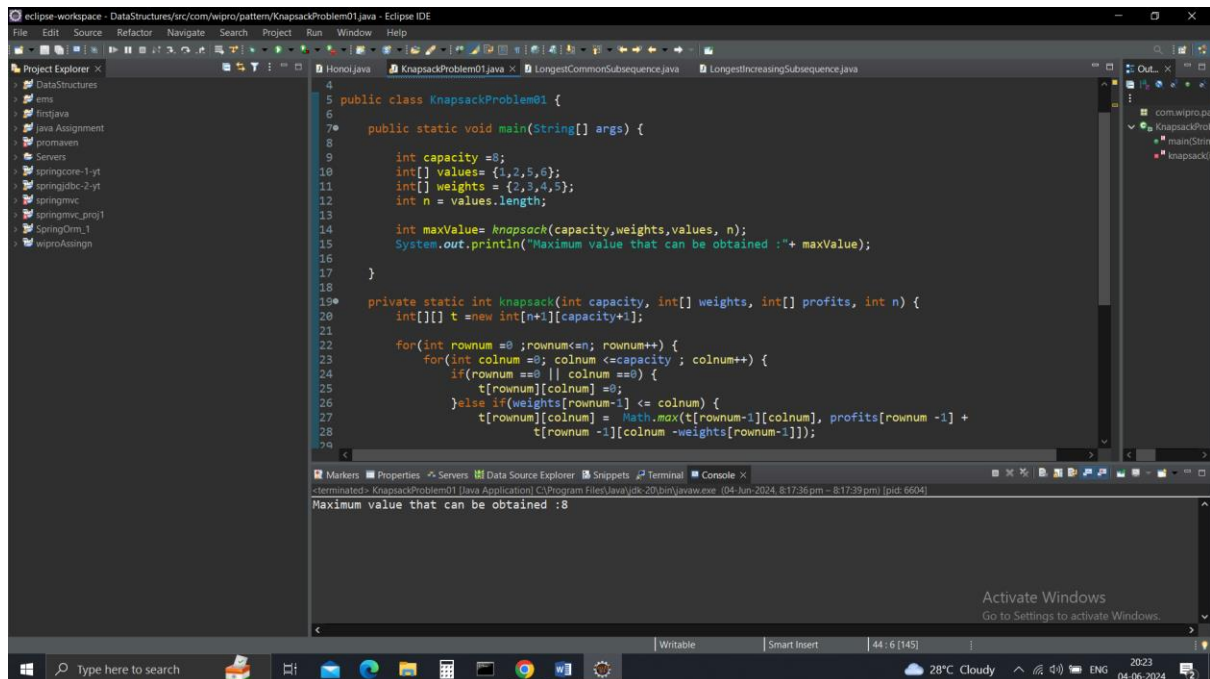


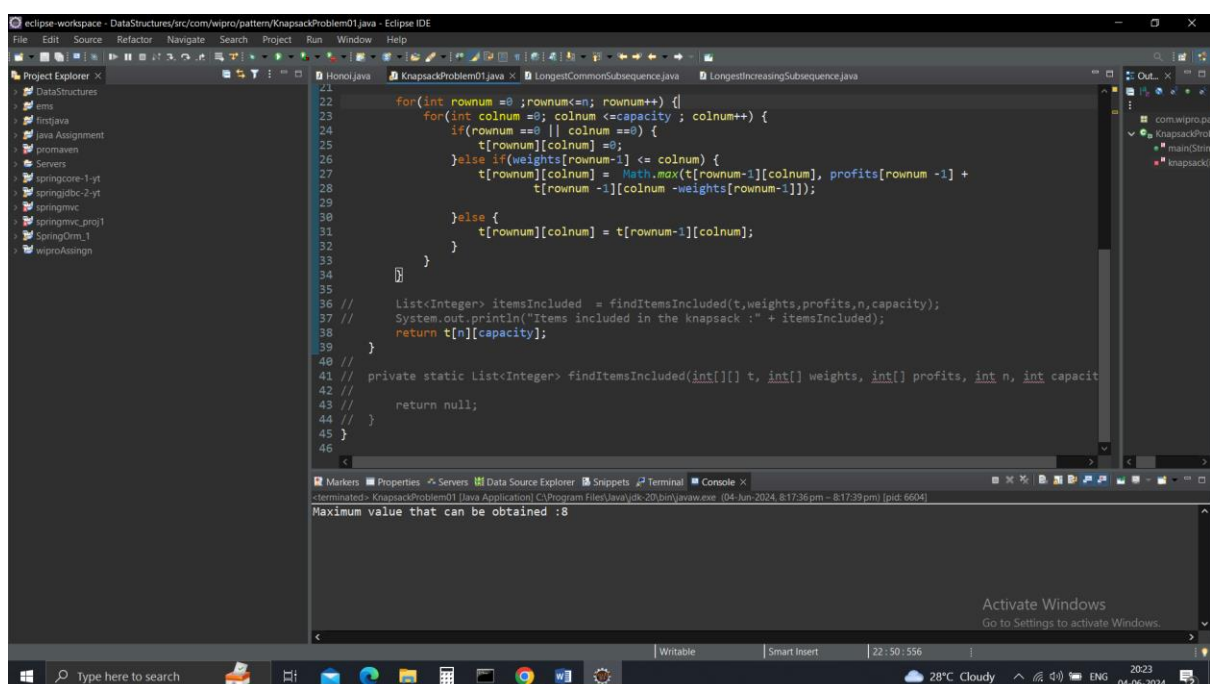
## Task 1: Knapsack Problem

Write a function `int Knapsack(int W, int[] weights, int[] values)` in C# that determines the maximum value of items that can fit into a knapsack with a capacity `W`. The function should handle up to 100 items. Find the optimal way to fill the knapsack with the given items to achieve the maximum total value. You must consider that you cannot break items, but have to include them whole.



```
4 public class KnapsackProblem01 {
5
6     public static void main(String[] args) {
7
8         int capacity = 8;
9         int[] values = {1,2,5,6};
10        int[] weights = {2,3,4,5};
11        int n = values.length;
12
13        int maxValue = Knapsack(capacity, weights, values, n);
14        System.out.println("Maximum value that can be obtained :"+ maxValue);
15    }
16
17    private static int knapsack(int capacity, int[] weights, int[] profits, int n) {
18        int[][] t = new int[n+1][capacity+1];
