COSC 237

Assignment # 3

DUE DATE: Thursday, 11/18/2021

For this assignment you have 4 programming problems. Same type of submission as in previous assignments, both email and hard copy. **Turn in ONLY** the methods/client files you modified/created. Have a Word document with just the methods merge/split for #1, #2, #3. Turn in the client programs (.java) for #1, #2, #3 and for #4 (**NOTE:** #4 is using the ArrayList class from Java, NOT a user-created class, so you submit a single program with main plus the required methods).

You will work in teams (as assigned), but all teammates should contribute seriously - please let me know if that's not the case. In principle, you should not discuss the homework with anyone, except your partners and me. If you do discuss it with someone else (better don't), you should say this in the preamble of the assignment. You should only submit work that is completely your own and you should be able to explain your solutions if asked to do so.
Make sure you handle **invalid input, including type mismatch** - input validation is a requirement! Focus on **good design**, **good style**, and, most importantly, **reuse of code**.
Start early and remember the rules about late assignments/syllabus → **no late assignments!**

NOTE: Try to use this quote as a guide when writing code (I do!): *"Any fool can write code that a computer can understand. Good programmers write code that humans can understand."*
--- Martin Fowler, Refactoring: Improving the Design of Existing Code

HONORS STATEMENT: "I will not cheat, fabricate materials, plagiarize, or help another to undertake such acts of academic dishonesty, nor will I protect those who engage in acts of academic dishonesty. I pledge on my honor as a student, that I have not received or provided any unauthorized help on this assignment."

1**. (Unsorted List: array implementation)** This unsorted list ADT discussed in class should be extended by the addition of two new methods:

1. A method named **`merge`** that concatenates 2 unsorted lists into a third. Assume that `list_1` and `list_2` don't have any values in common. The resulting list should be an unsorted list that contains all of the values from `list_1` and `list_2` (preserve the order).
2. A method named **`split`** that divides a list into 2 lists according to a key. If `list_1` and `list_2` are the resulting lists, `list_1` should contain all the items of the original list whose values are less than or equal to the key passed and `list_2` should contain all the items of the original list whose values are greater than the key passed.

Next, create a client to test your program. The client should work with 3 unsorted lists named `list_1`, `list_2` and `result`. Read the data for `list_1` and `list_2` from the files `list1.txt` and `list2.txt` (take input from user the names of the input files, handle the `FileNotFoundException` exception (`try/catch`) and consume unwanted input.) Merge `list_1` and `list_2` into `result` and split `result` according to a key (input from the user). Make sure you handle **all** possible errors.
SAMPLE OUTPUT:

```
Please input the name of the file to be opened for first list: list1.txt
Please input the name of the file to be opened for second list: list2.txt
The first list is:
13  25  34  67  56  10  20  27  2  5  1  45  59
The second list is:
73  29  14  87  72  100  200  127  22  15  19  145  159  78
```

```
The merged list is:
13  25  34  67  56  10  20  27  2  5  1  45  59  73  29  14  87  72  100  200  127  22  1
5  19  145  159  78
Enter key for split: 49
The first list after split is:
13  25  34  10  20  27  2  5  1  45  29  14  22  15  19
The second list after split is:
67  56  59  73  87  72  100  200  127  145  159  78
```

FOR input files:

```
list1.txt: 13 c v b 25 34 x x 67 56 10 a a 20 27 2 a s 5 1 45 59
list2.txt: 73 29 c c c 14 87 72 100 200 c c c 127 22 15 19 c v v v 145 159 78
```

---

2. (Sorted List: array implementation) Same problem for **this** sorted list ADT discussed in class.

NOTE: The `merge` and `split` methods for the sorted list shouldn't look like the same method for the unsorted list.

Program requirement: You should merge the two lists in **one traversal** (do not make repeated calls to `insert` – very inefficient!). You will not get credit for the same method in both classes. Same for method `split`: shouldn't look like the same method for the unsorted list. It can be improved. Do not call any version of `insert` in both methods `merge` and `split`!

SAMPLE OUTPUT:

```
Please input the name of the file to be opened for first list: list1.txt
Please input the name of the file to be opened for second list: list2.txt
The first list is:
2  5  8  11  14  29  33  43  45  51  55  65  70  75
The second list is:
1  4  7  9  10  15  35  49  57  59  67
The merged list is:
1  2  4  5  7  8  9  10  11  14  15  29  33  35  43  45  49  51  55  57  59  65  67  70
75
Enter key for split: 13
The first list after split is:
1  2  4  5  7  8  9  10  11
The second list after split is:
14 15  29  33  35  43  45  49  51  55  57  59  65  67  70  75
```

FOR input files:

```
list1.txt: 2 5 8 m m b 11 14 29 x 33 43 45 51 z z 55 65 70 b 75
list2.txt: 1 ccc 4 bb 7 mm 9 10 15 35 x x x 49 57 59 67
```

---

3. Modify the class **`UnorderedLinkedListInt`** by adding 2 more methods: one to concatenate/merge 2 unsorted linked lists and one to split/divide a list into 2 lists according to a key.

