Lesson 02 Demo 05

Sorting Arrays with Methods

Objective: Sort an array using methods and a bubble sort algorithm **Tools required:** Eclipse IDE **Prerequisites:** None

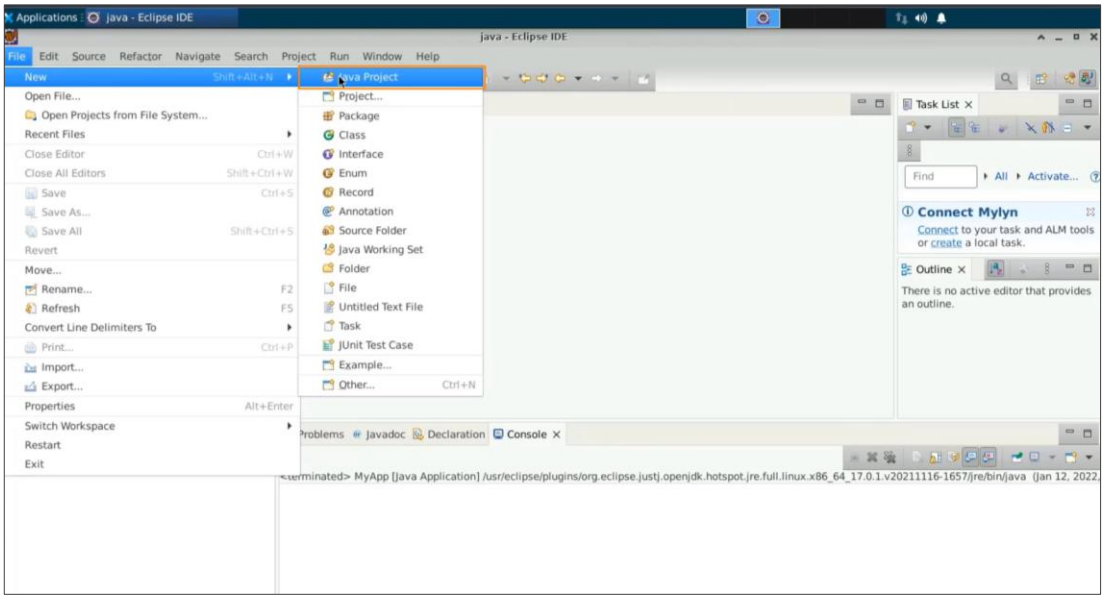
Steps to be followed:

1. Create a class and a main method
2. Create a method of static void print to print the array
3. Execute the bubble sort method and reprint the array
4. Calculate the length of the array and swap values
5. Dry run the bubble sort algorithm and add a debugger view

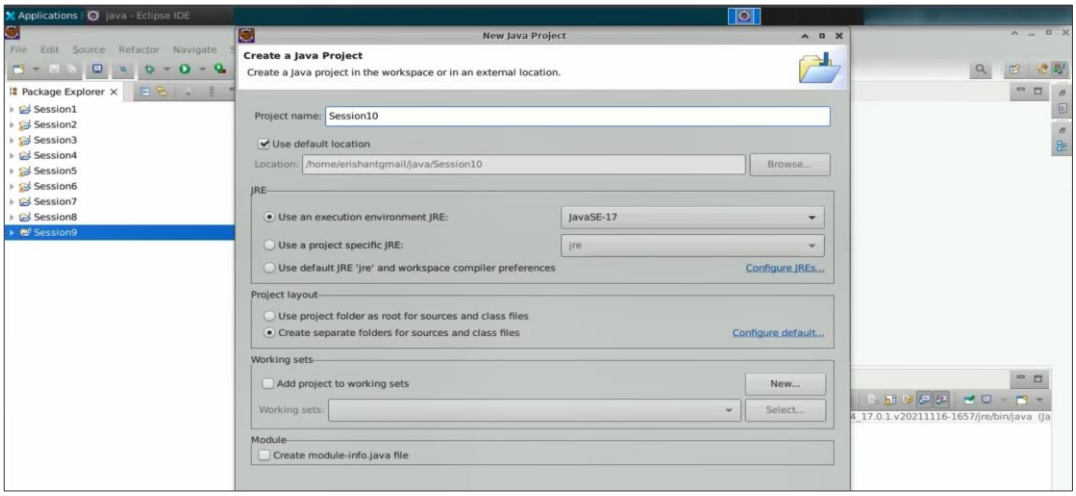
Step 1: Create a class and a main method

1.1 Open the **Eclipse IDE**

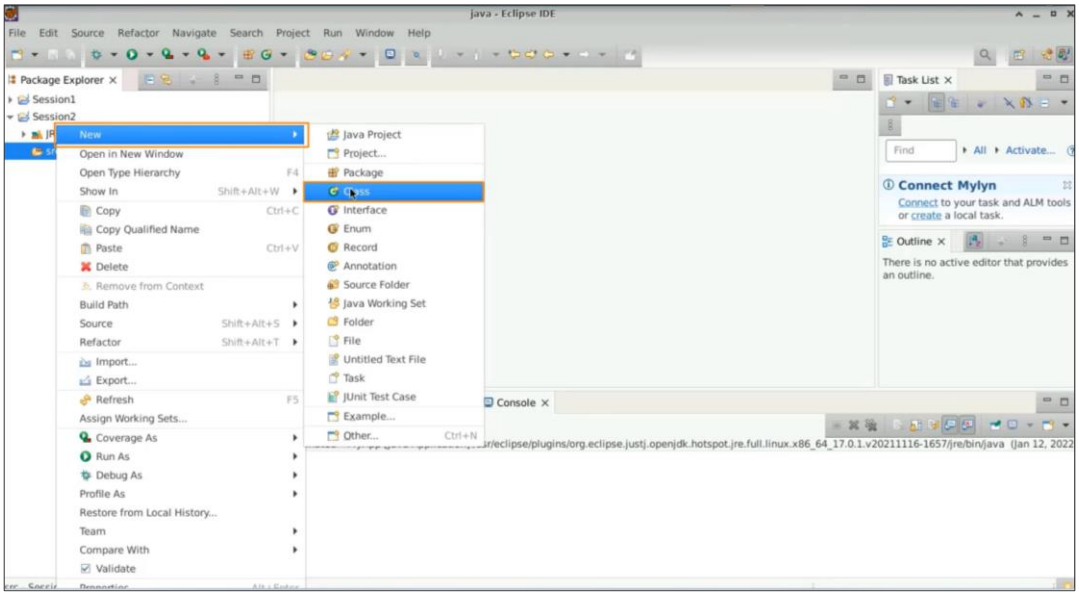
1.2. Select **File**, then **New,** and then **Java project**



