# FUNDAMENTALS OF OBJECT ORIENTED PROGRAMMING

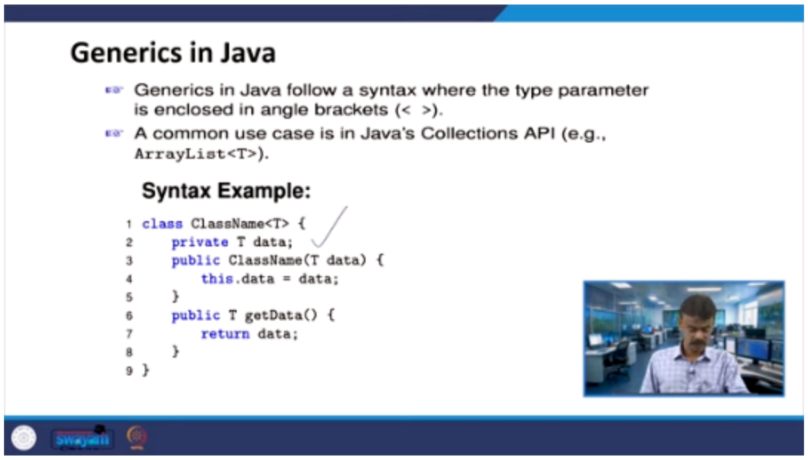
## Lecture38

## Lecture 38: Generics in Java

Welcome to lecture 38 templates and generics. So, in the last class we talked about template in C++ and the basic introduction on Java, right? So in Java, you call the same template as generics, right? So here what we do, we use the angle bracket and in the angle bracket, assume that let us have name of the class, right?

So usually what we do, class, class name and then we proceed. But here if you look, After class name, you have the angle bracket and t, exactly like what we have studied in C++, right. So, this you call it as generics in Java. And then, when you are declaring, let us say, the variable, right, you call it as a member data or data member, t data, right, private t data.

So, suppose you want to use integer, t becomes integer. When you want to use character, t becomes character and so on. These things you know. And then if you look at the constructor right one argument constructor class name data which is t right and then data you are copying into this pointer this dot data and one member function you are finding t get data right and it is returning data. So, this is how the syntax works in Java and you call this is an example for generics in Java.



## Generics in Java

- Generics in Java follow a syntax where the type parameter is enclosed in angle brackets (< >).
- A common use case is in Java's Collections API (e.g., ArrayList<T>).

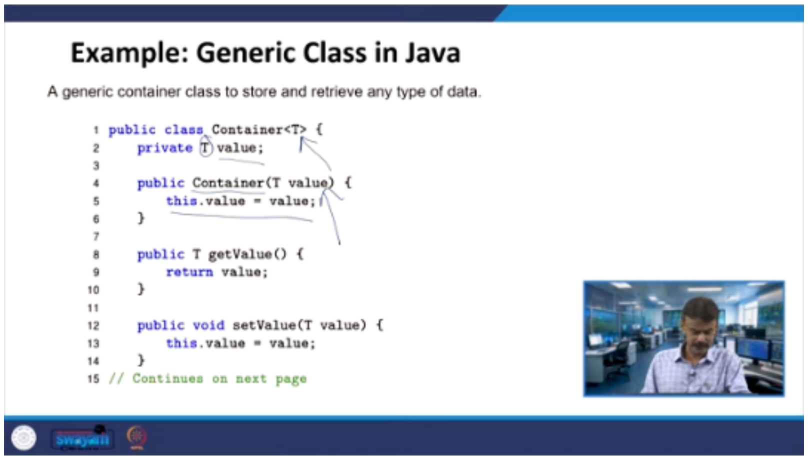
**Syntax Example:**

```java
class ClassName<T> {
    private T data;
    public ClassName(T data) {
        this.data = data;
    }
    public T getData() {
        return data;
    }
}
```

So, now let us see this particular program. So, here you have the name of the class is container. So, container and then you have the generic. So, you call it as T and you have a private data member value. So, you have a private data member value and here you have the generic.

That is a type and you have A constructor called container, all right. So, you are, it is a one argument constructor. We will pass value and this value will be copied properly when you use this dot value. Otherwise, it will be 0, right.

Initial value will be there. If you are not, you are simply assume that you are returning value or something or value equal to value if you put, that is also wrong. So, this dot value, we have seen this pointer, right. This dot value equal to value. And you have one member function.



## Example: Generic Class in Java

A generic container class to store and retrieve any type of data.

```java
public class Container<T> {
    private T value;

    public Container(T value) {
        this.value = value;
    }

    public T getValue() {
        return value;
    }

    public void setValue(T value) {
        this.value = value;
    }
// Continues on next page
```

You call it as a method in Java. One method, get value. That is returning value. Get value return type is T. And you have another method called set value. Correct.

So you are going to pass one argument. Right. Pass one parameter. You have one argument. That is a method.

Set value method. So set value method value is copying into this dot value. Right. So you have main method here. So that means this is your container is your driver class.

Right. Container is your driver class. So now you have main. I am creating an object in container, right? Integer container.

