

# FUNDAMENTALS OF OBJECT ORIENTED PROGRAMMING

## Lecture19

### Lecture 19: Method Overloading

Thank you. So welcome to lecture 19 polymorphism. So here we are going to see method overloading in Java. So we had seen function overloading in C++ right. So it is exactly same as the function overloading that we had seen in C++.

So here we are going to consider in Java and in Java this is called method overloading. Suppose assume that a class is having more than one method right. So they can have right same name but they should have the different parameter list all right so either in number or type or both all right for example we had seen I want to find maximum of two numbers right let us say max v1 version 1 all right so int a comma int b I am writing and the return type is integer another one I can have a same integer type return type same name max v1 but here assume that I have three different parameters list right.

So in this case we know that one it is a maximum of two numbers right. So you have two parameters and here you have maximum of three numbers three parameters. So now this will be considered during compile time right which one will be chosen. Right. So that will be desired during compile time.

For example, so you have max V1. Correct. So max V1 and then assume that you are passing 2, 3. Right. You are passing the value 2, 3.

Correct. So in this case, assume that you are having some, let us say L. Another one is M. Max V1, 10, 20, 30. Right. So in fact, we had seen C++ we had seen. Which one will be chosen?

## Method Overloading in JAVA

In Java, method overloading occurs when a class has more than one method with the same name but different parameter lists (either in number, type, or both).

int maxV1(int a, int b) (2)  
int maxV1(int a, int y, int z) (3)  
maxV1(2,3)



So, this will be decided during compile time, right? So, in fact, this is the compile time polymorphism. So, in this case, in the case of Java, we call this as method overloading because max v1 and max v1, those functions, right? These two functions. So, they are called methods in Java.

So, which one will be called, right? So, this will be decided during compilation time, right? So, during compilation time, this will be decided and when it is returning, right, L and M, correct? So, which one will be chosen?

So, when you are calling this function max V1, 2, 3, this one will be chosen, right? And max V1, 10, 20, 30, when you are passing, this one will be chosen. So, how this will be decided? So, this will be decided during compilation time. So, that is why it is called compile time polymorphism or static binding.

And in case of Java, this is called method overloading. So, we will see one example. So, let us consider this example. We have class calculator. So we have class calculator and here you have one method.

```
7  /* package whatever; // don't place package name! */  
8  
9  import java.util.*;  
10 import java.lang.*;  
11 import java.io.*;  
12  
13 class Calculator {  
14     // Method to add two integers  
15     public int add(int a, int b) {  
16         return a + b;  
17     }  
18 }
```

### Example 1



So that method is adding two numbers, right? You are passing two parameters A and B. In fact, you are passing two values. You have two parameters. So it is a simple addition it is adding two integers return a plus b and here you have three parameters int a b and c right int a b and c. So here you are adding and then if you look we are using the same method name. So which is called the overloading right you call it as a overloaded method and the third one you have two double right.

So here in the first case we have two parameters both are integers right so now if you look at what we have defined method overloading so same name but different parameter list right so one case we have two parameters both are integers and here you have the parameters list is different three parameters in the third case we have two parameters but both are double and return type is also double right so here also it is adding you are using the add function in fact that is the add method in Java and this is called the overloading right this is called in fact method overloading. So add function we are using in one case we have two parameters in another case you have three parameters all are integers return type is also integer and the last case it is completely different Even though you are having two parameters, both are double and the return type is also double, all right.

In the first case, the return type is integer. So, here you are return A plus B, you are adding two numbers. So, let us see what is happening, all right. So, here you have created an object calc whose class is calculator and here you have the default constructor, all right. So, we do not have any role on constructor.

Let me check, quickly check. Yes, we do not have. It is a default constructor, fine. So now you are calling calc. the addition of 10, 20, right?

Add of 10, 20. So that means you are passing two integers, 10 and 20, right? The calc is an object which is invoking the method call add and you are passing 10 and 20. So when you are passing 10 and 20, so let us see what is happening, right? So here,

You are passing 10 and 20. Both are integers, right? That means this particular method will be called. So when it is calling, you are passing A and B. What you have passed? 10 and 20, I believe.

