FUNDAMENTALS OF OBJECT ORIENTED PROGRAMMING

Lecture10

Lecture 10: Constructors in Java - Default and Parameterized

Thank you. So welcome to the last lecture of classes and objects. In the last lecture, we had seen separately how to use default constructor and parametric constructors. So now, so can we use both in a single program, right? So that we are going to see in this case study or the problem, right?

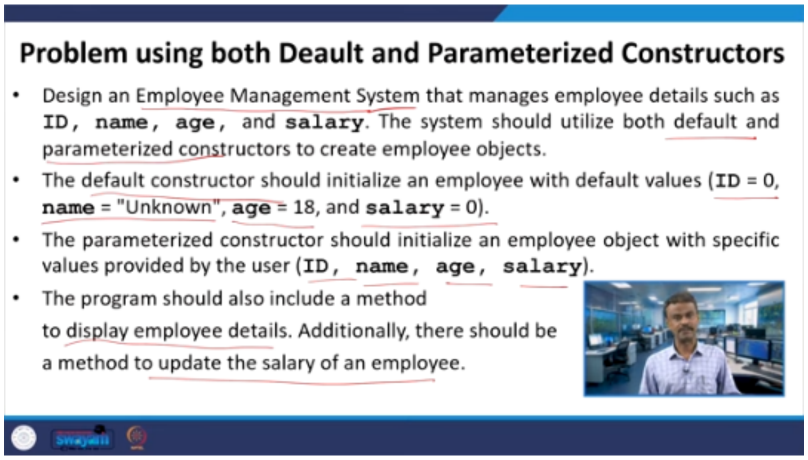
So let us consider employee management system, all right? So that is the class. So that has Employee details such as you have ID, name, age and salary, right. So, these are all the member data kind of, all right.

So, now the main idea is we have to use both default and parametrized constructors, right. So, in the same program. So, when you are going to create the employee objects. So, we are going to use default and parametrized constructors. So, how we are splitting this?

The default constructor should contains or initialize the employee with the values id is 0, name unknown, age should be 18 and salary is 0, ok. And then the parameters constructor should initialize. So, you should have a employee object, let us say another object, let us say E2 with certain values by the user. So, given by the user, the specific values given by the user for ID, name, age and salary.

Also, the program should include the display employee details. So, we have to display the employee details. Also, we have to do the update in the salary, right. For example, if some employee is getting the salary of 30,000, it should be 40,000. And similarly 3 lakhs, it should be 4 lakhs, something like that.

That should be update and that update we have to display. So the main idea here is how we can use both the default and parameterized constructors. So we will see the program, right. So we will see the C++ program. So I have class employee.



Problem using both Deault and Parameterized Constructors

- Design an Employee Management System that manages employee details such as `ID, name, age,` and `salary`. The system should utilize both default and parameterized constructors to create employee objects.
- The default constructor should initialize an employee with default values (`ID` = 0, `name` = "Unknown", `age` = 18, and `salary` = 0).
- The parameterized constructor should initialize an employee object with specific values provided by the user (`ID, name, age, salary`).
- The program should also include a method to display employee details. Additionally, there should be a method to update the salary of an employee.

So here I have 4 member data as suggested in the problem. So we have id whose data type is integer. Name whose data type is string. Age integer and salary is double. Okay.

So this is your member data. And here you have a constructor. Right. So this is the default constructor. You do not have any parameters.

So therefore you call this as a default constructor. So in the default constructor your ID is 0. Name is unknown. Age is equal to 18 salary is 0.0 right you have one cout statement the cout statement is printing id ok default constructor call the employee with id. So, this is in under quote id will be printed right in fact id is 0, 0 should be printed.