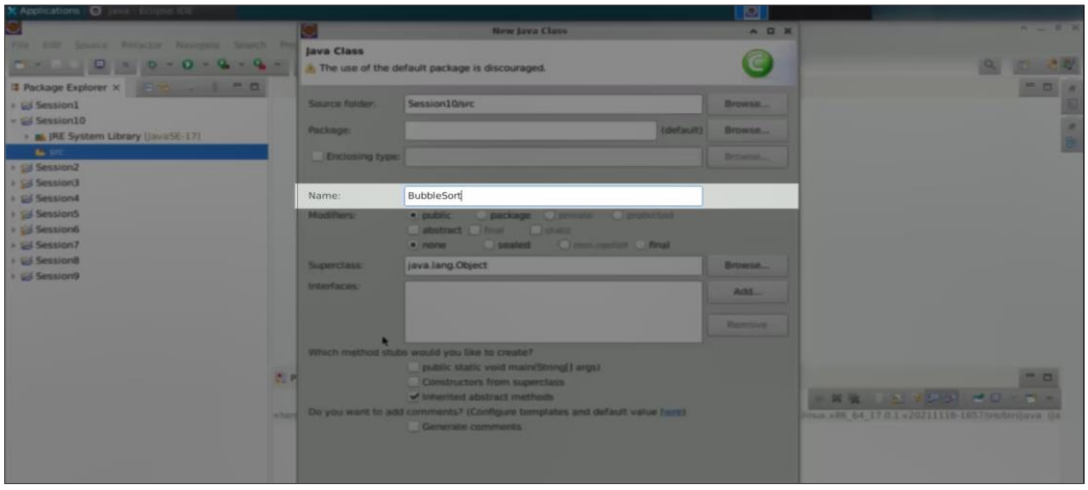
1.3 Name the project **“Session10”,** uncheck **“Create a module info dot Java file”**, and press **Finish**



