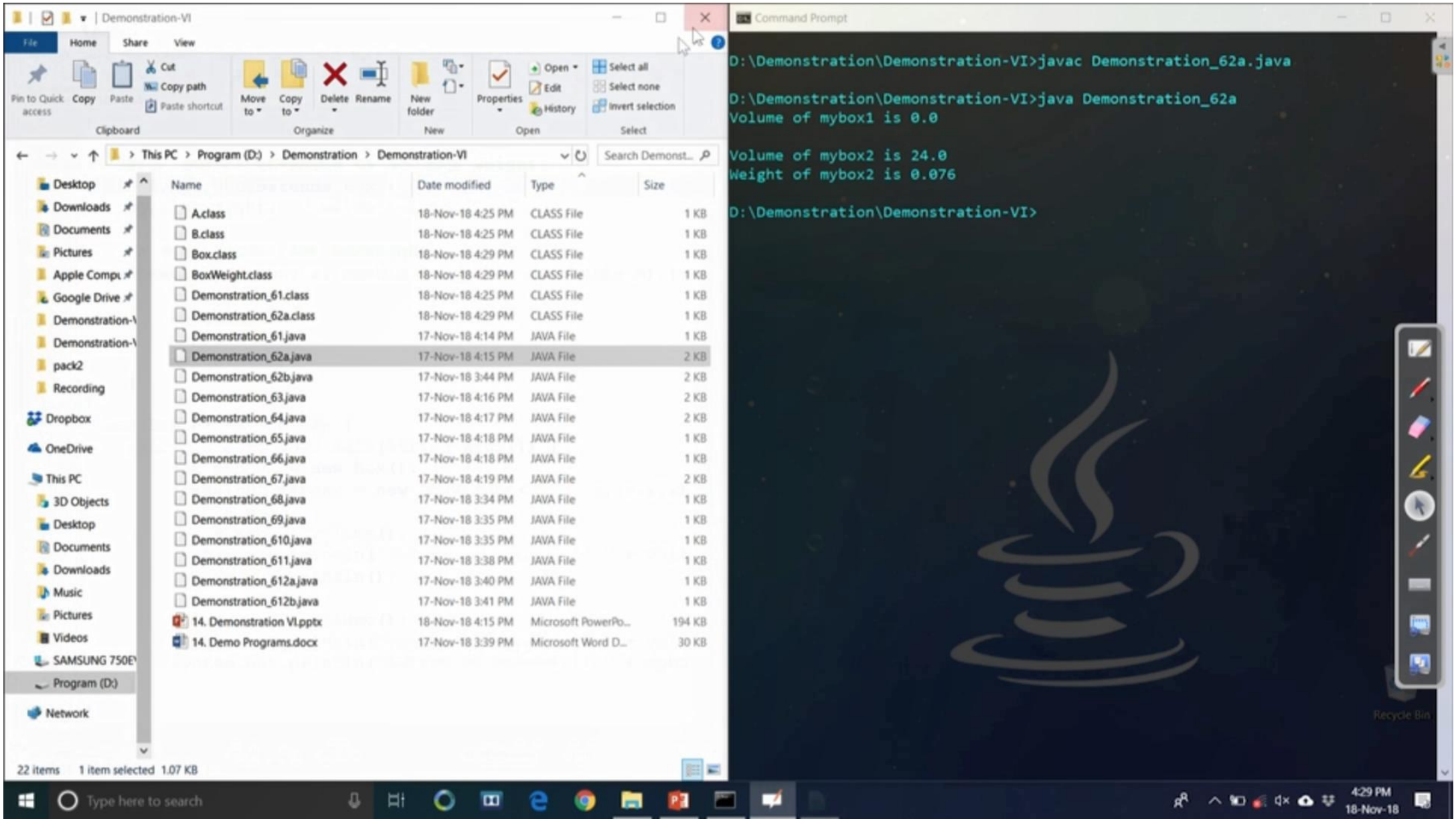
Java source file | length : 975 lines : 48 | Ln : 22 Col : 6 Sel : 0 | 0

Type here to search


Demonstration-VI
File
Home
Share
View
Pin to Quick access
Copy
Paste
Cut
Copy path
Paste shortcut
Clipboard
Move to
Copy to
Delete
Rename
Organize
New folder
New
Properties
Open
Edit
History
Open
Select all
Select none
Invert selection
Select
This PC › Program (D:) › Demonstration › Demonstration-VI
Search Demonst...
Desktop
Downloads
Documents
Pictures
Apple Compx
Google Drive
Demonstration-V
Demonstration-V
pack2
Recording
Dropbox
OneDrive
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
SAMSUNG 750E
Program (D:)
Network
Name
Date modified
Type
Size
A.class 18-Nov-18 4:25 PM CLASS File 1 KB
B.class 18-Nov-18 4:25 PM CLASS File 1 KB
Demonstration_61.class 18-Nov-18 4:25 PM CLASS File 1 KB
Demonstration_61.java 17-Nov-18 4:14 PM JAVA File 1 KB
Demonstration_62a.java 17-Nov-18 4:15 PM JAVA File 2 KB
Demonstration_62b.java 17-Nov-18 3:44 PM JAVA File 2 KB
Demonstration_63.java 17-Nov-18 4:16 PM JAVA File 2 KB
Demonstration_64.java 17-Nov-18 4:17 PM JAVA File 2 KB
Demonstration_65.java 17-Nov-18 4:18 PM JAVA File 1 KB
Demonstration_66.java 17-Nov-18 4:18 PM JAVA File 1 KB
Demonstration_67.java 17-Nov-18 4:19 PM JAVA File 2 KB
Demonstration_68.java 17-Nov-18 3:34 PM JAVA File 1 KB
Demonstration_69.java 17-Nov-18 3:35 PM JAVA File 1 KB
Demonstration_610.java 17-Nov-18 3:35 PM JAVA File 1 KB
Demonstration_611.java 17-Nov-18 3:38 PM JAVA File 1 KB
Demonstration_612a.java 17-Nov-18 3:40 PM JAVA File 1 KB
Demonstration_612b.java 17-Nov-18 3:41 PM JAVA File 1 KB
14. Demonstration VI.pptx 18-Nov-18 4:15 PM Microsoft PowerPo... 194 KB
14. Demo Programs.docx 17-Nov-18 3:39 PM Microsoft Word D... 30 KB
19 items 1 item selected 975 bytes
Command Prompt
D:\Demonstration\Demonstration-VI>javac Demonstration_61.java
D:\Demonstration\Demonstration-VI>java Demonstration_61
Contents of superOb:
i and j: 10 20
Contents of subOb:
i and j: 7 8
k: 9
Sum of i, j and k in subOb:
i+j+k: 24
D:\Demonstration\Demonstration-VI>
Recycle Bin
Type here to search
4:26 PM
18-Nov-18


D:\Demonstration\Demonstration-VI\Demonstration_62a.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Demonstration_62a.java
/* Inheritance example: initializing through constructor */
class Box {
double width;
double height;
double depth;
Box(){ // Default setting by this constructor
width = 0.0;
height = 0.0;
depth = 0.0;
}
Box(double w, double h, double d) {
width = w;
height = h;
depth = d;
}
double volume() { // compute and return volume
return width * height * depth;
}
}
// Here, Box is extended to include weight.
class BoxWeight extends Box {
double weight; // weight of box
// constructor for BoxWeight
BoxWeight(double w, double h, double d, double m) {
Java source file
length : 1,096 lines : 50
Ln : 1 Col : 1 Sel : 0 | 0
Type here to search


D:\Demonstration\Demonstration-VI\Demonstration_62a.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Demonstration_62a.java
}
// Here, Box is extended to include weight.
class BoxWeight extends Box {
double weight; // weight of box
// constructor for BoxWeight
BoxWeight(double w, double h, double d, double m) {
width = w;
height = h;
depth = d;
weight = m;
}
}
class Demonstration_62a {
public static void main(String args[]) {
Box mybox1 = new Box();
BoxWeight mybox2 = new BoxWeight(2, 3, 4, 0.076);
double vol;
vol = mybox1.volume();
System.out.println("Volume of mybox1 is " + vol);
System.out.println();
vol = mybox2.volume();
System.out.println("Volume of mybox2 is " + vol);
System.out.println("Weight of mybox2 is " + mybox2.weight);
}
}
Java source file
length : 1,096 lines : 50
Ln : 1 Col : 1 Sel : 0 | 0
Type here to search


Demonstration-VI
File
Home
Share
View
Pin to Quick access
Copy
Paste
Cut
Copy path
Paste shortcut
Clipboard
Move to
Copy to
Delete
Rename
Organize
New folder
New
Properties
Open
Edit
History
Open
Select all
Select none
Invert selection
Select
This PC › Program (D:) › Demonstration › Demonstration-VI
Search Demonst...
Desktop
Downloads
Documents
Pictures
Apple Compx
Google Drive
Demonstration-V
Demonstration-V
pack2
Recording
Dropbox
OneDrive
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
SAMSUNG 750E
Program (D:)
Network
Name
Date modified
Type
Size
A.class 18-Nov-18 4:25 PM CLASS File 1 KB
B.class 18-Nov-18 4:25 PM CLASS File 1 KB
Box.class 18-Nov-18 4:29 PM CLASS File 1 KB
BoxWeight.class 18-Nov-18 4:29 PM CLASS File 1 KB
Demonstration_61.class 18-Nov-18 4:25 PM CLASS File 1 KB
Demonstration_62a.class 18-Nov-18 4:29 PM CLASS File 1 KB
Demonstration_61.java 17-Nov-18 4:14 PM JAVA File 1 KB
Demonstration_62a.java 17-Nov-18 4:15 PM JAVA File 2 KB
Demonstration_62b.java 17-Nov-18 3:44 PM JAVA File 2 KB
Demonstration_63.java 17-Nov-18 4:16 PM JAVA File 2 KB
Demonstration_64.java 17-Nov-18 4:17 PM JAVA File 2 KB
Demonstration_65.java 17-Nov-18 4:18 PM JAVA File 1 KB
Demonstration_66.java 17-Nov-18 4:18 PM JAVA File 1 KB
Demonstration_67.java 17-Nov-18 4:19 PM JAVA File 2 KB
Demonstration_68.java 17-Nov-18 3:34 PM JAVA File 1 KB
Demonstration_69.java 17-Nov-18 3:35 PM JAVA File 1 KB
Demonstration_610.java 17-Nov-18 3:35 PM JAVA File 1 KB
Demonstration_611.java 17-Nov-18 3:38 PM JAVA File 1 KB
Demonstration_612a.java 17-Nov-18 3:40 PM JAVA File 1 KB
Demonstration_612b.java 17-Nov-18 3:41 PM JAVA File 1 KB
14. Demonstration VI.pptx 18-Nov-18 4:15 PM Microsoft PowerPo... 194 KB
14. Demo Programs.docx 17-Nov-18 3:39 PM Microsoft Word D... 30 KB
22 items 1 item selected 1.07 KB
Command Prompt
D:\Demonstration\Demonstration-VI>javac Demonstration_62a.java
D:\Demonstration\Demonstration-VI>java Demonstration_62a
Volume of mybox1 is 0.0
Volume of mybox2 is 24.0
Weight of mybox2 is 0.076
D:\Demonstration\Demonstration-VI>
Recycle Bin
Type here to search
4:29 PM
18-Nov-18

D:\Demonstration\Demonstration-VI\Demonstration_62b.java - Notepad++

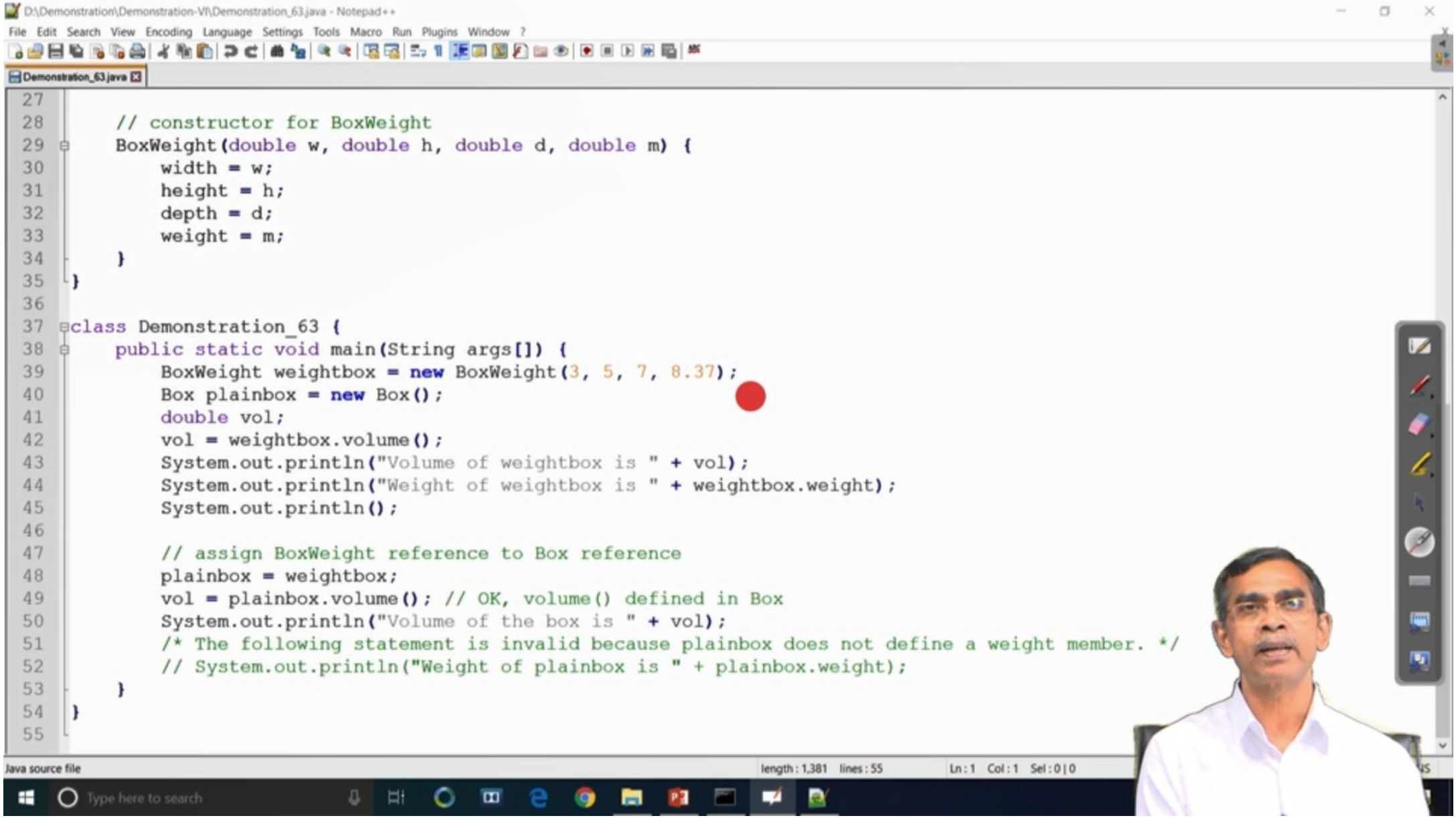
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