19
20        for(int rownum = 0; rownum <= n; rownum++) {
21            for(int colnum = 0; colnum <= capacity; colnum++) {
22                if(rownum == 0 || colnum == 0) {
23                    t[rownum][colnum] = 0;
24                } else if(weights[rownum-1] <= colnum) {
25                    t[rownum][colnum] = Math.max(t[rownum-1][colnum], profits[rownum-1] +
26                        t[rownum-1][colnum - weights[rownum-1]]);
27                }
28            }
29        }
30    }
31}
```

Maximum value that can be obtained :8

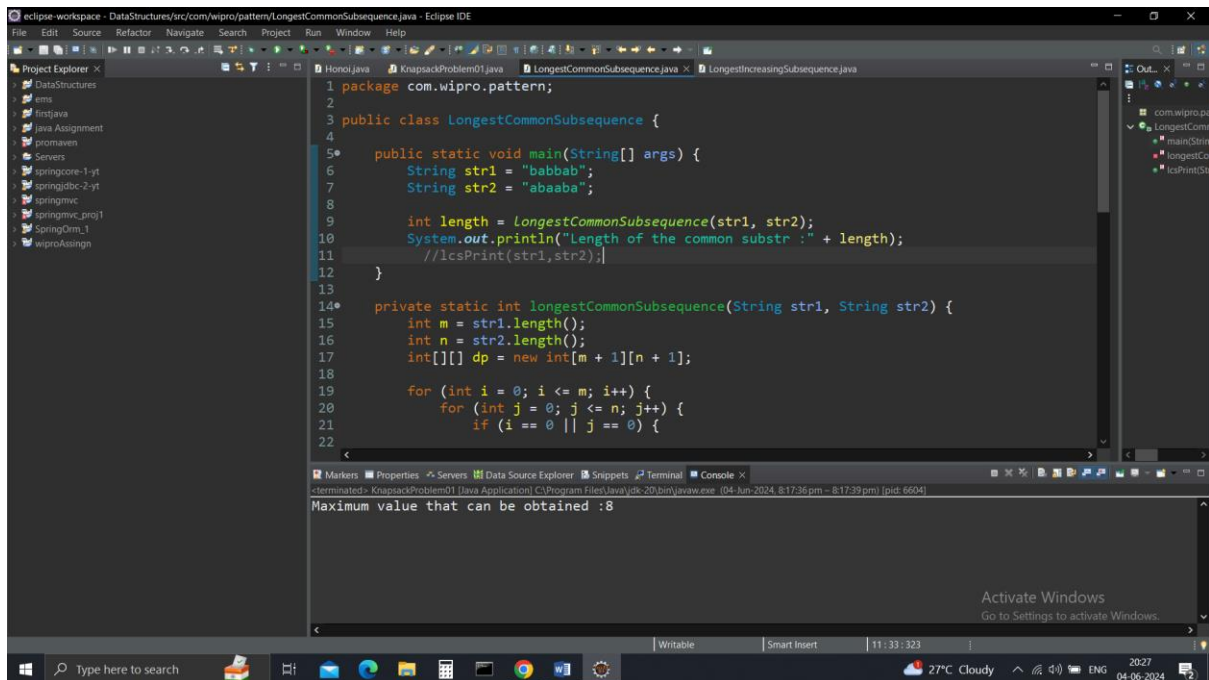


```
21 // List<Integer> itemsIncluded = findItemsIncluded(t, weights, profits, n, capacity);
22 // System.out.println("Items included in the knapsack : " + itemsIncluded);
23 // return t[n][capacity];
24 // }
25 // }
26 // }
27 // }
28 // }
29 // }
30 // }
31 // }
32 // }
33 // }
34 // }
35 // }
36 // }
37 // }
38 // }
39 // }
40 // }
41 // }
42 // }
43 // }
44 // }
45 // }
46 // }
```