1.4 With a **Session10** on the src, do a right-click and create a **new class**

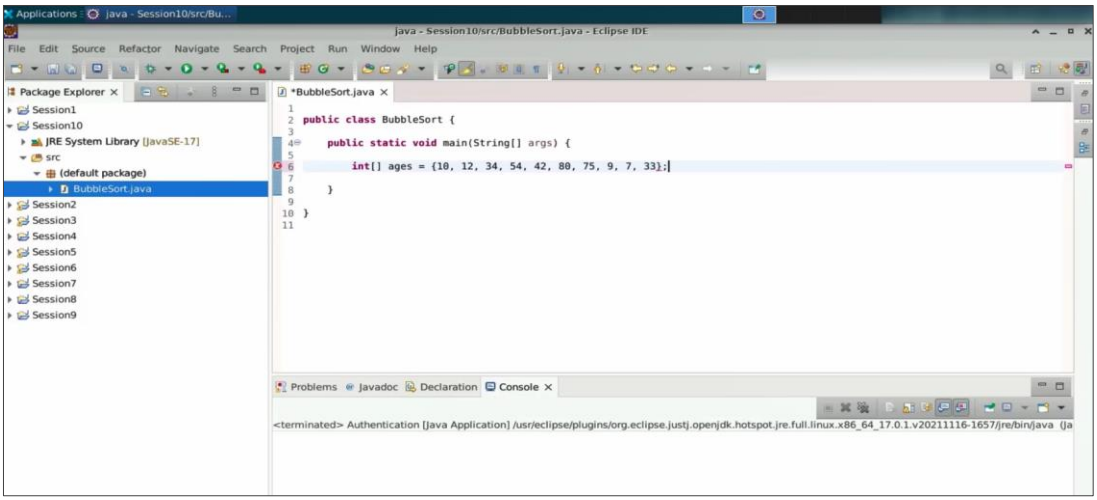


1.5 Name this class as an **BubbleSort**, then select the **main method,** and then select **finish**

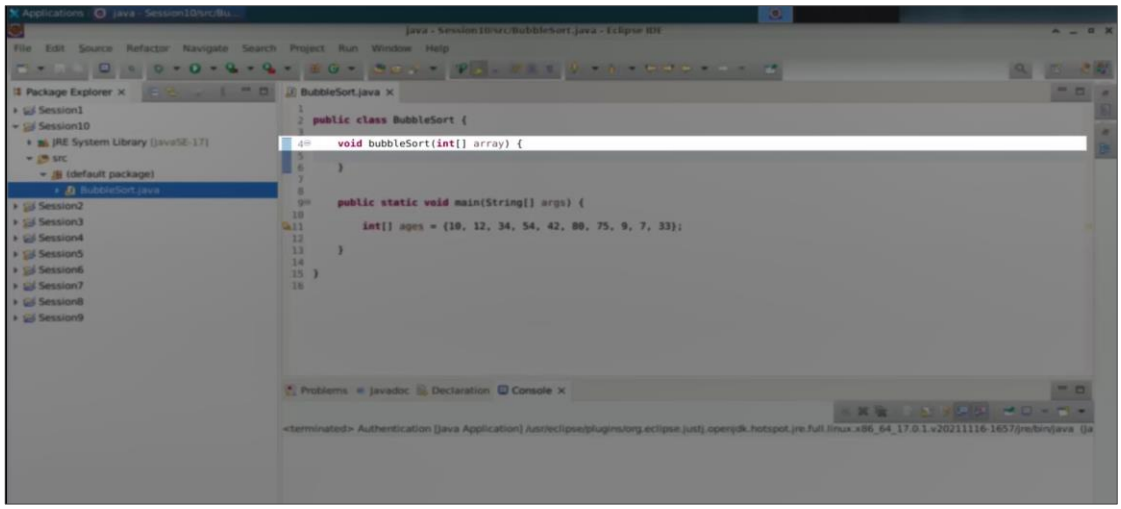


Step 2: Create a method of static void print to print the array

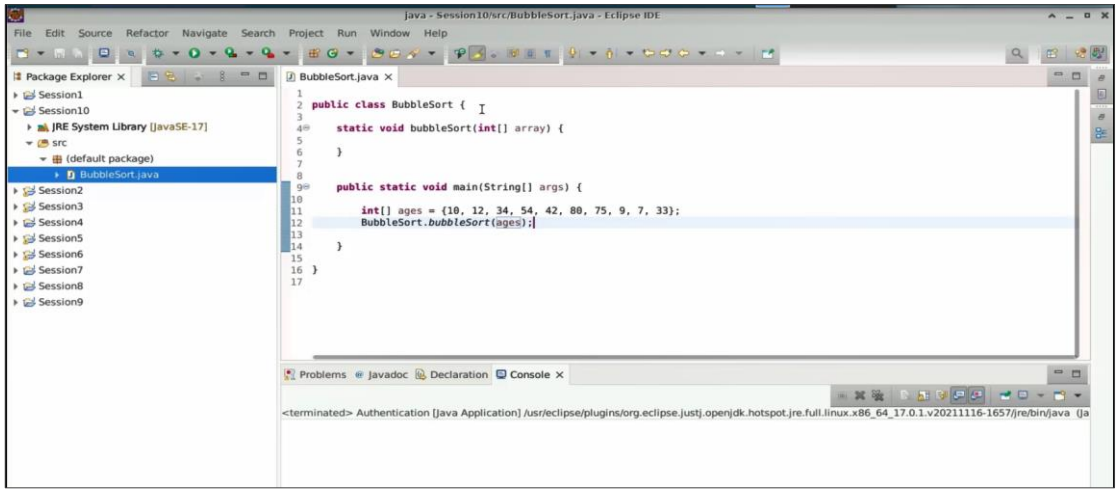
2.1 Consider you have an array named as ages. You have ten different entries



2.2 Now you want this array to be sorted. The simplest one is the bubble sort. Write a method void bubble sort and inside it, you will take one array as input, which you wish to sort. Here you go with the bubble sort, a method with an array as input

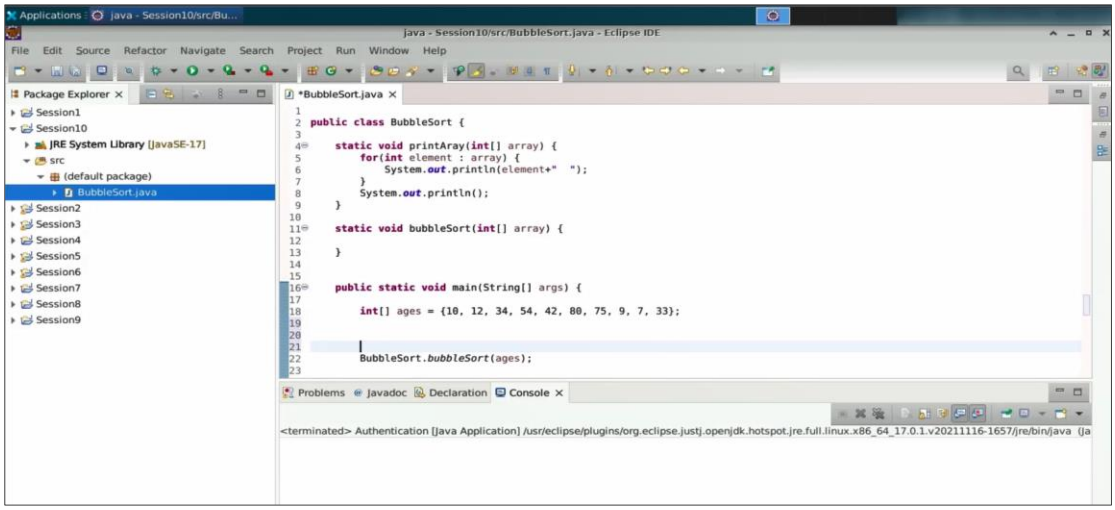


2.3 Make it static. Type bubble sort, and you'll execute this method called bubble sort. And you pass the edges as your array with the input to the method here