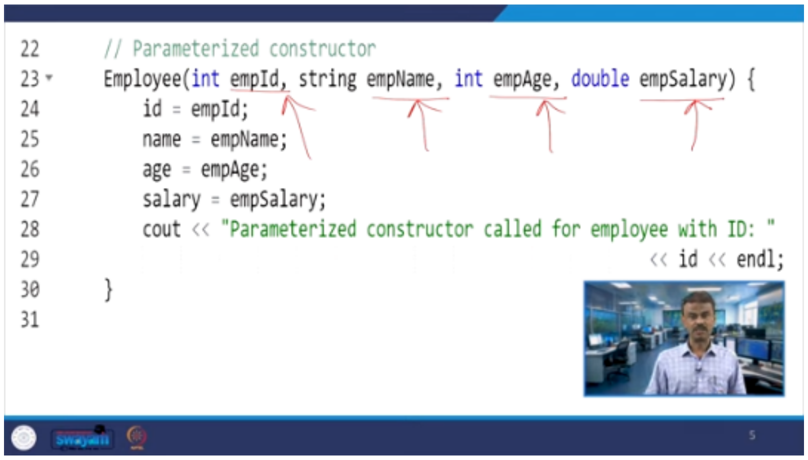
So, whenever you are creating an object right without any parameters. So, you call this as a default constructor. So, it will be automatically invoked and then the sequence of steps will be right executed and you have one CO statement. So, that means in the console you will get this particular output. So, now parameterized constructor.

So, here you go the parameterized constructor. So, you can see both default constructor and parameterized constructors. So, they are having same name as class. Also you can find the usual function we call this is a special member function because you cannot find any return type. So any member function you can find a return type.

It may be a y, it may be integer, it may be double etc. So whereas in this case you do not have any return type right. So that is why it is also called the special member function. So now the parameter is constructor you have employee id, string of employee name. Employee age and employee salary.

All right. So under this you have. Right. So whatever you are passing as a parameter passing as a value in the main. Right.

So that will be. Initialized to ID name age and salary. So again you are displaying the parameters constructor call for employee with ID. So whatever be the ID. So that particular ID will be printed.



```
    // Parameterized constructor
    Employee(int empId, string empName, int empAge, double empSalary) {
        id = empId;
        name = empName;
        age = empAge;
        salary = empSalary;
        cout << "Parameterized constructor called for employee with ID: "
                                                              << id << endl;
    }
```

So now, in the question it was asked, we have to do the display details, right? This is a special member function. And here you go, you are printing employee ID, which is ID, right? Name, you have name. Age, you have age, right?

And salary, you have the salary. right so this we are printing in the display details that means i want to print all the id name age and salary of the employees and then your update salary right so i am passing the update salary as a parameter right i am passing as the value all right so it will be the new salary under double right so the new salary will be assigned to salary that means there is a modification in the salary there is a updation in the salary so in the see out statement you can see salary updated for employee with id right so you have id and salary will be displayed and you can look here the class is ending over here okay so we have completed the class so now when we go to the main program so you have created The first object, EMP1, under employee class, you have EMP1, right? So, whenever you are creating this object, we know that, right?

So, whenever you are creating this object, so your default constructor will be automatically called, right? Employee 1, the default constructor. So, in the default constructor, so what is there? So, we had seen. So, the default constructor, you have ID 0, right?

All right. Name is unknown. Age is 18. Salary is 0.0. So initialize.

And then it is see out statement. And then you have one see out statement. So there you have default constructor call for employee with ID. All right. So the ID will be printed.

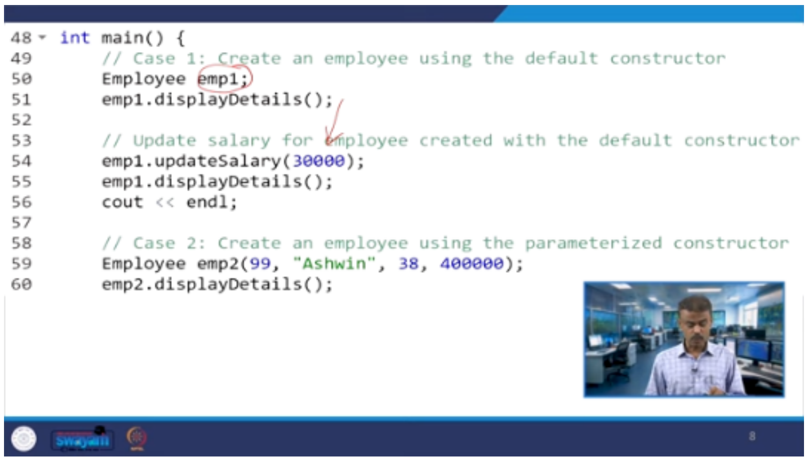
What is ID? ID is 0. So you will get the output. Default constructor call for employee with ID. ID.

