

Lesson 02 Demo 01

Working with Arrays

Objective: To create and use one-dimensional and two-dimensional arrays in Java.

Tools required: Eclipse IDE

Prerequisites: None

Steps to be followed:

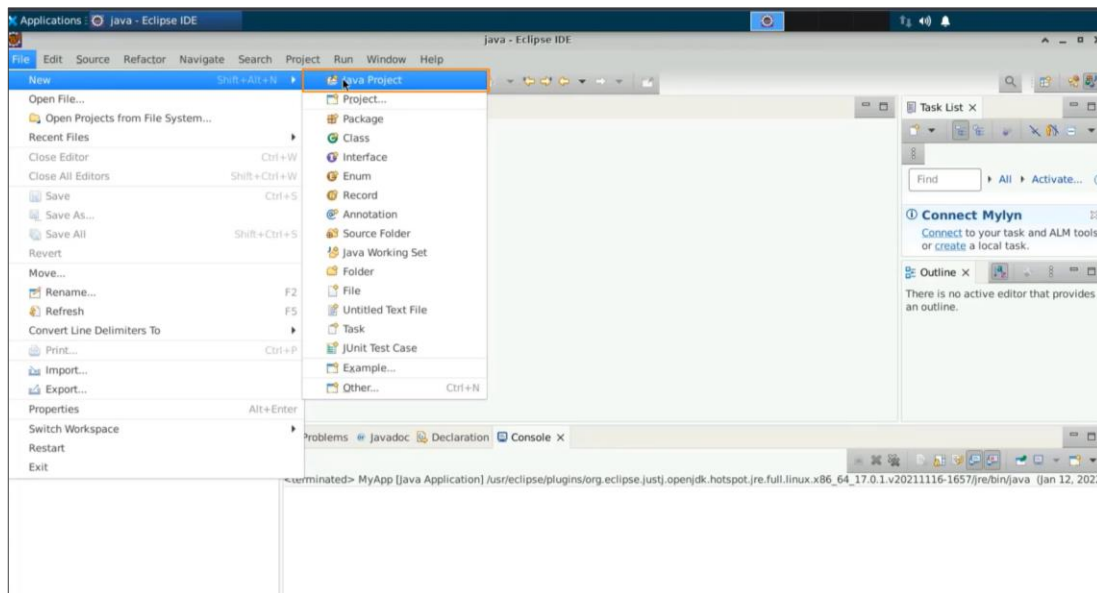
1. Use a one-dimensional array citing suitable examples
2. Create a one-dimensional array in different ways
3. Use the new operator to create an array inside a heap
4. Access elements in an array
5. Create an array with a specific size
6. Implement a two-dimensional array with suitable examples

Step 1: Use a one-dimensional array citing suitable examples

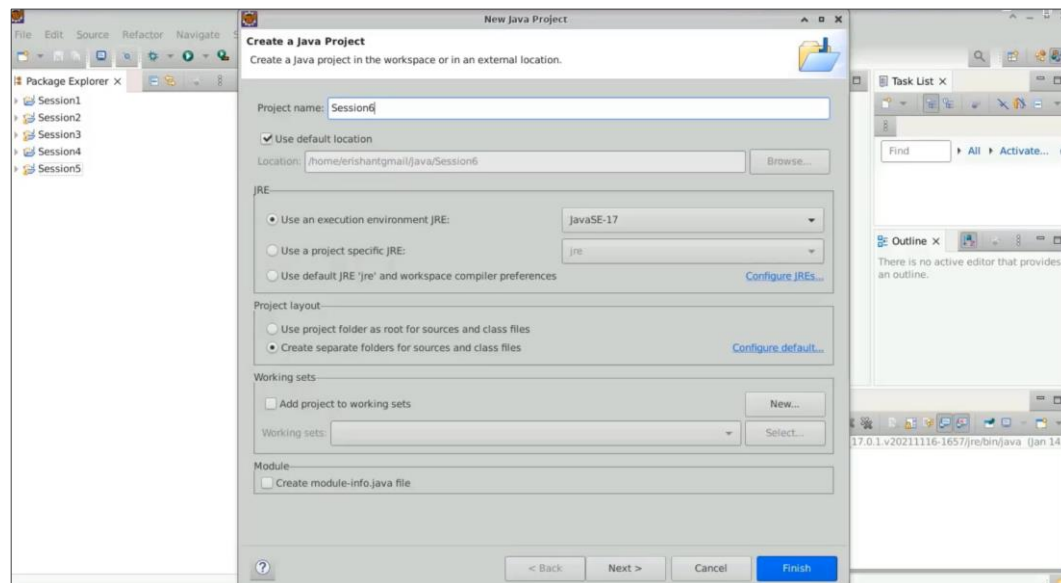
1.1 Open the Eclipse IDE



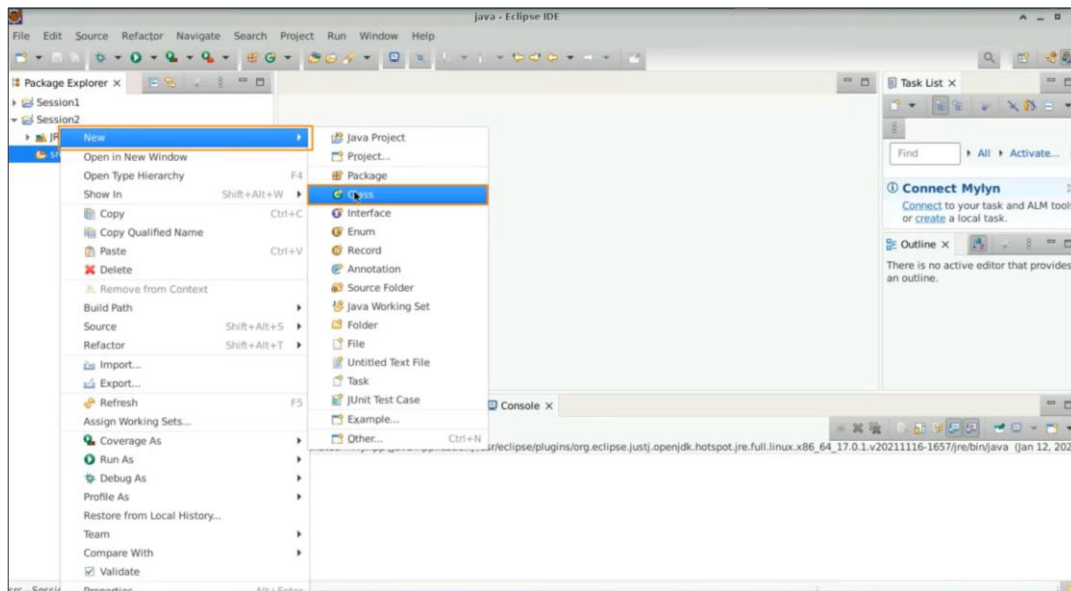
1.2. Select **File**, then **New**, and then **Java project**



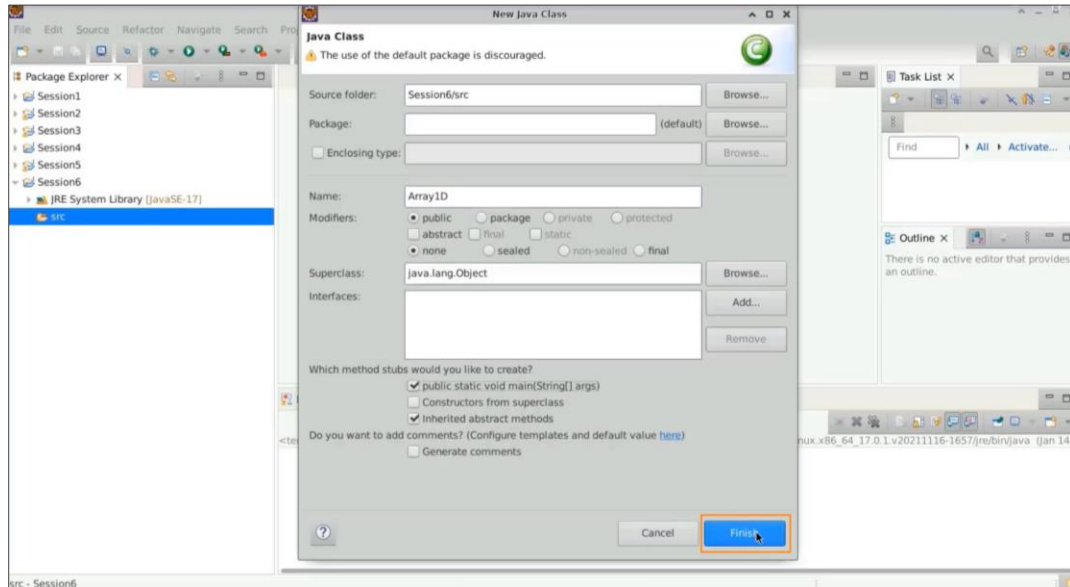
1.3 Name the project **“Session6”**, uncheck **“Create a module info dot Java file”**, and click on the **Finish** button



