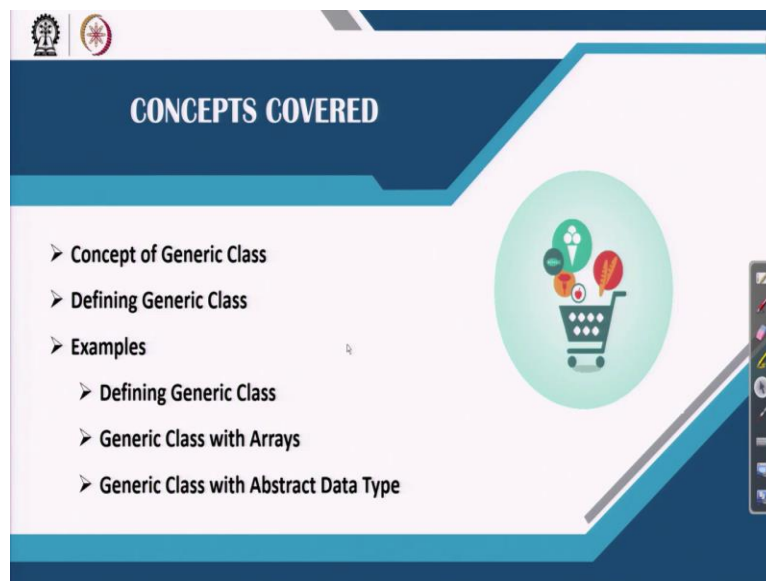


Data Structures and Algorithms Using Java
Professor Debasis Samanta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
Lecture 3
Topic: Basics of Generic Class

In this lecture we shall learn about how to define generic class. In the last video lectures we have covered about how to define generic method. It is little bit one advance step of that. So, inside a class, method will be there which we can make generic but as a whole the entire class can be made generic. And obviously if the class is generic then the inside this method, basically which operates on this data that also sometime needs to be generic or non-generic whatever is there.

Anyway, so if we can declare a class generic this means that, the path data field of this class can be any type. If it is not generic then only the data field can be specific type of data, but generic means it can be any type of data that can be there.

(Refer Slide Time: 1:30)



Now, in this lecture we will try to cover about the concept of generic class first, then we shall learn about how a generic class can be defined. We shall illustrate the concept and then generic class definition with few examples. The first example is very simple that how the generic class can be defined and then a generic class with array of elements, these array of elements can be of any type and then we will discuss about how a generic class also can be whole true for any type

of data other than the primitive data or user defined data. So, that is the topic that we are going to discuss about.

(Refer Slide Time: 2:23)

The slide features a background with a stylized tree of icons representing various fields like engineering, science, and technology. On the left, there is an icon of a steaming coffee cup. The title 'Concept of Generic Class' is written in a bold, blue font. At the bottom, there are logos for NPTEL and IIT Kharagpur.

Concept of Generic Class

NPTEL Online Certification Courses
IIT Kharagpur

The slide has the same background as the previous one. The title 'Why generic class?' is in blue. Below the title, there is a paragraph of text followed by a numbered list of three tasks. The bottom of the slide contains the NPTEL and IIT Kharagpur logos.

Why generic class?

Let us consider the case of processing of an array of any type of numbers using a Java program.

1. Initializing the array.
2. Printing the elements in the array.
3. Reversing the ordering of the elements in the array.

NPTEL Online Certification Courses
IIT Kharagpur

Now, let us first have the idea about concept of a generic class. Now, the concept is there, if we can define a class that is a program and if the class is generic this means that it can holds any type of data in it. Now, first let us explain this concept with an example. Suppose, this is one simple program that we are going to learn about, the program is basically initialize an array, then

print the elements in the array and it will do some task so that it will reverse the ordering of the array.

So, these are the three task that we want to do and we are interested to write a program. Now, this program we want to make a generic in the sense that the array that we will use it can store any type. Then the method which will print, it should print any type of arrays, elements and another method reverse which should work if the array store any type of data, it can be integer, it can be float, or it can be string, or it can be any user-defined data type. Now, let us see without any generic programming, if we want to do it then how we can do that.

(Refer Slide Time: 4:03)

Why generic class?

Here, is the program structure which you should consider, in case the array stores integer numbers.

```
class SpecificArrayInt {  
    // Declaring an array of integer values  
    // Constructor to load the array.  
    // Method to print the array elements.  
    // Method to reverse the array elements.  
}  
  
class MainClassInt {  
    /* This class utilize the class SpecificArrayInt to  
    manipulate some integer data */  
}
```

NPTEL Online Certification Courses
IIT Kharagpur

Now, in this first (ex), this is the basically structure of the program that you should have in order to deal with and in this case we are only considering that the elements should be stored in the array as integer. So, we can write a program, the name of the class, for example, can give at specific array int, then we should have a declaration of array of elements, then constructor to load the array that means initializing the array, method to print the array and another method to reverse the array.

And finally this program can be run using a driver class that is main class where the objects of this type will be declared and used or the all methods those are there in this class will be accessed. So, this is the case.

(Refer Slide Time: 5:00)

Example 3.1 : Program to handle an array of integers

```
class SpecificArrayInt {  
    // Declaring an array of integer numbers  
    int a;  
    // Constructor to load the array  
    SpecificArrayInt(int a[]) {  
        this.a = a;  
    }  
    // Method to print the array elements  
    void printInt() {  
        for(int x : a)  
            System.out.println(x);  
    }  
}
```

// Continued to next page ...

NPTEL Online Certification Course
IIT Kharagpur

Now, here is the program for you, very simple that you can check it. In this program, so this is the class that we are going to define and this basically integer a, that means it will take only integer elements and this is the constructor that basically take it. That means we will pass this constructor as an array of elements and it will initialize to this. So, this basically refer to an array index and then the whatever the array will pass to this that means to this program, it will initialize to this one. So, this constructor is to be defined here.

(Refer Slide Time: 5:53)

Example 3.1 : Program to handle an array of integers

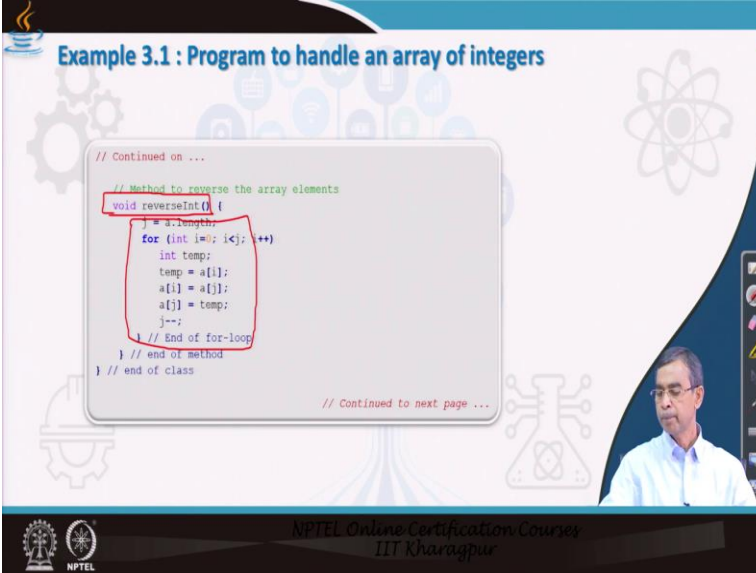
```
class SpecificArrayInt {  
    // Declaring an array of integer numbers  
    int a;  
    // Constructor to load the array  
    SpecificArrayInt(int a[]) {  
        this.a = a;  
    }  
    // Method to print the array elements  
    void printInt() {  
        for(int x : a)  
            System.out.println(x);  
    }  
}
```

// Continued to next page ...