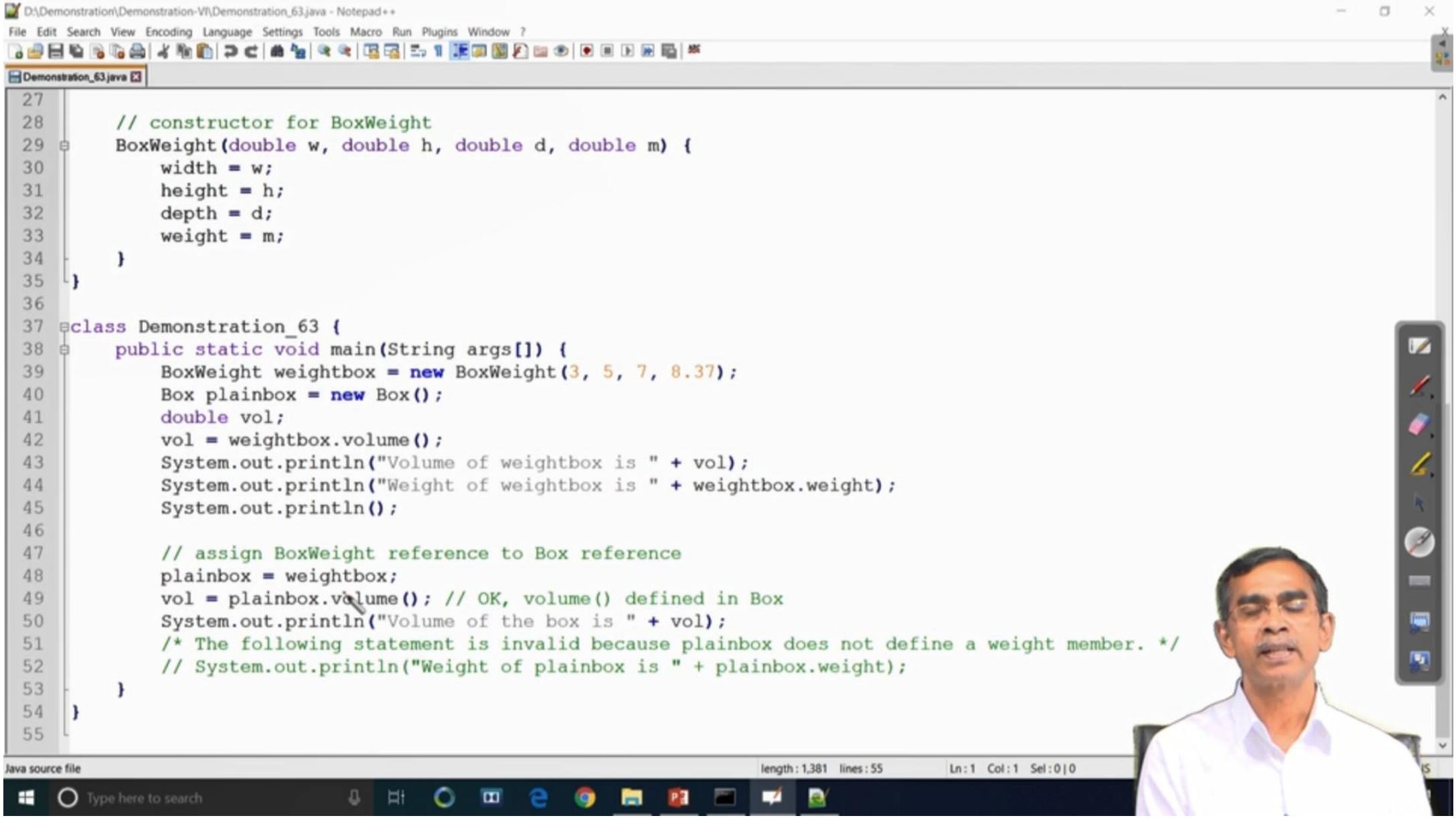
Demonstration_62b.java

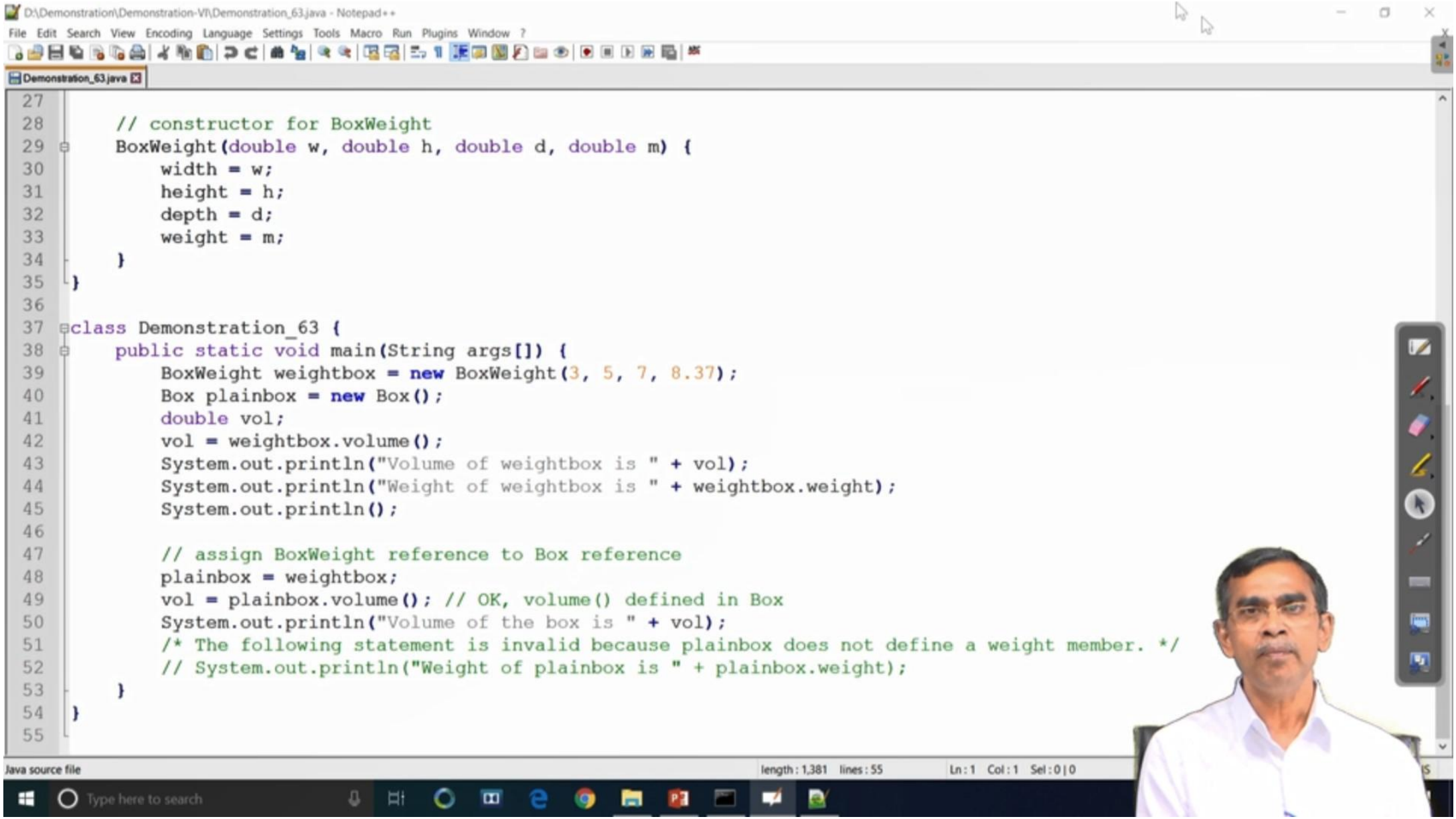
```
28      // Constructors for BoxWeight
29      BoxWeight() {       // Default constructor
30          super();            // Call the default constructor in the super class
31          weight = 0.0;
32      }
33
34
35      BoxWeight(double w, double h, double d, double m) {
36          super(w, h, d);  // Call the overloaded constructor in the super class
37          weight = m;
38      }
39  }
40
41  class Demonstration_62b {
42      public static void main(String args[]) {
43          Box mybox1 = new Box(10, 20, 15);
44          BoxWeight mybox2 = new BoxWeight(2, 3, 4, 0.076);
45          double vol;
46          vol = mybox1.volume();
47          System.out.println("Volume of mybox1 is " + vol);
48          // System.out.println("Weight of mybox1 is " + mybox1.weight); ERROR!
49          System.out.println();
50
51          vol = mybox2.volume();
52          System.out.println("Volume of mybox2 is " + vol);
53          System.out.println("Weight of mybox2 is " + mybox2.weight);
54      }
55  }
56
```

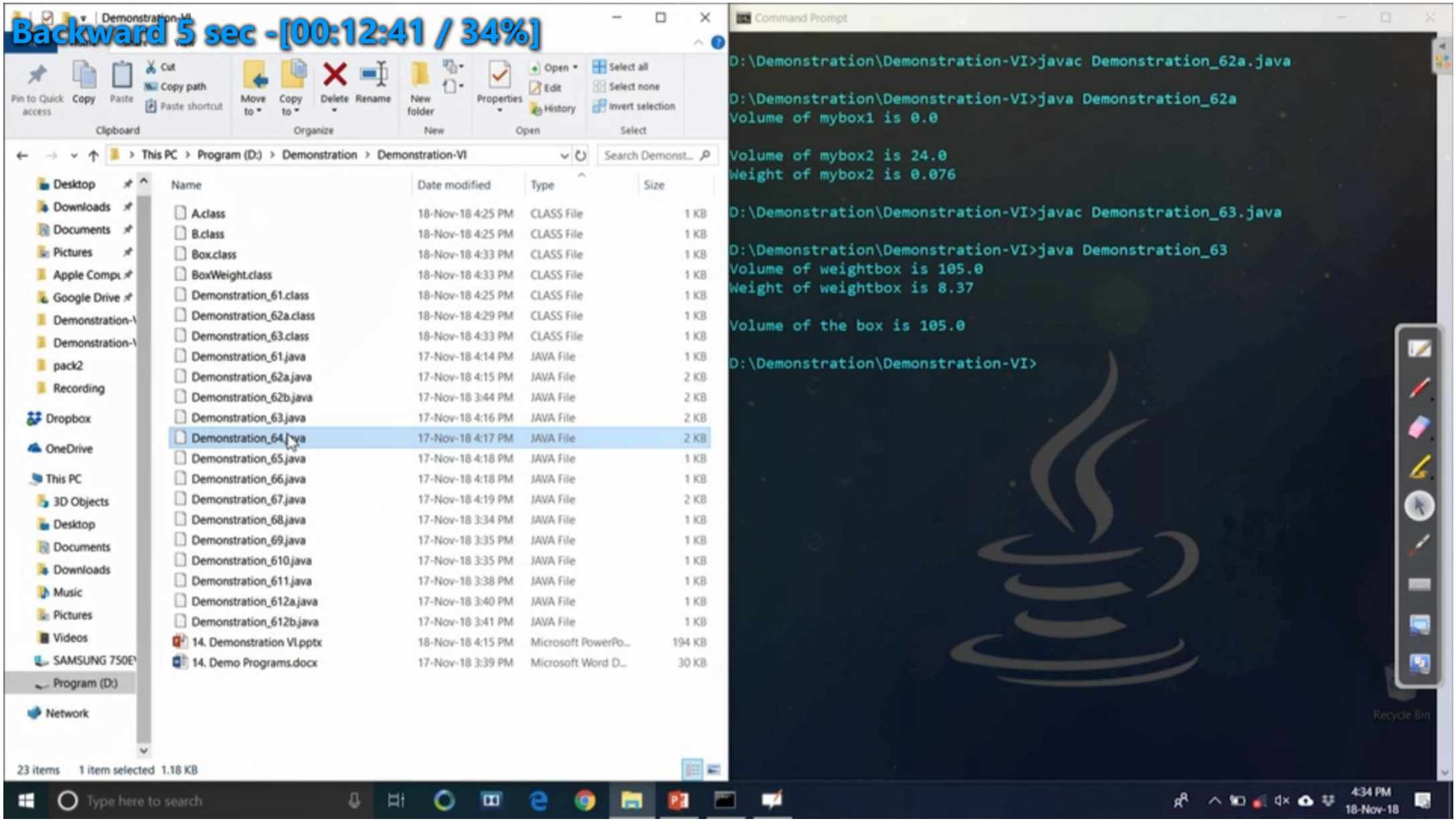
Java source file | length : 1,358 lines : 56 | Ln : 16 Col : 19 Sel : 0 | 0

Type here to search


D:\Demonstration\Demonstration-VI\Demonstration_63.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Demonstration_63.java
27
28 // constructor for BoxWeight
29 BoxWeight(double w, double h, double d, double m) {
30 width = w;
31 height = h;
32 depth = d;
33 weight = m;
34 }
35 }
36
37 class Demonstration_63 {
38 public static void main(String args[]) {
39 BoxWeight weightbox = new BoxWeight(3, 5, 7, 8.37);
40 Box plainbox = new Box();
41 double vol;
42 vol = weightbox.volume();
43 System.out.println("Volume of weightbox is " + vol);
44 System.out.println("Weight of weightbox is " + weightbox.weight);
45 System.out.println();
46
47 // assign BoxWeight reference to Box reference
48 plainbox = weightbox;
49 vol = plainbox.volume(); // OK, volume() defined in Box
50 System.out.println("Volume of the box is " + vol);
51 /* The following statement is invalid because plainbox does not define a weight member. */
52 // System.out.println("Weight of plainbox is " + plainbox.weight);
53 }
54 }
55
Java source file
length : 1,381 lines : 55
Ln : 1 Col : 1 Sel : 0 | 0
Type here to search


D:\Demonstration\Demonstration-VI\Demonstration_63.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Demonstration_63.java
// constructor for BoxWeight
BoxWeight(double w, double h, double d, double m) {
width = w;
height = h;
depth = d;
weight = m;
}
}
class Demonstration_63 {
public static void main(String args[]) {
BoxWeight weightbox = new BoxWeight(3, 5, 7, 8.37);
Box plainbox = new Box();
double vol;
vol = weightbox.volume();
System.out.println("Volume of weightbox is " + vol);
System.out.println("Weight of weightbox is " + weightbox.weight);
System.out.println();
// assign BoxWeight reference to Box reference
plainbox = weightbox;
vol = plainbox.volume(); // OK, volume() defined in Box
System.out.println("Volume of the box is " + vol);
/* The following statement is invalid because plainbox does not define a weight member. */
// System.out.println("Weight of plainbox is " + plainbox.weight);
}
}
Java source file
length : 1,381 lines : 55
Ln : 1 Col : 1 Sel : 0 | 0
Type here to search


D:\Demonstration\Demonstration-VI\Demonstration_63.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Demonstration_63.java
27
28 // constructor for BoxWeight
29 BoxWeight(double w, double h, double d, double m) {
30 width = w;
31 height = h;
32 depth = d;
33 weight = m;
34 }
35 }
36
37 class Demonstration_63 {
38 public static void main(String args[]) {
39 BoxWeight weightbox = new BoxWeight(3, 5, 7, 8.37);
40 Box plainbox = new Box();
41 double vol;
42 vol = weightbox.volume();
43 System.out.println("Volume of weightbox is " + vol);
44 System.out.println("Weight of weightbox is " + weightbox.weight);
45 System.out.println();
46
47 // assign BoxWeight reference to Box reference
48 plainbox = weightbox;
49 vol = plainbox.volume(); // OK, volume() defined in Box
50 System.out.println("Volume of the box is " + vol);
51 /* The following statement is invalid because plainbox does not define a weight member. */
52 // System.out.println("Weight of plainbox is " + plainbox.weight);
53 }
54 }
55
Java source file
length : 1,381 lines : 55
Ln : 1 Col : 1 Sel : 0 | 0
Type here to search


