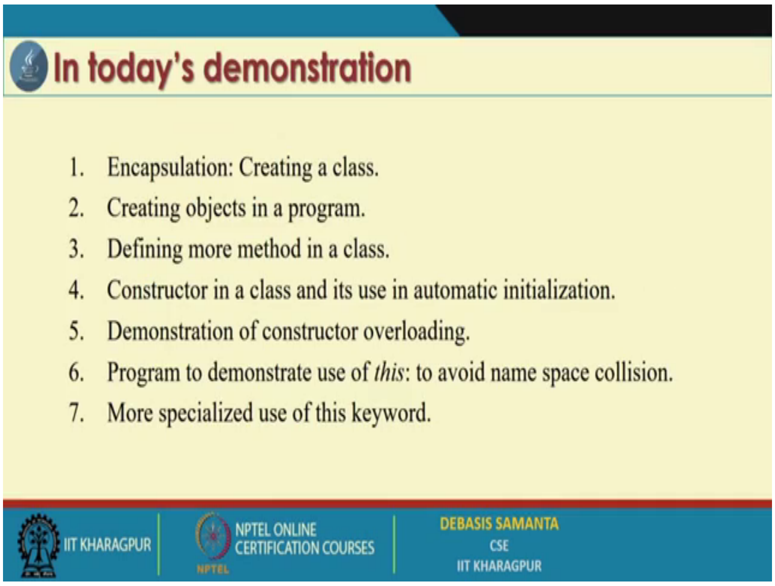
Programming In Java
Prof. Debasis Samanta
Department of Computer Science Engineering
Indian Institute of Technology, Kharagpur

Lecture – 08
Demonstration – III

So, in the last module you have learned about encapsulation. Encapsulation in fact, is a very important concept is a very important object oriented paradigm. So, after learning basic concept of encapsulation so, today's we have planned a demo where we learn about different aspects in encapsulation.

(Refer Slide Time: 00:39)



In today's demonstration

1. Encapsulation: Creating a class.
2. Creating objects in a program.
3. Defining more method in a class.
4. Constructor in a class and its use in automatic initialization.
5. Demonstration of constructor overloading.
6. Program to demonstrate use of *this*: to avoid name space collision.
7. More specialized use of this keyword.

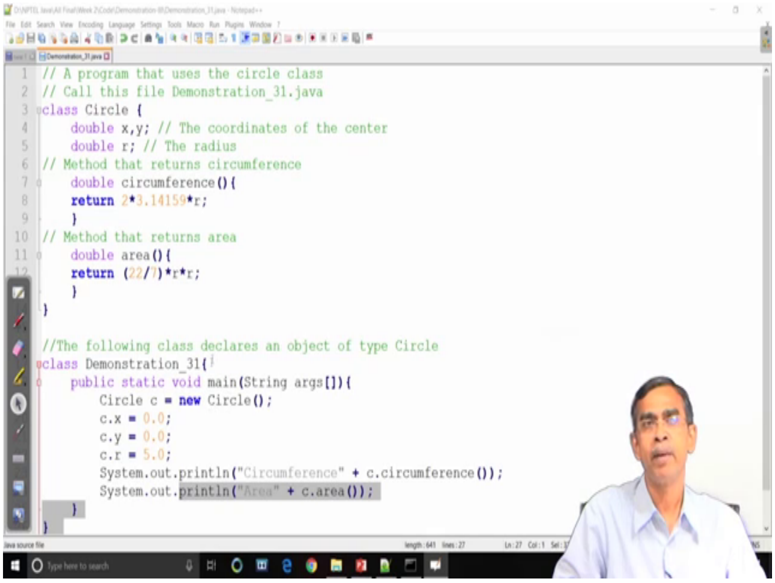
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NPTEL
DEBASIS SAMANTA
CSE
IIT KHARAGPUR

So, our today's demonstration include how we can create a class and then for a class, how the different objects can be created and then how we can add many methods in a class so, that a class can take its completes form.

An important concept in encapsulation which is very essential for any program is that automatic object initialization. So, this concept is achieved in java by means of constructor. So, there is a concept of constructor will be demonstrated and also we will discuss about constructor overloading. So, it is a one part of the polymorphism in java concept so that many constructors can be plan. So, that an object can be initialized in a different ways and there are some concept that is very much used to resolve certain collision, it is called the name space collision resolution.

So, this is achieved by means of a special java keyword call the; this. So, we shall learn about this in this lecture. So, let us have the demo. So, first we will discuss about how we can create a class right. We will discussed a program very small program; let us have a very small program.

(Refer Slide Time: 02:05)



```
// A program that uses the circle class
// Call this file Demonstration_31.java
class Circle {
    double x,y; // The coordinates of the center
    double r; // The radius
// Method that returns circumference
    double circumference(){
    return 2*3.14159*r;
    }
// Method that returns area
    double area(){
    return (22/7)*r*r;
    }
}

//The following class declares an object of type Circle
class Demonstration_31{
    public static void main(String args[]){
        Circle c = new Circle();
        c.x = 0.0;
        c.y = 0.0;
        c.r = 5.0;
        System.out.println("Circumference" + c.circumference());
        System.out.println("Area" + c.area());
    }
}
```

In our theoretical discussion, we have discussed about one class called the circle. So, here we can see this is the; where is the highlighter. So, here we see we declare the class Circle here.

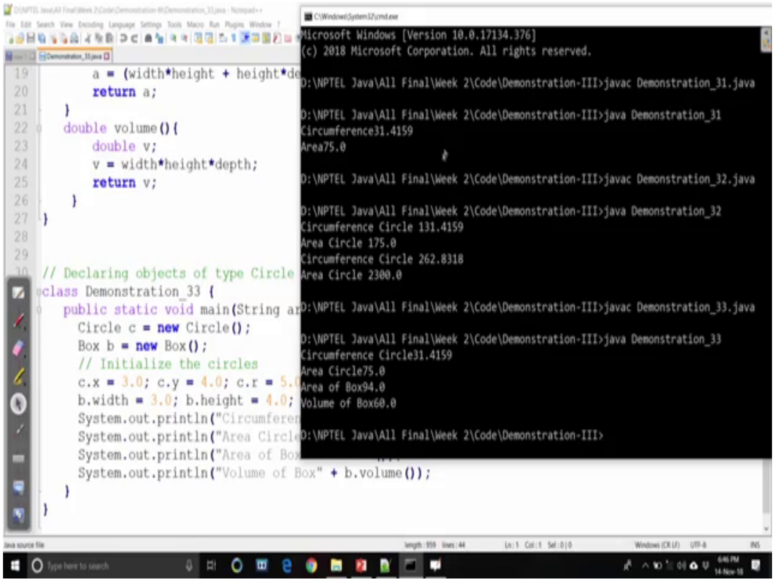
So, this class Circle has 3 data namely x, y and r all three data's are declared as double. In addition to this data also this class has two methods; one is circumference and another is area. So, the two methods are defined here. So, this completes the definition of a class called circle. Now, once this class is declared you can keep this class in a separate file or in the same file where the main class will include. Now here in the following we discuss the main class which is saved in the same file. The name of the main class in this case, is demonstration 31. Now in this class you see how we can create an object of the class type c circle.

So, we create an object the name of the objective c and to create an object, you can note it the new operator which is used here. So, new and then again Circle and then within the parentheses, it is basically the general syntax to declare an object. So, with this statement we declare an object called the c the object is created; however, the different data that is there for this object is yet to be initialized. So, in the following three statement we initialize the members x, y and r with the values 0.0, 0.0 and 5.0.

So, this completes the creation of an object, in this case the name of the objective is c. Now once this object is created, we can access any values in this object. So, in the next statement we just painting the circumference of the Circle that we have created. Now in order to access the method circumference, you note that we use c dot circumference. So, this means that we call the method circumference of the object c. In the next statement again, we call the method area of the object c. This means in this statement, we shall be able to print the area of the Circle which we have just now created

Now, let us have the demo. We will compile we will save this program as here the name of the main class is Demonstration underscore 31. We should save this file as the Demonstration underscore 31 dot java and then we can compile using java of c. So, let us see the compilation java c demonstration ok.