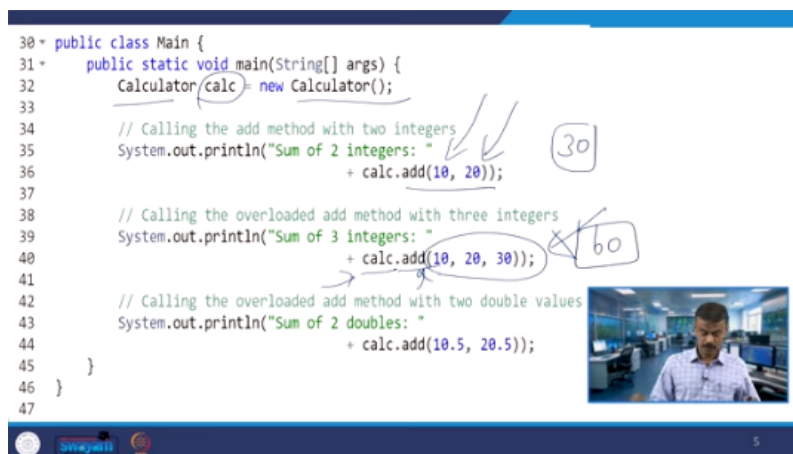
Yes, 10 and 20 when you pass. So A plus B, it will be 10 plus 20. So it will return 30, right? So it is returning 30. So here you are displaying sum of two integers, right?

When you are adding 10 and 20. So you will get 30. So this is the object oriented program is very elegant. You are simply calling this particular method. And even though you are using the same method.

So now you understand which one will be called. So this is called the concept of method overloading. So now the next one again calc. The object calc is calling again the add function. But which add function?

You are passing 2 values. You are passing 3 values. 10, 20 and 30. right so here we are passing three values so when I am passing three values you can see over here right so here you go you are passing three values what are all the values 10 20 and 30 so that means it will return 10 plus 20 plus 30 so 10 plus 20 plus 30 which is nothing but 60 so you will get 60 as the output right and finally You have sum of 2 doubles.

```
30 * public class Main {
31 *     public static void main(String[] args) {
32 *         Calculator calc = new Calculator();
33 *
34 *         // Calling the add method with two integers
35 *         System.out.println("Sum of 2 integers: "
36 *             + calc.add(10, 20));
37 *
38 *         // Calling the overloaded add method with three integers
39 *         System.out.println("Sum of 3 integers: "
40 *             + calc.add(10, 20, 30));
41 *
42 *         // Calling the overloaded add method with two double values
43 *         System.out.println("Sum of 2 doubles: "
44 *             + calc.add(10.5, 20.5));
45 *     }
46 * }
47 *
```



Here it is sum of 3 integers. And here you have sum of 2 doubles. So now this calcifunction. Here you have. This calcifunction which is invoking add.

You are passing 2 doubles. Right. 2 double values. 10.5 and 20.5. Right.


So 10 plus 20 is 30. 31. So we have to get 31.0 as output. Right. So in order to do this.

So this will invoke add. You are passing double values. So this particular function will be called line number 25 to 27. You are passing 10.5 and 20.5. So 10.5 plus 20.5 is 31.0.

So 31.0 will be the output. So what are all the output we have to get? 30, 60 and 31.0. So when I run the code, right.

```
1  /* package whatever; // don't place package name! */
2
3  import java.util.*;
4  import java.lang.*;
5  import java.io.*;
6
7  class MaxCalculator {
8
9      // Method to find the maximum of two integers
10     public int max(int a, int b) {
11         if (a > b) {
12             return a;
13         } else {
14             return b;
15         }
16     }
17 }
```