Right. So that will be printed. Right. So this is what the next line, line number 51, you have EMP1 dot display details. So the display details, here you go, all will be printed.

Right. The ID is 0, name is unknown, age is 18 and salary is 0.0. So this will be printed. So we have done up to line number 51 and line number 54. Right.

EMP1 is invoking update salary. And then you are passing 30,000. So when you are updating this salary, so now new salary will be assigned to salary. So now that means when you are calling EMP1, that means EMP1 is invoking display details, there is a change. So there is a change.

So what is the change? You can have, so this is the one, 0, unknown, 18, 0.0. And then since you have this update, Right. So you will have salary updated for employee with ID 0 to 30,000.



```
48  int main() {
49      // Case 1: Create an employee using the default constructor
50      Employee emp1;
51      emp1.displayDetails();
52
53      // Update salary for employee created with the default constructor
54      emp1.updateSalary(30000);
55      emp1.displayDetails();
56      cout << endl;
57
58      // Case 2: Create an employee using the parameterized constructor
59      Employee emp2(99, "Ashwin", 38, 400000);
60      emp2.displayDetails();
```

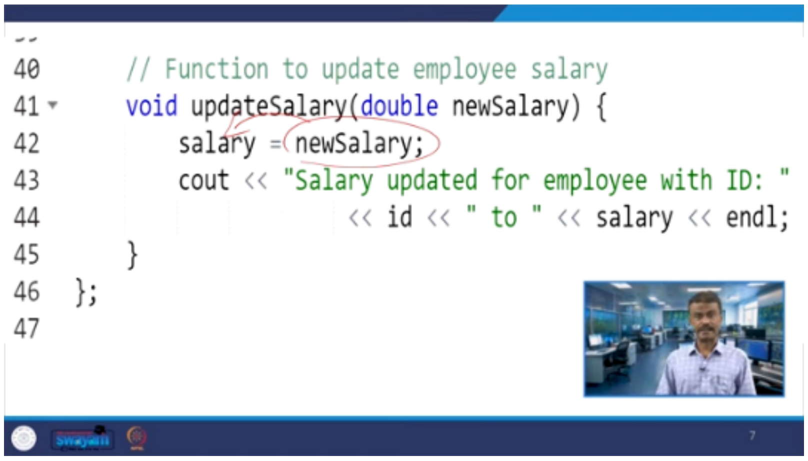
Right. So that will be printed for employee 1. Right. I hope it is clear. And then see out end L. That means it will go to the next line.

So now you are creating a second object. Right. And then you will have the parametrist. Right. So you are passing the parameters.

So therefore the parametrist constructor will be invoked. So, parameters constructor when invoked, you are passing 99, Ashwin, 38 and 400,000. So, that means when you go here, your employee ID. So, ID will be 99, name will be Ashwin, age 38 and salary will be 400,000. So, the ID is 99 will be printed over here.

right. So, when EMP2 is invoking display details, all these information will be printed in line number 60 and there is a salary update, right. That 4 lakhs will be now 5 lakhs, 400,000 will be now 500,000. So, in that case, when EMP2 is invoking update salary 500,000, so you can see that, right. So, here the new salary is

500,000, so that will become the salary. In line number 42, that will be assigned to salary. So, salary updated for employee with ID 99 to 500,000 will be printed, right. So, this is what exactly happening and then you have return 0, then you can see the list of output, right. So, initially your employee ID is 0, right.



```
40      // Function to update employee salary
41      void updateSalary(double newSalary) {
42          salary = newSalary;
43          cout << "Salary updated for employee with ID: "
44                      << id << " to " << salary << endl;
45      }
46  };
47
```

So, whenever you are creating an object, right whenever you are creating an object so in the default constructor so you have this display message right in the default constructor alright so in the default constructor you have this message what is the message the default constructor called for employee with id 0 right so that will be printed so it is what you are getting as a first output and then employee details employee id is 0 name is unknown age is 18 and salary is 0 and similarly employee id right for there is a salary update yes there is a salary update so the salary update function when you are calling right when employee 1 is invoking so you are getting this output salary updated for employee with id 0 to 30000 right employee id 0 so now he or she is going to get 30,000. That is the meaning.

