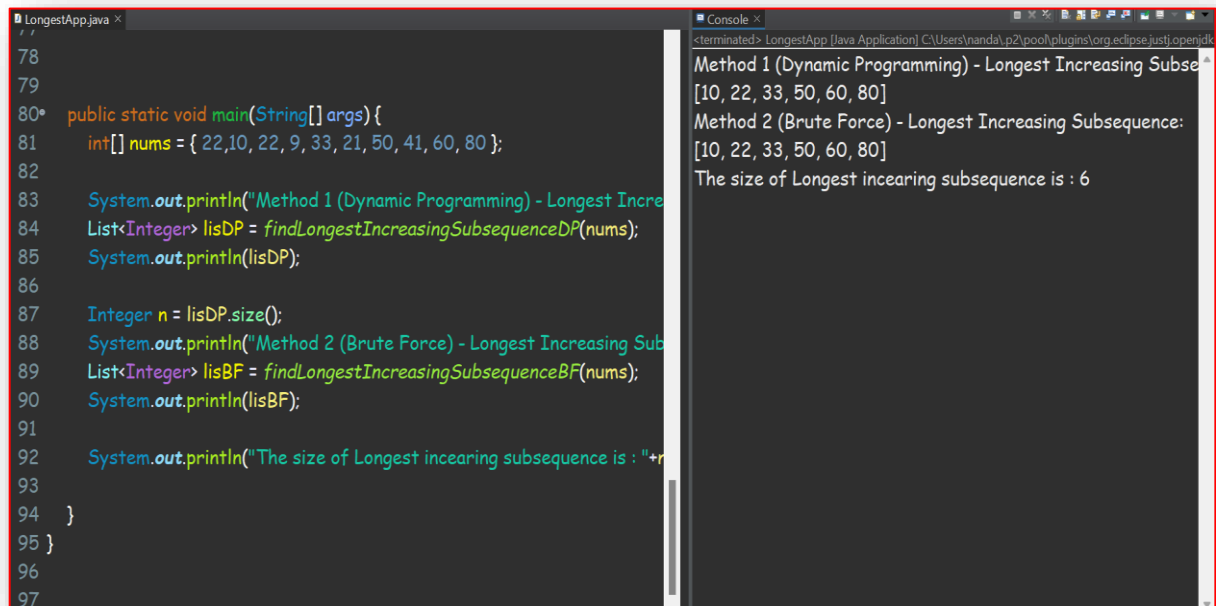


Different Output for Various Input

Case 01:-Giving +ve Input

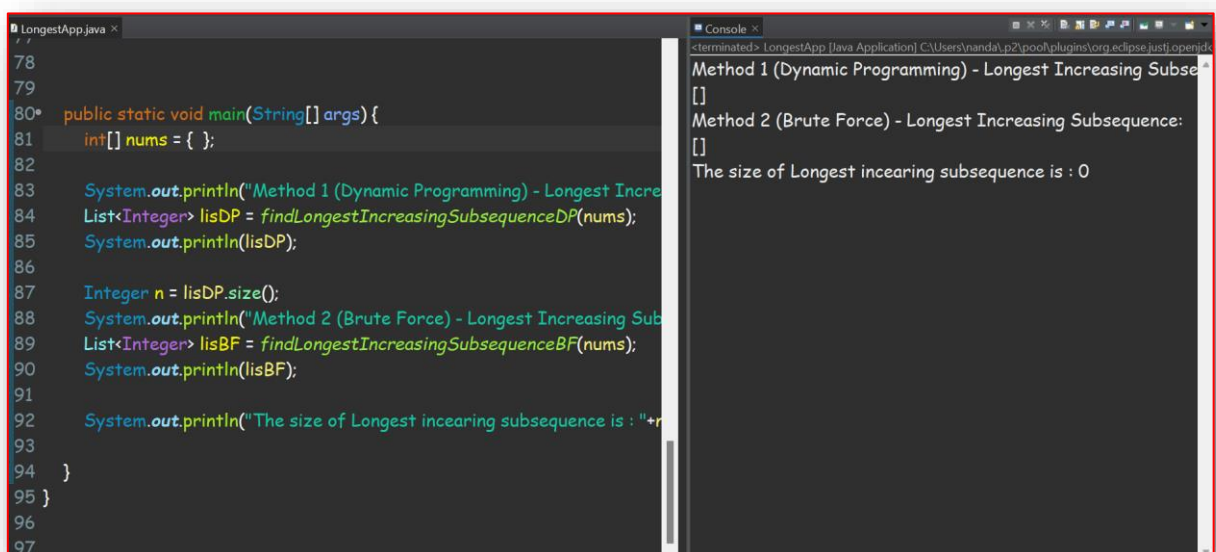


The screenshot shows an IDE with a Java file named 'LongestApp.java' and a console window. The code defines a main method that takes an array of integers and uses two methods to find the longest increasing subsequence: 'findLongestIncreasingSubsequenceDP' (Dynamic Programming) and 'findLongestIncreasingSubsequenceBF' (Brute Force). The input array is {22, 10, 22, 9, 33, 21, 50, 41, 60, 80}. The console output shows the results for both methods, which both return the subsequence [10, 22, 33, 50, 60, 80] and state that the size of the longest increasing subsequence is 6.

```
78
79
80* public static void main(String[] args) {
81     int[] nums = { 22, 10, 22, 9, 33, 21, 50, 41, 60, 80 };
82
83     System.out.println("Method 1 (Dynamic Programming) - Longest Incre
84     List<Integer> lisDP = findLongestIncreasingSubsequenceDP(nums);
85     System.out.println(lisDP);
86
87     Integer n = lisDP.size();
88     System.out.println("Method 2 (Brute Force) - Longest Increasing Sub
89     List<Integer> lisBF = findLongestIncreasingSubsequenceBF(nums);
90     System.out.println(lisBF);
91
92     System.out.println("The size of Longest incearing subsequence is : "+n
93
94 }
95 }