You have a memory. You are allocating a memory, right? And then you have the constructor called container is under class, right? So I said generic. So you are using, right, the generic notation, generic syntax.

And then 123, you are passing 123, right? So when you are passing 123, we know that it is an integer. And in fact, here you are writing container integer, right? So, here it will call the constructor container, correct. It will call the constructor container value is 123.

This dot value 123 means, so value will get actual 123. That is the meaning, right. Value is 123. And the second case, you have another object string container, right. First object is int container.

Second object is string container under container class. So, you can have the generic is string here generic is string here allocating a memory and here you have the constructor with the generic notation and then the string is generics in Java right the string is nothing but generics in Java fine. So, you are passing this that means now the second case T is becoming string and value that you are passing generics in Java. So now the second case value is nothing but whatever string you passed, right? So what string we have passed?

Generics in Java, right? So next we are printing system.out.println. You are printing integer value. So int container is invoking get value, right? Int container.

So this object is invoking get value. So when it is invoking get value, value will be returned, right? You know value is 123. So in this case, we are getting 123 as the output. And in the second case, the string container is invoking get value.

We know that value is nothing but the string. The string is nothing but generics in Java. So the second case, the generics in Java will be printed. So generic in

Java will be printed. right so let us run the code we will go to the java console yes so here you go the same program whatever i explain the same program so the java when you run right so if you run the code here you are seeing integer value is 123 and you are getting generics in java right so if you look we are writing only right we have one class right the generic class right container t

And then when you are creating the objects, so one is the integer object, another one is a string object. All right. And then the constructor will be called this one argument constructor. And accordingly, the values are getting printed. So you are seeing integer value 123.

Another one is string value generics in Java. I hope it is clear. It is not very difficult. Similarly, like a template function we saw in C++. All right.

Here, we call it a generic method. The same when you put the type name T we wrote in C++. Here, what we do is we use it like this. I mean, you can see here if you look at this particular line. So it is a method.

The name of the method is print array. And then what you are doing is using the generic syntax. Then the return type is void. That means nothing will be returned. And then you have the name of the method.

And here you go, T that is a array. So, array you can use it as integer, double or character, string, whatever you want, right. So, T and then you have a array notation, then array, right. So, now you have the for loop. So, you take T element, that means the base address of an array and the base address of an element will be same, right.

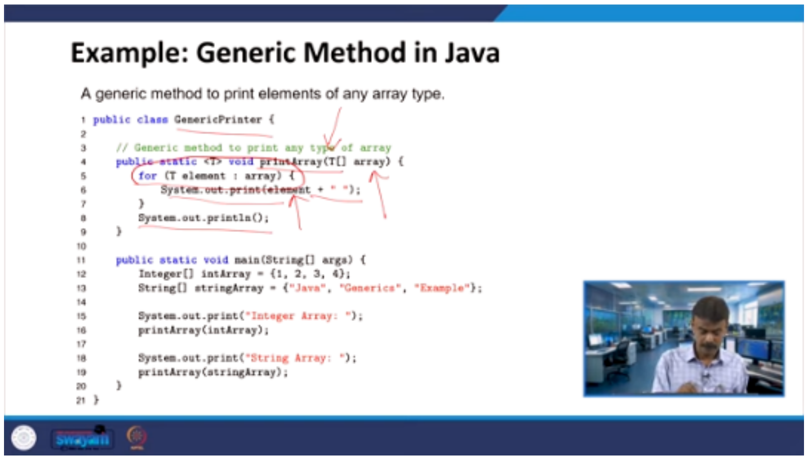
In the case of the identity hash map, Java it is identity hash map. Identity ashmap of array and identity ashmap of element is same now. So when you are printing, you are printing the complete array. That is the meaning over here. System.out.printl an element when you print, right?

So you have to get the complete array, right? So maybe we will take one example and I will explain in detail, right? So let us consider the class generic printer, fine? So now you have the method printArray. all right so the method print array so you have an array and here you go the generic type right the generic type so whatever we are using assume that we are using integer array t will become integer if you are using character or string t will become string okay so for t elements so whatever I have written in the previous slide so system dot I out dot println element

And then you are printing one blank space also. All right. So as I told, this is a for loop. So T element colon array. The meaning is both identity hash maps are the same.

Identity hash maps are the same element and array. Right. So then you are printing the element. So the complete array will be printed. And then you are printing one blank space.

Now your class is over here, and line number 9 is going to the main program. So your method is over. The print array method is over here, not the class. So it is continuing, which means it is the driver class. Generic printer is the driver class.



## Example: Generic Method in Java

A generic method to print elements of any array type.

```java
1  public class GenericPrinter {
2  
3      // Generic method to print any type of array
4      public static <T> void printArray(T[] array) {
5          for (T element : array) {
6              System.out.print(element + " ");
7          }
8          System.out.println();
9      }
10 
11     public static void main(String[] args) {
12         Integer[] intArray = {1, 2, 3, 4};
13         String[] stringArray = {"Java", "Generics", "Example"};
14 
15         System.out.print("Integer Array: ");
16         printArray(intArray);
17 
18         System.out.print("String Array: ");
19         printArray(stringArray);
20     }
21 }
```

So now, in the driver class, you have the main method. You are declaring, let us say, an integer array. The integer array notation is: integer array is 1, 2, 3, 4. So, the name of the array is int array, and the second one is string array, right?