Backward 5 sec -[00:12:41 / 34%]
Demonstration-VI
Pin to Quick access
Copy
Paste
Cut
Copy path
Paste shortcut
Clipboard
Move to
Copy to
Delete
Rename
Organize
New folder
New
Properties
Open
Edit
History
Open
Select all
Select none
Invert selection
Select
This PC › Program (D:) › Demonstration › Demonstration-VI
Search Demonst...
Desktop
Downloads
Documents
Pictures
Apple Compx
Google Drive
Demonstration-\
Demonstration-\
pack2
Recording
Dropbox
OneDrive
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
SAMSUNG 750E\
Program (D:)
Network
Name
Date modified
Type
Size
A.class 18-Nov-18 4:25 PM CLASS File 1 KB
B.class 18-Nov-18 4:25 PM CLASS File 1 KB
Box.class 18-Nov-18 4:33 PM CLASS File 1 KB
BoxWeight.class 18-Nov-18 4:33 PM CLASS File 1 KB
Demonstration_61.class 18-Nov-18 4:25 PM CLASS File 1 KB
Demonstration_62a.class 18-Nov-18 4:29 PM CLASS File 1 KB
Demonstration_63.class 18-Nov-18 4:33 PM CLASS File 1 KB
Demonstration_61.java 17-Nov-18 4:14 PM JAVA File 1 KB
Demonstration_62a.java 17-Nov-18 4:15 PM JAVA File 2 KB
Demonstration_62b.java 17-Nov-18 3:44 PM JAVA File 2 KB
Demonstration_63.java 17-Nov-18 4:16 PM JAVA File 2 KB
Demonstration_64.java 17-Nov-18 4:17 PM JAVA File 2 KB
Demonstration_65.java 17-Nov-18 4:18 PM JAVA File 1 KB
Demonstration_66.java 17-Nov-18 4:18 PM JAVA File 1 KB
Demonstration_67.java 17-Nov-18 4:19 PM JAVA File 2 KB
Demonstration_68.java 17-Nov-18 3:34 PM JAVA File 1 KB
Demonstration_69.java 17-Nov-18 3:35 PM JAVA File 1 KB
Demonstration_610.java 17-Nov-18 3:35 PM JAVA File 1 KB
Demonstration_611.java 17-Nov-18 3:38 PM JAVA File 1 KB
Demonstration_612a.java 17-Nov-18 3:40 PM JAVA File 1 KB
Demonstration_612b.java 17-Nov-18 3:41 PM JAVA File 1 KB
14. Demonstration VI.pptx 18-Nov-18 4:15 PM Microsoft PowerPo... 194 KB
14. Demo Programs.docx 17-Nov-18 3:39 PM Microsoft Word D... 30 KB
23 items 1 item selected 1.18 KB
Command Prompt
D:\Demonstration\Demonstration-VI>javac Demonstration_62a.java
D:\Demonstration\Demonstration-VI>java Demonstration_62a
Volume of mybox1 is 0.0
Volume of mybox2 is 24.0
Weight of mybox2 is 0.076
D:\Demonstration\Demonstration-VI>javac Demonstration_63.java
D:\Demonstration\Demonstration-VI>java Demonstration_63
Volume of weightbox is 105.0
Weight of weightbox is 8.37
Volume of the box is 105.0
D:\Demonstration\Demonstration-VI>
Recycle Bin
Type here to search
4:34 PM
18-Nov-18

Play

D:\Demonstration\Demonstration-VI\Demonstration_65.java - Notepad++

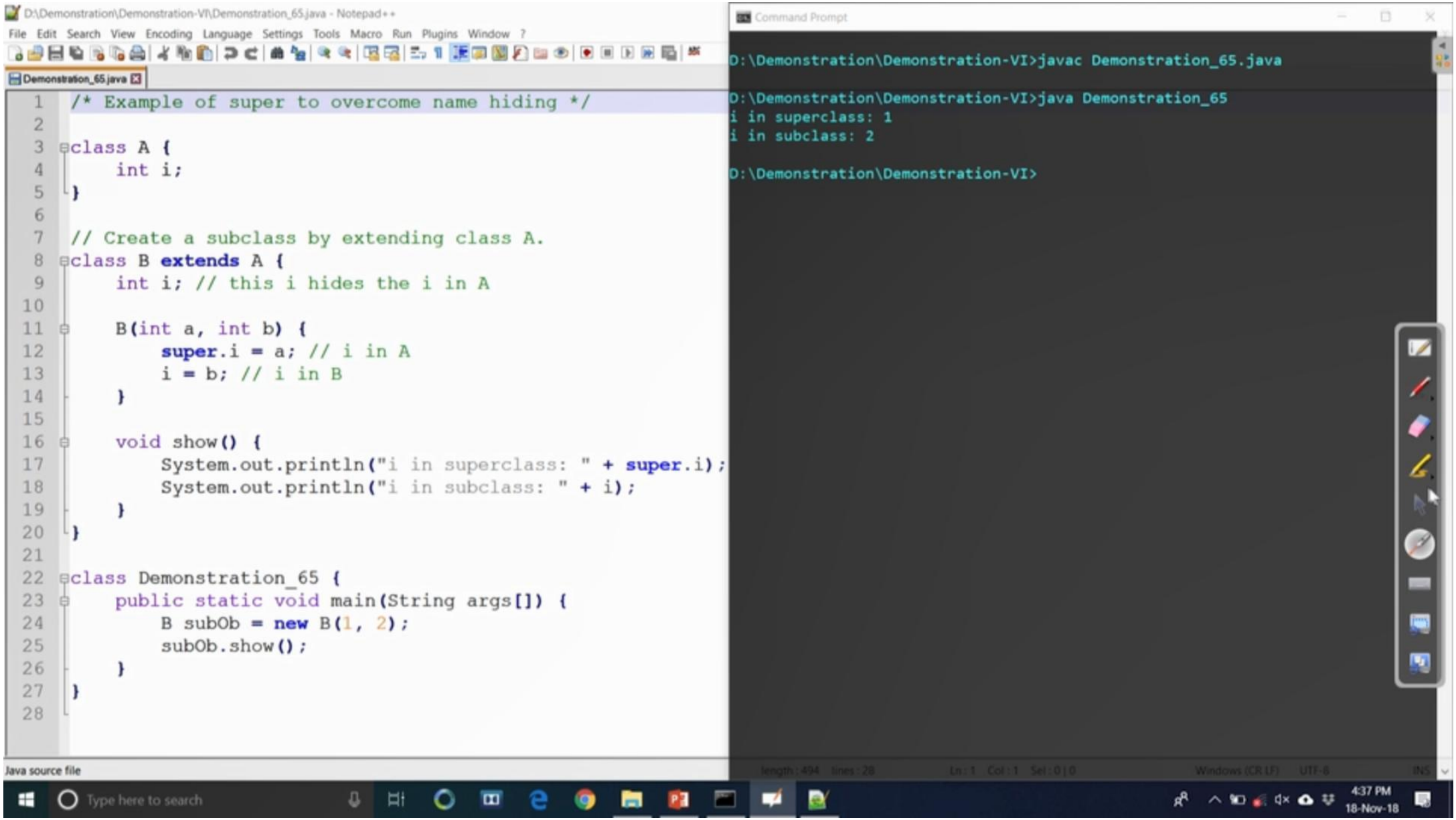
File Search View Encoding Language Settings Tools Macro Run Plugins Window ?

Demonstration_65.java

```
1  /* Example of super to overcome name hiding */
2
3  class A {
4      int i;
5  }
6
7  // Create a subclass by extending class A.
8  class B extends A {
9      int i; // this i hides the i in A
10
11     B(int a, int b) {
12         super.i = a; // i in A
13         i = b; // i in B
14     }
15
16     void show() {
17         System.out.println("i in superclass: " + super.i);
18         System.out.println("i in subclass: " + i);
19     }
20 }
21
22 class Demonstration_65 {
23     public static void main(String args[]) {
24         B subOb = new B(1, 2);
25         subOb.show();
26     }
27 }
28
```

Java source file | length : 494 lines : 28 | Ln : 1 Col : 1 Sel : 0 | 0

Type here to search


D:\Demonstration\Demonstration-VI\Demonstration_65.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Demonstration_65.java
/* Example of super to overcome name hiding */
class A {
int i;
}
// Create a subclass by extending class A.
class B extends A {
int i; // this i hides the i in A
B(int a, int b) {
super.i = a; // i in A
i = b; // i in B
}
void show() {
System.out.println("i in superclass: " + super.i);
System.out.println("i in subclass: " + i);
}
}
class Demonstration_65 {
public static void main(String args[]) {
B subOb = new B(1, 2);
subOb.show();
}
}
Java source file
Command Prompt
D:\Demonstration\Demonstration-VI>javac Demonstration_65.java
D:\Demonstration\Demonstration-VI>java Demonstration_65
i in superclass: 1
i in subclass: 2
D:\Demonstration\Demonstration-VI>
Type here to search
4:37 PM
18-Nov-18

D:\Demonstration\Demonstration-VI\Demonstration_66.java - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

Demonstration_66.java

```
1  /* Code sharing through super concept */
2
3  class Cat {
4      void speak() {
5          System.out.println("Meaon ! ");
6      }
7  }
8
9  class PetCat extends Cat {      // PetCat is one type of Cat
10     void speak() {
11         System.out.println(" Meow ! ");
12     }
13 }
14
15 class MagicCat extends Cat { // MagicCat is another kind of Cat
16     static boolean noOne;
17
18     void speak() {
19         if (noOne) {
20             super.speak(); // use the super class definition
21         } else {
22             System.out.println(" Hello World !");
23         }
24     }
25 }
26
27 class Demonstration_66 {
28     public static void main(String args[]) {
29         PetCat c1 = new PetCat();
```

Java source file | length : 706 lines : 38 | Ln : 1 Col : 1 Sel : 0 | 0

Type here to search

Forward 5 sec - [00:17:50 / 48%]

D:\Demonstration\Demonstration-VI\Demonstration_66.java - Notepad++

Demonstration_66.java

```
10      void speak() {
11          System.out.println(" Meow ! ");
12      }
13  }
14
15  class MagicCat extends Cat { // MagicCat is another kind of Cat
16      static boolean noOne;
17
18      void speak() {
19          if (noOne) {
20              super.speak(); // use the super class definition
21          } else {
22              System.out.println(" Hello World !");
23          }
24      }
25  }
26
27  class Demonstration_66 {
28      public static void main(String args[]) {
29          PetCat c1 = new PetCat();
30          MagicCat c2 = new MagicCat();
31          c2.noOne = true;
32          c2.speak();
33          c1.speak();
34          c2.noOne = false;
35          c2.speak();
36      }
37  }
38
```

Java source file | length : 706 lines : 38 | Ln : 1 Col : 1 Sel : 0 | 0

Type here to search

Backward 5 sec -[00:19:46 / 53%]

D:\Demonstration\Demonstration-VI\Demonstration_67.java - Notepad++

Demonstration_67.java

```java
/* Example of multilevel inheritance. */

// Start with Box.
class Box {
    private double width;
    private double height;
    private double depth;

    // constructor used when all dimensions specified
    Box(double w, double h, double d) {
        width = w;
        height = h;
        depth = d;
    }

    // compute and return volume
    double volume() {
        return width * height * depth;
    }
}

// Add weight.
class BoxWeight extends Box {
    double weight; // weight of box
    // constructor when all parameters are specified
    BoxWeight(double w, double h, double d, double m) {
        super(w, h, d); // call superclass constructor
        weight = m;
    }
```

Java source file | length : 1,575 lines : 60 | Ln : 1 Col : 1 Sel : 0 | 0

Type here to search

D:\Demonstration\Demonstration-VI\Demonstration_67.java - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

Demonstration_67.java

```
13          depth = d;
14      }
15
16      // compute and return volume
17      double volume() {
18          return width * height * depth;
19      }
20  }
21
22  // Add weight.
23  class BoxWeight extends Box {
24      double weight; // weight of box
25      // constructor when all parameters are specified
26      BoxWeight(double w, double h, double d, double m) {
27          super(w, h, d); // call superclass constructor
28          weight = m;
29      }
30  }
31
32  // Add shipping costs
33  class Shipment extends BoxWeight {
34      double cost;
35
36      // constructor when all parameters are specified
37      Shipment(double w, double h, double d, double m, double c) {
38          super(w, h, d, m); // call superclass constructor
39          cost = c;
40      }
41  }
```

Java source file | length : 1,575 lines : 60 | Ln : 1 Col : 1 Sel : 0 | 0

Type here to search

Backward 5 sec -[00:20:37 / 55%]

Demonstration_67.java

```
16      // compute and return volume
17      double volume() {
18          return width * height * depth;
19      }
20  }
21
22  // Add weight.
23  class BoxWeight extends Box {
24      double weight; // weight of box
25      // constructor when all parameters are specified
26      BoxWeight(double w, double h, double d, double m) {
27          super(w, h, d); // call superclass constructor
28          weight = m;
29      }
30  }
31
32  // Add shipping costs
33  class Shipment extends BoxWeight {
34      double cost;
35
36      // constructor when all parameters are specified
37      Shipment(double w, double h, double d, double m, double c) {
38          super(w, h, d, m); // call superclass constructor
39          cost = c;
40      }
41  }
42
43
44  class Demonstration_67 {
```

Java source file | length : 1,575 lines : 60 | Ln : 1 Col : 1 Sel : 0 | 0

Type here to search

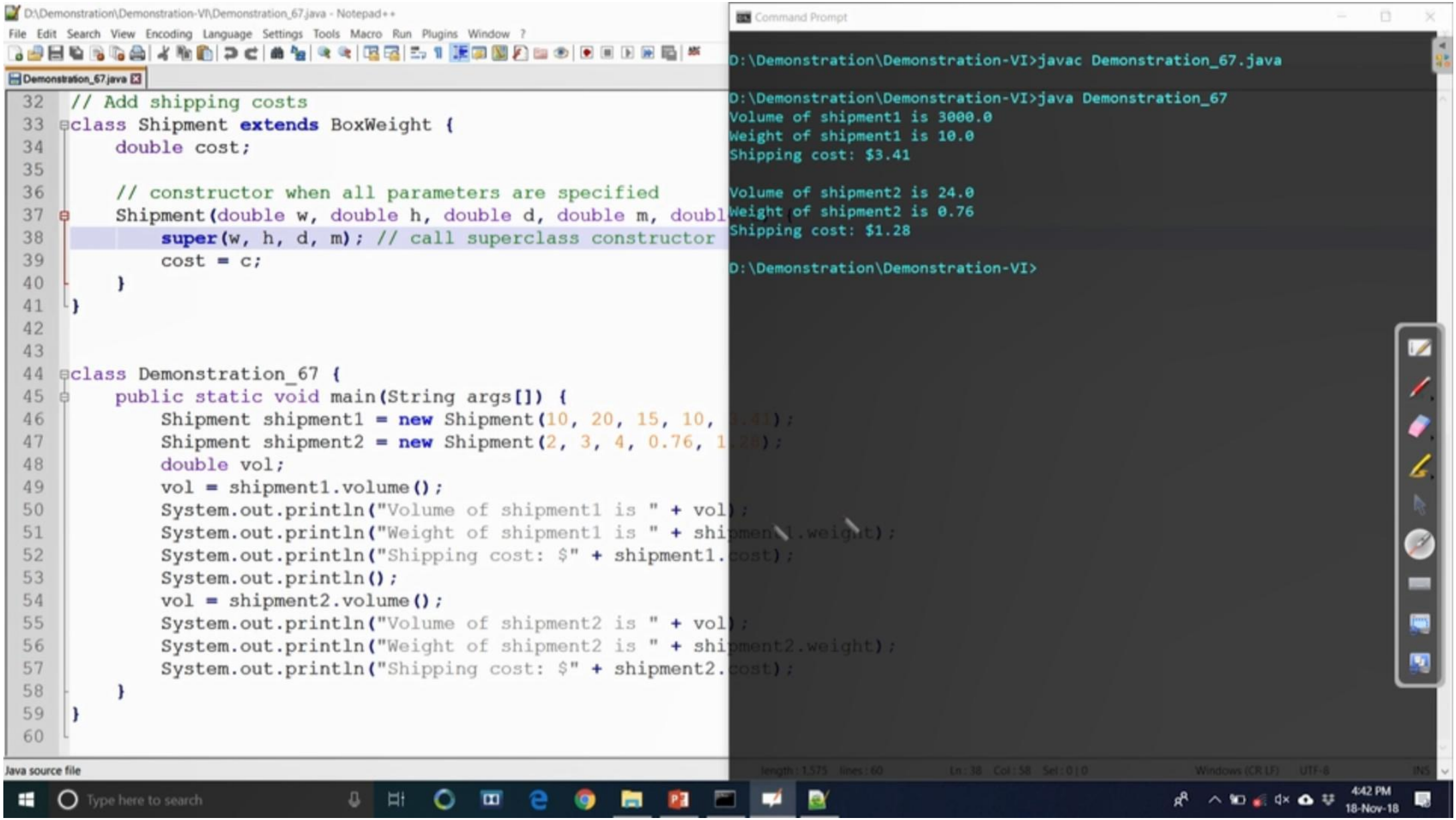
Backward 5 sec -[00:20:58 / 56%]

