

Lesson 02 Demo 03

Using Methods in Java

Objective: To depict how methods are implemented in Java

Tools required: Eclipse IDE

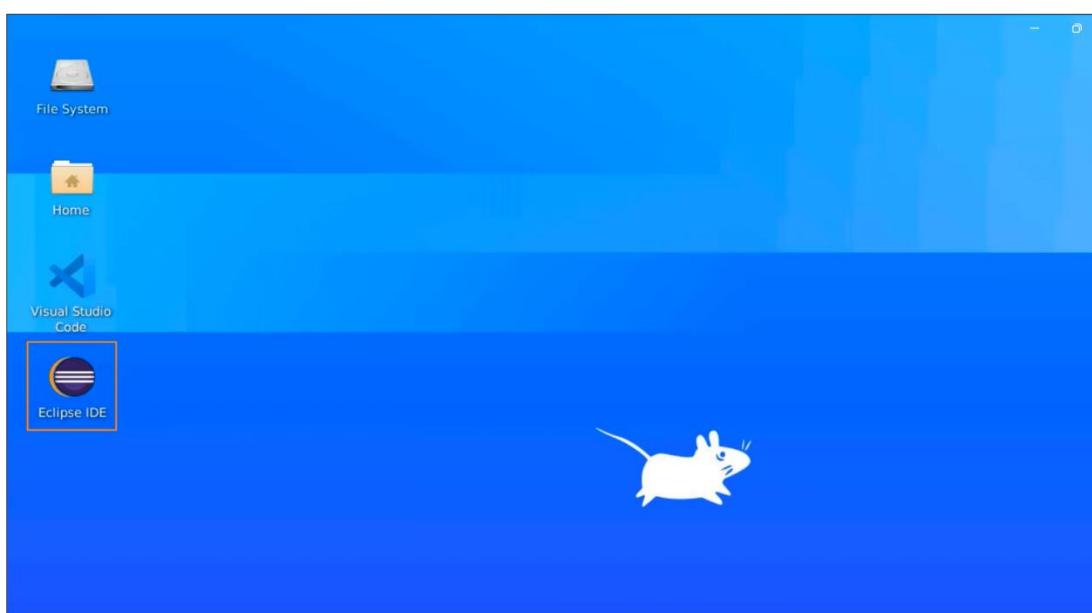
Prerequisites: None

Steps to be followed:

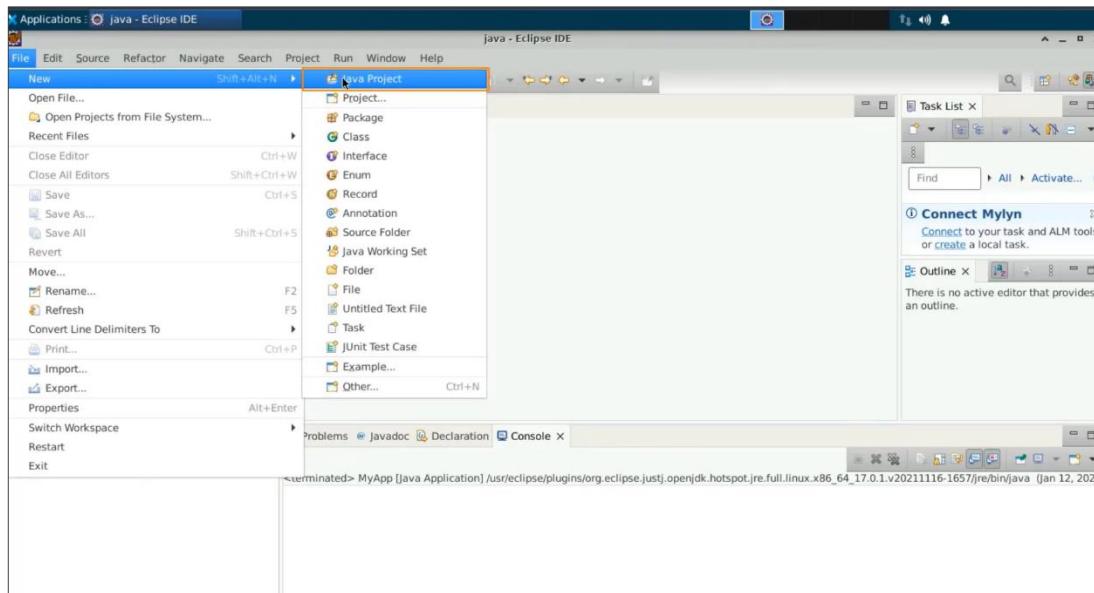
1. Write an algorithm and implement the same
2. Write arrays with cited examples
3. Run the code and get the output
4. Create and use a non-static method
5. Create an object with an object construction statement
6. Differentiate between running a static and a non-static method