So, the name is string array. You have a Java generics example, right? So, string array of 0 is Java. String array of 1 is generics, and string array 2 is example. So, now what do we do?

We will call system.out.print; the integer array is getting printed, right? So, now we are calling print array by passing int array, right? So, this is being called. The int array is being called, right? Print array is being called.

You are passing an int array. That means the array is now becoming an int array, and t is nothing but an integer, right? Your t is nothing but an integer now. And then you have an array, correct? So, what is the array?

1, 2, 3, 4. So, t, as I said, is an integer. So, for each integer element in the array, right? So, you are printing: System.out.print(element), plus you are printing a blank space. So that means we will expect 1, 2, 3, 4.

1, blank space, 2, blank space, 3, blank space, 4. So that will be printed. Similarly, when you go System.out.print(string array). So, print array, string array. So that means print array is again being called, and you are passing a string array.

So when you are passing a string array, now your array is becoming, right? Now your array is becoming a string array, right? And what is T? T is a string, right? The complete T will be a string.

So for a string element colon array. So both are pointing to the same hash map, and System.out.print element plus blank space. So what are the elements? Java blank space, generics blank space, and example blank space. So, let us see this program in the Java console.

So, you can see the same program I explained. So, here you can see when you have an integer array, right? It is printing 1, 2, 3, 4, and for the string array, we are getting Java blank space, generics blank space, example, and also you will have a blank space, okay. So, this is how. Your method is working. So we have written only one method.

Print array. It works for integers and strings. Exactly like what we saw in C++. There, you call it a template. So here, generics will work like this.

Okay. So this is what I explained. Now, we will see one more method. So, we have to be familiar with generics. Right.

So, we will take one example. A generic method to find the maximum in Java. So, let us have a class, GenericMaxFinder. So, here we are using one syntax. In line number 4, T extends Comparable.

So, the comparable is nothing but the interface. So, T is extending comparable. So, you can consider T, which is extending the interface comparable. Again, you are putting T. So, it is an inbuilt. Comparable is an inbuilt.

So, and then you have method called findMax, right? You have a method called findMax. Under this, you have a, b, c, right? You have a data type t. So, t will take integer or double or character, right? So, that we know.

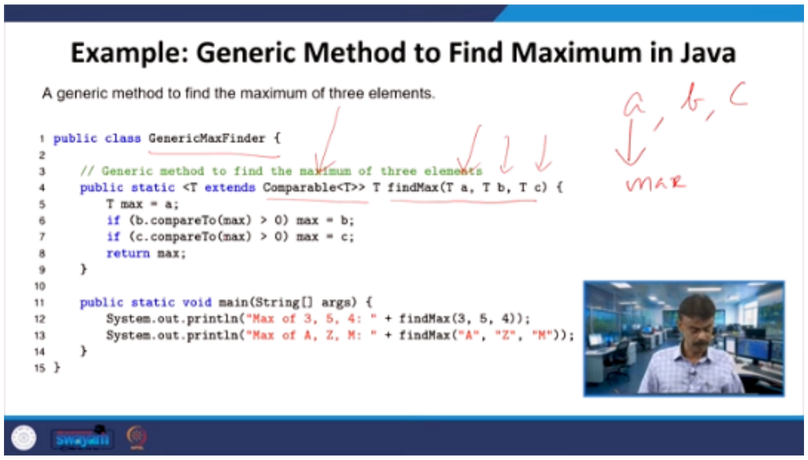
So, now you are putting t max is equal to a, all right? So, the comparable, we will have this compareTo function, right? CompareTo method that is a inbuilt. Right. Which is available and comparable.

That is what we extend. Right. So we are making use of some of the methods. So you can also write directly. But we are making use of some of the methods.

So initially we assign. So a, b, c are given numbers. So initially we put a as maximum. Right. Initially we will put a as maximum.

Right. And then you are comparing b dot compare to max greater than 0. Right. b dot compare to max greater than 0. Max will be equal to b.

right so that means a and b you are comparing right the meaning is so b minus a right b minus a if it is greater than 0 so obviously what is the implication b is greater than a so when b is greater than a what you are saying max equal to b that is what the logic we are writing right so next you know max right you are keeping max if it is not max will be a right if it is not satisfying this inequality if b minus a is less than 0 so b will be less than a So naturally A is maximum, right? So this will hold. So now again comparing this with C, right? So whatever be the maximum, that will be compared, all right?