1.4 With a **Session6** on the src, do a right-click and create a **new class**

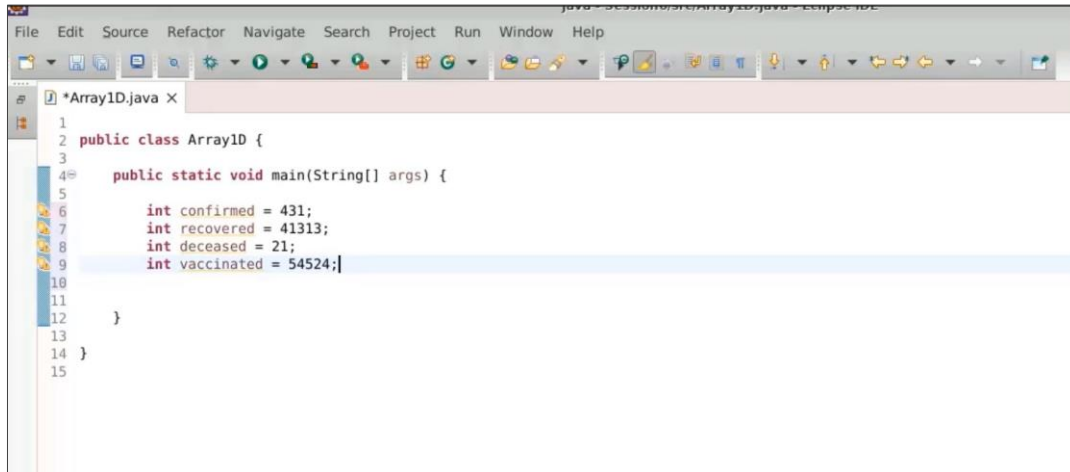


1.5 Name this class as an **Array1D**, then select the **main method**, and then select **finish**



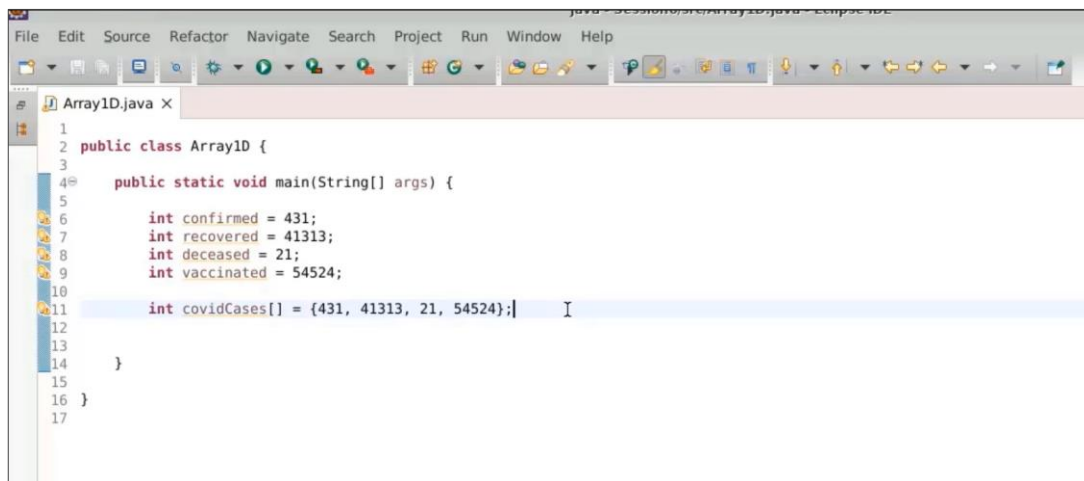
1.6 For a state, if you wish to save the COVID data, you must consider different attributes.

The first is the number of confirmed cases, followed by the number of recoveries and the number of deceased individuals. Next, consider how many are vaccinated; these figures can be stored in a single-value container. So, create these variables to represent the COVID data, and manage the information for one single state:



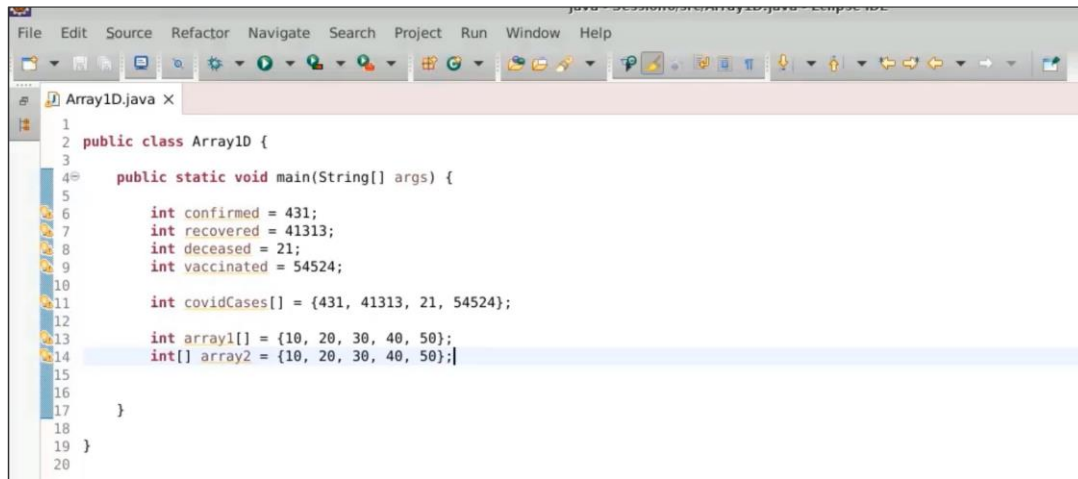
```
1
2 public class Array1D {
3
4     public static void main(String[] args) {
5
6         int confirmed = 431;
7         int recovered = 41313;
8         int deceased = 21;
9         int vaccinated = 54524;
10
11     }
12 }
13
14
15
```

1.7 You can also create one storage container by the name of COVID cases, and this can be created as an array and you need to add these square brackets, here you can represent the data in a single go and this will save your development time:



```
1
2 public class Array1D {
3
4     public static void main(String[] args) {
5
6         int confirmed = 431;
7         int recovered = 41313;
8         int deceased = 21;
9         int vaccinated = 54524;
10
11         int covidCases[] = {431, 41313, 21, 54524};
12
13     }
14 }
15
16
17
```

1.8 Create an array as **array1**, add a bracket, and write some numbers and this will be in one go. The bracket can be also placed before, as in here which is **array2**, which is the data

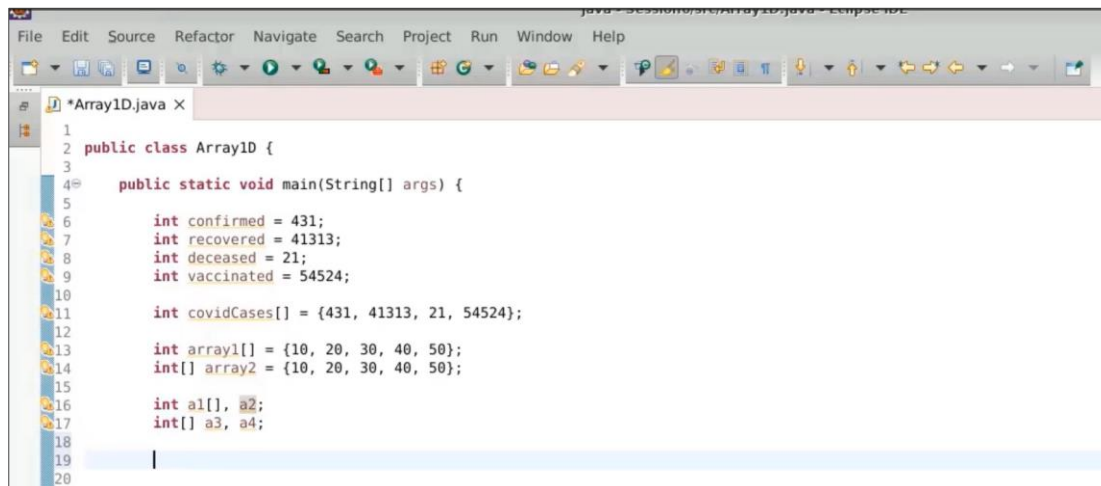


```

1
2 public class Array1D {
3
4     public static void main(String[] args) {
5
6         int confirmed = 431;
7         int recovered = 41313;
8         int deceased = 21;
9         int vaccinated = 54524;
10
11         int covidCases[] = {431, 41313, 21, 54524};
12
13         int array1[] = {10, 20, 30, 40, 50};
14         int[] array2 = {10, 20, 30, 40, 50};
15
16     }
17 }
18
19
20

```

1.9 What is the difference when you place the bracket before or the bracket after? The difference is that if you create an **array1** and then **a2**, then this **a2** is a normal integer whereas **a1** represents an array. And if you put the bracket before, then **a3** and **a4** are both arrays



```

1
2 public class Array1D {
3
4     public static void main(String[] args) {
5
6         int confirmed = 431;
7         int recovered = 41313;
8         int deceased = 21;
9         int vaccinated = 54524;
10
11         int covidCases[] = {431, 41313, 21, 54524};
12
13         int array1[] = {10, 20, 30, 40, 50};
14         int[] array2 = {10, 20, 30, 40, 50};
15
16         int a1[], a2;
17         int[] a3, a4;
18
19     }
20

```

Step 2: Create a one-dimensional array in different ways

2.1 In order to understand some more syntaxes, you can create an **array3**, hence bracket before or bracket after with this new integer and then you can pass the data. New creates an array of integers inside the heap area, this is an implicit syntax where this new integer and bracket is automatically taken care of by the compiler. Here **array1** is an implicit statement whereas **array3** is referred to as an explicit statement

```

1
2 public class Array1D {
3
4     public static void main(String[] args) {
5
6         int confirmed = 431;
7         int recovered = 41313;
8         int deceased = 21;
9         int vaccinated = 54524;
10
11         int covidCases[] = {431, 41313, 21, 54524};
12
13         int array1[] = {10, 20, 30, 40, 50};
14         int[] array2 = {10, 20, 30, 40, 50};
15
16         int a1[], a2;
17         int[] a3, a4;
18
19         int array3[] = new int[] {10, 20, 30, 40, 50};
20

