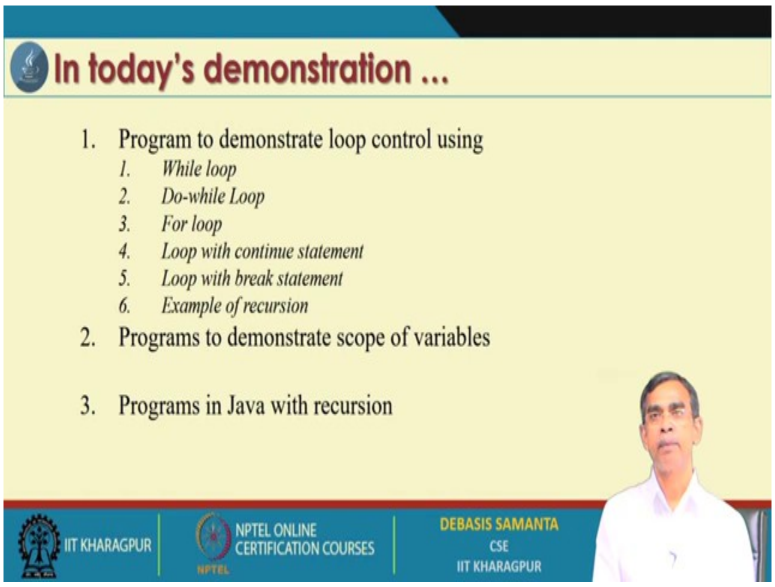
Programming in Java
Prof. Debasis Samanta
Department of Computer Science Engineering
Indian Institute of Technology, Kharagpur

Lecture – 12
Demonstration – V

In the last module, we have learned about the static scope role in Java and also a little bit about writing a recursive program. In this Demonstration, we have a quick illustration of the different concepts regarding the scope rules that we have learned and then recursive program writing.

(Refer Slide Time: 00:33)



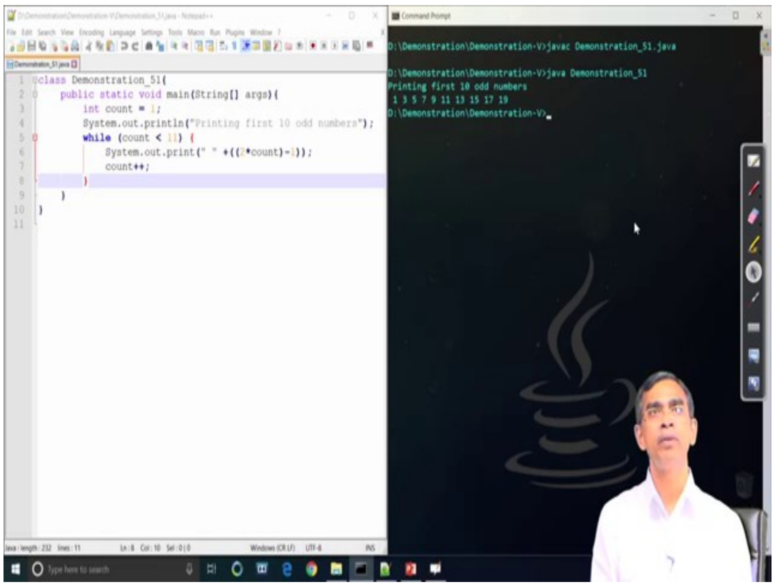
In today's demonstration ...

1. Program to demonstrate loop control using
   1. *While loop*
   2. *Do-while Loop*
   3. *For loop*
   4. *Loop with continue statement*
   5. *Loop with break statement*
   6. *Example of recursion*
2. Programs to demonstrate scope of variables
3. Programs in Java with recursion

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
DEBASIS SAMANTA
CSE
IIT KHARAGPUR

Before going to that we should have basic control structure, I know that you, if you are already an experienced C programmer for you all those basic control structures, is known to you, but those are new for them it is just to make the consistency we will just discuss quickly. So, there are different loop structures and then switch case will be covered and then finally, discuss the scope rule and then Java program writing using a recursive function. So, let us have the quick demo about first our example is how we can use the while loop let us see the program.

(Refer Slide Time: 01:23)



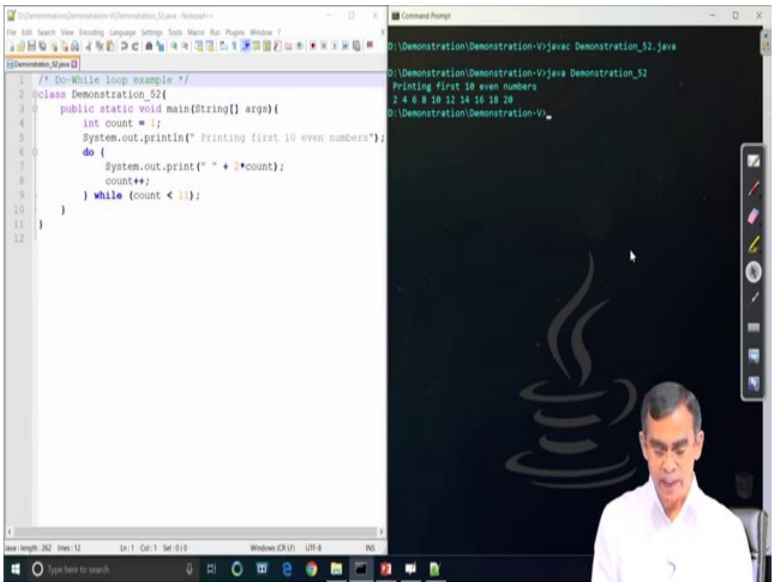
```java
class Demonstration_51{
    public static void main(String[] args){
        int count = 1;
        System.out.println("Printing first 10 odd numbers");
        while (count < 11) {
            System.out.print(" " +((2*count)-1));
            count++;
        }
    }
}
```

```
D:\Demonstration\Demonstration-V>javac Demonstration_51.java

D:\Demonstration\Demonstration-V>java Demonstration_51
Printing first 10 odd numbers
 1 3 5 7 9 11 13 15 17 19
D:\Demonstration\Demonstration-V>
```

So, this is one program as you can see this program basically uses the while and within this while and there is a condition; that means, this loop will continue until the value of count is less than 11. And you can guess that what this while loop into it basically includes on print statement, it will basically print the odd numbers starting from 0 0 next sort numbers star after 0 is 1 of course, so, 1 3 5 7 like this one.

Now, let us run this program. So, that you can see how this while loop will roll it, you write the compilation, you have to compile it right the Java C. So, this is a program that is compiled the here we can see the program is compiled successfully now let us run this program. So, Java yes and as you can see this a printing first ten odd numbers it is here. Now, this is a one form of while loop now let us have another form we can open 5.2 program

(Refer Slide Time: 02:35)



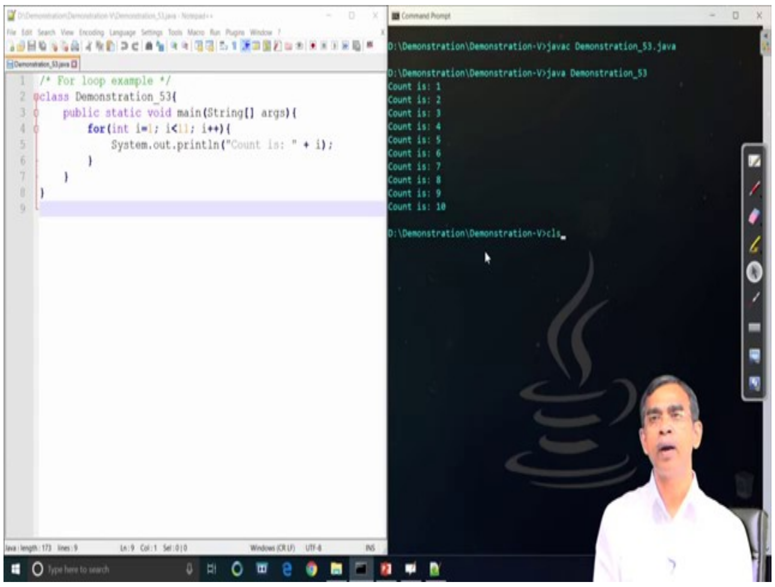
```java
/* Do-While loop example */
class Demonstration_52{
    public static void main(String[] args){
        int count = 1;
        System.out.println(" Printing first 10 even numbers");
        do {
            System.out.print(" " + 2*count);
            count++;
        } while (count < 11);
    }
}
```

```
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>
```

So, this is another program and it is the the same thing in the earlier program it is printing the odd number, but as you can see this program is to print the first ten even numbers. And here instead of while we have used the do while statement and is mostly the same thing, but obviously, there is a difference between the while and do while. Now let us run this program and then let us first we have to compile it yeah so, running this program. So, it basically prints it, now if I asked you what is the difference between while and do while?

