

Ocean Lu
CS 4080.02
Professor Yang
10/29/19

Assignment #5

Task 1: <https://colab.research.google.com/drive/1eHqLmLP3Ni0UaR-MKmjCFFfAU6qsJCyY6>

Program source code: Python

```
# Initialize the input data in the program instead of entering from keyboard or reading from file.
import pandas as pd

df = pd.DataFrame([["Dan", 45], ["Adam", 39], ["Fiona", 42], ["Kathy", 44], ["Dan", 34], ["Adam", 41],
                  ["Zehr", 43], ["Mona", 42], ["Kevin", 35], ["Elma", 48]], columns=list('AB'))
print("Original Data\n", df, "\n")

# (1) sort in ascending order of names, if same name ascending order of scores;
part1 = df.sort_values(by=['A', 'B'], ascending=[True, True])
print("Sort in ascending order of names, if same name ascending order of scores\n", part1, "\n")

# (2) sort in ascending order of names, if same name descending order of scores;
part2 = df.sort_values(by=['A', 'B'], ascending=[True, False])
print("Sort in ascending order of names, if same name descending order of scores\n", part2, "\n")

# (3) sort in descending order of names, if same name descending order of scores.
part3 = df.sort_values(by=['A', 'B'], ascending=[False, False])
print("Sort in descending order of names, if same name descending order of scores\n", part3, "\n")
```

Output runs:

| Original Data | | |
|---------------|-------|----|
| | A | B |
| 0 | Dan | 45 |
| 1 | Adam | 39 |
| 2 | Fiona | 42 |
| 3 | Kathy | 44 |
| 4 | Dan | 34 |
| 5 | Adam | 41 |
| 6 | Zehr | 43 |
| 7 | Mona | 42 |
| 8 | Kevin | 35 |
| 9 | Elma | 48 |

| Sort in ascending order of names, if same name ascending order of scores | | |
|--|-------|----|
| | A | B |
| 1 | Adam | 39 |
| 5 | Adam | 41 |
| 4 | Dan | 34 |
| 0 | Dan | 45 |
| 9 | Elma | 48 |
| 2 | Fiona | 42 |
| 3 | Kathy | 44 |
| 8 | Kevin | 35 |
| 7 | Mona | 42 |
| 6 | Zehr | 43 |

Sort in ascending order of names, if same name descending order of scores

| | A | B |
|---|-------|----|
| 5 | Adam | 41 |
| 1 | Adam | 39 |
| 0 | Dan | 45 |
| 4 | Dan | 34 |
| 9 | Elma | 48 |
| 2 | Fiona | 42 |
| 3 | Kathy | 44 |
| 8 | Kevin | 35 |
| 7 | Mona | 42 |
| 6 | Zehr | 43 |

Sort in descending order of names, if same name descending order of scores

| | A | B |
|---|-------|----|
| 6 | Zehr | 43 |
| 7 | Mona | 42 |
| 8 | Kevin | 35 |
| 3 | Kathy | 44 |
| 2 | Fiona | 42 |
| 9 | Elma | 48 |
| 0 | Dan | 45 |
| 4 | Dan | 34 |
| 5 | Adam | 41 |
| 1 | Adam | 39 |

Task 2:

Program Source Code: Java

```
16 public class Test {
17
18     static int findIntMax(int mat[][]) {
19         int maxElement = mat[0][0];
20         for (int i = 0; i < mat.length; i++) {
21             for (int j = 0; j < mat[i].length; j++) {
22                 if (mat[i][j] > maxElement) {
23                     maxElement = mat[i][j];
24                 }
25             }
26         }
27         return maxElement;
28     }
29
30     static void findIntIndex(int mat[][], int maxElement) {
31         boolean found = false;
32         for (int i = 0; i < mat.length; i++) {
33             for (int j = 0; j < mat[i].length; j++) {
34                 if (mat[i][j] == maxElement) {
35                     System.out.println("Index at: [" + i + ", " + j + "]");
36                     found = true;
37                     break;
38                 }
39             }
40             if (found) {
41                 break;
42             }
43         }
44     }
45
46     static String findStringMax(String mat[][]) {
47         String maxElement = mat[0][0];
48         for (int i = 0; i < mat.length; i++) {
49             for (int j = 0; j < mat[i].length; j++) {
50                 if ((maxElement.compareTo(mat[i][j])) < 0) {
51                     maxElement = mat[i][j];
52                 }
53             }
54         }
55         return maxElement;
56     }
57
58     static void findStringIndex(String mat[][], String maxElement) {
59         boolean found = false;
60         for (int i = 0; i < mat.length; i++) {
61             for (int j = 0; j < mat[i].length; j++) {
62                 if (mat[i][j].equals(maxElement)) {
63
64                     System.out.println("Index at: [" + i + ", " + j + "]");
65                     found = true;
66                     break;
67                 }
68             }
69             if (found) {
70                 break;
71             }
72         }
73     }
74
75     public static void main(String[] args) {
76         int integers[][] = {{1,2,4,4},{5,5,4,2},{3,1,1,5}};
77         System.out.println("Largest Element: " + findIntMax(integers));
78         findIntIndex(integers, findIntMax(integers));
79
80         String strings[][] = {{ "Apple", "Banana", "Pork", "Beef"}, {"David",
81             "Kelin", "Peter", "Zag", "Diana"}, {"Elin", "Adam", "Young",
82             "Peter", "Zag"}};
83         System.out.println("Largest Element: " + findStringMax(strings));
84         findStringIndex(strings, findStringMax(strings));
85     }
86 }
```