```

2.2 The same part of **array3** here with the bracket before can also be used, hence new integer can be given or it can be skipped. Hence, these are the different ways to come up with and create an array

```

1
2 public class Array1D {
3
4     public static void main(String[] args) {
5
6         int confirmed = 431;
7         int recovered = 41313;
8         int deceased = 21;
9         int vaccinated = 54524;
10
11         int covidCases[] = {431, 41313, 21, 54524};
12
13         int array1[] = {10, 20, 30, 40, 50};
14         int[] array2 = {10, 20, 30, 40, 50};
15
16         int a1[], a2;
17         int[] a3, a4;
18
19         int array3[] = new int[] {10, 20, 30, 40, 50};
20         int[] arrays = new int[] {10, 20, 30, 40, 50};

```

2.3 In case you want to create an array with the size, you can give as **array5** is a new integer and mention the size as **5**. You can even give as bracket before, **array6** is a new integer with a size as **10** as shown below:

```

1
2 public class Array1D {
3
4     public static void main(String[] args) {
5
6         int confirmed = 431;
7         int recovered = 41313;
8         int deceased = 21;
9         int vaccinated = 54524;
10
11         int covidCases[] = {431, 41313, 21, 54524};
12
13         int array1[] = {10, 20, 30, 40, 50};
14         int[] array2 = {10, 20, 30, 40, 50};
15
16         int a1[], a2;
17         int[] a3, a4;
18
19         int array3[] = new int[] {10, 20, 30, 40, 50};
20         int[] array4 = new int[] {10, 20, 30, 40, 50};
21
22         // All the values will be 0 by default initially
23         int array5[] = new int[5];
24         int[] array6 = new int[10];
25
26     }

```

2.4 Now, what is the difference? previously you have the array elements along with the values but all the values are zero by default initially. But for **array5** and **array6**, you can provide the data, hence let us give as for the **array5** the **first index be 20** and then for **array5** of the **zeroth index as 10**

```

1
2 public class Array1D {
3
4     public static void main(String[] args) {
5
6         int confirmed = 431;
7         int recovered = 41313;
8         int deceased = 21;
9         int vaccinated = 54524;
10
11         int covidCases[] = {431, 41313, 21, 54524};
12
13         int array1[] = {10, 20, 30, 40, 50};
14         int[] array2 = {10, 20, 30, 40, 50};
15
16         int a1[], a2;
17         int[] a3, a4;
18
19         int array3[] = new int[] {10, 20, 30, 40, 50};
20         int[] array4 = new int[] {10, 20, 30, 40, 50};
21
22         // All the values will be 0 by default initially
23         int array5[] = new int[5];
24         int[] array6 = new int[10];
25
26         array5[0] = 10;
27         array5[1] = 20;
28
29     }
30
31 }

```

2.5 The upper ones are your primitive types and called as single value containers as they store only one single value. Whereas when it comes to arrays, these are the reference type and they store multiple values

```

1  public class Array1D {
2
3      public static void main(String[] args) {
4
5          // Primitive Types: Single Value Containers
6          int confirmed = 431;
7          int recovered = 41313;
8          int deceased = 21;
9          int vaccinated = 54524;
10
11          // Reference Type: Multiple Value Container
12          int covidCases[] = {431, 41313, 21, 54524};
13
14          int array1[] = {10, 20, 30, 40, 50};
15          int[] array2 = {10, 20, 30, 40, 50};
16
17          int a1[], a2;
18          int[] a3, a4;
19
20          int array3[] = new int[] {10, 20, 30, 40, 50};
21          int[] array4 = new int[] {10, 20, 30, 40, 50};
22
23          // All the values will be 0 by default initially
24          int array5[] = new int[5];
25          int[] array6 = new int[10];
26
27          array5[0] = 10;
28          array5[1] = 20;
29
30
31
32      }

```

2.6 Print and write "**confirmed is**" followed by the value of the confirmed variable, which means you are just printing the variable's name and its value. Similarly, write "**COVID cases are**" followed by the value of the COVID cases variable. When you execute the program, you will see the value 431 for confirmed. However, for COVID cases, you will see a hash code indicating that it is an array of integers

```

File Edit Source Refactor Navigate Search Project Run Window Help
Array1D.java
1  public class Array1D {
2
3      public static void main(String[] args) {
4
5          // Primitive Types: Single Value Containers
6          int confirmed = 431;
7          int recovered = 41313;
8          int deceased = 21;
9          int vaccinated = 54524;
10
11          // Reference Type: Multiple Value Container
12          int covidCases[] = {431, 41313, 21, 54524};
13
14          int array1[] = {10, 20, 30, 40, 50};
15          int[] array2 = {10, 20, 30, 40, 50};
16
17          int a1[], a2;
18          int[] a3, a4;
19
20          int array3[] = new int[] {10, 20, 30, 40, 50};
21          int[] array4 = new int[] {10, 20, 30, 40, 50};
22
23          // All the values will be 0 by default initially
24          int array5[] = new int[5];
25          int[] array6 = new int[10];
26
27          array5[0] = 10;
28          array5[1] = 20;
29
30          System.out.println("confirmed is: "+confirmed);
31          System.out.println("covidCases is: "+covidCases);
32
33      }
34
35  }
36
37
38

```

Console Output:

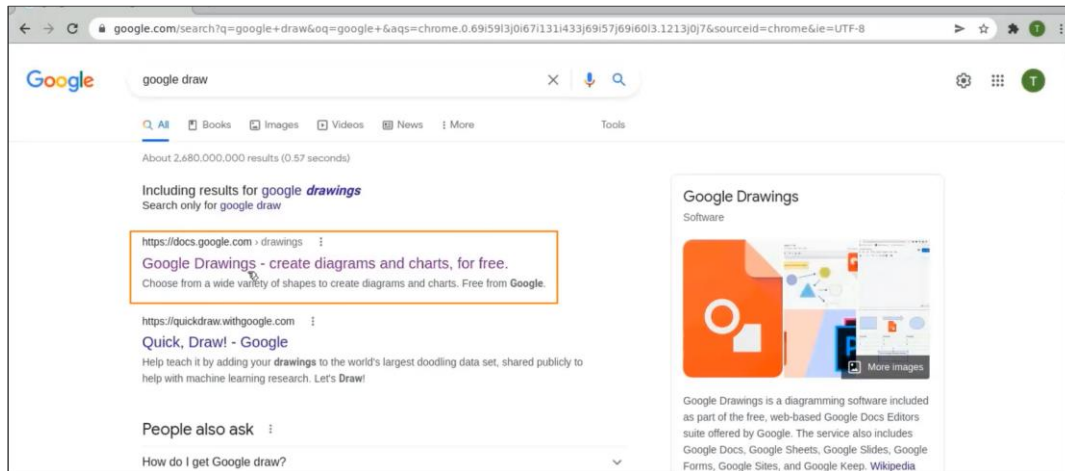
```

<terminated> Array1D [Java Application] /usr/eclipse/plugins/org.eclipse.justi.openjdk.hotspot.jre.full/bin/linux.x86_64/java:
confirmed is: 431
covidCases is: [I@53e25b76

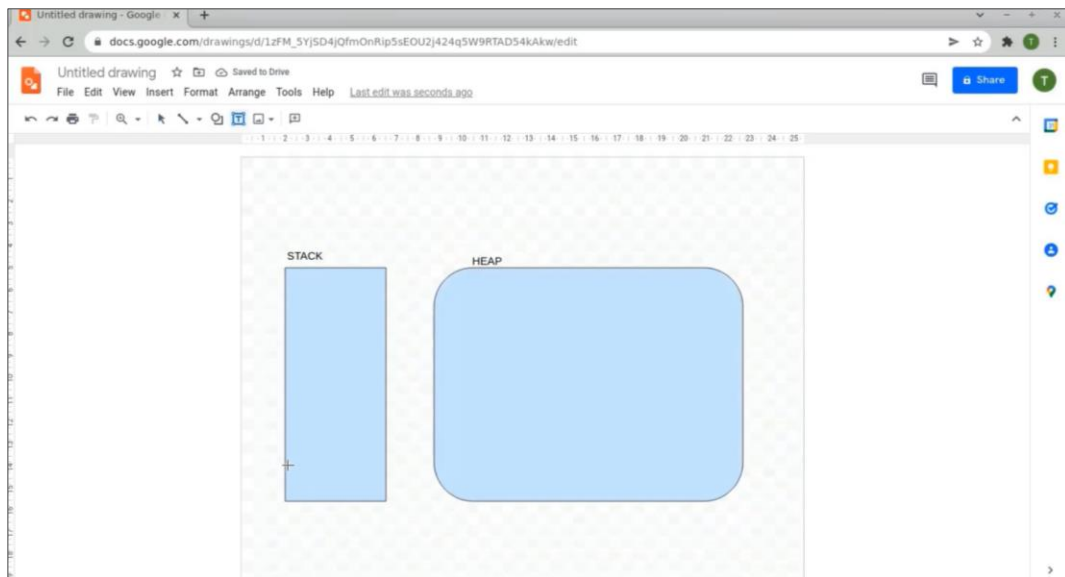
```