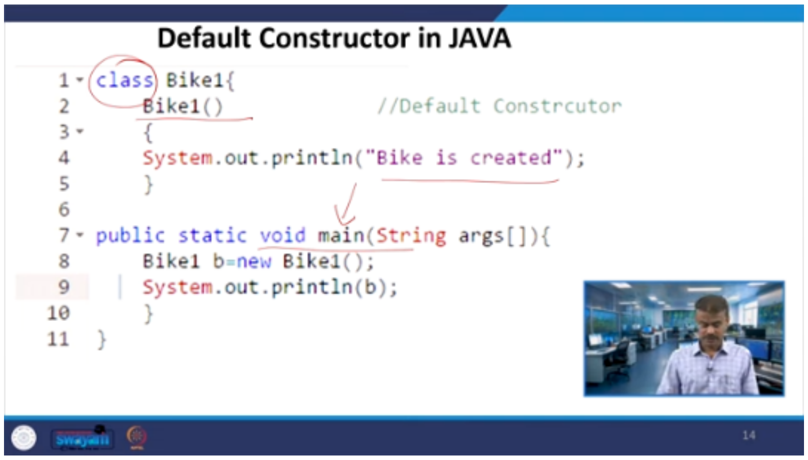
And rest of them are similar, ID, name, age. And if you look salary now, it is becoming 30,000. So this is the default constructor. And in the parameterized constructor, right, so whenever it is calling, when you are creating an object EMP2, right, under the employee ID 99, right, so employee ID is 99, name is Ashwin, Age is 38 and salary is 400,000, right?

So, initially and there is a salary update. Salary update has gone from 400,000 to 500,000. So, when EMP2 is invoking, right, the details, all right, employee details or updated details. So, salary is updating from 400,000 to 500,000 so that will be displayed for id 99 to 500,000 and rest id name age is same and salary there is a change now the display is 500,000 right i hope you have understood this particular program so here we can see right in this problem so they are asking us to use both

constructors default constructor as well as the parametric constructor and i hope it is very clear now so similarly in java we will see the concept of default constructor right so let us consider in this case there are two class right in this case there is a class bike one right and then you have in fact this is a driver class right so you have void main so therefore you call this as a driver class so you have only one class And in this class itself, you have a void main. So, you call this as a driver class in Java, right. So, the class bike1 and this carries same name. So, we know that from C++.

So, we know that this is a constructor and no parameter. Therefore, you call it as a default constructor. So, here system.out.println, it is equivalent to cout, bike is created, right. So, this is what it is going to display in the case of construct when you are creating an object. So now here you are creating an object.

So slightly different from our usual C++. So everything is dynamic in Java. So here you have an object small b under the class bike1. And dynamically you are creating a memory for bike1. So new is the operator for dynamically allocating a memory, the new operator.



Default Constructor in JAVA

```
class Bike1{
    Bike1()                 //Default Constrcutor
    {
    System.out.println("Bike is created");
    }

public static void main(String args[]){
    Bike1 b=new Bike1();
    System.out.println(b);
    }
}
```

And bike1, right, it is a constructor. Correct? So b. So this B will have, will be in some address, right? This object, object B will be in some address.

So now we are printing out system.out.printl and B, right? So let us see what is the output we are going to get. First when you are creating an object, bike is created, will be printed. And the second one, right, so we will see. This is the output we are going to get.

