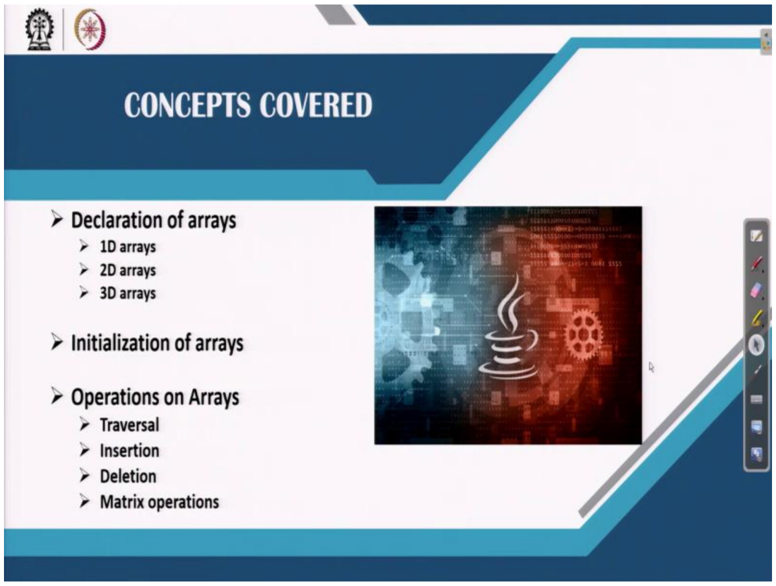
Data Structures and Algorithms Using Java
Professor. Debasis Samanta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
Lecture 12

Programming for Arrays

We are discussing module 4 of this course, module 4 is about arrays. We have learned basic concept of array in the last lecture, today we shall learn about programming with arrays.

(Refer Slide Time: 00:49)



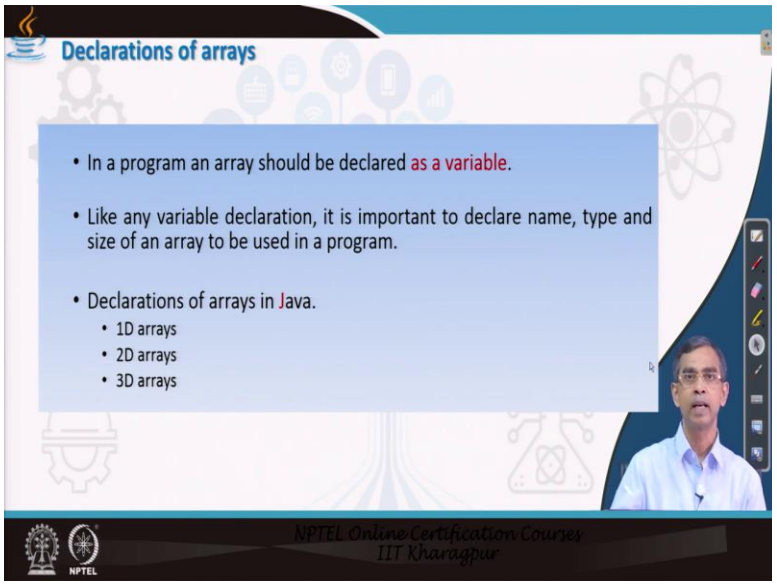
CONCEPTS COVERED

- Declaration of arrays
  - 1D arrays
  - 2D arrays
  - 3D arrays
- Initialization of arrays
- Operations on Arrays
  - Traversal
  - Insertion
  - Deletion
  - Matrix operations

So, there is a Java programming features, because array is a simple yet very effective one data structure, no any programming language can think about without any array in it. So in Java also, no exception array has given due importance and we shall learn about how an array can be declared and the array can be created.

It is basically initialized with elements, there are different form of arrays that we have discussed, we shall try to understand how the different types of array can be handled in Java programs. And there are basic few operations are very important we shall quickly go through a programming aspects regarding the operations on arrays.

(Refer Slide Time: 02:08)



Declarations of arrays

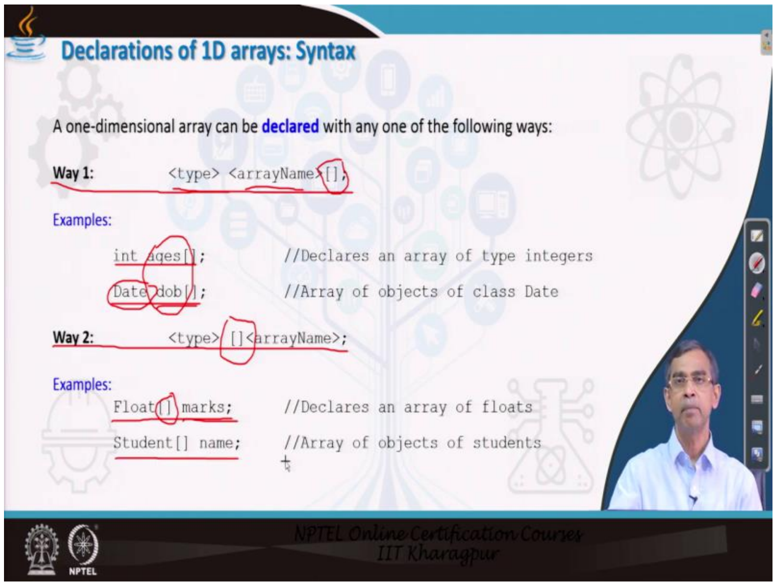
- In a program an array should be declared as a variable.
- Like any variable declaration, it is important to declare name, type and size of an array to be used in a program.
- Declarations of arrays in Java.
  - 1D arrays
  - 2D arrays
  - 3D arrays

NPTEL Online Certification Courses
IIT Kharagpur
NPTEL

Now, let us first start with the Declaration of Arrays. Now, in programming language like Java or C, C++ array is considered like other variable is a variable. So, like any other variable and array can be declared also, this means that like a variable declaration it has its own name, the name should be unique and type. Now, giving the name of a variable, whatever the rules are applicable to Java programming is also applicable to array.

Now, there are obviously different syntaxes are to be followed in order to declare different type of arrays like, one dimensional array, two dimensional array, three dimensional array., in fact Java supports you any dimensional of array can be declared. But we shall not discuss higher dimensional array declaration, we shall limit our discussion only up to 3D arrays and operations on the arrays we shall discuss up to 2D arrays.

(Refer Slide Time: 03:06)



Declarations of 1D arrays: Syntax

A one-dimensional array can be **declared** with any one of the following ways:

Way 1: `<type> <arrayName>[];`

Examples:

```
int ages[];        //Declares an array of type integers
Date dob[];        //Array of objects of class Date
```

Way 2: `<type> []<arrayName>;`

Examples:

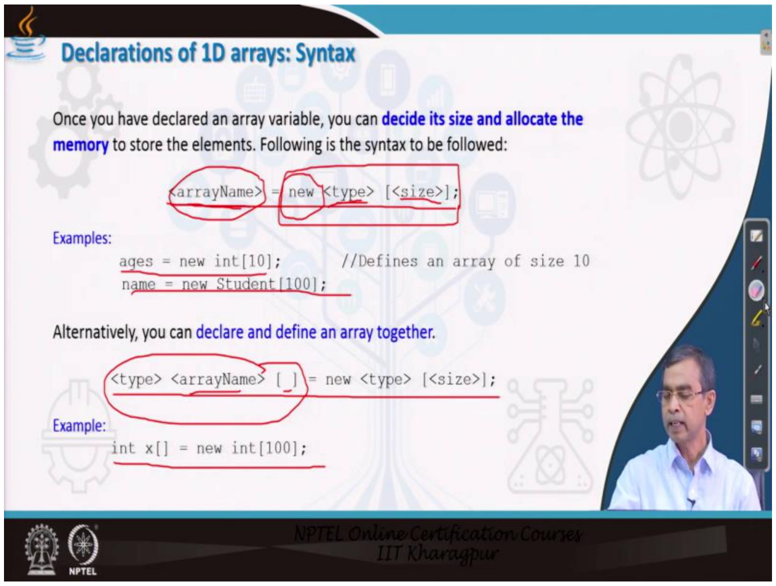
```
Float[] marks;      //Declares an array of floats
Student[] name;     //Array of objects of students
```

Now let us consider first, Declaration of 1D array. There are many ways, in fact by which array can be declared. The first way that we have mentioned here, this is basically the syntax by which array can be declare, it is same as the variable declaration, except this is the one symbols that we have to use. So, this is basically starting and closing square brackets to indicate that these variable name is basically name of an array.

As an example, this is an example to declare an array, the name of the array is ages and type is int. Like you can define an array of any type, for example, if data is a user defined type and then giving this syntax, so dob is an array of type date. Now, here you can see one thing, here we declare the name of the array but we did not mention what about the size. Now, there are certain procedure to be followed that how the size of an array can be specified at the time of declaration.

Now, let us again continue the discussion of declaring an array, so this is the one way that we have discussed there is a another way also the array can be declared, it is basically same thing only the square bracket can be mentioned here. As an example, as we see here float is a float type and this is basically indicates that what we want to declare is a type is basically array and the name of the array is marks, likewise this also declared the name is an array of type student. So, there are different way that we can consider in order to declare an array.