NPTEL Online Certification Course
IIT Kharagpur

And then there is another method, the print method is basically print the whole array. So, here basically declaration of an array of elements and initialization and this is the printing.

(Refer Slide Time: 6:28)



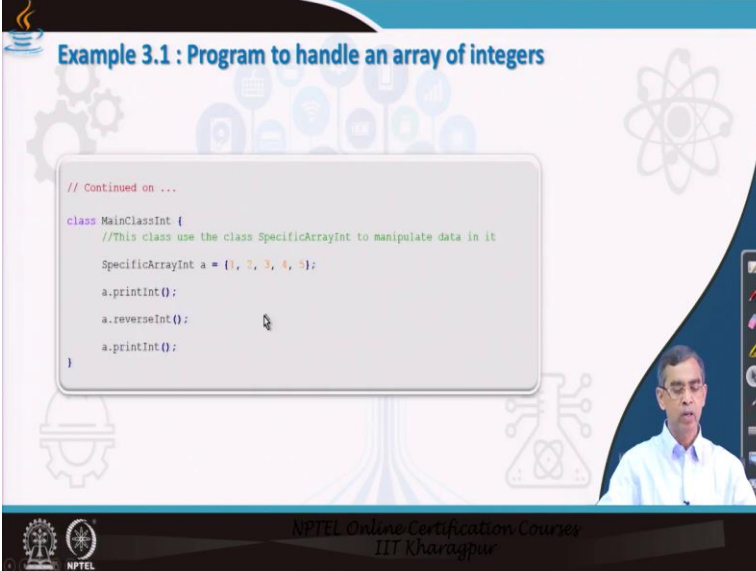
Example 3.1 : Program to handle an array of integers

```
// Continued on ...  
  
// Method to reverse the array elements  
void reverseInt() {  
    j = a.length;  
    for (int i=0; i<j; i++)  
        int temp;  
        temp = a[i];  
        a[i] = a[j];  
        a[j] = temp;  
        j--;  
    } // End of for-loop  
} // end of method  
} // end of class  
  
// Continued to next page ...
```

Then our next task is to reverse the ordering of element, this method is shown here in the slide. Yeah, so this is the method that we can see, the method name is reverseInt and this is the code of the method, a simple logic by which the ordering of the elements can be made in the reverse order and the elements will be stored in the same array and this basically gives the method of reversing.

So, three methods, one is the method of initializing arrays we have discussed, then printing all the elements in an array will be, how it will be it is discussed and finally the reversing of the array elements. So, this completes the declaration of a class, in this case this class will work specific to an integer array.

(Refer Slide Time: 7:26)



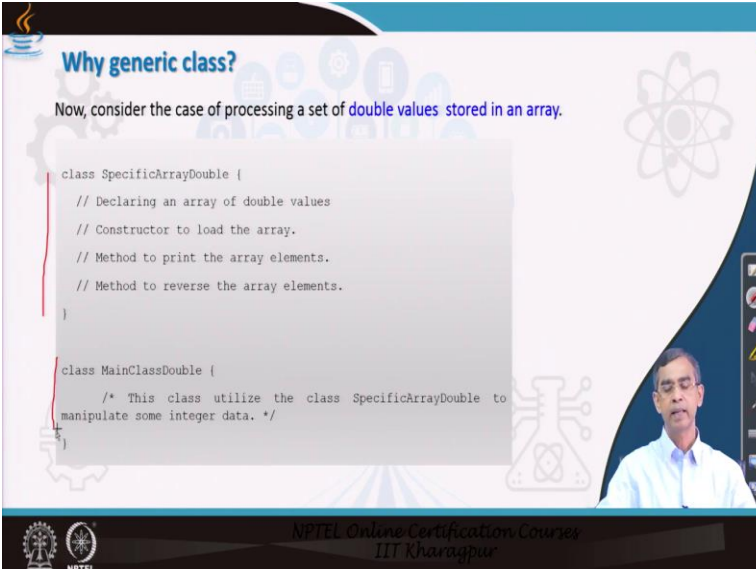
Example 3.1 : Program to handle an array of integers

```
// Continued on ...  
  
class MainClassInt {  
    //This class use the class SpecificArrayInt to manipulate data in it  
  
    SpecificArrayInt a = {1, 7, 3, 4, 5};  
  
    a.printInt();  
  
    a.reverseInt();  
  
    a.printInt();  
}
```

NPTEL Online Certification Courses
IIT Kharagpur

And finally the main method which will look like this just here as we see, we declared an array a, then we call this objects a for this print method to print the elements of the array a, and then this is the reversing the elements and then again after reversing, printing the array elements. So, here we just create an object as we see, the name of the objects that we have created a is basically is an specific array and then is an integer array basically, now this is for integer.

(Refer Slide Time: 8:21)



Why generic class?

Now, consider the case of processing a set of double values stored in an array.

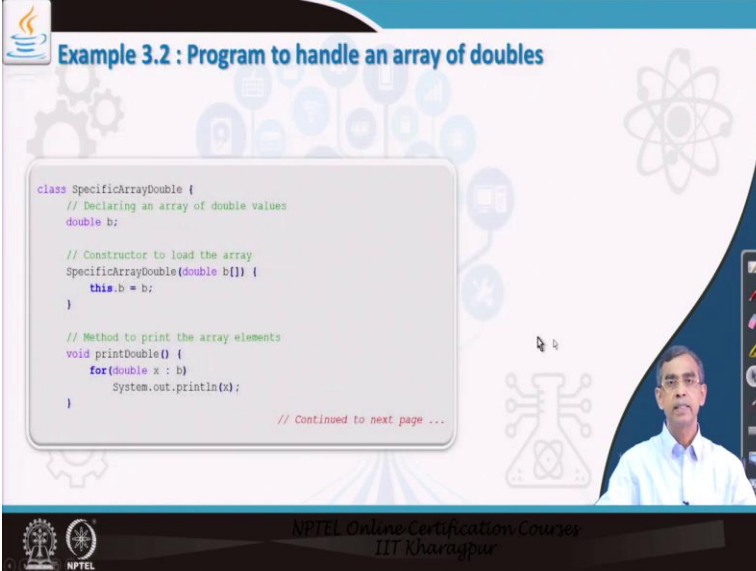
```
class SpecificArrayDouble {  
    // Declaring an array of double values  
    // Constructor to load the array.  
    // Method to print the array elements.  
    // Method to reverse the array elements.  
}
```

```
class MainClassDouble {  
    /* This class utilize the class SpecificArrayDouble to  
    manipulate some integer data. */  
}
```

NPTEL Online Certification Courses
IIT Kharagpur

If we want to do the same job but for another type of data, for example, say it is a floating point values then we can repeat the same program but for only floating point values. As we see here the same program structure declaring an array of float values, initializing the arrays, then method to print the array of elements and finally reversing and we can call this method in a main class.