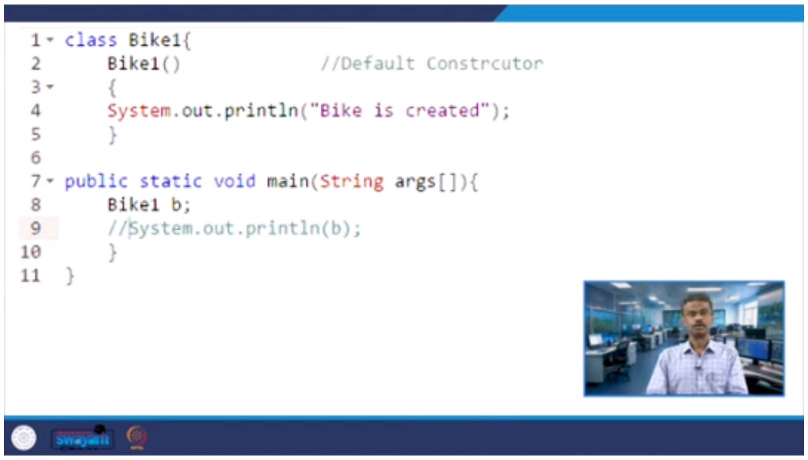
So the second output, so we have run in shell, right, the Linux. So javac is for compilation. Bike1.java is the name of the file. And the second one is running the program, java blank space bike1, right. So you can see the output.

Bike is created, will be printed. So that means whenever you are creating an object, Default constructor will be invoked and bike is created will be printed. All right. And next one you call this as the identity hash map.

All right. You call address in C++. All right. You call this as address in C++. In Java you call this as identity hash map.

So when you are trying to print B the object in fact the location is equivalent to this. All right. The location. The location in Java, you call it as identity ashmap. Alright?

So, that identity ashmap will be printed. So, the address location. In other words, it is called address location. So, now, what will happen if I create exactly like your C++? Bike B. Right?



```
1  class Bike1{
2      Bike1()              //Default Constrcutor
3      {
4      System.out.println("Bike is created");
5      }
6
7  public static void main(String args[]){
8      Bike1 b;
9      //System.out.println(b);
10     }
11 }
```

Bike B. I have not put any print statement so when you are creating an object so what is happening so we will see right I try to compile and run the code right so you can see unlike in C++ first of all it is not calling the default construct program is running right I can able to compile and run the code right so whereas if you look I am not getting any output Correct? So, the object is being created, but it is not invoking the default constructor. So, line number 8.

So, that is the reason. So, it should be the dynamic in nature. So, if you look at line number 8. So, this is the right way to do in Java. So, bike1 b, that means b is the object, bike1 is a class that is equal to the new operator and bike1.

Exactly like the dynamic array that you do it. Right? It is a dynamic object. right so we have done right or you might have done int star a I put star in C++ equal to new int of 10 dynamic array of length 10 right or this is in C++ the same code in Java int you have the array notation a equal to new int of 10.

So, when you create an array you use this. So, similarly here new right new and then bike 1 right the same integer data type. So, here we are writing int with the size. So, here you are writing bike 1 it is like a constructor exactly you are putting the constructor syntax annotation. But the problem is when you are not dynamically allocated the memory, so even the default constructor will not be called.

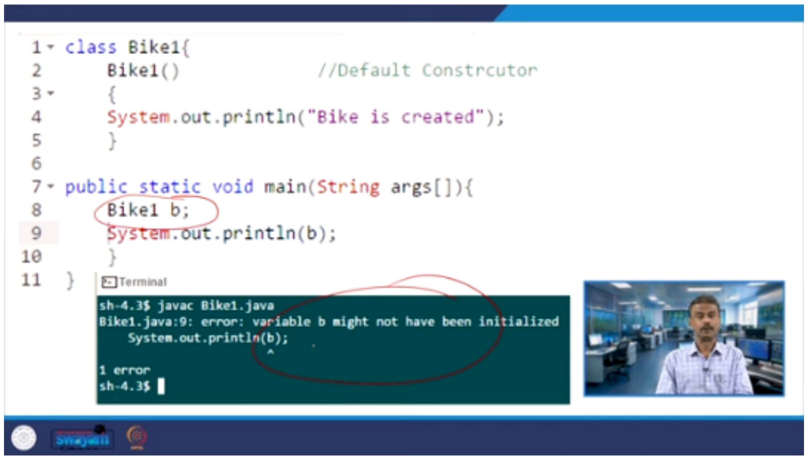
So now we will see. The program will run. It is not giving any error. So I can able to compile. Java C stands for compile.

