

# FUNDAMENTALS OF OBJECT ORIENTED PROGRAMMING

## Lecture07

### Lecture 7: Classes and Objects in Java and Solved problems

#### Static variable

```
1  #include <stdio.h>
2  int main()
3  {
4      printf("%d",func());
5      printf("\n%d",func());
6      return 0;
7  }
8
9  int func()
10 {
11     int count=0; // variable initialization
12     count++; // incrementing counter variable
13     return count;
14 }
15 |
```



So, welcome to Lecture 7: Classes and Objects. So, in the last class, we had seen a program, right? Multiplication of two complex numbers, right, using classes and objects. So, there we had a keyword called static, right. So, what is the difference between a static variable and an ordinary variable, right?

So, when I am talking about an ordinary variable, it is limited to a certain scope, right. So, which is well-defined, whereas when I am talking about the scope of a static variable All right. So, that extends throughout the program. We will see with an example.

All right. So that you can understand in a better way. So, here you go. All right. So, for example, here I have, let us say, a function.

All right. So, it is a simple program. All right. So, here you have a C program. All right.

So, int main, I take it as C. You can change it to C++ also. Not a problem. So, int main, right? So here you have a function, right? So, int function, right?

Whose return type is integer. So, int count is equal to 0, which is not very difficult. Count equal to 0. And then, line number 12, I am using count++, all right? And then I have return count.

I take an example of a procedural-oriented language, C. So, you can just put cout when you put cout, that is all that is called C++. Now, I am calling the function, right? When I am calling the function, you can tell me what will happen, alright. So, function when I call, so count is 0, count++, so I can expect the output, what? 1. Good. So, you will get the output 1, alright? And next, again, I am calling, alright? The new line, again, I am calling the function.

So, what will happen? It goes count 0, count++, count is 0, count++, return count. Alright.  $0 + 1$  is 1. So, I have to get two 1s as output.

Alright. So, I have to get two 1s as output. So, you can see that. Alright. So, I will get two 1s as output.

Okay. So, what will I do? Shall I change this variable, right, with static. We will see what will happen, right.


So, when I change this to static, let us see what is happening, yes. Same program, I have put static so that you can understand, right, those who are learning about static variables for the first time, this program will give you an idea. So now, as usual, I am calling the function. So, the first one, count is 0. Count++.

## Static variable

```
1  #include <stdio.h>
2  int main()
3  {
4      printf("%d",func());
5      printf("\n%d",func());
6      return 0;
7  }
8
9  int func()
10 {
11     static int count=0; // static
12     count++; // incrementing counter variable
13     return count;
14 }
15
```

Handwritten annotations:

- A red circle around the word `static` in line 11.
- A red arrow pointing from the `count` variable in line 11 to the `count` variable in line 5.
- A red circle around the text `count → 1` with a red checkmark.
- A red box containing the numbers 1 and 2, representing the output of the two function calls.
- A red checkmark next to the `stdout` label.



So now, count is 1. Alright. So here, I am writing count is 1. Alright. Count++.

Count is. So return count. So it will return 1. Alright. So now when I use this variable.

Right. Static variable. The keyword static. Now my count is 1. That is why I separately wrote.

Last time I did not write anything. Alright. So now my count is 1. So again when I am calling the function in line number 5. In line number 5, when I am again calling this function, static int right? Your count is 1, count++, so now you will get the output 2, right.

So this is the output I can expect, return count. So this is the output I can expect, right. So you can see that when I run this code, I can get the output like this. So this is the difference between Simple variable and static variable, so this program. These two programs will be useful to understand what is the meaning of the keyword static. If you look at the last class, we have used the keyword called static. Now you understand, based on these two programs you can understand. So without a static variable, right? Line number 11, if I don't use static, I am getting output as 1, 1. If I use static, I

So, the count will be considered as 1 before going to line number 5, and then when it is calling the function again, the count++ 1 + 1 becomes 2, and here we are getting the output 1 and 2. So, this is how the static variable is being defined.

So, now the same complex number problem, right? The multiplication of two complex number problem. So, let us consider using, we will write a program in Java, all right?

```
1  /* package whatever; // don't place package name! */
2
3  import java.util.*;
4  import java.lang.*;
5  import java.io.*;
6
7  //import java.util.Scanner;
8
9  // Define a class to represent a complex number
10 class Complex {
11     double real;
12     double imag;
13
14     // Method to set values for real and imaginary parts
15     void setValues(double r, double i) {
16         real = r;
17         imag = i;
18     }
```



So, here you go. So, these are all the utilities I am importing in Java. Now, I create almost the same. The syntax is almost the same. Few things here and there.