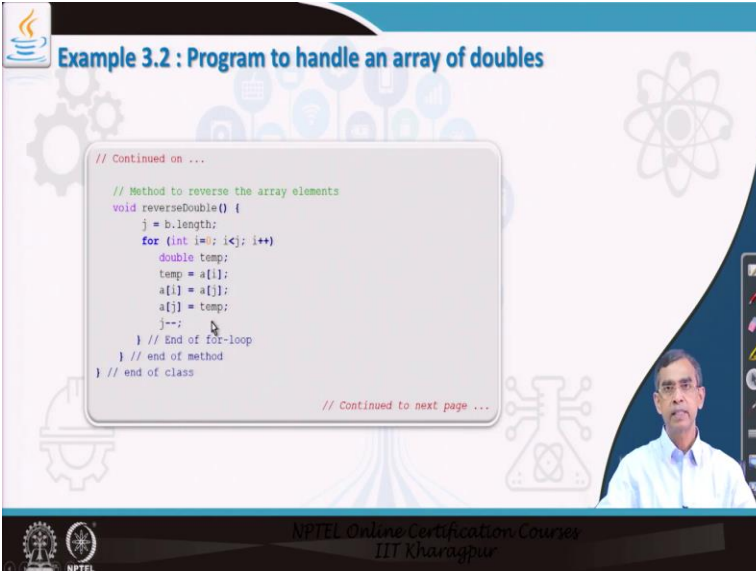
(Refer Slide Time: 8:49)



Example 3.2 : Program to handle an array of doubles

```
class SpecificArrayDouble {  
    // Declaring an array of double values  
    double b;  
  
    // Constructor to load the array  
    SpecificArrayDouble(double b[]) {  
        this.b = b;  
    }  
  
    // Method to print the array elements  
    void printDouble() {  
        for(double x : b)  
            System.out.println(x);  
    }  
    // Continued to next page ...  
}
```

NPTEL Online Certification Courses
IIT Kharagpur



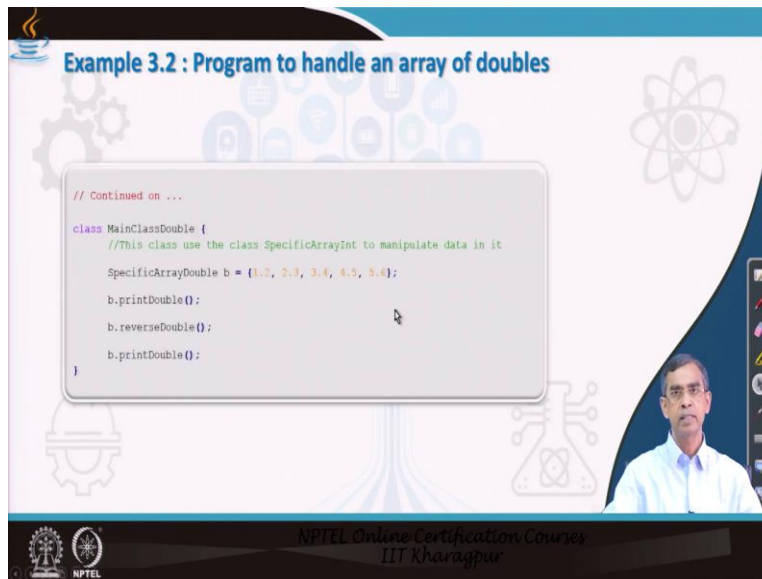
Example 3.2 : Program to handle an array of doubles

```
// Continued on ...  
  
// Method to reverse the array elements  
void reverseDouble() {  
    j = b.length;  
    for (int i=0; i<j; i++)  
        double temp;  
        temp = a[i];  
        a[i] = a[j];  
        a[j] = temp;  
        j--;  
    } // End of for-loop  
} // end of method  
} // end of class  
// Continued to next page ...
```

NPTEL Online Certification Courses
IIT Kharagpur

The code is very similar to the previous one and this is the code and as you see this code is for declaring the arrays, array of element initialization, print method and then this is the reversing, the logic is same program is same only data type has changed.

(Refer Slide Time: 9:09)



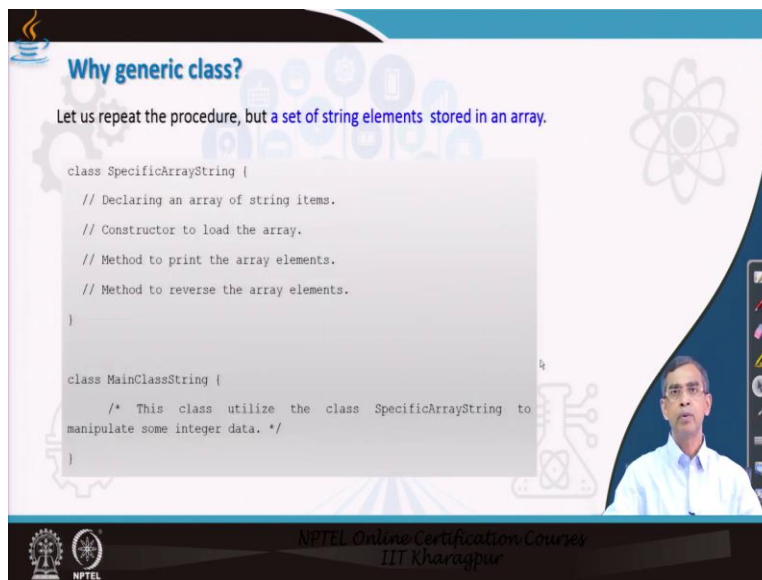
Example 3.2 : Program to handle an array of doubles

```
// Continued on ...  
  
class MainClassDouble {  
    //This class use the class SpecificArrayint to manipulate data in it  
  
    SpecificArrayDouble b = {1.2, 2.3, 3.4, 4.5, 5.6};  
  
    b.printDouble();  
  
    b.reverseDouble();  
  
    b.printDouble();  
  
}
```

NPTEL Online Certification Courses
IIT Kharagpur

And we can call this array of floating point values and then run it, this is the program but for floating point values.

(Refer Slide Time: 9:24)



Why generic class?