So, while basically check the condition first whereas, in the do while at least one loop will roll and then once the one loop is done then the condition check at the end. So, the difference is that while loop may not execute in a single loop, but at least do while at least one loop will be executed.

(Refer Slide Time: 03:47)



```java
/* For loop example */
class Demonstration_53{
    public static void main(String[] args){
        for(int i=1; i<11; i++){
            System.out.println("Count is: " + i);
        }
    }
}
```

```
D:\Demonstration\Demonstration-V>javac Demonstration_53.java

D:\Demonstration\Demonstration-V>java Demonstration_53
Count is: 1
Count is: 2
Count is: 3
Count is: 4
Count is: 5
Count is: 6
Count is: 7
Count is: 8
Count is: 9
Count is: 10

D:\Demonstration\Demonstration-V>cls
```

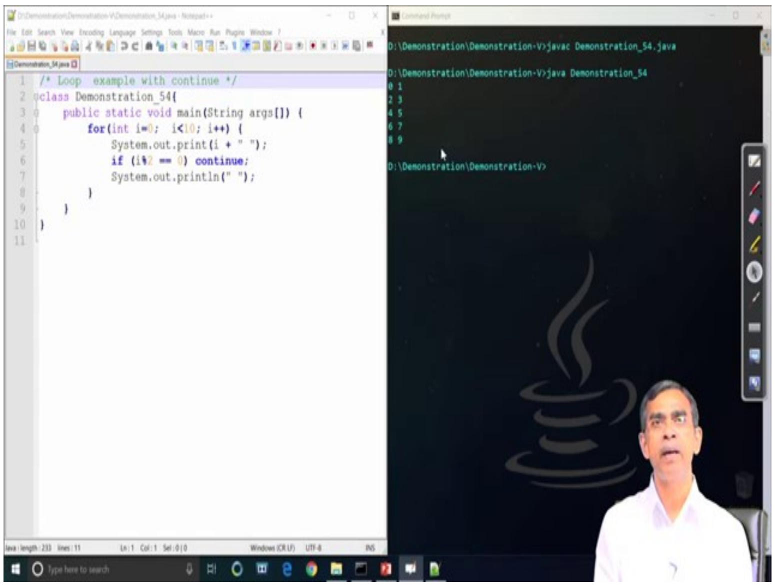
Now, similar to while and do while there is another constructor in Java it is called the for loop. The for loop is totally different than the while loop or do while loop. Now, in case of any loop, there are three things are to be considered initialization for the loop variable, in the first two examples the loop variable was count.

Now, initialization of the loop variable and then the condition checking; that means, whether the loop has reaches it's the termination condition or not and then updating of the loop variable. Now all the three things can be done using for the statement in one line. Now here we can see the first statement here int i, in this case, it is the loop variable; that means, it is the this is the variable i which controls the execution of the loop. So, here int i = 1; that means, initialization of the loop variable as 1 and this is the condition checking that look will continue until the value of i is less than 11 till the value of i is less than 11.

Once, the value of i=11 or more it will exit the loop and i++ once the loop is completed one its term then it will increase the value i automatically is in the one. Now here in this for loop as you can see is a simple print statement which basically prints the value of i. This means that this for loop will print i=1 to i=10 the 10th first integer number. So, if we run this program then we will be able to see exactly how this for loop works for you. So, this for loop as we expected that it prints 1 to 10 consecutively.

Now, so, this is an example of the do while and for loop or the three loop structures in Java program, it is very similar to the structures that are there already in C and C++.

(Refer Slide Time: 06:03)



```
/* 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(" ");
        }
    }
}
```

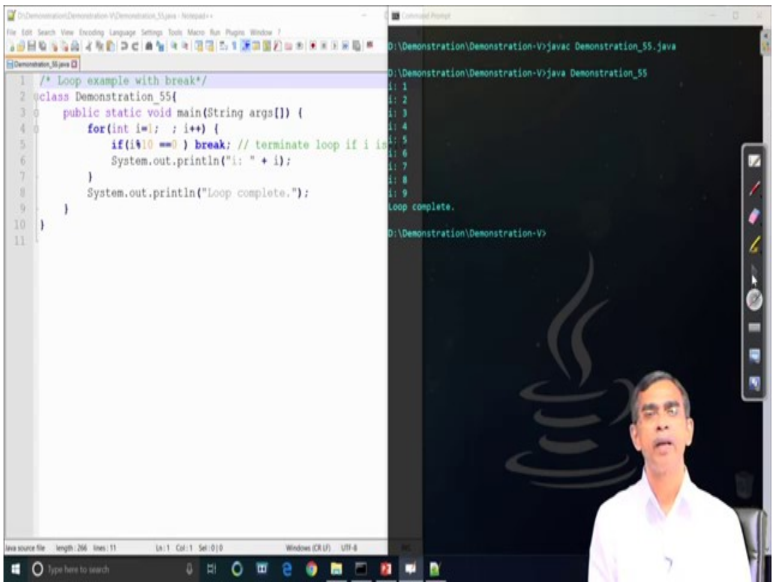
```
D:\Demonstration\Demonstration-V>javac Demonstration_54.java

D:\Demonstration\Demonstration-V>java Demonstration_54
0 1
2 3
4 5
6 7
8 9

D:\Demonstration\Demonstration-V>
```

Now, the loop can be controlled or it can be terminated abnormally and before for this thing there are two more statements is there called the break and continue. So, the purpose of the continue statement is that if some condition is satisfied it will not execute the remaining part in the for loop block.

For example, in this example here the for loop we have this if i; that means, i not 2 there means if i is an even number, then it will not execute the next statement namely system.out print ln and it will go to the next round of the loop. So, continue is basically skip the remaining part in the for loop is going to the loop again. Now if we run this program as it is you can anticipate that this will print the first statement in the for loop is print; that means, it will print two numbers in one line and whenever there is a mode it will go to the next line like this one. So, is the print statement will print the values here.

Here we can see 0 1 in one line and then mode so, 2 it goes to the next and so on so on. So, continue statement is executed after every odd-even number in the for loop.

(Refer Slide Time: 07:35)



```java
/* Loop example with break*/
class Demonstration_55{
    public static void main(String args[]) {
        for(int i=1;  ; i++) {
            if(i%10 ==0 ) break; // terminate loop if i is
            System.out.println("i: " + i);
        }
        System.out.println("Loop complete.");
    }
}
```

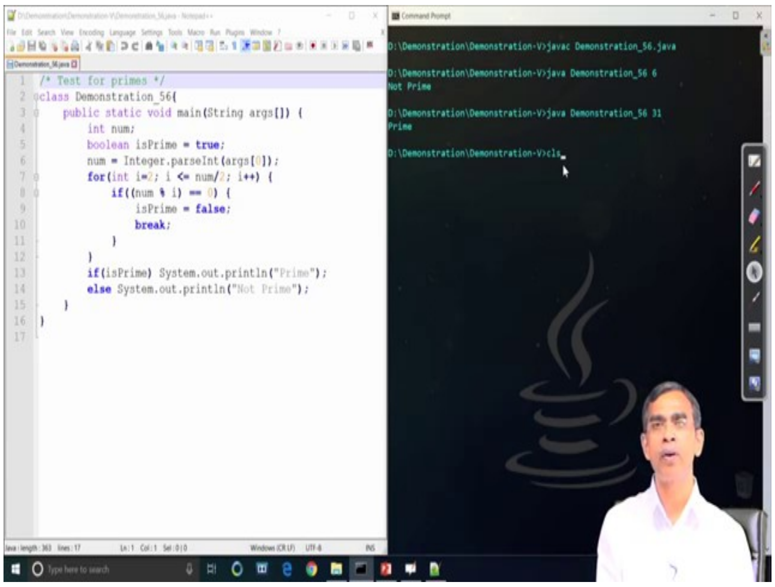
```
D:\Demonstration\Demonstration-V>javac Demonstration_55.java

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>
```

So, this is the idea about the continue like continue there is one break statement; the break statement whenever a condition reaches. So, it is basically terminate the loop without any satisfaction of the loop control termination criteria. So, this is an example here we can see here i mode 10 indicates that whenever the value of i reaches to 10 or multiple of 10 whatever it is there so, the loop will be terminated.

So, in this for loop as we can see int i=1 and you can see the condition here is basically null we do not give any condition; that means, if you do not put any condition checking there then that loop will suppose to execute for infinite times. But if there is a break statement and if there is a condition which satisfies the condition then the break statement is executed then the loop will be terminated there. So, although it looks like an infinite loop, because of the break, there is a condition that it will terminate the loop.

Now, let us run this program as you can see it will pin i=1 2 3 and till i=till it reaches i=10; once i=10 loops will break and it will terminate the loop yeah. So, it is basically the concept that how the break statement is there? Now so, we have learned about basic control structures there mainly the while do while and for loop with break and continue.