There will be a change. So, class Complex, you have double real and double imaginary, right? You have two member data or data members, and then set values, the same like what we had seen in C++, set values, and here the multiplication, right? So, here you cannot have the reference operator. In Java, you cannot have the reference operator and the star operator, right?

When I define complex C1, So, C1 is an object. So, I can print the identity hash map. So, suppose I print, let us say, System.out.print and C1, I will get the identity hash map. Whereas in the case of C++, you have to slightly use the address operator.

```

19
20 // Method to multiply two complex numbers
21 static Complex multiply(Complex c1, Complex c2) {
22     Complex result = new Complex();
23     result.real = c1.real * c2.real - c1.imag * c2.imag;
24     result.imag = c1.real * c2.imag + c1.imag * c2.real;
25     return result;
26 }
27
28 // Method to display the complex number
29 void display() {
30     if (imag >= 0)
31         System.out.println(real + " + " + imag + "i");
32     else
33         System.out.println(real + " - " + (-imag) + "i");
34 }
35 }

```



Slightly change the syntax, I mean to say. So you will have the address operator. So here, complex C1, and here you have complex C2. And you are having the result. Which is an object.

Whose class is complex. Right. So dynamically allocating memory using the new operator. And this we are going to study. Right.

This syntax, please note. This is nothing but the constructor. Right. So line number 22. You are dynamically allocating memory for result.

Dynamic memory allocation. Complex result. You put the new operator. So dynamically allocate the memory. And you have complex followed by the function notation; you call it as the constructor, right.

So now, as usual, the multiplication of two complex numbers, what we had seen in C++, the same syntax you are writing, and return the result, right. So a few things are changing. Here we are not using the address operator on line number 21, whereas we have used it if you recall. We used the address operator, right, and complex result. So here you go. This is, right, the new syntax in Java, right. So a few things here and there, there will be a change. And then display, right, the void display, the same member function.


Here we call it a method, right. The same member function that we have used in C++. So here it will be the method, and the method will give you the real + i into imaginary, right. And then, what is the main program? Alright.

So, the main program. So it is a driver class. The driver class is main because you have the main, almost like in C++. Right. Int main.

So here the syntax is static. You know the meaning of static. Void main. So this we will see slightly later. Right.

```
36
37 public class Main {
38     public static void main(String[] args) {
39         Scanner sc = new Scanner(System.in);
40
41         // Create two objects for complex numbers
42         Complex c1 = new Complex();
43         Complex c2 = new Complex();
44
45         // Read real and imaginary parts of the first complex number
46         System.out.print("Enter real and imaginary part of first complex number: ");
47         double real1 = sc.nextDouble();
48         double imag1 = sc.nextDouble();
49         c1.setValues(real1, imag1);
50
```

2.0 + 3.0i



String arguments. String of arguments. Alright. So now this function, this built-in one is for taking the input from the user. Alright.

So this is the Syntax that you are writing, which means you are going to take some input from the user. In fact, in line numbers 47 and 48, we are taking the input from the user for this. So this built-in statement is required in line number 39. Let us create two objects, right?

Let us instantiate two objects, c1 and c2. c1, whose class is complex, and then here we use a new operator, right? Right. And then you use a constructor. And then another object.

So dynamically, you are allocating memory for c1 and c2. That is the meaning. Okay. So now enter your real and imaginary. Right.

Enter your real and imaginary parts. Or the first complex number. So that you are taking the input from the user. This is what you use. The scanner.

Right. So it is like a scanner for c1. `sc.nextDouble()`, `nextInt`, that means the integer data type, `nextDouble`. So this function, the inbuilt function, using your object `sc` under the class `Scanner`, right? So here you go, you are inbuilt, right?

```
51 // Read real and imaginary parts of the second complex number
52 System.out.print("Enter real and imaginary part of second complex number: ");
53 double real2 = sc.nextDouble();
54 double imag2 = sc.nextDouble();
55 c2.setValues(real2, imag2);
56
57 // Multiply the two complex numbers
58 Complex result = Complex.multiply(c1, c2);
59
60 // Display the result
61 System.out.print("Result of multiplication: ");
62 result.display();
63
64 sc.close();
65 }
```

Handwritten notes:  $4.0 + 5.0i$  (in red) with arrows pointing to the `real2` and `imag2` variables. A red circle is drawn around the word `result` in line 58, and a red arrow points from the `result` variable to the `display()` method in line 62. A red checkmark is next to line 58.