- Method signature: `public void merge1(UnorderedLinkedListInt list1)`. This method will append `list1` to the original list. Side effect: the original list is lost.
- Method signature: `public UnorderedLinkedListInt merge2(UnorderedLinkedListInt list1)`. This method should create a new list with the concatenation result. The original lists should be preserved
- Method signature: `public void split(UnorderedLinkedListInt list1, UnorderedLinkedListInt list2, int key)`. This method should split the original list into `list1` and `list2`. The original list should be preserved.

Use this client to test the second method to merge (`merge2`):

```
import java.util.*;
public class ClientMerge_2 {
    static Scanner input = new Scanner(System.in);
    public static void main(String[] args) {
        UnorderedLinkedListInt list1 = new UnorderedLinkedListInt();
        UnorderedLinkedListInt list2 = new UnorderedLinkedListInt();
        UnorderedLinkedListInt list3 = new UnorderedLinkedListInt();
        int num;
        System.out.println("Enter integers for the first list(999 to stop)");
        num = input.nextInt(); //call your own getInt() method
        while (num != 999) {
            list1.insertLast((Integer) num);
            num = input.nextInt(); //call your own getInt() method
        }
        System.out.println("Enter integers for the second list(999 to stop)");
        num = input.nextInt(); //call your own getInt() method
        while (num != 999) {
            list2.insertLast((Integer) num);
            num = input.nextInt(); //call your own getInt() method
        }
        System.out.print("\nThe first list is: ");
        list1.print();
        System.out.println("\nThe length of the first list is: " + list1.length());
        if (!list1.isEmptyList()) {
            System.out.println("First element/list1: " + list1.front());
            System.out.println("Last element/list1: "  + list1.back());
        }
        System.out.print("\nThe second list is: ");
        list2.print();
        System.out.println("\nThe length of the second list is: " + list2.length());
        if (!list2.isEmptyList()) {
            System.out.println("First element/list2: " + list2.front());
            System.out.println("Last element/list2: "  + list2.back());
        }
        list3 = list1.merge2(list2);
        System.out.print("\nAfter concatenating the 2 lists, the merged list1 is: ");
        list3.print();
        System.out.println("\nThe length of the merged list is: " + list3.length());
        if (!list3.isEmptyList()) {
            System.out.println("First element/merged list: " + list3.front());
            System.out.println("Last element/merged list: "  + list3.back());
        }
        System.out.println("Enter key for split: ");
        num = input.nextInt(); //call your own getInt() method
        list3.split(list1, list2, num);
        System.out.print("\nThe first list after split is: ");
        list1.print();
        System.out.print("\nThe second list after split is: ");
        list2.print();
        System.out.println();
    }
}
```

SAMPLE OUTPUT:

```
Enter integers for the first list(999 to stop)
37 10 88 59 27 20 14 32 89 100 12 999
Enter integers for the second list(999 to stop)
23 56 34 15 78 19 999
```

```
The first list is: 37 10 88 59 27 20 14 32 89 100 12
The length of the first list is: 11
First element/list1: 37
Last element/list1: 12

The second list is: 23 56 34 15 78 19
The length of the second list is: 6
First element/list2: 23
Last element/list2: 19

After concatenating the 2 lists, the merged list1 is: 37 10 88 59 27 20 14 32 89 100 12
23 56 34 15 78 19
The length of the merged list is: 17
First element/merged list: 37
Last element/merged list: 19

Enter key for split: 20
The first list after split is: 10 20 14 12 15 19
The second list after split is: 37 88 59 27 32 89 100 23 56 34 78
```

TASK: Write a slightly modified version of this client to test the first method to merge.

---

4. Same problem using the **Java** `ArrayList` class (of type `<Integer>`). Create the following methods: `merge, split, bubbleSort.` Use these **notes** for a short list of methods in `ArrayList` class.

SAMPLE OUTPUT:

```
Please input the name of the file to be opened for first list: list1.txt
Please input the name of the file to be opened for second list: list2.txt
The first list is: [13, 25, 34, 67, 56, 10, 20, 27, 2, 5, 1, 45, 59]
The second list is: [73, 29, 14, 87, 72, 100, 200, 127, 22, 15, 19, 145, 159, 78]
The merged list is: [13, 25, 34, 67, 56, 10, 20, 27, 2, 5, 1, 45, 59, 73, 29, 14, 87, 72,
100, 200, 127, 22, 15, 19, 145, 159, 78]
The merged list sorted is: [1, 2, 5, 10, 13, 14, 15, 19, 20, 22, 25, 27, 29, 34, 45, 56,
59, 67, 72, 73, 78, 87, 100, 127, 145, 159, 200]
Enter key for split: 33
The first list after split is: [1, 2, 5, 10, 13, 14, 15, 19, 20, 22, 25, 27, 29]
The second list after split is: [34, 45, 56, 59, 67, 72, 73, 78, 87, 100, 127, 145, 159,
200]
```

FOR input files:

```
list1.txt: 13 c v b 25 34 x x 67 56 10 a a 20 27 2 a s 5 1 45 59
list2.txt: 73 29 c c c 14 87 72 100 200 c c c 127 22 15 19 c v v v 145 159 78
```

NOTE: **AGAIN –** Email and print **ONLY** the methods/client files you modified/created.