Let us repeat the procedure, but a set of string elements stored in an array.

```
class SpecificArrayString {  
    // Declaring an array of string items.  
    // Constructor to load the array.  
    // Method to print the array elements.  
    // Method to reverse the array elements.  
  
}  
  
class MainClassString {  
    /* This class utilize the class SpecificArrayString to  
    manipulate some integer data. */  
  
}
```

NPTEL Online Certification Courses
IIT Kharagpur

Example 3.3 : Program to handle an array of Strings


```

class SpecificArrayString {
    // Declaring an array of double values
    String c;

    // Constructor to load the array
    SpecificArrayString(String c[]) {
        this.c = c;
    }

    // Method to print the array elements
    void printString() {
        for(String x : c)
            System.out.println(x);
    }
}
//Continued to next page ...

```




Example 3.3 : Program to handle an array of Strings

```

// Continued on ...
class MainClassString {
    //This class use the class SpecificArrayString to manipulate data in it
    SpecificArrayString c = {"A", "B", "C", "D", "E"};
    c.printString();
    c.reverseString();
    c.printString();
}

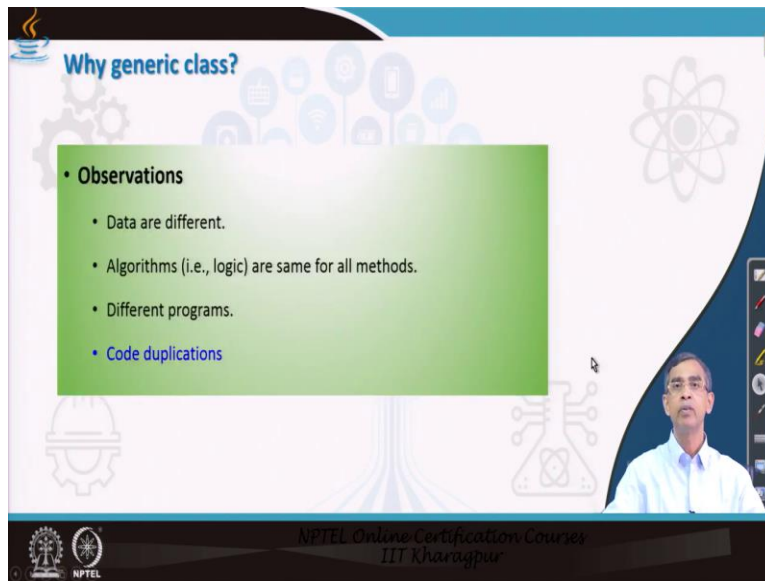
```



Again if we want to repeat it for the arrays of string then same, it is the same structure of codes, same reverse method and then similar kind of what is called the execution of the method using driver class here. Now, so three different executions we have discussed for which we ought to write three programs specific array int, specific array double, and specific array string, this should be written as string, by mistake it is written as double.

So, three methods we have discussed and we call it. Now, we could do the same thing but writing instead of three different versions of the program for three different types but using generic class declaration. So, how this can be done?

(Refer Slide Time: 10:34)



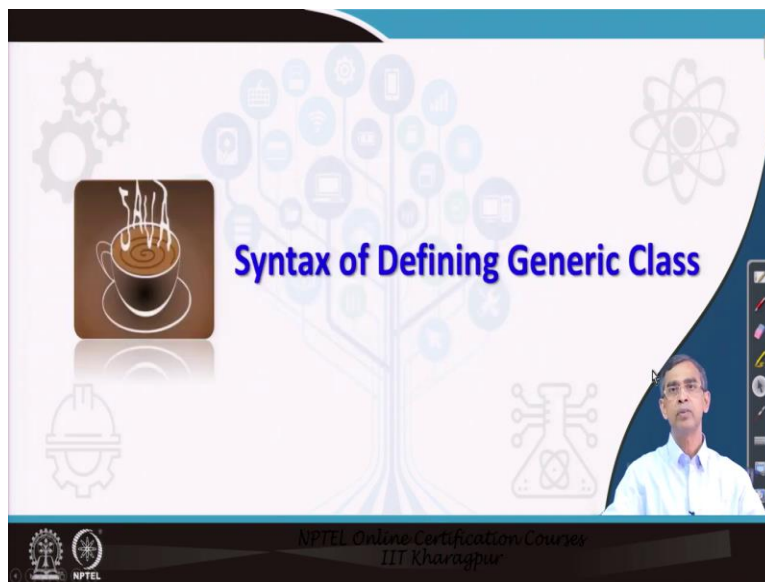
The slide is titled "Why generic class?" and features a green box with the following observations:

- Observations
 - Data are different.
 - Algorithms (i.e., logic) are same for all methods.
 - Different programs.
 - Code duplications

The slide also includes the NPTEL logo and the text "NPTEL Online Certification Course IIT Kharagpur" at the bottom.

Here is an example that how it can be. If the types are different but logics are same, then we should not go for writing different programs rather we can avoid code duplications.

(Refer Slide Time: 10:51)



The slide is titled "Syntax of Defining Generic Class" and features a coffee cup icon with steam rising from it. The slide also includes the NPTEL logo and the text "NPTEL Online Certification Course IIT Kharagpur" at the bottom.

Now, here is the idea about that how a class, that means a program can be made generic so that it can take care about any type of data to run.

(Refer Slide Time: 11:09)

The slide is titled "Syntax for generic class definition". It contains the following text and code:

The syntax for declaring a generic class is as follows:

```
[<<Access>] class <ClassName><<Type1> [, <Type2>, ...]> {  
    ... body of the class  
}
```

Here, is the full syntax for declaring a reference to a generic class and instance creation:

```
<className><typeList> varName = new <className><typeList> (<InputArray>);
```

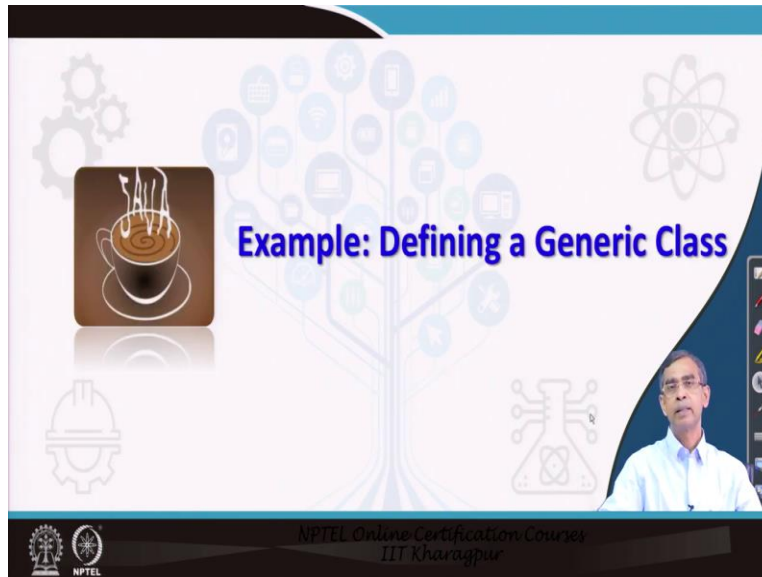
The slide also features a small video inset of a man in a white shirt and glasses, and a footer with the NPTEL logo and text: "NPTEL Online Certification Course IIT Kharagpur".