Step 3: Use the new operator to create an array inside a heap

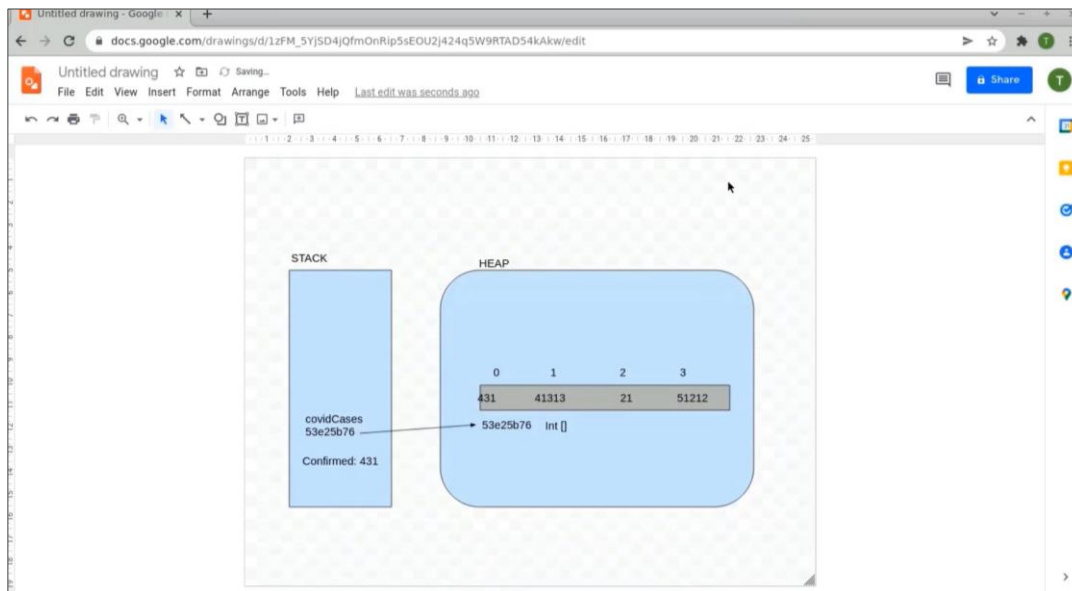
3.1 Open the browser window and open Google draw



3.2 Consider the memory region as the stack and this memory region as the heap:



- 3.3 The confirmed variable with the value 431 is stored in the stack. The array **covidCases** is also stored in the stack region. The hash code for **covidCases** is available, indicating another storage container that stores the actual data, which is the integer array. This array will be indexed and is created at the hash code, linking it to the storage container. The data is available as part of your array and inside this storage container. When you create this array, it is automatically indexed. This is how memory allocation occurs for the array.



- 3.4 New is the operator that creates your array in the heap. This means that if you create an array and assign its data to **COVIDcases**, it is known as a reference copy operation. The hash code of **COVIDcases** is copied into the data

```

1 public class ArrayID {
2     public static void main(String[] args) {
3
4         // Primitive Types: Single Value Containers
5         int confirmed = 431;
6         int recovered = 41313;
7         int deceased = 21;
8         int vaccinated = 54524;
9
10        // Reference Type: Multiple Value Container
11        int covidCases[] = {431, 41313, 21, 54524};
12
13        int array1[] = {10, 20, 30, 40, 50};
14        int[] array2 = {10, 20, 30, 40, 50};
15
16        int a1[], a2;
17        int[] a3, a4;
18
19        int array3[] = new int[] {10, 20, 30, 40, 50};
20        int[] array4 = new int[] {10, 20, 30, 40, 50};
21
22        // All the values will be 0 by default initially
23        int array5[] = new int[5];
24        int[] array6 = new int[10];
25
26        array5[0] = 10;
27        array5[1] = 20;
28
29        System.out.println("confirmed is: "+confirmed);
30        System.out.println("covidCases is: "+covidCases);
31
32        int[] data = covidCases; // Reference Copy Operation
33        // HashCode of covidCases is copied into data
34
35    }
36
37 }

```

3.5 Now print out the data, you will observe that both data and COVID cases refer to the same array. There are just two different names but you get the same array:

```

1 public class ArrayID {
2
3     public static void main(String[] args) {
4
5         // Primitive Types: Single Value Containers
6         int confirmed = 431;
7         int recovered = 41313;
8         int deceased = 21;
9         int vaccinated = 54524;
10
11         // Reference Type: Multiple Value Container
12         int covidCases[] = {431, 41313, 21, 54524};
13
14         int array1[] = {10, 20, 30, 40, 50};
15         int[] array2 = {10, 20, 30, 40, 50};
16
17         int a1[], a2;
18         int[] a3, a4;
19
20         int array3[] = new int[] {10, 20, 30, 40, 50};
21         int[] array4 = new int[] {10, 20, 30, 40, 50};
22
23         // All the values will be 0 by default initially
24         int array5[] = new int[5];
25         int[] array6 = new int[10];
26
27         array5[0] = 10;
28         array5[1] = 20;
29
30         System.out.println("confirmed is: "+confirmed);
31         System.out.println("covidCases is: "+covidCases);
32
33         int[] data = covidCases; // Reference Copy Operation
34         // HashCode of covidCases is copied into data
35         System.out.println("data is: "+data);
36     }
37 }
  
```

```

<terminated> ArrayID [Java Application] /usr/eclipse/plugins/org.eclipse.justi.openjdk.hotspot.jre.full/bin/linux.x86_64/java
confirmed is: 431
covidCases is: [I@5eb5c224
data is: [I@5eb5c224
  
```

3.6 Try to write as the data of one is a number and here you are updating this one which is this index 141313 in the covid cases with the data one and now if you give as print, the covid cases of one and run this code, you will get the value as 66412, with this you can understand the context called reference copy

```

7     int confirmed = 431;
8     int recovered = 41313;
9     int deceased = 21;
10    // int vaccinated = 54524;
11
12    // Reference Type: Multiple Value Container
13    int covidCases[] = {431, 41313, 21, 54524};
14
15    int array1[] = {10, 20, 30, 40, 50};
16    int[] array2 = {10, 20, 30, 40, 50};
17
18    int a1[], a2;
19    int[] a3, a4;
20
21    int array3[] = new int[] {10, 20, 30, 40, 50};
22    int[] array4 = new int[] {10, 20, 30, 40, 50};
23
24    // All the values will be 0 by default initially
25    int array5[] = new int[5];
26    int[] array6 = new int[10];
27
28    array5[0] = 10;
29    array5[1] = 20;
30
31    System.out.println("confirmed is: "+confirmed);
32    System.out.println("covidCases is: "+covidCases);
33
34    int[] data = covidCases; // Reference Copy Operation
35    // HashCode of covidCases is copied into data
36    System.out.println("data is: "+data);
37
38    data[1] = 66412;
39    System.out.println("covidCases[1]: "+covidCases[1]);
40
41 }
  
```

```

<terminated> ArrayID [Java Application] /usr/eclipse/plugins/org.eclipse.justi.openjdk.hotspot.jre.full/bin/linux.x86_64/java
confirmed is: 431
covidCases is: [I@5eb5c224
covidCases[1]: 66412
  
```

3.7 Let us write a loop for integers, with the index value starting at zero. The loop continues while the index is less than the length of **covidCases**, and increments the index by one in each iteration. Inside the loop, write **covidCases[index]** followed by the value of **covidCases[index]**

```

1  // Array1D.java
2  int confirmed = 431;
3  int recovered = 41313;
4  int deceased = 21;
5  // int vaccinated = 54524;
6
7  // Reference Type: Multiple Value Container
8  int covidCases[] = {431, 41313, 21, 54524};
9
10 int array1[] = {10, 20, 30, 40, 50};
11 int[] array2 = {10, 20, 30, 40, 50};
12
13 int a1[], a2;
14 int[] a3, a4;
15
16 int array3[] = new int[] {10, 20, 30, 40, 50};
17 int[] array4 = new int[] {10, 20, 30, 40, 50};
18
19 // All the values will be 0 by default initially
20 int array5[] = new int[5];
21 int[] array6 = new int[10];
22
23 array5[0] = 10;
24 array5[1] = 20;
25
26 System.out.println("confirmed is: "+confirmed);
27 System.out.println("covidCases is: "+covidCases);
28
29 int[] data = covidCases; // Reference Copy Operation
30 // HashCode of covidCases is copied into data
31 System.out.println("data is: "+ data);
32
33 data[1] = 66412;
34 System.out.println("covidCases[1]: "+covidCases[1]);
35
36 for(int idx=0; idx<covidCases.length; idx++) {
37     System.out.println("covidCases["+idx+"] is: "+covidCases[idx]);
38 }

```

3.8 Re-run the program and you will get the data:

```

1  // Array1D.java
2  int confirmed = 431;
3  int recovered = 41313;
4  int deceased = 21;
5  // int vaccinated = 54524;
6
7  // Reference Type: Multiple Value Container
8  int covidCases[] = {431, 41313, 21, 54524};
9
10 int array1[] = {10, 20, 30, 40, 50};
11 int[] array2 = {10, 20, 30, 40, 50};
12
13 int a1[], a2;
14 int[] a3, a4;
15
16 int array3[] = new int[] {10, 20, 30, 40, 50};
17 int[] array4 = new int[] {10, 20, 30, 40, 50};
18
19 // All the values will be 0 by default initially
20 int array5[] = new int[5];
21 int[] array6 = new int[10];
22
23 array5[0] = 10;
24 array5[1] = 20;
25
26 System.out.println("confirmed is: "+confirmed);
27 System.out.println("covidCases is: "+covidCases);
28
29 int[] data = covidCases; // Reference Copy Operation
30 // HashCode of covidCases is copied into data
31 System.out.println("data is: "+ data);
32
33 data[1] = 66412;
34 System.out.println("covidCases[1]: "+covidCases[1]);
35
36 for(int idx=0; idx<covidCases.length; idx++) {
37     System.out.println("covidCases["+idx+"] is: "+covidCases[idx]);
38 }