**Example 2**



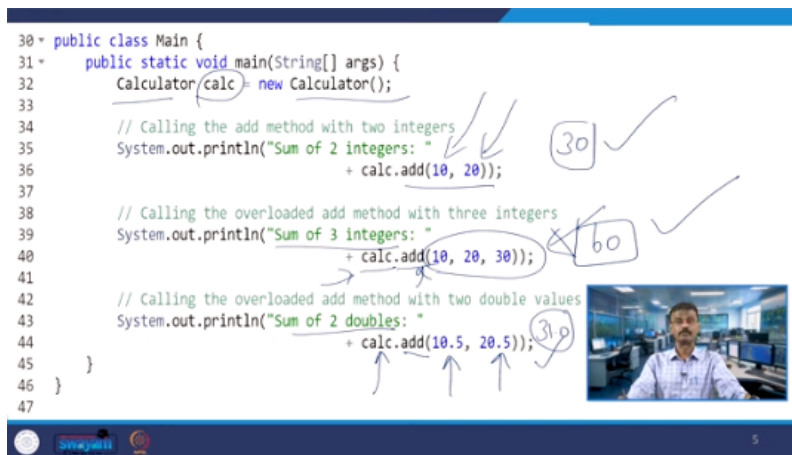
So, I will get the output 30, 60 and 31.0 right. So, this is the example of the method overloading right. So, I have addition of two integers and the second case it is addition of three integers and the third case addition of two doubles. So, when you run this code you will get the output of this. So, you can see sum of two integers

So, this particular add method is being called. So, 10 plus 20 you are getting 30 and when you are running this particular right when calc is invoking this particular add method. So, 10 plus 20 plus 30 you are getting 60. And finally, when calc is invoking this particular add method by passing two doubles. So you are getting 10.5 plus 20.5, you are getting 31 as the output.

So this is one example for method overloading. So that means during compilation time, it will be decided. So line number 35 it is calling, which one will be called? Line number 39 it is calling, which one will be called? And similarly line number 43, it has been called and which one will be called?

right. So, this will be decided during compilation time. So, therefore, this is called the compile time polymorphism or static binding or this is called method overloading in Java. So, maybe we will see one more example, right. So, here you have let us say as I already said the maximum of

```
30 public class Main {
31     public static void main(String[] args) {
32         Calculator calc = new Calculator();
33
34         // Calling the add method with two integers
35         System.out.println("Sum of 2 integers: "
36                             + calc.add(10, 20));
37
38         // Calling the overloaded add method with three integers
39         System.out.println("Sum of 3 integers: "
40                             + calc.add(10, 20, 30));
41
42         // Calling the overloaded add method with two double values
43         System.out.println("Sum of 2 doubles: "
44                             + calc.add(10.5, 20.5));
45     }
46 }
47
```



two numbers or three numbers correct so here you have the class max calculator right so here you have the class max calculator and here you go you have the maximum of two numbers right the maximum of two numbers so that means you are passing two parameters right you are passing two parameters if a greater than b that means you are passing two integers if a greater than b return a else return b If A greater than B return A else return B. So this is what happening and in this case when I am passing two parameters when I am passing two values so that means you have two parameters so that means this will calculate the maximum of two numbers. So this member function will call the maximum of two numbers. This method will find out The maximum of two numbers.

So now we have one more. Right. So you are passing three values. So that means it has three parameters. So here you have the logic to find out the maximum of three numbers.

Right. If A greater than B and A greater than C. That means A is greater than both B and C means return A. That means A is the maximum. Else if B greater than C. So you are comparing between B and C. Right. So return B. Otherwise obviously you have to return C. Right.

So first initially A is compared with B and C. That is what line number 20 you are doing. Right. So that means if it is satisfying both the cases. So then it is obvious that A is maximum. So return A. Else if B greater than C. Right.

```

18 // Overloaded method to find the maximum of three integers
19 public int max(int a, int b, int c) {
20     if (a > b && a > c) {
21         return a; ✓
22     } else if (b > c) {
23         return b; ✓
24     } else {
25         return c; ✓
26     }
27 }
28 }
29

```



So now my comparison will be if any one fails. Right. It will be false. Right. So that is going to line number 22 and B and C will be compared.

So when you are comparing B and C, what is happening? Return B, right? That means if B greater than C, return B. That means B is maximum. Otherwise, C is maximum. So now if you look at line number 10,

you have maximum of 2 numbers and line number 19, you have maximum of 3 numbers. Correct? So, now come to the main program. Right? So, in the main program, you have class called calculator whose object is max calculator.

Right? So, here in fact, we have the class called max calculator. Right, so your object is calculator and the class is max calculator and with the help of the memory allocation using new, so you have the constructor called max calculator. Right, now here you are printing, right, this is a print statement, maximum of 10 and 20, right. So calculator is the object.

