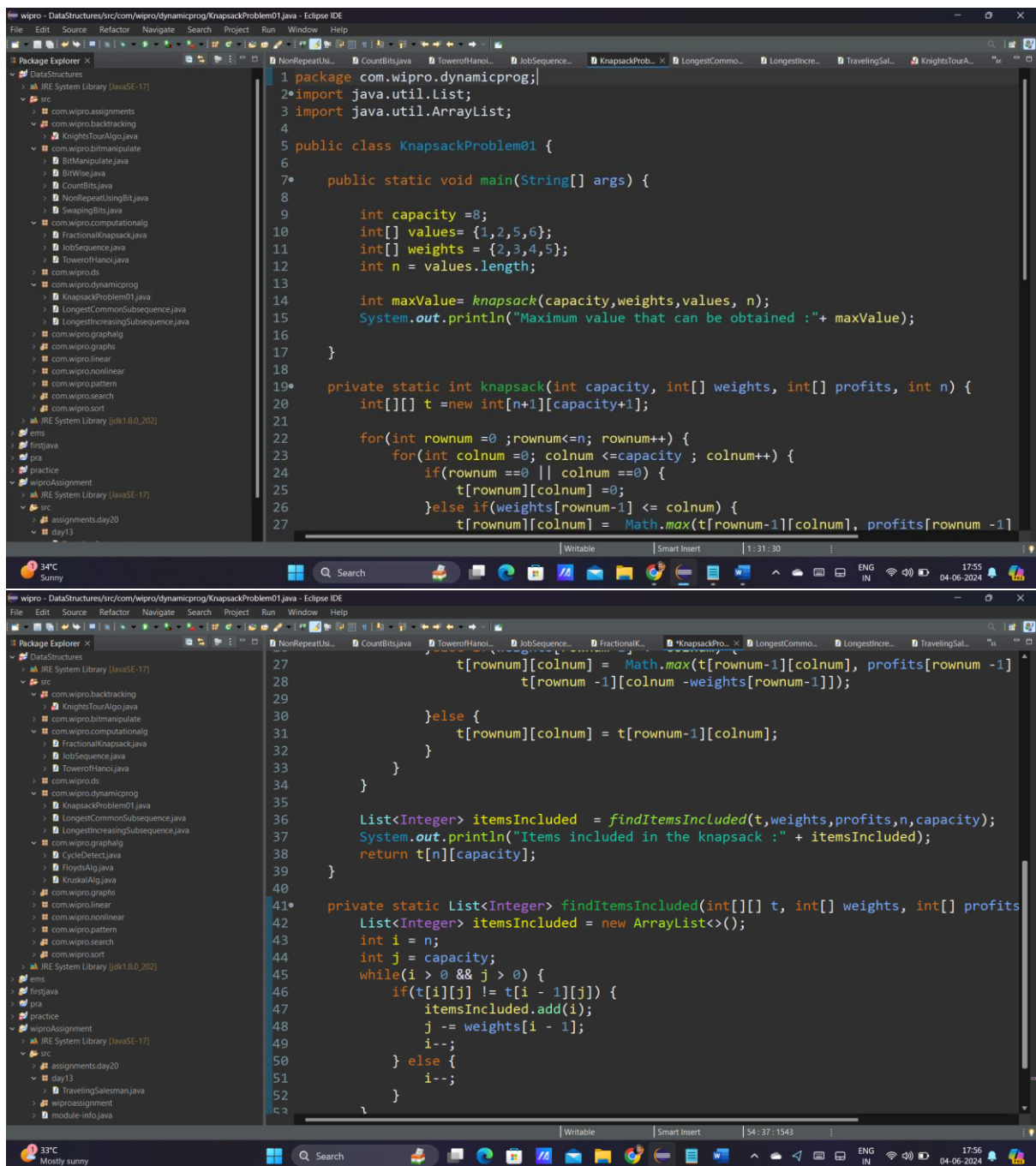


Assignments-Day15,16

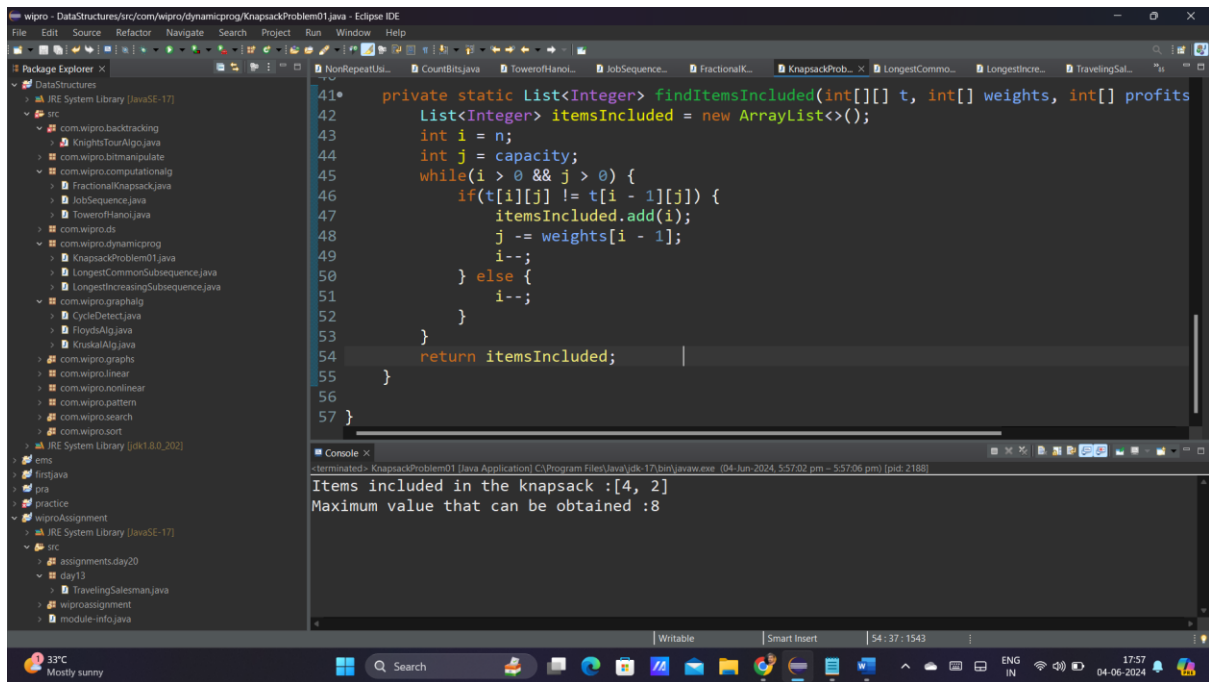
Day 15 and 16:

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.



```
1 package com.wipro.dynamicprog;
2 import java.util.List;
3 import java.util.ArrayList;
4
5 public class KnapsackProblem01 {
6
7     public static void main(String[] args) {
8
9         int capacity = 8;
10        int[] values = {1,2,5,6};
11        int[] weights = {2,3,4,5};
12        int n = values.length;
13
14        int maxValue = knapsack(capacity, weights, values, n);
15        System.out.println("Maximum value that can be obtained : "+ maxValue);
16    }
17
18    private static int knapsack(int capacity, int[] weights, int[] profits, int n) {
19        int[][] t = new int[n+1][capacity+1];
20
21        for(int rownum = 0; rownum <= n; rownum++) {
22            for(int colnum = 0; colnum <= capacity; colnum++) {
23                if(rownum == 0 || colnum == 0) {
24                    t[rownum][colnum] = 0;
25                } else if(weights[rownum-1] <= colnum) {
26                    t[rownum][colnum] = Math.max(t[rownum-1][colnum], profits[rownum-1] + t[rownum-1][colnum - weights[rownum-1]]);
27                } else {
28                    t[rownum][colnum] = t[rownum-1][colnum];
29                }
30            }
31        }
32
33        List<Integer> itemsIncluded = findItemsIncluded(t, weights, profits, n, capacity);
34        System.out.println("Items included in the knapsack : " + itemsIncluded);
35        return t[n][capacity];
36    }
37
38    private static List<Integer> findItemsIncluded(int[][] t, int[] weights, int[] profits, int n, int capacity) {
39        List<Integer> itemsIncluded = new ArrayList<>();
40        int i = n;
41        int j = capacity;