(Refer Slide Time: 09:03)



```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");
    }
}
```

```
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>cls
```

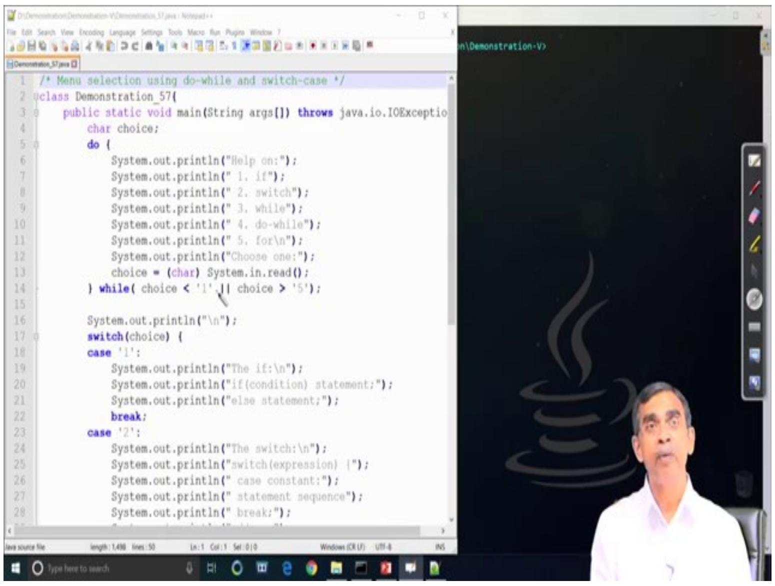
Now, in addition to the simple control structures there is basically a decision structure if else; if else is very simple we do not have to discuss it in a detailed manner, but like this is if int is a multiple decision checking is done in Java by means of the switch case. Now, we will discuss the next program in switch case; now here is one example before going to this to discuss the switch case statement is basically how we can test whether a number is prime or not.

So, it is basically usage of both for loop as well as if statement now for basically you have to test till a certain number is prime or not the condition is that i less than num 2 because, you have to check for this kind of things; that means, if any number within this num 2 if we have to test that whether num is a prime or not. So, up to half of these things we have to roll this loop. So, that we can check that any number within this range it is divisible by this number or not so, the divisibility check by num not i. So, it basically checks the divisibility.

That means it is divisible by this i where i is basically 2 to that range and then if it is there then it will check that ok, if it is divisible then it is not a prime. So, each prime is a Boolean variable declared a false and then is broken because no more testing is required otherwise it will continue. Now, we can test this program by passing input from the keyboard and here is the execution of this program capital so, 5.6 5.6. So, now, v 6 as it is on the 6 is prime now let us run this 31; 31is a prime so, it will check that prime.

So, like this so, you can test that using this control we can test this. So, logic is basically any logic can be implemented using for or while and then if else statement there another logic structures is there as I told you the switch case structure let us consider one simple program.

(Refer Slide Time: 11:23)

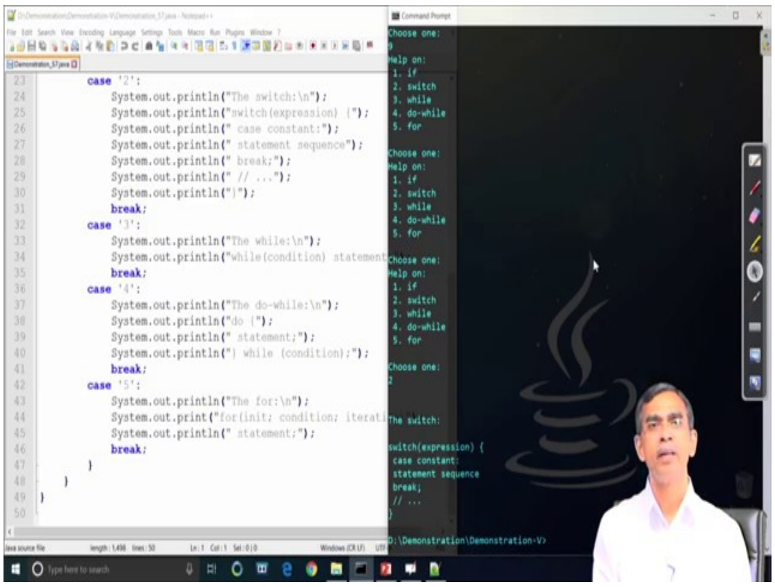


```java
/* Menu selection using do-while and switch-case */
class Demonstration_57{
    public static void main(String args[]) throws java.io.IOExceptio
        char choice;
        do {
            System.out.println("Help on:");
            System.out.println(" 1. if");
            System.out.println(" 2. switch");
            System.out.println(" 3. while");
            System.out.println(" 4. do-while");
            System.out.println(" 5. for\n");
            System.out.println("Choose one:");
            choice = (char) System.in.read();
        } while( choice < '1' || choice > '5');

        System.out.println("\n");
        switch(choice) {
        case '1':
            System.out.println("The if:\n");
            System.out.println("if(condition) statement;");
            System.out.println("else statement;");
            break;
        case '2':
            System.out.println("The switch:\n");
            System.out.println("switch(expression) {");
            System.out.println(" case constant:");
            System.out.println(" statement sequence");
            System.out.println(" break;");
```

And this switch case is basically here we can see. So, and we can see the first few blocks is there do while blocked it basically print something until you type 1 to 5 any number if you type anything. So, this loop will continue ok. So, in order to come out from this loop, you have to either print 1 or any number less than greater than less than 5 actually here.

So, if any number you type better than 5 it basically breaks the loop now let us see once the value of i chosen from here. That means the user can enter any value which is greater than or=1 to less than auricles to 5; now once the value is read and then there is a case.

(Refer Slide Time: 12:11)



```
            case '2':
                System.out.println("The switch:\n");
                System.out.println("switch(expression) {");
                System.out.println(" case constant:");
                System.out.println(" statement sequence");
                System.out.println(" break;");
                System.out.println(" // ...");
                System.out.println("}");
                break;
            case '3':
                System.out.println("The while:\n");
                System.out.println("while(condition) statement
                break;
            case '4':
                System.out.println("The do-while:\n");
                System.out.println("do {");
                System.out.println(" statement;");
                System.out.println("} while (condition);");
                break;
            case '5':
                System.out.println("The for:\n");
                System.out.print("for(init; condition; iterati
                System.out.println(" statement;");
                break;
        }
    }
}
```

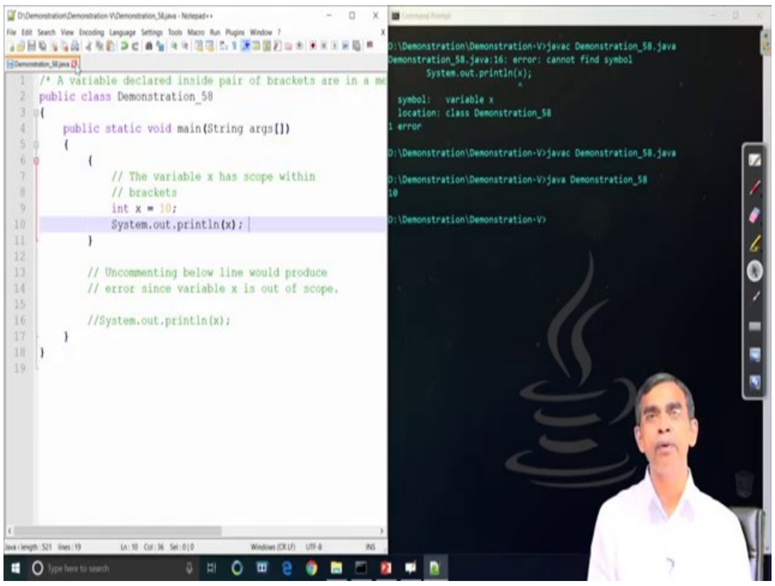
```
Choose one:
9
Help on:
 1. if
 2. switch
 3. while
 4. do-while
 5. for

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>
```

So, depending on the type the number that you have chosen we have a lot of switch cases is there; that mean if i that we have written in the first statement here the choice basically. So, it depends on the choice it will depend on the different states will be executed.

If choice=1 so, the case 1 code will be executed and if choice=2 the case 2 code will be executed. Similarly, for all 5 cases, the different statements are there in this program and you can see for. So, here basically switch based on the value of same switch variable here choice the different cases will be executed all cases are basically few courses are there. And, then all case statement will be a term at the end of this case statement there will be a break; that means after this case is where it will break that switch loop actually.

So, this is a structure of the switch case concept in Java program. Now let us run this program so, that you can understand how it will work. So, the advice is that you should have this code and then try to practice of your own; so, that you can understand how the codes are working. So, here now choice 1 if I press say 9, then you can understand what will happen it will loop so, it is asking this one; now again choice says 2. So, it basically tells about the switch. So, this is a way that the program can be executed. So, this is a simple example of the switch case statement.