Step 1: Write an algorithm and implement the same

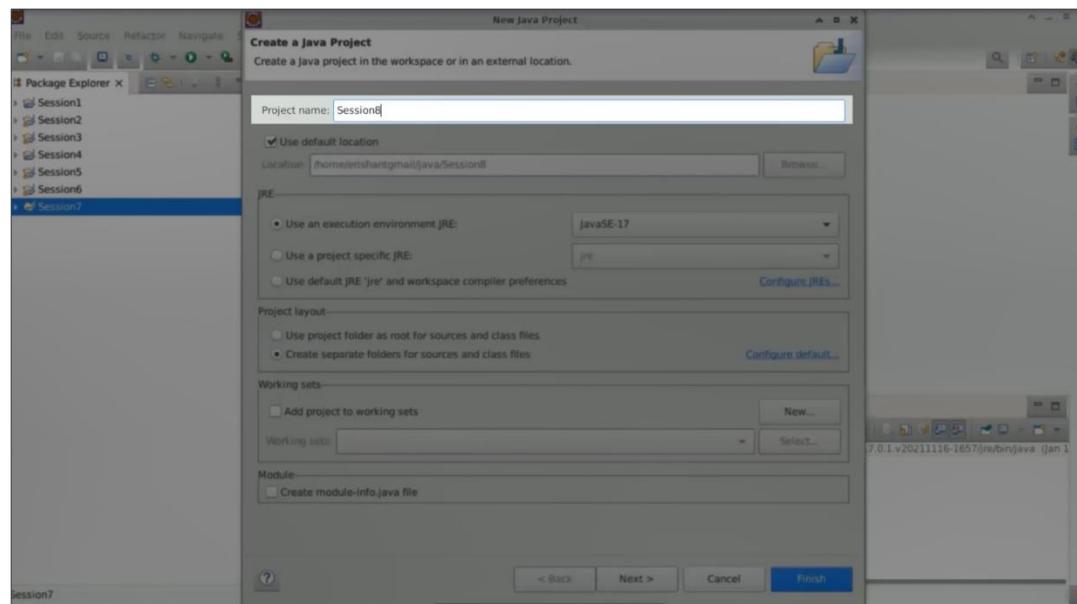
1.1 Open the Eclipse IDE



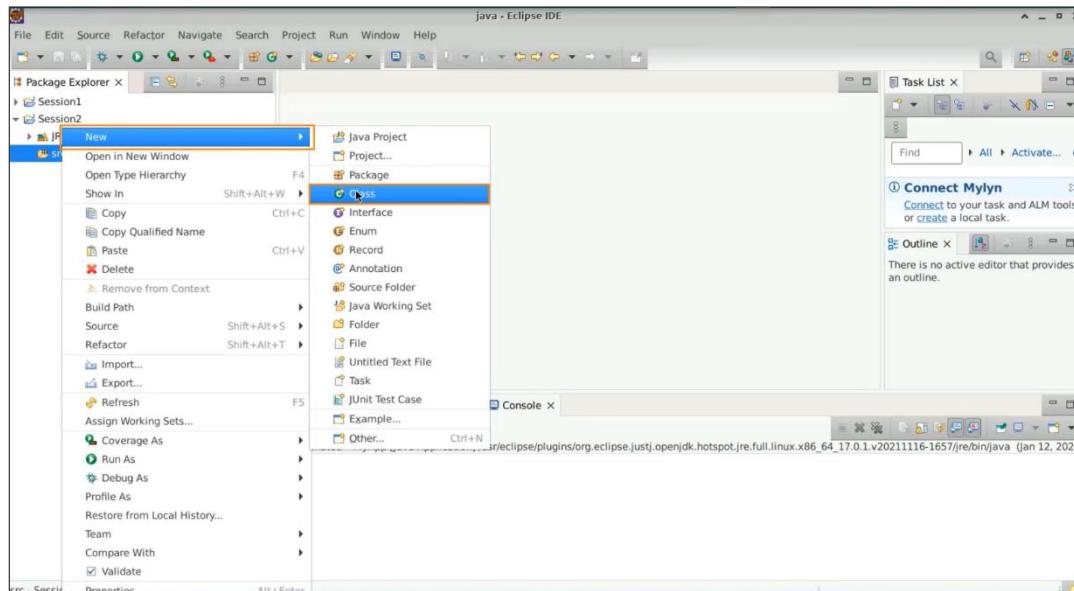
1.2 Select File, then New, and then Java project



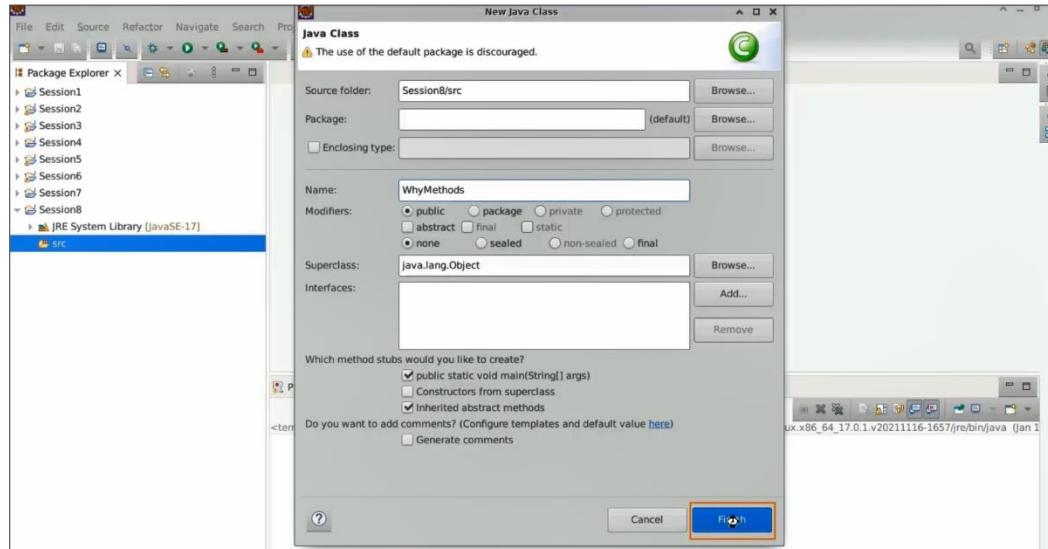
1.3 Name the project “Session8”, uncheck “Create a module info dot Java file”, and press Finish