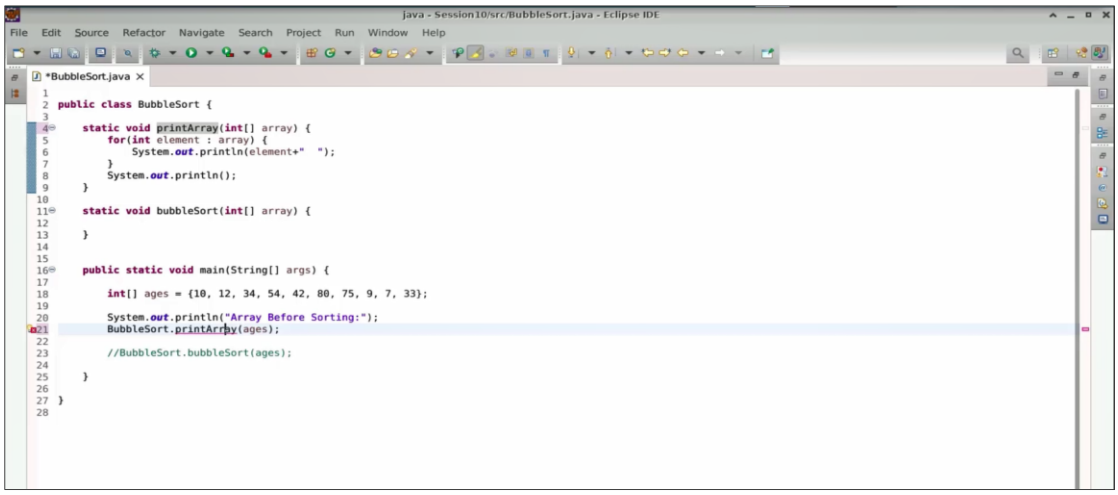


Step 3: Execute the bubble sort method and reprint the array

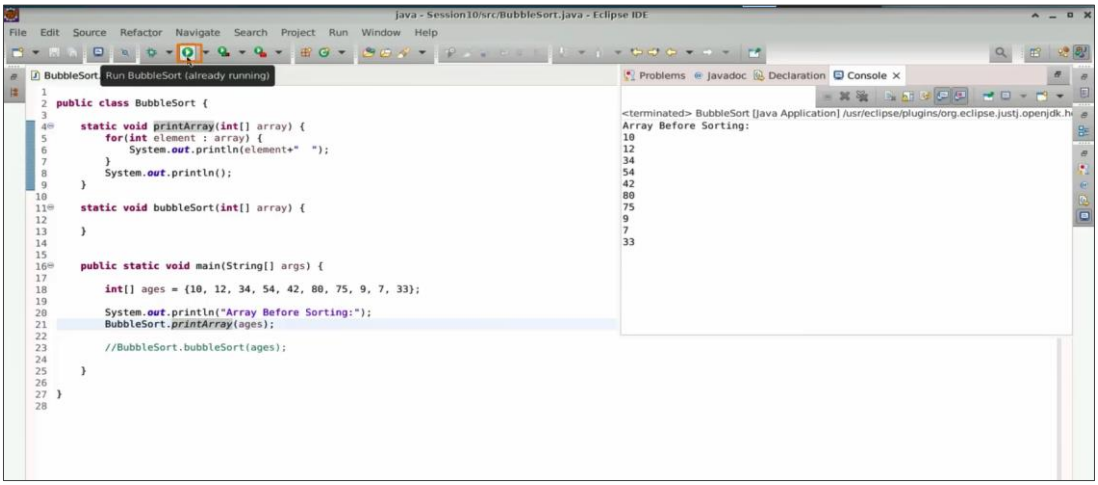
3.1 You will create another method called a static void print array. This method is also going to take one integer array as input. And type for integer element in the array. Print down this element with the space. Once the loop is finished, add an empty print line



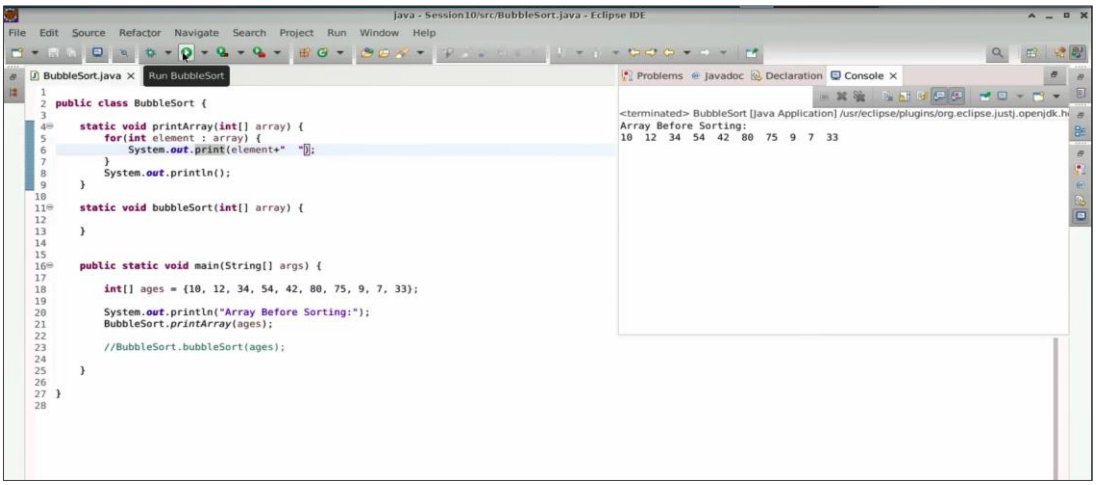
3.2 Type Array before sorting. You can add the class name that's the bubble sort dot print the array and you pass the ages, then just print the array



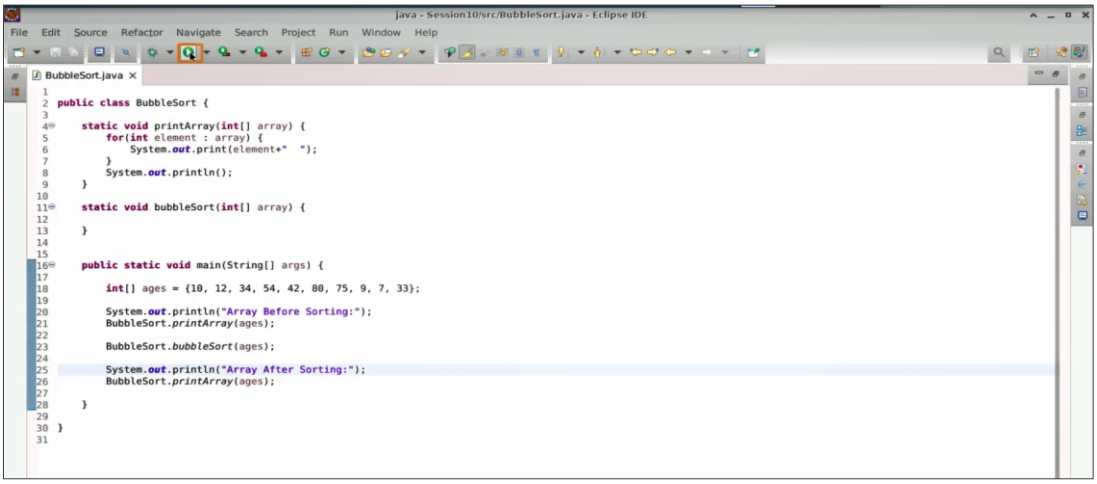
3.3 Run the code and see what kind of order it is:



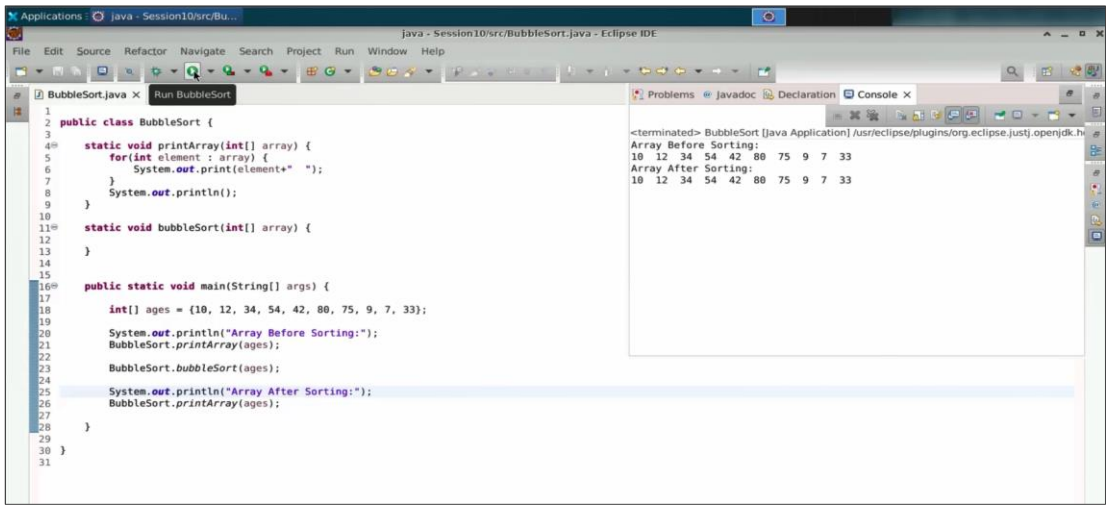
3.4 Type print, not the print line, let it be printed in the same line, so you can see 10 12 34 54. It is exactly in the same order in which you have created an array