Bycone.java is the name of the file. And for executing or running the code, we use Java bycone without extension. So you can try in Linux. Or you might have already done in the case of the basic programming right procedural oriented programming so now what i will do i will just remove the comment statement right so here line number nine i made a comment statement i am able to right run the code but with comment statement what is happening right so without comment statement what is happening so now if i try to run this code so i will get an error

Right. So B is not being initialized. Right. The error message that I am getting B is not initialized. Right.

So that means you are not able to dynamically create a memory for B. So where it is being stored, the compiler does not know. Right. So when I try to compile itself, you are getting the compilation error. Right. Where it is being stored and then you are trying to write the storage location.

Right. You are trying to write the identity ashmap or you are going to find out the identity ashmap. Right. So therefore, if you look, I am getting an error. Right.



```
1  class Bike1{
2        Bike1()              //Default Constrcutor
3        {
4        System.out.println("Bike is created");
5        }
6
7  public static void main(String args[]){
8        Bike1 b;
9        System.out.println(b);
10       }
11 }
```

```
sh-4.3$ javac Bike1.java
Bike1.java:9: error: variable b might not have been initialized
    System.out.println(b);
                       ^

1 error
sh-4.3$
```

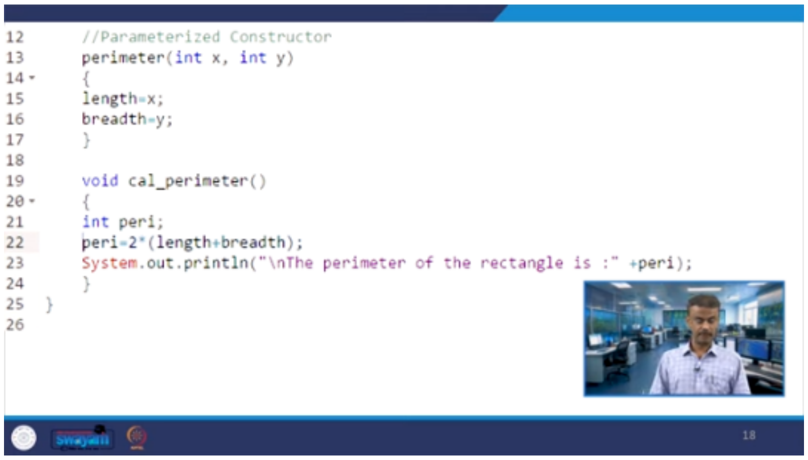
So line number eight, what we have correctly did. So this is required. Right. So dynamically, we have to allocate the memory. So maybe we look at one program with the combination of both default and parametrize constructor in Java.

All right. So let us consider class perimeter. Right. So let us consider class perimeter. So there you are having two member data.

One is length and another one is breadth. Right. So here you have the default constructor. So under this default constructor, your length is zero and breadth is zero. Right.

Your length is zero. And breadth is 0. In the default constructor you are initializing length equal to 0. And breadth is equal to 0. Alright.

So now we will see the parametrist constructor. Parametrist constructor you have two parameters. Right. So I have two arguments. One is int x and int y. So x I am assigning to length.

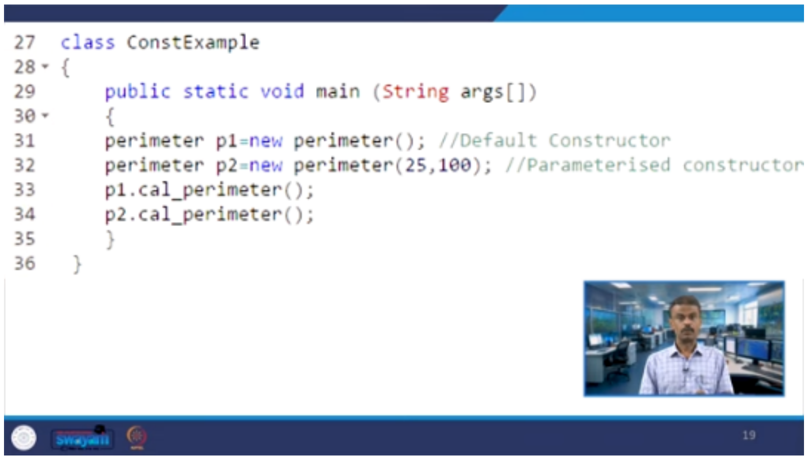


```
12      //Parameterized Constructor
13      perimeter(int x, int y)
14      {
15      length=x;
16      breadth=y;
17      }
18
19      void cal_perimeter()
20      {
21      int peri;
22      peri=2*(length+breadth);
23      System.out.println("\nThe perimeter of the rectangle is :" +peri);
24      }
25  }
26
```