(Refer Slide Time: 05:15)



Declarations of 1D arrays: Syntax

Once you have declared an array variable, you can **decide its size and allocate the memory** to store the elements. Following is the syntax to be followed:

```
<arrayName> = new <type> [<size>];
```

Examples:

```
ages = new int[10];        //Defines an array of size 10
name = new Student[100];
```

Alternatively, you can declare and define an array together.

```
<type> <arrayName> [ ] = new <type> [<size>];
```

Example:

```
int x[] = new int[100];
```

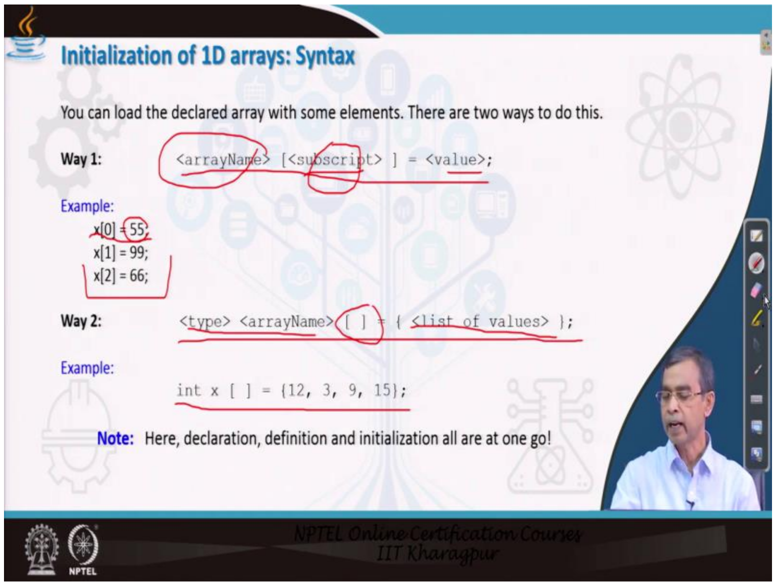
Now, after the declaration it is it is necessary to declare the size of an array. So, this is the one way by which we can declare the size of the array, the syntax that it is required is like this and here you see name of the array which we have already specified in the previous declaration. And then this types should be same as by which this type, otherwise it will report an error. And this basically indicates, what is the number of elements that you want to store in your array, the maximum capacity.

And here, this is very important new is an operator in Java programming it basically invokes Java runtime memory management manager to allocate the memory, the type of the memory is this and this is the size. So basically, the Java runtime manager will consider what amount of array that is required to store this amount of elements of type this one. And then it will return a pointer, pointer means the starting point of the memory allocation, which will be stored here in the name of array, so this is the basic concept that it follows.

And here is an example as we see these basically, we declare ages is an array of integer of size 10, name is an array of type student of size 100, and so on. There is a another way also we can declare and define the name of the array as well as the memory for the array at the same declaration, this is most, most of the program are basically follow this syntax because it is the shortest way that you can declare both array as well as size.

So, here you see same as this one type, these are name of the array, and this is the array, and then new is the operator that we have discussed this one. So, as an example, as you can see x is an array of integer of size 100, so this is the shortest way of declaring both the name of the array as well as type and its size. So, this is the way that you can declare the array. There are few more ways also, I just discussed it in the slides, so this is the one way that we can discuss. So, these are the different way the array can be declared.

(Refer Slide Time: 08:05)



Initialization of 1D arrays: Syntax

You can load the declared array with some elements. There are two ways to do this.

Way 1: `<arrayName> [<subscript> ] = <value>;`

Example:

```
x[0] = 55;
x[1] = 99;
x[2] = 66;
```

Way 2: `<type> <arrayName>[ ] = { <list of values> };`

Example:

```
int x [ ] = {12, 3, 9, 15};
```

Note: Here, declaration, definition and initialization all are at one go!

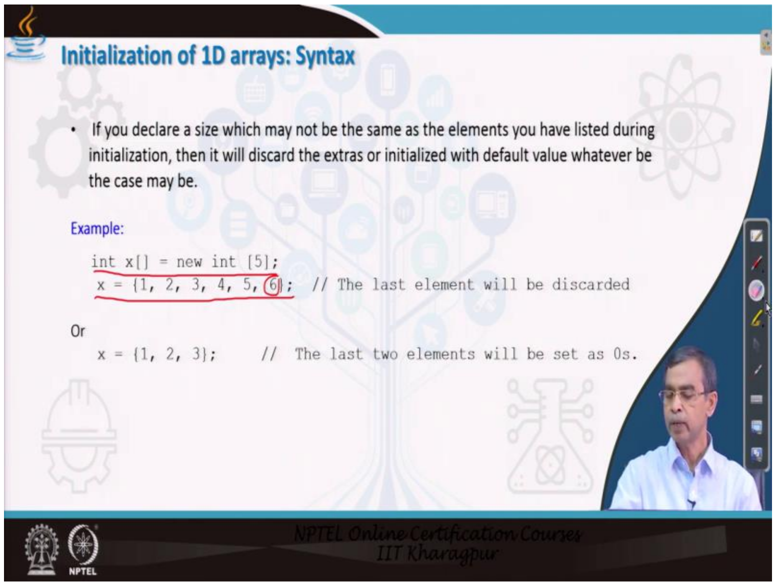
Now, let us see how an array can be initialized. So, now it is, once the array is declared its size is fixed, then we are to load the array with the elements in it. So, there are many way again the initialization can be done, the simple most initialization which is basically fixing the array elements and then the value edit. Now here, if we say the name of the array and then within the square bracket and integer value, it should be always integer, it is called the subscript of the array it is or it is called the index of the array, this subscripts should very 0 to the maximum size minus 1.

Now, this is the value the element that needs to be stored. As an example as you see, x 0 that means at the zeroth position of the array, that mean index is 0 the element 55 needs to be stored and so on. So, this is the basically way by which specifying a particular index and a particular value can be initialized or can be loaded into an array.

Now, there is another way also by which we can also load the array, this is more convenient way many programmer like it. What is the idea, idea is that within the second brackets whatever the values that you want to load into the array you can specify it by closing semicolon and this basically declared the array, array name of the array and size need not to be mentioned here, but or by the virtue of giving the total number of elements, the lower (())(09:33) manager will fix, what amount of size the array has to be.

As an example, here we can see this example says x is an array of type integer and the size of the array at the time of this declaration as well as initialization together is sizes is 4. So, this way the x is an array of size 4 and it elements in it are 12, 3, 9, and 15. So, this is one what is the most easiest and shortest ways or direct ways of declaring as well as initializing and array. So, these are the different way the 1D array can be initialized.

(Refer Slide Time: 10:25)



Initialization of 1D arrays: Syntax

- If you declare a size which may not be the same as the elements you have listed during initialization, then it will discard the extras or initialized with default value whatever be the case may be.

Example:

```
int x[] = new int [5];
x = {1, 2, 3, 4, 5, 6};  // The last element will be discarded
```

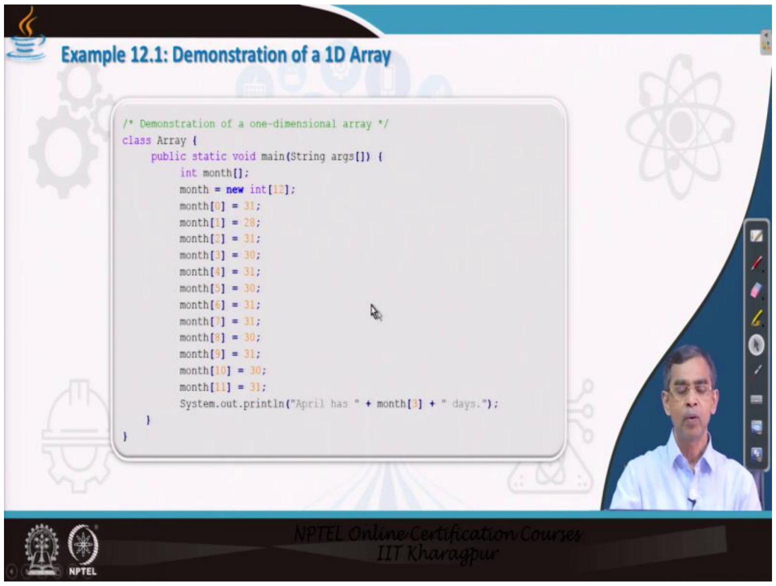
Or

```
x = {1, 2, 3};     //  The last two elements will be set as 0s.
```

Now let us, there are few points also you should consider in this regard. You can do it, here for example, we declared the array x of type integer of size 5 and later on we can initialize it also this way, this is also possible in Java, but here you see the size of the array at the time of declaration is 5 and we want to initialize the array with 6 elements, so it will not be able to load the last element.

So, in that case this element will be truncated automatically, so you will lose the data then. On the other hand, if the size of the array is much more than the number of elements by which it needs to be loaded, then it will load all the elements which are there and the remaining element will be initialized as 0. In this case, for example, last two elements will be 0. So, these are the different way we have learned about how an array can be declared, how 1D array can be initialized, and how 1D array can be loaded in a different fashion.

(Refer Slide Time: 11:30)

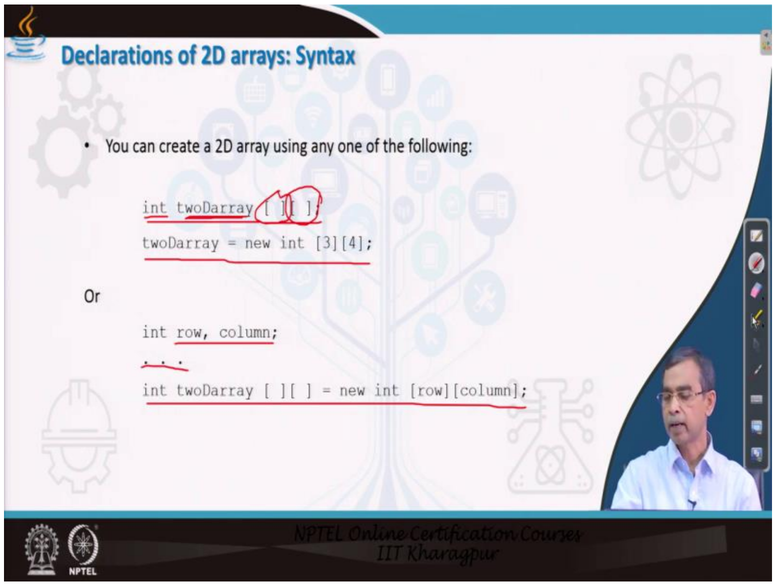


Example 12.1: Demonstration of a 1D Array

```
/* Demonstration of a one-dimensional array */
class Array {
    public static void main(String args[]) {
        int month[];
        month = new int[12];
        month[0] = 31;
        month[1] = 28;
        month[2] = 31;
        month[3] = 30;
        month[4] = 31;
        month[5] = 30;
        month[6] = 31;
        month[7] = 31;
        month[8] = 30;
        month[9] = 31;
        month[10] = 30;
        month[11] = 31;
        System.out.println("April has " + month[3] + " days.");
    }
}
```

Now, this program is a very simple most program you can try and run you will see this declare an array of integer name of the array is month, size is 12. And we one by one, we initialize the array elements in it and finally we can print any elements or all elements. So, all these programs that will be discussed here in this lecture, I should suggest you to just run the program in your own local machine and see the output and try to understand different statement in the program.