1.4 With a Session8 on the src, do a right-click and create a new class

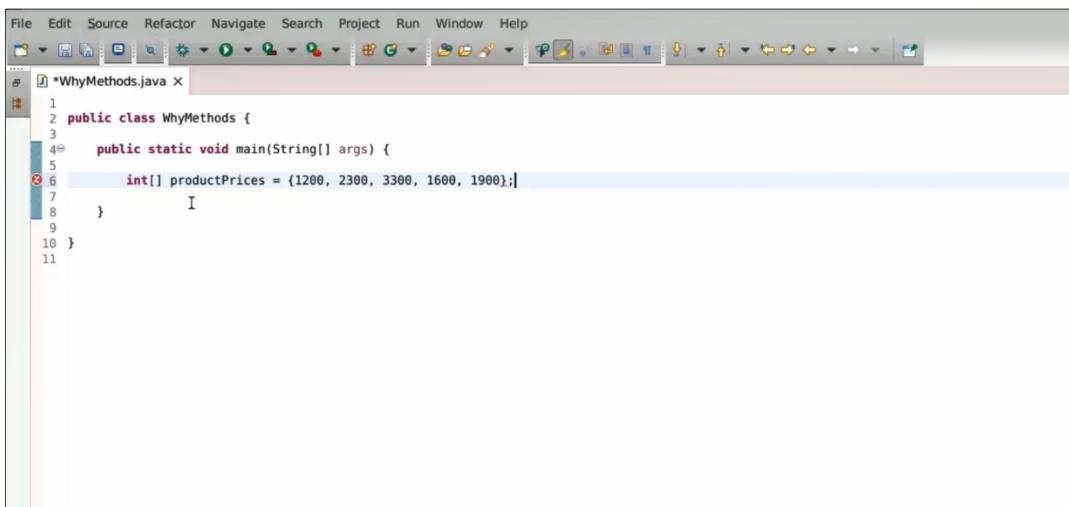


1.5 Name this class as WhyMethods, then select the main method, and then select finish



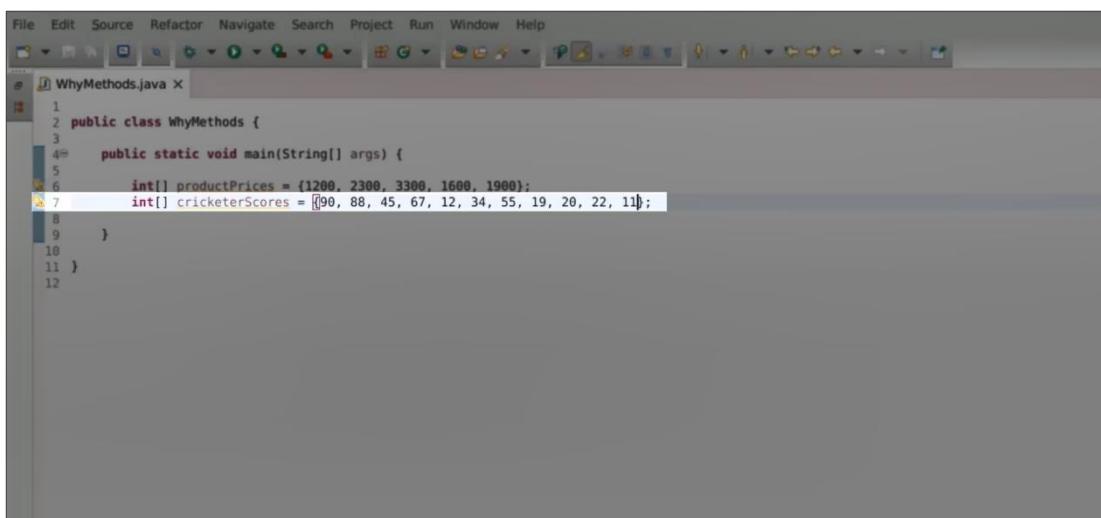
Step 2: Write arrays with cited examples

2.1 Write an array with product prices with five different elements in it



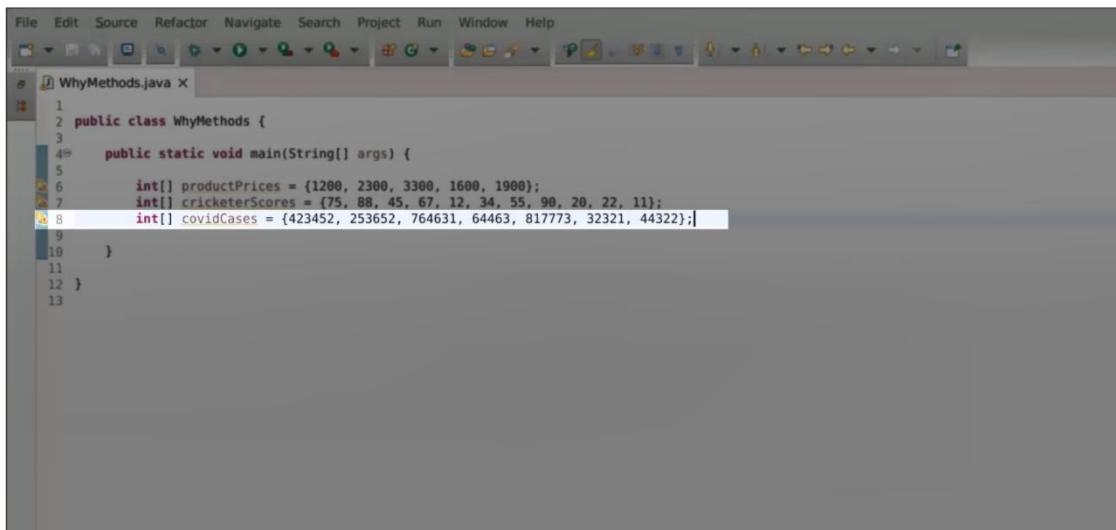
```
File Edit Source Refactor Navigate Search Project Run Window Help
*WhyMethods.java X
1
2 public class WhyMethods {
3
4     public static void main(String[] args) {
5
6         int[] productPrices = {1200, 2300, 3300, 1600, 1900};
7     }
8 }
9
10
11
```