The syntax for these things, here is very similar to the concept of defining generic method, so what we can do is that a class can be declared a generic by specifying a template of it. Now, in order to declare a template, so here the template is basically to be enclosed as we have done the same thing for method.

But in this case after the name of class within these angular brackets starting and closing and whatever the different types that you want to handle in this program you can mention as a template. That means without mentioning the actual data type, only the template type like T1, T2, T3 if you want to process three different types of data in your program they are three different templates T1, T2, T3 like.

And then we can pass after the template discussion, for example, these are the list of templates may be 1 or any numbers, so template list accordingly and if you want to pass any arrays of this template type, you can again within this first bracket and within this triangular brackets the name of the array that you can declare, is a template array. Now, let us have a quick demonstration of this program to understand how a class can be defined in a generic way.

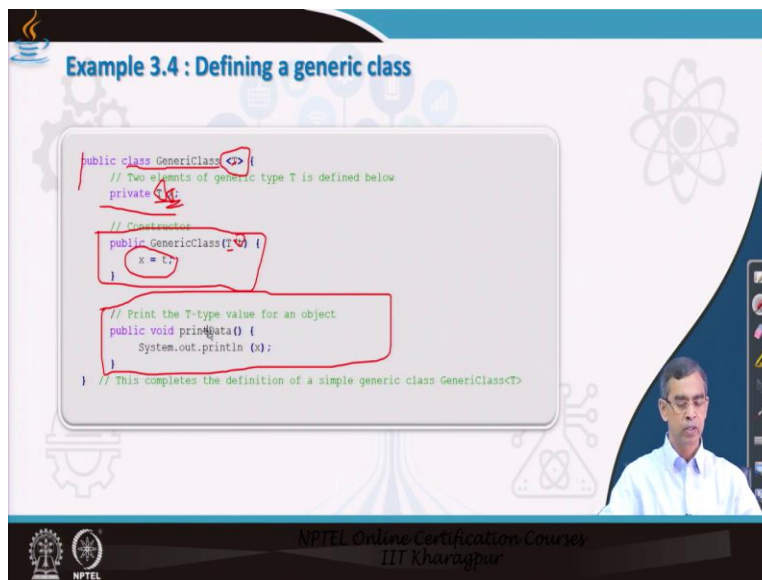
(Refer Slide Time: 13:03)



The slide features a central title "Example: Defining a Generic Class" in blue text. To the left is an icon of a coffee cup with steam. The background is decorated with various icons like gears, a tree with nodes, and a molecular structure. A presenter is visible in the bottom right corner. The footer includes the NPTEL logo and the text "NPTEL Online Certification Course IIT Kharagpur".

Now, we are going to define or going to illustrate one example how a generic class can be defined with the syntax that we have discussed about.

(Refer Slide Time: 13:12)



The slide is titled "Example 3.4 : Defining a generic class". It displays a code editor window with the following Java code:

```
public class GenericClass <T> {  
    // Two elements of generic type T is defined below  
    private T x;  
  
    // Constructor  
    public GenericClass(T t) {  
        x = t;  
    }  
  
    // Print the T-type value for an object  
    public void printData() {  
        System.out.println(x);  
    }  
} // This completes the definition of a simple generic class GenericClass<T>
```

Red annotations highlight the class name `GenericClass`, the generic type parameter `<T>`, the `private T x;` declaration, the constructor `public GenericClass(T t) { x = t; }`, and the `printData()` method.

The presenter is visible in the bottom right corner. The footer includes the NPTEL logo and the text "NPTEL Online Certification Course IIT Kharagpur".

Now, here just closely look at this program, this is the one class we are going to discuss, the name of the class is generic and within this angular bracket we want to mention that this class is a template, is a generic class and we mention this genericity with only one template that means this program will take care on template type of data, generic type of data. Now, so this class has

one members, the members is basically type of T, this x can be a single data type or it can be array name.

So, it is basically T x, that means x is a type of data of type capital T. Then we should declare one constructor, it is very simple the constructor will basically initialize the field x, so if we pass any elements of type T then it will initialize the value of x, so this is the way that the generic class can be declared and construct it, in it can be declared.

Now, after these things any number of methods in it can be declared, either they are of generic type or simple non generic also in this case we just give an example of a simple method to print all the elements, which are stored here in x using this method. So, simple print method will print the elements which are stored there in x. So, this is the simple way that a generic class is declared in this example. So, this is the way that a simple generic class that you can declare.

(Refer Slide Time: 15:22)

The slide is titled "Example 3.4: Using the defined GenericClass<T>". It contains the text: "The driver class is programmed below, which creates different types of objects." Below this is a code block for a Java class named GenericClassTest. The code is as follows:

```
class GenericClassTest {
    public static void main( String args[] ) {
        // A data with the member as String
        GenericClass<String> a = new GenericClass<String> ("Java");
        a.printData();
        // A data with the member as Integer value
        GenericClass<Integer> b = new GenericClass<Integer> (10);
        b.printData();
        // A data with the member as float value
        GenericClass<Double> c = new GenericClass<Double> (3.14);
        c.printData();
    }
}
```

The code is annotated with red circles and lines. Red circles highlight the generic type parameters: <String> in the first line, <Integer> in the second line, and <Double> in the third line. Red lines connect these circles to the corresponding generic type parameter in the class definition: <T> in the class name, <String> in the first parameterized constructor call, <Integer> in the second parameterized constructor call, and <Double> in the third parameterized constructor call. The slide also features a small video inset of a man in the bottom right corner and logos for NPTEL and IIT Kharagpur at the bottom.

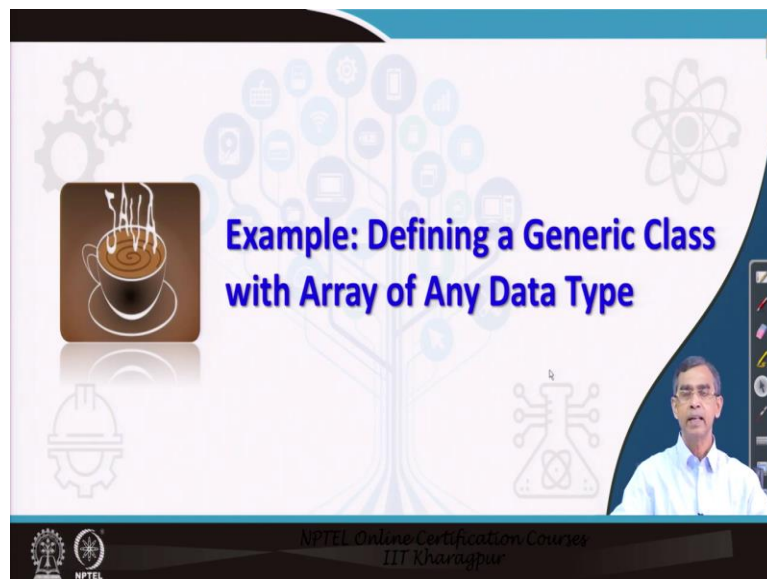
And now let us see how we can utilize it. And this is the one main method which basically utilize the previous declaration. Now, in this method as we see we create an object a, here we mention the template T as a string, that means this is the generic class that we are going to have is for the type string and a is the name of the objects of type generic.

