EECS 233 Programming Assignment #1
100 points
Due September 28, 2018 before 11:59pm

Create a class named Person that contains a string data field named ID for the person's ID and a LinkedList (from java.util.LinkedList<E>) data field named phoneNums for all the numbers that the person possesses (the numbers saved in the LinkedList should be *String*). Provide an abstract data type PBList, which stands for phone book list, that can be used to represent sequences of objects of class Person.
Note: The IDs are unique for each person but a person can have multiple entries in the phonebook. In other words there can be multiple entries in PBList with the same ID which belong to the same person.

The abstract data type PBList must support the following operations:

- *int size()* - Returns the current size of the list (number of entries in the phonebook).
- *void addPerson(int i, Person person)* - Adds a new element before the i-th element of the list (the index of the first entry is 0). If *i* is greater than the number of the elements in the list, adds it to the end.
- *boolean addNumber(String ID, String phoneNum)* - Adds a new phone number to the first occurrence of ID in the list and returns true. Returns false if ID does not exist in the list.
- *boolean delete(int i)* - Deletes the i-th element of the list and returns true. In the case when the list has less than *i* elements returns false.
- *Person search(int i)* - Returns the i-th element of the list. In the case when the sequence has less than *i* elements, it returns null.
- *void merge()* - Merges the phone numbers of all the entries with the same ID.

Provide two implementations of this abstract data structure: a class named *PBArrayList* and another class named *PBLinkedList*. The first one must use an array to store the sequence of persons and the second a singly linked list.
Notes regarding the implementation of these ADT:

- Using built-in Java classes, such as ArrayList and LinkedList, is not allowed (LinkedList is allowed only for the implementation and usage of class Person).
- Do not forget the issue of managing the "real estate" (i.e., the fixed size of the array) in the case of the *PBArrayList* implementation (e.g. you will need to internally re-allocate the array as it grows too much).

Write a demo application utilizing this abstract data structure. Your application should include the following components:

- Two static methods: *int prefixCountArrayList(PBArrayList list, String prefix)* and *int prefixCountLinkedList(PBLinkedList list, String prefix)* which return the total number of entries in the phonebook with the phone number matching the prefix.
  Note: The running time of this method should be linear in the size of the phone book.

- The main method where you will:
    - Create an instance of *PBArrayList*, insert several elements into it, test the PBArrayList methods and then use the *prefixCountArrayList* method to print the number of elements in the sequence with a given phone prefix.
    - Repeat the above with a *PBLinkedList*.

Submission and Grading: The submissions will be evaluated on completeness, correctness, and clarity. Please provide sufficient comments in your source code to help the TAs read it. Please generate a single zip file containing all your *.java files needed for this assignment and a README file with instructions on how to run your demo. Name your file P1_YourCaseID_YourLastName.zip. Submit your zip file electronically to canvas. Grading:

- Abstract data type specification: 10 points
- *PBArrayList* implementation: 25 points
- *PBLinkedList* implementation: 25 points
- Demo application: 20 points
- Proper encapsulation/information hiding: 10 points
- Design and style: 10 points
    - Style: 3 points. Are variable names descriptive and convey their purpose? Of course simple concepts like a loop variable do not require descriptive names; e.g., it is perfectly fine, even preferable, to use i for a loop variable. Is formatting clear and consistent?
    - Comments: 4 points. Comments should aid the reader to understand the code. Comments that restate what is already clear from the code are redundant and not helpful. Nor are comments that are not consistent with the code. For each method implementation, state in the comments its worst-case running time.
    - Design: 3 points. Are there lines that never execute? Inelegant constructions? Convoluted or unnecessarily inefficient ways to achieve some result?