2.2 Write cricketerScores with scores made by cricketers in a Test match



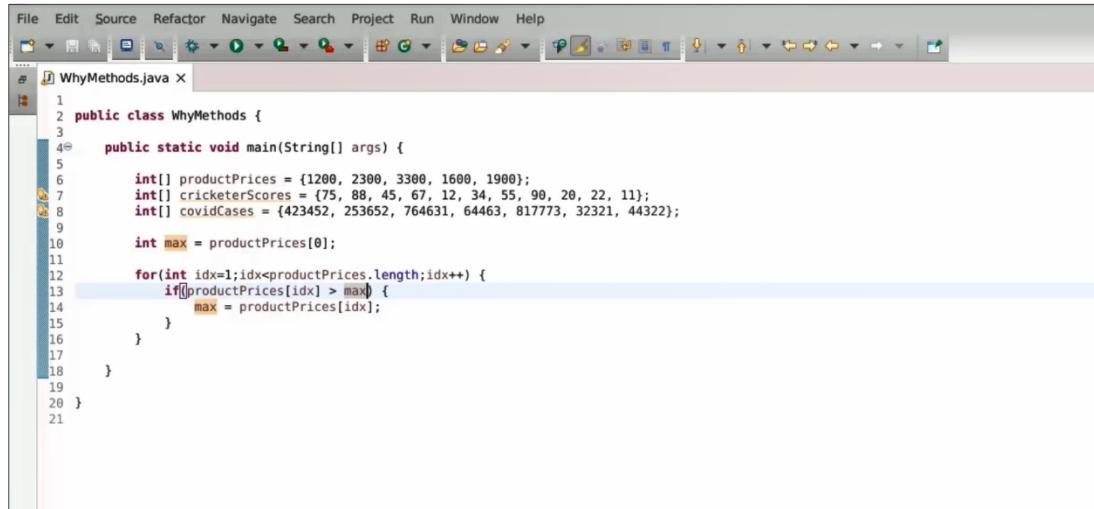
```
File Edit Source Refactor Navigate Search Project Run Window Help
*WhyMethods.java X
1
2 public class WhyMethods {
3
4     public static void main(String[] args) {
5
6         int[] productPrices = {1200, 2300, 3300, 1600, 1900};
7         int[] cricketerScores = {99, 88, 45, 67, 12, 34, 55, 19, 20, 22, 11};
8
9     }
10
11 }
```

2.3 Now you have one more array, let us use covidCases for the world, and these are the active cases



```
File Edit Source Refactor Navigate Search Project Run Window Help
# WhyMethods.java X
1
2 public class WhyMethods {
3
4     public static void main(String[] args) {
5
6         int[] productPrices = {1200, 2300, 3300, 1600, 1900};
7         int[] cricketerScores = {75, 88, 45, 67, 12, 34, 55, 90, 20, 22, 11};
8         int[] covidCases = {423452, 253652, 764631, 64463, 817773, 32321, 44322};}
9
10    }
11
12 }
13
```

2.4 Consider the maximum value in the product prices array to be the first element, which is at the zeroth index. You can assume that 1200 is the maximum value in this entire array. Now, start a loop that begins with the index at one and continues until the last element, which is **productPrices.length**. Iterate through the array one by one. If max is less than the product price at the current index, update max to be the product price at that index



```
File Edit Source Refactor Navigate Search Project Run Window Help
# WhyMethods.java X
1
2 public class WhyMethods {
3
4     public static void main(String[] args) {
5
6         int[] productPrices = {1200, 2300, 3300, 1600, 1900};
7         int[] cricketerScores = {75, 88, 45, 67, 12, 34, 55, 90, 20, 22, 11};
8         int[] covidCases = {423452, 253652, 764631, 64463, 817773, 32321, 44322};}
9
10    int max = productPrices[0];
11
12    for(int idx=1;idx<productPrices.length;idx++) {
13        if(productPrices[idx] > max) {
14            max = productPrices[idx];
15        }
16    }
17
18 }
19
20 }
21
```

2.5 Print "Maximum in product prices is " followed by max. Here, you are getting the max value from the product prices array

```

File Edit Source Refactor Navigate Search Project Run Window Help
# WhyMethods.java X
1
2 public class WhyMethods {
3
4     public static void main(String[] args) {
5
6         int[] productPrices = {1200, 2300, 3300, 1600, 1900};
7         int[] cricketerScores = {75, 88, 45, 67, 12, 34, 55, 90, 20, 22, 11};
8         int[] covidCases = {423452, 253652, 764631, 64463, 817773, 32321, 44322};
9
10        int max = productPrices[0];
11
12        for(int idx=1;idx<productPrices.length;idx++) {
13            if(productPrices[idx] > max) {
14                max = productPrices[idx];
15            }
16        }
17
18        System.out.println("Maximum in Product Prices is: "+max);
19
20    }
21
22 }
23

```

Step 3: Run the code and get the output

3.1 Run the code and see if it works fine or not, it states that the maximum is 3300 and that is the maximum value

```

File Edit Source Refactor Navigate Search Project Run Window Help
# WhyMethods.java X
1
2 public class WhyMethods {
3
4     public static void main(String[] args) {
5
6         int[] productPrices = {1200, 2300, 3300, 1600, 1900};
7         int[] cricketerScores = {75, 88, 45, 67, 12, 34, 55, 90, 20, 22, 11};
8         int[] covidCases = {423452, 253652, 764631, 64463, 817773, 32321, 44322};
9
10        int max = productPrices[0];
11
12        for(int idx=1;idx<productPrices.length;idx++) {
13            if(productPrices[idx] > max) {
14                max = productPrices[idx];
15            }
16        }
17
18        System.out.println("Maximum in Product Prices is: "+max);
19
20    }
21
22 }
23

```

<terminated> WhyMethods [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk
Maximum in Product Prices is: 3300