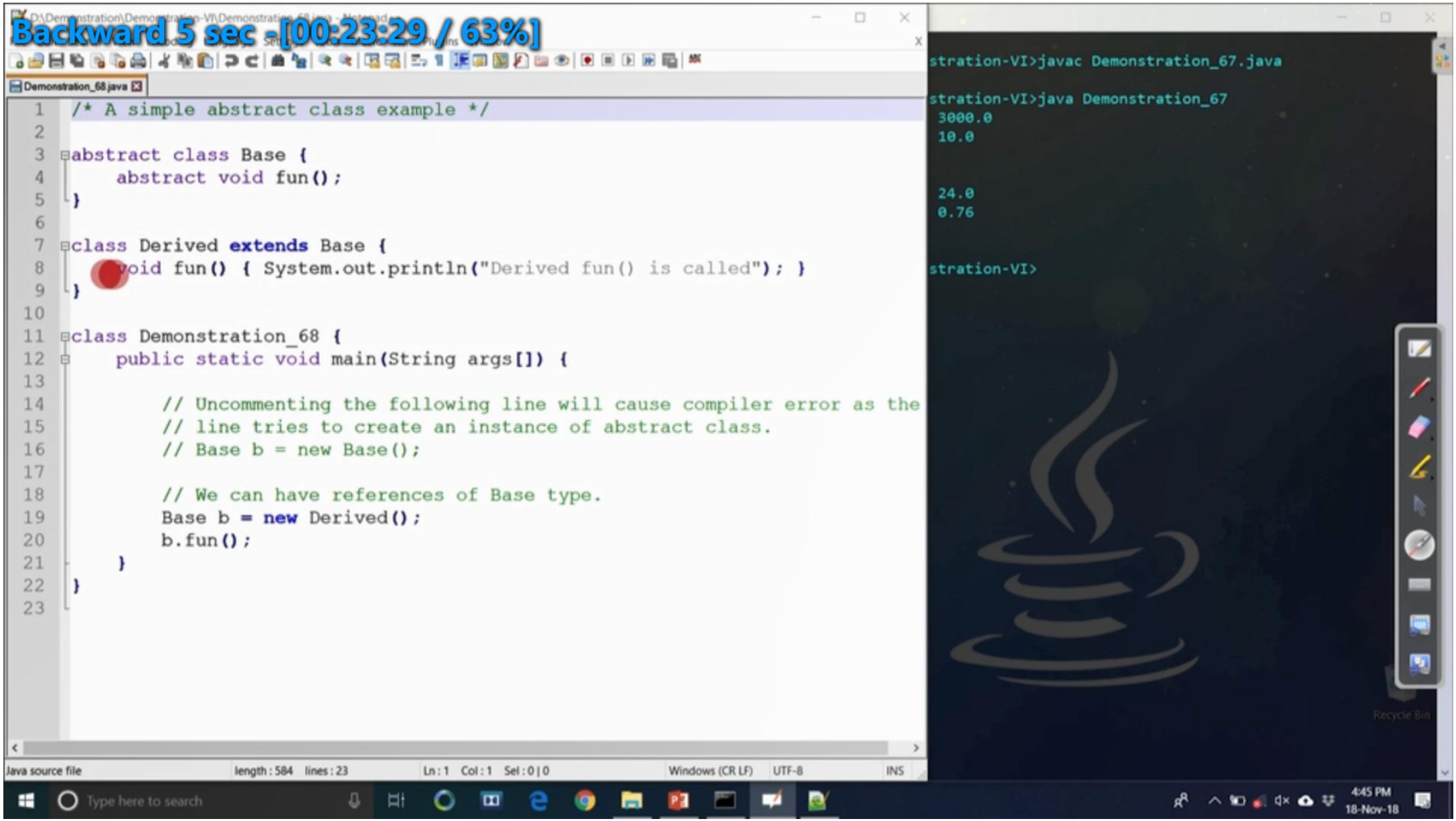
Demonstration_67.java

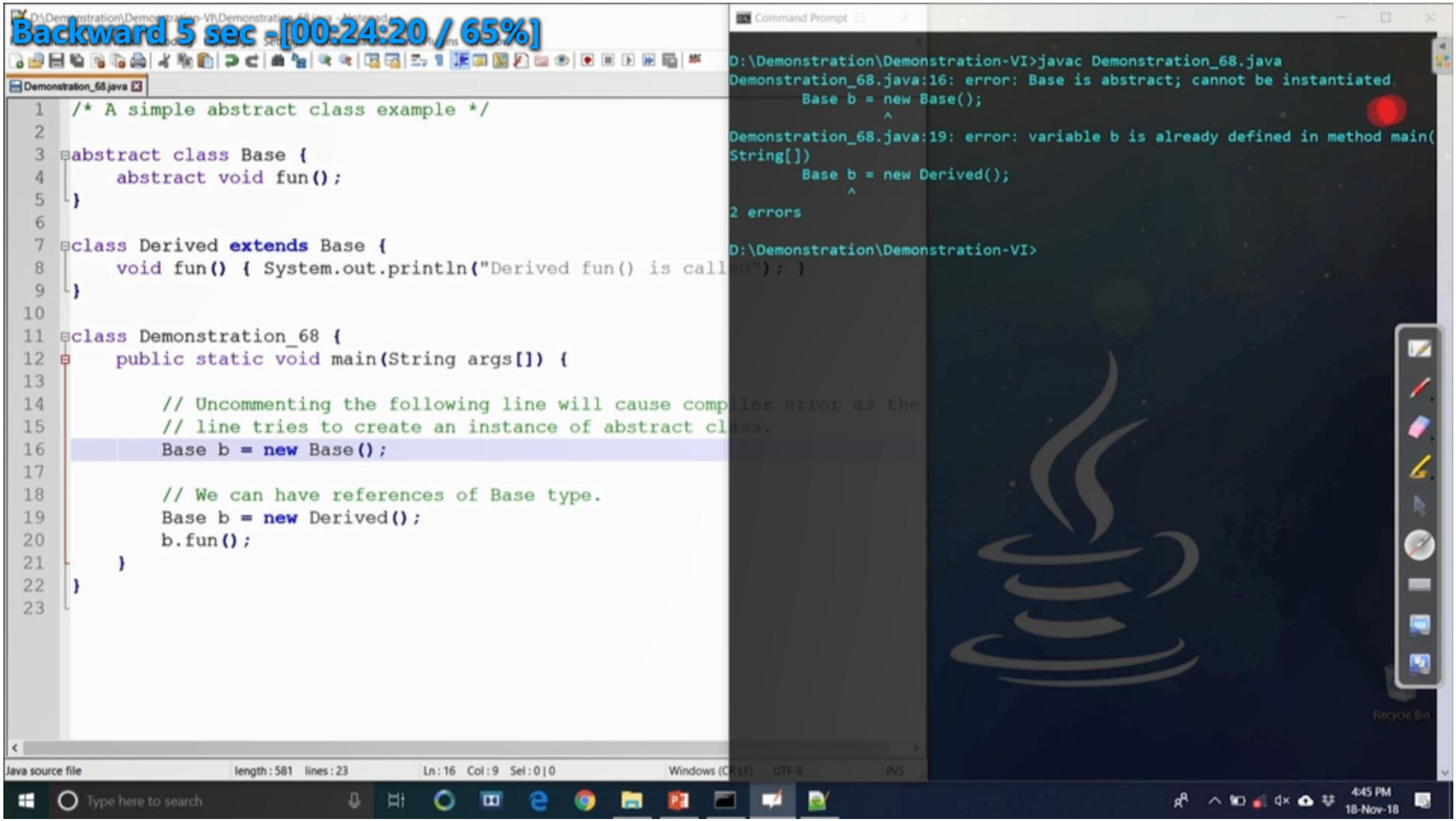
```
// Add shipping costs
class Shipment extends BoxWeight {
    double cost;

    // constructor when all parameters are specified
    Shipment(double w, double h, double d, double m, double c) {
        super(w, h, d, m); // call superclass constructor
        cost = c;
    }
}


class Demonstration_67 {
    public static void main(String args[]) {
        Shipment shipment1 = new Shipment(10, 20, 15, 10, 3.41);
        Shipment shipment2 = new Shipment(2, 3, 4, 0.76, 1.28);
        double vol;
        vol = shipment1.volume();
        System.out.println("Volume of shipment1 is " + vol);
        System.out.println("Weight of shipment1 is " + shipment1.weight);
        System.out.println("Shipping cost: $" + shipment1.cost);
        System.out.println();
        vol = shipment2.volume();
        System.out.println("Volume of shipment2 is " + vol);
        System.out.println("Weight of shipment2 is " + shipment2.weight);
        System.out.println("Shipping cost: $" + shipment2.cost);
    }
}

```

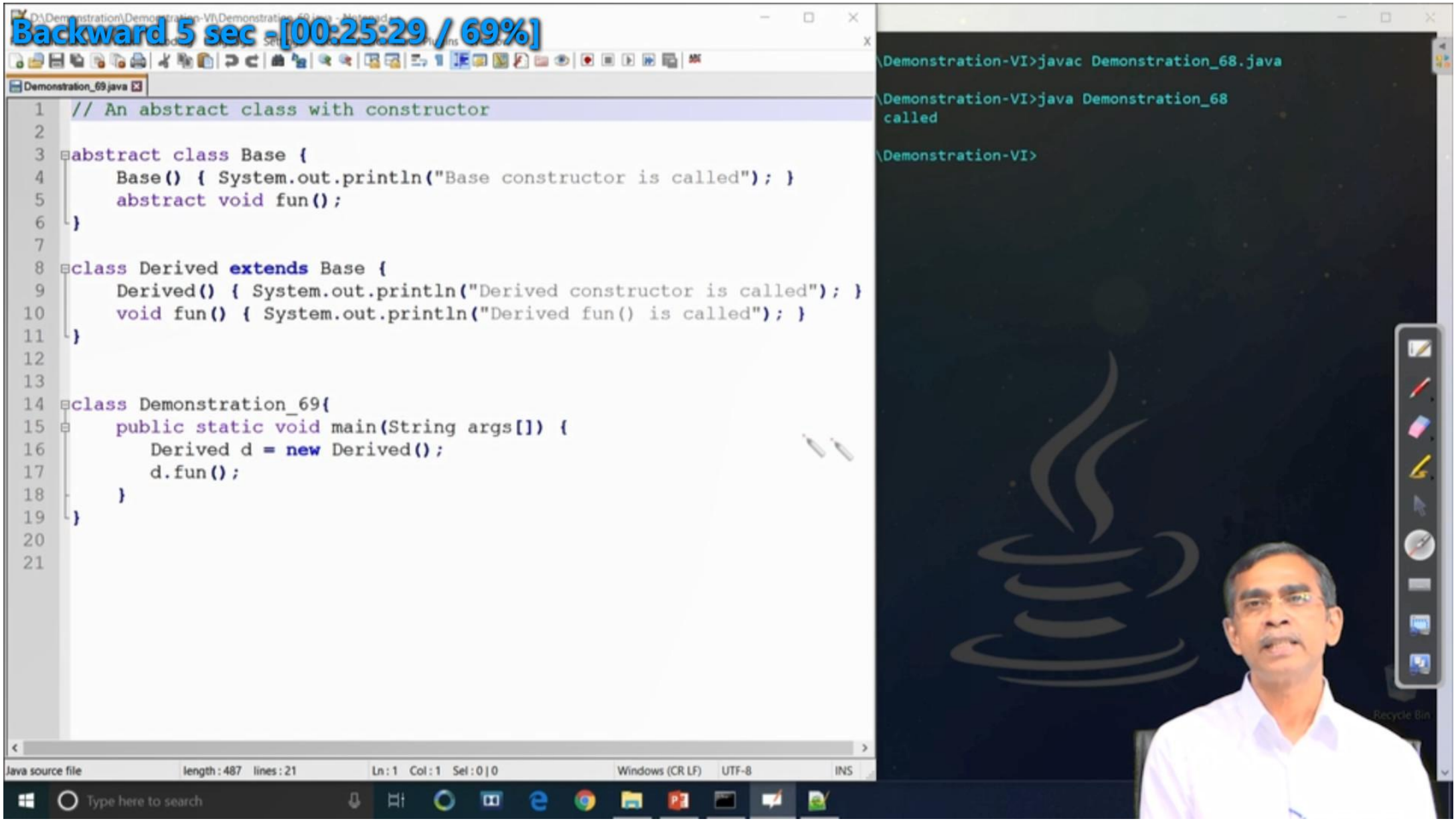
Java source file | length : 1,575 lines : 60 | Ln : 38 Col : 58 Sel : 0 | 0