This `sc` and `Scanner`, they are all the inbuilt class, right? So which is useful? This `sc` object is invoking the `nextDouble()` function. `nextDouble()` method, inbuilt method. So that means whatever the user is entering will be stored in `real` one.

Similarly, next line, line number 48. Right? This `sc`, the object, is invoking the inbuilt method `nextDouble()`. It will be assigned to `imaginary` one. Right?

So, you are taking the input from the user. And then here, `C1` is invoking `setValues`. Exactly like what we studied in C++. Okay. So, `real 1` you are passing `imaginary 1`.



 input

2 3

4 5

Result of multiplication: -7.0 + 22.0i



So, assume that it is  $2.0 + 3.0i$ . Something like that. So, real 1 is 2.0 and imaginary 1 is 3.0. Similarly, the second. Alright.

Let us say  $4 + 5i$ .  $4.0 + 5.0i$ . So, real 2. 4.0 and imaginary 2. Right.

5.0. So, take the input from the user and set values. So, then the result method. Right. The result object, whose class is Complex.

Right. So here you have. So here, the Complex is invoking. Right. The class Complex.

So which is invoking the multiply method. All right. So the multiply method is being invoked here. You are passing c1 and c2 as the arguments. Exactly like what we have studied in C++.

Almost similar. All right. So the Complex. So it is more or less like as if you have created some objects. So that object is invoking multiply.

So this syntax is also valid. Or you create some Complex. Let us say c3. All right. So c3 will invoke the multiply.

That is also possible. Or, otherwise, simply put the class name of the class dot multiply c1 comma c2. So then the resultant will be stored in the result. That is the meaning. Then, the result of the multiplication.

So, result dot display will show result dot real and result dot imaginary. And then, sc dot close will close the file because you are taking the input using sc, and then



you do `sc dot close`. Right. This object is invoking the `close`. It is a kind of file closing, right.

So, now when I run the code, I assume that even if you put 2 3, it is assumed that you put 2.0 and 3.0, 4.0 and 5.0. The same input we will give, and you can see the result of multiplication. You will get this is the output, ok. So, how to create a class? Right, and how can you define a class and how to create objects? Right, all these we have studied with the help of C++ as well as a Java program also. Now you are familiar with the static keyword, right.

```
1  #include <iostream>
2  using namespace std;
3
4  // Define a class to represent a distance
5  class Distance {
6      int feet;
7      int inches;
8
9  public:
10     // Method to set the values of feet and inches
11     void setValues(int f, int i) {
12         feet = f;
13         inches = i;
14     }
15
```



So, we will see one more case study, right, one more problem. So, write a C++ program

To add two distances where feet and inches are given. So, you are given feet. Let us say 9 and 7.

Another one is 8 and 6. So this is feet. And these are all inches. So if you are given this feet and inches.

I want the output of the addition. So 12 inches is 1 foot. So 7 + 6 is 13. That means 1 foot will be added.

```

32 ▾ int main() {
33     // Create two Distance objects and one for the result
34     Distance d1, d2, result;
35
36     int feet1, inches1, feet2, inches2;
37
38     // Input for the first distance
39     cout << "Enter feet and inches for the first distance: ";
40     cin >> feet1 >> inches1;
41     d1.setValues(feet1, inches1);
42
43     // Input for the second distance
44     cout << "Enter feet and inches for the second distance: ";
45     cin >> feet2 >> inches2;
46     d2.setValues(feet2, inches2);
47

```

$result = d1 + d2;$

9 8  
8 6



```


48     // Add the two distances
49     result = d1.add(d2);
50
51     // Display the result
52     cout << "The sum of the two distances is: ";
53     result.display();
54
55     return 0;
56 }
57

```

18 2



1 + 8 is 9. 9 + 9 is 18. So this I have to get the output and 1 inch. When I am adding these two, I have to get the output. 18 feet and 1 inch.

 stdin

15 9

The sum of the two distances is: 34 feet 4 inches

18 7

33 16  
34 4  
7 1



```
17
18 // Method to add two Distance objects
19 Distance add(Distance d) {
20     Distance result = new Distance();
21     result.inches = inches + d.inches;
22     // Convert inches to feet if > 12
23     result.feet = feet + d.feet + (result.inches / 12);
24     result.inches %= 12; // Keep the remainder in inches
25     return result;
26 }
27
28 // Method to display the distance
29 void display() {
30     System.out.println(feet + " feet " + inches + " inches");
31 }
32 }
33
```