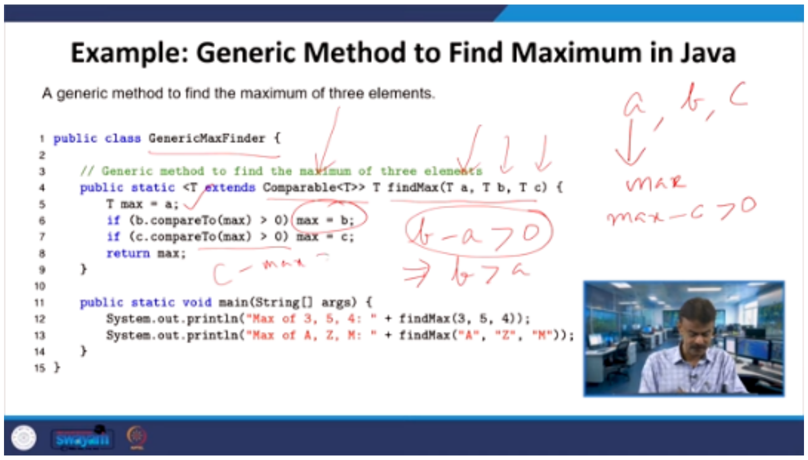
## Example: Generic Method to Find Maximum in Java

A generic method to find the maximum of three elements.

```java
public class GenericMaxFinder {

    // Generic method to find the maximum of three elements
    public static <T extends Comparable<T>> T findMax(T a, T b, T c) {
        T max = a;
        if (b.compareTo(max) > 0) max = b;
        if (c.compareTo(max) > 0) max = c;
        return max;
    }

    public static void main(String[] args) {
        System.out.println("Max of 3, 5, 4: " + findMax(3, 5, 4));
        System.out.println("Max of A, Z, M: " + findMax("A", "Z", "M"));
    }
}
```

So if a max minus, let us say C, which is greater than 0, correct? So then max will be as it is. Otherwise, C is maximum, right? So here you go like this, C minus max. In fact, the meaning here is C minus max, right?

greater than 0. So C greater than max. So C greater than max means max will be equal to C. That is the meaning. Okay. So we found maximum.



## Example: Generic Method to Find Maximum in Java

A generic method to find the maximum of three elements.

```java
public class GenericMaxFinder {

    // Generic method to find the maximum of three elements
    public static <T extends Comparable<T>> T findMax(T a, T b, T c) {
        T max = a;
        if (b.compareTo(max) > 0) max = b;
        if (c.compareTo(max) > 0) max = c;
        return max;
    }

    public static void main(String[] args) {
        System.out.println("Max of 3, 5, 4: " + findMax(3, 5, 4));
        System.out.println("Max of A, Z, M: " + findMax("A", "Z", "M"));
    }
}
```

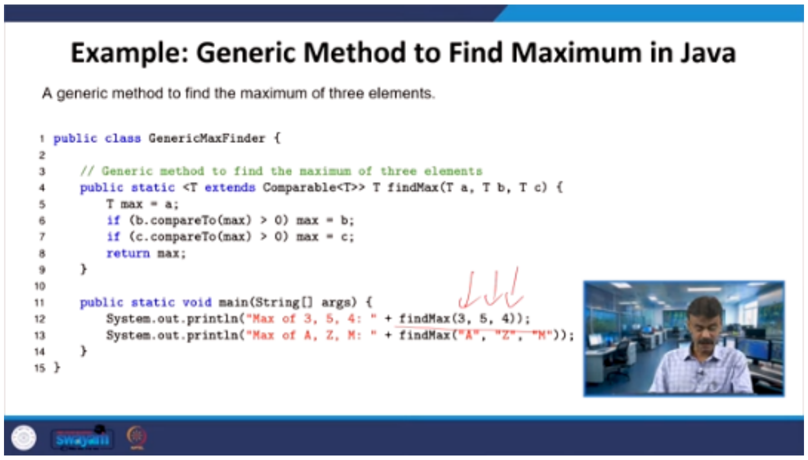
So this is one way, or if you are not comfortable with using comparatives, you can write the usual logic. Okay. So now, in the driver class, you have a main method. So in the main method, you have arguments. A string array.

Right. You have arguments and a string array. So, System.out.println. We want the maximum of 3, 5, and 4. Right.

You are calling the method. Find max. By passing 3, 5, and 4. When you are passing 3, 5, and 4. They are nothing but integers.

A is 3. B is 5. And C is 4. They are nothing but integers. Correct.

So they are nothing but integers. And once it becomes. All t becomes int. Correct. All right.



## Example: Generic Method to Find Maximum in Java

A generic method to find the maximum of three elements.