3.2 Repeat the same algorithm by copying and pasting the previous code. Replace **productPrices** with **cricketerScores**.

```

File Edit Source Refactor Navigate Search Project Run Window Help
# WhyMethods.java X
1
2 public class WhyMethods {
3
4     public static void main(String[] args) {
5
6         int[] productPrices = {1200, 2300, 3300, 1600, 1900};
7         int[] cricketerScores = {75, 88, 45, 67, 12, 34, 55, 90, 20, 22, 11};
8         int[] covidCases = {423452, 253652, 764631, 64463, 817773, 32321, 44322};
9
10        int max = productPrices[0];
11
12        for(int idx=1;idx<productPrices.length;idx++) {
13            if(productPrices[idx] > max) {
14                max = productPrices[idx];
15            }
16        }
17
18        System.out.println("Maximum in Product Prices is: "+max);
19
20        max = cricketerScores[0];
21
22        for(int idx=1;idx<cricketerScores.length;idx++) {
23            if(cricketerScores[idx] > max) {
24                max = cricketerScores[idx];
25            }
26        }
27
28        System.out.println("Maximum in Cricketer Scores is: "+max);
29
30
31    }
32
33 }
34

```

3.3 Re-run the code, and here you are with the value of Max as 90, which is the highest value

```

File Edit Source Refactor Navigate Search Project Run Window Help
# WhyMethods.java X
1
2 public class WhyMethods {
3
4     public static void main(String[] args) {
5
6         int[] productPrices = {1200, 2300, 3300, 1600, 1900};
7         int[] cricketerScores = {75, 88, 45, 67, 12, 34, 55, 90, 20, 22, 11};
8         int[] covidCases = {423452, 253652, 764631, 64463, 817773, 32321, 44322};
9
10        int max = productPrices[0];
11
12        for(int idx=1;idx<productPrices.length;idx++) {
13            if(productPrices[idx] > max) {
14                max = productPrices[idx];
15            }
16        }
17
18        System.out.println("Maximum in Product Prices is: "+max);
19
20        max = cricketerScores[0];
21
22        for(int idx=1;idx<cricketerScores.length;idx++) {
23            if(cricketerScores[idx] > max) {
24                max = cricketerScores[idx];
25            }
26        }
27
28        System.out.println("Maximum in Cricketer Scores is: "+max);
29
30
31    }
32
33 }
34

```

Console Output:

```

<terminated> WhyMethods [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk
Maximum in Product Prices is: 3300
Maximum in Cricketer Scores is: 90

```

3.4 Repeat the same algorithm by copy-pasting the previous code. Replace the **cricketerScores** with **covidCases**

```

File Edit Source Refactor Navigate Search Project Run Window Help
WhyMethods.java X
7 int[] productPrices = {75, 88, 45, 67, 12, 34, 55, 98, 20, 22, 11};
8 int[] covidCases = {423452, 253652, 764631, 64463, 817773, 32321, 44322};
9
10 int max = productPrices[0];
11
12 for(int idx=1;idx<productPrices.length;idx++) {
13     if(productPrices[idx] > max) {
14         max = productPrices[idx];
15     }
16 }
17
18 System.out.println("Maximum in Product Prices is: "+max);
19
20 max = cricketerScores[0];
21
22 for(int idx=1;idx<cricketerScores.length;idx++) {
23     if(cricketerScores[idx] > max) {
24         max = cricketerScores[idx];
25     }
26 }
27
28 System.out.println("Maximum in Cricketer Scores is: "+max);
29
30 max = covidCases[0];
31
32 for(int idx=1;idx<covidCases.length;idx++) {
33     if(covidCases[idx] > max) {
34         max = covidCases[idx];
35     }
36 }
37
38 System.out.println("Maximum in Covid Cases is: "+max);
39
40

```

3.5 Run this code, here you are with the maximum in the covidCases with the value 817773

```

File Edit Source Refactor Navigate Search Project Run Window Help
WhyMethods.java X
7 int[] productPrices = {75, 88, 45, 67, 12, 34, 55, 98, 20, 22, 11};
8 int[] covidCases = {423452, 253652, 764631, 64463, 817773, 32321, 44322};
9
10 int max = productPrices[0];
11
12 for(int idx=1;idx<productPrices.length;idx++) {
13     if(productPrices[idx] > max) {
14         max = productPrices[idx];
15     }
16 }
17
18 System.out.println("Maximum in Product Prices is: "+max);
19
20 max = cricketerScores[0];
21
22 for(int idx=1;idx<cricketerScores.length;idx++) {
23     if(cricketerScores[idx] > max) {
24         max = cricketerScores[idx];
25     }
26 }
27
28 System.out.println("Maximum in Cricketer Scores is: "+max);
29
30 max = covidCases[0];
31
32 for(int idx=1;idx<covidCases.length;idx++) {
33     if(covidCases[idx] > max) {
34         max = covidCases[idx];
35     }
36 }
37
38 System.out.println("Maximum in Covid Cases is: "+max);
39
40

```

Output:

```

Problems Javadoc Declaration Console X
<terminated> WhyMethods [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk
Maximum in Product Prices is: 3300
Maximum in Cricketer Scores is: 90
Maximum in Covid Cases is: 817773

```

3.6 When a certain task is repeatedly executed, why waste time writing the same code again? This is where methods are useful to save development time. In the same class, create a method with an integer return type, named getMax, and take one array as input. Then, run this entire algorithm on the array instead of on the product prices. Finally, return the maximum value. This is known as a non-static method

```

File Edit Source Refactor Navigate Search Project Run Window Help
# WhyMethods.java x
1  public class WhyMethods {
2      // non static method
3      int getMax(int[] array) {
4          int max = array[0];
5          for(int idx=1;idx<array.length;idx++) {
6              if(array[idx] > max) {
7                  max = array[idx];
8              }
9          }
10         return max;
11     }
12     public static void main(String[] args) {
13         int[] productPrices = {1200, 2300, 3300, 1600, 1900};
14         int[] cricketerScores = {75, 88, 45, 67, 12, 34, 55, 90, 20, 22, 11};
15         int[] covidCases = {423452, 253652, 764631, 64463, 817773, 32321, 44322};
16
17         int max = productPrices[0];
18
19         for(int idx=1;idx<productPrices.length;idx++) {
20             if(productPrices[idx] > max) {
21                 max = productPrices[idx];
22             }
23         }
24         System.out.println("Maximum in Product Prices is: "+max);
25         max = cricketerScores[0];
26
27     }
28
29 }