(Refer Slide Time: 13:57)



```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);
    }
}
```

```
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>
```

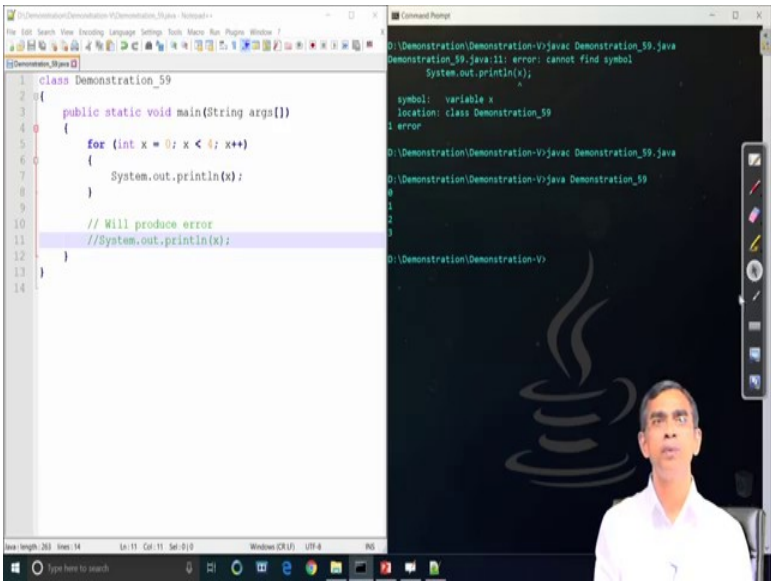
Now, so, these are the control structures which are widely used in your Java program and you have to learn it you have to practice it. So, that you should have very skill about all these programs; now once that things are there our next topics in this discussion demonstration is the usage of scope rule. As you have discussed that Java follows 2 scope rule static and dynamic so, in this demonstration, we will limit our discussion do static scope full only.

Now, let us have a quick look to this program and here we can define a class the name of the class is demonstration_58, these a name of the class and under this class only main method very simple class. And you can see them, for the main method the matching and then beginning curly brackets are there. In addition to this, we define a block in a program can be specified by means of {}. So, here the block it is there; now in this block inside this block, you see x=10 as an integer variable is declared.

Now so, far the scope rule is concerned the value of x is valid within this block only; that means, it is valid that system.out.print ln x is correct. However, outside this block if we try to access a system.out.print ln x then it will give an error and that errors can be reported during the compilation time. Now, let us change this the last statement system.out.print ln x; that means, we are trying to print it although it does not have the scope now save the program and run this program and let us see what will happen. So, it will give an error because System.out.prinln(x) the last statement in this class has out of scope the x does not. So, it is basically a compilation error.

Now, again if we make it commenting the program and then we will see this program a will be compiled successfully and also executable save this program ok. So, the program is now compiled successfully earlier it was giving an error now the program will give the result that it is x is=10. So, you can understand the scope it is a very simple example, but it is understandable that: what is a scope of a variable inside a block. So, this is the scope in a block.

(Refer Slide Time: 16:39)



```java
class Demonstration_59
{
    public static void main(String args[])
    {
        for (int x = 0; x < 4; x++)
        {
            System.out.println(x);
        }

        // Will produce error
        //System.out.println(x);
    }
}
```

```
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>javac Demonstration_59.java

D:\Demonstration\Demonstration-V>java Demonstration_59
0
1
2
3

D:\Demonstration\Demonstration-V>
```

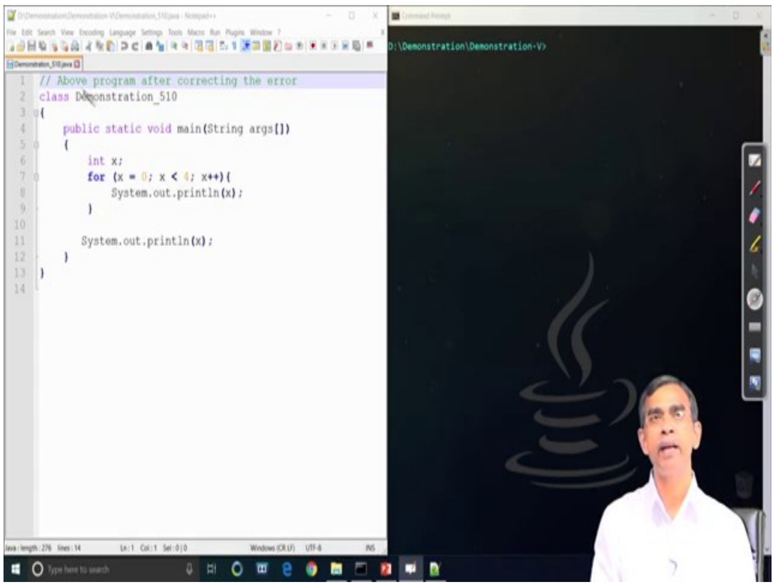
Now there is a few more example. This is another example, you can see and how you can understand also what is the scope of this example. Now can you tell me what is the scope of this example; where x is the variable for which we want to discuss the scope.

Now, if you see here int x=0 it is declared within the for the loop this means that this is the scope of x within the for loop only. So, within the for a loop, there is a print statement which prints the value of x it is, but outside of this, for loop, if you attempt to print it will give an error. So, let us uncomment this one and then run this program then you will be able to understand that it will give a compile-time error because here the statement it is out of scope.

So, here the x the scope of these values variable x is within the for this loop. So, it is an error is the definition that the scope is not there; now again comment it run the save the program and then run this program here no compilation error and run it will print basically for loop will be executed successfully in this case.

So, for loop is executed you can see it. Now how you can change the scope rule now let us come to another example so, that we can see how the scope can be increased here. So, is the next example similar to the last example.

(Refer Slide Time: 18:01)

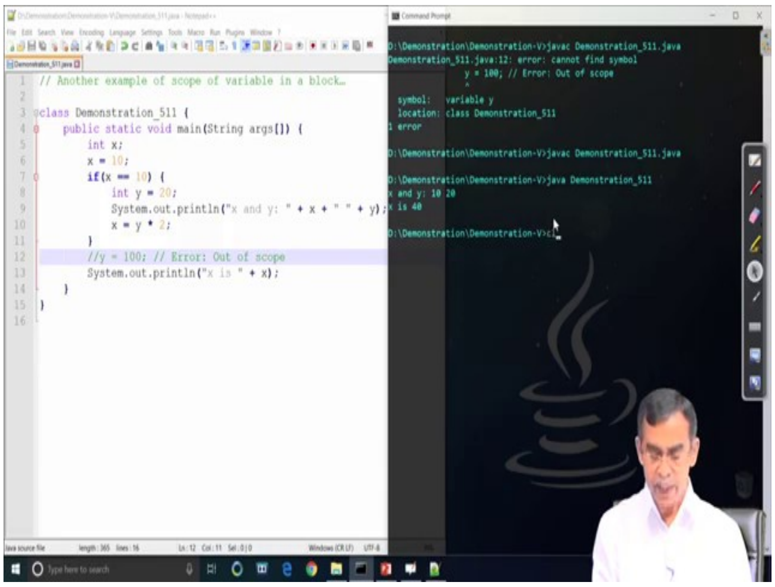


```
// Above program after correcting the error
class Demonstration_510
{
    public static void main(String args[])
    {
        int x;
        for (x = 0; x < 4; x++){
            System.out.println(x);
        }

       System.out.println(x);
    }
}
```

```
D:\Demonstration\Demonstration-V>
```

Here you can see int x which declared outside the for the loop this means that the value of x is spread throughout the for loop as well as outside this for loop also. Earlier when it was declared within the for it was only scoped within the for, but in this case, the scope is the entire program. So, this program is self-explanatory as we have already understood about it. So, this is about the simple concept of scope and fine. So, let us discuss one more example which has the different contexts of the scope let us run the program 5 points 5.11 yeah fine.