```

```

<terminated> Array1D [Java Application] /usr/eclipse/plugins/org.eclipse.justi.openjdk.hotspot
confirmed is: 431
covidCases is: [I@5eb5c224
data is: [I@5eb5c224
covidCases[1]: 66412
covidCases[0] is: 431
covidCases[1] is: 66412
covidCases[2] is: 21
covidCases[3] is: 54524

```

Step 4: Access elements in an array

4.1 Let us give an empty print line and then again, a print iterating in 1D array and run the program

```

14
15 int array1[] = {10, 20, 30, 40, 50};
16 int[] array2 = {10, 20, 30, 40, 50};
17
18 int a1[], a2;
19 int[] a3, a4;
20
21 int array3[] = new int[] {10, 20, 30, 40, 50};
22 int[] array4 = new int[] {10, 20, 30, 40, 50};
23
24 // All the values will be 0 by default initially
25 int array5[] = new int[5];
26 int[] array6 = new int[10];
27
28 array5[0] = 10;
29 array5[1] = 20;
30
31 System.out.println("confirmed is: "+confirmed);
32 System.out.println("covidCases is: "+covidCases);
33
34 int[] data = covidCases; // Reference Copy Operation
35 // Hashcode of covidCases is copied into data
36 System.out.println("data is: "+ data);
37
38 data[1] = 66412;
39 System.out.println("covidCases[1]: "+covidCases[1]);
40 System.out.println();
41
42 System.out.println("Iterating in 1D Array");
43 for(int idx=0;idx<covidCases.length;idx++) {
44     System.out.println("covidCases["+idx+"] is: "+covidCases[idx]);
45 }
46
47
48
49
50 }
51

```

```

-terminated- Array1D [Java Application] /usr/eclipse/plugins/org.eclipse.justi.openjdk.hotspot...
confirmed is: 431
covidCases is: [105eb5c224
data is: [105eb5c224
covidCases[1]: 66412

Iterating in 1D Array
covidCases[0] is: 431
covidCases[1] is: 66412
covidCases[2] is: 21
covidCases[3] is: 54524

```

4.2 Let us write an iteration using an enhanced for loop. Here, you do not have to work with indexes; it is a read-only loop. Therefore, you can write it as for (**int element : covidCases**). The variable element can be any name you choose

```

19 int[] a3, a4;
20
21 int array3[] = new int[] {10, 20, 30, 40, 50};
22 int[] array4 = new int[] {10, 20, 30, 40, 50};
23
24 // All the values will be 0 by default initially
25 int array5[] = new int[5];
26 int[] array6 = new int[10];
27
28 array5[0] = 10;
29 array5[1] = 20;
30
31 System.out.println("confirmed is: "+confirmed);
32 System.out.println("covidCases is: "+covidCases);
33
34 int[] data = covidCases; // Reference Copy Operation
35 // Hashcode of covidCases is copied into data
36 System.out.println("data is: "+ data);
37
38 data[1] = 66412;
39 System.out.println("covidCases[1]: "+covidCases[1]);
40 System.out.println();
41
42 System.out.println("Iterating in 1D Array");
43 for(int idx=0;idx<covidCases.length;idx++) {
44     System.out.println("covidCases["+idx+"] is: "+covidCases[idx]);
45 }
46
47 System.out.println("Iterating with Enhanced for Loop");
48 for(int element: covidCases) {
49     System.out.println("Element is: "+element);
50 }
51
52
53
54 }
55
56

```

4.3 Execute this code, you will get the same output, which means it is just that the indexes are not in our hands and you have direct access to the data or the value or the elements

```

19  int[] a3, a4;
20
21  int array3[] = new int[] {10, 20, 30, 40, 50};
22  int[] array4 = new int[] {10, 20, 30, 40, 50};
23
24  // All the values will be 0 by default initially
25  int array5[] = new int[5];
26  int[] array6 = new int[10];
27
28  array5[0] = 10;
29  array5[1] = 20;
30
31  System.out.println("confirmed is: "+confirmed);
32  System.out.println("covidCases is: "+covidCases);
33
34  int[] data = covidCases; // Reference Copy Operation
35  // HashCode of covidCases is copied into data
36  System.out.println("data is: "+ data);
37
38  data[1] = 66412;
39  System.out.println("covidCases[1]: "+covidCases[1]);
40  System.out.println();
41
42  System.out.println("Iterating in 1D Array");
43  for(int idx=0;idx<covidCases.length;idx++) {
44      System.out.println("covidCases["+idx+"] is: "+covidCases[idx]);
45  }
46
47  System.out.println("Iterating with Enhanced for Loop");
48  for(int element: covidCases) {
49      System.out.println("Element is: "+element);
50  }

```

```

<terminated> Array1D [Java Application] /usr/eclipse/plugins/org.eclipse.justi.openjdk.hotspot.jre.full.j2se-17.0.2-80-linux-x86_64/bin/java
confirmed is: 431
covidCases is: [I@5eb5c224
data is:[I@5eb5c224
covidCases[1]: 66412

Iterating in 1D Array
covidCases[0] is: 431
covidCases[1] is: 66412
covidCases[2] is: 21
covidCases[3] is: 54524
Iterating with Enhanced for Loop
Element is: 431
Element is: 66412
Element is: 21
Element is: 54524

```

4.4 Instead of covidCases, write here as array five and run this code, then what you get is 0 and 20 and 0 and 0

```

19  int[] a3, a4;
20
21  int array3[] = new int[] {10, 20, 30, 40, 50};
22  int[] array4 = new int[] {10, 20, 30, 40, 50};
23
24  // All the values will be 0 by default initially
25  int array5[] = new int[5];
26  int[] array6 = new int[10];
27
28  array5[0] = 10;
29  array5[1] = 20;
30
31  System.out.println("confirmed is: "+confirmed);
32  System.out.println("covidCases is: "+covidCases);
33
34  int[] data = covidCases; // Reference Copy Operation
35  // HashCode of covidCases is copied into data
36  System.out.println("data is: "+ data);
37
38  data[1] = 66412;
39  System.out.println("covidCases[1]: "+covidCases[1]);
40  System.out.println();
41
42  System.out.println("Iterating in 1D Array");
43  for(int idx=0;idx<covidCases.length;idx++) {
44      System.out.println("covidCases["+idx+"] is: "+covidCases[idx]);
45  }
46
47  System.out.println("Iterating with Enhanced for Loop");
48  for(int element: array5) {
49      System.out.println("Element is: "+element);
50  }

```

```

<terminated> Array1D [Java Application] /usr/eclipse/plugins/org.eclipse.justi.openjdk.hotspot.jre.full.j2se-17.0.2-80-linux-x86_64/bin/java
confirmed is: 431
covidCases is: [I@5eb5c224
data is:[I@5eb5c224
covidCases[1]: 66412

Iterating in 1D Array
covidCases[0] is: 431
covidCases[1] is: 66412
covidCases[2] is: 21
covidCases[3] is: 54524
Iterating with Enhanced for Loop
Element is: 0
Element is: 20
Element is: 0
Element is: 0

```

4.5 Go back and uncomment this section where **array5[0]** was **10** and **array5[1]** was **20**, with the rest of the elements being **0**. When you create an array with a specific size, all the elements are initialized to **0** by default

The screenshot shows the Eclipse IDE with a Java file named `Array1D.java`. The code defines several integer arrays and prints their contents. The console output shows the execution results, including the values of `covidCases` and `data` arrays, and the iteration of the `array5` array.

```

18 int a1[], a2;
19 int a3, a4;
20
21 int array3[] = new int[] {10, 20, 30, 40, 50};
22 int[] array4 = new int[] {10, 20, 30, 40, 50};
23
24 // All the values will be 0 by default initially
25 int array5[] = new int[5];
26 int[] array6 = new int[10];
27
28 array5[0] = 10;
29 array5[1] = 20;
30
31 System.out.println("confirmed is: "+confirmed);
32 System.out.println("covidCases is: "+covidCases);
33
34 int[] data = covidCases; // Reference Copy Operation
35 // hashCode of covidCases is copied into data
36 System.out.println("data is: "+data);
37
38 data[1] = 66412;
39 System.out.println("covidCases[1]: "+covidCases[1]);
40 System.out.println();
41
42 System.out.println("Iterating in 1D Array");
43 for(int idx=0;idx<covidCases.length;idx++) {
44     System.out.println("covidCases["+idx+"] is: "+covidCases[idx]);
45 }
46
47 System.out.println("Iterating with Enhanced for Loop");
48 for(int element: array5) {