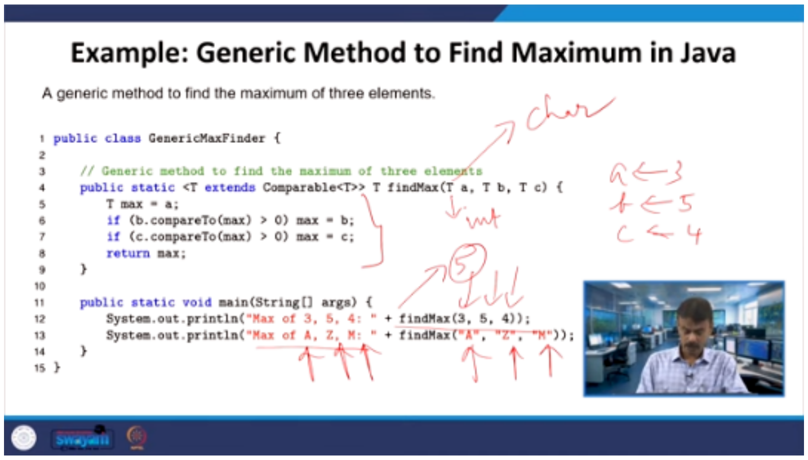
```java
public class GenericMaxFinder {

    // Generic method to find the maximum of three elements
    public static <T extends Comparable<T>> T findMax(T a, T b, T c) {
        T max = a;
        if (b.compareTo(max) > 0) max = b;
        if (c.compareTo(max) > 0) max = c;
        return max;
    }

    public static void main(String[] args) {
        System.out.println("Max of 3, 5, 4: " + findMax(3, 5, 4));
        System.out.println("Max of A, Z, M: " + findMax("A", "Z", "M"));
    }
}
```

So you will have 3, 5, 4. A is 3. B is 5. And C is 4. So according to this logic, we know that the maximum is 5.

All right. So here, you have to get the output 5. Correct. And next, you want to find the characters. Maximum of three characters.

A, Z, and M. Right. You are passing A, Z, M. So we know the ASCII value. The ASCII value of capital A is what? 65. Right, and then it goes B, C, D, M, right? And then Z. Obviously, here, the maximum is Z, right? So find out the ASCII value of Z, and we know that obviously A, B, C, D... when it goes to Z, it is the maximum. So when you do the same logic, you are passing now T, which is nothing but what character? All the T is becoming a character.

And then, right, how is it finding? It is finding based on the ASCII value. So, the comparison will be made based on the ASCII value. So, we know that in that case, Z will be the maximum, right. One output we will get is 5.



## Example: Generic Method to Find Maximum in Java

A generic method to find the maximum of three elements.

```java
public class GenericMaxFinder {

    // Generic method to find the maximum of three elements
    public static <T extends Comparable<T>> T findMax(T a, T b, T c) {
        T max = a;
        if (b.compareTo(max) > 0) max = b;
        if (c.compareTo(max) > 0) max = c;
        return max;
    }

    public static void main(String[] args) {
        System.out.println("Max of 3, 5, 4: " + findMax(3, 5, 4));
        System.out.println("Max of A, Z, M: " + findMax("A", "Z", "M"));
    }
}
```

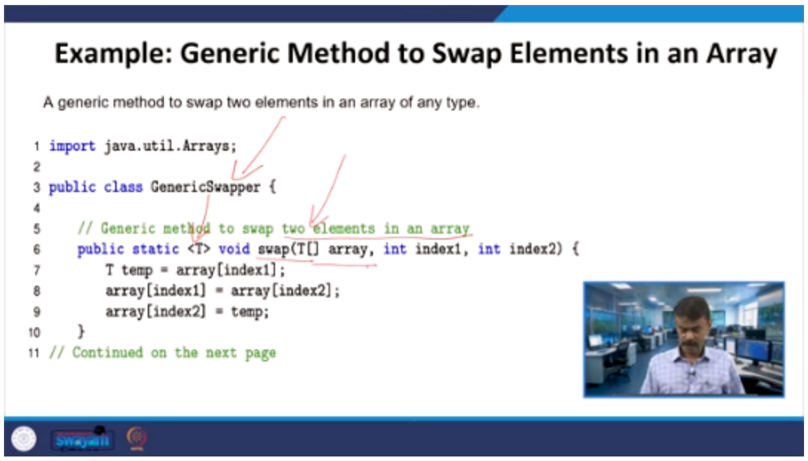
Another output we will get is Z, okay. So, let us see. Let us run this code. Yes, this is what I explained. So, you can see that the maximum of 3, 5, 4 is 5 after running the code.

And the maximum of a, z, m is z, right? I hope it is clear for everyone, right? So, this is what I explained, right? So, this is for your reference. You can see that again, right?

And then you can find the maximum of three integers and three characters. So, now we will see one more example. Swapping of, let us say, two elements, right? In an array, two positions, right? So, swapping of

Two elements or positions are the same, two elements in an array, all right. So, let us have a generic swapper as the class. In fact, it is a driver class, and I have one swap method, all right. So, you have a generic notation here. We are going to use the array of T, all right. So, T may become an integer, double, or character, right. So, you have an array.

And then you are passing index 1 and index 2. So, which index do you want to swap, right? So, those indices are nothing but which indices have to be swapped. That is the meaning. So, let us consider a temporary variable.



## Example: Generic Method to Swap Elements in an Array

A generic method to swap two elements in an array of any type.

```java
import java.util.Arrays;

public class GenericSwapper {

    // Generic method to swap two elements in an array
    public static <T> void swap(T[] array, int index1, int index2) {
        T temp = array[index1];
        array[index1] = array[index2];
        array[index2] = temp;
    }
// Continued on the next page
```

So, array of index 1, you put temp. And then array of index 2, you put array of index 1. And temp will go to index 2, right? Array of index 2. This is what is swap, the usual swap program, right?