(Refer Slide Time: 18:47)



```java
// Another example of scope of variable in a block…

class Demonstration_511 {
    public static void main(String args[]) {
        int x;
        x = 10;
        if(x == 10) {
            int y = 20;
            System.out.println("x and y: " + x + " " + y);
            x = y * 2;
        }
        //y = 100; // Error: Out of scope
        System.out.println("x is " + x);
    }
}
```

```
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>javac Demonstration_511.java

D:\Demonstration\Demonstration-V>java Demonstration_511
x and y: 10 20
x is 40

D:\Demonstration\Demonstration-V>cl
```

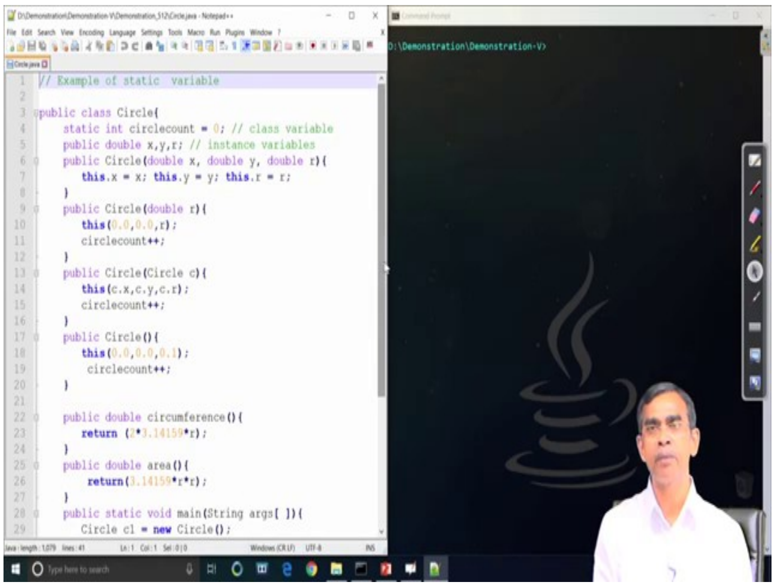
So, this is the one program; now let us quickly examine this program watch this program little bit here x and y are the 2 variables declared and the 2 variables have a different scope. So, for the scope of x is concerned as it is declared in the main method this means that the scope of this variable x is the entire method.

So, when you have from this method the value can be accessed on the other hand. If we see the int y 20 which is declared within the block with under this block under a; that means, the scope of this y is within this block. So, in the statement after System.out.println() x and y the next statement is valid because it will be within this scope. However, outside this y=100 it will give an error.

Now let us run this program and then we can see that it is giving the compile-time error. So, all these errors will be reported during the compile time; that means, Java compiler will check the scope of all variables for you. So, if there is an error in the scope then it will report it the competition will not be successful. So, this is the compile-time error we can say.

Now, if we comment it then definitely because the scope is now as closed and then it is it will work perfectly. So, here no compilation error program will give the output according to the program ok. So, we can understand the concept of scope here. Now, there are many more things about the scope we have discussed the global variable idea in Java program, Java does not support the declaration of the global variable.

(Refer Slide Time: 20:37)



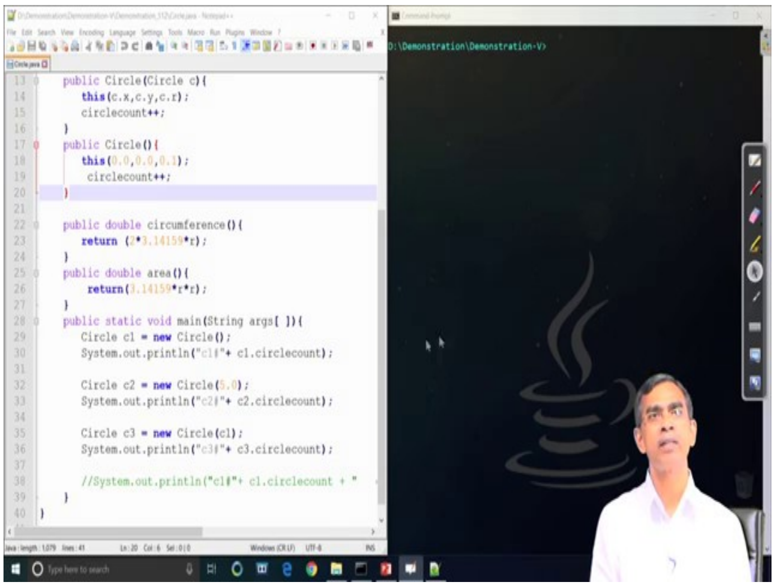
```
// Example of static  variable

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();
```

```
D:\Demonstration\Demonstration-V>
```

However, the concept of global variable is by means of the concept of a class variable; a class variable and then instance variable as we have already learned about it. Here, for example, x y r these are the instance variable because they are declared as simple as a type; on the other hand if we declare a variable with a skiver static then it is called the class variable, in this case, we can see the circle count is an example of a class variable.

So, we have declared here one class variable namely circle count and then three instance variable x y r and these are the usual code of the constructor of the class circle. So, we have already discussed is I do not want to discuss it more and the two methods circumference error as usual.

(Refer Slide Time: 21:27)



```
    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();
        System.out.println("c1#"+ c1.circlecount);

        Circle c2 = new Circle(5.0);
        System.out.println("c2#"+ c2.circlecount);

        Circle c3 = new Circle(c1);
        System.out.println("c3#"+ c3.circlecount);

        //System.out.println("c1#"+ c1.circlecount + "
    }
}
```

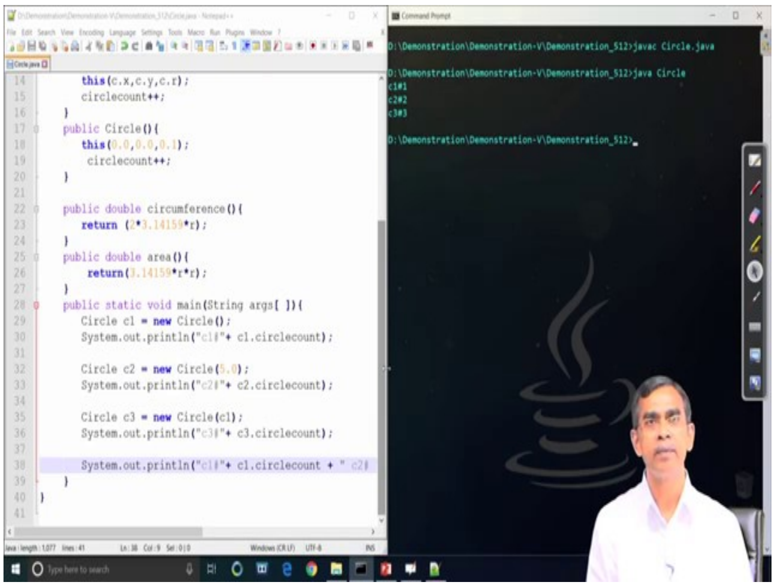
```
D:\Demonstration\Demonstration-V>
```

Now, let us come to the main method here; now you have to just watch the main method a little bit carefully. So, in the first statement, we create an object of the class circle the name of the object is c 1 and it will print the c 1.circle count. Now here once the object c 1 is created it will call its constructor. Now, here it is the default constructor. So, if you go to the default constructor then you see in the default constructor here in the last constructor that we can see. So, it will basically it will initialize all the instance variable as 0 0.1 0.00 and 0 point 1 x y and r and the circle will be increased by 1.

Initially when we create when we define this circle count be 0. So, the global variable is initialized as 0 and after this whenever the constructor is called for this it will increment this one. So, now, let us come to the main method here again yeah. So, once the circle c 1 is created so, c 1 has the value circle count this means that if we print in this it will print the value 1. On the other and next c 2 create an object. Now it will call another constructor 5.0 which is the second constructor in the line of different constructors are here.

So, public circle double r here also circle count++ that mean it will increase the circle count by 1. So, after the c 2 object is created in the main so, the circle count becomes 2. So, it will print the 2 and similarly circle c this is the 3rd constructor the 1st concert I should say it will be invoked to create the circle class object c 3 and then c 3. And, then circle count, in this case, will be again in increased by 1 and then here the fine right circle c this is the 2nd constructor right this one now in the 2 now let us run this program and then you will be able to see the output.

(Refer Slide Time: 23:33)



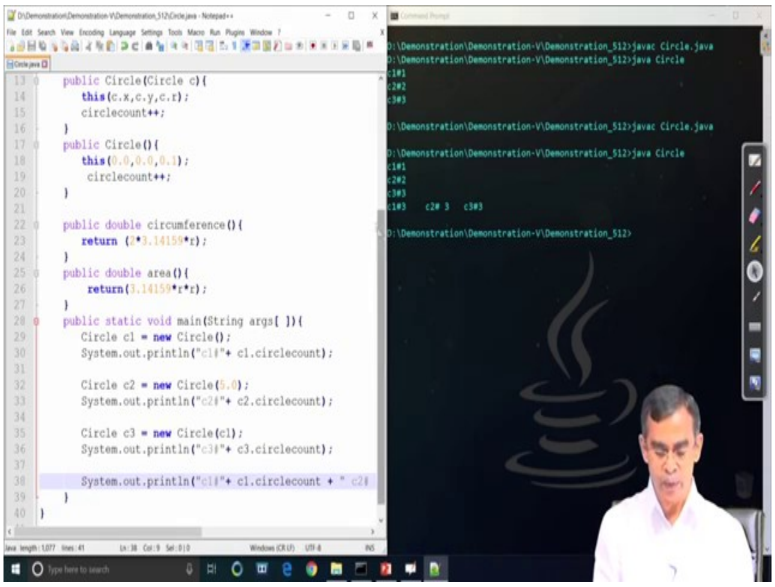
```
        this(c.x,c.y,c.r);
        circlecount++;
    }
    public Circle(){
        this(0.0,0.0,0.1);
         circlecount++;
    }

    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();
        System.out.println("c1#"+ c1.circlecount);

        Circle c2 = new Circle(5.0);
        System.out.println("c2#"+ c2.circlecount);

        Circle c3 = new Circle(c1);
        System.out.println("c3#"+ c3.circlecount);

        System.out.println("c1#"+ c1.circlecount + " c2#
    }
}
```

```
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>_
```

So, where for each circle an object is created once the circle object is created the static variable will be processed according to the construction of the object and then it will basically run it will print the current value of this one. And here we can understand that the circle count is a global variable look like for all instances of the objects there are only 1 instance of this variable. So, here we can see the running of the output is basically 1 2 3.