(Refer Slide Time: 12:17)



Declarations of 2D arrays: Syntax

- You can create a 2D array using any one of the following:

```
int twoDarray [ ][ ];
twoDarray = new int [3][4];
```

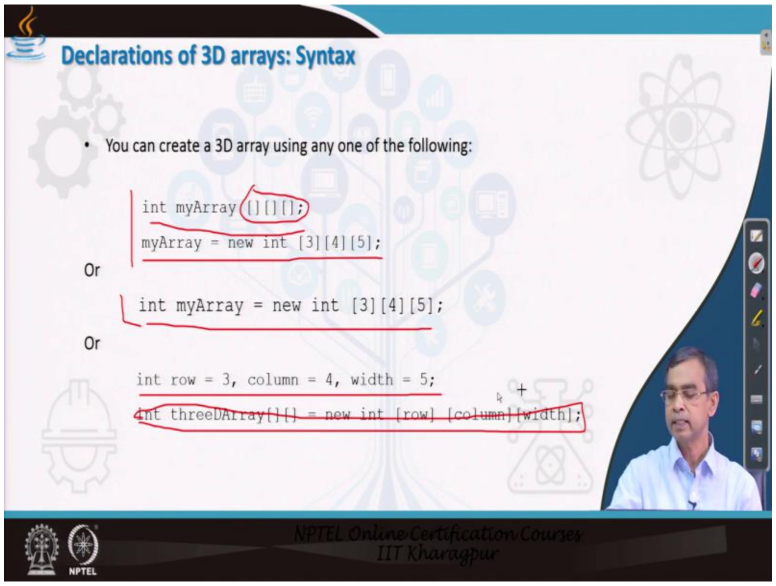
Or

```
int row, column;
. . .
int twoDarray [ ][ ] = new int [row][column];
```

Now, let us consider the Declaration of 2D arrays. Now, 2D arrays declaration more or less similar to 1D array, only the difference is that in case of 1D array 1 closing square and this kind of syntax needs to be considered, but in case of 2D array you have to consider 3. Now, here we declare 2D array, this is the name of the array of type int and then it is basically array, 2D array because two array symbols are mention here. And here also we can specify what should be the size, that mean in this 2D array the size is 3 4, mean 3 rows and 4 columns, it is always 3 rows and 4 columns, not that 4 rows 3 column.

Now, so we can say that in this array we will be able to store all together 3 into 4 equal to 12 number of elements. Alternatively, you can do this kind of declaration if you want to initialize and declare the array which size which can be known only at the time of running the program, but before that you have to ensure that the values of row and column is known to you. So, here is a statement by which each row and column values should be initialize coming to this one, otherwise, there will be a runtime error. So, that precaution should be, so at that checking should be always there. Now, so this is a 2D array declaration and initialization likewise in the 1D array.

(Refer Slide Time: 13:45)



Declarations of 3D arrays: Syntax

- You can create a 3D array using any one of the following:

```
int myArray[][][];
myArray = new int [3][4][5];
```

Or

```
int myArray = new int [3][4][5];
```

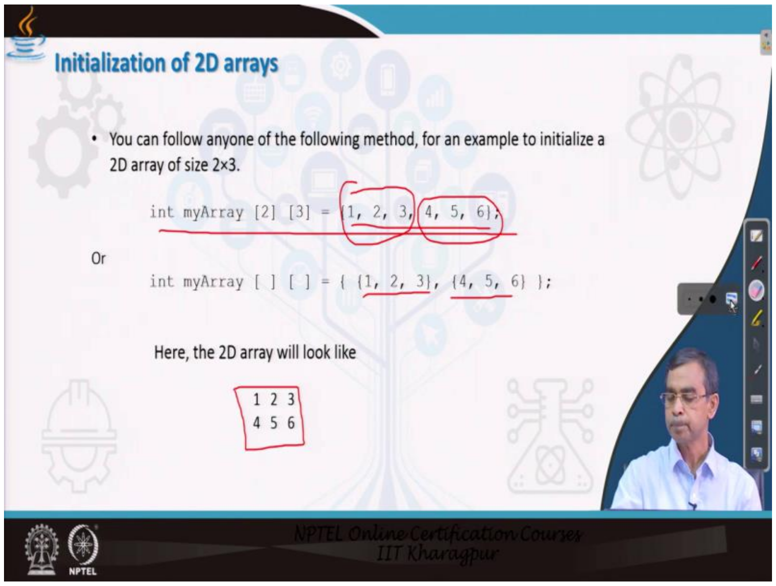
Or

```
int row = 3, column = 4, width = 5;
int threeDArray[][][] = new int [row] [column][width];
```

NPTEL Online Certification Courses
IIT Kharagpur

Here is a some example that we can discuss. This is about 3D array because is a 2D array, 1D array, 3D array, and 4 dimensional array or n dimensional array it will basically increase, that is all. Now here in this case, is a 3D array of size 3 rows 4 columns and 5 is the depth, this is also one way we can declare this is the two statement is required, in one statement we can declare it. Alternatively, if you know the row, column and with values well in advance then at any time in your program, you can specify this also, this way to indicate that it is a 3D arrays.

(Refer Slide Time: 14:39)



Initialization of 2D arrays

- You can follow anyone of the following method, for an example to initialize a 2D array of size 2×3.

```
int myArray [2] [3] = {1, 2, 3, 4, 5, 6};
```

Or

```
int myArray [ ] [ ] = { {1, 2, 3}, {4, 5, 6} };
```

Here, the 2D array will look like

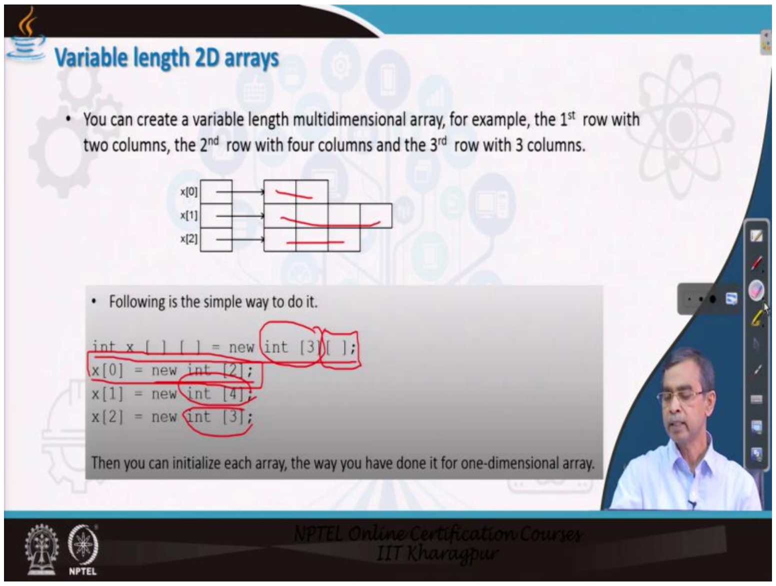
```
1 2 3
4 5 6
```

NPTEL Online Certification Courses
IIT Kharagpur

Now, so we have learned about different type of array that we can declare. And initialize on of 2D arrays, it is more or less similar as we can see this one the number elements can be put it within the single within the curly brackets. And here as 2 and row, so this means that a 2 rows in the first row this element will go, in the second row this element will go.

And otherwise you can do need to be in a structured way for your better understanding, all the rows element can be within curly brackets and this way also, both are the same what is called the to serve the similar purpose. So, this way is a 2 dimensional array can be stored if we follow this kind of declaration in your program.

(Refer Slide Time: 15:27)



Variable length 2D arrays

- You can create a variable length multidimensional array, for example, the 1st row with two columns, the 2nd row with four columns and the 3rd row with 3 columns.

- Following is the simple way to do it.

```
int x [ ] [ ] = new int [3][ ];
x[0] = new int [2];
x[1] = new int [4];
x[2] = new int [3];
```

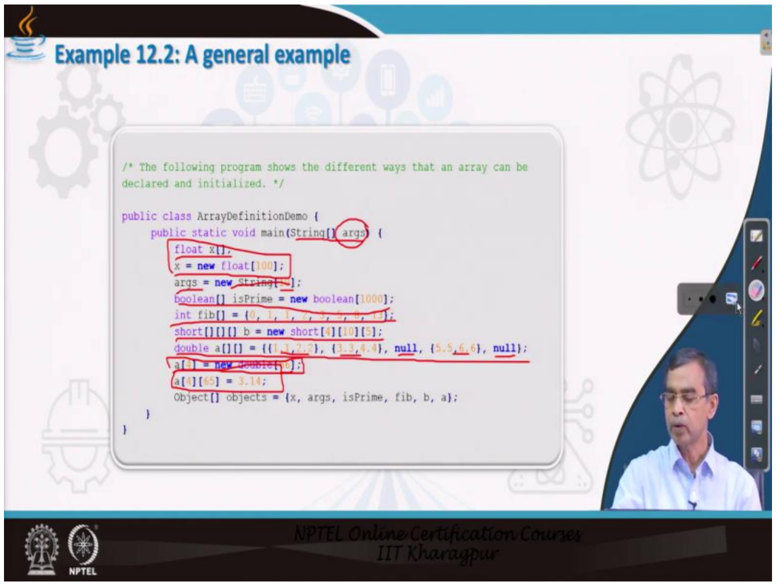
Then you can initialize each array, the way you have done it for one-dimensional array.

Now, there are also possible that we can declare the number of what is called the columns in each row not of same size. In the previous example as we see, the number of rows number of columns in each row is same, but here this is an example I am going to give you that you can declare the variable length 2 dimensional array. The idea it is like this, say suppose this is the first row it contains only 2 column, second row contains 4 columns, and third row contains 3 or size of the row in each rows in (())(16:09)varying.

Now how we can declare it? It is very simple way to declare, first we have to declare because it is essentially a 2 dimensional array, whose row is fixed this is the number of row, so we can declare that, it is an integer array we want to store integer and number of rows will be 3. And however, we did not mention the size of column here, if we mentioned the size of column it indicates that all rows are of same size.