And y I am assigning to breadth. Alright. So now I have one method. So the member function in C++, in Java, you call it as a method. So cal underscore perimeter.

We are calculating the perimeter. So the perimeter is nothing but 2 into length plus breadth, right? The perimeter is nothing but 2 into length plus breadth, okay? So it is printing. So whatever you are calculating will be assigning to peri.

And system dot out dot printer length, the perimeter of the rectangle is, right. So, you are printing the perimeter. And line number 25, your class is getting over, right. Your class is getting ended over here. So, now, if I go to the main, all right.



```
27  class ConstExample
28  {
29      public static void main (String args[])
30      {
31      perimeter p1=new perimeter(); //Default Constructor
32      perimeter p2=new perimeter(25,100); //Parameterised constructor
33      p1.cal_perimeter();
34      p2.cal_perimeter();
35      }
36  }
```

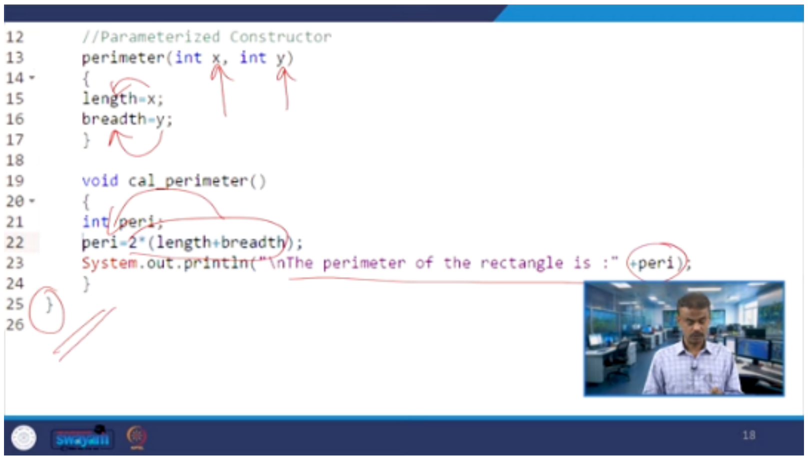
So, class const example is a driver class, right. So, you have two classes here. So one class is perimeter, right? Since you do not have a main over here, this is not a driver class. And here you go.

So const example is the driver class. So in the driver class, you have a main program, right? So I have two objects here, right? One is perimeter P1. P1 is an object.

We are using the new operator and perimeter means it is a default constructor, right? So now you are understanding, right? So the bike, we have done the same thing. Bike 1 B equal to new bike 1, right? It is a default constructor it is calling.

And here line number 32 you can see perimeter P2. P2 is an object. So that is equal to new perimeter of 25 comma 100. You are passing 25 comma 100. So when I am creating P1, right?

What is happening? So when I am creating P1. So when I am creating P1. The default constructor will be called. Your length is 0 and breadth is 0.



```
    //Parameterized Constructor
    perimeter(int x, int y)
    {
    length=x;
    breadth=y;
    }

    void cal_perimeter()
    {
    int peri;
    peri=2*(length+breadth);
    System.out.println("\nThe perimeter of the rectangle is :" +peri);
    }
}
```

That's it. Right. Your length is 0 and breadth is 0. I hope you understand. Length is 0 and breadth is 0.

And when you are creating an object P2. You are passing 25 and 100. Right. When you are calling P2, you are passing 25 and 100. So what is happening?

We will see. When you are passing 25 and 100, so this is your parameterized constructor. X will take the value 25 and Y will take the value 100. So now that will be assigned to length and breadth for object P2, right? So this is what exactly happening, right?