So this is how you do it with the help of object-oriented programming? You can think of it. So how do you think? You can create a class, Distance. And then you have to include feet and inches as the object.

```

34 public class Main {
35     public static void main(String[] args) {
36         Scanner sc = new Scanner(System.in);
37
38         // Create two Distance objects and one for the result
39         Distance d1 = new Distance();
40         Distance d2 = new Distance();
41         Distance result;
42
43         // Input for the first distance
44         System.out.print("Enter feet and inches for the first distance: ");
45         int feet1 = sc.nextInt();
46         int inches1 = sc.nextInt();
47         d1.setValues(feet1, inches1);
48

```



```

1  /* package whatever; // don't place package name! */
2
3  import java.util.*;
4  import java.lang.*;
5  import java.io.*;
6
7  // Define a class to represent a distance
8  class Distance {
9      int feet;
10     int inches;
11
12     // Method to set the values of feet and inches
13     void setValues(int f, int i) {
14         feet = f;
15         inches = i;
16     }
17

```



And then you have to think of getter and setter values. Display values right, and then in the main program, you have to write the addition of two distances, alright. So, this we will see. So, let us consider I have a class Distance, alright. So, you include iostream because I am going to use cin and cout.

So, therefore, using namespace std. So, class Distance, the name of the class is Distance. I am using feet and inches. Alright. I am using feet and inches.

So here I have setValues. setValues int f and int i. Int f and int i. So feet is equal to f. So whatever input you are taking, that is being passed. It will be stored as feet and inches under that particular object. So, feet is equal to f and inches is equal to i, right? So now here you can see I have the add function, right?


Name of the member function, and I have an object d whose class is Distance, and the return type is also Distance, and here I have Distance result. So that means I will have two, right? I need two Distances. So, for example, d1 + d2, which I will store in d3. Result understood.

So, now I can use here, right, without two arguments. In the last program, in the multiplication of two complex numbers, we used two arguments. So, can you write it with one argument? Yes, you can do that. What will it do?

One object will invoke this add function. You will have something like this: d1 dot add, and then you are passing d2. You will call it similar to this. So that means here you have inches, simply inches. You are not having d1 dot inches.

```
49      // Input for the second distance
50      System.out.print("Enter feet and inches for the second distance: ");
51      int feet2 = sc.nextInt();
52      int inches2 = sc.nextInt();
53      d2.setValues(feet2, inches2);
54
55      // Add the two distances
56      result = d1.add(d2);
57
58      // Display the result
59      System.out.print("The sum of the two distances is: ");
60      result.display();
61
62      sc.close();
63  }
64 }
```

*Handwritten notes:*  
d1 → 9 8  
d2 → 8 6



Simply inches, and similarly, simply feet. So this will be taken care of. That is nothing but d1 dot inches, d1 dot feet. Alright, so one object's inches will be added with another object's inches. Alright, so for example, I have 9, 7, 8, 6.

Right, so first, what I will do is I will add the inches. So here I will add 7 + 6, what will happen? 13. Alright, so here I have to use some logic. When it is going beyond 12, alright, I have to increment my feet.

Right? So, we will do that. So, inches you are adding first. So, we are getting result dot inches as 13 right now. Assume that.

Right? Then what will I do? I will add the feet.  $9 + 8$ . Feet + d dot feet.

The object that you are going to invoke. So, that is going to be your 9, and the rest is your passing 8. Right? So,  $9 + 8$ . Correct?

So, you will have how much? 17. And then, so here you have result dot inches integer division 12. So, 13 by 12 integer division. What is it? 1.

Alright. So 1 will be added. Feet + d1 dot feet and result dot inches slash 12 divided by 12. 13 by 12 integer division 1. So  $17 + 1$ .

So right now it will be 18. And then what you have to do? You have to do mod. Right? So  $13 \bmod 12$ .

$13 \bmod 12$  is what? 1. Your result dot inches will be 1. Right? Your result dot inches will be 1.

$13 \bmod 12$ . So  $17 + 1$ , 18.  $13 \bmod 12$  is 1. So this will be my resultant for this test case. This test case.

So, this is what you are returning as a result, this is what you are returning as a result. So, in the wide display, you put feet and inches; in the wide display, you put feet and inches. So, let us see because it is slightly different code from what we have studied in the case of addition or multiplication of two complex numbers, all right. So, here we will see how we are going to use this syntax in the main program.