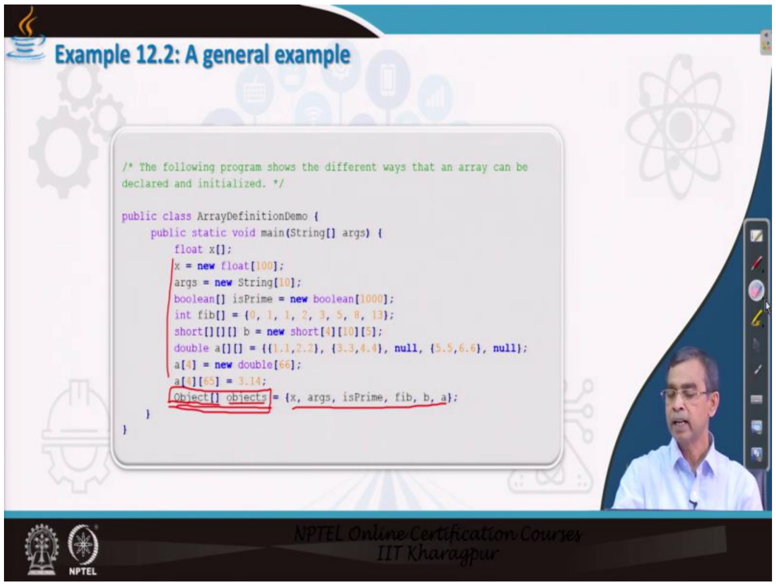
But here if we keep it blank later on, we can fix like x 0 new int 2 this indicates that in the first row we want to store 2 integer element. So, it is like 2 size. Similarly, in the second row we want to store 4 element it is like 4, in the third row we want to store 3 elements, so this one, so this way the variable number of elements in each row can be store. And this is a one way of storing the variable length array in Java program.

(Refer Slide Time: 17:20)



Example 12.2: A general example

```
/* The following program shows the different ways that an array can be
declared and initialized. */

public class ArrayDefinitionDemo {
     public static void main(String[] args) {
          float x[];
          x = new float[100];
          args = new String[10];
          boolean[] isPrime = new boolean[1000];
          int fib[] = {0, 1, 1, 2, 3, 5, 8, 13};
          short[][][] b = new short[4][10][5];
          double a[][] = {{1.1,2.2}, {3.3,4.4}, null, {5.5,6.6}, null};
          a[4] = new double[66];
          a[4][65] = 3.14;
          Object[] objects = {x, args, isPrime, fib, b, a};
     }
}
```



Example 12.2: A general example

```
/* The following program shows the different ways that an array can be
declared and initialized. */

public class ArrayDefinitionDemo {
     public static void main(String[] args) {
          float x[];
          x = new float[100];
          args = new String[10];
          boolean[] isPrime = new boolean[1000];
          int fib[] = {0, 1, 1, 2, 3, 5, 8, 13};
          short[][][] b = new short[4][10][5];
          double a[][] = {{1.1,2.2}, {3.3,4.4}, null, {5.5,6.6}, null};
          a[4] = new double[66];
          a[4][65] = 3.14;
          Object[] objects = {x, args, isPrime, fib, b, a};
     }
}
```

Now, let us consider one simple example, but yet very effective. And in this example, as you can see the different way the array can be declare, this is a very simple way that we declaring x is a float type, and this is basically size is fixed, and memory allocation is complete here. Now, here args with (())(17:37) args is not declare earlier as an array or whatever it is there, but args is declare while we write the main, that means array in this declaration we declare that args is an array of string. But here we are fixing what will be the number of element that we want to store in this array, so 10.

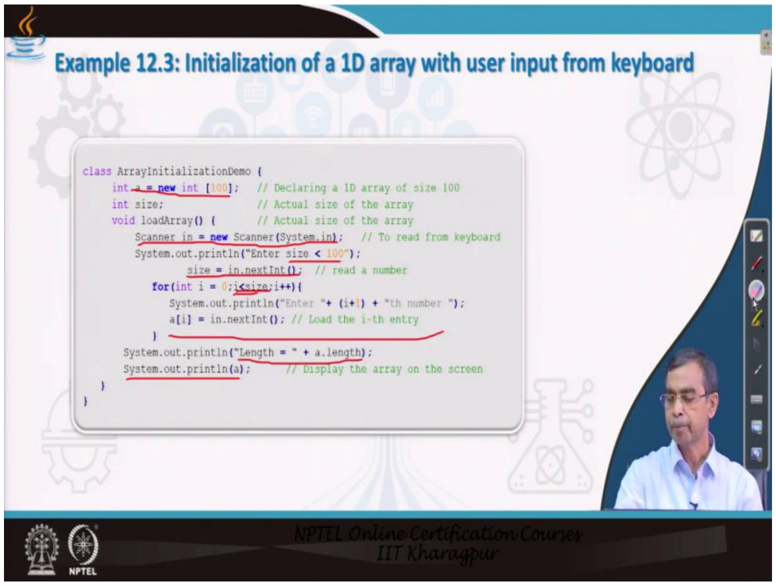
Now, this is quite simple as earlier, so Boolean is an array of type Boolean that means only true and false can be stored, size is 1000. And here also we can see, we can declare an integer array whose size is automatically fixed as we are initializing same thing together. Now here, this is another example, there also we are declaring 3 dimensional array specifying the size of each dimension. Now, here this is an another example, as we can see is a variable length array and in the first row this element, second row this element, third row does not contain element, fourth row, and fifth row does not contain element.

Now, here that as these rows remain back end we can we can initialize these rows with any number of elements. For example here, the fourth row we initialize with a double elements of size 66 all these rows are with only 2, but here this is 66 and here you can see access a particular element also can be done correctly at this is the fourth row and in the sixty fifth column we want to store 3.14.

So, these are the simple example that we have discussed and there is another good example that you can note it, the last statement is very interesting. Last statement basically we declare an array of type objects, as I already told you, objects is a superclass of any class is a galaxy class though any number of objects it can be stored in that array if you declare an array of type object. So, this is one example here, we declare the name of the array is objects, type is object and here you see all the arrays that we have declare so far, we can store it in.

So, this basically is a variable length array where variable length not only as it can is as it is a type of object so any type it can hold, that is the one important aspects. And we know that array is a homogeneous data that means it is stored always the similar in this sense it is basically storing the similar elements in it. So, this is an example that you can understand how array can be versatile manner, it can be declared, it can be initialized, it can be used in your program.

(Refer Slide Time: 20:21)



Example 12.3: Initialization of a 1D array with user input from keyboard

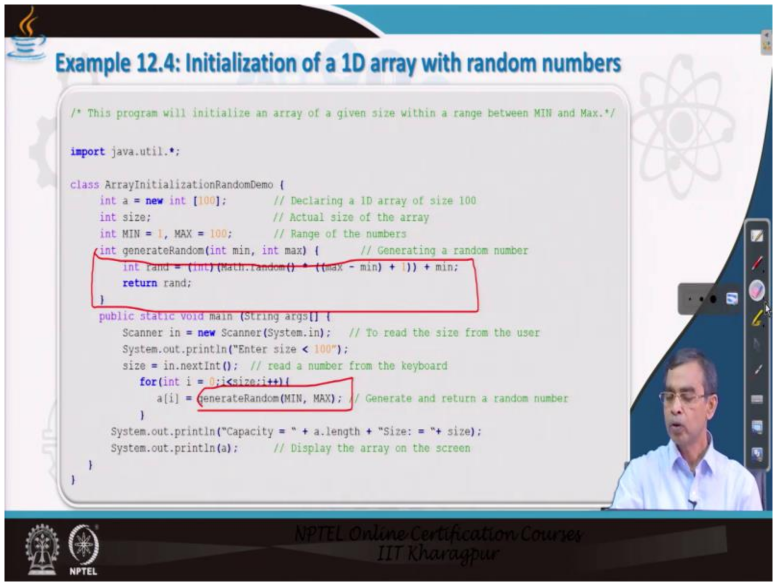
```java
class ArrayInitializationDemo {
    int a = new int [100];    // Declaring a 1D array of size 100
    int size;                 // Actual size of the array
    void loadArray() {        // Actual size of the array
        Scanner in = new Scanner(System.in);   // To read from keyboard
        System.out.println("Enter size < 100");
                size = in.nextInt();  // read a number
            for(int i = 0;i<size;i++){
                System.out.println("Enter "+ (i+1) + "th number ");
                a[i] = in.nextInt(); // Load the i-th entry
            }
      System.out.println("Length = " + a.length);
      System.out.println(a);        // Display the array on the screen
   }
}
```

Now, let us quickly come to the discussion of different operations on an array. Now, let us first discuss about how a 1D array can be created while you run a program. I gave an example for your easy understanding, again I advise you to download the program in your own computer and then run this program try to understand.

Here actually what I want to, my objective is to read the number of elements from the array including what is the size of the array. So n is the size of the array, size is the array which will be read from the keyboard, once the size is known for each element we want to load the value from reading each time on element, once it is complete its all elements are read from the keyboard, it basically print this element.

Now, here one thing length, now if you declare an array a, for example here array a, and you know, the length of this array is 100, whatever the size that we have declared. But the length is 100 but right now or you can see the capacity of the array is 100. But here we only store size, size is always less than 100, it should be 0 to 99 like, whatever it is there, size should be less than a 100. So, if we print a dot length, it will print 100 actually not that size, because it basically print the capacity of the array. So, this example that you can understand it.