3.5 Now, to execute this method called **bubbleSort.bubbleSort()**, once you execute this method, you can print the array after sorting to see the changes

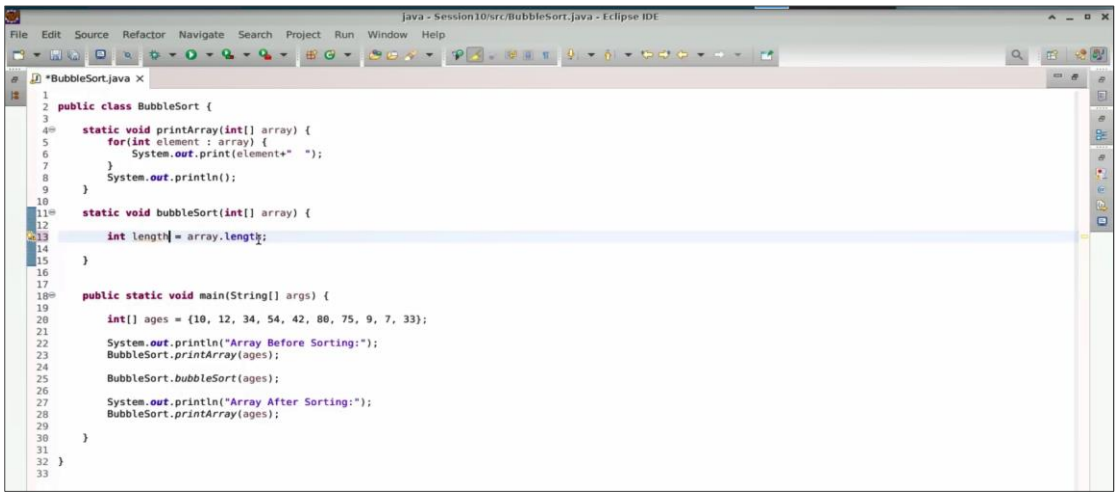


3.6 Run the code and you see is that before sorting and after sorting, the output is exactly same. That is for reason why you have not implemented the bubble sort method yet

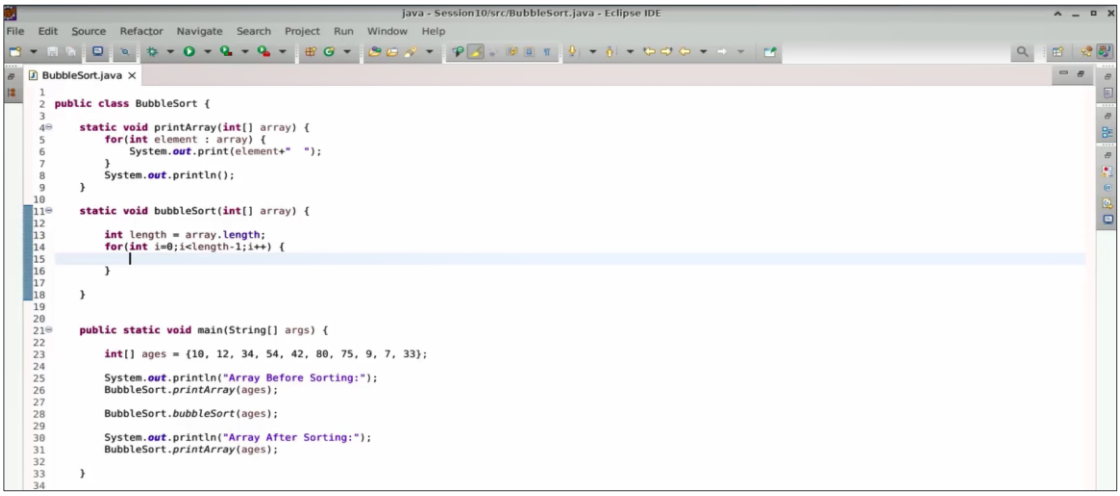


Step 4: Calculate the length of the array and swap values

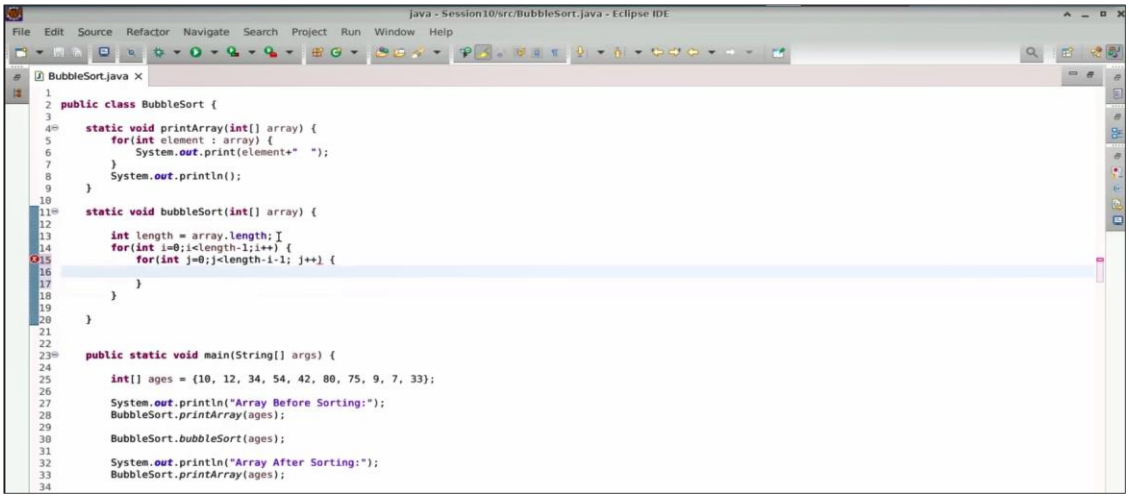
4.1 Now let us get started with the bubble sort method. The first thing you are going to do is, capture the length of the array in a length variable that you have created now