42        while(i > 0 && j > 0) {
43            if(t[i][j] != t[i-1][j]) {
44                itemsIncluded.add(i);
45                j -= weights[i-1];
46                i--;
47            } else {
48                i--;
49            }
50        }
51        return itemsIncluded;
52    }
53 }
```



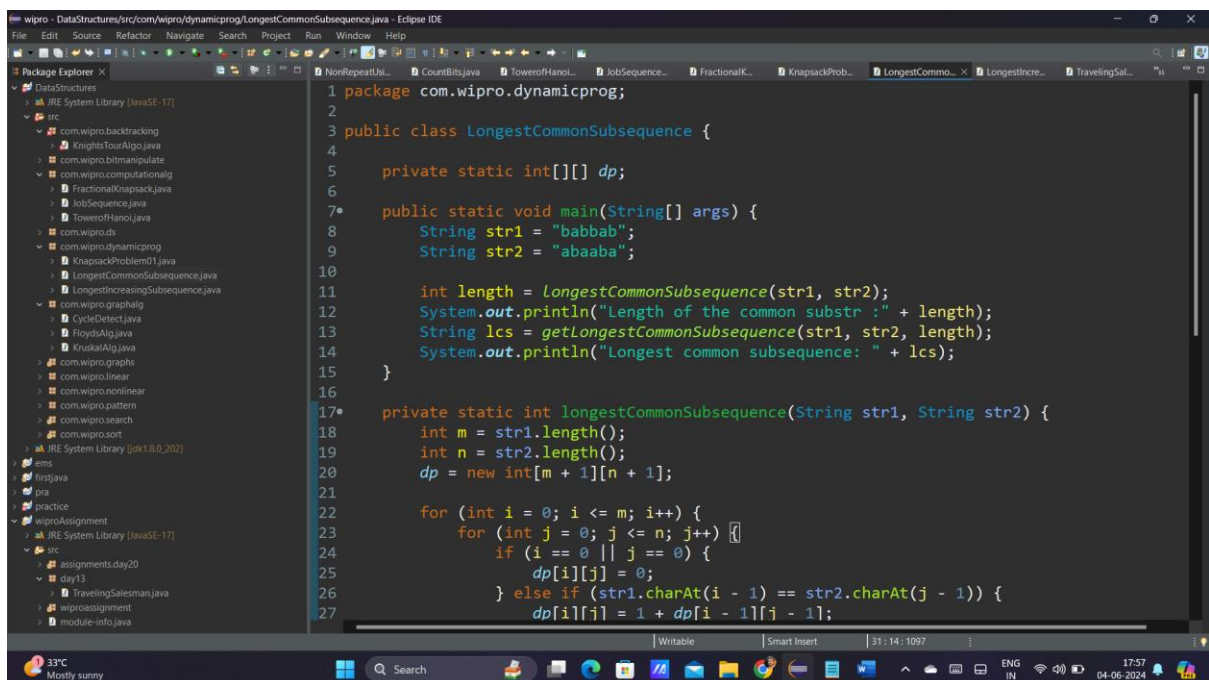
```
41* private static List<Integer> findItemsIncluded(int[][] t, int[] weights, int[] profits
42     List<Integer> itemsIncluded = new ArrayList<>();
43     int i = n;
44     int j = capacity;
45     while(i > 0 && j > 0) {
46         if(t[i][j] != t[i - 1][j]) {
47             itemsIncluded.add(i);
48             j -= weights[i - 1];
49             i--;
50         } else {
51             i--;
52         }
53     }
54     return itemsIncluded;
55 }
56
57 }
```

Console Output:

```
<terminated> KnapsackProblem01 [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (04-Jun-2024 5:57:02 pm - 5:57:06 pm) [pid: 2188]
Items included in the knapsack :[4, 2]
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.dynamicprog;
2
3 public class LongestCommonSubsequence {
4
5     private static int[][] dp;
6
7* public static void main(String[] args) {
8     String str1 = "babbbab";
9     String str2 = "abaaba";
10
11     int length = LongestCommonSubsequence(str1, str2);
12     System.out.println("Length of the common substr : " + length);
13     String lcs = getLongestCommonSubsequence(str1, str2, length);
14     System.out.println("Longest common subsequence: " + lcs);
15 }
16
17* private static int longestCommonSubsequence(String str1, String str2) {
18     int m = str1.length();
19     int n = str2.length();
20     dp = new int[m + 1][n + 1];
21
22     for (int i = 0; i <= m; i++) {
23         for (int j = 0; j <= n; j++) {