(Refer Slide Time: 21:55)



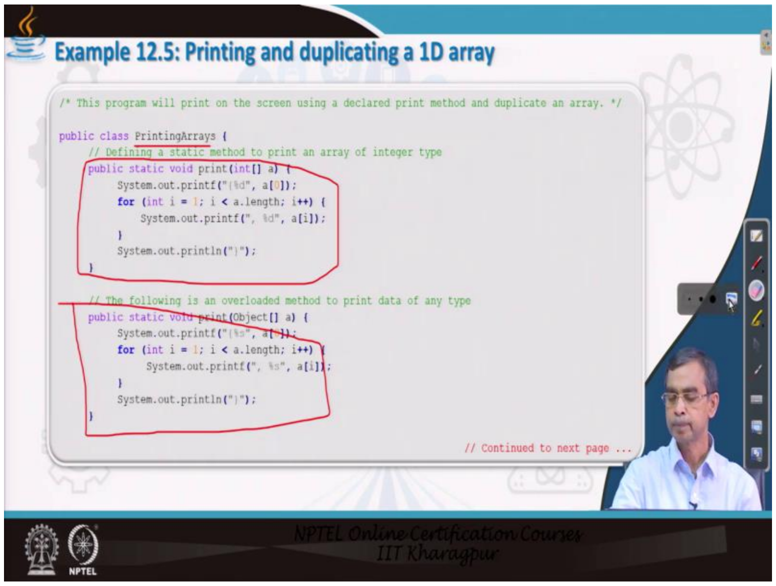
Example 12.4: Initialization of a 1D array with random numbers

```
/* This program will initialize an array of a given size within a range between MIN and Max.*/

import java.util.*;

class ArrayInitializationRandomDemo {
    int a = new int [100];          // Declaring a 1D array of size 100
    int size;                       // Actual size of the array
    int MIN = 1, MAX = 100;         // Range of the numbers
    int generateRandom(int min, int max) {       // Generating a random number
        int rand = (int)(Math.random() * ((max - min) + 1)) + min;
        return rand;
    }
    public static void main (String args[] {
        Scanner in = new Scanner(System.in);   // To read the size from the user
        System.out.println("Enter size < 100");
        size = in.nextInt();  // read a number from the keyboard
            for(int i = 0;i<size;i++){
                a[i] = generateRandom(MIN, MAX); // Generate and return a random number
            }
       System.out.println("Capacity = " + a.length + "Size: = "+ size);
       System.out.println(a);       // Display the array on the screen
   }
}
```

The similar example little bit with twist, by which we can load an array in a random number. So, this example again you can try how the random number can be created. I have declared one method for you, so this way the number in between the range min and max can be created.

And then again similar things, you have to read the what is the size of the array and once you know the size of the array, the random number generator method can be called, so that it will generate the random number and it load into the array. So, this way this program is basically load the array of a desire size and then the elements are in a random way. So, these are random array can be generated using this program.

(Refer Slide Time: 22:47)

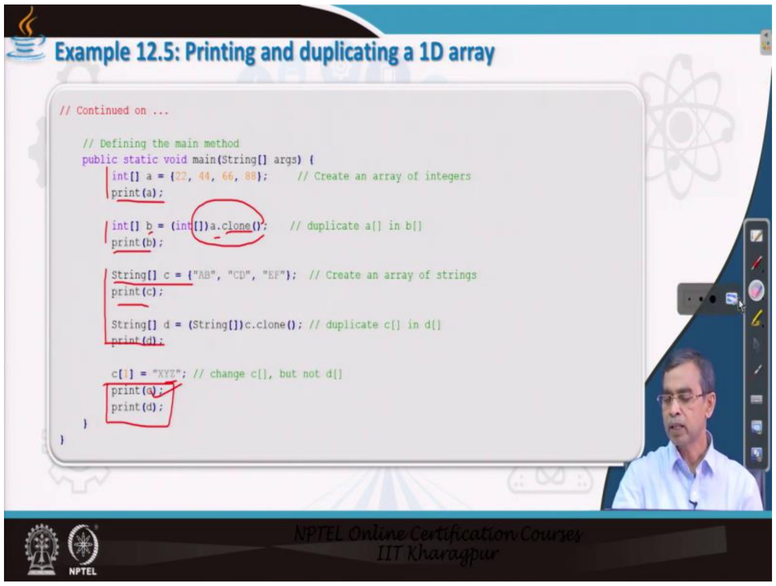


Example 12.5: Printing and duplicating a 1D array

```
/* This program will print on the screen using a declared print method and duplicate an array. */

public class PrintingArrays {
    // Defining a static method to print an array of integer type
    public static void print(int[] a) {
        System.out.printf("{%d", a[0]);
        for (int i = 1; i < a.length; i++) {
            System.out.printf(", %d", a[i]);
        }
        System.out.println("}");
    }

    // The following is an overloaded method to print data of any type
    public static void print(Object[] a) {
        System.out.printf("{%s", a[0]);
        for (int i = 1; i < a.length; i++) {
            System.out.printf(", %s", a[i]);
        }
        System.out.println("}");
    }

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

Now there are many operations, as we have discussed about creating other than creating inserting, deletion, traverstion, let us first discuss about the traverstion. Traversing is basically scanning the entire array starting from the zeroth location. And this is a one example that we can discussed about, we are declaring is an array and the name of the class that we have discuss is a print array, where we discussed the two print method; this print method will print if the array is integer, and this print method will print if the arrays of strings, so two method that we have discussed here in this program.

(Refer Slide Time: 23:21)



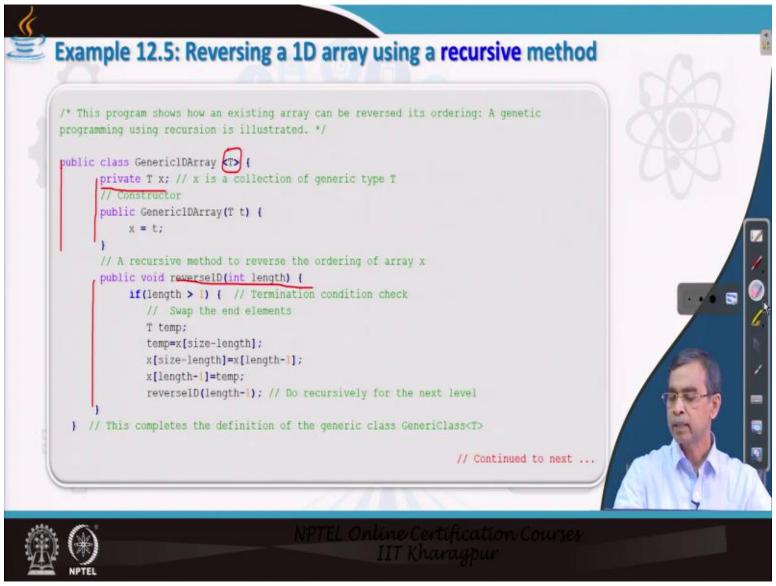
Example 12.5: Printing and duplicating a 1D array

```
// Continued on ...

    // Defining the main method
    public static void main(String[] args) {
        int[] a = {22, 44, 66, 88};      // Create an array of integers
        print(a);

        int[] b = (int[])a.clone();     // duplicate a[] in b[]
        print(b);

        String[] c = {"AB", "CD", "EF"};  // Create an array of strings
        print(c);

        String[] d = (String[])c.clone(); // duplicate c[] in d[]
        print(d);

        c[1] = "XYZ"; // change c[], but not d[]
        print(c);
        print(d);
    }
}
```

And next the program is continued here, in the next we can see how we can declare the different arrays, integer array it is declare and print method is declare, is another array. Now here you can see clone method, a clone is a method which is declared in the class object, it basically make a replica of the array a. So, this means that b and a are the two arrays, they are basically two instances of the arrays, that means a will remain b will remain, so what we can say two copies of the array which include the same elements will be maintained here. So, if we print b and print a we can see the same.

So, basically two array instances, but the name of the arrays are, they contain the same elements, but they are the different instances actually. Now, this is another example which basically declare first string arrays and clone it, and then print the clone. And here you can see, we can just change or update the, the index 1 with XYZ, initially 0 and 1 CD, so if you do it CD. And then if we print this one, you can see it will change, change will be effective in the c arrays but d arrays remain same, although they are initially cloned but after the modification one array has changed and, so this is the idea about the clone or copying an array by in a program.

(Refer Slide Time: 24:53)



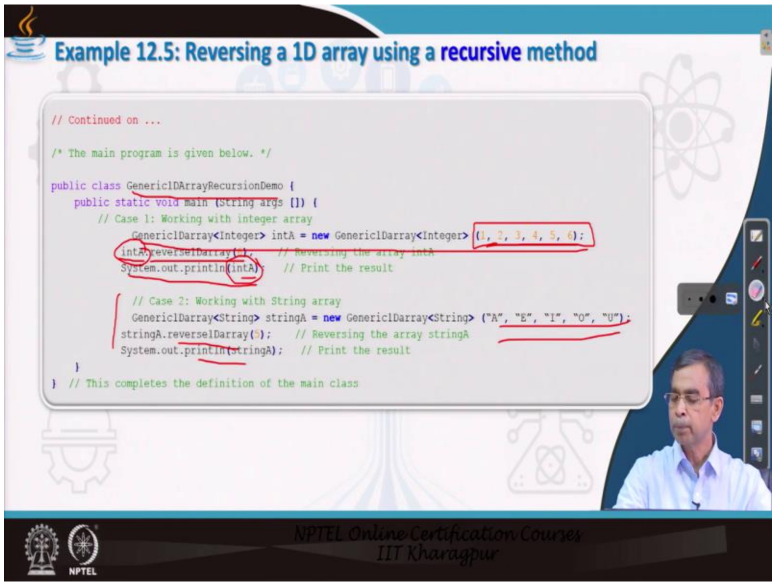
Example 12.5: Reversing a 1D array using a recursive method

```
/* This program shows how an existing array can be reversed its ordering: A genetic
programming using recursion is illustrated. */

public class Generic1DArray <T> {
      private T x; // x is a collection of generic type T
      // Constructor
      public Generic1DArray(T t) {
           x = t;
      }
      // A recursive method to reverse the ordering of array x
      public void reverse1D(int length) {
           if(length > 1) {  // Termination condition check
              //  Swap the end elements
              T temp;
              temp=x[size-length];
              x[size-length]=x[length-1];
              x[length-1]=temp;
              reverse1D(length-1); // Do recursively for the next level
      }
  }  // This completes the definition of the generic class GeneriClass<T>

                                                        // Continued to next ...
```

Now, this is the one example that I want to discuss again, it is a generic array, what is the generic array you see, I declare an array which will include not only integer or string it can, it can store any type. So, any type the generic type T, I have declared here and this is a usual convention that you already know how to declare a generic class.

So, these basically declaring a generic class, and in this class we declare one method, if this method is basically reversing all the elements in the array, so it is a reverse 1D that means a reverse. Reverse means ordering, if the array containing 1, 2, 3 after the reverse method is called for the array it will be print 3, 2, 1. So, this basically ordering reversing the order of the elements there. Now, so we have discussed one generic class with only one method, the reverse and then structure it is there.

(Refer Slide Time: 25:51)



Example 12.5: Reversing a 1D array using a recursive method

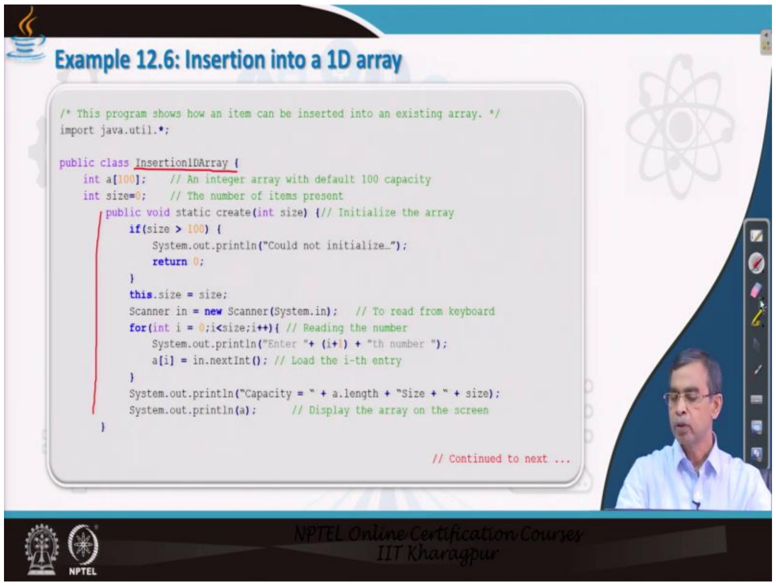
```
// Continued on ...