So, you create three objects: one is two distances,  $d1 + d2$ , and put it in the result, right. So, basically, I want it like this: result equals  $d1 + d2$ , all are objects. So, d1 dot inches, d1 dot feet, d1 dot feet, d1 dot inches, d2 dot feet, and d2 dot inches. So, the result that you will get is result dot feet and result dot inches.

Okay. And then you have a set of variables. All right. Feet, inches, feet1, inches1, feet2, and inches2. Okay.

So now you are entering feet1 and inches1. So, assume that I entered 9 and 8. All right. I entered 9 and 8. So, you are setting the values.

Feet1, inches1. That means d1 is invoking setValues. So, the setValues member function here you go feet and inches will be set. Okay, feet is equal to f and inches is equal to i. So, 9 and 8, whatever, right. So, 9 and 8 and the second one I am giving 8 and 6, right. So, you are entering your feet2 is 8 and inches2 is 6.

Feet2 and inches2. So, you are setting the values. So, this will be set. So, this is for d1 and this is for d2 because d2 is invoking here, alright. SetValues will be called.

So, d2 dot feet is 8 and d2 dot inches is 6. So, now This is what the syntax I wrote if you recall, right. So, d1 dot add d2 exactly I wrote this, right. So, d1 dot add d2 you are putting whatever be the result you are putting as the result that is the object, right.

So, the result object is under class Distance, okay. So, now what is happening d1 is invoking The member function add by passing d2. Right? So, that means you are pausing what?

8 and 6. You are passing 8 and 6. So when I am passing 8 and 6. Right? So here your d dot inches is what?

8. Right? Inches. 6. Right?

D dot inches is 6. 8 and 6. So here I change 9 8 and 8 6. What about my simple inches? I do not have any object, dot inches.

So which is nothing but this inches? 8. So that means here what is happening?  $8 + 6$  will be added. So here it is nothing but  $8 + 6$  will be added.

So when 8 and 6 are being added, it is 14. Right? This is exactly what is happening. 8 and 6 are nothing but 14. Right? d1 is invoking. You may ask the question how this d1 dot inches is going. Right?

So d1 is invoking. Since d1 is invoking, so here the simple inches + d dot inches. d dot inches we know. It is 8. Inches is 6.

So  $8 + 6$  are being added, and it is nothing but 14. Now you add feet. So, feet is nothing but 9 and 8. So, 9 and 8 will be added along with result dot inches slash 12. So, here I have 14.



14 slash 12 is 1. Right. So,  $9 + 8$  is 17.  $17 + 1$  is 18. 18 feet I got. Still, inches I have not got. Right. So, inches is nothing but  $14 \bmod 12$ .  $14 \bmod 12$  is what? 2. Right. So, that means 18 feet. 18 feet 2 inches I will get as the output, right? When I am adding this 18 and 2,  $14 \bmod 12$  is right. This is what we are telling:  $14 \bmod 12$  is line number 22. You look at line number 22,  $14 \bmod 12$  is 2. So we will get 2 as the output, so totally you will have 18 feet and 2 inches as the output. Alright, so this we are displaying. Result dot display we are doing. Alright, so let us see what are the test cases.

Test cases, assume that, yeah, I will give you another test case. 15, 9. 15 feet, 9 inches. 18 feet and 7 inches. When you add this, alright, so  $9 + 7$ , I will get 16.  $15 + 18$ , I will get 33. So,  $33 + 1$  will be added.  $16 \bmod 12$ , which will be 4.  $16 \bmod 12$  is 4. So  $33 + 1$ , I will get 34.  $16 \bmod 12$ , I will get 4. So 34 feet and 4 inches I have to get as an output.

So if you look at, alright, so I am getting exactly the expected output. We are getting this as an output: 34 feet and 4 inches. I hope it is clear. This problem is clear. Can we write the same program in Java? Right?

So, almost the same. So let us try. So earlier, what you can do is try to write it in both C++ and Java, and then later at one point in time, we will stick only to C++. Okay? So here you have a class distance, feet and inches, and here you have a member function, the same member function what we have studied in

C++, what we have seen in C++, set values, right? So, similar, right? Whatever we have done, the same thing you are doing. Distance is a return type, alright. So, the return type is an object, distance add distance d, right? So, d is an object, you have a distance. So, returning an object is possible in these two programs, whatever program that we have seen in C++ as well as in Java.