Now let us switch to the program again and you can see we are just giving to the last statement which has been commented here, I just removing the comment here let us uncomment this statement yeah up uncomment this statement fine. Now, comment it now little bit if bigger the window here fine yes now if we see the right.

(Refer Slide Time: 24:33)



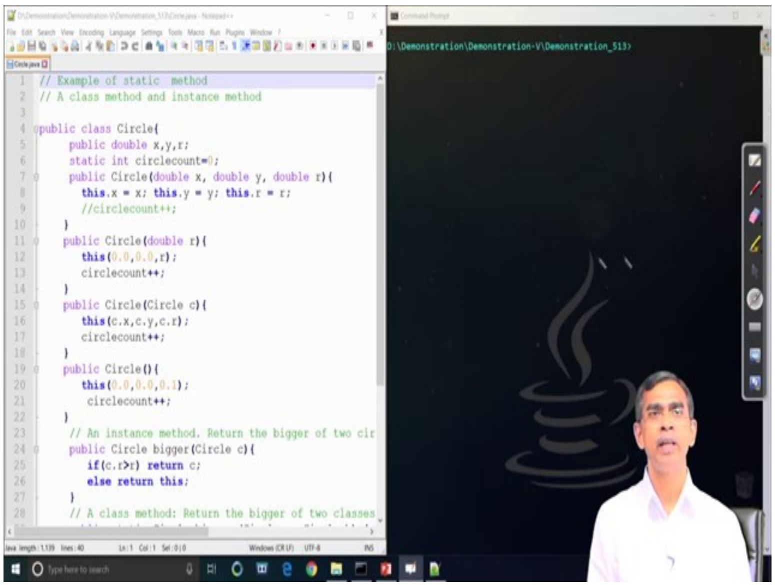
```java
    public Circle(Circle c){
       this(c.x,c.y,c.r);
       circlecount++;
    }
    public Circle(){
       this(0.0,0.0,0.1);
        circlecount++;
    }

    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();
       System.out.println("c1#"+ c1.circlecount);

       Circle c2 = new Circle(5.0);
       System.out.println("c2#"+ c2.circlecount);

       Circle c3 = new Circle(c1);
       System.out.println("c3#"+ c3.circlecount);

       System.out.println("c1#"+ c1.circlecount + " c2#
    }
}
```

```
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>
```

So, after creating this c 1 circle right one and we are now in this statement now can you tell me what exactly the value that it will print. So, here in the first object circle count has been created 1 2 3 and then when you are come to here. So, the circle count for this c 1 is 3 and again 3 and 3. So, basically, the latest values of the circle count will be accessed. So, it basically indicates that let us say the compile this program and run it. So, here in the last statement, we will print the value of circle count which is the ultimate value after the end of this program.

So, here you can see it printing the 3 3 3 this is the latest value after ah. So, it has basically one instance and then one it is updated by any object it will be reflected any other object which basically has the access to this value. Now so, this is the idea about the static variable concept likewise there is a concept of class method and instance method our next program will illustrate the concept of these two methods in a Java program the class method and an instance method.

(Refer Slide Time: 25:45)



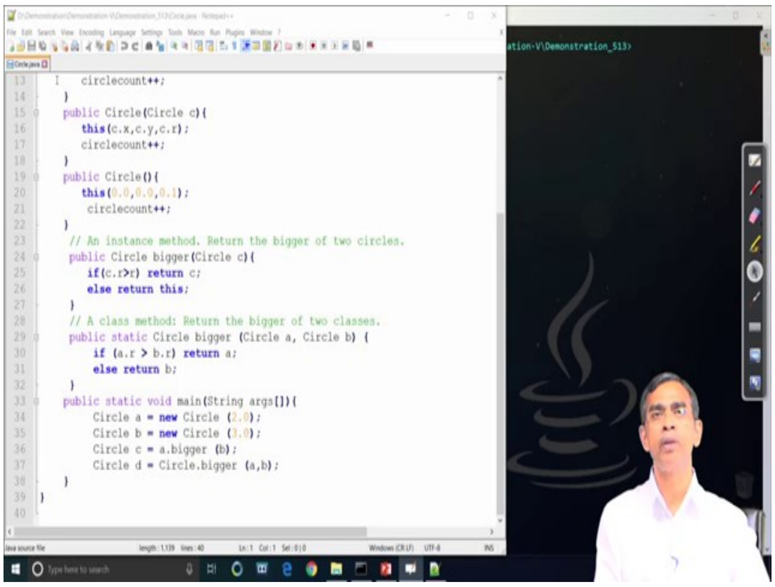
```
// Example of static  method
// A class method and instance method

public class Circle{
      public double x,y,r;
      static int circlecount=0;
      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 cir
      public Circle bigger(Circle c){
         if(c.r>r) return c;
         else return this;
      }
      // A class method: Return the bigger of two classes
```

```
D:\Demonstration\Demonstration-V\Demonstration_513>
```

We have already discussed that a method is an instance method in order to execute this method an object is required. So, via an object, the method will be called. On the other hand, the class method is a method which does not require any object to be created the method can be called itself.

So, this example basically to illustrate the concept of class method and instance method; now let us here all the circles are basically instance method because, whenever you have to create an object all these constructors will be called automatically, say they are by virtue of default that the instance method. Now we declare one method let us go here the circle figure public circle bigger the 2nd years. Now, this method little bit bigger the window yes yeah so, fine.

(Refer Slide Time: 26:43)



```
        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);
         Circle b = new Circle (3.0);
         Circle c = a.bigger (b);
         Circle d = Circle.bigger (a,b);
    }
}
```

Now, is the 1st method we can see bigger. So, this is the method public circle bigger circle c. So, the method will basically argument will be an object circle is the c.r greater than r return c; that means, that c is a bigger circle else return these that mean the current circle. So, this is the one simple method and this is called the instance method.

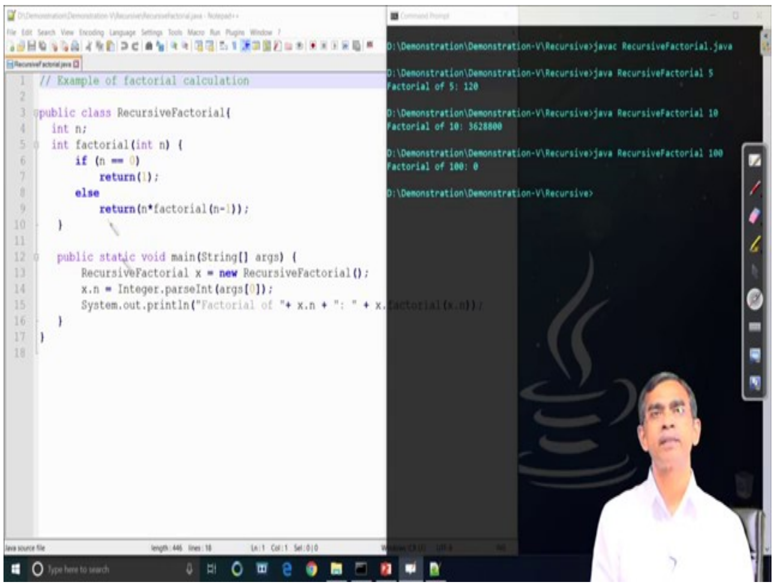
On the other hand another method bigger is an overriding method we have discussed, but this method is different from the previous method by two things; one is called the static keyword the 2nd one has the static; this indicates that this method bigger is basically the class method and also it is different from the argument point of view. Earlier only one argument whereas, this is the two arguments all arguments are of type circle type of objects.

The code is like this is a.r greater than b.r return a then a is a bigger circle else return b. So, it will basically return that is a return type is a circle here in this case. Now, let us see the main method here we create an object 2 object, 3 objects, 4 objects here of course. Anyway so, 1st two objects are type class circle a b and then here we can see the b a.bigger. So, basically, a bigger method is invoked with reference to the object a. So, it is an instance method, on the other hand, the second call.