(Refer Slide Time: 05:27)



```
        a = (width*height + height*de
        return a;
    }
    double volume(){
        double v;
        v = width*height*depth;
        return v;
    }
}


// Declaring objects of type Circle
class Demonstration_33 {
    public static void main(String ar
      Circle c = new Circle();
      Box b = new Box();
      // Initialize the circles
      c.x = 3.0; c.y = 4.0; c.r = 5.
      b.width = 3.0; b.height = 4.0;
      System.out.println("Circumferen
      System.out.println("Area Circle
      System.out.println("Area of Box
      System.out.println("Volume of Box" + b.volume());
    }
}
```

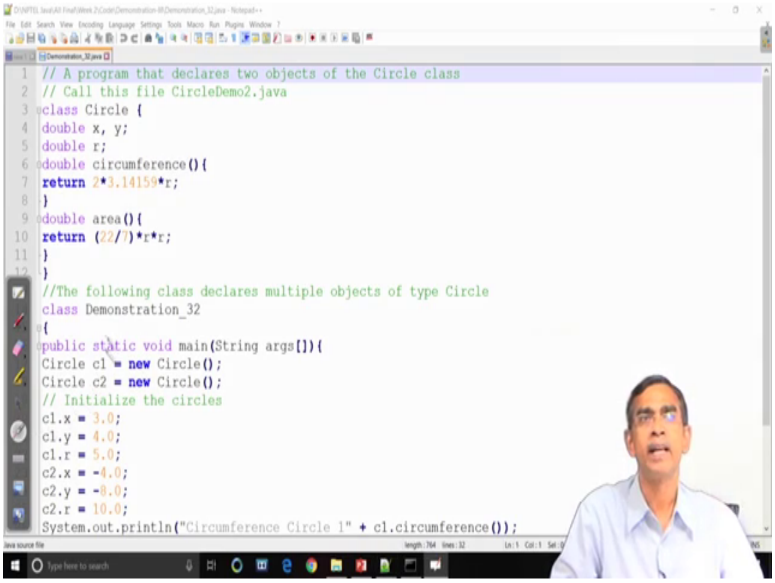
```
Microsoft Windows [Version 10.0.17134.376]
(c) 2018 Microsoft Corporation. All rights reserved.

D:\NPTEL Java\All Final\Week 2\Code\Demonstration-III>javac Demonstration_31.java

D:\NPTEL Java\All Final\Week 2\Code\Demonstration-III>java Demonstration_31
Circumference31.4159
Area75.0

D:\NPTEL Java\All Final\Week 2\Code\Demonstration-III>javac Demonstration_32.java

D:\NPTEL Java\All Final\Week 2\Code\Demonstration-III>java Demonstration_32
Circumference Circle 131.4159
Area Circle 175.0
Circumference Circle 262.8318
Area Circle 2300.0

D:\NPTEL Java\All Final\Week 2\Code\Demonstration-III>javac Demonstration_33.java

D:\NPTEL Java\All Final\Week 2\Code\Demonstration-III>java Demonstration_33
Circumference Circle31.4159
Area Circle75.0
Area of Box94.0
Volume of Box60.0

D:\NPTEL Java\All Final\Week 2\Code\Demonstration-III>
```

So, we just compile it as there is no error. So, compilation is successful, we can run this program using the java comment java is an interpreter. So, run this program and you see running this program and then circumference here in the Circle that is calculated as 39.4159 and the area is calculated 75.

Now, we have learned about how encapsulation is possible here in the form of encapsulation, we create a class Circle and for this class we created an object. Now here question is arised that whether we can create only one object in one program or many objects in a program. The answer is that multiple objects can be created for one class even multiple objects can be created for other classes also many classes also.

Now, in our next demonstration, we will see how for the same class say Circle the multiple circles can be created and let us have the demo here.

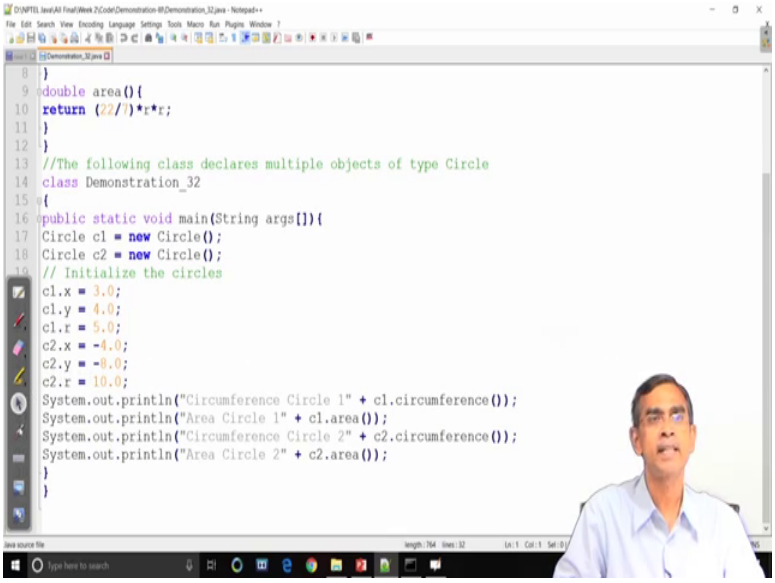
(Refer Slide Time: 06:41)



```
// A program that declares two objects of the Circle class
// Call this file CircleDemo2.java
class Circle {
double x, y;
double r;
double circumference(){
return 2*3.14159*r;
}
double area(){
return (22/7)*r*r;
}
}
//The following class declares multiple objects of type Circle
class Demonstration_32
{
public static void main(String args[]){
Circle c1 = new Circle();
Circle c2 = new Circle();
// Initialize the circles
c1.x = 3.0;
c1.y = 4.0;
c1.r = 5.0;
c2.x = -4.0;
c2.y = -8.0;
c2.r = 10.0;
System.out.println("Circumference Circle 1" + c1.circumference());
```

Again we use the same declaration of the Circle class here. It is as usual in the earlier one. Now only we change the main program main class. So, name this main class as Demonstration underscore 32. Now here we can see, we have created two objects c 1 and c 2. In the earlier example we have created only one object, here we have created two objects and for these two objects c 1 and c 2. We initialize its data values namely for the first object x y and r are initializes three 3.0, 4.0, 5.0.

(Refer Slide Time: 07:19)



D:\NPTEL Java\All Final\Week 2\Code\Demonstration-III\Demonstration_32.java - Notepad++

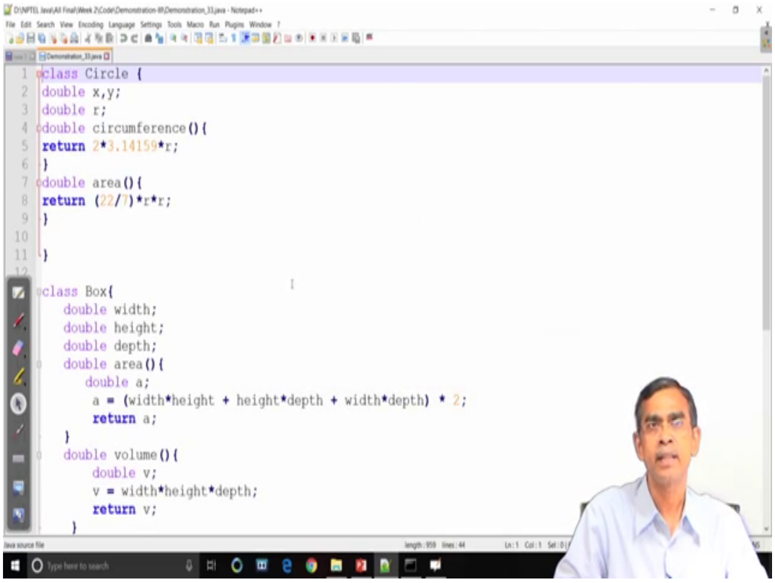
```java
}
double area(){
return (22/7)*r*r;
}
}
//The following class declares multiple objects of type Circle
class Demonstration_32
{
public static void main(String args[]){
Circle c1 = new Circle();
Circle c2 = new Circle();
// Initialize the circles
c1.x = 3.0;
c1.y = 4.0;
c1.r = 5.0;
c2.x = -4.0;
c2.y = -8.0;
c2.r = 10.0;
System.out.println("Circumference Circle 1" + c1.circumference());
System.out.println("Area Circle 1" + c1.area());
System.out.println("Circumference Circle 2" + c2.circumference());
System.out.println("Area Circle 2" + c2.area());
}
}
```

Whereas, for the second object c 2, we initially the same set of values with minus 4.0, minus 8.0, 10.0. Thus two objects are created and their data are also initialized.