```

Console Output:

```

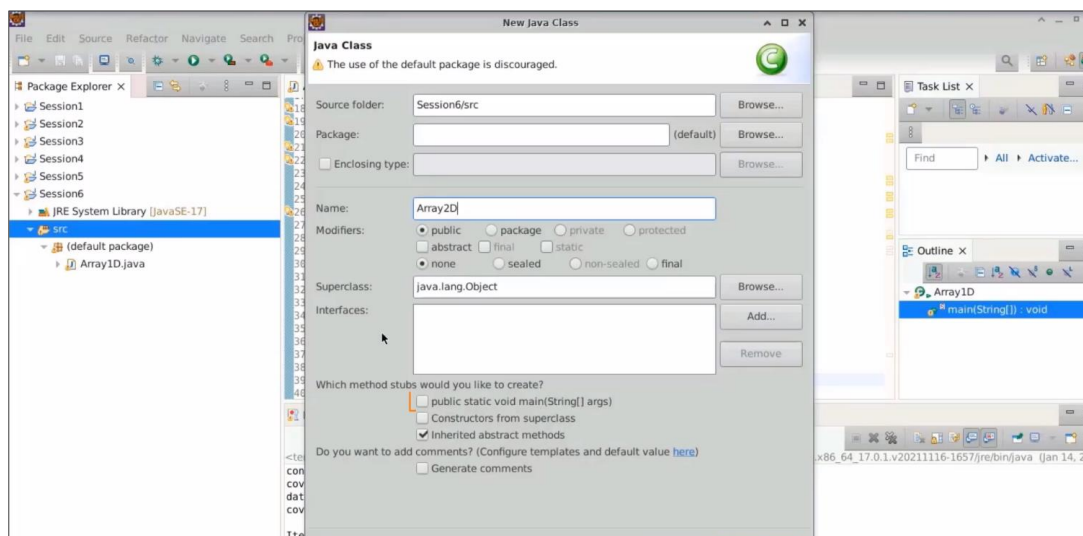
<terminated> Array1D [Java Application] /usr/eclipse/plugins/org.eclipse.justi.openjdk.hotspot...
confirmed is: 431
covidCases is: [I@5eb5c224
data is: [I@5eb5c224
covidCases[1]: 66412

Iterating in 1D Array
covidCases[0] is: 431
covidCases[1] is: 66412
covidCases[2] is: 21
covidCases[3] is: 54524
Iterating with Enhanced for Loop
Element is: 10
Element is: 20
Element is: 0
Element is: 0
Element is: 0

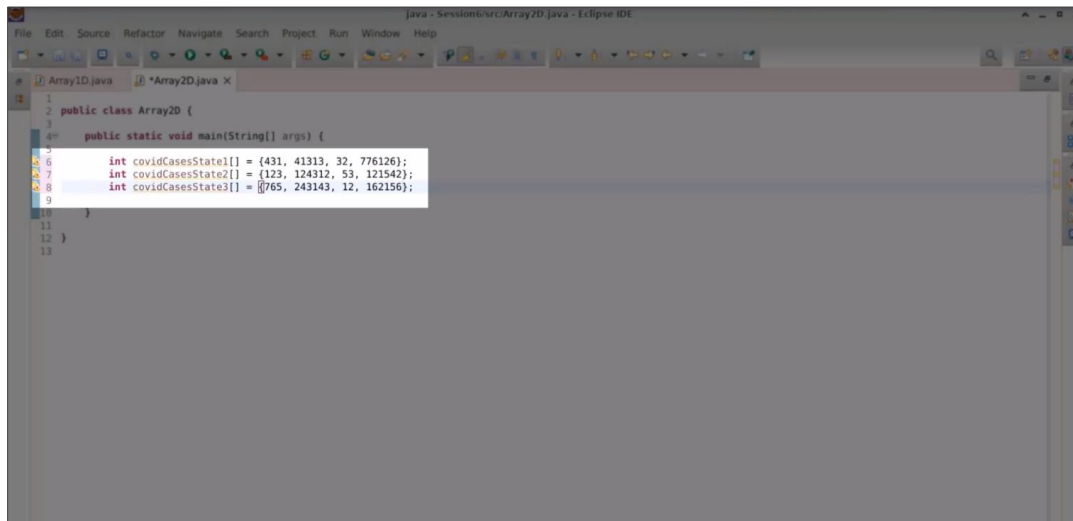
```

Step 5: Create an array with a specific size

5.1 Create another program called **Array2D**, with the main method

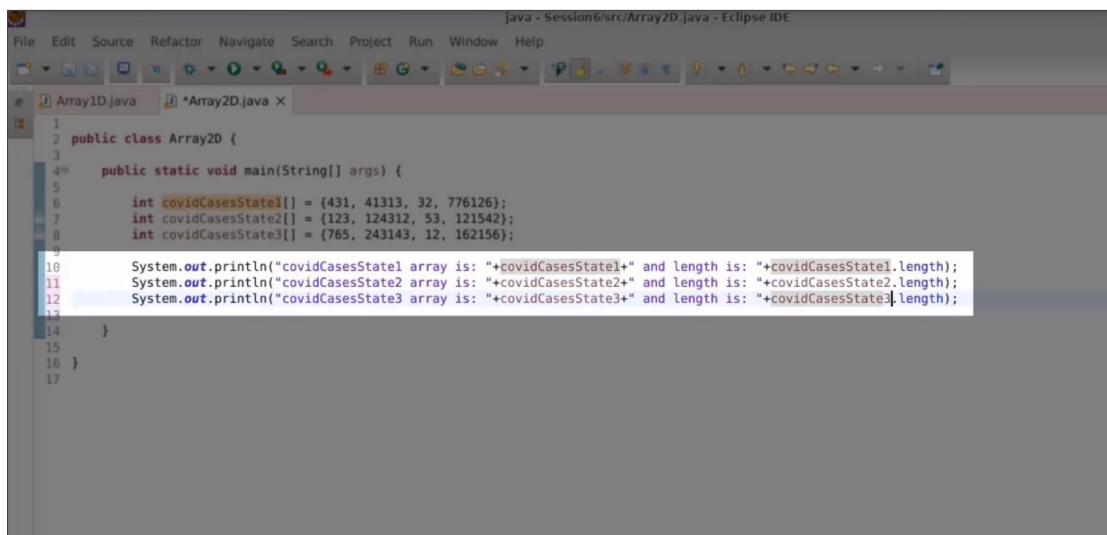


5.2 Write covidCases for three different states, with different numbers



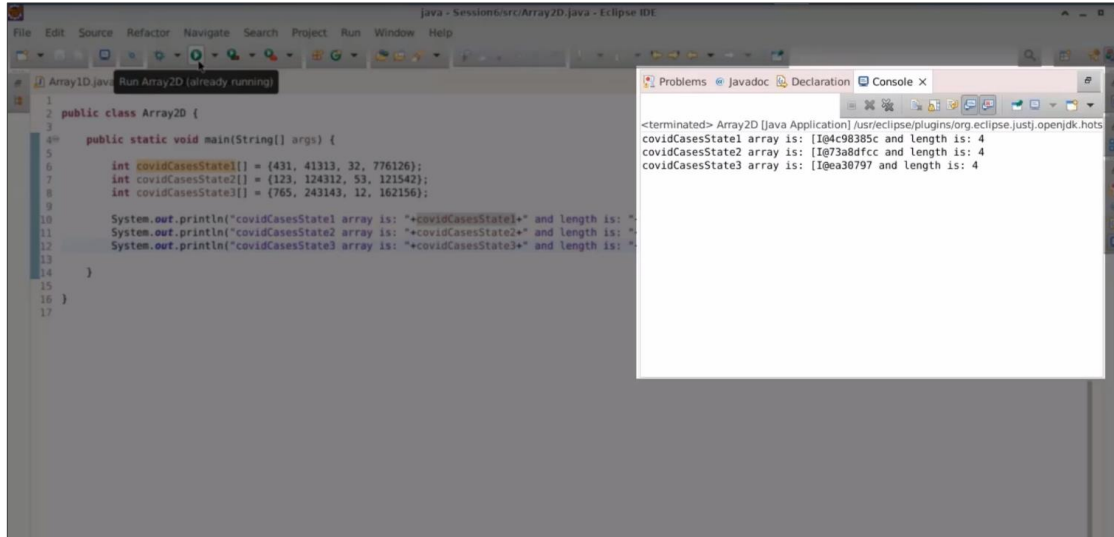
```
1 public class Array2D {
2
3     public static void main(String[] args) {
4
5         int covidCasesState1[] = {431, 41313, 32, 776126};
6         int covidCasesState2[] = {123, 124312, 53, 121542};
7         int covidCasesState3[] = {765, 243143, 12, 162156};
8
9     }
10 }
11
12
13 }
```

5.3 Write covidCases state1 array is followed by the covidCases state1 array, and length is followed by covidCases state1.length. Similarly, print the other two arrays, changing the index to 2 and 3 accordingly



```
1 public class Array2D {
2
3     public static void main(String[] args) {
4
5         int covidCasesState1[] = {431, 41313, 32, 776126};
6         int covidCasesState2[] = {123, 124312, 53, 121542};
7         int covidCasesState3[] = {765, 243143, 12, 162156};
8
9         System.out.println("covidCasesState1 array is: "+covidCasesState1+" and length is: "+covidCasesState1.length);
10        System.out.println("covidCasesState2 array is: "+covidCasesState2+" and length is: "+covidCasesState2.length);
11        System.out.println("covidCasesState3 array is: "+covidCasesState3+" and length is: "+covidCasesState3.length);
12
13    }
14 }
15
16
17 }
```

