# ICE 9: Collections

**Resource**: http://blue.smu.edu.sg/is442/ice9-resource.zip

1. **[ Difficulty: * ]** Write the `FrequencyApp` program: This program will keep prompting the user for words until the user enters an empty string. It will then print the frequency of all the words entered so far.

```
D:\ice9\Q1> java FrequencyApp
Enter the word> apple
Enter the word> apple
Enter the word> orange
Enter the word> pear
Enter the word> orange
Enter the word> pear
Enter the word> pear
Enter the word>

Frequency Count:
2 orange
2 apple
3 pear
```

2. **[ Difficulty: * ]** Study the given `IteratorEx` class. Write the code for **removeMatches** – this method goes through the **List** and removes "flavours" that match parameter **toMatch** from the **List**. Check that **IteratorEx** prints out the corrected list of **flavours** (without "weird") after you have modified **removeMatches**.

3. **[ Difficulty: * ]** Write a method called **reverse** that accepts a map from strings to strings (key String, value string) as parameter and returns a new map that is the reverse of the original. The reverse of a map is a new map that uses the values from the original as its keys and the keys from the original as its values. Since a map's values need not be unique but its keys must be, you should have each value (of the original map) map to a List of keys. For example, if the input parameter map consists of the following

| Key (Person's name) | Value (Quiz Score) |
|---|---|
| Alfred | 81 |
| Elise | 61 |
| Billy | 41 |
| Daniel | 41 |
| Charlie | 54 |

The return value of the method should be a map consisting of the following:

| Key(Quiz Score) | Value (Person's name) |
|---|---|
| 41 | Billy, Daniel |
| 61 | Elise |
| 81 | Alfred |
| 54 | Charlie |

(The order of the keys and values does not matter.)

Your test code should print the following:

```
Input:  {Billy=41, Charlie=54, Alfred=81, Elise=61, Daniel=41}
Output: {81=[Alfred], 61=[Elise], 41=[Billy, Daniel], 54=[Charlie]}
```

4. **[ Difficulty: * ]** You have just graduated and are currently employed by a top financial bank in Antarctica. For your first assignment, you joined a team responsible for the implementation of a new online banking portal.

   You have to implement a piece of code to transfer money from one account to another account. A user of the system would provide the account numbers of the two accounts (**accFrom**, **accTo**) and the amount to transfer (**amount**). The money can only be transferred when there is sufficient balance in **accFrom**. In the case that amount is more than the balance in **accFrom**, an unsuccessful status should be returned to the user, and the original balances of the accounts corresponding to **accTo** and **accFrom** would remain unchanged.
   Along with the above description of the requirement, your team leader passes you the following diagrams.