Now, in the next 4 statements we see how we can access the different methods in these two objects namely circumference and its area. So, it is the same as earlier program accept that for two different objects we have used it. Now let us have the demo for this. So, in this case two objects are created and you will be able to get the values of the two objects here. So, same process we can compile it and then we can run it. So, now, we can see the two different objects and as the objects are the different their area and then circumference as printed area also different.

So, now we have understood that how for a class many objects can be created. In this case, we have considered to their maybe a large number of objects of the same class can be created. Now, in the our next demonstration is that whether one program can include more than one classes or not.

(Refer Slide Time: 08:37)

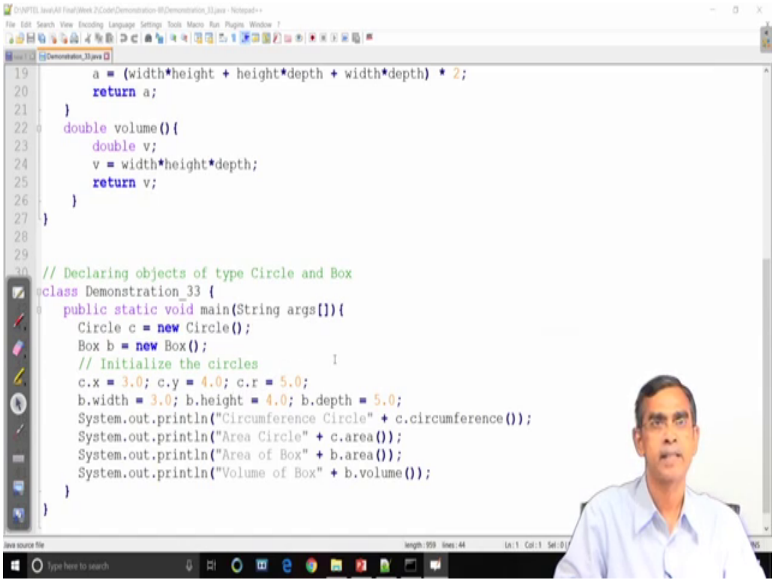


```
class Circle {
double x,y;
double r;
double circumference(){
return 2*3.14159*r;
}
double area(){
return (22/7)*r*r;
}

}

class Box{
   double width;
   double height;
   double depth;
   double area(){
      double a;
       a = (width*height + height*depth + width*depth) * 2;
       return a;
   }
   double volume(){
       double v;
       v = width*height*depth;
       return v;
    }
```

So, this demonstration includes the same. Now here we include one class namely, the Circle which has the same definition as earlier. In addition to this class, we declare one more class the name of this class which it has been given is called the Box. This class Box has 3 data field, width, height and depth. In addition to these 3 field, it has also 3 2 methods namely area and volume.

You note that the method area which is defined in class Box is totally different the method which is defined for the class circle. So, the two methods are defined in two classes, but they are different; they are not same which we have defined in a different way.

(Refer Slide Time: 09:29)



```
        a = (width*height + height*depth + width*depth) * 2;
        return a;
    }
    double volume(){
        double v;
        v = width*height*depth;
        return v;
     }
}


// Declaring objects of type Circle and Box
class Demonstration_33 {
    public static void main(String args[]){
      Circle c = new Circle();
      Box b = new Box();
      // Initialize the circles
      c.x = 3.0; c.y = 4.0; c.r = 5.0;
      b.width = 3.0; b.height = 4.0; b.depth = 5.0;
      System.out.println("Circumference Circle" + c.circumference());
      System.out.println("Area Circle" + c.area());
      System.out.println("Area of Box" + b.area());
      System.out.println("Volume of Box" + b.volume());
    }
}
```

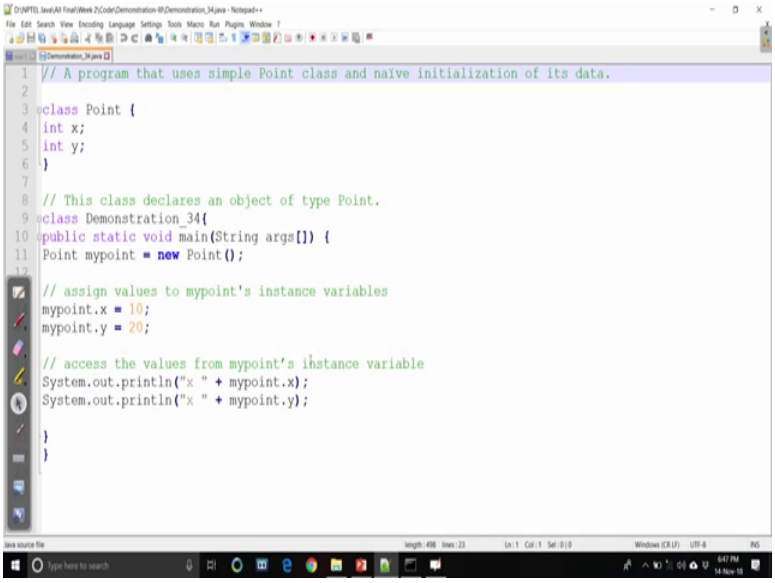
On the other hand circumference is a unique method in the class Circle where is the volume is also unique method in the class Box. So, this way we create two classes namely Circle and box, we can save all these two classes in same file as the main class. Now let us have the look of the main class, we give the name of the main class as Demonstration underscore 33. This is the main class which includes the main method and now you see that in this main method, we create two objects one object of the type class Circle another object of the type class box. So, the two object c small c and small b are created here.

Now, again as it is as it was in the case of circle, we initialize the value. Here also the objects that we have created to initialize their values. So, here in the first statement, we create a we initialize the value of the Circle objects with 3.0, 4.0 and 5.0 whereas, for the Box object we initialize the value of width height and depth as the values 3.0, 4.0, 5.0. So, this completes creation of two different type of objects of two different classes.

Now, in the next four statements, we can access the different methods. As we have seen here for the Circle c, we access the area and circumference and for the Box b we access the area and volume. Now let us have the execution of this program, we will be able to see that the two objects they are defined in terms of their classes and then they can access by the main method. So, now, we can see that for the first two printout is basically the display basically Circle circumference and area and next two output is for the Box area of the Box and volume of the Box

Now, we have understood about that how many classes can be created. It is not necessary that all the classes should be maintain in the same file in the main class. We will discussed about organization of the classes in your in our program that will be discussed in due time not now. So, now, let us see here we can see when we create an objects. So, it is our responsibility to initialize. The objects initialization of an object means initialization of all the member elements that belongs to the object. For example, for the Circle x y r for the Box width height and depth, we have to the initialization, but this initialization as we have done. In the last few examples little bit t d s because we have to do forcefully anyway, but the initialization can be done more sophisticated way in our next program. We will see how the initialization of the objects can be done in an easy way.