5.4 Print all three arrays, and printing the arrays will give the hash codes, and as shown three different hash codes come in as they are multi-value containers and the length is four



```

1 public class Array2D {
2
3     public static void main(String[] args) {
4
5         int covidCasesState1[] = {431, 41313, 32, 776126};
6         int covidCasesState2[] = {123, 124312, 53, 121542};
7         int covidCasesState3[] = {765, 243143, 12, 162156};
8
9         System.out.println("covidCasesState1 array is: "+covidCasesState1+" and length is: "+covidCasesState1.length);
10        System.out.println("covidCasesState2 array is: "+covidCasesState2+" and length is: "+covidCasesState2.length);
11        System.out.println("covidCasesState3 array is: "+covidCasesState3+" and length is: "+covidCasesState3.length);
12    }
13 }
14
15
16
17

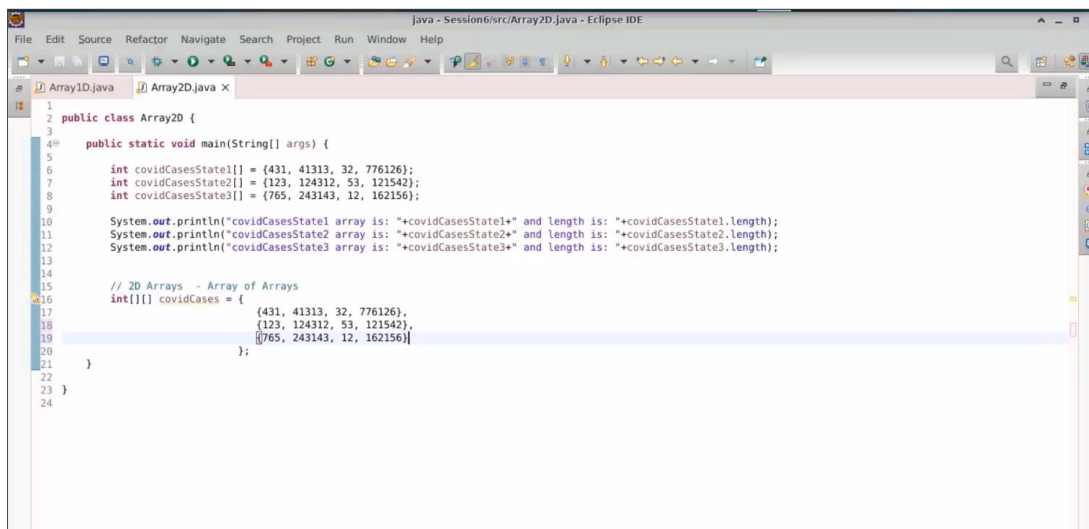
```

```

<terminated> Array2D [Java Application] /usr/eclipse/plugins/org.eclipse.justi.openjdk.hotspot.jre.full/bin/linux.x86_64/java
covidCasesState1 array is: [I@4c98385c and length is: 4
covidCasesState2 array is: [I@73a8dfcc and length is: 4
covidCasesState3 array is: [I@ea30797 and length is: 4

```

5.5 Then, the next step is to create a 2D array and for this, you need to use two brackets either before or after and name this as **covidCases** this is a 1D array and inside this 1D array you will place these three arrays and hence, it will make this structure as 2D (array of the array).



```

1 public class Array2D {
2
3     public static void main(String[] args) {
4
5         int covidCasesState1[] = {431, 41313, 32, 776126};
6         int covidCasesState2[] = {123, 124312, 53, 121542};
7         int covidCasesState3[] = {765, 243143, 12, 162156};
8
9         System.out.println("covidCasesState1 array is: "+covidCasesState1+" and length is: "+covidCasesState1.length);
10        System.out.println("covidCasesState2 array is: "+covidCasesState2+" and length is: "+covidCasesState2.length);
11        System.out.println("covidCasesState3 array is: "+covidCasesState3+" and length is: "+covidCasesState3.length);
12
13        // 2D Arrays - Array of Arrays
14        int[][] covidCases = {
15            {431, 41313, 32, 776126},
16            {123, 124312, 53, 121542},
17            {765, 243143, 12, 162156}
18        };
19
20    }
21 }
22
23
24

```

5.6 If you want to create a 2D array as a new integer then you need to mention the number of arrays, that are going to be inside your array, let us assume it as five arrays, with three elements in each array. It means you are going to create five 1D arrays with three elements

```

1 public class Array2D {
2
3     public static void main(String[] args) {
4
5         int covidCasesState1[] = {431, 41313, 32, 776126};
6         int covidCasesState2[] = {123, 124312, 53, 121542};
7         int covidCasesState3[] = {765, 243143, 12, 162156};
8
9         System.out.println("covidCasesState1 array is: "+covidCasesState1+" and length is: "+covidCasesState1.length);
10        System.out.println("covidCasesState2 array is: "+covidCasesState2+" and length is: "+covidCasesState2.length);
11        System.out.println("covidCasesState3 array is: "+covidCasesState3+" and length is: "+covidCasesState3.length);
12
13        // 2D Arrays - Array of Arrays
14        int[][] covidCases = {
15            {431, 41313, 32, 776126},
16            {123, 124312, 53, 121542},
17            {765, 243143, 12, 162156}
18        };
19
20        int[][] array2D = new int[5][3]; // 5 1-D Arrays with 3 elements each
21    }
22 }

```

5.7 It is like five 1D arrays, but the elements are not decided yet. For that, you will write in the array 2D, the zeroth element will be an array of three integers, and same way in the array 2D, the third index can be for a new integer which can be 10, and same way you can have other indexes too

```

1 public class Array2D {
2
3     public static void main(String[] args) {
4
5         int covidCasesState1[] = {431, 41313, 32, 776126};
6         int covidCasesState2[] = {123, 124312, 53, 121542};
7         int covidCasesState3[] = {765, 243143, 12, 162156};
8
9         System.out.println("covidCasesState1 array is: "+covidCasesState1+" and length is: "+covidCasesState1.length);
10        System.out.println("covidCasesState2 array is: "+covidCasesState2+" and length is: "+covidCasesState2.length);
11        System.out.println("covidCasesState3 array is: "+covidCasesState3+" and length is: "+covidCasesState3.length);
12
13        // 2D Arrays - Array of Arrays
14        int[][] covidCases = {
15            {431, 41313, 32, 776126},
16            {123, 124312, 53, 121542},
17            {765, 243143, 12, 162156}
18        };
19
20        //int[][] array2D = new int[5][3]; // 5 1-D Arrays with 3 elements each
21        int[][] array2D = new int[5][1]; // 5 1-D Arrays but elements are not decided yet
22        array2D[0] = new int[3];
23        array2D[3] = new int[10];
24    }
25 }

```

5.8 To iterate through a 2D array, you will need a nested for loop. First, write an outer loop as **for (int i = 0; i < covidCases.length; i++)**

```

1 public class Array2D {
2
3     public static void main(String[] args) {
4
5         int covidCasesState1[] = {431, 41313, 32, 776126};
6         int covidCasesState2[] = {123, 124312, 53, 121542};
7         int covidCasesState3[] = {765, 243143, 12, 162156};
8
9         System.out.println("covidCasesState1 array is: "+covidCasesState1+" and length is: "+covidCasesState1.length);
10        System.out.println("covidCasesState2 array is: "+covidCasesState2+" and length is: "+covidCasesState2.length);
11        System.out.println("covidCasesState3 array is: "+covidCasesState3+" and length is: "+covidCasesState3.length);
12
13
14        // 2D Arrays - Array of Arrays
15        int[][] covidCases = {
16            {431, 41313, 32, 776126},
17            {123, 124312, 53, 121542},
18            {765, 243143, 12, 162156}
19        };
20
21        //int[][] array2D = new int[5][3]; // 5 1-D Arrays with 3 elements each
22        int[][] array2D = new int[5][1]; // 5 1-D Arrays but elements are not decided yet
23        array2D[0] = new int[3];
24        array2D[3] = new int[10];
25
26
27        for(int i=0; i<covidCases.length; i++) {
28
29        }
30    }
31
32 }
33

```

5.9 Here, the length of **covidCases** is 3, which means that if you print something similar to **covidCases** here and refer to it as covid cases array, the length would be represented by **covidCases.length**

```

1 public class Array2D {
2
3     public static void main(String[] args) {
4
5         int covidCasesState1[] = {431, 41313, 32, 776126};
6         int covidCasesState2[] = {123, 124312, 53, 121542};
7         int covidCasesState3[] = {765, 243143, 12, 162156};
8
9         System.out.println("covidCasesState1 array is: "+covidCasesState1+" and length is: "+covidCasesState1.length);
10        System.out.println("covidCasesState2 array is: "+covidCasesState2+" and length is: "+covidCasesState2.length);
11        System.out.println("covidCasesState3 array is: "+covidCasesState3+" and length is: "+covidCasesState3.length);
12
13
14        // 2D Arrays - Array of Arrays
15        int[][] covidCases = {
16            {431, 41313, 32, 776126},
17            {123, 124312, 53, 121542},
18            {765, 243143, 12, 162156}
19        };
20
21        //int[][] array2D = new int[5][3]; // 5 1-D Arrays with 3 elements each
22        System.out.println("covidCases array is: "+covidCases+" and length is: "+covidCases.length);
23        int[][] array2D = new int[5][1]; // 5 1-D Arrays but elements are not decided yet
24        array2D[0] = new int[3];
25        array2D[3] = new int[10];
26
27
28        for(int i=0; i<covidCases.length; i++) {
29
30        }
31
32 }
33
34