So, assume that all of you know swap. Correct. So, now we will go to the main method, right? So, this is inside the driver class GenericSwapper, and here you go, all right. So, I am creating an integer array, you call it an int array, right?

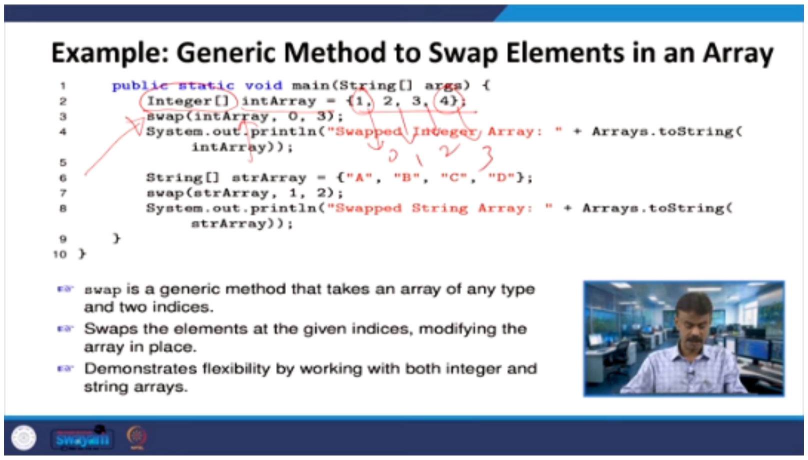
So, an integer array of integers, and int array you have 1, 2, 3, 4, right? Int array you have 1, 2, 3, 4, and swap, you're passing int array 0 and 3, right? The 0th index and 3rd index, right? So, which is the 0th index? This is the 0th index 1, this is 2, and this is 3.

0th index, 1st index, 2nd index, and 3rd index. So, I am passing this. Int array, the complete array will be passed, and the indices are 0 and 3. So, that means 1 and 4 should be swapped. Positions should be swapped.

So this will happen. These all become an integer array. You are passing index 1 as 0, index 2 as 3. And then, right, so here you have array of index 1 is nothing but 0, right, array of 0. Array of 0 is 1.

And array of 3 is 4. So you are swapping these two, right. So you are calling the swap method. Int array is being passed. That means identity hash map of int array is being passed.

And 0th index you are passing. 3rd index you are passing. So, 0 and third index will be swapped. So, once it is swapped, right? Now, you are printing.



## Example: Generic Method to Swap Elements in an Array

```
    public static void main(String[] args) {
        Integer[] intArray = {1, 2, 3, 4};
        swap(intArray, 0, 3);
        System.out.println("Swapped Integer Array: " + Arrays.toString(
            intArray));

        String[] strArray = {"A", "B", "C", "D"};
        swap(strArray, 1, 2);
        System.out.println("Swapped String Array: " + Arrays.toString(
            strArray));
    }
}
```

- swap is a generic method that takes an array of any type and two indices.
- Swaps the elements at the given indices, modifying the array in place.
- Demonstrates flexibility by working with both integer and string arrays.

The arrays is invoking, right? Arrays is invoking. You are already having java.utility.arrays to string, all right? And then you are printing the integer array, right? You are completely printing this integer array, all right?

So, now what will happen? Output is 4, 2, 0. 3 and 1. This is the output we will get. We will see.

Right. So similarly you have a array of strings. Right. Array of strings. So in the previous case what will happen to t?

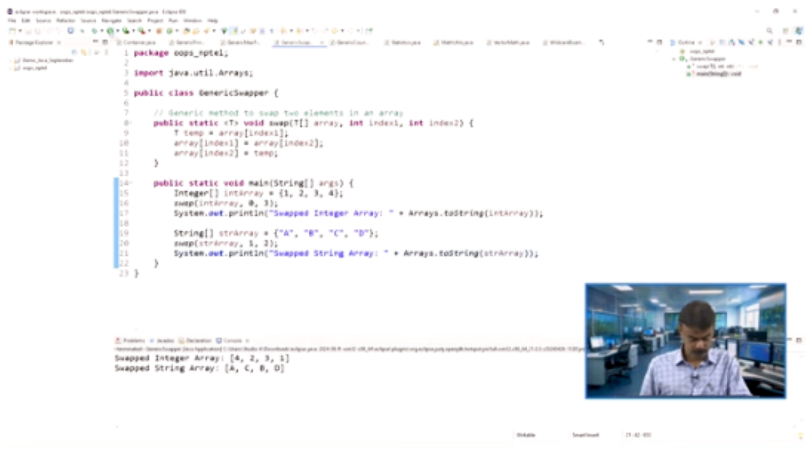
t is integer. Right. t is integer array. And in this case you are using string. Right.

So when I am calling this particular method this t will become string. Right. So you have string array. String array you have a, b, c, d. Right. So that is nothing but 0, 1,

2 and 3. So now the question is when you are passing, I mean to say when you are calling this swap, passing str array, the base address you call in C++, here you call it as identity ashmap in Java. You are passing index 1 and index 2. Index 1 is what b and index 2 is what c. So now all t will become string. right.