And when we create this object we initialize the field x in it passing a value here as a Java. So, this means a string and a is basically whole the value Java at the moment. Now, in the next one,

in this example, it basically does the same thing but for numeric values. In the third one, again we repeat it, we again create an object of the generic class, but for floating point values. Now, so what we can see is that, we declare one class as a generic and whenever we create the math objects of that class and any operation in it then can be called for that type of.

For example, print data here, this basically print data for integer type, here print data for string and here print data for double values. So, this is, this way the generic execution can be done depending on the type of values that you want to process. So, this is the way that generic class and its utilization can be done. Now, so this example gives you a simple idea about how a simple generic class can be declared.

(Refer Slide Time: 17:44)



Now, let us extend this idea little bit in a more deeper sense in the concept that how we can define a generic class with array that array can store any data type. In the previous examples, we have discussed about a generic class which contains a data of any type, but it is not array of any type data. Now, in the next example we are going to discuss, repeat the similar procedure but only for arrays. Now, let us note what are the difference, so for array and then only a single elements.

(Refer Slide Time: 18:28)

Example 3.5: Processing arrays with any data type

```
class GenericArrayClass {
    // Declaring a generic array
    // Constructor to load the array.
    // Method to print the array elements
    // Method to reverse the array elements
}

class MainClass {
    //This class utilize the class GenericArrayClass to manipulate data of any type.
}
```

NPTEL Online Certification Course
IIT Kharagpur

Now, this is the structure of the program that are going to illustrate so that we can understand how a generic class for array with any type of data can be declared. So, few steps like declaring a generic array, then constructor to load the array, then method to print the array and method to reverse the array so that any type of elements integer, double or string or any type can be used and all this operation can be done. You can recall the class that we have discussed for integer array, string and the strings separately we deal it but we can do it in a generic way.

(Refer Slide Time: 19:13)

Example 3.5: Defining class to process arrays with any data type

```
class GenericArray<T> {
    // Declaring an array, which should store any type T of data
    T a[];
    GenericArray(T x) { // Define a constructor
        a = x;
    }
    T getData(int i) { // To return the element stored in the i-th place in the array
        return a[i];
    }
    void static printData (T b) { // A generic method to print the elements in array b
        for(int i = 0; i < b.length(); i++)
            System.out.print(b.toString(i) + " "); // Print the i-th element in b
        System.out.println(); // Print a new line
    }
}
```

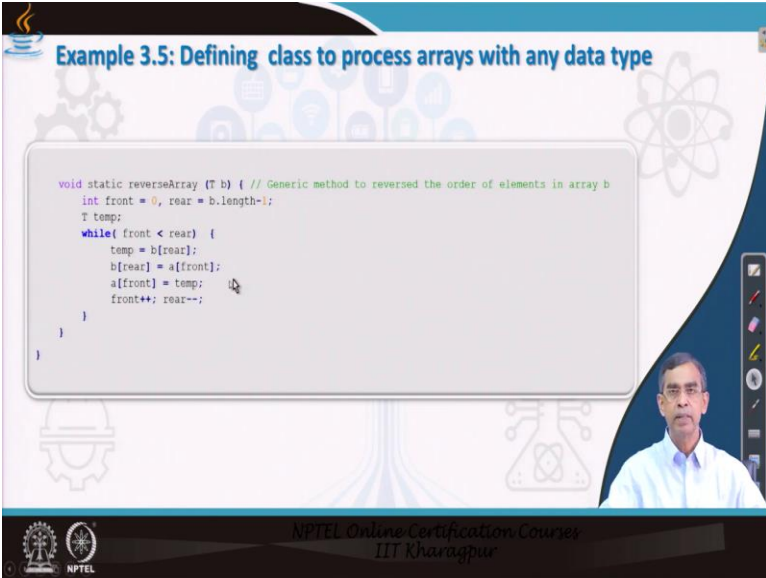
NPTEL Online Certification Course
IIT Kharagpur

Now, so here is the code, you should note it, the code is very simple again, similar to the previous only minor difference. This is same as the previous one, we declare within this syntax that it is a generic class, this is the declaration that we have added other than the existing we as we know how to declare a class but only the template within angular brackets that is all.

Now, here only you see in the previous example we declared simpler array but here we declare an array as an, a as an array of elements. Then this is the constructor, it is same as earlier one and then this is the one method we just want to have it, so that it can access any elements into this array using this method.

So, getData integer i that means it will return the ith element which is stored there in the array. And then the print data is basically print T b, b if the array reference then for the entire array starting from 0 to the length of the array, it will print each elements. So, these are very simple method that we have discussed here in this class. One is the name, the value that it can store array and then finally the print method and how to access each elements into the array.

(Refer Slide Time: 20:47)



The slide displays a code editor window with the following Java code:

```
void static reverseArray (T b) { // Generic method to reversed the order of elements in array b
int front = 0, rear = b.length-1;
T temp;
while( front < rear) {
temp = b[rear];
b[rear] = a[front];
a[front] = temp;
front++; rear--;
}
}
```

The slide also features a video feed of a presenter in the bottom right corner and logos for NPTEL and IIT Kharagpur at the bottom.

Now, let us see the main class that we can discuss, so this is another method in the same array to reverse the element. So, this completes the discussion of three methods, reading initializing an array, printing elements and reversing this one and we can use this method for the three different type of arrays passing into it.

(Refer Slide Time: 21:10)

Example 3.5: Defining class to process arrays with any data type

```
class GenericClassArrayDemo {
    public void static main(String args a[]) {
        //Creating an array of integer data
        Integer x[] = {10, 20, 30, 40, 50}; // It is an array of Integer numbers

        // Store the data into generic array
        GenericArray<Integer> aInt = new GenericArray<Integer>(x);
        // Alternatively:
        // GenericArray<Integer> aInt = new GenericArray<Integer>(new Integer x[] {10, 20, 30, 40, 50});

        // Printing the data ...
        printData(aInt); // Printing the array of integer objects

        //Reverse ordering of data ...
        reverseArray(aInt);

        // Printing the data after reverse ordering ...
        printData(aInt); // Printing the array of integer objects after reversing

        // Continued to next page ...
    }
}
```

NPTEL Online Certification Course
IIT Kharagpur

So, here first we declare x is an array of integer and you note down with did not declare it as a int x rather integer x, that mean x is an array of integers of course, but here integers are treated as an object. So, in order to declare that object of integer array, we declare this declaration. So, that is the thing that you should do in case of generic class, you always, you have to declare an array of objects instead of array of primitive type.