```

5.10 Run this code. It will show that the **covidCases** array will have two brackets in front, indicating a 2D array. Previously, there was only one bracket in front. The output will show the hash code and the length as three, indicating that there are three arrays inside this array

```

1 public class Array2D {
2
3     public static void main(String[] args) {
4
5         int covidCasesState1[] = {431, 41313, 32, 776126};
6         int covidCasesState2[] = {123, 124312, 53, 121542};
7         int covidCasesState3[] = {765, 243143, 12, 162156};
8
9         System.out.println("covidCasesState1 array is: "+covidCasesState1+" and length is: "+covidCasesState1.length);
10        System.out.println("covidCasesState2 array is: "+covidCasesState2+" and length is: "+covidCasesState2.length);
11        System.out.println("covidCasesState3 array is: "+covidCasesState3+" and length is: "+covidCasesState3.length);
12
13        // 2D Arrays - Array of Arrays
14        int[][] covidCases = {
15            {431, 41313, 32, 776126},
16            {123, 124312, 53, 121542},
17            {765, 243143, 12, 162156}
18        };
19
20        System.out.println("covidCases array is: "+covidCases+" and length is: "+covidCases.length);
21
22        //int[][] array2D = new int[5][3]; // 5 1-D Arrays with 3 elements each
23        int[][] array2D = new int[5][1]; // 5 1-D Arrays but elements are not decided yet
24        array2D[0] = new int[3];
25        array2D[3] = new int[10];
26
27        for(int i=0;i<covidCases.length;i++) {
28
29        }
30    }
31 }
32
33
34 }

```

```

<terminated> Array2D [Java Application] /usr/eclipse/plugins/org.eclipse.justi.openjdk.hotspot.jre.full/bin/linux.x86_64/java
covidCasesState1 array is: [I@4cdf35a9 and length is: 4
covidCasesState2 array is: [I@53e25b76 and length is: 4
covidCasesState3 array is: [I@73a8dfcc and length is: 4
covidCases array is: [[I@ea30797 and length is: 3

```

Step 6: Implement a two-dimensional array with suitable examples

6.1 Now, write another for loop inside, with **j** starting at zero. The condition is **j < covidCases[i].length**, which means **covidCases[0].length**, **covidCases[1].length**, and **covidCases[2].length**. This means it will always be four, as the arrays are aligned with the same length

```

1 public class Array2D {
2
3     public static void main(String[] args) {
4
5         int covidCasesState1[] = {431, 41313, 32, 776126};
6         int covidCasesState2[] = {123, 124312, 53, 121542};
7         int covidCasesState3[] = {765, 243143, 12, 162156};
8
9         System.out.println("covidCasesState1 array is: "+covidCasesState1+" and length is: "+covidCasesState1.length);
10        System.out.println("covidCasesState2 array is: "+covidCasesState2+" and length is: "+covidCasesState2.length);
11        System.out.println("covidCasesState3 array is: "+covidCasesState3+" and length is: "+covidCasesState3.length);
12
13        // 2D Arrays - Array of Arrays
14        int[][] covidCases = {
15            {431, 41313, 32, 776126}, // 0
16            {123, 124312, 53, 121542}, // 1
17            {765, 243143, 12, 162156} // 2
18        };
19
20        System.out.println("covidCases array is: "+covidCases+" and length is: "+covidCases.length);
21
22        //int[][] array2D = new int[5][3]; // 5 1-D Arrays with 3 elements each
23        int[][] array2D = new int[5][1]; // 5 1-D Arrays but elements are not decided yet
24        array2D[0] = new int[3];
25        array2D[3] = new int[10];
26
27        for(int i=0;i<covidCases.length;i++) { // 0-3 times
28            for(int j=0;j<covidCases[i].length;j++) {
29
30            }
31        }
32
33
34 }

```

6.2 Next, let us print the **covidCases[i][j]** element, followed by a space, and continue printing on the same line. After printing one of the arrays, insert an empty print line. Then run the program.

```

1 public class Array2D {
2
3     public static void main(String[] args) {
4
5         int covidCasesState1[] = {431, 41313, 32, 776126};
6         int covidCasesState2[] = {123, 124312, 53, 121542};
7         int covidCasesState3[] = {765, 243143, 12, 162156};
8
9         System.out.println("covidCasesState1 array is: "+covidCasesState1+" and length is: "+covidCasesState1.length);
10        System.out.println("covidCasesState2 array is: "+covidCasesState2+" and length is: "+covidCasesState2.length);
11        System.out.println("covidCasesState3 array is: "+covidCasesState3+" and length is: "+covidCasesState3.length);
12
13        // 2D Arrays - Array of Arrays
14        int[][] covidCases = {
15            //
16            {431, 41313, 32, 776126}, // 0
17            {123, 124312, 53, 121542}, // 1
18            {765, 243143, 12, 162156} // 2
19        };
20
21        System.out.println("covidCases array is: "+covidCases+" and length is: "+covidCases.length);
22
23        //int[][] array2D = new int[5][3]; // 5 1-D Arrays with 3 elements each
24        int[][] array2D = new int[5][]; // 5 1-D Arrays but elements are not decided yet
25        array2D[0] = new int[3];
26        array2D[3] = new int[10];
27
28        for(int i=0;i<covidCases.length;i++) { // 3 times
29            for(int j=0;j<covidCases[i].length;j++) {
30                System.out.print(covidCases[i][j]+" ");
31            }
32            System.out.println();
33        }
34    }
35 }

```

Console Output:

```

<terminated> Array2D [Java Application] /usr/eclipse/plugins/org.eclipse.justi.openjdk.hotspot.jre.full/bin/linux.x86_64/eclipse
covidCasesState1 array is: [I@4cdf35a9 and length is: 4
covidCasesState2 array is: [I@53e25b76 and length is: 4
covidCasesState3 array is: [I@73a8dfcc and length is: 4
covidCases array is: [[I@ea30797 and length is: 3
431 41313 32 776126
123 124312 53 121542
765 243143 12 162156

```

6.3 Next, let's write another for loop which can be used as an enhanced version of the loop. Hence, whenever you will iterate inside the array, you will get one array and it can be any name. Iterating in covidCases, you will get one integer array, and thereafter write one more loop where you will get the element inside the array. Write as iterating with Enhanced for loop and this is iterating with basic for loop

```

1 public class Array2D {
2
3     public static void main(String[] args) {
4
5         int covidCasesState1[] = {431, 41313, 32, 776126};
6         int covidCasesState2[] = {123, 124312, 53, 121542};
7         int covidCasesState3[] = {765, 243143, 12, 162156};
8
9         System.out.println("covidCasesState1 array is: "+covidCasesState1+" and length is: "+covidCasesState1.length);
10        System.out.println("covidCasesState2 array is: "+covidCasesState2+" and length is: "+covidCasesState2.length);
11        System.out.println("covidCasesState3 array is: "+covidCasesState3+" and length is: "+covidCasesState3.length);
12
13        // 2D Arrays - Array of Arrays
14        int[][] covidCases = {
15            //
16            {431, 41313, 32, 776126}, // 0
17            {123, 124312, 53, 121542}, // 1
18            {765, 243143, 12, 162156} // 2
19        };
20
21        System.out.println("covidCases array is: "+covidCases+" and length is: "+covidCases.length);
22
23        //int[][] array2D = new int[5][3]; // 5 1-D Arrays with 3 elements each
24        int[][] array2D = new int[5][]; // 5 1-D Arrays but elements are not decided yet
25        array2D[0] = new int[3];
26        array2D[3] = new int[10];
27
28        System.out.println("Iterating with For Loop");
29        for(int i=0;i<covidCases.length;i++) { // 3 times
30            for(int j=0;j<covidCases[i].length;j++) {
31                System.out.print(covidCases[i][j]+" ");
32            }
33            System.out.println();
34        }
35
36        System.out.println("Iterating with Enhanced For Loop");
37        for(int[] array: covidCases) {
38            for(int element: array) {
39                System.out.print(element+" ");
40            }
41            System.out.println();
42        }
43    }
44 }

```

6.4 Run it and you will get the same output coming in, it is just the way of using your loops which is different but output is going to be exactly the same

```

11 System.out.println("covidCasesState2 array is: "+covidCasesState2+" and length is: ");
12 System.out.println("covidCasesState3 array is: "+covidCasesState3+" and length is: ");
13
14 // 2D Arrays - Array of Arrays
15 int[][] covidCases = {
16     //
17     {431, 41313, 32, 776126}, // 0
18     {123, 124312, 53, 121542}, // 1
19     {765, 243143, 12, 162156} // 2
20 };
21
22 System.out.println("covidCases array is: "+covidCases+" and length is: "+covidCases.length);
23
24 //int[][] array2D = new int[5][3]; // 5 1-D Arrays with 3 elements each
25 int[][] array2D = new int[5][1]; // 5 1-D Arrays but elements are not decided yet
26 array2D[0] = new int[3];
27 array2D[3] = new int[10];
28
29 System.out.println("Iterating with For Loop");
30 for(int i=0; i<covidCases.length; i++) { // 3 times
31     for(int j=0; j<covidCases[i].length; j++) {
32         System.out.print(covidCases[i][j]+" ");
33     }
34     System.out.println();
35 }
36
37 System.out.println("Iterating with Enhanced For Loop");
38 for(int[] array: covidCases) {
39     for(int element: array) {
40         System.out.print(element+" ");
41     }
42     System.out.println();
43 }
44 }

```

```

<terminated> Array2D [Java Application] /usr/clipse/plugins/org.eclipse.justi.openjdk.hotspot.jre.full/bin/linux64/eclipse/
covidCasesState1 array is: [I@4cdf35a9 and length is: 4
covidCasesState2 array is: [I@53e25b76 and length is: 4
covidCasesState3 array is: [I@73a8dfcc and length is: 4
covidCases array is: [[I@ea38797 and length is: 3
Iterating with For Loop
431 41313 32 776126
123 124312 53 121542
765 243143 12 162156
Iterating with Enhanced For Loop
431 41313 32 776126
123 124312 53 121542
765 243143 12 162156

```

By following the above steps, you can successfully create and use one-dimensional and two-dimensional arrays in Java.