24             if (i == 0 || j == 0) {
25                 dp[i][j] = 0;
26             } else if (str1.charAt(i - 1) == str2.charAt(j - 1)) {
27                 dp[i][j] = 1 + dp[i - 1][j - 1];
```

```
wipro - DataStructures/src/com/wipro/dynamicprog/LongestCommonSubsequence.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer
DataStructures
  IRE System Library (JavaSE-17)
  src
    com.wipro.backtracking
      KnightsTourAlgo.java
    com.wipro.bitmanipulate
    com.wipro.computationalg
      FractionalKnapsack.java
      JobSequence.java
      TowerofHanoi.java
    com.wipro.ds
    com.wipro.dynamicprog
      KnapsackProblem01.java
      LongestCommonSubsequence.java
      LongestIncreasingSubsequence.java
    com.wipro.graphalg
      CycleDetect.java
      FloydAlg.java
      KruskalAlg.java
    com.wipro.graphs
    com.wipro.linear
    com.wipro.nonlinear
    com.wipro.pattern
    com.wipro.search
    com.wipro.sort
  IRE System Library (jdk1.8.0_202)
  ems
  firstjava
  ora
  practice
  wiproAssignment
    IRE System Library (JavaSE-17)
    src
      assignments.day20
      day13
      TravelingSalesman.java
      wiproassignment
      module-info.java

28         } else {
29             dp[i][j] = Math.max(dp[i - 1][j], dp[i][j - 1]);
30         }
31     }
32     }
33     return dp[m][n];
34 }
35
36* private static String getLongestCommonSubsequence(String str1, String str2, int length
37     int m = str1.length();
38     int n = str2.length();
39
40     char[] lcs = new char[length];
41     int index = length - 1;
42
43     int i = m, j = n;
44     while (i > 0 && j > 0) {
45         if (str1.charAt(i - 1) == str2.charAt(j - 1)) {
46             lcs[index--] = str1.charAt(i - 1);
47             i--;
48             j--;