(Refer Slide Time: 21:49)

Example 3.5: Defining class to process arrays with any data type

```
class GenericClassArrayDemo {
    public void static main(String args a[]) {
        //Creating an array of integer data
        Integer x[] = {10, 20, 30, 40, 50}; // It is an array of Integer numbers

        // Store the data into generic array
        GenericArray<Integer> aInt = new GenericArray<Integer>(x);
        // Alternatively:
        // GenericArray<Integer> aInt = new GenericArray<Integer>(new Integer x[] {10, 20, 30, 40, 50});

        // Printing the data ...
        printData(aInt); // Printing the array of integer objects

        //Reverse ordering of data ...
        reverseArray(aInt);

        // Printing the data after reverse ordering ...
        printData(aInt); // Printing the array of integer objects after reversing

        // Continued to next page ...
    }
}
```

NPTEL Online Certification Course
IIT Kharagpur

And again the next example as we see, the same class method is basically instantiated for the template here is integer, this is the name of the object that we have created passing x, x is basically is a array object to it, that mean aInt is basically is a collection we can say which store all the elements which is there in x.

Alternatively this is another syntax also you can do at the same time, here you can see the integer array has been passed using this concept, here actually using these array instantiation created by new and this array. So, in this case this is not required, so if we do not want to have it rather in the same object's declaration you can do it or else this one it can be done.

(Refer Slide Time: 22:51)

Example 3.5: Defining class to process arrays with any data type

```
class GenericClassArrayDemo {
    public void static main(String args a[]) {
        //Creating an array of integer data
        Integer x[] = {10, 20, 30, 40, 50}; // It is an array of Integer numbers

        // Store the data into generic array
        GenericArray<Integer> aInt = new GenericArray<Integer>(x);
        // Alternatively:
        // GenericArray<Integer> aInt = new GenericArray<Integer>(new Integer x[] {10, 20, 30, 40, 50});

        // Printing the data ...
        printData(aInt); // Printing the array of integer objects

        //Reverse ordering of data ...
        reverseArray(aInt);

        // Printing the data after reverse ordering ...
        printData(aInt); // Printing the array of integer objects after reversing

        // Continued to next page ...
    }
}
```

NPTEL Online Certification Courses
IIT Kharagpur

Now, so next is basically once this collection, aInt is a collection of array of type integer is created, we can pass this and then printData method can be call and it basically print all the elements in it. reverseArray another method it is basically there, we can pass this collection that means array of elements, it can reverse ordering and finally again if we print the same it will print the data here in the same order, here in the reverse order. So, this is, this works fine for an integer array.

(Refer Slide Time: 23:46)

Example 3.5: Defining class to process arrays with any data type

```
// Continued on ...  
  
//Creating an array of integer data  
String[] l = {"A", "B", "C", "D", "E"}; // It is an array of String objects  
  
// Store the data into a generic array  
GenericArray<String> bString = new GenericArray<String>(l);  
  
// Printing the data ...  
printData(bString); // Printing the array of string objects  
  
//Reverse ordering of data ...  
reverseArray(bString);  
  
// Printing the data after reverse ordering ...  
printData(bString); // Printing the array of string objects after reversing  
  
// Continued to next page ...
```

NPTEL Online Certification Course
IIT Kharagpur

Now, we can repeat the same program but for other type of array. For example, let us see here see, how the same program but it is for string, so this basically again run the program but for other collection other type of array called the string, here string array is declared here. So, these are the string array, we pass this string array here and then print reverse and again after reversing printing this one. And you can see the same way we have thus, we did not write any separate code for string, same code which works for integer array also now is working for the character array.

(Refer Slide Time: 24:43)

Example 3.5: Defining class to process arrays with any data type

```
// Continued on ...  
  
//Creating an array of double data  
Double z[] = {1.2, 2.3, 3.4, 4.5, 5.6}; // It is an array of double values  
  
// Store the data into a generic array  
GenericArray<Double> cDouble = new GenericArray<Double>(z);  
  
// Printing the data ...  
printData(cDouble); // Printing the array of double values  
  
//Reverse ordering of data ...  
reverseArray(cDouble);  
  
// Printing the data after reverse ordering ...  
printData(cDouble); // Printing the array of double values after reversing  
} // End of main method  
} // End of GenericArrayClassDemo class
```

NPTEL Online Certification Course
IIT Kharagpur

So, if it works for integer it will work also for double values and this is an example where it also same program we can run but for double arrays, arrays of double values. And only thing that you should note is that why we are creating or passing the values into this program, we always create object arrays.

(Refer Slide Time: 25:17)

Example 3.5: Defining class to process arrays with any data type

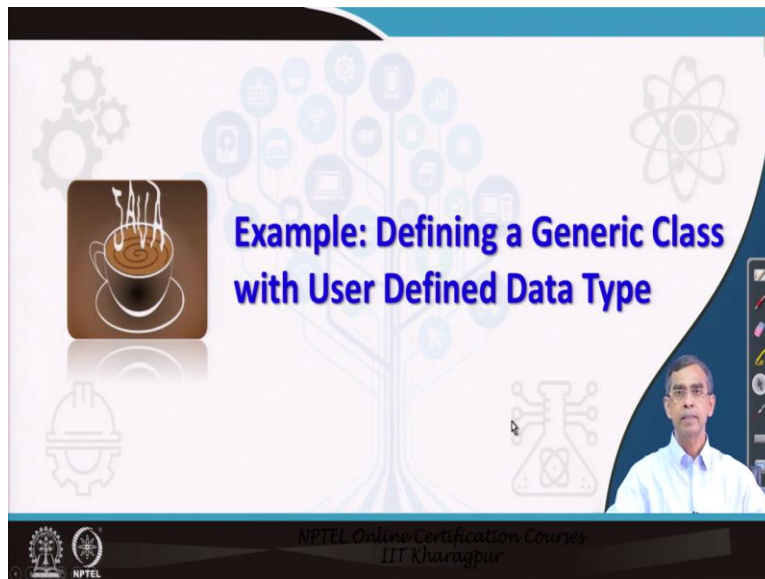
```
// Continued on ...  
  
//Creating an array of double data  
Double z[] = {1.2, 2.3, 3.4, 4.5, 5.6}; // It is an array of double values  
  
// Store the data into a generic array  
GenericArray<Double> cDouble = new GenericArray<Double>(z);  
  
// Printing the data ...  
printData(cDouble); // Printing the array of double values  
  
//Reverse ordering of data ...  
reverseArray(cDouble);  
  
// Printing the data after reverse ordering ...  
printData(cDouble); // Printing the array of double values after reversing  
} // End of main method  
} // End of GenericArrayClassDemo class
```

NPTEL Online Certification Course
IIT Kharagpur

Like in the previous last two examples the object array that we have used integer INTEGER in capital, capital I, integer, here double, similarly string always it should be. If you declare instead

of this one say simple double, DOUBLE then it will not work you. So, basically you have to pass an object because in case of generic class whatever the template you define, this template holds for only objects not for any general values.

(Refer Slide Time: 26:04)



So, we have learnt about how the generic class can be declared, we have also learnt about how the generic class with array of any type also can be controlled. Now, next example that we should learn about whether the same declaration is valid for any other data type other than conventional numbers data type or string data type.