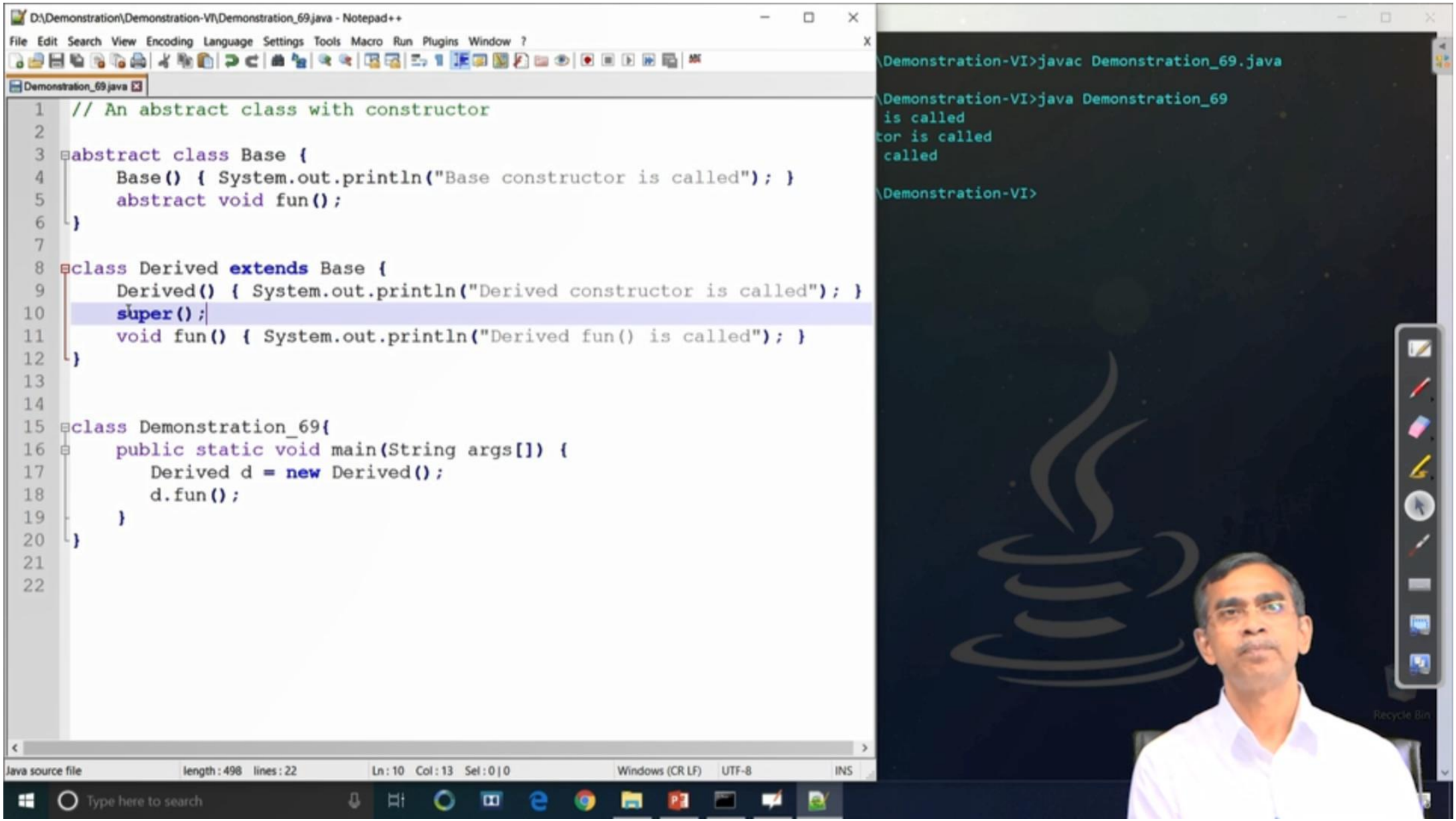
Type here to search

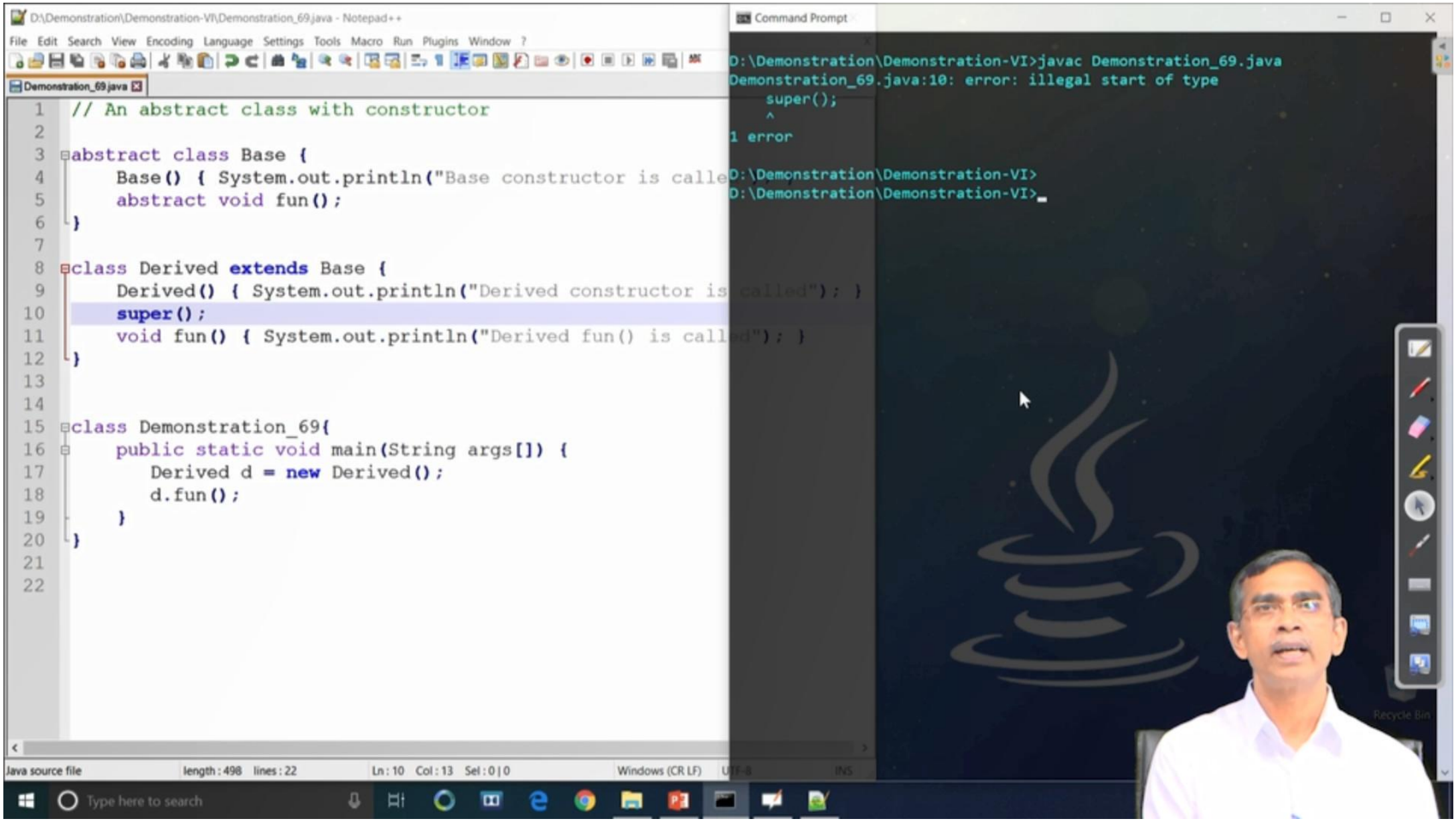

D:\Demonstration\Demonstration-VI\Demonstration_67.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Demonstration_67.java
// Add shipping costs
class Shipment extends BoxWeight {
double cost;
// constructor when all parameters are specified
Shipment(double w, double h, double d, double m, doubl
super(w, h, d, m); // call superclass constructor
cost = c;
}
}
class Demonstration_67 {
public static void main(String args[]) {
Shipment shipment1 = new Shipment(10, 20, 15, 10,
Shipment shipment2 = new Shipment(2, 3, 4, 0.76, 1
double vol;
vol = shipment1.volume();
System.out.println("Volume of shipment1 is " + vol);
System.out.println("Weight of shipment1 is " + shipment1.weight);
System.out.println("Shipping cost: $" + shipment1.cost);
System.out.println();
vol = shipment2.volume();
System.out.println("Volume of shipment2 is " + vol);
System.out.println("Weight of shipment2 is " + shipment2.weight);
System.out.println("Shipping cost: $" + shipment2.cost);
}
}
Java source file
Command Prompt
D:\Demonstration\Demonstration-VI>javac Demonstration_67.java
D:\Demonstration\Demonstration-VI>java Demonstration_67
Volume of shipment1 is 3000.0
Weight of shipment1 is 10.0
Shipping cost: $3.41
Volume of shipment2 is 24.0
Weight of shipment2 is 0.76
Shipping cost: $1.28
D:\Demonstration\Demonstration-VI>
Type here to search
4:42 PM
18-Nov-18

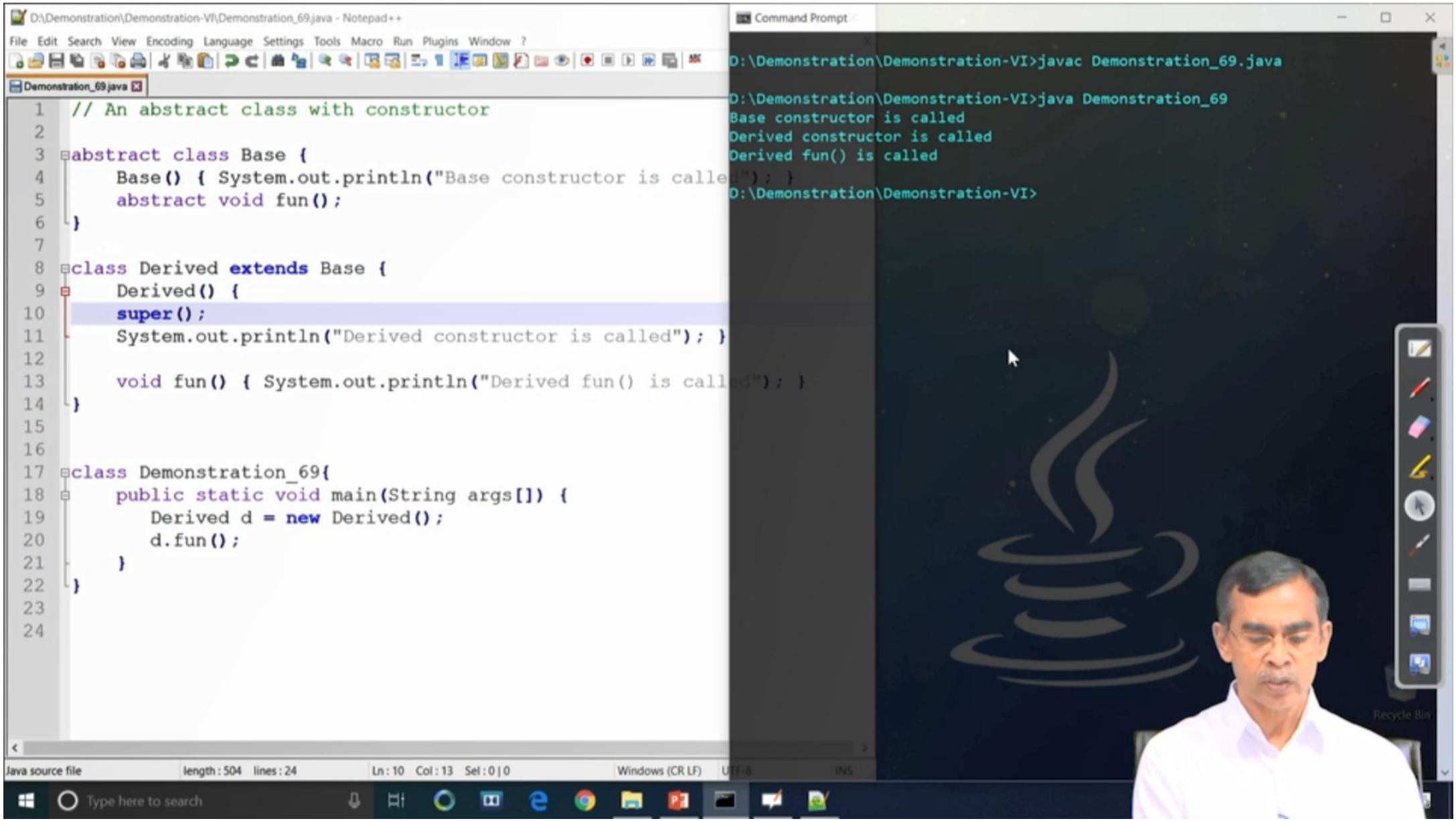

Backward 5 sec - [00:23:29 / 63%]
Demonstration_68.java
/* A simple abstract class example */
abstract class Base {
abstract void fun();
}
class Derived extends Base {
void fun() { System.out.println("Derived fun() is called"); }
}
class Demonstration_68 {
public static void main(String args[]) {
// Uncommenting the following line will cause compiler error as the
// line tries to create an instance of abstract class.
// Base b = new Base();
// We can have references of Base type.
Base b = new Derived();
b.fun();
}
}
Java source file
length : 584 lines : 23
Ln : 1 Col : 1 Sel : 0 | 0
Windows (CR LF)
UTF-8
INS
stration-VI>javac Demonstration_67.java
stration-VI>java Demonstration_67
3000.0
10.0
24.0
0.76
stration-VI>
Type here to search
4:45 PM
18-Nov-18

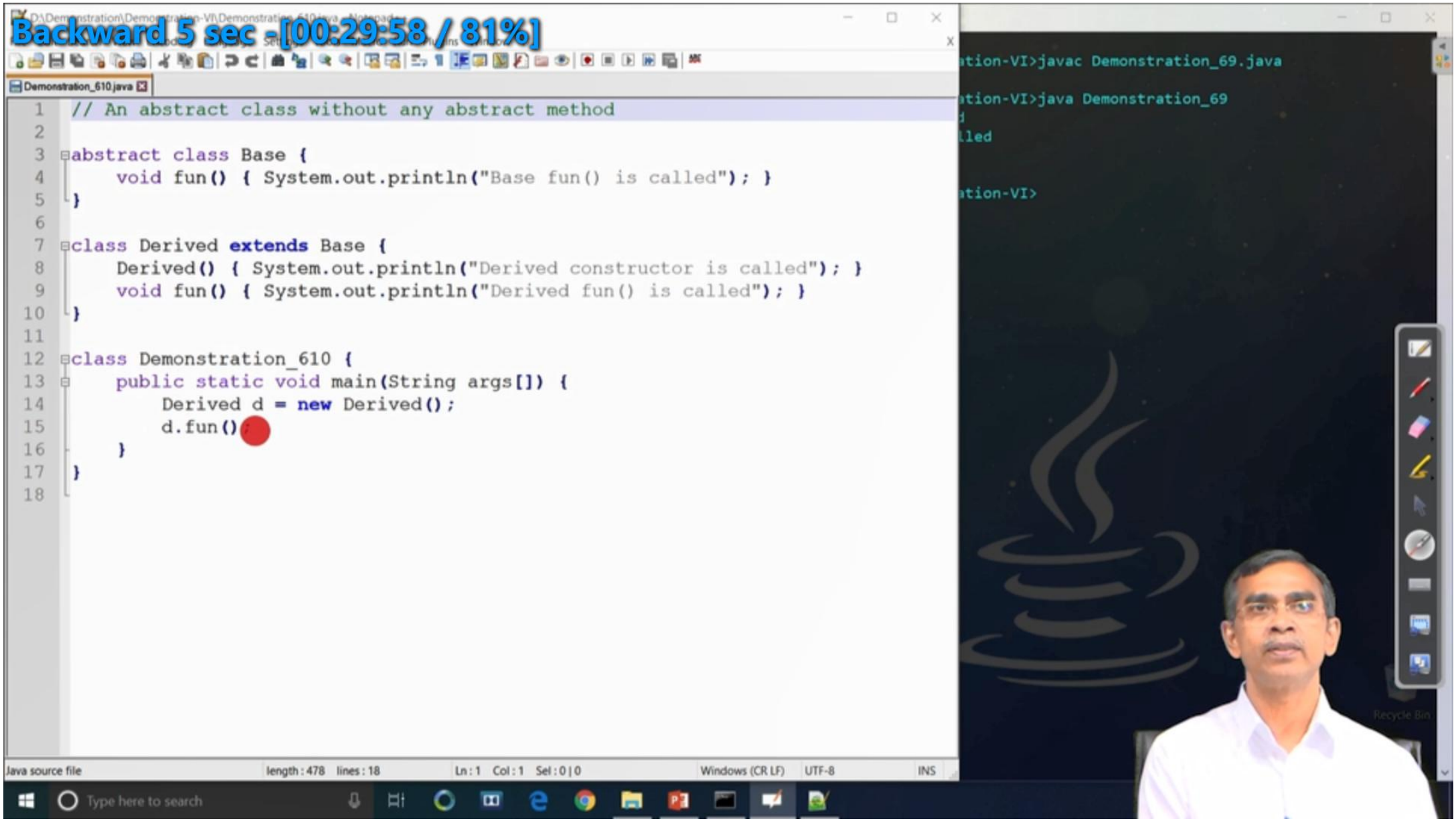

Backward 5 sec - [00:24:20 / 65%]
Demonstration_68.java
/* A simple abstract class example */
abstract class Base {
abstract void fun();
}
class Derived extends Base {
void fun() { System.out.println("Derived fun() is call
}
class Demonstration_68 {
public static void main(String args[]) {
// Uncommenting the following line will cause compiler error as the
// line tries to create an instance of abstract class.
Base b = new Base();
// We can have references of Base type.
Base b = new Derived();
b.fun();
}
}
Command Prompt
D:\Demonstration\Demonstration-VI>javac Demonstration_68.java
Demonstration_68.java:16: error: Base is abstract; cannot be instantiated
Base b = new Base();
^
Demonstration_68.java:19: error: variable b is already defined in method main(
String[])
Base b = new Derived();
^
2 errors
D:\Demonstration\Demonstration-VI>
Java source file
length : 581 lines : 23
Ln : 16 Col : 9 Sel : 0 | 0
Type here to search
4:45 PM
18-Nov-18