```

3.7 To use this method, comment out the entire previous code. Now, create an object of the class. Write **WhyMethods** and create a reference variable. Use the new operator, followed by your class name with parentheses; this is known as the object construction statement. Write **System.out.println("Maximum in product prices is " + referenceVariable.getMax(productPrices));**. This will execute the **getMax** method using the reference variable and pass the **productPrices** array

```

File Edit Source Refactor Navigate Search Project Run Window Help
# WhyMethods.java x
1  public class WhyMethods {
2      // non static method
3      int getMax(int[] array) {
4          int max = array[0];
5          for(int idx=1;idx<array.length;idx++) {
6              if(array[idx] > max) {
7                  max = array[idx];
8              }
9          }
10         return max;
11     }
12     public static void main(String[] args) {
13
14         int[] productPrices = {1200, 2300, 3300, 1600, 1900};
15         int[] cricketerScores = {75, 88, 45, 67, 12, 34, 55, 90, 20, 22, 11};
16         int[] covidCases = {423452, 253652, 764631, 64463, 817773, 32321, 44322};
17
18         // Object Construction Statement
19         WhyMethods ref = new WhyMethods();
20         System.out.println("Maximum in Product Prices is: "+ref.getMax(productPrices));
21
22         /*int max = productPrices[0];
23
24         for(int idx=1;idx<productPrices.length;idx++) {
25             if(productPrices[idx] > max) {
26                 max = productPrices[idx];
27             }
28         }
29
30         System.out.println("Maximum in Product Prices is: "+max);
31
32 */
33
34 }

```

3.8 Run this code, and now you can see the maximum in product prices is 3300

The screenshot shows the Eclipse IDE interface. On the left, the code editor displays a Java file named `WhyMethods.java`. The code contains a non-static method `getMax` that finds the maximum value in an integer array. It also includes a `main` method that initializes three arrays: `productPrices`, `cricketerScores`, and `covidCases`, and prints the maximum value from `productPrices`. On the right, the `Console` view shows the output of the application, which is `Maximum in Product Prices is: 3300`.

```
1  public class WhyMethods {  
2  
3     // non static method  
4     int getMax(int[] array) {  
5  
6         int max = array[0];  
7  
8         for(int idx=1;idx<array.length;idx++) {  
9             if(array[idx] > max) {  
10                 max = array[idx];  
11             }  
12         }  
13     }  
14  
15     return max;  
16 }  
17  
18  
19# public static void main(String[] args) {  
20  
21     int[] productPrices = {1200, 2300, 3300, 1600, 1900};  
22     int[] cricketerScores = {75, 88, 45, 67, 12, 34, 55, 90, 20, 22, 11};  
23     int[] covidCases = {423452, 253652, 764631, 64463, 817773, 32321, 44322};  
24  
25     // Object Construction Statement  
26     WhyMethods obj = new WhyMethods();  
27     System.out.println("Maximum in Product Prices is: "+obj.getMax(productPrices));  
28  
29     /*int max = productPrices[0];  
30  
31     for(int idx=1;idx<productPrices.length;idx++) {  
32         if(productPrices[idx] > max) {  
33             max = productPrices[idx];  
34         }  
35     }  
36 */  
37 }
```

3.9 Now you will execute the same method repeatedly, rather than writing the same code repeatedly, and you can even use the same method on different arrays to get the maximum out of it

```
1 public class WhyMethods {
2     // non static method
3     int getMax(int[] array) {
4         int max = array[0];
5         for(int idx=1;idx<array.length;idx++) {
6             if(array[idx] > max) {
7                 max = array[idx];
8             }
9         }
10        return max;
11    }
12
13    public static void main(String[] args) {
14        int[] productPrices = {1200, 2300, 3300, 1600, 1900};
15        int[] cricketerScores = {75, 88, 45, 67, 12, 34, 55, 90, 20, 22, 11};
16        int[] covidCases = {423452, 253652, 764631, 64463, 817773, 32321, 44322};
17
18        // Object Construction Statement
19        WhyMethods ref = new WhyMethods();
20
21        System.out.println("Maximum in Product Prices is: "+ref.getMax(productPrices));
22        System.out.println("Maximum in Cricketer Scores is: "+ref.getMax(cricketerScores));
23        System.out.println("Maximum in Covid Cases is: "+ref.getMax(covidCases));
24
25        //int max = productPrices[0];
26
27        for(int idx=1;idx<productPrices.length;idx++) {
28            if(productPrices[idx] > max) {
29                max = productPrices[idx];
30            }
31        }
32
33        System.out.println("Maximum in Product Prices is: "+max);
34    }
}
```