The return? Can you return the object? Right, so these two programs are telling you yes, you can return the objects, and here you go. You have a result object whose class is distance. Dynamically, you are creating a memory space with the new operator, and distance is nothing but the constructor. So, the rest is similar. You are adding inches + d inches, d dot inches, whatever we have studied in C++. Similarly, feet + d dot feet. And then, result dot inches slash 12, integer division. So, whatever the result is, then you are putting it over here: result dot feet. Similarly, result dot inches is equal to result dot inches mod 12, the

remainder. Return a result, right? So, almost the similar syntax, so only here, line number 20, there is a change, right?

So, dynamically, you are creating an object. Then, void display. So, display, you are displaying feet and inches. So, this is also similar. And then, if you look at the class, you are not putting any semicolon.

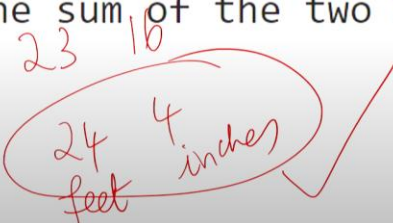
The class ends in line number 32. And here you go. I may use certain input. Therefore, I have an object sc under the inbuilt Scanner, inbuilt class Scanner. All right.


Your program starts here. The class main you call it as the driver class. All right. You have a main program that starts from here. So, you are creating 3 objects, right? So, d1 and d2, right? Dynamically creating an object, and the third one is for the result, right? So, d1, d2, new operator constructor, d2, new operator constructor, and then you have result, alright.

 input

15 9  
8 7

The sum of the two distances is: 24 feet 4 inches





So feet 1, right? So feet 1 is taking the input, whatever input you are giving or input that you are getting from the user, that will be stored in feet 1, line number 45. And similarly, whatever the input you are getting from the user, you are storing in inches 1, right? So feet 1 and inches 1, that is your first object, right? First objects: feet 1 and inches, or feet and inches, simply.

So feet 1 and inches will be set. Alright. So the object d1 is invoking the setValues member function, alright. So feet 1 and inches 1, whatever you are passing, alright. So that will be set for d1 dot feet and d1 dot inches, right.

So this is what exactly will happen. If you look at line number 14 and 15, d1 dot feet and d1 dot inches will be set. Similarly, for the object d2, right. So you are creating an object d2. It will be set.

Whatever feet 2 you are taking as an input from the user. Whatever inches 2 you are taking as an input from the user. So d2 dot setValues feet 2 comma inches 2. Alright. So now d1 is invoking add d2.

So, same. If I do 9, 8 for d1 and d2, if I have 8, 6. So, d1, the object d1, which is invoking d2. So, that means I am passing 8 and 6. 8 feet and 6 inches are being passed.

This is what is exactly happening over here. Right? So, d dot inches. What is d dot inches? 6. Right? So, d dot inches will be added with inches.

What is that inches? So, that will be passed when the object d1 is invoking at. So, that means the inches is nothing but 8. Right?

In the function member function, or you call it as a method. So, in the method you call line number 21, you have inches. So, that 8. Right? So,  $8 + 6$ , which is nothing but 14.

Right? So here you go, you have 14 and feet, you are doing feet + d dot feet. So your feet is nothing but 9 and d2 dot feet is nothing but 8, there it is d dot feet. So  $9 + 8$  will be added, which is 17, plus you have some more statement result dot inches slash 12.

So  $14 \text{ slash } 12$  is 1, alright. So 1 will be added. So here you have 18 and then  $14 \text{ mod } 12$ ,  $14 \text{ mod } 12$  is 2. So, that means 2, 18 inches, 18 feet, 2 inches, right. So, this you are displaying.

So, the sum of the distance result is invoking display, and then you have close, right. In sc is invoking close. So, when I give this as an input, right, 15.0, 9.0, 8.0, and 7.0. So, when I add  $9 + 7$  is 16, right.  $15 + 8$  is 23. So,  $23 + 1$ , 24, this will be 4,  $16 \text{ mod } 12$  is 4.

So, 24 feet 4 inches, I have to get this as the output, alright. So, let us see what you are going to get as output. So, you can see 24 feet and 4 inches, right. So, this we got as an output. So the last two classes, what we have done, how to define a class and how to create an object.

And we have seen two case studies, all right, in C++ as well as Java, all right. So now you are able to write a program. So we have not seen the access modifiers like public or private. Soon we are going to see them. And you have also used the static keyword in the last class.

So, now you are able to write certain basic programs. So, take these two as an example and then try to work them out using many problems, right, using the concepts you have studied so far, right. So, with this, I am concluding today's class. Thank you so much.