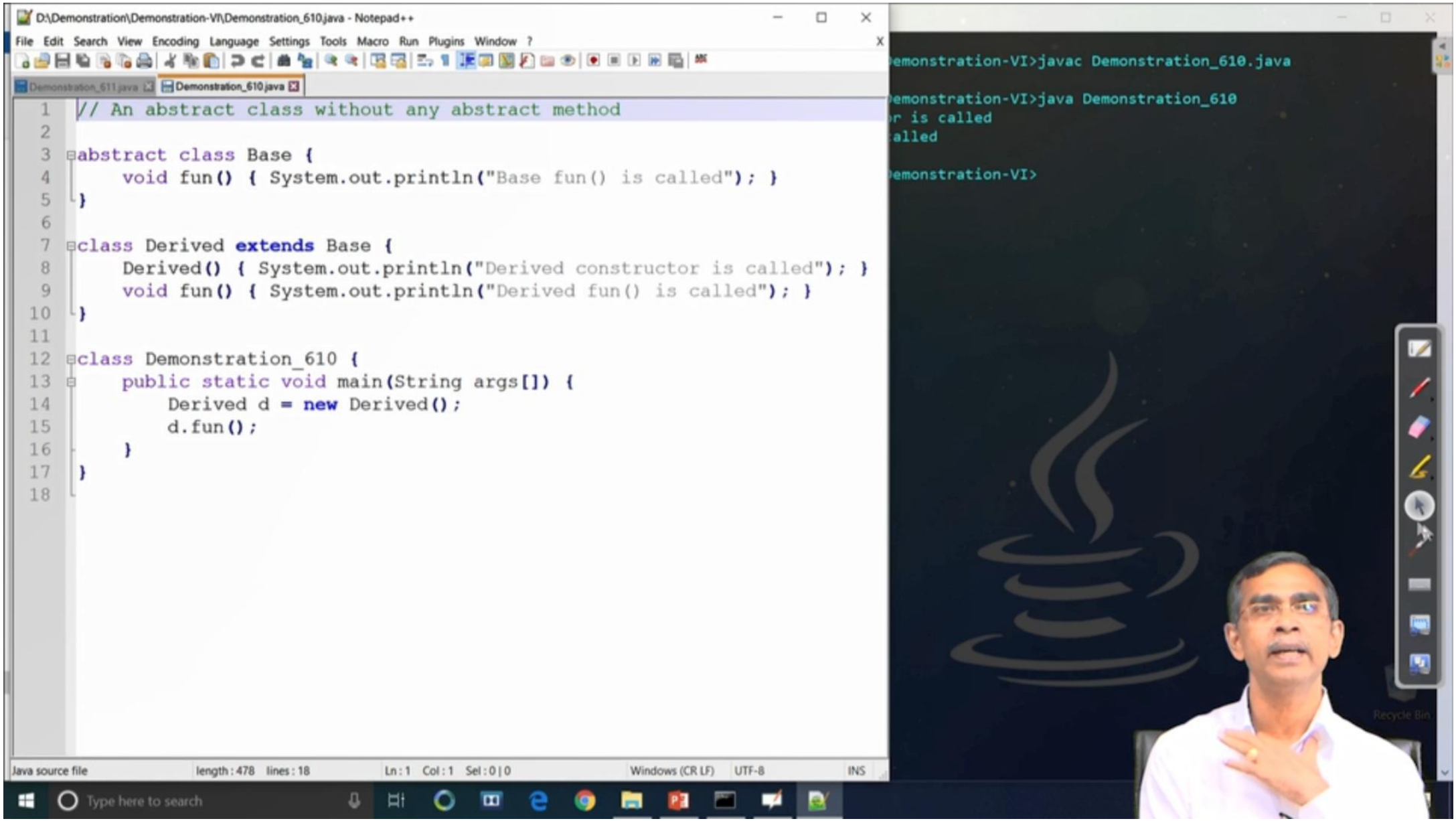

Backward 5 sec - [00:25:29 / 69%]
Demonstration_69.java
// An abstract class with constructor
abstract class Base {
Base() { System.out.println("Base constructor is called"); }
abstract void fun();
}
class Derived extends Base {
Derived() { System.out.println("Derived constructor is called"); }
void fun() { System.out.println("Derived fun() is called"); }
}
class Demonstration_69{
public static void main(String args[]) {
Derived d = new Derived();
d.fun();
}
}
Java source file
length : 487 lines : 21
Ln : 1 Col : 1 Sel : 0 | 0
Windows (CR LF)
UTF-8
INS
\Demonstration-VI>javac Demonstration_68.java
\Demonstration-VI>java Demonstration_68
called
\Demonstration-VI>
Type here to search

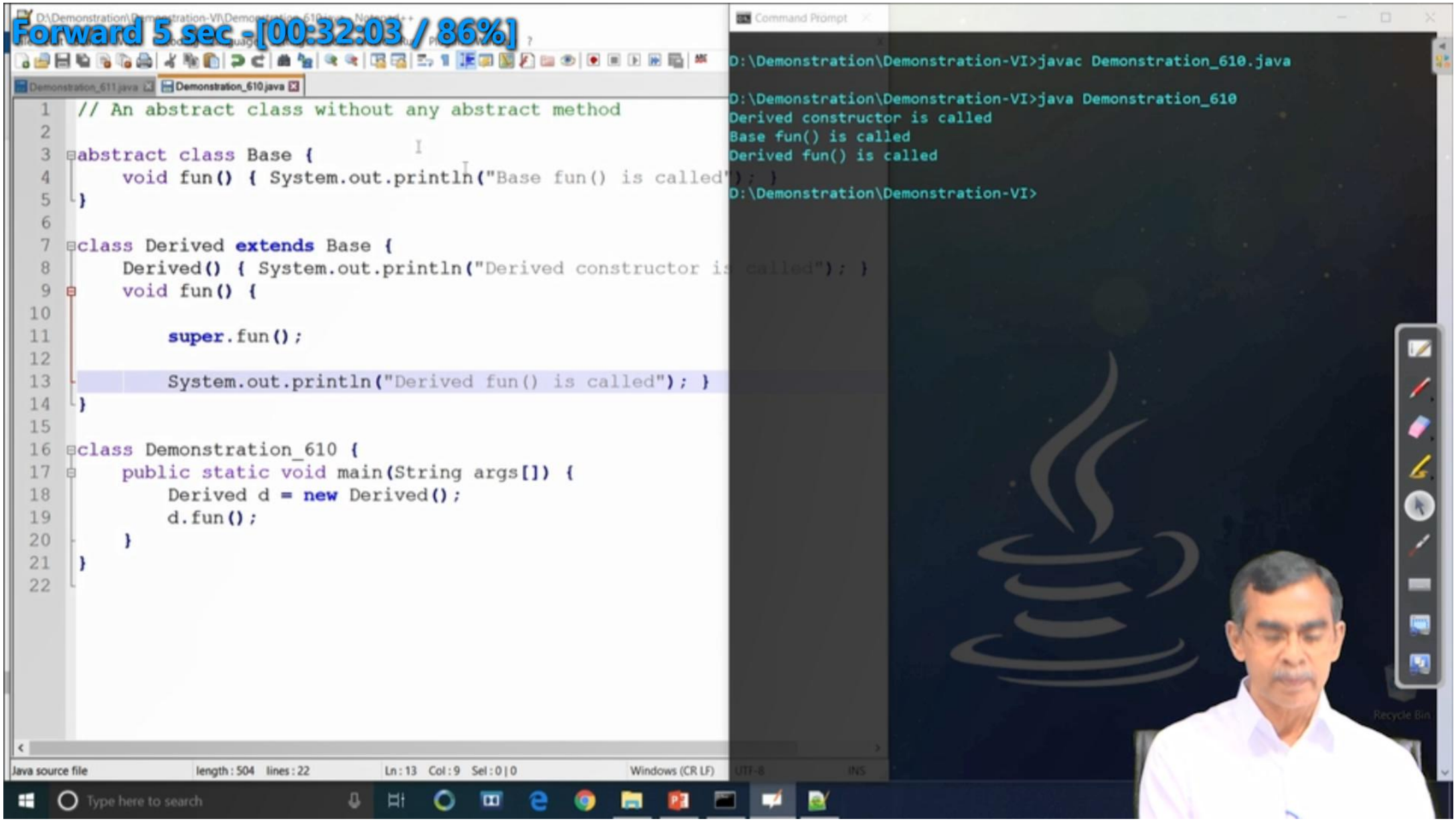

D:\Demonstration\Demonstration-VI\Demonstration_69.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Demonstration_69.java
1 // An abstract class with constructor
2
3 abstract class Base {
4 Base() { System.out.println("Base constructor is called"); }
5 abstract void fun();
6 }
7
8 class Derived extends Base {
9 Derived() { System.out.println("Derived constructor is called"); }
10 super();
11 void fun() { System.out.println("Derived fun() is called"); }
12 }
13
14
15 class Demonstration_69{
16 public static void main(String args[]) {
17 Derived d = new Derived();
18 d.fun();
19 }
20 }
21
22
Java source file length : 498 lines : 22 Ln : 10 Col : 13 Sel : 0 | 0 Windows (CR LF) UTF-8 INS
\Demonstration-VI>javac Demonstration_69.java
\Demonstration-VI>java Demonstration_69
is called
tor is called
called
\Demonstration-VI>
Type here to search
Recycle Bin

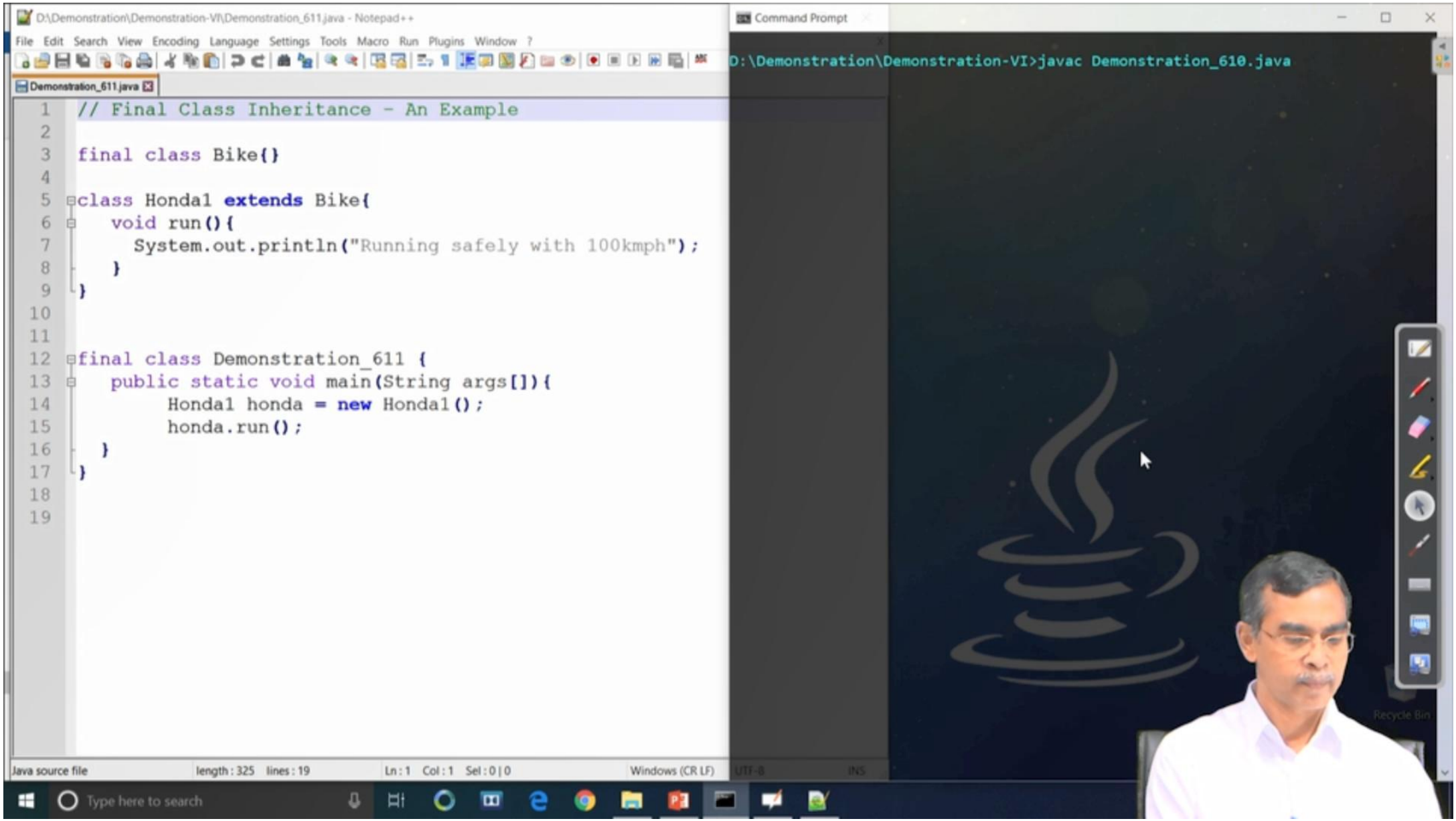

D:\Demonstration\Demonstration-VI\Demonstration_69.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Demonstration_69.java
1 // An abstract class with constructor
2
3 abstract class Base {
4 Base() { System.out.println("Base constructor is calle
5 abstract void fun();
6 }
7
8 class Derived extends Base {
9 Derived() { System.out.println("Derived constructor is called"); }
10 super();
11 void fun() { System.out.println("Derived fun() is called"); }
12 }
13
14
15 class Demonstration_69{
16 public static void main(String args[]) {
17 Derived d = new Derived();
18 d.fun();
19 }
20 }
21
22
Java source file length : 498 lines : 22 Ln : 10 Col : 13 Sel : 0 | 0 Windows (CR LF) UTF-8 INS
Command Prompt
D:\Demonstration\Demonstration-VI>javac Demonstration_69.java
Demonstration_69.java:10: error: illegal start of type
super();
^
1 error
D:\Demonstration\Demonstration-VI>
D:\Demonstration\Demonstration-VI>
Type here to search
Recycle Bin