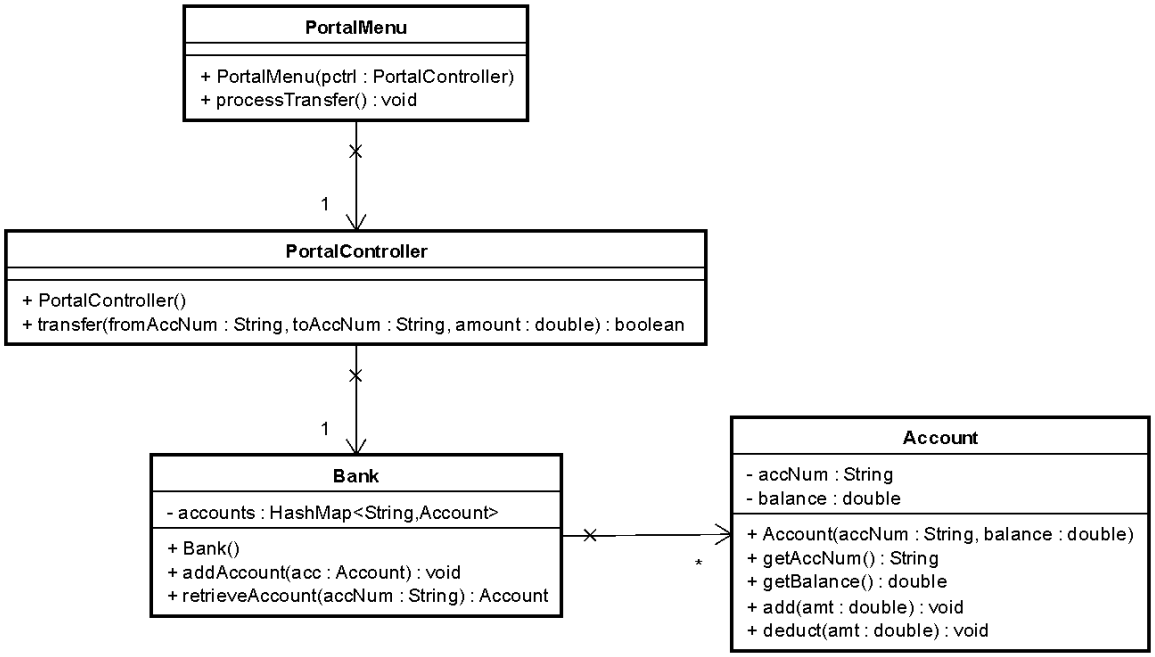



Figure 1: Class Diagram

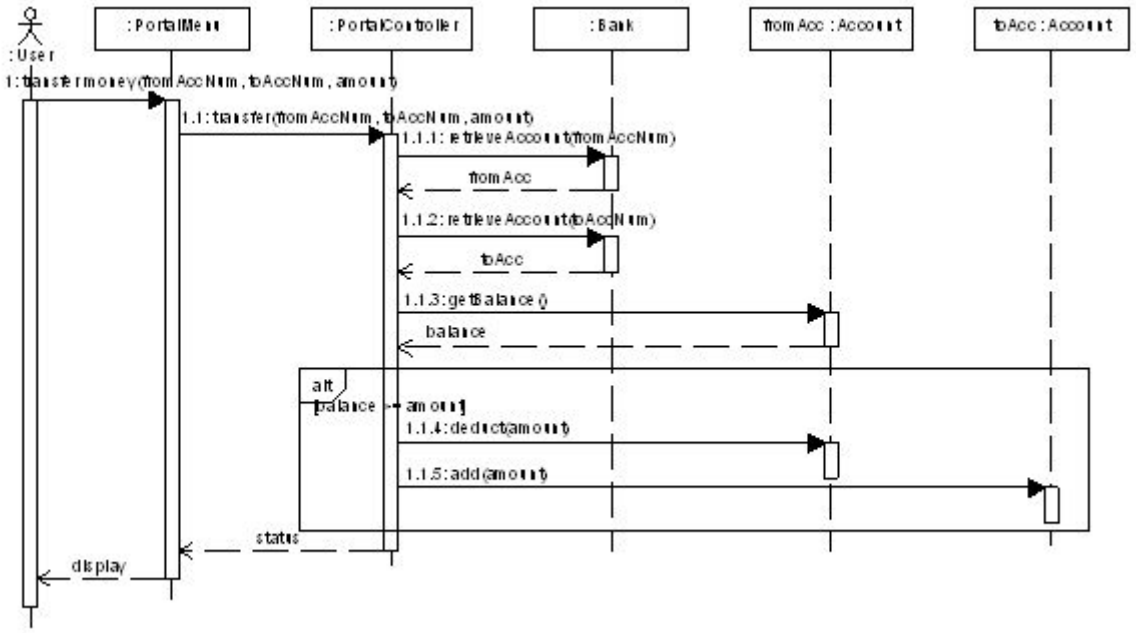



Figure 2: Sequence Diagram

You are given the following java files:

- Account.java
- Bank.java
- PortalApp.java (contains the **main** method)

(a) Implement the **PortalController** class (a partial implementation is given).
(b) Implement the **PortalMenu** class (a partial implementation is given).