3.10 Run the code and you will get the same output

```

File Edit Source Refactor Navigate Search Project Run Window Help
WhyMethods.java Run WhyMethods
1 public class WhyMethods {
2     // non static method
3     int getMax(int[] array) {
4         int max = array[0];
5         for(int idx=1;idx<array.length;idx++) {
6             if(array[idx] > max) {
7                 max = array[idx];
8             }
9         }
10        return max;
11    }
12
13    public static void main(String[] args) {
14        int[] productPrices = {1200, 2300, 3300, 1600, 1900};
15        int[] cricketerScores = {75, 88, 49, 67, 12, 34, 55, 90, 20, 22, 11};
16        int[] covidCases = {423432, 253652, 764631, 64463, 817773, 32321, 44322};
17
18        // Object Construction Statement
19        WhyMethods ref = new WhyMethods();
20        System.out.println("Maximum in Product Prices is: "+ref.getMax(productPrices));
21        System.out.println("Maximum in Cricketer Scores is: "+ref.getMax(cricketerScores));
22        System.out.println("Maximum in Covid Cases is: "+ref.getMax(covidCases));
23
24        /*int max = productPrices[0];
25        for(int idx=1;idx<productPrices.length;idx++) {
26            if(productPrices[idx] > max) {
27                max = productPrices[idx];
28            }
29        }
30
31        */
32
33        for(int idx=1;idx<productPrices.length;idx++) {
34            if(productPrices[idx] > max) {
35                max = productPrices[idx];
36            }
37        }
38    }
}

```

<terminated> WhyMethods [Java Application] /usr/eclipse/plugins/org.eclipse.jdt.core/openjdk
Maximum in Product Prices is: 3300
Maximum in Cricketer Scores is: 90
Maximum in Covid Cases is: 817773

Step 4: Create and use a non-static method

4.1 The same method can also be rewritten, write **getMaxFromArray**, here you have changed the name since you cannot have two methods with the same name. This time you can add a static keyword in front of it. So this method which is marked as static is known as a static method

```

File Edit Source Refactor Navigate Search Project Run Window Help
WhyMethods.java X
1 public class WhyMethods {
2     // non static method | Executed with object's reference
3     int getMax(int[] array) {
4         int max = array[0];
5         for(int idx=1;idx<array.length;idx++) {
6             if(array[idx] > max) {
7                 max = array[idx];
8             }
9         }
10        return max;
11    }
12
13    // static method | Executed by Class Name
14    static int getMaxFromArray(int[] array) {
15        int max = array[0];
16
17        for(int idx=1;idx<array.length;idx++) {
18            if(array[idx] > max) {
19                max = array[idx];
20            }
21        }
22
23        return max;
24    }
25
26    public static void main(String[] args) {
27        int[] productPrices = {1200, 2300, 3300, 1600, 1900};
28
29    }
}

```

4.2 Now, instead of using the object's reference, write the class name followed by `.getMaxFromArray`



The screenshot shows a Java IDE interface with a toolbar at the top containing various icons for file operations, search, and navigation. The main window displays the code for `WhyMethods.java`. The code defines a static method `getMaxFromArray` that iterates through an array to find the maximum value. It also contains a `main` method that initializes three arrays: `productPrices`, `cricketerScores`, and `covidCases`, and prints the maximum values for each using the `getMaxFromArray` method. The code uses System.out.println statements to output the results.

```
File Edit Source Refactor Navigate Search Project Run Window Help

WhyMethods.java X
19  static int getMaxFromArray(int[] array) {
20
21     int max = array[0];
22
23     for(int idx=1;idx<array.length;idx++) {
24         if(array[idx] > max) {
25             max = array[idx];
26         }
27     }
28
29     return max;
30 }
31
32
33 public static void main(String[] args) {
34
35     int[] productPrices = {1200, 2300, 3300, 1600, 1900};
36     int[] cricketerScores = {75, 88, 45, 67, 12, 34, 55, 98, 20, 22, 11};
37     int[] covidCases = {423452, 253652, 764631, 64463, 817773, 32321, 44322};
38
39     // Object Construction Statement
40
41     //WhyMethods ref = new WhyMethods();
42     //System.out.println("Maximum in Product Prices is: "+ref.getMax(productPrices));
43     //System.out.println("Maximum in Cricketer Scores is: "+ref.getMax(cricketerScores));
44     //System.out.println("Maximum in Covid Cases is: "+ref.getMax(covidCases));
45
46     System.out.println("Maximum in Product Prices is: "+WhyMethods.getMaxFromArray(productPrices));
47     System.out.println("Maximum in Cricketer Scores is: "+WhyMethods.getMaxFromArray(cricketerScores));
48     System.out.println("Maximum in Covid Cases is: "+WhyMethods.getMaxFromArray(covidCases));
49
50     /*int max = productPrices[0];
51
52     for(int idx=1;idx<productPrices.length;idx++) {
53
54         if(productPrices[idx] > max) {
55             max = productPrices[idx];
56         }
57     }
58
59     System.out.println("Maximum in Product Prices is: "+max);
60
61     max = cricketerScores[0];
62
63     for(int idx=1;idx<cricketerScores.length;idx++) {
64
65         if(cricketerScores[idx] > max) {
66             max = cricketerScores[idx];
67         }
68     }
69
70     System.out.println("Maximum in Cricketer Scores is: "+max);
71
72     max = covidCases[0];
73
74     for(int idx=1;idx<covidCases.length;idx++) {
75
76         if(covidCases[idx] > max) {
77             max = covidCases[idx];
78         }
79     }
80
81     System.out.println("Maximum in Covid Cases is: "+max);
```

4.3 Run the code and you will get the same output

The screenshot shows the Eclipse IDE interface with the following details:

- Left Panel (Code Editor):** The file `WhyMethods.java` is open. The code defines a class with a static method `getMaxFromArray` and a main method. The `getMaxFromArray` method iterates through an array to find the maximum value. The `main` method initializes three arrays: `productPrices`, `cricketerScores`, and `covidCases`, and prints the maximum values for each.
- Right Panel (Console):** The console output shows the results of the program execution:

```
<terminated> WhyMethods [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk
Maximum in Product Prices is: 3300
Maximum in Cricketer Scores is: 90
Maximum in Covid Cases is: 817773
```