So, the bigger method is called with reference to an object it is a circle in terms of circle.bigger so; that means, without creating any object this method is called. So, this is an example of invoking the class method. So, here the two invocations the instance method and class method and it indicates that the way of the class method works differently than the instance method like this.

So, so this is an example of an instance method and class method ok. So, these are the main concept those are there in java so, far the static scope rule is concerned. Now, let us switch to our demonstration to give the execution of recursive program we have already discussed while we are discussing the theory is that recursion recursive program writing in Java. So, let us first run the recursive calculation recursive way calculating the factorial.

(Refer Slide Time: 29:09)



```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));
   }
}
```

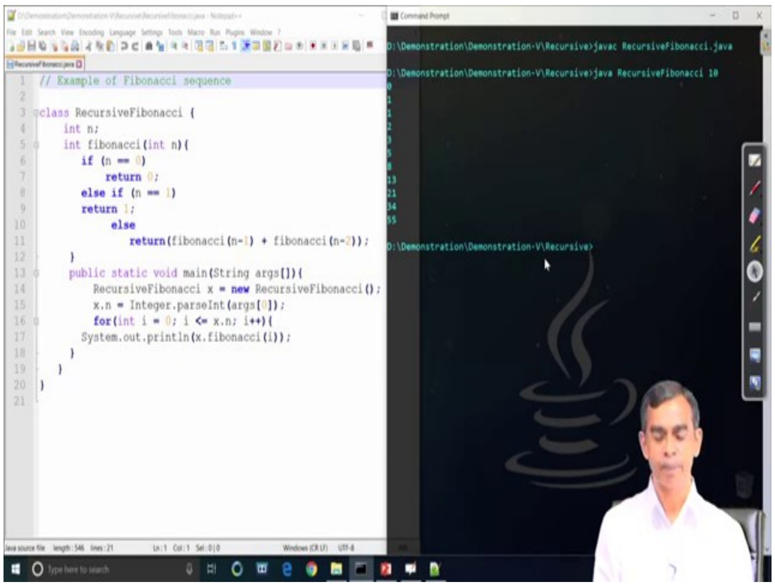
```
D:\Demonstration\Demonstration-V\Recursive>javac RecursiveFactorial.java

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>
```

So, this program is well understood I believe. So, here factorial is a method which is defined in the class recursive factorial and then recursion is basically following the recursive definition of factorial calculation the code is like this. Very simple n factorial=n star factor n minus 1 with termination conditions that 0 factorial=1.

So, this is the implementation of this factorial definition of n; now in the main method we create an object of type of the class recursive factorial here and then pass the value to this subject from the keyboard as an input and then we call this x.factorial x.n; that means, then we call the method in these subject of the class recursive factorial and run it. Now let us run this program quickly you can understand that how it will run recursive fine.

Now, as it is an input to be passed because r 0 is there. So, we should give the input say 5 right if we run this program again with some say larger value say 10 it will also execute for you yeah. So, it is like this, but it cannot take a very large number say for example if we say 100 you will see whether your program is now. So, a factor of 100 if 0 it is giving it is because it is out of the pound of this one. So, in that case, an integer has its own limit. So, if we declare say long integer instead of int it may take some larger values anyway changing this we can change this one, but we have to changes for that it is later then exercised for you. Now, let us come to another recursive program to print the Fibonacci sequence.

(Refer Slide Time: 31:01)



```
// 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));
     }
   }
}
```

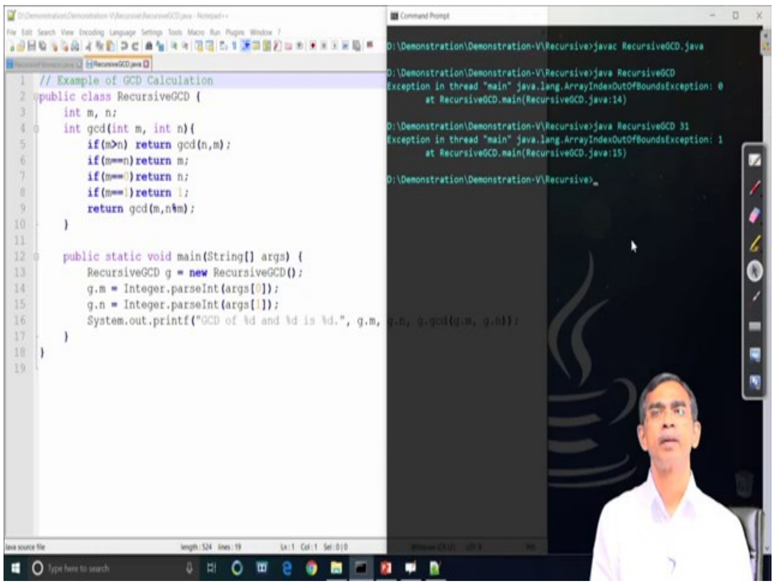
```
D:\Demonstration\Demonstration-V\Recursive>javac RecursiveFibonacci.java

D:\Demonstration\Demonstration-V\Recursive>java RecursiveFibonacci 10
0
1
1
2
3
5
8
13
21
34
55

D:\Demonstration\Demonstration-V\Recursive>
```

So, this is the program we have already familiar with the program that we have discussed in our last module and. So, here again, the recursive definition of an nth factorial is basically n minus 1 factorial plus n minus 2 factorial and then termination condition is that 0th factorial=0 and then 1 factorial is 1. So, these are the 2 termination condition followed by the recursive definition it is the same as writing the factorial we create an object of type recursive.

Fibonacci here is the class where we have defined the recursive method Fibonacci and we call this and this for loop is basically print in succession all the Fibonacci number till the x.n this is the user different number that up to which the Fibonacci number you want to print.

Now, suppose you want to print up to the tenth Fibonacci number. So, we can run this program with passing input; here input needs to be passed through the keyboard let it be 10 and you can understand that how it will print 10 Fibonacci numbers in the Fibonacci series 10 ok. So, it prints the first 10 Fibonacci numbers in the few Fibonacci series. So, this basically the recursive now let us come to the GCD calculation that we have discussed in our theoretical class.

(Refer Slide Time: 32:25)



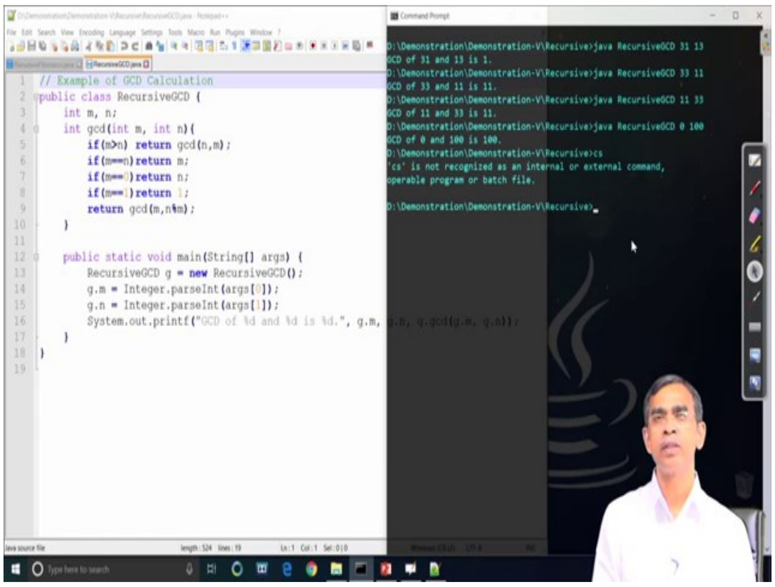
```java
// Example of GCD Calculation
public class RecursiveGCD {
    int m, n;
    int gcd(int m, int n){
        if(m>n) return gcd(n,m);
        if(m==n)return m;
        if(m==0)return n;
        if(m==1)return 1;
        return gcd(m,n%m);
    }

    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));
    }
}
```

```
D:\Demonstration\Demonstration-V\Recursive>javac RecursiveGCD.java

D:\Demonstration\Demonstration-V\Recursive>java RecursiveGCD
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 0
        at RecursiveGCD.main(RecursiveGCD.java:14)

D:\Demonstration\Demonstration-V\Recursive>java RecursiveGCD 31
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 1
        at RecursiveGCD.main(RecursiveGCD.java:15)

D:\Demonstration\Demonstration-V\Recursive>_
```

So, this GCD calculation basically has this kind of recursion rule is like this one and we have implemented there is a few more termination condition. Now, you can see in case of recursive factorial only one termination in case of Fibonacci there is two termination condition. Whereas in case of GCD is a couple of termination condition there and all termination condition followed by the recursive call so, in this case, recursive call gcd mn mod m. So, this is as per the recursive definition of this one.