/* The main program is given below. */

public class Generic1DArrayRecursionDemo {
    public static void main (String args []) {
        // Case 1: Working with integer array
           Generic1Darray<Integer> intA = new Generic1Darray<Integer> (1, 2, 3, 4, 5, 6);
          intA.reverse1Darray(6);    // Reversing the array intA
          System.out.println(intA);    // Print the result

           // Case 2: Working with String array
           Generic1Darray<String> stringA = new Generic1Darray<String> ("A", "E", "I", "O", "U");
          stringA.reverse1Darray(5);    // Reversing the array stringA
          System.out.println(stringA);    // Print the result
    }
}  // This completes the definition of the main class
```

Now, let us see how we can write the main program to give a demo, so this is basically the main program that are giving a demo of this one. Here in this one, working with the integer array we declare the integer array and initialize this array with this one, this is the convention already you have learned about while we are discussing about generic program. And these basically call for this array object is basically integer array is a collection, integer array is a collection which includes integer elements and the elements are like this one.

And then we reverse array, we call this method and the size is 6 because it is the size is 6, then it will print and you can see how the collection can be printed. We have already learned how the collection can be printed giving the name of the collection, so it will print all the elements in this statement.

It is basically same thing, but it is for the string collection the strings are there A, E, I, O, U and it basically reverse, and you will print, it is in the order is reversed. So, these are the two simple example, you can understand about how the generic class also can be extended to our arrays and array can be mentioned genetically, that means generic programming can be apply to our conventional array also.

(Refer Slide Time: 27:05)



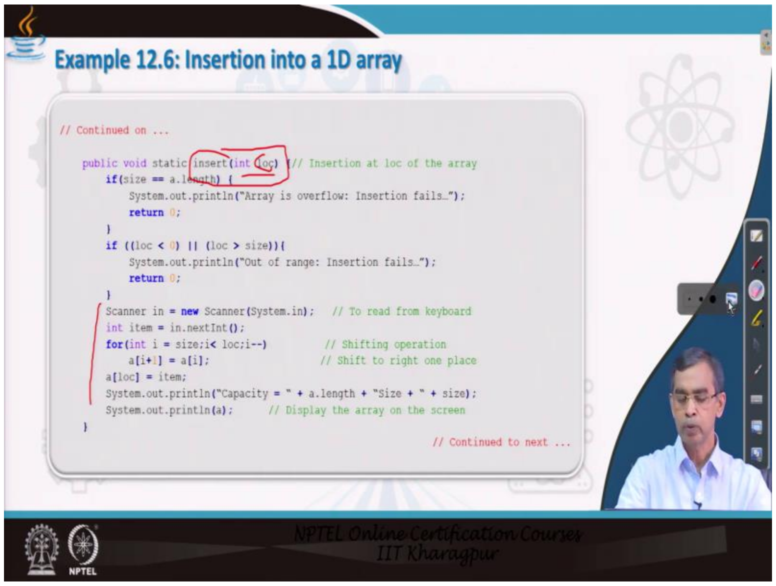
Example 12.6: Insertion into a 1D array

```
/* This program shows how an item can be inserted into an existing array. */
import java.util.*;

public class Insertion1DArray {
    int a[100];    // An integer array with default 100 capacity
    int size=0;    // The number of items present
      public void static create(int size) {// Initialize the array
         if(size > 100) {
            System.out.println("Could not initialize…");
            return 0;
         }
         this.size = size;
         Scanner in = new Scanner(System.in);   // To read from keyboard
         for(int i = 0;i<size;i++){ // Reading the number
            System.out.println("Enter "+ (i+1) + "th number ");
            a[i] = in.nextInt(); // Load the i-th entry
         }
         System.out.println("Capacity = " + a.length + "Size + " + size);
         System.out.println(a);      // Display the array on the screen
      }

                                                  // Continued to next ...
```

NPTEL Online Certification Courses
IIT Kharagpur

Now, let us come to the discussion about Insertion array. The insertion array is very simple, in this program, this program is basically insertion 1D array. First, we give a method by which an array can be created by reading the number from the keyboard, which we have already discussed earlier.

(Refer Slide Time: 27:28)



Example 12.6: Insertion into a 1D array

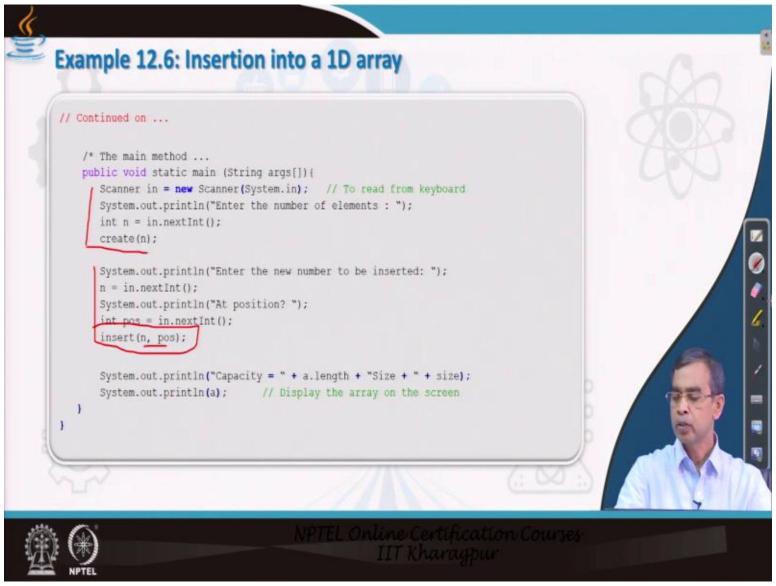
```
// Continued on ...

    public void static insert(int loc) {// Insertion at loc of the array
       if(size == a.length) {
          System.out.println("Array is overflow: Insertion fails…");
          return 0;
       }
       if ((loc < 0) || (loc > size)){
          System.out.println("Out of range: Insertion fails…");
          return 0;
       }
       Scanner in = new Scanner(System.in);   // To read from keyboard
       int item = in.nextInt();
       for(int i = size;i< loc;i--)          // Shifting operation
          a[i+1] = a[i];                     // Shift to right one place
       a[loc] = item;
       System.out.println("Capacity = " + a.length + "Size + " + size);
       System.out.println(a);      // Display the array on the screen
    }
                                                  // Continued to next ...
```

NPTEL Online Certification Courses
IIT Kharagpur

And then the after the array is inserted, array created our objective is to insert an element into a position. So this position here in this program, we read the position from the, from the user let it be the loc, and then this method we declare so that it can insert into a position loc reading the elements from the user. Now here if the loc is greater than length, then definitely it should gave a message that array is full or if it is less than then it will insert it.

But here while the insertion is there, you know, the rest of the elements needs to be pushed to later. So, that is why these basically create a room to make the place for the array to be placed here. So, this is the code that you can consider to create a room for this one and finally it will load the element into the loc position.

(Refer Slide Time: 28:19)

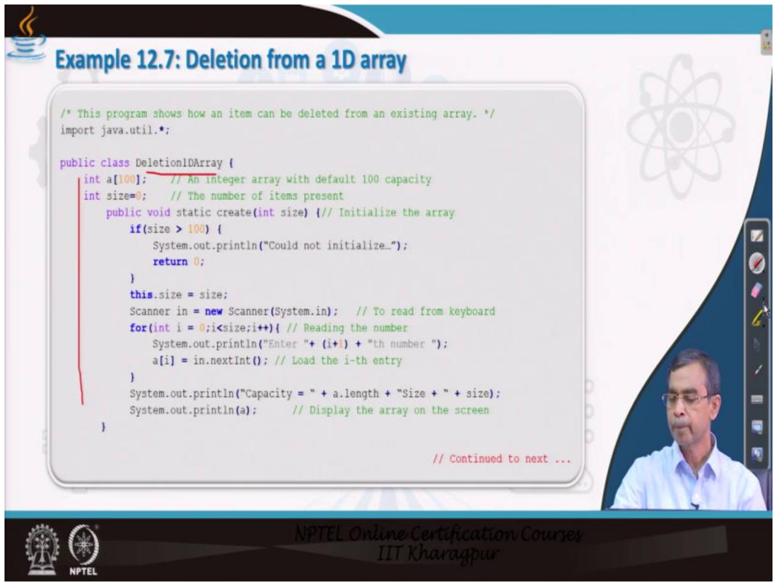


Example 12.6: Insertion into a 1D array

```
// Continued on ...