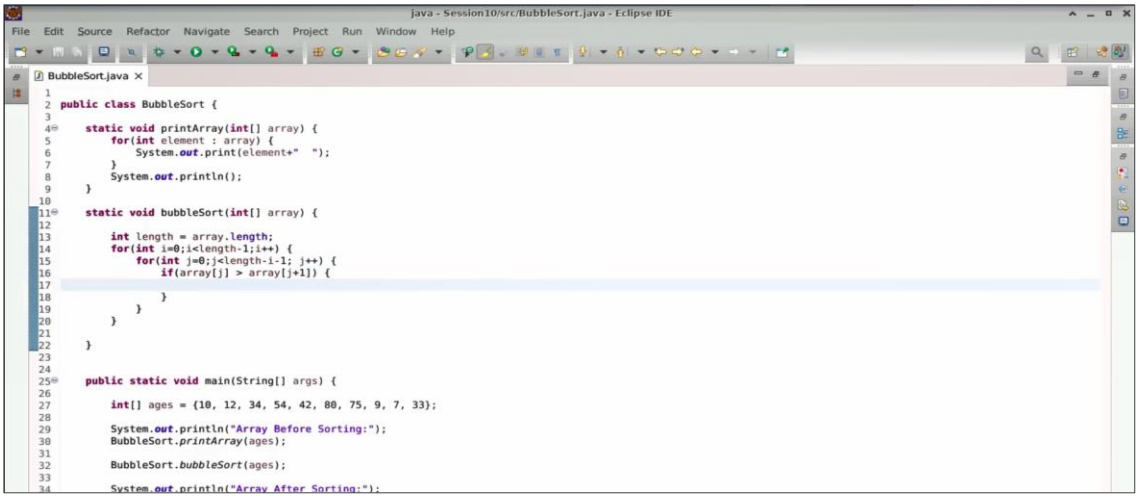
4.2 Now, you are going to write a for loop that starts with the i value as zero. The loop runs while i is less than length - 1, and increments i by one. This loop begins from the first index and continues until it is less than length - 1.



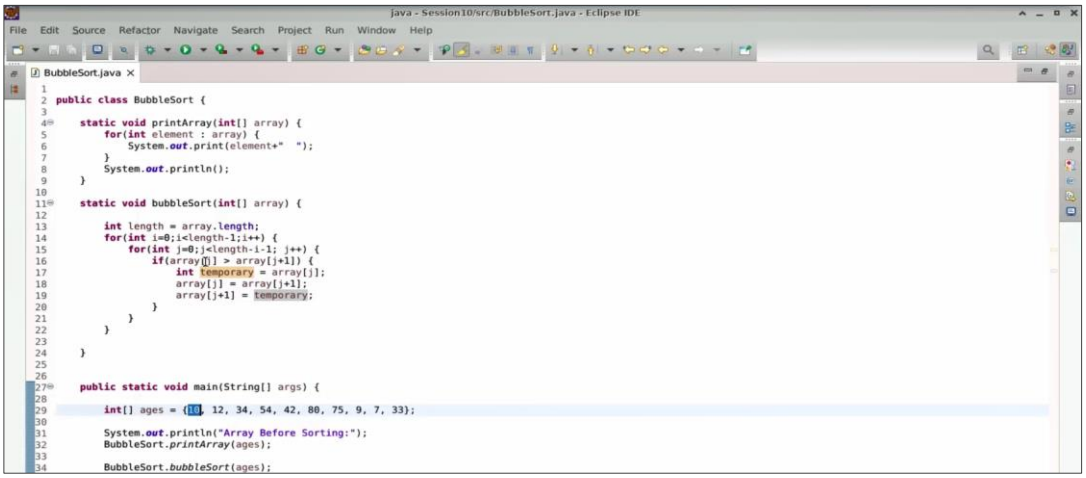
4.3 Inside this you are going to create another for loop which goes as j value 0. And the j will be limited to the length-i-1 and then give j++. Now length-i-1 would typically decrease. The i variable from the length and then one more -1. If you begin with the zero, so the loop is going to go from zero till 8 and skip 9



4.4 Now, inside the loop, add an if-statement: if **array[j]** is greater than **array[j+1]**. This is a standard algorithm. If an element is greater than the next element, you need to swap them. To perform the swap, you need to create a temporary variable

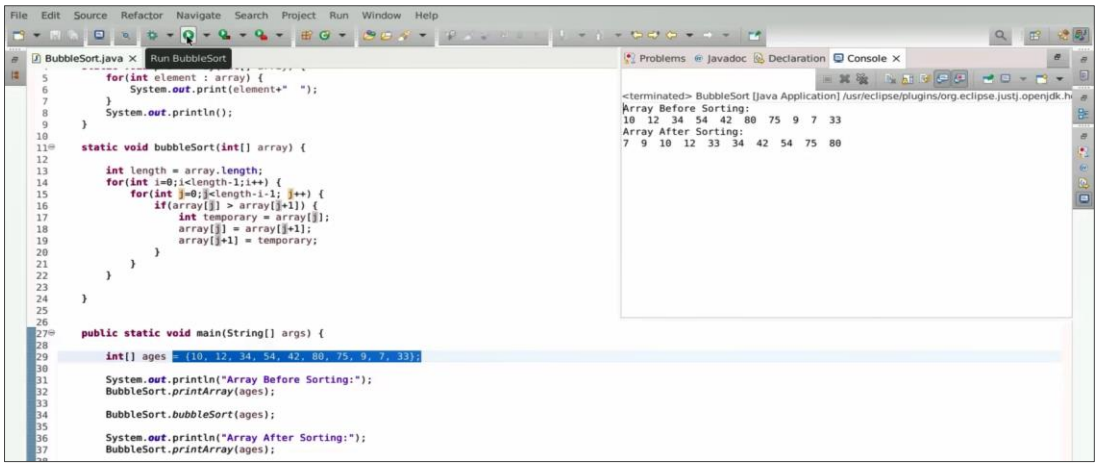


4.5 Let us add a temporary variable in which you will copy the array of j and then add the array of j that is j+1. And then in the array of j+1, you can copy your temporary. Now you can perform the swapping in much more different ways. But to use the swap concept with a very basic approach, the temporary variable which will hold the value of array of j