(Refer Slide Time: 12:49)

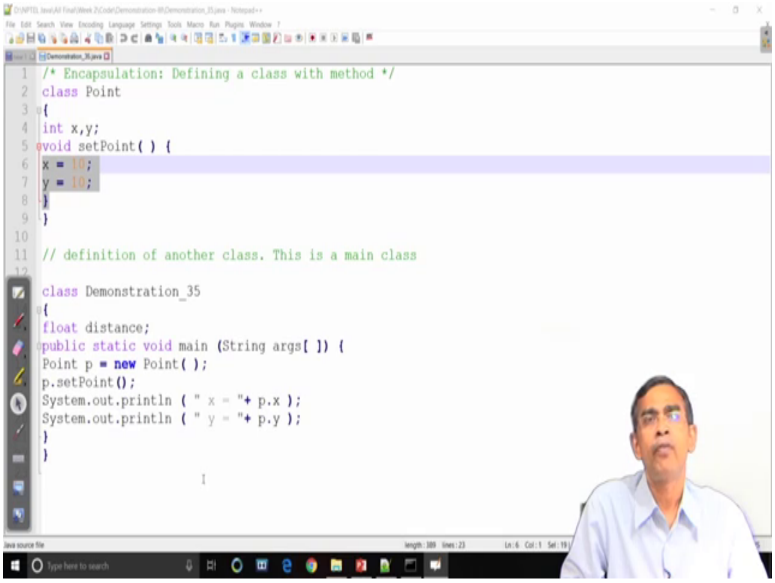


```
// A program that uses simple Point class and naïve initialization of its data.

class Point {
int x;
int y;
}

// This class declares an object of type Point.
class Demonstration_34{
public static void main(String args[]) {
Point mypoint = new Point();

// assign values to mypoint's instance variables
mypoint.x = 10;
mypoint.y = 20;

// access the values from mypoint's instance variable
System.out.println("x " + mypoint.x);
System.out.println("x " + mypoint.y);

}
}
```

So, here is a one example a 3.4 right let us see. So, now, we have declared here one, another class Point it has two member elements x and y declared as integer. It does not have any method in this case. Anyway we want to discuss about how this x y value can be initialized. Now here if you see, this is the usual procedure that we can create an object of type class and for this thing we can initialize the value from the main program. This is a usual procedure that you have learn earlier apart from this process.

Now, we will discussed about we can initialize by calling some methods. Now so, do to do this thing we have to declare a method in the class Point itself. Now let us see the next class 3.5 as a demo, we can see the class Point is little bit redefined with another method.

(Refer Slide Time: 13:41)



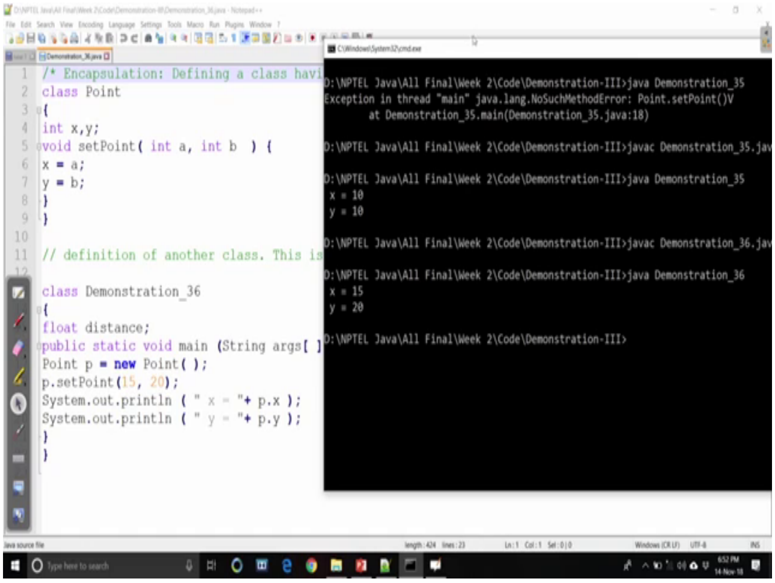
```
/* Encapsulation: Defining a class with method */
class Point
{
int x,y;
void setPoint( ) {
x = 10;
y = 10;
}
}

// definition of another class. This is a main class

class Demonstration_35
{
float distance;
public static void main (String args[ ]) {
Point p = new Point( );
p.setPoint();
System.out.println ( " x = "+ p.x );
System.out.println ( " y = "+ p.y );
}
}
```

The name of the method which we have include here set point. The setPoint has been planned to initialize its value. So, here example we can see the setPoint method has the 2 initialization statement x equals to 10 and y equals to 10.

Now, come to the you creation of an object. Now we create this object by main class here. The name of the main class is Demonstration underscore 35 and we have declared one variable distance ok. It will be it not useful here in this context anyway, but here see we create an object p of class Point is the next statement here and then we call the setPoint method is basically initialize the values for the object p and as you know by means of initialization one method setPoint, it will initialize as x equals 10 and y equals to 10.

Now, in the next two statement if we can print the value of the x and y of the object p, it will be printed like this. Now let us run this program.

(Refer Slide Time: 14:59)



```java
/* Encapsulation: Defining a class havi
class Point
{
int x,y;
void setPoint( int a, int b  ) {
x = a;
y = b;
}
}

// definition of another class. This is

class Demonstration_36
{
float distance;
public static void main (String args[ ]
Point p = new Point( );
p.setPoint(15, 20);
System.out.println ( " x = "+ p.x );
System.out.println ( " y = "+ p.y );
}
}
```

```
D:\NPTEL Java\All Final\Week 2\Code\Demonstration-III>java Demonstration_35
Exception in thread "main" java.lang.NoSuchMethodError: Point.setPoint()V
        at Demonstration_35.main(Demonstration_35.java:18)

D:\NPTEL Java\All Final\Week 2\Code\Demonstration-III>javac Demonstration_35.jav

D:\NPTEL Java\All Final\Week 2\Code\Demonstration-III>java Demonstration_35
 x = 10
 y = 10

D:\NPTEL Java\All Final\Week 2\Code\Demonstration-III>javac Demonstration_36.jav

D:\NPTEL Java\All Final\Week 2\Code\Demonstration-III>java Demonstration_36
 x = 15
 y = 20

D:\NPTEL Java\All Final\Week 2\Code\Demonstration-III>
```

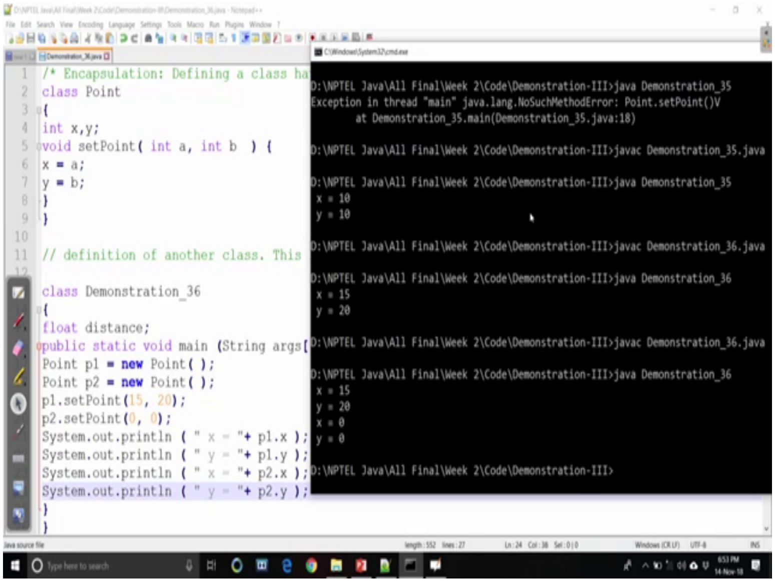
We are now executing this program. The program that Demonstration underscore 35 which basically includes the Point class declaration and then the setPoint the method their and then we will create an object of the type dot java. So, the object the program is now successfully compiled, we just going to run it 31.5 ok; fine.

So, let us see here. So, here we can see the object is created and we did not do any initialization in the main method. We did the initialization via the setPoint method and the initialization initiation here x equals to 10 and y equals to 10. Now here we can see that initialization this method, but how at this method always initialize only with certain initialization. Here for example, x equals to 10 y equals to 10. This means that if we create any other objects, it will and we call the setPoint method for initially, we always initialize with the same 10. So, this is not a good and not desirable.

Now we will have a good method that how the installation object can be done with the different data values. So, in that case we have to again rewrite this function this method using passing argument. Now here is an example, we just slightly change this class declaration and setPoint method also passing two values a and b here a and b are the two values to be passed to this method. So, that a can be used to initialize the value of x the and b can be used to initialize the value of y and the rest of the program is same as earlier one, but only the difference you can see when we call this setPoint we pass 1520; that means, in this case the objective p which we have created it can be initialized with its x value as 15 and y value as 20.

Now, let us run this program and then have a quick demo. So, 3.6, right. Now let us run this and we will see the object data member will be printed as 1520. Now let us come to the program, we can call this method or we can create another object say p 1 and p 2. We are creating two objects p 1 and p 2 here right; the p 1 and p 2 ok.