   /* The main method ...
   public void static main (String args[]){
      Scanner in = new Scanner(System.in);   // To read from keyboard
      System.out.println("Enter the number of elements : ");
      int n = in.nextInt();
      create(n);

      System.out.println("Enter the new number to be inserted: ");
      n = in.nextInt();
      System.out.println("At position? ");
      int pos = in.nextInt();
      insert(n, pos);

      System.out.println("Capacity = " + a.length + "Size + " + size);
      System.out.println(a);      // Display the array on the screen
   }
}
```

So, this basically consider that read the element from the current location, create an array, and then elements to be inserted into the position this one. So, again, I should advise you to suggest, I should advise you to run this program and then see how it works for you. But in case of insertion as we have learned is that, in order to make a place for this without deleting the existing element you have to shift down one place each starting from the current position where you want to insert.

(Refer Slide Time: 29:13)



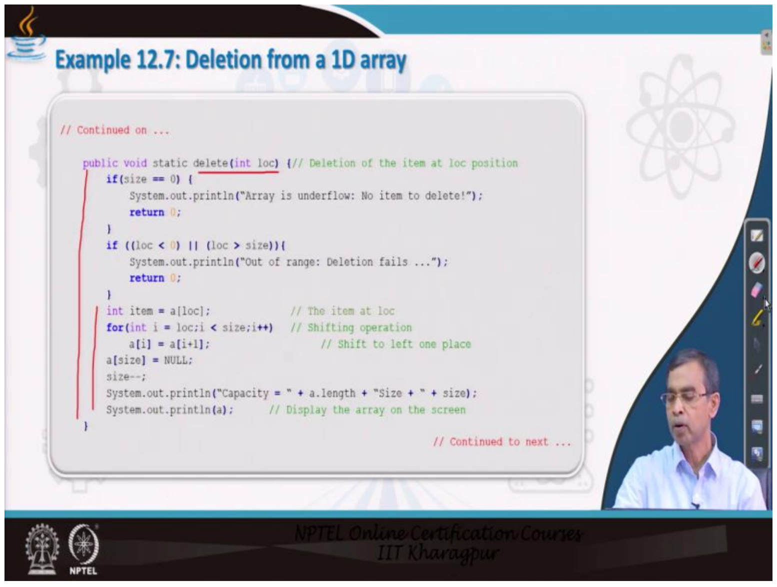
Example 12.7: Deletion from a 1D array

```java
/* This program shows how an item can be deleted from an existing array. */
import java.util.*;

public class Deletion1DArray {
    int a[100];    // An integer array with default 100 capacity
    int size=0;    // The number of items present
        public void static create(int size) {// Initialize the array
            if(size > 100) {
                System.out.println("Could not initialize…");
                return 0;
            }
            this.size = size;
            Scanner in = new Scanner(System.in);   // To read from keyboard
            for(int i = 0;i<size;i++){ // Reading the number
                System.out.println("Enter "+ (i+1) + "th number ");
                a[i] = in.nextInt(); // Load the i-th entry
            }
            System.out.println("Capacity = " + a.length + "Size + " + size);
            System.out.println(a);      // Display the array on the screen
        }

                                                    // Continued to next ...
```

NPTEL Online Certification Courses
IIT Kharagpur

Now, likewise deletion operation is the same way but deletion in case of insertion you have to push down, whereas in case of deletion if you want to remove it remove the element, but do not keep the place void so we have to push up one element from the rest to fill the void. This is the same program, it basically deletion 1D array, first this the method how to create the array, once the array is created again we just do the deletion method.

(Refer Slide Time: 29:30)



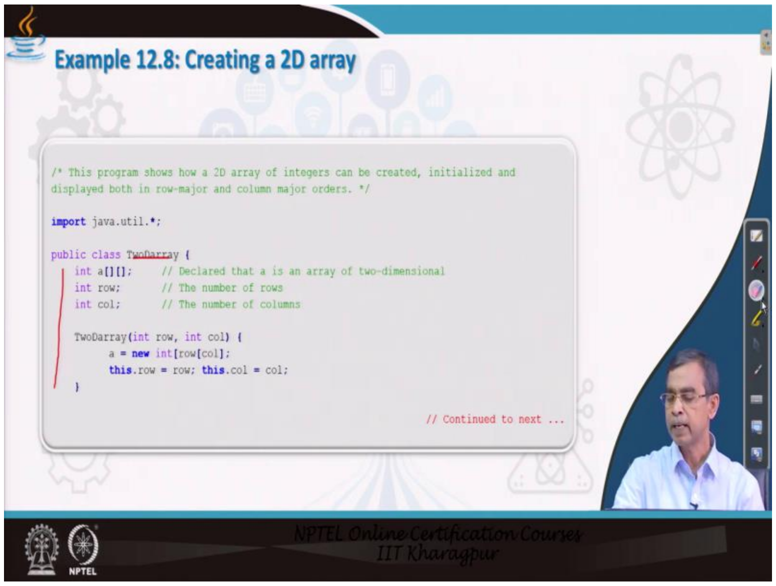
Example 12.7: Deletion from a 1D array

```java
// Continued on ...

    public void static delete(int loc) {// Deletion of the item at loc position
        if(size == 0) {
            System.out.println("Array is underflow: No item to delete!");
            return 0;
        }
        if ((loc < 0) || (loc > size)){
            System.out.println("Out of range: Deletion fails ...");
            return 0;
        }
        int item = a[loc];                 // The item at loc
        for(int i = loc;i < size;i++)      // Shifting operation
            a[i] = a[i+1];                         // Shift to left one place
        a[size] = NULL;
        size--;
        System.out.println("Capacity = " + a.length + "Size + " + size);
        System.out.println(a);      // Display the array on the screen
    }
                                                    // Continued to next ...
```

NPTEL Online Certification Courses
IIT Kharagpur

Here the deletion method is declared here. This is the deletion method, delete int loc that mean we want to delete an element from the loc position. And this is a usual code that you can check it and this basically the after deletion the push up one location for all elements, so that the vacant position can be filled. So, this a deletion operation.

(Refer Slide Time: 30:00)

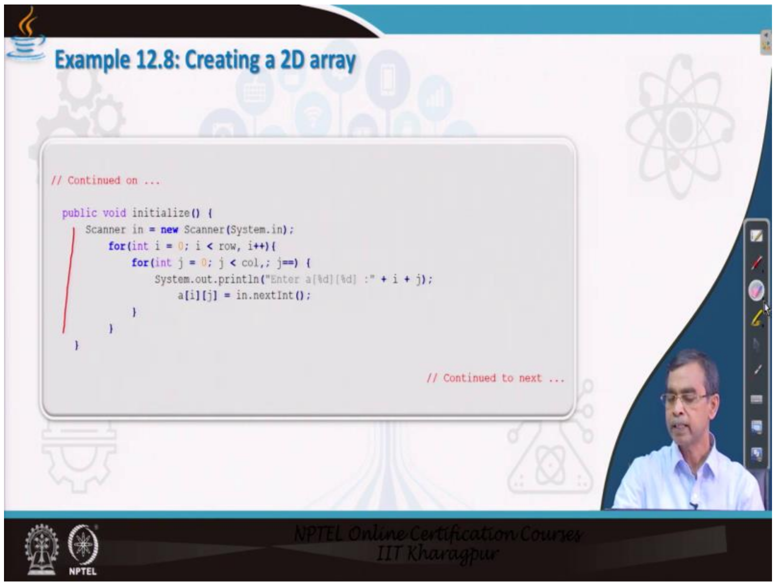


Example 12.8: Creating a 2D array

```
/* This program shows how a 2D array of integers can be created, initialized and
displayed both in row-major and column major orders. */

import java.util.*;

public class TwoDarray {
    int a[][];      // Declared that a is an array of two-dimensional
    int row;        // The number of rows
    int col;        // The number of columns

    TwoDarray(int row, int col) {
          a = new int[row[col];
          this.row = row; this.col = col;
    }

                                                    // Continued to next ...
```

Now, matrix operations, we will quickly discuss about how the matrix manipulation can be done. It is basically the addition or subtraction or multiplication. Now, this example explain you how a 2D array can be initialized, this is basically the idea that a array can be initialized. Once the array is initialized, we can load the array using the element.

(Refer Slide Time: 30:20)

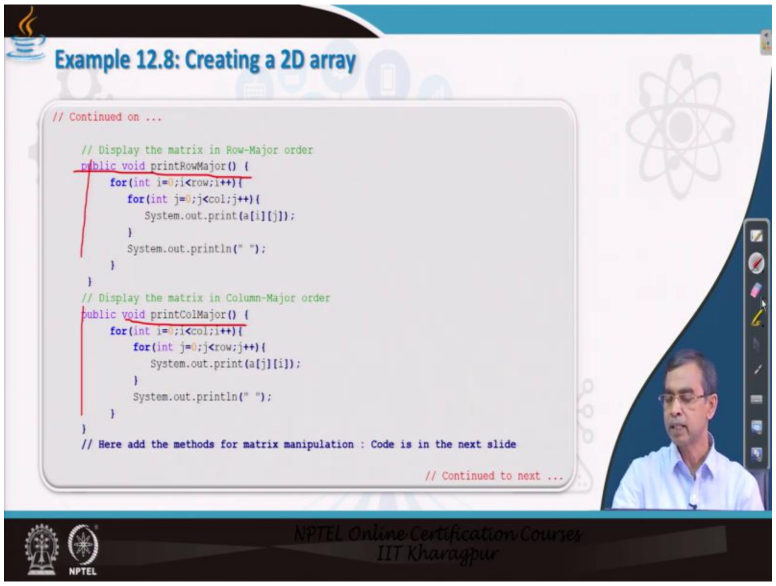


Example 12.8: Creating a 2D array

```
// Continued on ...

  public void initialize() {
      Scanner in = new Scanner(System.in);
          for(int i = 0; i < row, i++){
              for(int j = 0; j < col,; j==) {
                  System.out.println("Enter a[%d][%d] :" + i + j);
                      a[i][j] = in.nextInt();
              }
          }
    }

                                                    // Continued to next ...
```

So, here the load is basically it is, basically reading the number of elements from the keyboard this method tell you how the elements in the matrix can be loaded. So this way any number of matrix can be initialized, once the array is initialized then many operations that we can perform.

(Refer Slide Time: 30:37)

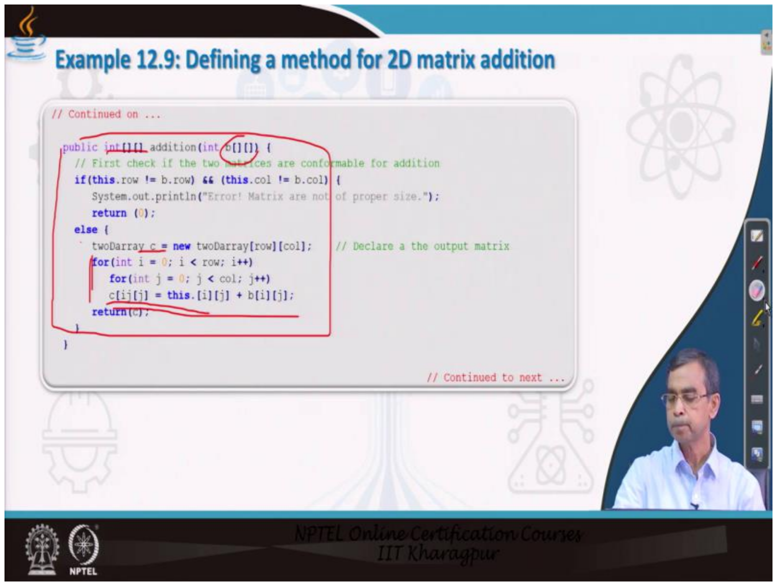


Example 12.8: Creating a 2D array

```java
// Continued on ...