So this object is invoking max, right. So this object is invoking max and here you have, right. 2max method. Alright. Let us see.

You are passing 10 and 20. That means I am passing 2 values. So when I am passing 2 values we know that this particular function will be called. 2 values. I am passing 2 values.

So that means here I want the maximum of 10 and 20. Alright. So maximum of 10 and 20 A will get the value 10 and B will get the value 20. So now I am comparing. Correct.

So now I am comparing A and B. If A greater than B, 10 greater than 20 false. So it is going else. So return B. So 20 obviously between 10 and 20, 20 is maximum. So return B. It is returning B. So therefore here in this case 20 will be printed.

Correct. And in the second case calculator object is invoking again max. But now you are pausing. three values 10, 20 and 30 correct. So you are passing three values 10, 20 and 30.

So in that case this particular function will be invoked, this particular method will be invoked right. So 10, 20, 30 you are passing. So you are comparing whether 10 greater than 20 and 10 greater than 30. What do you say?

Both are false. False and false is false. All right. So it is coming over here. What is your B?


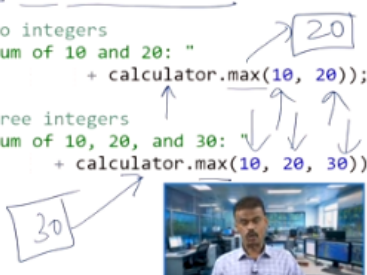
B is 20 and C is 30. So whether B greater than C, B greater than C, whether 20 greater than 30, false, alright. So it is going to this else part, return C, C is 30, so 30 will be returned, alright. So when you are returning 30, alright, so here maximum of 10, 20, 30, so 30 will be printed. So here in this case maximum of 10 and 20, 20 will be printed.

So in line number 35 and 39, which one will be decided, right? When you are calling this in the main, in fact, this is the main program. So which function will be called? This is decided during compile time. So this is another example of compile time polymorphism or you call it as method overloading.

So max is the function which is being overloaded. Max is the method which is being overloaded, right? So that is another example of method overloading. overloading right so in fact the maximum of 10 and 20 what we are getting max is 20 and maximum of 10 20 and 30 the maximum is 30 right i hope it is clear for everyone in fact we have taken two different examples correct and then the concept of method overloading or you can say compile time polymorphism or static binding being explained so we have taken The first case add, right?



```
30 * public class Main {
31 *     public static void main(String[] args) {
32 *         MaxCalculator calculator = new MaxCalculator();
33 *
34 *         // Find the maximum of two integers
35 *         System.out.println("Maximum of 10 and 20: "
36 *             + calculator.max(10, 20));
37 *
38 *         // Find the maximum of three integers
39 *         System.out.println("Maximum of 10, 20, and 30: "
40 *             + calculator.max(10, 20, 30));
41 *     }
42 * }
43
```



Three times we are calling. So which one will be decided during compile time? And similarly, in this case, we have taken two max, right? So in fact, line number 35 and 39, which one will be called, right? So this will be decided during compile time, right?

And in fact, this is your output. So output you are seeing maximum of 10 and 20 is 20 and maximum of 10, 20 and 30 is 30. So the next example we'll see method overloading in Java, right? So we can see one case study. So in the case study, we consider student management system using method overloading.

Student management system using method overloading. So the problem statement goes like this. A university needs a system where a student's academic performance can be evaluated in different formats. What are all the formats? Percentage based grading.

So this calculates the grade based on the percentage. And the second one is letter grade. So that letter grade which will calculate the grade based on the letter. So the letter A, B, C, etc. In some of the universities you have A+, A, B+, B, etc.

These are all the grades. And the last one is GPA. right. So the GPA we calculate GPA is nothing but grade point average. So this we can calculate right.

So that means GPA is nothing but that is calculating the grade based on the GPA right. So we will see percentage based letter grade and GPA. So we will use the concept of the overloading so here you have class student right class student here you have name of the string right which is a name whose data type is string which is a member data right and here you have the constructor parameterized

constructor so you are passing name so the name is assigned to this dot name right. So we had seen this arrow operator and here in this case this dot name.