Step 5: Create an object with an object construction statement

5.1 Return back to your Package explorer, open the Session number 7 folder, and here select the CovidCases dot java file. In this file, you can see that some statements are repeatedly used

```

File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer X CovidCases.java X
1 import java.util.Scanner;
2
3 public class CovidCases {
4
5     public static void main(String[] args) {
6
7         String[] countryNames = {"USA", "India", "Brazil", "UK", "France"};
8
9         int totalCases = 0;
10        int totalRecoveredCases = 1;
11        int activeCases = 2;
12
13        int [][] covidCases = {
14            {66995533, 43090644, 23031325}, // 0 -> USA
15            {37380253, 35237461, 1656310}, // 1 -> India
16            {23006952, 621099, 675022}, // 2 -> Brazil
17            {15217280, 151987, 3676112}, // 3 -> UK
18            {14172384, 9019484, 5025933}, // 4 -> France
19        };
20
21        System.out.println("Total Cases\tTotal Recovered\tActive Cases");
22        System.out.println("-----");
23    }
}

```

Console X

```

<terminated> WhyMethods [java Application] /usr/eclipse/plugins/org.eclipse.jdt.openjdk.hotspot.jre.full.linux.x86_64_17.0.1.v20211116-1657/jre/bin/java (Jan
Maximum in Product Prices is: 3309
Maximum in Cricketer Scores is: 90
Maximum in Covid Cases is: 817773

```

5.2 Select the piece of logic which is printing the data, copy this, and inside your class, write Static void print cases, and paste this logic here:

```

File Edit Source Refactor Navigate Search Project Run Window Help
*CovidCases.java X
1 import java.util.Scanner;
2
3 public class CovidCases {
4
5     static void printCases() {
6         for(int i=0;i<covidCases.length;i++) {
7             System.out.println(countryNames[i]);
8             System.out.println("-----");
9             System.out.print(covidCases[i][totalCases]);
10            System.out.println();
11            System.out.println();
12        }
13    }
14
15    public static void main(String[] args) {
16
17        String[] countryNames = {"USA", "India", "Brazil", "UK", "France"};
18
19        int totalCases = 0;
20        int totalRecoveredCases = 1;
21        int activeCases = 2;
22
23        int [][] covidCases = {
24            {66995533, 43090644, 23031325}, // 0 -> USA
25            {37380253, 35237461, 1656310}, // 1 -> India
26            {23006952, 621099, 675022}, // 2 -> Brazil
27            {15217280, 151987, 3676112}, // 3 -> UK
28            {14172384, 9019484, 5025933} // 4 -> France
29        };
30
31    }
}

```

5.3 You can even take one more array, like the array of strings which is the country names, instead of these total cases, you can pass down this filter now. You have a method created as a static method, which will take the array of COVID cases, an array of country names, and a filter as input

The screenshot shows a Java IDE interface with the following details:

- File Menu:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help.
- Toolbar:** Includes icons for New, Open, Save, Cut, Copy, Paste, Find, Select All, and others.
- Code Editor:** Displays the `CovidCases.java` file with the following code:

```
1 import java.util.Scanner;
2
3 public class CovidCases {
4
5
6    static void printCases(int[][] covidCases, String[] countryNames, int filter) {
7        for(int i=0;i<covidCases.length;i++) {
8
9            System.out.println(countryNames[i]);
10           System.out.println("-----");
11
12           System.out.print(covidCases[i][totalCases]);
13
14
15           System.out.println();   I
16           System.out.println();
17       }
18   }
19
20
21
22
23
24     String[] countryNames = {"USA", "India", "Brazil", "UK", "France"};
25
26     int totalCases = 0;
27     int totalRecoveredCases = 1;
28     int activeCases = 2;
29
30     int [][] covidCases = {
31         {66995533, 43090644, 23031325}, // 0 -> USA
32         {37380253, 35237461, 1656310}, // 1 -> India
33         {23006952, 621099, 675022}, // 2 -> Brazil
34         {15217280, 151987, 3676112}, // 3 -> UK
35         {15217280, 151987, 3676112} // 4 -> France
36     };
37 }
```

Step 6: Differentiate between running a static and a non-static method

6.1 Scroll back to your case number 2, where you want to print the total cases. Now, you can replace this logic with a single line of code: **CovidCases.printCases(covidCases, countryNames, totalCases);**. Do the same for cases 3 and 4

The screenshot shows a Java IDE interface with a menu bar (File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help) and a toolbar with various icons. The main window displays a Java file named "CovidCases.java". The code prints total cases and recovered cases for each country. The code uses System.out.println() for output and includes comments explaining the logic.

```
 87     }
 88 }
 89
 90     break;
 91
 92     case 2:
 93
 94         System.out.println("Total Cases");
 95         System.out.println("-----");
 96
 97         /*for(int i=0;i<covidCases.length;i++) {
 98
 99             System.out.println(countryNames[i]);
100             System.out.println("-----");
101
102             System.out.print(covidCases[i][totalCases]);
103
104
105             System.out.println();
106             System.out.println();
107         }*/
108
109     CovidCases.printCases(covidCases, countryNames, totalCases);
110
111     break;
112
113
114     case 3:
115
116         System.out.println("Recovered Cases");
117         System.out.println("-----");
118
119         for(int i=0;i<covidCases.length;i++) {
120
121             System.out.println(countryNames[i]);
122             System.out.println("-----");
123 }
```

6.2 Run the code. Now, if you try to filter based on the total cases, it will work the same way. If you try to filter based on recovered cases, it will give you the recovered cases. Similarly, if you filter based on active cases, it will give you active cases

The screenshot shows the Eclipse IDE interface with the following details:

- File Bar:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help.
- Editor:** Shows the code for `CovidCases.java`. The code includes a switch statement with cases for 1, 2, 3, and 4. Case 4 prints "Active Cases". The code also includes a loop that prints country names and active cases for each country.
- Console:** Shows the output of the application. It lists countries and their active cases:
 - India: 35237461
 - Brazil: 621099
 - UK: 151987
 - France: 9019484

By following the above steps, you have successfully depicted how methods are used and implemented in Java.