    // Display the matrix in Row-Major order
    public void printRowMajor() {
        for(int i=0;i<row;i++){
           for(int j=0;j<col;j++){
              System.out.print(a[i][j]);
           }
           System.out.println(" ");
        }
    }
    // Display the matrix in Column-Major order
    public void printColMajor() {
        for(int i=0;i<col;i++){
           for(int j=0;j<row;j++){
              System.out.print(a[j][i]);
           }
           System.out.println(" ");
        }
    }
    // Here add the methods for matrix manipulation : Code is in the next slide

                                                        // Continued to next ...
```

And this is a one example where we can do the addition operation of two arrays. So, this method is basically print the array in row major order, this is another print method in the column major order, the logic you can understand easily. Because the theory that we have discussed if you follow then this logic will be clearly understandable to you. Once the, these are the method that we have declared for the program. Now, let us see how the different operations like addition can be declared.

(Refer Slide Time: 31:08)

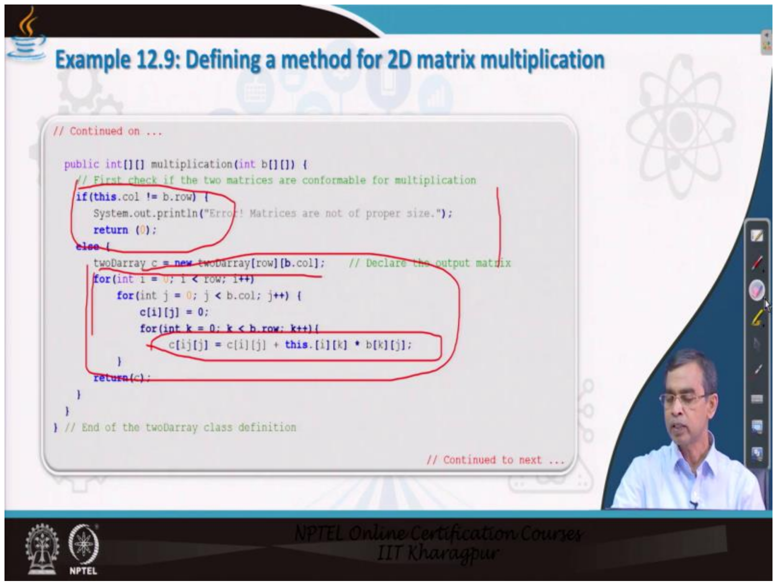


Example 12.9: Defining a method for 2D matrix addition

```
// Continued on ...

public int[][] addition(int b[][]) {
  // First check if the two matrices are conformable for addition
  if(this.row != b.row) && (this.col != b.col) {
     System.out.println("Error! Matrix are not of proper size.");
     return (0);
  else {
     twoDarray c = new twoDarray[row][col];    // Declare a the output matrix
     for(int i = 0; i < row; i++)
        for(int j = 0; j < col; j++)
        c[ij[j] = this.[i][j] + b[i][j];
     return(c);
  }
}

                                              // Continued to next ...
```

Now, this is the method by which that two arrays, two matrix can be added, addition of the two matrix can be done. So, you have to basically call this method for passing an argument of another method. So, if the this method is called for an array a and then pass the argument b, then it is basically a plus b. The logic is very simple, so you have to create a another resultant matrix is called the c and then initialize this matrix by this one. So, for every rows and columns element should be added with the rows and columns of the other arrays, the result will be stored this one. And here you see two loops are to be followed here. So, this is a matrix addition.

(Refer Slide Time: 31:57)



Example 12.9: Defining a method for 2D matrix multiplication

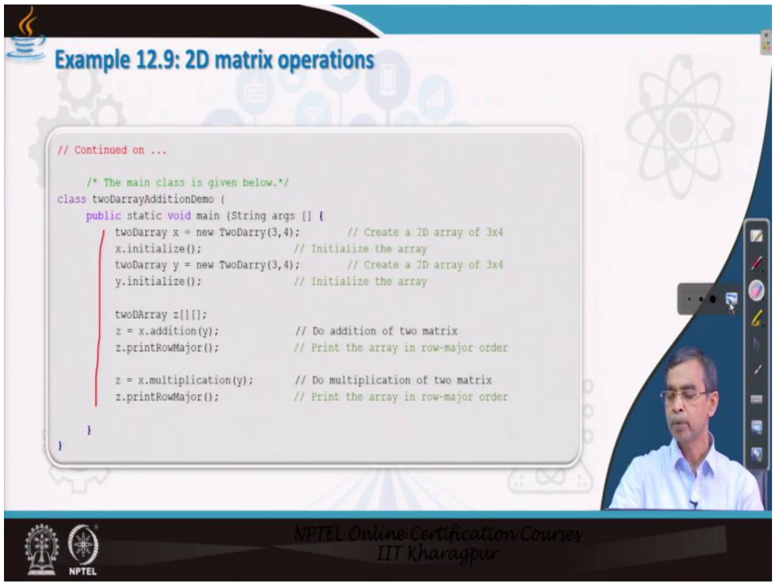
```
// Continued on ...

 public int[][] multiplication(int b[][]) {
   // First check if the two matrices are conformable for multiplication
   if(this.col != b.row) {
      System.out.println("Error! Matrices are not of proper size.");
      return (0);
   else {
      twoDarray c = new twoDarray[row][b.col];    // Declare the output matrix
      for(int i = 0; i < row; i++)
         for(int j = 0; j < b.col; j++) {
            c[i][j] = 0;
            for(int k = 0; k < b.row; k++){
               c[ij[j] = c[i][j] + this.[i][k] * b[k][j];
         }
      return(c);
   }
 }
} // End of the twoDarray class definition

                                              // Continued to next ...
```

Likewise, the matrix multiplication algorithm also can be there, but for the matrix multiplication you have to first check whether the two arrays are ready for the matrix multiplication or not. So, this basically the checking is there if they do not satisfy this criteria, then array multiplication cannot be done. But for this multiplication again same as addition like the two matrix a and b, c equals to a cross b, here is the three, this is basically initialize the array and then multiply the matrix this is the logic that you can show. And here you see three loops are to be there, so for the matrix multiplication operation is concerned.

Anyway, logic is bit standard logic rather, then you can check that logic whether it is working or not. Suggestion is that you can try the code of your own considering the logic that algorithm already I have discussed in the last video lectures and then try to write the code of your own. If it works that is fine, it will give really a very much good confidence to you, otherwise you can check this program and then you can run it, and then you can taste it.

And then also you can compare out of the two program one written by you without any help from anybody and without this, with the taking the help from this code, how the two things are comparable. So, this is the matrix multiplication operation that we have discuss.

(Refer Slide Time: 33:18)

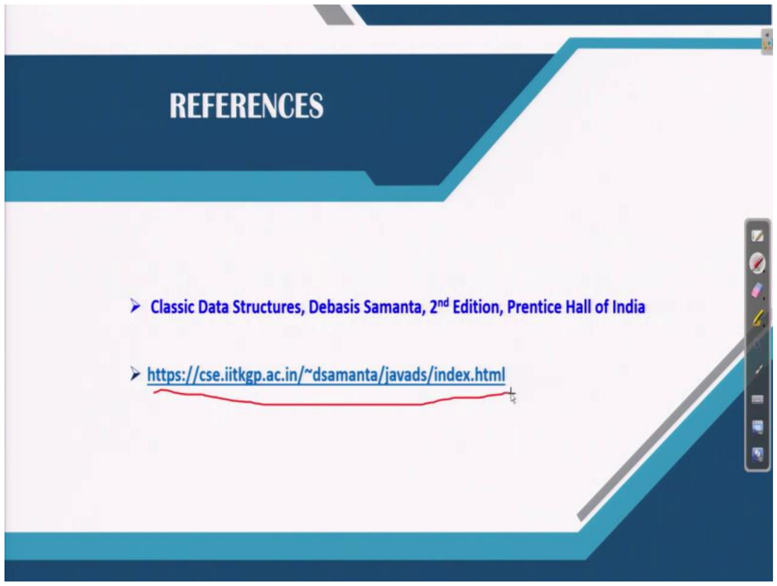


Example 12.9: 2D matrix operations

```
// Continued on ...

    /* The main class is given below.*/
class twoDarrayAdditionDemo {
    public static void main (String args [] {
        twoDarray x = new TwoDarry(3,4);        // Create a 2D array of 3x4
        x.initialize();                      // Initialize the array
        twoDarray y = new TwoDarry(3,4);        // Create a 2D array of 3x4
        y.initialize();                      // Initialize the array

        twoDArray z[][];
        z = x.addition(y);                   // Do addition of two matrix
        z.printRowMajor();                   // Print the array in row-major order

        z = x.multiplication(y);             // Do multiplication of two matrix
        z.printRowMajor();                   // Print the array in row-major order

    }
}
```

And this is basically the main program, which call all the method that we have discuss addition, subtraction, and printing the matrix in row major order, printing the matrix in column major order, multiplication and everything. So, this is the program that you can test with your own effort.

(Refer Slide Time: 33:40)



REFERENCES

- Classic Data Structures, Debasis Samanta, 2nd Edition, Prentice Hall of India
- https://cse.iitkgp.ac.in/~dsamanta/javads/index.html

And you can see the different algorithms for the data structure related manipulation, this book you can follow. And the all the course that I have followed in this video lecture, you can get all the code from this link also. So, code is ready available and then you can try it. Thank you.