(Refer Slide Time: 17:47)



```java
/* Encapsulation: Defining a class ha
class Point
{
int x,y;
void setPoint( int a, int b  ) {
x = a;
y = b;
}
}

// definition of another class. This

class Demonstration_36
{
float distance;
public static void main (String args[
Point p1 = new Point( );
Point p2 = new Point( );
p1.setPoint(15, 20);
p2.setPoint(0, 0);
System.out.println ( " x = "+ p1.x );
System.out.println ( " y = "+ p1.y );
System.out.println ( " x = "+ p2.x );
System.out.println ( " y = "+ p2.y );
}
}
```

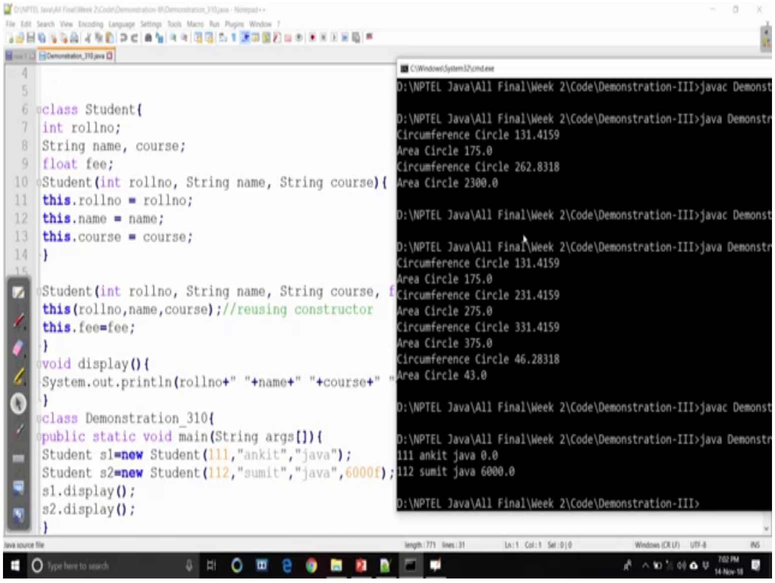
```
D:\NPTEL Java\All Final\Week 2\Code\Demonstration-III>java Demonstration_35
Exception in thread "main" java.lang.NoSuchMethodError: Point.setPoint()V
        at Demonstration_35.main(Demonstration_35.java:18)

D:\NPTEL Java\All Final\Week 2\Code\Demonstration-III>javac Demonstration_35.java

D:\NPTEL Java\All Final\Week 2\Code\Demonstration-III>java Demonstration_35
 x = 10
 y = 10

D:\NPTEL Java\All Final\Week 2\Code\Demonstration-III>javac Demonstration_36.java

D:\NPTEL Java\All Final\Week 2\Code\Demonstration-III>java Demonstration_36
 x = 15
 y = 20

D:\NPTEL Java\All Final\Week 2\Code\Demonstration-III>javac Demonstration_36.java

D:\NPTEL Java\All Final\Week 2\Code\Demonstration-III>java Demonstration_36
 x = 15
 y = 20
 x = 0
 y = 0

D:\NPTEL Java\All Final\Week 2\Code\Demonstration-III>
```

Again setPoint we just on keep it. One another p 2, another say 0 and 0 p 1 p 2 0 p 1 and p 2 and 0 0 ok.

So, we create two objects one is 15, 20 as the value of x and y ' another is 0,0. Now again we can print it just two statements. So, we can we can print the values of two objects here right for first one is p 1 dot x p 1 dot y p 2 dot x p 2 dot y right. Now let us save this program. Now here we have created two objects and the two objects are initialized with their own values and then we use this main method to print their values by which is at. Now we can see that the two objects are created and they have in been initialized by their own values by means of setPoint method

Now, here is the one idea about we have to explicitly initialize the object by means of declaring and defining the method.

(Refer Slide Time: 18:53)



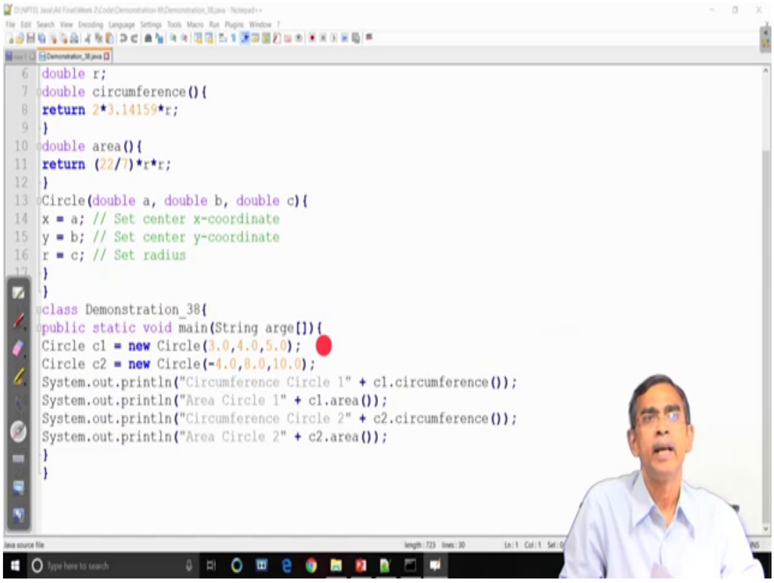
```
class Student{
int rollno;
String name, course;
float fee;
Student(int rollno, String name, String course){
this.rollno = rollno;
this.name = name;
this.course = course;
}

Student(int rollno, String name, String course, f
this(rollno,name,course);//reusing constructor
this.fee=fee;
}
void display(){
System.out.println(rollno+" "+name+" "+course+" "
}
class Demonstration_310{
public static void main(String args[]){
Student s1=new Student(111,"ankit","java");
Student s2=new Student(112,"sumit","java",6000f);
s1.display();
s2.display();
}
```

```
D:\NPTEL Java\All Final\Week 2\Code\Demonstration-III>javac Demonst

D:\NPTEL Java\All Final\Week 2\Code\Demonstration-III>java Demonstr
Circumference Circle 131.4159
Area Circle 175.0
Circumference Circle 262.8318
Area Circle 2300.0

D:\NPTEL Java\All Final\Week 2\Code\Demonstration-III>javac Demonst

D:\NPTEL Java\All Final\Week 2\Code\Demonstration-III>java Demonstr
Circumference Circle 131.4159
Area Circle 175.0
Circumference Circle 231.4159
Area Circle 275.0
Circumference Circle 331.4159
Area Circle 375.0
Circumference Circle 46.28318
Area Circle 43.0

D:\NPTEL Java\All Final\Week 2\Code\Demonstration-III>javac Demonst

D:\NPTEL Java\All Final\Week 2\Code\Demonstration-III>java Demonstr
111 ankit java 0.0
112 sumit java 6000.0

D:\NPTEL Java\All Final\Week 2\Code\Demonstration-III>
```

In java a unique features are available which basically help java programmer to automatically initialize the objects and that initialnce through at the time when the object is created itself. Now this concept the automatic initialization at the time of creating an object is called constructor. In our next example, we will see how the automatic object initialization using the concept of constructor is possible.

Now, let us have the program here not this program; go to the program number 3.8.

(Refer Slide Time: 19:39)



```
double r;
double circumference(){
return 2*3.14159*r;
}
double area(){
return (22/7)*r*r;
}
Circle(double a, double b, double c){
x = a; // Set center x-coordinate
y = b; // Set center y-coordinate
r = c; // Set radius
}
}
class Demonstration_38{
public static void main(String arge[]){
Circle c1 = new Circle(3.0,4.0,5.0);
Circle c2 = new Circle(-4.0,8.0,10.0);
System.out.println("Circumference Circle 1" + c1.circumference());
System.out.println("Area Circle 1" + c1.area());
System.out.println("Circumference Circle 2" + c2.circumference());
System.out.println("Area Circle 2" + c2.area());
}
}
```

So, let us have a simple program which basically helps to automatically create the objects and then initialize it. Now let us have the program here. Now the class again we can continue our this demonstration with the class Circle this class Circle has the 3 member elements x y r two methods circumference and area.