A sample execution of **PortalApp** produces the following output:

```
D:\ice9\Q4>java PortalApp
Bank Portal
Enter your account number > acc01
Enter account to transfer to > acc03
Enter amount to transfer > 1000
1000.0 is transferred from acc01 to acc03

D:\ice9\Q4>java PortalApp
Bank Portal
Enter your account number > acc01
Enter account to transfer to > acc02
Enter amount to transfer > 3000
acc01 has insufficient funds for transfer
```

5. **[ Difficulty: *** ]** Write the following `UnionApp` program. This program will prompt the user for two sets of numbers, and prints the unique numbers in both the list. You may wish to consider the use of `java.util.HashSet` class.

```
Hint: The split method of the String class can be used to split values separated by a
delimiter.

String s = "a,b,c";
String[] arr = s.split(","); // arr[0] stores "a", arr[1] stores "b",
                             // arr[2] stores "c"
```

```
D:\ice9> java UnionApp
Enter first line> 1,4,5,7,9
Enter second line> 4,5,9,11,13
1,4,5,7,9,11,13
```

6. **[ Difficulty: ** ]** You are given the data file (result.csv) that contains all the students' result for Eunoia Management University (EMU). Write a program that prints the name of all students who are on the Dean's List. To qualify for a place on the Dean's List, a student must have achieved a minimum GPA of 3.7 or above.

    a. The data file (result.csv) is pretty large. Use a smaller data set to develop your application. The suggestion would be to use 1 – 5 rows of data.
    b. After you get your application working with the small data file you create in step (a), try with the original data file (result.csv). If your program crashes, insert println statements. Discuss with your neighbour, where you would include your println statements and the reason for doing so at the particular location. **Note:** You do not need to fix the error.

7. **[ Difficulty: *** ]** Use Apache Commons CSV ( https://commons.apache.org/proper/commons-csv ) to solve Q6. The following steps are suggested for you:

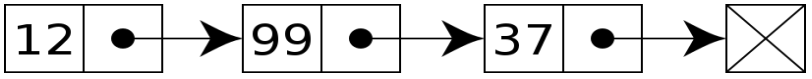
    a. Look for a "Download" link. Most websites will have that for you to download the compiled version of their codes. Always download the "Binaries" version (not "Source").

    b. Look for a "Getting Started" or "User Guide", they will usually give you some sample codes that you can reuse in your codes.

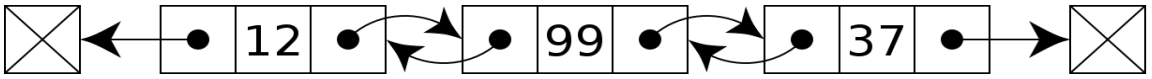
    c. Set up your directory structure to place the source files, libraries and output classes, and write the compile.bat and run.bat after you have completed your codes.

8. **[ Difficulty: *** ]** A linked list is a linear data structure where each element is a separate object. Each node of a list is made up of the data and references to the previous/next node.

   **a. Singly linked list** (https://en.wikipedia.org/wiki/Linked_list#/media/File:Singly-linked-list.svg)




   **b. Double linked list** (https://en.wikipedia.org/wiki/Linked_list#/media/File:Doubly-linked-list.svg)




Below shows the implementation of a `Node` class used in a **Singly linked list:**

```
public class Node {
    private int val;   // this stores the value
    private Node next; // this stores the reference to the next node

    public Node(int v) {
       val = v;
    }

    // getters and setters not shown for brevity
}
```

**Task:** Implement a **double linked list**. Complete the following `LinkedList` class with the following methods:

- `LinkedList()`
  Constructs a `LinkedList` with 0 elements

- `add(int element)`
  Appends the specified element to the end of this list.