So now we will see what are the further methods it is calling. So, now P1 is invoking the method cal underscore perimeter, right. So, P1 is invoking cal underscore perimeter. So, when P1 is invoking this, so here you go, right. So, you have a local variable peri.

We know for the P1 length is 0 and breadth is 0, right. So, this is what happening in the default constrictor. So the default constructor when I go back, so you can see over here, your length is 0 and breadth is 0, correct? So now when it is calculating 2 into 0, so the result should be 0 in the case of first object, right? Your x is 0 and y is 0, right?

So the perimeter should be 0. And in the second case, you are passing 25 and 100, right? When it is calling, calculating the perimeter function, right, cal underscore perimeter method, right, what is happening? You are passing x 25, right, length becomes 25, right.

So, length becomes 25 and breadth becomes 100. So, we know that 100 plus 25, 125, 125 into 2. So, I have to get the output 250 for second case, right. I have to get 250 for the second case. So first output is 0, second output should be 250.

So this is what the output we have to get. So when I run the code, right, so javac, so const example.java is the name of the program, right, name of the file, const example.java. So javac is the compilation. And next one, next line, we are running the code, execution. So javaconst example without extension.

So, here you can see the output. The first output is the perimeter of the rectangle is 0, all right. Even you can modify the perimeter of the first rectangle or the first object, right. So, whatever you are writing, this is going to be printed, right. So, the perimeter of the rectangle, in fact, yeah.

So, here you can modify. So, here write line number 23. So, we can modify accordingly. But the case is you have the member 0. function or method so you have written the perimeter of the rectangle so that is what it is printing maybe what we can do in the constructor itself we will write cout some statement right so we are trying to find out the perimeter of the rectangle correct something like that otherwise in both the cases we are getting the same output so you may ask the question right so this we can control in the constructor you make one cout statement

I mean to say system dot out dot println statement. So, the perimeter of the rectangle is 0 and the perimeter of the rectangle you are getting 250, right. So, in this chapter, we had seen extensively, right. You have in fact, initially I started with objects and classes, right. So, you know how to define a class and how to create the objects, right.

And now, you can also know how to create the dynamic objects with the help of, alright, in Java, right. So, you know the difference between the static object and dynamic object. So, in particular, alright, you can use a static object in C++, which is not a problem. It is still called the default constructor. Whereas, we are finding some problem in Java when you are creating only the static object

The program is running, but when you are trying to find out what is the location or identity ashmap, it is giving an error. So, in the case of Java, so what we have to do, we have to create the dynamic objects. So, once you are creating the dynamic objects, the syntax should be like a line number 31 and 32, right. So, you are creating two objects, P1 and P2. So, P1 you can see the right hand side new that is the operator and then you have the constructor.

So, that constructor is nothing, but the default constructor all right. So, this is nothing, but the default constructor and the second case it is a parametric constructor. So, P2 your classes perimeter with the help of new operator. So, you have default constructor. So, you can try the similar code in C++ also.

Alright. So, the dynamic nature it will work because in the case of array I gave an example int star a equal to new int of 10. Alright. You are using the star operator. Alright.

The dereference operator. Correct. So, in a similar way when you have the constructor like a default constructor and parameterized constructor when there is no arguments you call this as a default constructor. So, this is a default constructor. And if you are passing certain parameters.

Right. If it is having or you are passing certain values. Or in the constructor if it is having some arguments. You call this as a parametrized constructor. So you know whenever you are creating an object.

So which one should be called. And as I also said the constructors are nothing but the special member function. Or the special method. It looks exactly like a method. Or it is looking exactly like a function.

But it does not have. Any return type. So this is what. In Java also we had seen. So line number 13 if you look.

It does not have any return type. And they have the same name as of class. So therefore. It is called the constructors. So in both the programs we had seen.

Usage of default. And parametric constructors. In the same program. So, we had seen different examples, the default parameter is separately and in C++, we had solved one case study, the combination of default and parametrics. And similarly, in the case of Java, we had seen the default constructor and also one example of using both default and parametrics constructor.

So, in this chapter, we extensively studied this class objects and the constructors. So, with this I am concluding. Thank you.