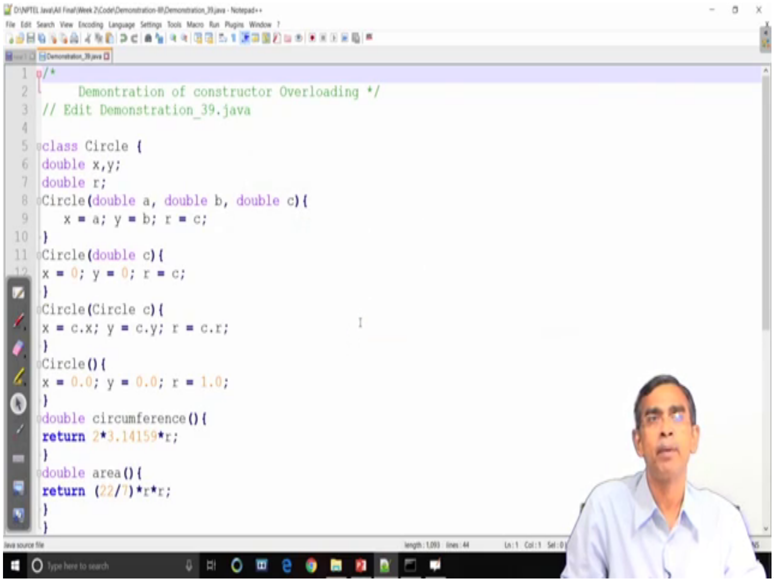
Now, we declare one method. This is a new method that we have added here in this class the name of this method. You see the is the same as the name of the class. This is a unique thing here the name of the method is same as the name of the class and also you can note in this case this method does not have any return type. So, this is called a constructor which is basically has the same name as the name of the class itself and it should not have any return type

Now, here the name of the constructor is there for Circle and this constructor is defined with 3 arguments. All these three arguments are of type double a b and c here. Basically a is the argument which will be passed to the value x, b is another argument whose value will be passed to y and c is the another argument whose value will be passed to r. This means that if we can create an object passing 3 values and they this 3 values will be used to initialize its member element. Now let us have the main class here after defining the Circle we can initialize the objects now here we see at the time of creating the object.

So, we are creating the object using the new operator here. Now in the first creation we create the object c 1 and we passed 3 values 3.0, 4.0, 5.0. This many means that constructor, it will call the method Circle which is the constructor method and pass this value to the Circle objects c 1. Similarly c 2 object will be initialize.After the initiation is over, the method will be accessed and then method will print the different values the according to the calculation or definition of the method.

Now, let us have the execution is the same as earlier. There is no different so far, the compilation execution is concerned; however, from the result point of view it is more sophisticated way of initialization. So, we have created the object at that time we have initialize the object at the time of creating the objects and then this is by means of constructor this concept is call constructor as I told you.

(Refer Slide Time: 22:39)


D:\NPTEL Java\All Final\Week 2\Code\Demonstration-III\Demonstration_39.java - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Demonstration_39.java
Java source file
length : 1,093 lines : 44
Type here to search

```
/*
      Demontration of constructor Overloading */
// Edit Demonstration_39.java

class Circle {
double x,y;
double r;
Circle(double a, double b, double c){
   x = a; y = b; r = c;
}
Circle(double c){
x = 0; y = 0; r = c;
}
Circle(Circle c){
x = c.x; y = c.y; r = c.r;
}
Circle(){
x = 0.0; y = 0.0; r = 1.0;
}
double circumference(){
return 2*3.14159*r;
}
double area(){
return (22/7)*r*r;
}
}
```

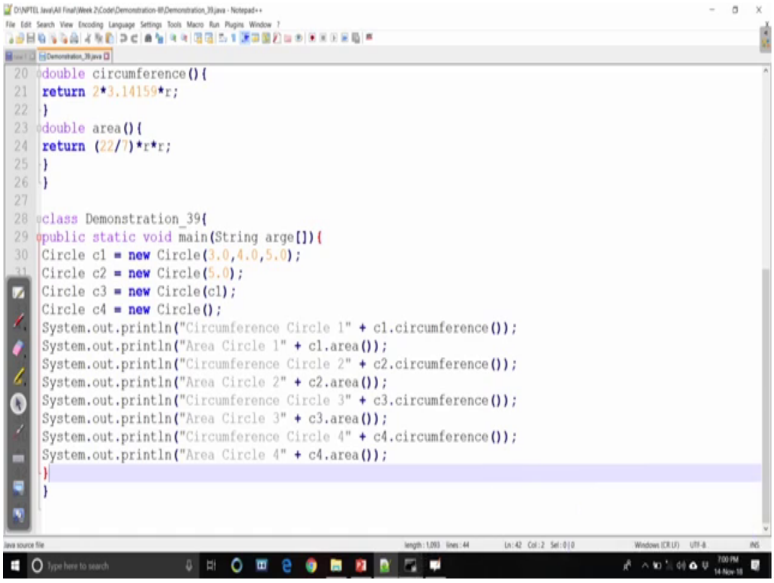
Now, constructor has its more dimension. In the sense that it also gives a lot of flexibility to a programmer to initialize an object in a different way. Now if we if you want to initialize the object in a different way this then you have to define. So, many constructor for your own requirement Now this concept is called the constructor overloading, now here again let us come to the discussion of the class Circle in the last example we have discussed about how the automatic initialization by means of a constructor with three arguments we have defined it.

Now, in addition to this, we want to define another way of initializing. Here in the next constructor that we have declared. Here if you see, this constructor has only one argument double c and the constructor is defined like this if we call a constructor with one argument, then it will initialize their member elements with x equals to 0 y equals to 0 and r equals to 0 as for this definition. That mean if we create an object passing only 1 argument, then the java runtime compiler will automatically understand that this constructor should be invoke.

If we create an object passing 3 arguments, then the first constructor will be called. Now here again you see the third constructor that we have defined here. It is a little bit different; here we can pass and argument which is the object itself that is quite possible. So, here the Circle c another object which has been passed. if we passed it this means that the new object that will be created having the same data member as the Circle c here. So, this is the one constructor; that means, a duplicate one object can be created duplicate object in the sense that they are different objects; however, same data values.

Another way is called the default constructor. If you define a constructor without passing and argument,t then it is called the default constructor.

(Refer Slide Time: 24:43)



```
double circumference(){
return 2*3.14159*r;
}
double area(){
return (22/7)*r*r;
}
}

class Demonstration_39{
public static void main(String arge[]){
Circle c1 = new Circle(3.0,4.0,5.0);
Circle c2 = new Circle(5.0);
Circle c3 = new Circle(c1);
Circle c4 = new Circle();
System.out.println("Circumference Circle 1" + c1.circumference());
System.out.println("Area Circle 1" + c1.area());
System.out.println("Circumference Circle 2" + c2.circumference());
System.out.println("Area Circle 2" + c2.area());
System.out.println("Circumference Circle 3" + c3.circumference());
System.out.println("Area Circle 3" + c3.area());
System.out.println("Circumference Circle 4" + c4.circumference());
System.out.println("Area Circle 4" + c4.area());
}
}
```

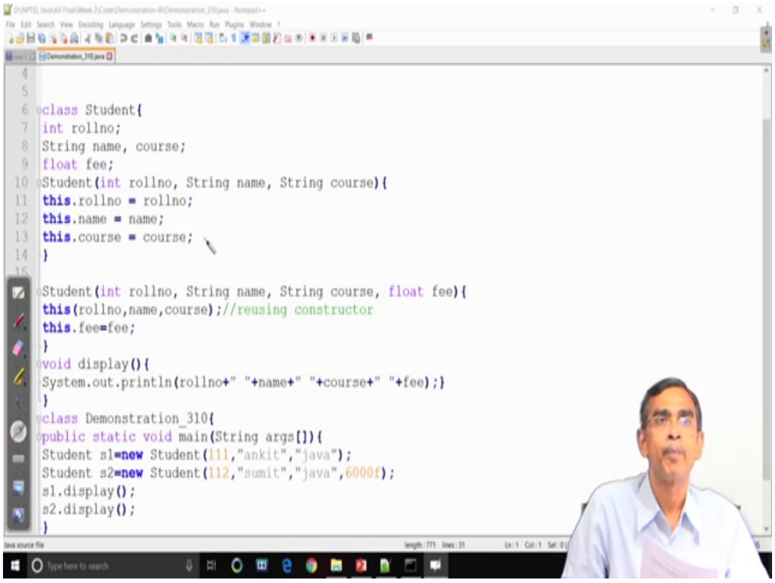
.