49         } else if (dp[i - 1][j] > dp[i][j - 1]) {
50             i--;
51         } else {
52             j--;
53         }
54     }
55 }
```

```
wipro - DataStructures/src/com/wipro/dynamicprog/LongestCommonSubsequence.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer
DataStructures
  IRE System Library (JavaSE-17)
  src
    com.wipro.backtracking
      KnightsTourAlgo.java
    com.wipro.bitmanipulate
    com.wipro.computationalg
      FractionalKnapsack.java
      JobSequence.java
      TowerofHanoi.java
    com.wipro.ds
    com.wipro.dynamicprog
      KnapsackProblem01.java
      LongestCommonSubsequence.java
      LongestIncreasingSubsequence.java
    com.wipro.graphalg
      CycleDetect.java
      FloydAlg.java
      KruskalAlg.java
    com.wipro.graphs
    com.wipro.linear
    com.wipro.nonlinear
    com.wipro.pattern
    com.wipro.search
    com.wipro.sort
  IRE System Library (jdk1.8.0_202)
  ems
  firstjava
  ora
  practice
  wiproAssignment
    IRE System Library (JavaSE-17)
    src
      assignments.day20
      day13
      TravelingSalesman.java
      wiproassignment
      module-info.java

41     int index = length - 1;
42
43     int i = m, j = n;
44     while (i > 0 && j > 0) {
45         if (str1.charAt(i - 1) == str2.charAt(j - 1)) {
46             lcs[index--] = str1.charAt(i - 1);
47             i--;
48             j--;
49         } else if (dp[i - 1][j] > dp[i][j - 1]) {
50             i--;
51         } else {
52             j--;
53         }
54     }
55     return String.valueOf(lcs);
56 }
57 }
```

```
Console
-terminated- LongestCommonSubsequence [Java Application] C:\Program Files\Java\jdk-17\bin\java.exe (04-Jun-2024 5:58:39 pm - 5:58:41 pm) [pid: 30460]
Length of the common substr :4
Longest common subsequence: abab
```