In fact we have already seen this dot name and here name right which is assigned to this dot name. So now I have calculate grade. I have a function, I have a method called calculateGrade. Here I am passing the percentage. Whatever percentage of marks got by the students I am passing and that is a double data type and calculateGrade whose return type is y. If percentage is greater than or equal to 90.

If percentage is greater than or equal to 90. So it is going to print the name of the person because the name you will have over here. right in the constructor so that will be printed whatever be the name that you are going to pass so this particular name will be printed over here right if that particular student has got more than or equal to 90 so the student has secured grade A and then you have else else if percentage is greater than or equal to 80 right else if the percentage is greater than or equal to 80 right So you have system dot out dot println name of the student has secured grade B. Has secured grade B. Else if the percentage is greater than or equal to 70. Alright.

So system dot out dot println the name of the student has secured grade C. Else if right percentage greater than or equal to 60 student has secured grade D. Alright. So rest of the case the student who is getting less than 60 is considered as failed. So, when you are giving the marks or percentage based on this percentage, if I want to calculate these four grades A, B, C and D. So, based on the marks, I can print the grade of the student. Greater than or equal to 90 is A, greater than or equal to 80 is B, greater than or equal to 70 is C and greater than or equal to 60 is D. Otherwise, so here you can see one more. Right.

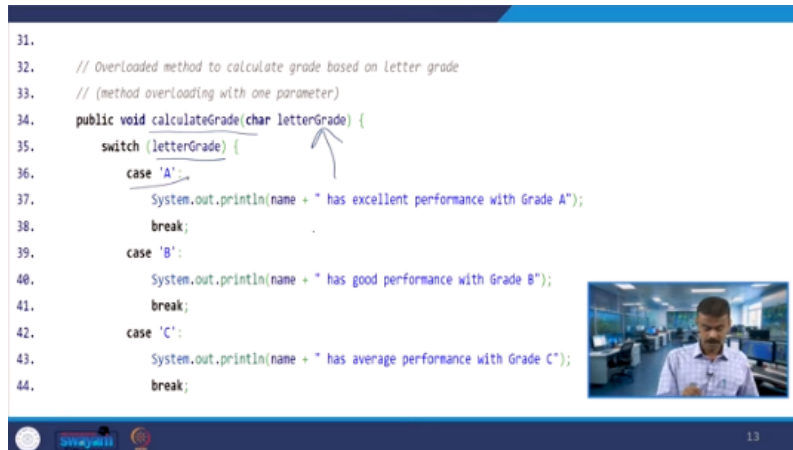
If a student is getting less than 60. Right. So that will be considered as fail. So now again I have overloaded method calculate grade. Right.

So here you have calculate grade. I have a same calculate grade. So here you are passing letter grade as a parameter. Right. So here letter grade is an argument.

So we are passing. If anything you are passing the grade. Right. The character. Right.

So which is the character data type that you are passing. right? So, switch statement, I am using switch statement with letter grade. So, if I get case A, right? So, the particular person's name, student's name has excellent performance with grade A, has excellent performance with grade A, right?

```
31.
32. // Overloaded method to calculate grade based on letter grade
33. // (method overloading with one parameter)
34. public void calculateGrade(char letterGrade) {
35.     switch (letterGrade) {
36.         case 'A':
37.             System.out.println(name + " has excellent performance with Grade A");
38.             break;
39.         case 'B':
40.             System.out.println(name + " has good performance with Grade B");
41.             break;
42.         case 'C':
43.             System.out.println(name + " has average performance with Grade C");
44.             break;
```



Then you have break statement. And case B, right? The particular student has good performance with grade B, grade B, break statement, right? And case C, the student name has average performance with grade C, so your break statement, right, with the letter grade A, B, C, D, what is the performance, so you are printing, and D, the student has poor performance with grade D, break, so the default, the student has failed, right, so this is the second case, right, second overloaded method, so this is the first method, calculate grade, second method, calculate grade, which is overloaded method, and the third method name, that is also