So, in this case the last constructor the fourth constructor in the series is called the default constructor. Here the default constructor means that x equals to 0.0, y equals to 0.0, r equals to 0.0. Now let us have the main program. So, main program is defined here by means of main class demonstration underscore 39. Now here we see we have created 4 objects and each objects are created to with their own constructor.

So, the first objects c one created with the first constructor second object with the second constructor and third object with the third constructor and finally, this one here ok. So, so, here if you see the c3 and c1 when we create the object c3 we pass the c1 c1 which is a which is an object of the class Circle as for the third constructor declaration. And the rest of the things are basically printing the different parameters area and circumference for the four Circle that we have created and definitely c1 and c1 c3 should give the same result. Now let us have the Demonstration. So, let us have the quick demo, here we are overloading constructor now. So, all 4 objects are created automatically and the objects are created at the time of creation of at the time of creating object; I mean declaration and by passing the values

So, this is about the overloading constructors. Now there are few more important and interesting facts about this class and then object constructor.

(Refer Slide Time: 26:31)



```
class Student{
int rollno;
String name, course;
float fee;
Student(int rollno, String name, String course){
this.rollno = rollno;
this.name = name;
this.course = course;
}

Student(int rollno, String name, String course, float fee){
this(rollno,name,course);//reusing constructor
this.fee=fee;
}
void display(){
System.out.println(rollno+" "+name+" "+course+" "+fee);}
}
class Demonstration_310{
public static void main(String args[]){
Student s1=new Student(111,"ankit","java");
Student s2=new Student(112,"sumit","java",6000f);
s1.display();
s2.display();
}
```

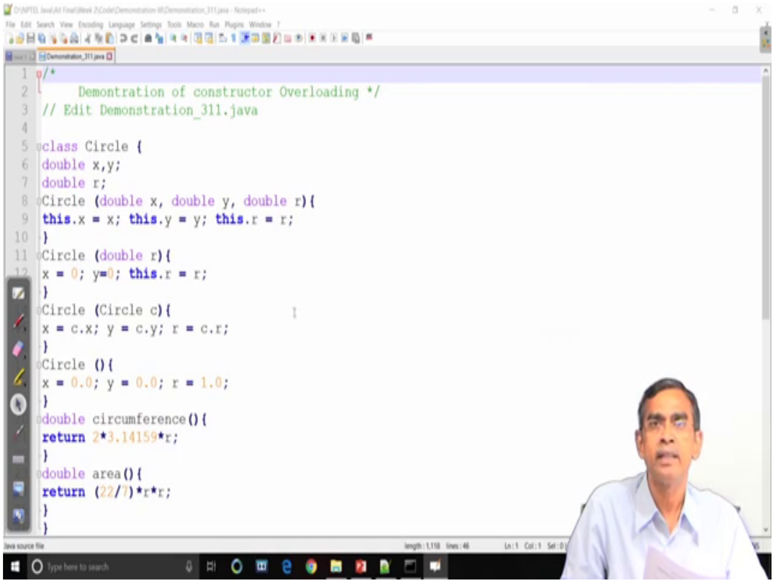
We will just want to discuss quickly one concept it is called the this keyword now this keyword concept can be understood very easily if we follow this example. Now let us consider the declaration of one class call the student. Here it this class has 4member elements. Here integer roll number name and course as the String type and then key as the float.

Now, here again we see the this is the constructor declaration, roll number, name, String course and it. Now here you can note the arguments that we have passed they have their name and in this case the name of the data members are also same. Now if we try to initialize then definitely. So, for roll number equals to roll number, then this is not the correct similarly name equals to name it is also little bit ambiguous. So, what you can do is that we can we can define it by means of this one. So, this dot roll number indicates that this roll number belongs to this class member whereas, only roll number is the argument that has been passed similarly this dot name specify this the name members of the class Student and this dot course the name member of the class course.

Now so now, if we run this one we will see exactly how we the next part will discuss latter on. Let us see how we can use it. So, this is basically this keyword which is basically helps to resolve the namespace collision. So, in this case roll number name and course is been collided with the data member it is there. So, in order to resolve it we can use the this keyword. So, here we have used it and then we passed it.

Now, you can see the first time when the object is created first object on keep java 0.0 and now let us come to another way. Now this point we will discuss ok; let us come to the special use of this keyword in another 3.11 let us run the 3.11 demo.

(Refer Slide Time: 28:39)

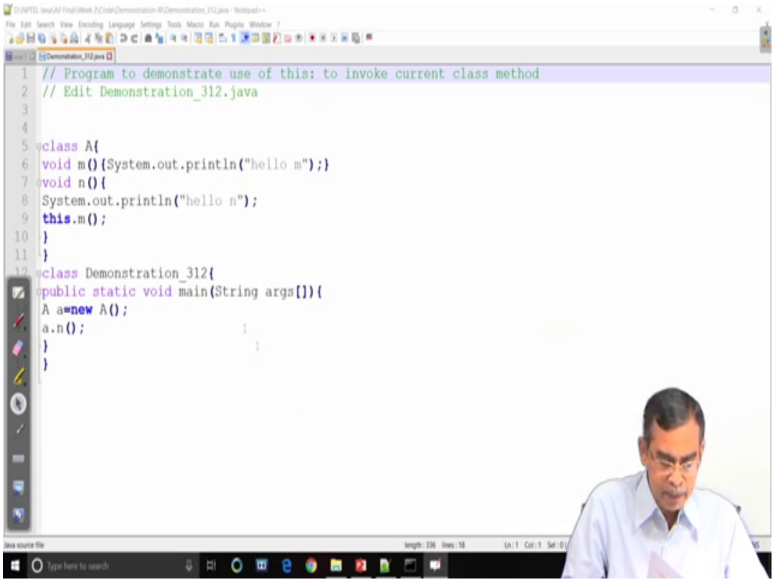


```
/*
      Demontration of constructor Overloading */
// Edit Demonstration_311.java

class Circle {
double x,y;
double r;
Circle (double x, double y, double r){
this.x = x; this.y = y; this.r = r;
}
Circle (double r){
x = 0; y=0; this.r = r;
}
Circle (Circle c){
x = c.x; y = c.y; r = c.r;
}
Circle (){
x = 0.0; y = 0.0; r = 1.0;
}
double circumference(){
return 2*3.14159*r;
}
double area(){
return (22/7)*r*r;
}
}
```

Now we will just discuss on more important thing here. Now in the context of Circle again the this keyword can be understood very precisely. So, here again constructor, we can see the constructor has been defined with x y r which have the same name as the member element and we resolve it by using this it is at the same thing applicable to the other also

So, if we want to resolve some name which belongs to the same class with other variable. So, we should use this. So, this is the one use of this keyword in java program. now there is another use of this program in the java keyword to understand this things let us have a simple program; let us have the program 3.12.

(Refer Slide Time: 29:23)



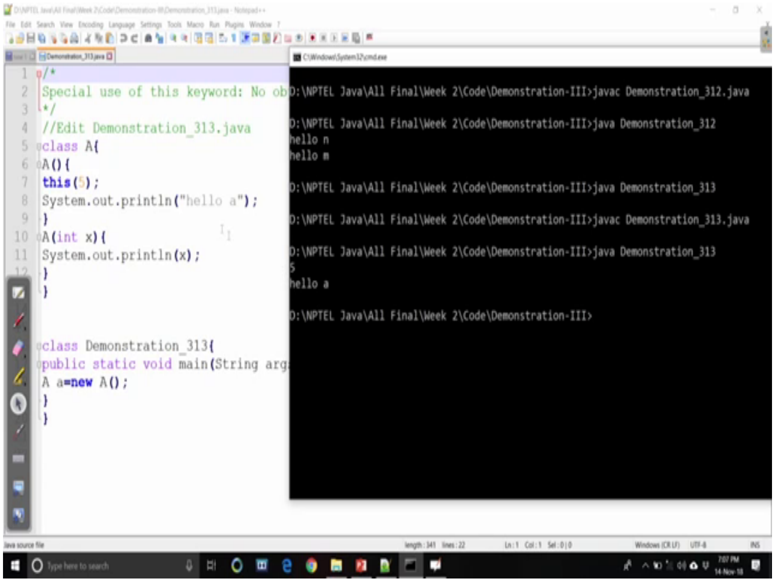
```
// Program to demonstrate use of this: to invoke current class method
// Edit Demonstration_312.java


class A{
void m(){System.out.println("hello m");}
void n(){
System.out.println("hello n");
this.m();
}
}
class Demonstration_312{
public static void main(String args[]){
A a=new A();
a.n();
}
}
```

Now, here we can see that this can be used to have some other advantage see. Here let us have program very simple one, it is very easy to understand. A is the class which declared here and it has one method m another method n the method m will simple print hello m, n also will print simple one String hello n.