Now here two integer values are to be passed and then Java program will call for this recursive function which is defined here as a gcd and we call the gcd with the two value passed to it and it will call. So, g is an object of the type this class is created and call this function and then it will execute a let us run this program so, that you can understand about its execution ok. First, let us see enter 31 and then 13. So, this is 2 input you have to give it give 2 input 31 and 13 13.

(Refer Slide Time: 33:39)



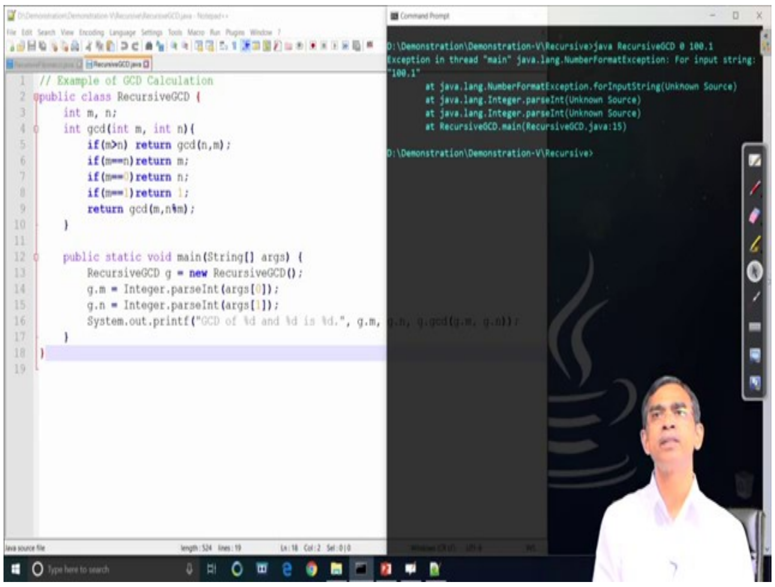
```java
// Example of GCD Calculation
public class RecursiveGCD {
    int m, n;
    int gcd(int m, int n){
        if(m>n) return gcd(n,m);
        if(m==n)return m;
        if(m==0)return n;
        if(m==1)return 1;
        return gcd(m,n%m);
    }

    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));
    }
}
```

```
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>cs
'cs' is not recognized as an internal or external command,
operable program or batch file.

D:\Demonstration\Demonstration-V\Recursive>
```

Now you can see that GCD is 1; now let us run this program with say 33 and 11 you can understand that the gcd that value it will give you right. Even if you give you 11 and 33 also it will work yeah, now if we give 0 and say 100 you can understand that what is the gcd of this number 100 so, it will print the 100.

Now, if you give. So, this is the way that the GCD and with this function is a call for integral value only you should pass always integer if you give some non-integer value it will report an error.

(Refer Slide Time: 34:25)

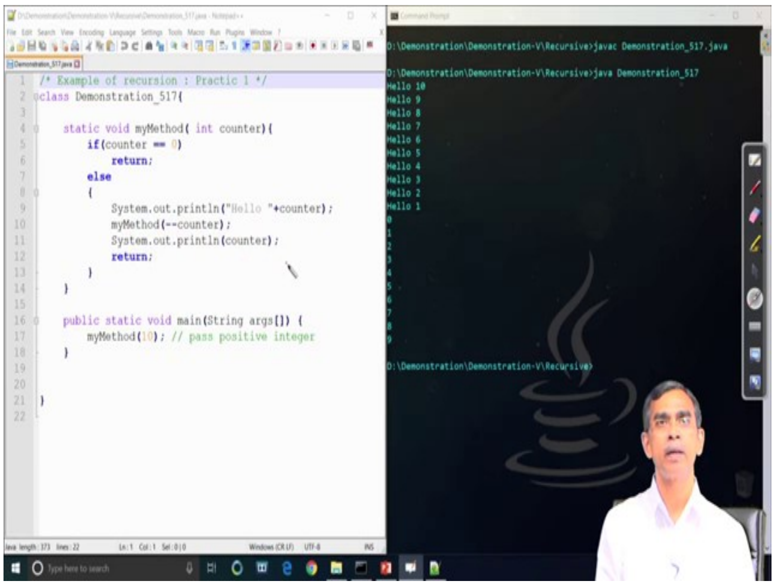


```java
// Example of GCD Calculation
public class RecursiveGCD {
    int m, n;
    int gcd(int m, int n){
        if(m>n) return gcd(n,m);
        if(m==n)return m;
        if(m==0)return n;
        if(m==1)return 1;
        return gcd(m,n%m);
    }

    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));
    }
}
```

```
D:\Demonstration\Demonstration-V\Recursive>java RecursiveGCD 0 100.1
Exception in thread "main" java.lang.NumberFormatException: For input string:
"100.1"
        at java.lang.NumberFormatException.forInputString(Unknown Source)
        at java.lang.Integer.parseInt(Unknown Source)
        at java.lang.Integer.parseInt(Unknown Source)
        at RecursiveGCD.main(RecursiveGCD.java:15)

D:\Demonstration\Demonstration-V\Recursive>
```

For example, if we run this program passing say non-integral value it is not acceptable to the run time in, but as in Java run time invertise say giving an error. So, it is basically you have to give the integral value always. Now so, we have learned about factorial calculation; so, factorial calculation, Fibonacci series calculation, GCD calculation like this one. So, the most important thing that you should understand how you can cast a program by following the recursive definition.

(Refer Slide Time: 34:53)



```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
    }


}
```

```
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>
```

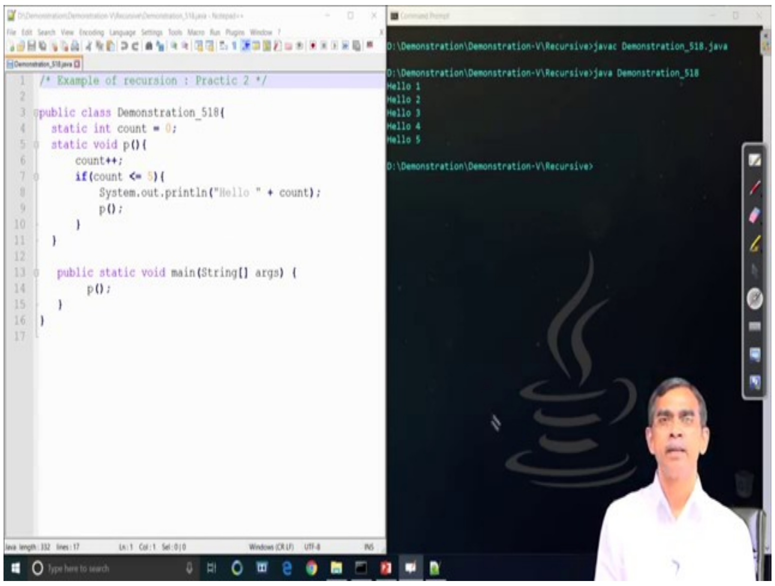
If every program has its loop all this program has its counterexample of recursion actually. So, if a program needs to be solved by means of some looping construct then the same program can be solved by means of loop recursive version also. Now let us have a quick look of this simple example you can understand this is the program we have declared recursively. So, it basically here is my method is a recursive method which is declared in the class demonstration under 517 and the method has the termination and one important thing is that every recursive function should have termination statement.

Without termination, the recursion will go on infinite execution never terminate this is not desirable. So, every recursive program should have in this case we can see that counter recursive is 0. So, along counter is not=0 so, along counter is not=0 it will go on counting and in this case my method whenever it will call first time it will print the value of count and then my method call will be called again with reducing the value of counts. So, if we pass say counter value it will call subsequently less counter so on so on.

And then print all the 10 counter and when the recursion is over it will come to the previous counter value. So, now, if you see whenever before recursive call it will pin and then on recursion, it will pin some values and go on printing. And then let us see exactly: what is the output of this call. So, if we call this recursion with 10 as a value as counter how it will do it; I can explain that why this output is where you can also explain that how this output is happened to this program.

You can see here the two series of statement because of the two print statement and then recursion is going on printing this one and then recursive call and when the recursion is back from the loop, it is again starts this one. So, there are two series of the statement and the recursion is the there actually recursion exhibition for use a stack it basically before terminating recursion it goes to the call of the recursion so on. So, that is why it is there. So, this is the one example now let us have the quick look of another example for your own practice.

(Refer Slide Time: 37:11)



```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();
   }
}
```

```
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>
```

So, again you can learn it from these series in the same line of the previous example one; you can guess that what input it should give it again p is the method which is defined here recursively. Now let us run this program on it after the execution of this program with the value of p as 5 here we have do not have to value 5 it will print this one.

Anyways so, regarding the scope, the control structures and then recursive program writing is a matter of practice. So, I advise you to practice more and more programs in this line so, that you can learn it. For your own practice, if you need the program all those courses that we have given here a demonstration you can have it, you just send me to send us a request. So, that we can send it to you, thank you very much and have fun for the Java programming more.

Thank you.