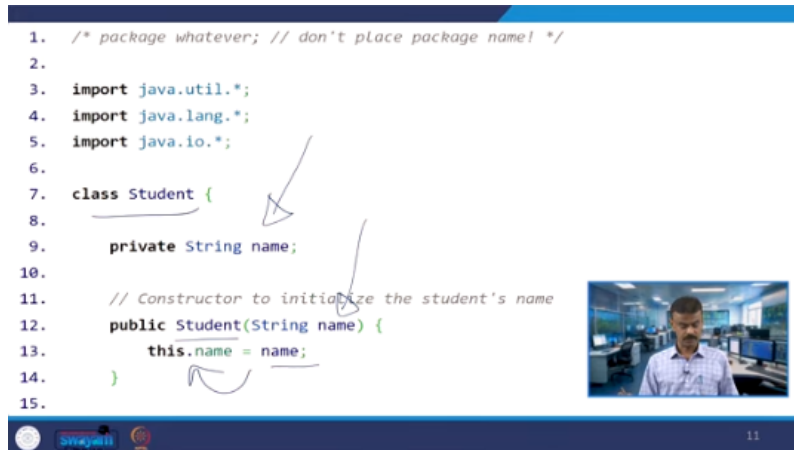
Calculate grade, right. So here you have if GPA is greater or equal to 4.0, alright. You are printing the student has secured grade A, right. It is GPA based. That is what we said, right.

If GPA is greater or equal to 4.0, alright. So there are universities 4 out of how much you are getting, alright. So GPA greater than or equal to 4.0, the student has secured grade A. If GPA is greater than or equal to 3.0, right, so the student has secured grade B, right, greater than or equal to 3.0, the student has secured grade B. If GPA is greater than or equal to 2.0, the student has secured grade C. Else if GPA is greater than or equal to 1.0, the student has secured grade D. Otherwise, failed. So now here you go. So you have an object student.

This is a driver class. You have an object student whose class is student and with the help of a new operator you have a constructor. You are passing the string Ravi Ashwin. So here you go. So when you have the

Constructor. So, name is Ravi Ashwin. So, this dot name is Ravi Ashwin. Right. So, now next case we will see.

```
1.  /* package whatever; // don't place package name! */
2.
3.  import java.util.*;
4.  import java.lang.*;
5.  import java.io.*;
6.
7.  class Student {
8.
9.      private String name;
10.
11.      // Constructor to initialize the student's name
12.      public Student(String name) {
13.          this.name = name;
14.      }
15.
```

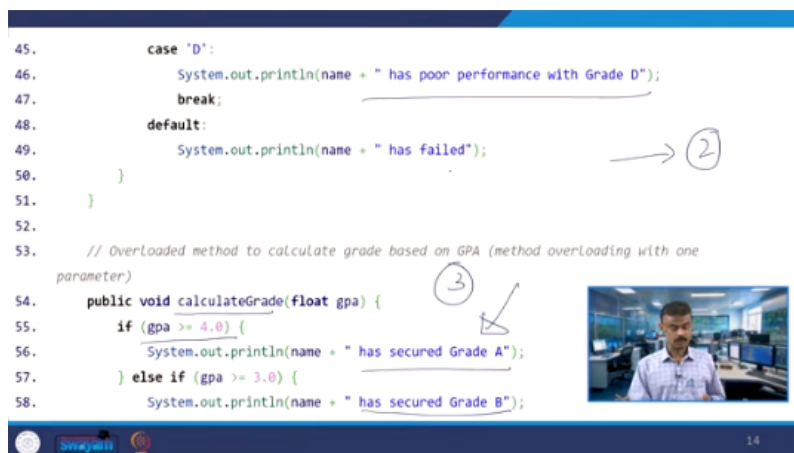


So, calculate. Student is invoking with help of dot operator. Calculate grade. You are passing 85.5. So, when you are passing 85.5.

Alright. So, when it is greater than 80. Alright. It is not greater than or equal to 90. False.

So, it is coming over here. 85.5, right? You are passing 85.5 here, right? So, it is obviously greater than or equal to 80. So, then in that case, the student Ravi Ashwin has secured grade B.

```
45.      case 'D':
46.          System.out.println(name + " has poor performance with Grade D");
47.          break;
48.      default:
49.          System.out.println(name + " has failed");
50.  }
51.  }
52.
53.  // Overloaded method to calculate grade based on GPA (method overloading with one parameter)
54.  public void calculateGrade(float gpa) {
55.      if (gpa >= 4.0) {
56.          System.out.println(name + " has secured Grade A");
57.      } else if (gpa >= 3.0) {
58.          System.out.println(name + " has secured Grade B");
59.      }
60.  }
```



So this will be printed. Ravi Ashwin has secured grade B. So that will be printed. That is your first output. And again student is invoking same calculated grade but this time it is passing B grade. So when you are passing this character letter grade.