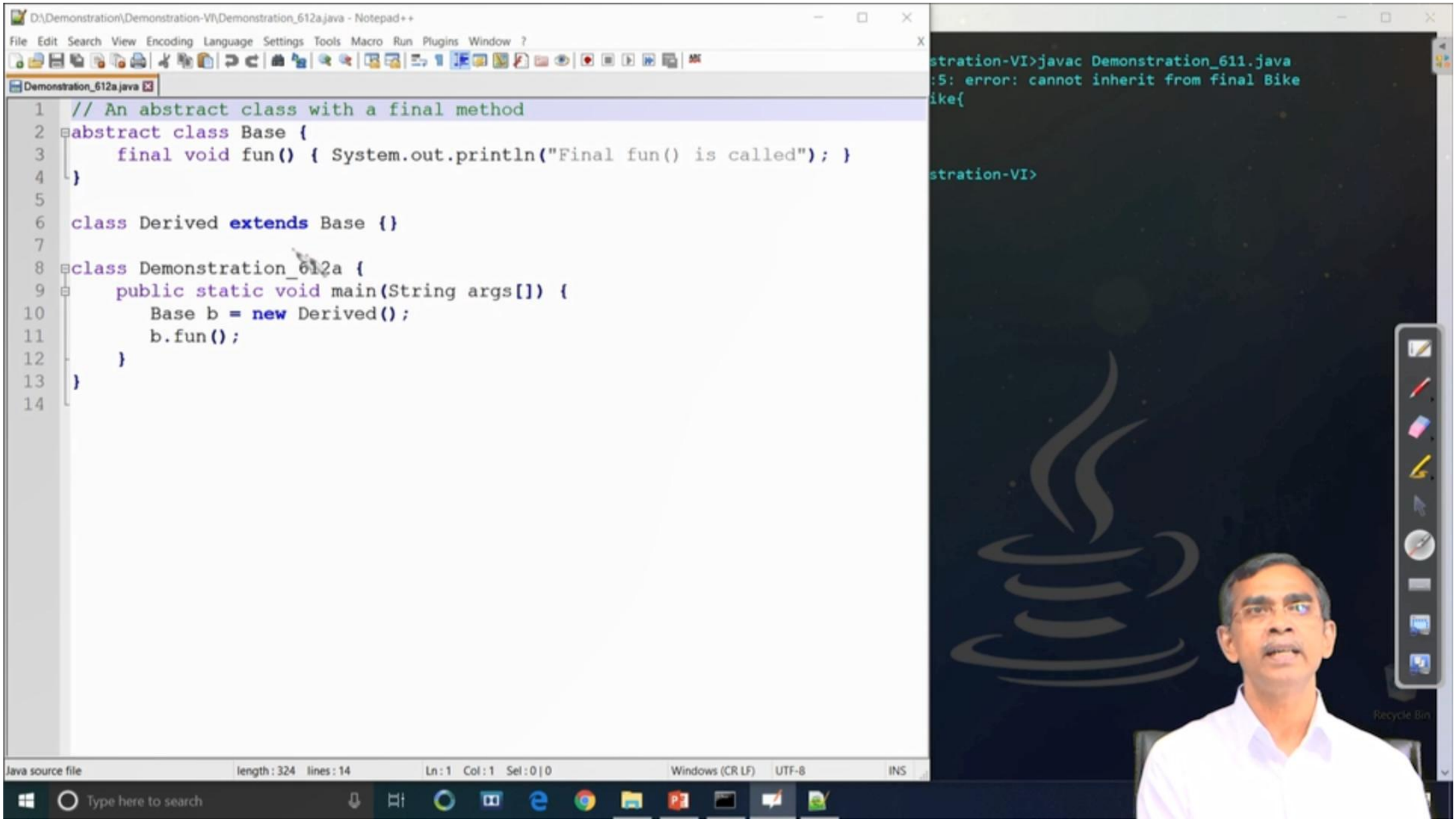

D:\Demonstration\Demonstration-VI\Demonstration_69.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Demonstration_69.java
1 // An abstract class with constructor
2
3 abstract class Base {
4 Base() { System.out.println("Base constructor is calle
5 abstract void fun();
6 }
7
8 class Derived extends Base {
9 Derived() {
10 super();
11 System.out.println("Derived constructor is called"); }
12
13 void fun() { System.out.println("Derived fun() is call
14 }
15
16
17 class Demonstration_69{
18 public static void main(String args[]) {
19 Derived d = new Derived();
20 d.fun();
21 }
22 }
23
24
Java source file length : 504 lines : 24 Ln : 10 Col : 13 Sel : 0 | 0 Windows (CR LF) UTF-8 INS
Command Prompt
D:\Demonstration\Demonstration-VI>javac Demonstration_69.java
D:\Demonstration\Demonstration-VI>java Demonstration_69
Base constructor is called
Derived constructor is called
Derived fun() is called
D:\Demonstration\Demonstration-VI>
Type here to search
Recycle Bin

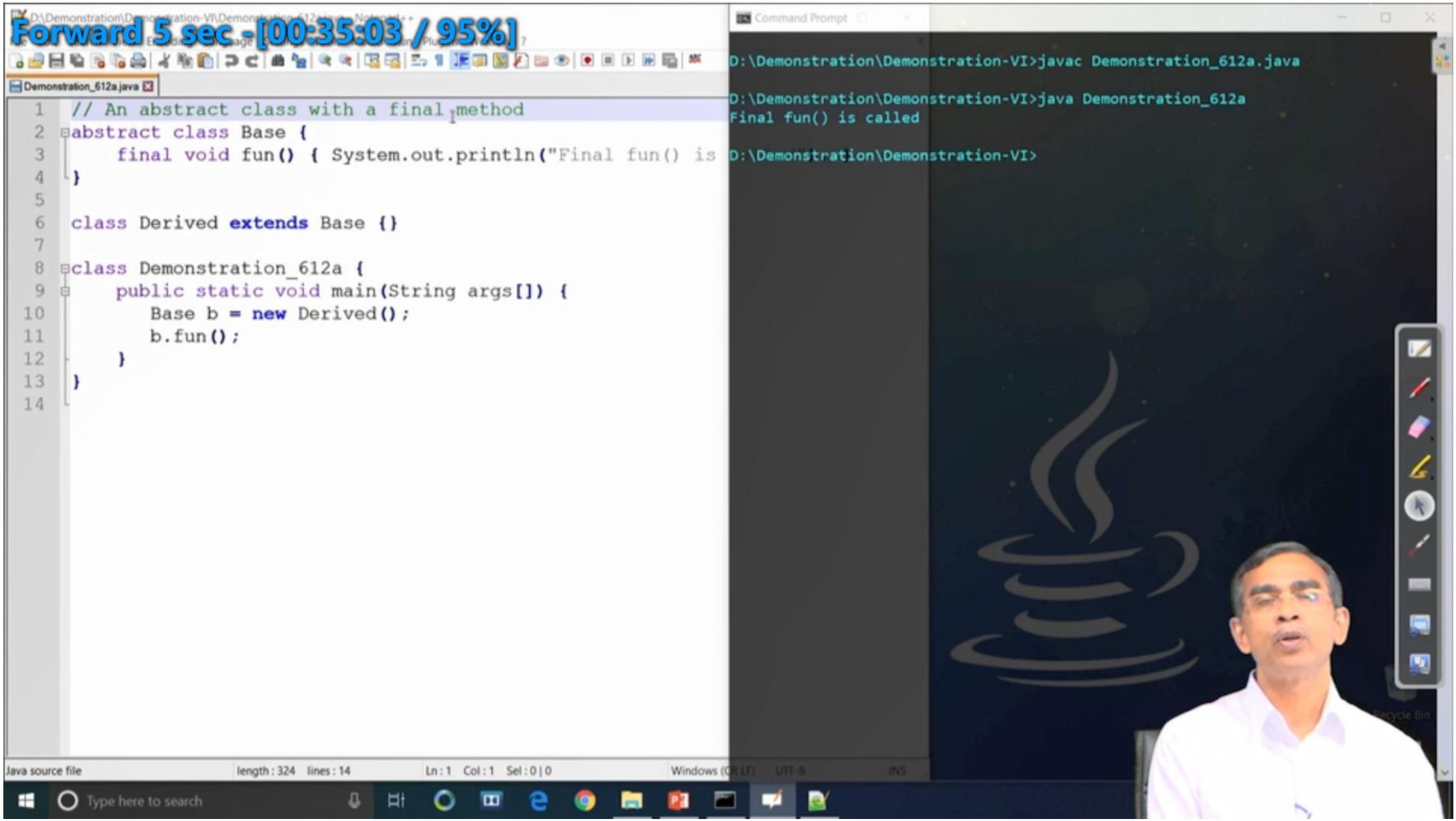

Backward 5 sec - [00:29:58 / 81%]
Demonstration_610.java
// An abstract class without any abstract method
abstract class Base {
void fun() { System.out.println("Base fun() is called"); }
}
class Derived extends Base {
Derived() { System.out.println("Derived constructor is called"); }
void fun() { System.out.println("Derived fun() is called"); }
}
class Demonstration_610 {
public static void main(String args[]) {
Derived d = new Derived();
d.fun();
}
}
Java source file
length : 478 lines : 18
Ln : 1 Col : 1 Sel : 0 | 0
Windows (CR LF)
UTF-8
INS
ation-VI>javac Demonstration_69.java
ation-VI>java Demonstration_69
lled
ation-VI>
Type here to search

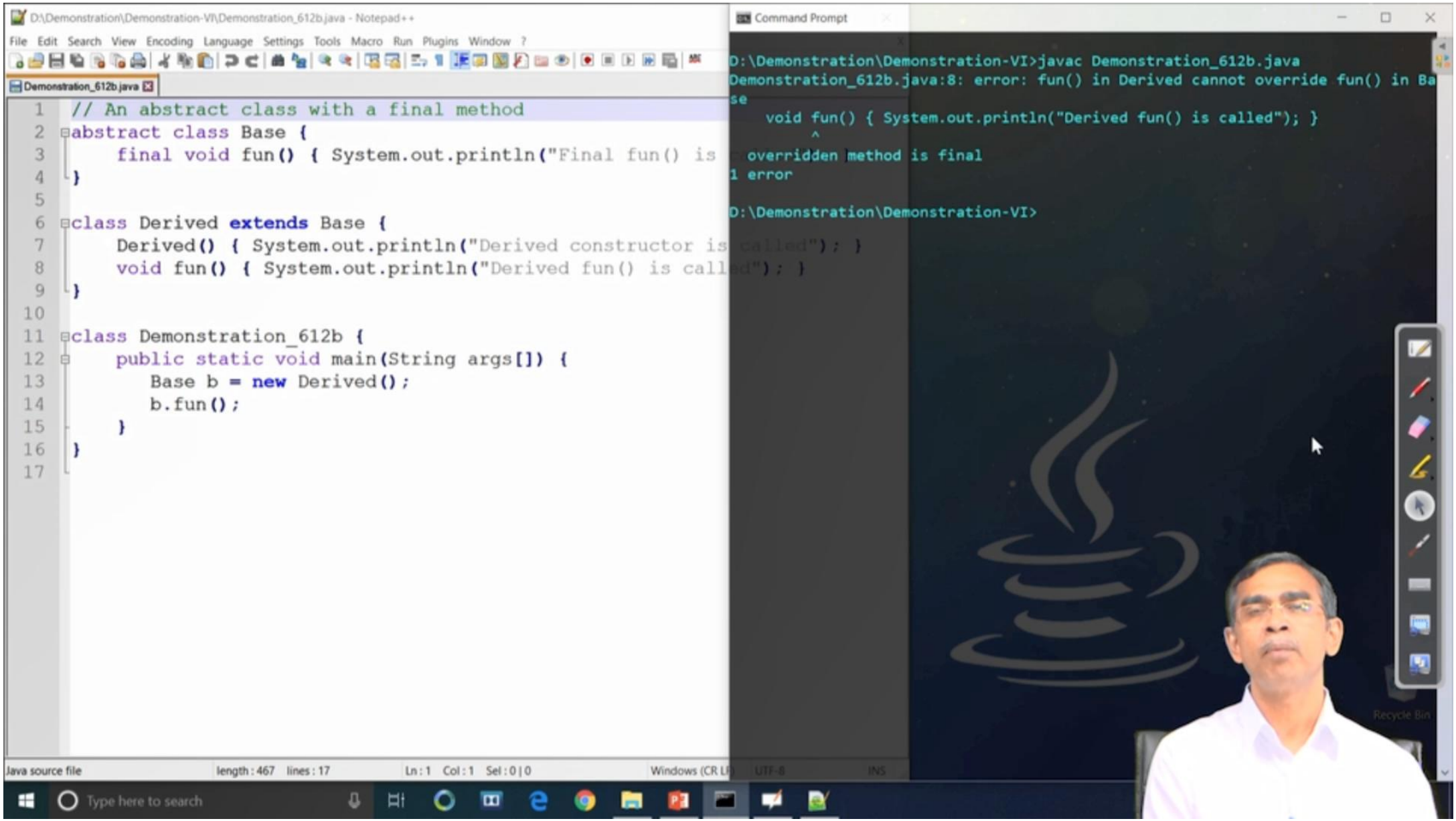

D:\Demonstration\Demonstration-VI\Demonstration_610.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Demonstration_611.java
Demonstration_610.java
1 // An abstract class without any abstract method
2
3 abstract class Base {
4 void fun() { System.out.println("Base fun() is called"); }
5 }
6
7 class Derived extends Base {
8 Derived() { System.out.println("Derived constructor is called"); }
9 void fun() { System.out.println("Derived fun() is called"); }
10 }
11
12 class Demonstration_610 {
13 public static void main(String args[]) {
14 Derived d = new Derived();
15 d.fun();
16 }
17 }
18
Java source file length : 478 lines : 18 Ln : 1 Col : 1 Sel : 0 | 0 Windows (CR LF) UTF-8 INS
emonstration-VI>javac Demonstration_610.java
emonstration-VI>java Demonstration_610
r is called
alled
emonstration-VI>
Type here to search
Recycle Bin


Forward 5 sec - [00:32:03 / 86%]
Demonstration_611.java
Demonstration_610.java
// An abstract class without any abstract method
abstract class Base {
void fun() { System.out.println("Base fun() is called"); }
}
class Derived extends Base {
Derived() { System.out.println("Derived constructor is called"); }
void fun() {
super.fun();
System.out.println("Derived fun() is called"); }
}
class Demonstration_610 {
public static void main(String args[]) {
Derived d = new Derived();
d.fun();
}
}
Java source file
length : 504 lines : 22
Ln : 13 Col : 9 Sel : 0 | 0
Windows (CR LF)
UTF-8
INS
Command Prompt
D:\Demonstration\Demonstration-VI>javac Demonstration_610.java
D:\Demonstration\Demonstration-VI>java Demonstration_610
Derived constructor is called
Base fun() is called
Derived fun() is called
D:\Demonstration\Demonstration-VI>
Type here to search
Recycle Bin


D:\Demonstration\Demonstration-VI\Demonstration_611.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Demonstration_611.java
// Final Class Inheritance - An Example
final class Bike{}
class Honda1 extends Bike{
void run(){
System.out.println("Running safely with 100kmph");
}
}
final class Demonstration_611 {
public static void main(String args[]){
Honda1 honda = new Honda1();
honda.run();
}
}
Java source file
length : 325 lines : 19
Ln : 1 Col : 1 Sel : 0 | 0
Windows (CR LF)
UTF-8
INS
Command Prompt
D:\Demonstration\Demonstration-VI>javac Demonstration_610.java
Type here to search
Recycle Bin


D:\Demonstration\Demonstration-VI\Demonstration_612a.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Demonstration_612a.java
1 // An abstract class with a final method
2 abstract class Base {
3 final void fun() { System.out.println("Final fun() is called"); }
4 }
5
6 class Derived extends Base {}
7
8 class Demonstration_612a {
9 public static void main(String args[]) {
10 Base b = new Derived();
11 b.fun();
12 }
13 }
14
Java source file length : 324 lines : 14 Ln : 1 Col : 1 Sel : 0 | 0 Windows (CR LF) UTF-8 INS
stration-VI>javac Demonstration_611.java
:5: error: cannot inherit from final Bike
ike{
stration-VI>
Type here to search
Recycle Bin

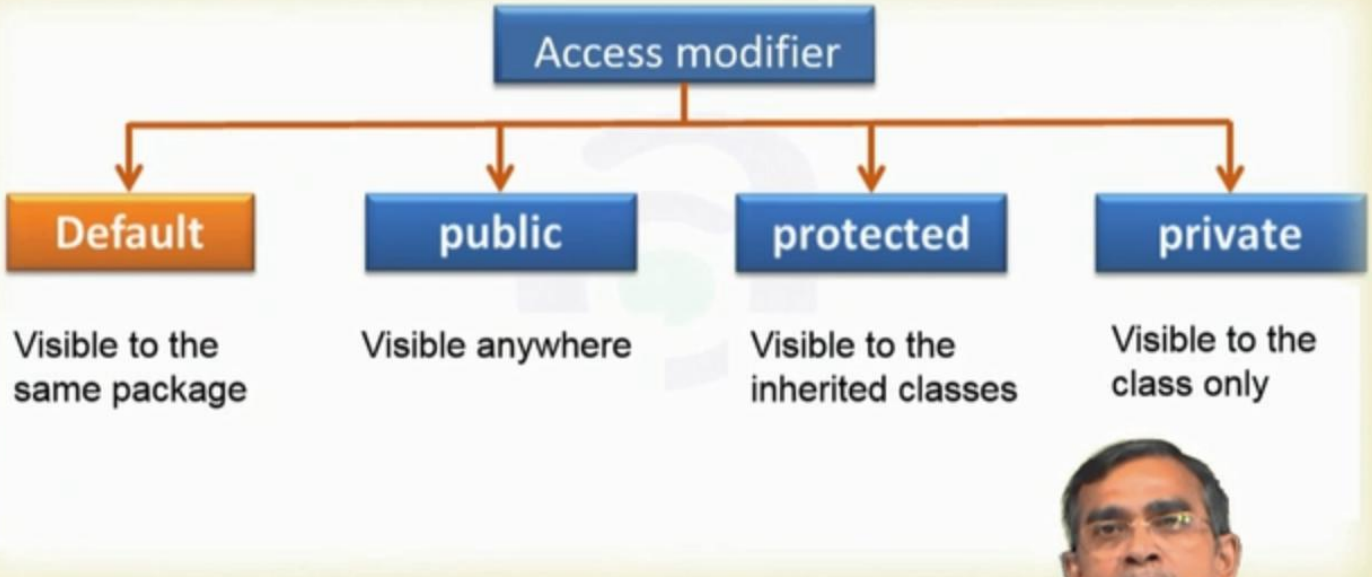

Forward 5 sec - [00:35:03 / 95%]
Demonstration_612a.java
1 // An abstract class with a final method
2 abstract class Base {
3 final void fun() { System.out.println("Final fun() is
4 }
5
6 class Derived extends Base {}
7
8 class Demonstration_612a {
9 public static void main(String args[]) {
10 Base b = new Derived();
11 b.fun();
12 }
13 }
14
Command Prompt
D:\Demonstration\Demonstration-VI>javac Demonstration_612a.java
D:\Demonstration\Demonstration-VI>java Demonstration_612a
Final fun() is called
D:\Demonstration\Demonstration-VI>
Java source file
length : 324 lines : 14
Ln : 1 Col : 1 Sel : 0 | 0
Type here to search