96
97
```

Method 1 (Dynamic Programming) - Longest Increasing Subsequence: [10, 22, 33, 50, 60, 80]
Method 2 (Brute Force) - Longest Increasing Subsequence: [10, 22, 33, 50, 60, 80]
The size of Longest incearing subsequence is : 6

Case 02:-Giving Empty Input

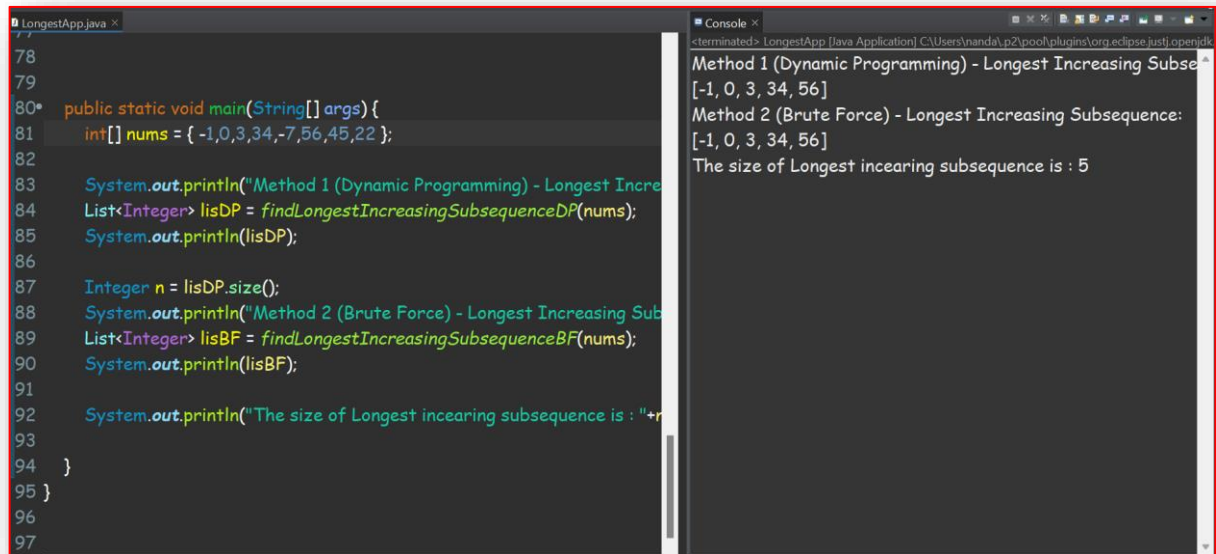


The screenshot shows the same IDE setup as Case 01, but with an empty input array. The code is identical, but the input array is now {}. The console output shows that both methods return an empty list [] and that the size of the longest increasing subsequence is 0.

```
78
79
80* public static void main(String[] args) {
81     int[] nums = { };
82
83     System.out.println("Method 1 (Dynamic Programming) - Longest Incre
84     List<Integer> lisDP = findLongestIncreasingSubsequenceDP(nums);
85     System.out.println(lisDP);
86
87     Integer n = lisDP.size();
88     System.out.println("Method 2 (Brute Force) - Longest Increasing Sub
89     List<Integer> lisBF = findLongestIncreasingSubsequenceBF(nums);
90     System.out.println(lisBF);
91
92     System.out.println("The size of Longest incearing subsequence is : "+n
93
94 }
95 }
96
97
```

Method 1 (Dynamic Programming) - Longest Increasing Subsequence: []
Method 2 (Brute Force) - Longest Increasing Subsequence: []
The size of Longest incearing subsequence is : 0