Output:

```
Output - test (run) x
run:
Largest Element: 5
Index at: [1, 0]
Largest Element: Zag
Index at: [1, 3]
BUILD SUCCESSFUL (total time: 0 seconds)
```

Tests:

Test to find the max value in a 2D array:

```
16 public class Test {
17
18     static int findMax(int mat[][]) {
19         // Initializing max element as INT_MIN
20         int maxElement = Integer.MIN_VALUE;
21         // checking each element of matrix
22         // if it is greater than maxElement,
23         // update maxElement
24         for (int i = 0; i < mat.length; i++) {
25             for (int j = 0; j < mat[i].length; j++) {
26                 if (mat[i][j] > maxElement) {
27                     maxElement = mat[i][j];
28                 }
29             }
30         }
31         // finally return maxElement
32         return maxElement;
33     }
34
35     public static void main(String[] args) {
36         int mat[][] = { { 1, 2, 3, 4 },
37                         { 25, 6, 7, 8 },
38                         { 9, 10, 11, 12 },
39                         { 13, 14, 15, 16 } };
40
41         System.out.println(findMax(mat)) ;
42     }
43 }
```

```
Output - test (run) x
run:
25
BUILD SUCCESSFUL (total time: 0 seconds)
```

Test to find index (all), will be edited to return the first:

```

35 static void findIndex(int mat[][], int maxElement) {
36     for (int i = 0; i < mat.length; i++) {
37         for (int j = 0; j < mat[i].length; j++) {
38             if (mat[i][j] == (maxElement)) {
39                 System.out.println("Index at: [" + i + ", " + j + "]");
40             }
41         }
42     }
43 }
44
45 public static void main(String[] args) {
46     int mat[][] = { { 1, 25, 3, 4 },
47                     { 25, 6, 7, 8 },
48                     { 9, 10, 11, 12 },
49                     { 13, 14, 15, 16 } };
50
51     System.out.println(findIntMax(mat));
52     findIndex(mat, findIntMax(mat));
53 }
54
55

```

Output - test (run) ×

```

run:
25
Index at: [0, 1]
Index at: [1, 0]
BUILD SUCCESSFUL (total time: 0 seconds)

```

Testing given integer data:

```

46 public static void main(String[] args) {
47     int integers[][] = {{1,2,4,4},{5,5,4,2},{3,1,1,5}};
48     System.out.println("Largest Element:" + findIntMax(integers));
49     findIntIndex(integers, findIntMax(integers));
50 }
51
52

```

Output - test (run) ×

```

run:
Largest Number:5
Index at: [1, 0]
BUILD SUCCESSFUL (total time: 0 seconds)

```

Testing max value of String in 2D array:

```

46 static String findStringMax(String mat[][]) {
47     String maxElement = mat[0][0];
48     for (int i = 0; i < mat.length; i++) {
49         for (int j = 0; j < mat[i].length; j++) {
50             if ((maxElement.compareTo(mat[i][j])) < 0) {
51                 maxElement = mat[i][j];
52             }
53         }
54     }
55     return maxElement;
56 }
57
58 public static void main(String[] args) {
59     int integers[][] = {{1,2,4,4},{5,5,4,2},{3,1,1,5}};
60     System.out.println("Largest Element:" + findIntMax(integers));
61     findIntIndex(integers, findIntMax(integers));
62
63     String strings[][] = {{ "Apple", "Banana", "Pork", "Beef", "David", "Kelin", "Peter", "Zag", "Diana", "Elin", "Adam", "Young", "Peter", "Zag" }};
64     System.out.println("Largest Element:" + findStringMax(strings));
65 }
66
67
68

```

Output - test (run) ×

```

run:
Largest Element: 5
Index at: [1, 0]
Largest Element: Zag
BUILD SUCCESSFUL (total time: 0 seconds)

```

Testing to find the index of the String:

```
58 static void findStringMax(String mat[][], String maxElement) {
59     boolean found = false;
60     for (int i = 0; i < mat.length; i++) {
61         for (int j = 0; j < mat[i].length; j++) {
62             if (mat[i][j].equals(maxElement)) {
63                 System.out.println("Index at: [" + i + ", " + j + "]");
64                 found = true;
65                 break;
66             }
67         }
68         if (found) {
69             break;
70         }
71     }
72 }
73
74 public static void main(String[] args) {
75     int integers[][] = {{1,2,4,4},{5,5,4,2},{3,1,1,5}};
76     System.out.println("Largest Element: " + findIntMax(integers)) ;
77     findIntIndex(integers, findIntMax(integers));
78
79     String strings[][] = {{ "Apple", "Banana", "Pork", "Beef"}, {"David", "Kelin", "Peter", "Zag", "Diana"}, {"Elin", "Adam", "Young", "Peter", "Zag"}};
80     System.out.println("Largest Element: " + findStringMax(strings));
81     findStringMax(strings, findStringMax(strings));
82 }
83
```

Output - test (run) X

```
> run:
> Largest Element: 5
> Index at: [1, 0]
> Largest Element: Zag
> Index at: [1, 3]
BUILD SUCCESSFUL (total time: 0 seconds)
```