So B. Case B. So case B it will go here. Ravi Ashwin has good performance with grade B will be printed. This will be printed. So, this will be printed, right. In the previous case, which one will be printed?

Gradner equal to 80, that means 85.5 Ravi Ashwin S. Court grade A, right. And what about the third case? Third case is student object is invoking calculate grade. You are passing 3.7 F, 3.7 floating point number you are passing. So, in that case, so what is happening?

Here it is. 3.7 floating point number. So is 3.7 greater or equal to 4.0? False. So it is coming here.

So 3.7 is greater or equal to 3.0 which is right. So it goes over here. So you will get `system.out.println` As secured grade B. All right. That is it.

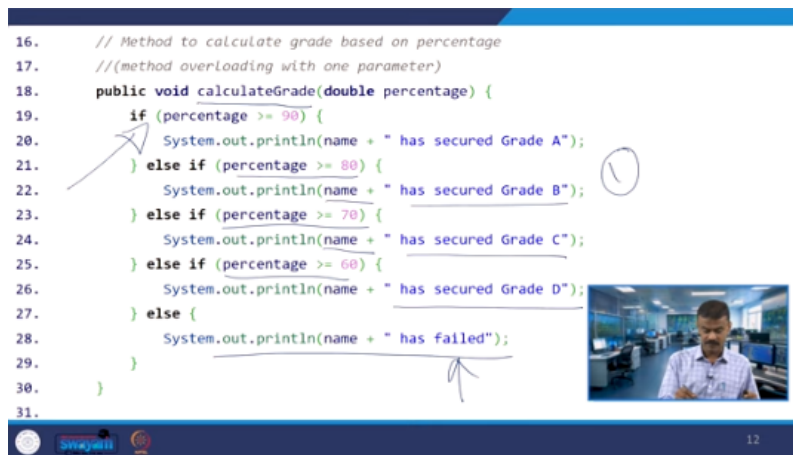
It will not go to else. It will not go to else. This else also will not be done. So as secured grade B will be printed. Ravi Ashwin as secured grade B will be printed.

All right. So these are all the three print statements. One is grade A, grade B and this is grade B. Let us see what is the output we are getting, right. So, here you go. So, the first one I think I answer 85 point, what is the score?

85.5, all right. So, 85.5, yes. I think by mistake I have done this one encircled. So, 85.5 means this particular case is being satisfied, all right. So, this particular case 85.5, all right.

This is not 85.5. 85.5 greater than or equal to 90 is false. So it is coming over here. 85.5 is greater than or equal to 80 which is true. So this one will be printed.

```
16. // Method to calculate grade based on percentage
17. //(method overloading with one parameter)
18. public void calculateGrade(double percentage) {
19.     if (percentage >= 90) {
20.         System.out.println(name + " has secured Grade A");
21.     } else if (percentage >= 80) {
22.         System.out.println(name + " has secured Grade B");
23.     } else if (percentage >= 70) {
24.         System.out.println(name + " has secured Grade C");
25.     } else if (percentage >= 60) {
26.         System.out.println(name + " has secured Grade D");
27.     } else {
28.         System.out.println(name + " has failed");
29.     }
30. }
31.
```



That is the first output. The second output is Ashwin has a good performance with grade B. And third output is this one. Ravi Ashwin has secured grade B. So right now this is a method. This is a function. So these three output we should get.

And here you go. Ravi Ashwin has secured grade B. And Ravi Ashwin has good performance with grade B. And the third one is Ravi Ashwin has secured grade B. So, in this lecture, so we talked about method overloading by taking two examples. And one example we had seen in terms of case study. So, the case study we have considered the university system with the help of method overloading.

And you can see the method overloading which has overloaded the percentage based grading. letter based grading and the GPA based grading. So with this I am concluding this particular lecture. Thank you very much.