Case 03:-Giving -ve integer Input



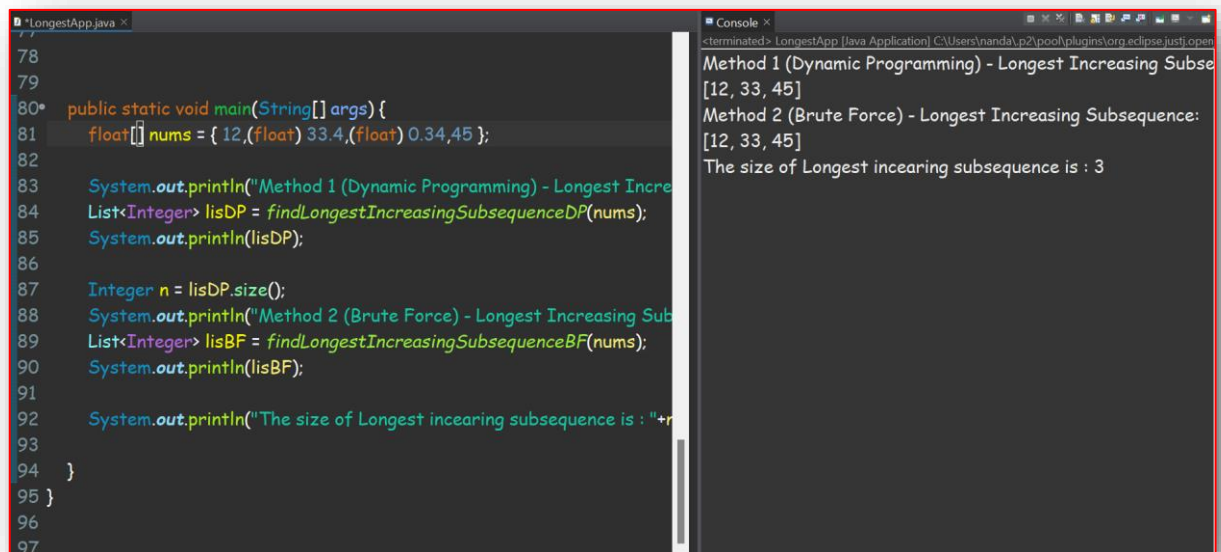
The screenshot shows an IDE with a Java file named 'LongestApp.java' and a console window. The code defines a main method that takes an array of integers and uses two methods to find the longest increasing subsequence: 'findLongestIncreasingSubsequenceDP' (Dynamic Programming) and 'findLongestIncreasingSubsequenceBF' (Brute Force). The input array is [-1, 0, 3, 34, -7, 56, 45, 22]. The console output shows the results for both methods, which both return the subsequence [-1, 0, 3, 34, 56] and a size of 5.

```
78
79
80* public static void main(String[] args) {
81     int[] nums = { -1,0,3,34,-7,56,45,22 };
82
83     System.out.println("Method 1 (Dynamic Programming) - Longest Increasing Subsequence:");
84     List<Integer> lisDP = findLongestIncreasingSubsequenceDP(nums);
85     System.out.println(lisDP);
86
87     Integer n = lisDP.size();
88     System.out.println("Method 2 (Brute Force) - Longest Increasing Subsequence:");
89     List<Integer> lisBF = findLongestIncreasingSubsequenceBF(nums);
90     System.out.println(lisBF);
91
92     System.out.println("The size of Longest inearing subsequence is : "+n);
93 }
94 }
95 }
96 }
97 }
```

Console Output:

```
<terminated> LongestApp [Java Application] C:\Users\nanda\p2\pool\plugins\org.eclipse.justi.openjdk
Method 1 (Dynamic Programming) - Longest Increasing Subsequence:
[-1, 0, 3, 34, 56]
Method 2 (Brute Force) - Longest Increasing Subsequence:
[-1, 0, 3, 34, 56]
The size of Longest inearing subsequence is : 5
```

Case 04:-Type casted Input



The screenshot shows an IDE with a Java file named 'LongestApp.java' and a console window. The code is similar to Case 03, but the input array is of type 'float' and contains values [12, 33.4, 0.34, 45]. The console output shows the results for both methods, which both return the subsequence [12, 33, 45] and a size of 3.

```
78
79
80* public static void main(String[] args) {
81     float[] nums = { 12,(float) 33.4,(float) 0.34,45 };
82
83     System.out.println("Method 1 (Dynamic Programming) - Longest Increasing Subsequence:");
84     List<Integer> lisDP = findLongestIncreasingSubsequenceDP(nums);
85     System.out.println(lisDP);
86
87     Integer n = lisDP.size();
88     System.out.println("Method 2 (Brute Force) - Longest Increasing Subsequence:");
89     List<Integer> lisBF = findLongestIncreasingSubsequenceBF(nums);
90     System.out.println(lisBF);
91
92     System.out.println("The size of Longest inearing subsequence is : "+n);
93 }
94 }
95 }
96 }
97 }
```

Console Output:

```
<terminated> LongestApp [Java Application] C:\Users\nanda\p2\pool\plugins\org.eclipse.justi.openjdk
Method 1 (Dynamic Programming) - Longest Increasing Subsequence:
[12, 33, 45]
Method 2 (Brute Force) - Longest Increasing Subsequence:
[12, 33, 45]
The size of Longest inearing subsequence is : 3
```