All t will become string and then you are swapping. So, you will get. So, 1 and 2 when you swap, you have to get the output a, c, b and d, a, c, b and d, ok. So, these are the two output we will get. So, now what we will do?

We will go to the program and run it, right. So, this is the one I explained. So, when you run the code, So, what we have seen 4, 2, 3, 1 you can see that right the first one is right 4, 2, 3, 1 and the second one will go to the eclipse again A, C, B, D right. So, what did I write A, C, B, D right I hope it is clear and you understood and this is for your reference whatever I have explained.



```
package oops_nptel;

import java.util.Arrays;

public class GenericSwapper {

    // Generic method to swap two elements in an array
    public static <T> void swap(T[] array, int index1, int index2) {
        T temp = array[index1];
        array[index1] = array[index2];
        array[index2] = temp;
    }

    public static void main(String[] args) {
        Integer[] intArray = {1, 2, 3, 4};
        swap(intArray, 0, 3);
        System.out.println("Swapped Integer Array: " + Arrays.toString(intArray));

        String[] strArray = {"A", "B", "C", "D"};
        swap(strArray, 1, 2);
        System.out.println("Swapped String Array: " + Arrays.toString(strArray));
    }
}
```

```
Swapped Integer Array: [4, 2, 3, 1]
Swapped String Array: [A, C, B, D]
```

So, you can go through this. So, now we will see one more generic method right. So, we will see one more generic method. So, that will be counting the occurrences of element in an array, all right. So, here you go count occurrences as a method, all right.

Array of t, right, we know this is a generic and you are having one element, you are passing. For example, I have array of let us say 100 elements and I am passing how many times 2 is occurring or how many times 10 is occurring, that element you are passing, right. So, now count 0. So, you can write whatever way I mean you want this is one of the ways. So, I am using this for loop write t item that means a identity hash map of item and array are same now and then if I am calling one method item dot equals element.

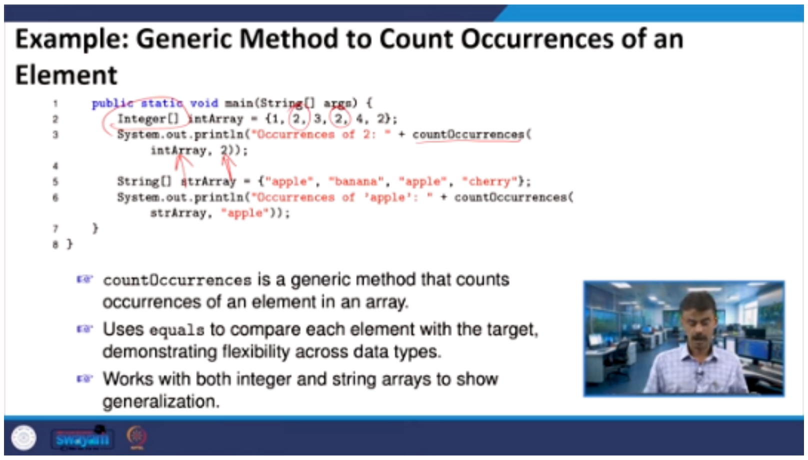
So that means in an array, if I call this function, whatever number, assume that I pass 2. So it is checking. Starting from array 0 to array n minus 1, it is checking whether it is equal to 2. So if it is true, count will be incremented by 1. Initially count is 0.

Count will be incremented by 1. And then what you are doing? Return count. So now we will see what are all the arrays we are going to consider. So we consider 1 as integer array.

Right. So, integer array you have 1, 2, 3, 2, 4, 2, right. So now, what we are doing is we are calling the method count occurrences, right? Count occurrences. Passing int array, the identity hash map, right? It is an integer array, and you are passing 2. The element is 2, right. So now, what will happen? T is becoming integer. Capital T is nothing but integer now.

So now, What we have to do, we are passing element 2, right. So the element is nothing but 2. So now, it is checking how many times. So first time, it is seeing 1 is not equal to 2.

Second time, 2 equal to 2. Then count will become 1, right. So 3 not equal to 2, right. So count will become 2. And when it comes here, count will become 3.



## Example: Generic Method to Count Occurrences of an Element

```
    public static void main(String[] args) {
        Integer[] intArray = {1, 2, 3, 2, 4, 2};
        System.out.println("Occurrences of 2: " + countOccurrences(
            intArray, 2));

        String[] strArray = {"apple", "banana", "apple", "cherry"};
        System.out.println("Occurrences of 'apple': " + countOccurrences(
            strArray, "apple"));
    }
}
```

- countOccurrences is a generic method that counts occurrences of an element in an array.
- Uses equals to compare each element with the target, demonstrating flexibility across data types.
- Works with both integer and string arrays to show generalization.

So here, I have to get the output 3. Now, the next case is becoming a string, right? So, a string array—we have a string array. What are those there? Apple, banana, apple, cherry.

And then you are calling the method 'count occurrences.' So, you are passing the identity hash map of the string array, str array. And now, the element is 'apple.' It is a string array. So, the meaning is the 't' will become a string.