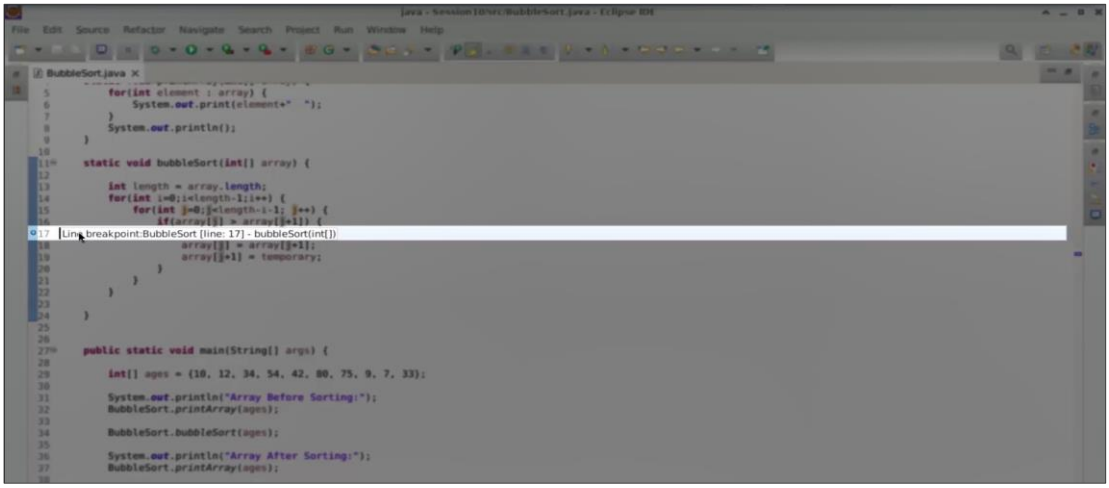


Step 5: Dry run the bubble sort algorithm and add a debugger view

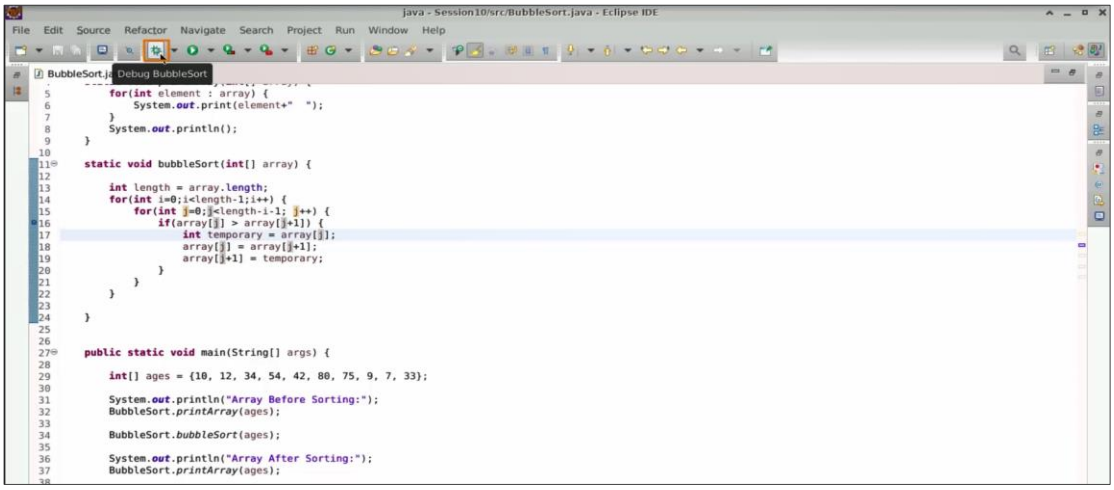
5.1 Let us run this algorithm. Now here you are with 7, 9, 10, 12, 13, 34 42, 54, 75 and 80. This algorithm is implemented well



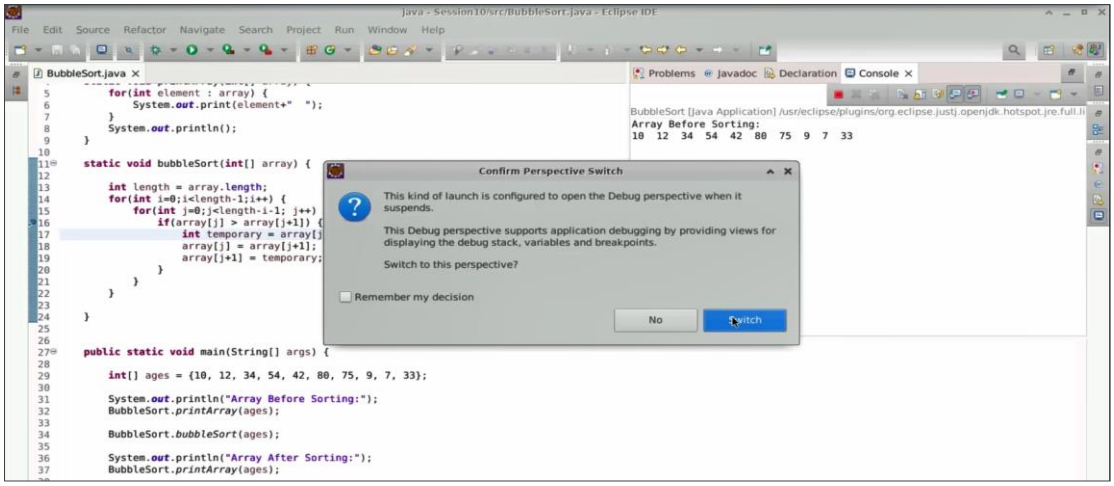
5.2 Put a breakpoint on the if condition and the temporary variable