Now, we if we want to access the same method in some other method, then definitely we have only learned that in order to access a method we should create an object. But whenever we are declaring a method or defining a method there is no question of creating any objects. Now if the objects belongs to the same; that means, if you want to access the method which is in the same class, then we can resolve it by specifying this dot m. For example, in the method void n we call the method m which is defined in this class itself.

So, in order to resolve it we used the this dot m this means that void n whenever we call for a new object created here for example, A then this will call the method m as well as method n. So, this so, so, when a dot n; that means, we call the method n; it will print hello n and then it will print that hello m. Now let us have the quick execution of this program so, that we can understand about this one.

(Refer Slide Time: 30:53)



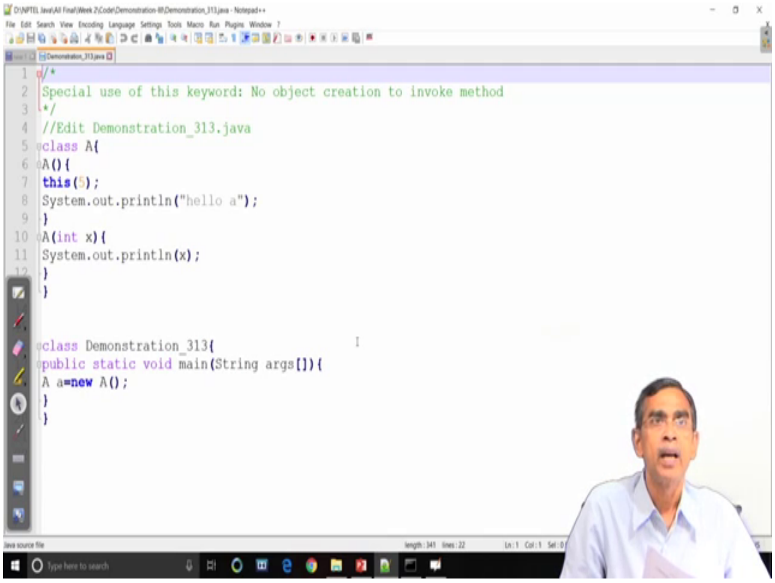
```
/*
Special use of this keyword: No ob
*/
//Edit Demonstration_313.java
class A{
A(){
this(5);
System.out.println("hello a");
}
A(int x){
System.out.println(x);
}
}


class Demonstration_313{
public static void main(String arg
A a=new A();
}
}
```

```
D:\NPTEL Java\All Final\Week 2\Code\Demonstration-III>javac Demonstration_312.java

D:\NPTEL Java\All Final\Week 2\Code\Demonstration-III>java Demonstration_312
hello n
hello m

D:\NPTEL Java\All Final\Week 2\Code\Demonstration-III>java Demonstration_313

D:\NPTEL Java\All Final\Week 2\Code\Demonstration-III>javac Demonstration_313.java

D:\NPTEL Java\All Final\Week 2\Code\Demonstration-III>java Demonstration_313
5
hello a

D:\NPTEL Java\All Final\Week 2\Code\Demonstration-III>
```

So, this method is used to refer one method which belongs to the same class without creating an object. Now this is the demonstration execution run of the program that we have discussed now ok.

We can understand. So, that the method m has been called from via n method which is defined now let us have the another quick example of this method it has many purpose that can be used who you want to give another.

(Refer Slide Time: 31:29)



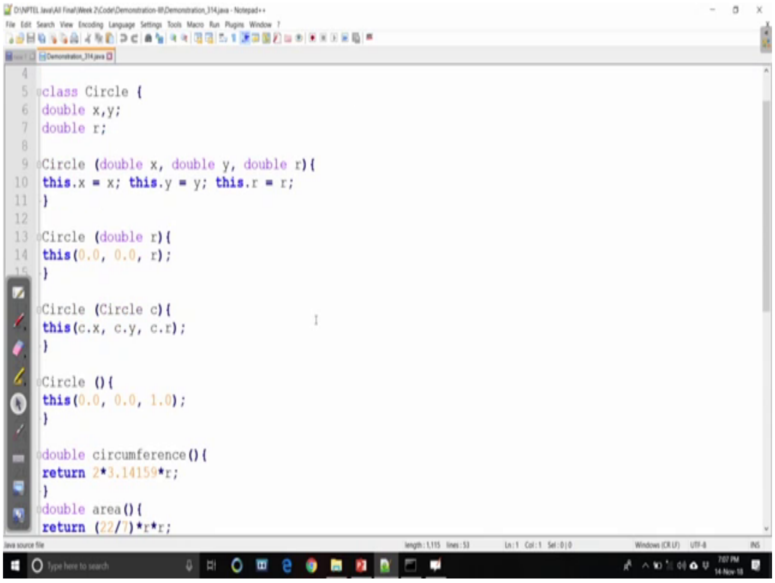
```
/*
Special use of this keyword: No object creation to invoke method
*/
//Edit Demonstration_313.java
class A{
A(){
this(5);
System.out.println("hello a");
}
A(int x){
System.out.println(x);
}
}


class Demonstration_313{
public static void main(String args[]){
A a=new A();
}
}
```

So, this is the one method you see. So, class A is declared and A is the constructor go to the constructor definition there a constructor is there fine. We are clearing the A constructor; now here the constructor there are two constructors right the first constructor ok. Let us come to the second constructor what is the second constructor we define the second constructor by means of passing an argument then type integer. So, this means it will like this one right. Now here one default constructor which define who is basically initialized by a default value say 5.

But that installation calling by the one already defined constructor namely A intakes. So, if you want to call any constructor within another constructor, then you can use this and then specifying the parameter values that you want to have now let us have the execution of this program. So, this basically call a constructor in a constructor and this constructor is basically a nesting type of call it is it is also part of the overloading constructor. So, execution is there. So, you can see how the whenever we create an object here in this statement a equals to new A. We call the default constructor when first constructor whenever the first constructor is called it basically initialize the member elements with value 5. We have no member actually, the value is 5 is passed to the class object and then it print hello a using this value x.

So, basically it basically print as 5. So, this is the way. Now let us have a last day demo it is more interesting to understand.

(Refer Slide Time: 33:21)



D:\NPTEL Java\All Final\Week 2\Code\Demonstration-III\Demonstration_314.java - Notepad++

```
class Circle {
double x,y;
double r;

Circle (double x, double y, double r){
this.x = x; this.y = y; this.r = r;
}

Circle (double r){
this(0.0, 0.0, r);
}

Circle (Circle c){
this(c.x, c.y, c.r);
}

Circle (){
this(0.0, 0.0, 1.0);
}

double circumference(){
return 2*3.14159*r;
}
double area(){
return (22/7)*r*r;
```

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Demonstration_314.java

So, the last demo, it will be easier to understand. Now again Circle class. We create the one constructor is basically using 3 parameters. Now this constructor, then can be used to define other constructor by means of this the methods. So, circuit for example, second one this we call the same constructor here by using the this and passing 0.0, 0.0 r similar is the second constructor we use this c dot x c dot y c dot r. Here this and all these this basically indicates the one privilege constructor which is which has the three parameters and automatically it called them.

So, this way it basically has a little bit change in the similar program that we have discussed earlier that the these are use of these operator. So, we have discussed about constructor overloading and I just last things I just want to mention about. So, what the constructor overloading is concerned to constructors can be treated as a different; if they are different in terms of the argument number of arguments and the type of arguments. So, that is only things are there that you can already realize about it and if you can go through the all examples, then you will be able to understand that all three constructors or whatever the constructors they are different because of the different argument that we have passed ok. So, this concludes our demonstration on in encapsulation.

Thank you very much.