Maximum value that can be obtained :8

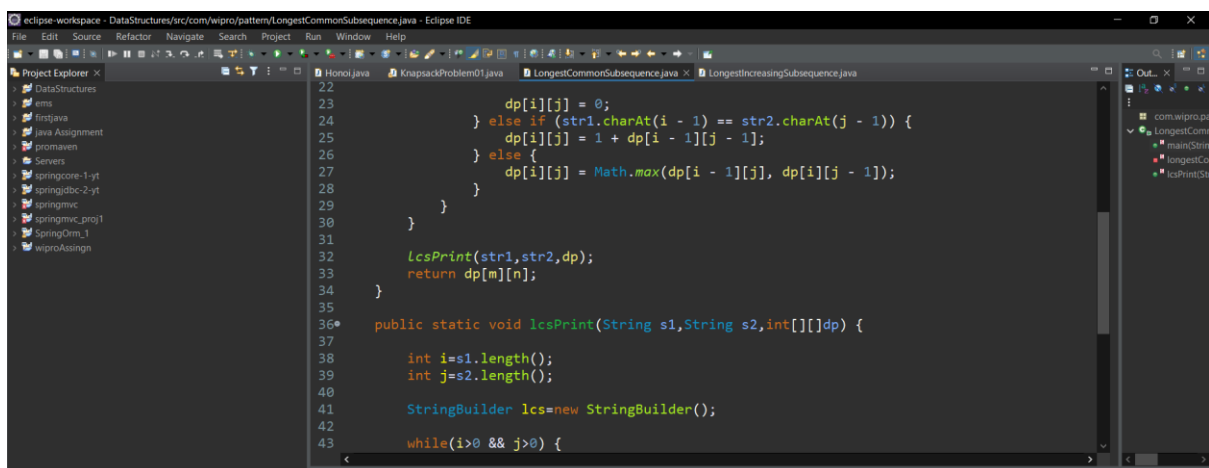
## Task 2: Longest Common Subsequence

Implement `int LCS(string text1, string text2)` to find the length of the longest common subsequence between two strings.



```
1 package com.wipro.pattern;
2
3 public class LongestCommonSubsequence {
4
5     public static void main(String[] args) {
6         String str1 = "babbar";
7         String str2 = "abaaba";
8
9         int length = LongestCommonSubsequence(str1, str2);
10        System.out.println("Length of the common substr : " + length);
11        //lcsPrint(str1, str2);
12    }
13
14    private static int longestCommonSubsequence(String str1, String str2) {
15        int m = str1.length();
16        int n = str2.length();
17        int[][] dp = new int[m + 1][n + 1];
18
19        for (int i = 0; i <= m; i++) {
20            for (int j = 0; j <= n; j++) {
21                if (i == 0 || j == 0) {
22
```

Maximum value that can be obtained :8



```
22        dp[i][j] = 0;
23    } else if (str1.charAt(i - 1) == str2.charAt(j - 1)) {
24        dp[i][j] = 1 + dp[i - 1][j - 1];
25    } else {
26        dp[i][j] = Math.max(dp[i - 1][j], dp[i][j - 1]);
27    }
28    }
29    }
30    }
31
32    lcsPrint(str1, str2, dp);
33    return dp[m][n];
34    }
35
36    public static void lcsPrint(String s1, String s2, int[][] dp) {
37
38        int i = s1.length();
39        int j = s2.length();
40
41        StringBuilder lcs = new StringBuilder();
42
43        while (i > 0 && j > 0) {
```

The screenshot shows the Eclipse IDE with the file `LongestCommonSubsequence.java` open. The `lcsPrint` method is implemented as follows:

```
34 }
35
36 public static void lcsPrint(String s1,String s2,int[][]dp) {
37
38     int i=s1.length();
39     int j=s2.length();
40
41     StringBuilder lcs=new StringBuilder();
42
43     while(i>0 && j>0) {
44         if(s1.charAt(i-1)==s2.charAt(j-1))
45         {
46             lcs.append(s1.charAt(i-1));
47             i--;
48             j--;
49         }
50         else if (dp[i-1][j]>dp[i][j-1]) {
51             i--;
52         }
53         else {
54             j--;
55         }
56     }
57 }
```

The screenshot shows the Eclipse IDE with the `lcsPrint` method completed. The final code is:

```
41     StringBuilder lcs=new StringBuilder();
42
43     while(i>0 && j>0) {
44         if(s1.charAt(i-1)==s2.charAt(j-1))
45         {
46             lcs.append(s1.charAt(i-1));
47             i--;
48             j--;
49         }
50         else if (dp[i-1][j]>dp[i][j-1]) {
51             i--;
52         }
53         else {
54             j--;
55         }
56     }
57
58     System.out.println(lcs.reverse().toString());
59 }
60 }
61 }
62 }
```

The console output shows the result of the program execution:

```
abab
Length of the common substr :4
```