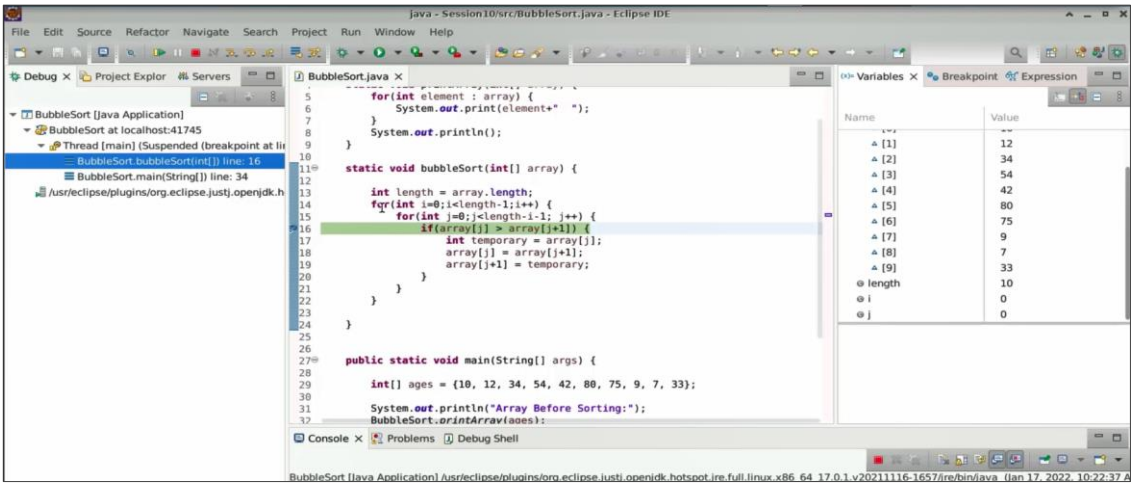
5.3 And now instead of selecting run, you can select debug the bubble sort



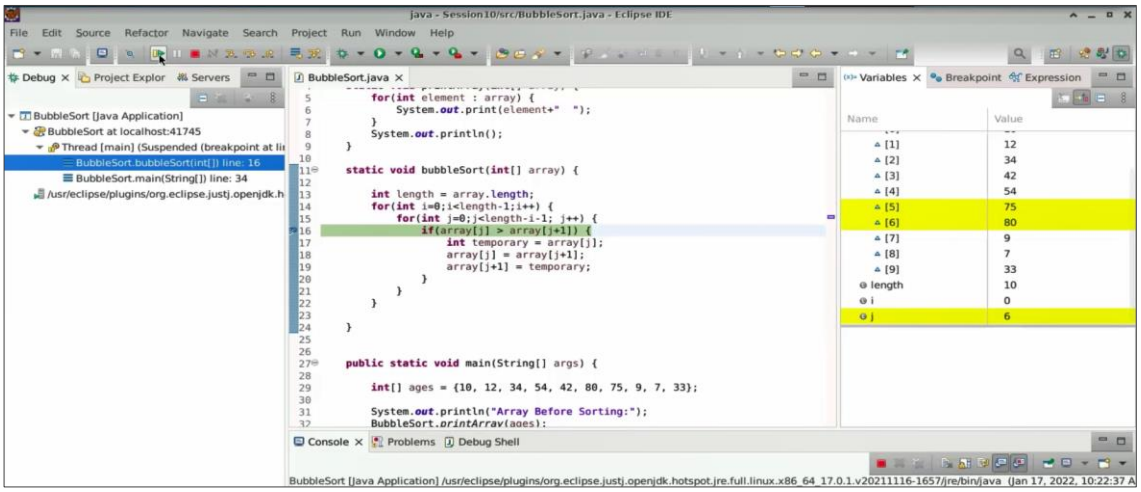
5.4 You are going to open other debugging perspectives



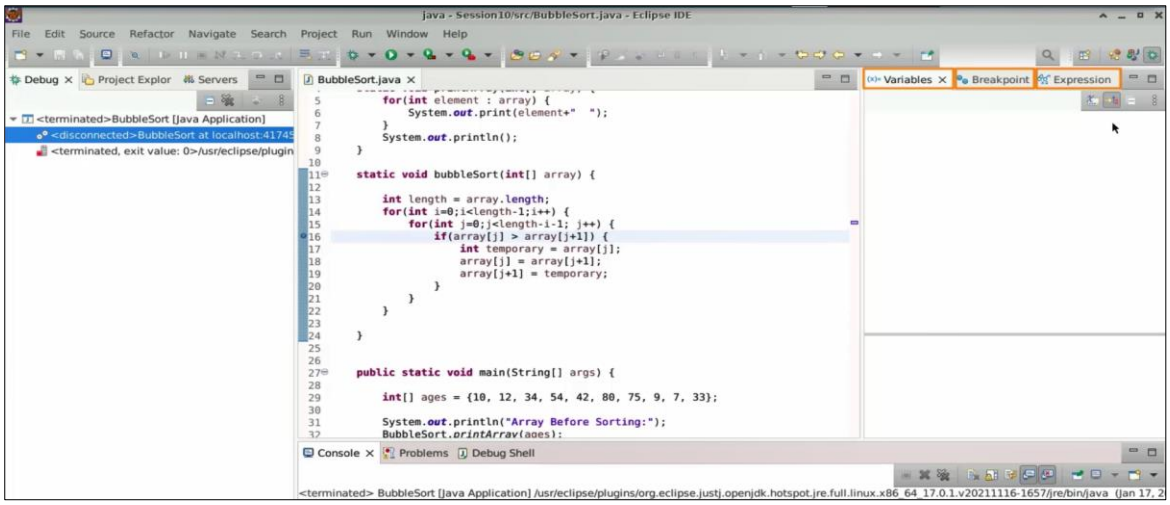
5.5 When you are executing this program, the array is there right with the data having the zero till the 9th index. The length is 10, i is zero and j is 0 at this point. These are the variables and you do have the values



5.6 Let us resume now. The j value is 1, and then it becomes 2, leading to swaps and manipulations within the array. Wherever a swap occurs, the updated value of the array is displayed. In the very first pass, the largest element automatically moves to the end



5.7 All these algorithmic approaches, you must observe through the debugger attached. You can Simply put a line break and rather than executing your program normally, you can simply attach a debugger view. Remember, it is a debug explorer, and you will have variable breakpoints or the expressions shown on the right-hand side. You can see here; that the perspective is the debug perspective



By following the above steps, you have successfully sorted an array using methods and a bubble sort algorithm.