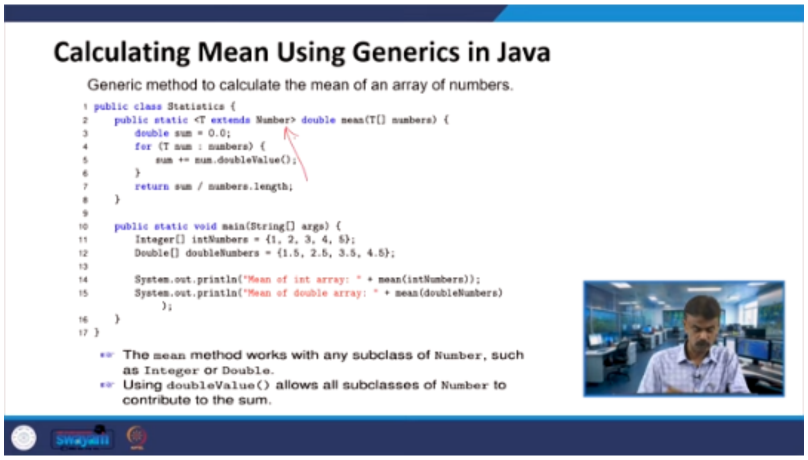
And the element is nothing but 'apple,' right? The element is nothing but 'apple.' So, it will be counting, right? So, initially, the first one is 'apple,' the count will become 1, right? And 'banana,' it is not equal.

The third one is apples 2. So here, I have to get the output 2, right? So when we run the code, here you have 2 appearing 3 times. So that is what I said: 2 is appearing 3 times. And here, occurrences of apple are appearing 2 times, right?

So what did you write? 2 times. So this is how the generic method works. We will see one more program. Calculating mean using generics in Java.

How do you calculate the mean? We know the mean. Suppose an array of elements is given. I mean, add everything and divide by the number of numbers. So now, again, we are using the extends.

And then we are using the superclass number. Superclass or you can say interface. right number so public static t extends number right and you have mean method correct and then return type is double you have array of numbers whose data type is t t is the generic t and what you are doing double sum is equal to 0 and for right t now number and numbers right both are pointing to the same location you can say that both are same identity hash map and sum is equal to sum plus num dot right now it is becoming num suppose I have array 1, 2, 3, 4, 5 right elements right so now num dot double value so num now it is num right num double value first double value 1 so sum is equal to sum plus this so 0 plus 1 now sum is 1 right next one is 2 so like this it goes that is what it is indicating so num is Invoking double value from the interface or superclass number.



## Calculating Mean Using Generics in Java

Generic method to calculate the mean of an array of numbers.

```
1  public class Statistics {
2      public static <T extends Number> double mean(T[] numbers) {
3          double sum = 0.0;
4          for (T num : numbers) {
5              sum += num.doubleValue();
6          }
7          return sum / numbers.length;
8      }
9
10     public static void main(String[] args) {
11         Integer[] intNumbers = {1, 2, 3, 4, 5};
12         Double[] doubleNumbers = {1.5, 2.5, 3.5, 4.5};
13
14         System.out.println("Mean of int array: " + mean(intNumbers));
15         System.out.println("Mean of double array: " + mean(doubleNumbers)
               );
16     }
17 }
```

- The mean method works with any subclass of Number, such as Integer or Double.
- Using doubleValue() allows all subclasses of Number to contribute to the sum.

So line number 7 it is returning sum divided by numbers dot length. Anyway I can use it. Make use of numbers. How many number of elements. So that will give you the length.

So this is nothing but the mean. This is nothing but the mean. So now you have int numbers right now you have int numbers whose array is integer right whose array is integer you have 1 2 3 4 5 right 1 2 3 4 5. So that means t is now becoming integer your t is nothing but the integer right. So numbers correct so t is nothing but the integer and here when you have a double t will become double.

So when it is assumed that you are calling the method. Assume that you are calling the method mean, right? You are passing int numbers. So, int numbers is nothing but integer. T is becoming integer, right?

So, T is becoming integer. And then, right? You are doing a sum. So, it will do sum and then sum divided by 5, right? So, whatever the sum is, it will be divided by 5 and returned here, right?

So, next one is double, right? So, mean is calling double numbers, right? Passing double numbers. You are calling mean by passing double numbers. Double numbers are nothing but 1.5, 2.5, 3.5, and 4.5.

So now your T is nothing but double. So rest is similar. 1.5 we will add it with 2.5 then 3.5 then 4.5 and the length you can see which is nothing but 4 divided by 4, right. So, let us see this code.

So, when you run the code, yes, first you are getting 3 as a average, right, mean, right, you can see that 5 plus 4 plus 3 plus 2 plus 1, right, 15 by 5, 15 by 5 is 3. And similarly, when you add this, you should get 15.0, yeah, 4 elements, 12.0, yes, it will be 12.0, 12.0 when you are dividing by 4 it will be 3.0. So that is how it is calculating the mean. So up to here we have seen right.

So now you are mostly familiar with this generic method and generic class. And in the next class we will see some more program on Java and I will introduce Python. Okay. So in Python also we can talk about generics. We will have a little bit of Python and then we will talk about the generics in Python.

Thank you.