D:\Demonstration\Demonstration-VI\Demonstration_612b.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Demonstration_612b.java
1 // An abstract class with a final method
2 abstract class Base {
3 final void fun() { System.out.println("Final fun() is
4 }
5
6 class Derived extends Base {
7 Derived() { System.out.println("Derived constructor is
8 void fun() { System.out.println("Derived fun() is call
9 }
10
11 class Demonstration_612b {
12 public static void main(String args[]) {
13 Base b = new Derived();
14 b.fun();
15 }
16 }
17
Java source file length : 467 lines : 17 Ln : 1 Col : 1 Sel : 0 | 0 Windows (CR LF) UTF-8 INS
Command Prompt
D:\Demonstration\Demonstration-VI>javac Demonstration_612b.java
Demonstration_612b.java:8: error: fun() in Derived cannot override fun() in Base
void fun() { System.out.println("Derived fun() is called"); }
^
overridden method is final
1 error
D:\Demonstration\Demonstration-VI>
Type here to search
Recycle Bin

Concept of access modifiers

The **access modifiers** in Java specify accessibility (scope) of a data member, method, constructor or class.

Type of access modifiers :



Access levels of modifiers

Access levels / Modifier	Class	Package	Sub class	Everywhere
Public	✓	✓	✓	✓
Protected	✓	✓	✓	✗
Default	✓	✓	✗	✗
Private	✓	✗	✗	✗

Access levels of modifiers

Access levels / Modifier	Class	Package	Sub class	Everywhere
Public	✓	✓	✓	✓
Protected	✓	✓	✓	✗
Default	✓	✓	✗	✗
Private	✓	✗	✗	✗

Default access modifier

If you don't use any modifier, it is treated as **default** by default.

The default modifier is accessible only within a package.

Default access modifier : An Example

```
//Save this program as A.java in a sub-directory "pack1"

class A {
   void msg(){System.out.println("Hi! I am in Class A");
  }
}
```

```
      /* Save this program as B.java in another
         sub-directory "pack2"  */

  class B{
     public static void main(String args[]){
          A obj = new A();    //Compile Time Error
          obj.msg();          //Compile Time Error
     }
  }
```

Here, two classes are with default access modifier. If they reside in two different directories, then the `class A` is not accessible to `class B` and vice-versa.

However, if they reside in the same directory, then there will be no error!

Default access modifier : An Example

```
//Save this program as A.java in a directory, say temp.

class A {
   void msg(){System.out.println("Hi! I am in Class A");
  }
}
```

```
/* Save this program as B.java in the same directory */

  class B{
     public static void main(String args[]){
          A obj = new A();    //Okay. It is accessible.
          obj.msg();          //It is also accessible.
    }
 }
```

Here, two classes are with default access modifier, and thy are residing in the same directory (or may be in the same file). Hence, in this case, the `class A` is accessible to `class B` and vice-versa.

Public access modifier

The **public** access modifier is accessible everywhere.

It has the widest scope among all other modifiers.

public access modifier – An Example

```
//Save as A.java in a sub-directory say pack1
 package pack1;
 public class A{
      public void msg(){
         System.out.println("Class A: Hello Java!");
      }
 }
```

```
//Save as B.java  in another sub-directory say pack2
 package  pack2;
 import pack1.*;

 class B{
      public static void main(String args[]){
           A obj = new A();
           obj.msg();
       }
 }
```

The `class A` in package `pack1` is `public`, so this class can be accessed from any class outside to this package, for example, in this case, from `class B` in `pack2` .

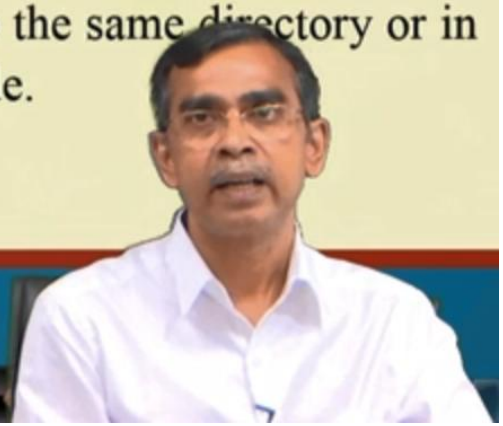
```
Class A: Hello Java!
```

public access modifier : An example

```
public class A{
    public int data = 40;
    public void msg(){
          System.out.println("Class A: Hello Java!");
    }
}

public class B{
    public static void main(String args[]){
        A obj = new A();     //OK : Class A is public
        System.out.println(obj.data);
                             //OK : data is public
        obj.msg();           //OK: msg is public
    }
}
```

We have created two classes `class A` and `class B`. The `class A` contains public data member and public method and are accessible to `class B`.

Note:
It does not matter whether the `class A` and `class B` belong to the same directory or in the same program file.



Public access modifier : Another example

```
//Save this program as A.java
package pack1;        // It is a sub-directory "pack1"

public class A {
  void msg(){System.out.println("Hi! I am in Class A");}
}
```

```
//Save this program as B.java
package pack2;  // It is another sub-directory "pack2"
import pack1.*;  /* Import all classes in pack1 here

class B{
    public static void main(String args[]){
        A obj = new A();    //Okay program!
        obj.msg();          //This is now public
    }
}
```

When a class is `public`, all its member with default access specifier are also `public`.

Private access modifier

The **private** access modifier is accessible only within the class.

Private access modifier : An example

```
public class A{
    private int data = 40;
    public void msg(){
          System.out.println("Class A: Hello Java!");
    }
}

 public class B{
     public static void main(String args[]){
          A obj = new A();     //OK : Class A is public
          System.out.println(obj.data);
             //Compile Time Error : data is private
          obj.msg(); //OK : msg is public
     }
}
```

Private access modifier : An example

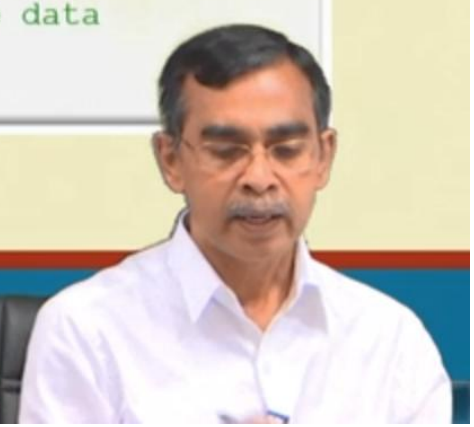
```
private class A{
    int data = 40;
    void msg(){
          System.out.println("Class A: Hello Java!");
    }
}

 public class B{
     public static void main(String args[]){
          A obj = new A();     //Error : Class A is public
          System.out.println(obj.data);
             //Compile Time Error : data is private
          obj.msg(); //Error : msg is private
     }
}
```

When a class is `private`, all its member with default access specifier are also `private`.

How, if a member in a public class is declared as public or protected?

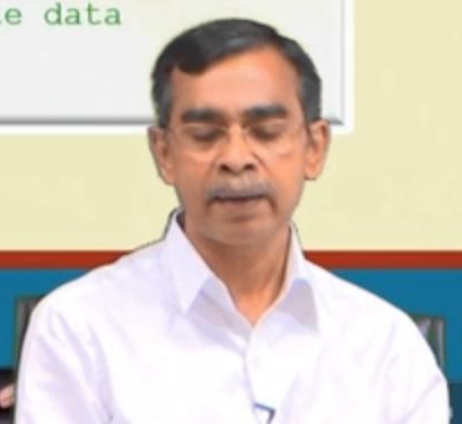
Think about this...

```
public class A{
   private int data = 40;
   public void msg(){
         System.out.println("Hello Java!" + data);
   }
}

 public class B{
    public static void main(String args[]){
       A obj = new A();     //OK : Class A is public
       System.out.println(obj.data);//Compile Time Error : data is private
       obj.msg(); //Calls msg() method of class A, which in turns private data
    }
 }
```



Think about this...

```
public class A{
   private int data = 40;
   public void msg(){
         System.out.println("Hello Java!" + data);
   }
}

 public class B{
    public static void main(String args[]){
       A obj = new A();     //OK : Class A is public
       System.out.println(obj.data);//Compile Time Error : data is private
       obj.msg(); //Calls msg() method of class A, which in turns private data
    }
 }
```



private constructor : An example

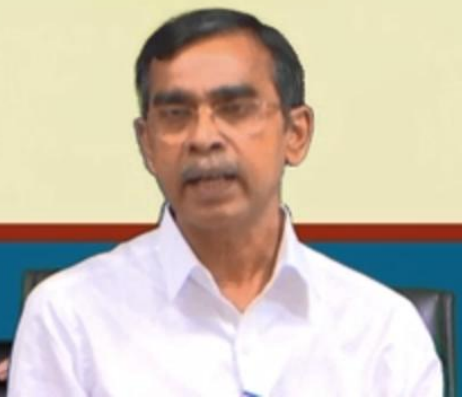
```
public class A{
   private A(){
         //private constructor
   }
   void msg(){
         System.out.println("Class A: Hello Java!");
   }
}

public class Simple{
   public static void main(String args[]){
      A obj = new A();    //Compile Time Error!
   }
}
```

If you make any class constructor **private**, you can not create an instance of that class from outside the class.

Protected access modifier

The **protected** access modifier is accessible within a package or from outside a package but through inheritance only.

The protected access modifier can be applied on the data member, method and constructor. It can't be applied on the class.



Pause

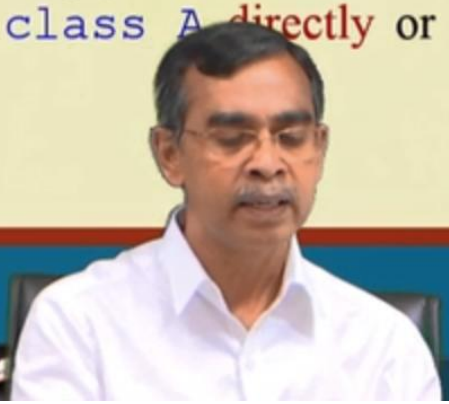
Protected access modifier : An Example

```
public class A{
    protected int i = 555;
    void msg(){
        System.out.println("Class A: Hello Java!" + i);
    }
}
```

```
class B {
    public static void main(String args[]){
        A obj = new A();
        obj.msg(); // Error: Compilation error
    }
}
```

Here, the protected data `i` is accessible to any methods in the same class. Alos, it is accessible to any of its sub class.

Here, `class A` is accessible to `class B` as it is declared `public`; however, any method in the `class B` (even they are in the same file or package) cannot access `protected` data of `class A` directly or indirectly.



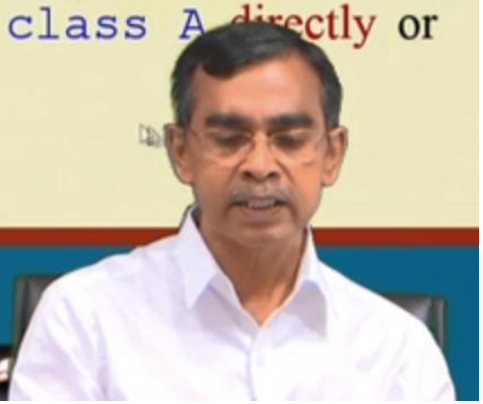
Protected access modifier : An Example

```
public class A{
    protected int i = 555;
    void msg(){
        System.out.println("Class A: Hello Java!" + i);
    }
}
```

```
class B {
    public static void main(String args[]){
        A obj = new A();
        obj.msg(); // Error: Compilation error
    }
}
```

Here, the protected data `i` is accessible to any methods in the same class. Alos, it is accessible to any of its sub class.

Here, `class A` is accessible to `class B` as it is declared `public`; however, any method in the `class B` (even they are in the same file or package) cannot access `protected` data of `class A` directly or indirectly.



Protected access modifier : An Example

```
public class A{
     public int i = 555;
     protected  void msg(){
          System.out.println("Hello Java! + i");
     }
}
```

```
class B extends A{
     public static void main(String args[]){
          B obj = new B();
          obj.msg();
     }
}
```

Hello Java! 555

Here, the `msg()` method of the `class A` is declared as `protected`, and hence, it can be accessed from outside the class only through inheritance.

What will happen if `i` is made private in `class A`?

Protected access modifier : An Example

```
public class A{
     public int i = 555;
     protected  void msg(){
          System.out.println("Hello Java! + i");
     }
}
```

```
class B extends A{
     public static void main(String args[]){
          B obj = new B();
          obj.msg();
     }
}
```

Hello Java! 555

Here, the `msg()` method of the `class A` is declared as `protected`, and hence, it can be accessed from outside the class only through inheritance.

What will happen if `i` is made private in `class A`?

Protected access modifier : Another Example

```
//Save as A.java  in a sub directory pack1
package pack1;

public class A{
    protected void msg(){
        System.out.println("Class A: Hello Java!");
    }
}
```

```
//Save as B.java  in another sub-directory pack2
package pack2;
import pack1.*;

class B extends A{
    public static void main(String args[]){
        B obj = new B();
        obj.msg();
    }
}
```

```
Hello Java!
```

We have created the two packages `pack1` and `pack2`.

The `class A` of `pack1` package is `public`, so can be accessed from outside the package. The method `msg()` of the `class A` is declared as `protected`, so it can be accessed from outside the class through inheritance.

Java access modifiers with method overriding

```
public class A{
    protected void msg(){
        System.out.println("Class A: Hello Java!");
    }
}

public class Simple extends A{
    void msg(){
        System.out.println("Class B: Welcome!");

    }

    public static void main(String args[]){
        Simple obj = new Simple();
        obj.msg();
    }
}
```

If you are **overriding** any method, overridden method (i.e. declared in sub class) must not be more restrictive.

The **default** modifier is more restrictive than **protected**.

What will happen if the `msg()` in class `Simple` is declared as `public` or `protected`?

Java access modifiers with method overriding

```
public class A{
    protected void msg(){
        System.out.println("Class A: Hello Java!");
    }
}

public class Simple extends A{
    void msg(){
        System.out.println("Class B: Welcome!");

    }

    public static void main(String args[]){
        Simple obj = new Simple();
        obj.msg();
    }
}
```

If you are **overriding** any method, overridden method (i.e. declared in sub class) must not be more restrictive.

The **default** modifier is more restrictive than **protected**.

What will happen if the `msg()` in class `Simple` is declared as `public` or `protected`?

Questions to think...

- How a package can be built ?
- Is it possible that two classes having the same name but in two different packages are to be used in another class outside the packages?