- `add(int position int element)`
  Inserts the specified element at the specified position in this list.

- `remove()`
  Removes the first element.

- `remove(int position)`
  Removes the element at the specified position in this list.

- `size()`
  Returns the number of elements in this list.

**Hint**: The idea is for the LinkedList to keep track of the first Node and the last Node and the number of elements it holds. When a new element is added at the end, update the count of elements and the last Node to point to the new element.

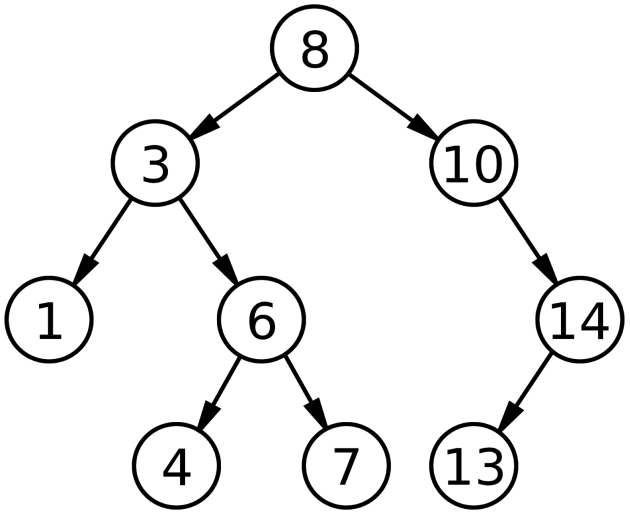
```
public class LinkedList {
    private Node head;
    private Node tail;

    // your code here
}
```

9. **[ Difficulty: *** ]** A Binary Search Tree is a node-based binary tree data structure which has the following properties:

    **a.** The left subtree of a node contains only nodes with keys lesser than the node's key.

    **b.** The right subtree of a node contains only nodes with keys greater than the node's key.

    **c.** The left and right subtree each must also be a binary search tree.




( https://en.wikipedia.org/wiki/Binary_search_tree#/media/File:Binary_search_tree.svg )

**Reference:** https://code.tutsplus.com/tutorials/the-binary-search-tree--cms-20668

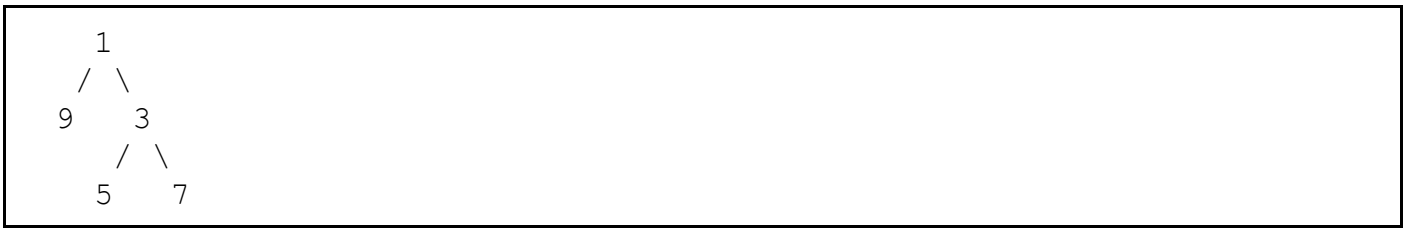
An implementation of `TreeNode` is shown:

```
public class TreeNode {
    private int val;
    private TreeNode left;
    private TreeNode right;
    public TreeNode(int x) { val = x; }

    // getters and setters   (not shown for brevity)
}
```

Implement a method to check if a tree is height-balanced, a binary tree in which the depth of the two subtrees of every node does not differ by more than 1.

```
public class Q9 {
    public static boolean isBalanced(TreeNode root) {

    }
}
```

This is a height-balanced tree.




This is not a height-balanced tree. The two nodes (with value 2) differ in depth by 2.