For example, say user can define their own data type, say book or a student or person, now can we repeat the same thing, that mean objects can be created, objects, arrays can be passed, they can be printed and ordering of the objects can be made like this. The answer is, of course, yes, because of this is the main strength of the generic programming which we can do it.

(Refer Slide Time: 27:02)

Example 3.6: Working with user defined class

```
//Define a class Student
class Student {
    String name;
    float marks;

    Student(String name, float marks) {
        this.name = name;
        this.marks = marks;
    }
}

class GenericClass<T> {
    T obj;
    GenericClass(T obj) {
        this.obj = obj;
    }
    public T getObject() {
        return this.obj;
    }
}
```

NPTEL Online Certification Course
IIT Kharagpur

Now, here is an example for you to illustrate the concept. In this example you can see, first we have to declare your own type of data as we have discussed here, student is a one type of data that we have discussed, it has only name and marks floating point and this basically is a constructor by which an object of type student can be initialized.

So, this is a user defined data type student and this is basically the similar concept for declaring a class as a generic class, generic class. T is the template that means it can hold any type integer, double, string as in earlier case. At the same time we want to show so that this also worked for this type of data that means this template T can also replace or holds student and also.

Now, this is basically a object is declared year and is a generic class for a constructor of this generic class and this is the one method that which basically to get object means it will get an object from the collection and then return this object and print it. So, this is the idea about this method declaration, a generic class declaration for any type.

(Refer Slide Time: 28:35)

Example 3.6: Working with user defined class

```
class GenericClassTest { // Driver class to test generic class facility
public static void main (String[] args) {
GenericClass <Integer> iObj = new GenericClass <Integer> (1);
// A class with Integer type
System.out.println(iObj.getObject());

GenericClass <String> sObj = new GenericClass <String> ("Java");
// Another class with String type
System.out.println(sObj.getObject());

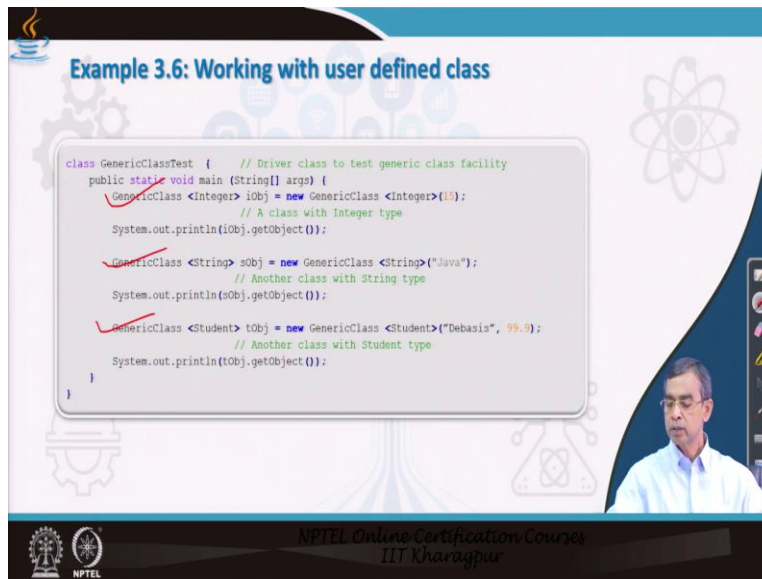
GenericClass <Student> tObj = new GenericClass <Student> ("Debasij", 99);
// Another class with Student type
System.out.println(tObj.getObject());
}
}
```

NPTEL Online Certification Course
IIT Kharagpur

And then we can declare a main method to use this definition and here is an example. So, this basically is a main class, driver class, main method. Now, here we use this class that we have discussed earlier, it is an, for an integer object, integer value that is passed to this is basically this value. So, this i object is basically is a generic objects, which basically store the value in this instance integer and then we just call this as a print method. So, get object is basically return and then it print, it is there.

Now, again it also work for the string type of data, this is initialized value, the float value and its print, so here. So, this program works for integer, this for string, this for print and in place of it if we declare here, for example, generic class student as you see, student is a basically type we have discussed already, this is an object of that, we created a generic and then here we initialize passing the value. Because it needs string name and then marks as a floating point.

(Refer Slide Time: 30:12)



Example 3.6: Working with user defined class

```
class GenericClassTest { // Driver class to test generic class facility
public static void main (String[] args) {
    GenericClass <Integer> iObj = new GenericClass <Integer>(15);
    // A class with Integer type
    System.out.println(iObj.getObject ());

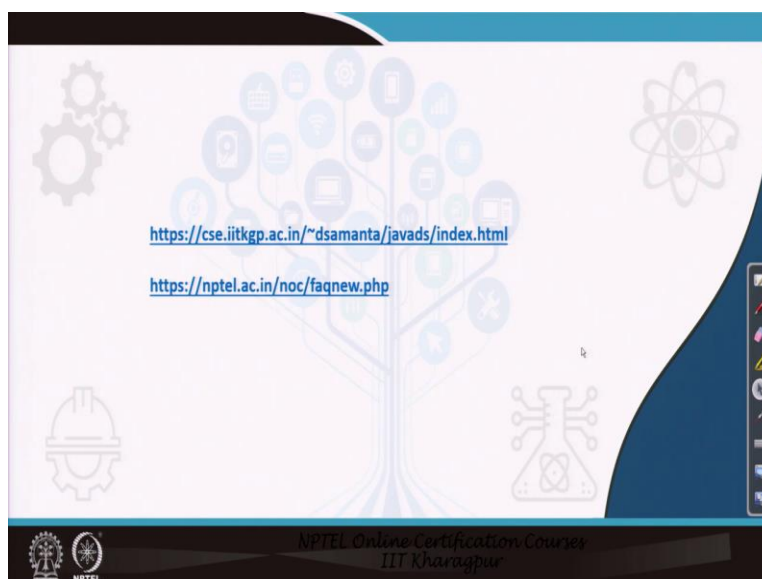
    GenericClass <String> sObj = new GenericClass <String>("Java");
    // Another class with String type
    System.out.println(sObj.getObject ());

    GenericClass <Student> tObj = new GenericClass <Student>("Debasis", 99.0);
    // Another class with Student type
    System.out.println(tObj.getObject ());
}
}
```

NPTEL Online Certification Course
IIT Kharagpur

So, this is the example which basically explain that it works for integer object, it works for string object, it for user defined data type and then process it. So, this way generic class, in fact, works for you for any type of data that you can pass to a program or your program can be instead of a special for only a specific type of data it can be made a general that is generic for, in order to work for any type of data.

(Refer Slide Time: 31:02)



<https://cse.iitkgp.ac.in/~dsamanta/javads/index.html>

<https://nptel.ac.in/noc/faqnew.php>

NPTEL Online Certification Course
IIT Kharagpur

So, this is the way the generic class can be defined. And we have learnt about how a class can be declared generic and it can be utilized. There are many more things also we have to learn and for the entire discussion that we have made in this lecture, you can have from the link that we have given and there are few more topics, which are treated as advanced topics so for generic class handling is concerned, we will discuss in our next video lectures, thank you.