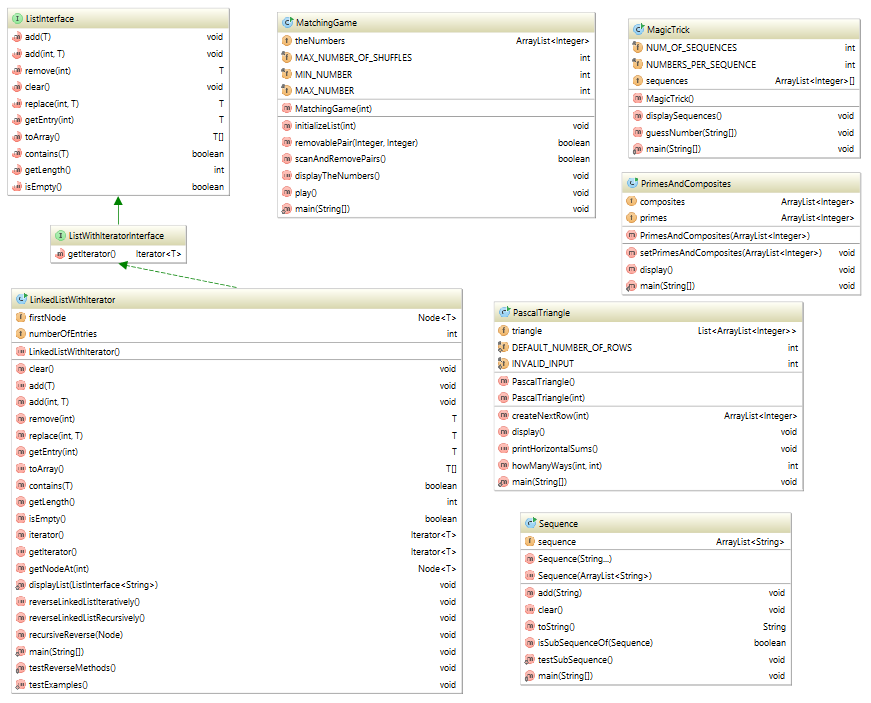
# Comp151 Lab09

In Lab09 you will be working on six separate applications:

1. LinkedListWithIterator
2. PascalTriangle
3. Sequence
4. PrimesAndComposites
5. MagicTrick
6. MatchingGame

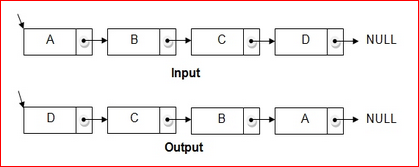
Each application has a main inside the corresponding .java file. The UML diagram below shows the six applications together, however there are no dependencies between any of them.

**UML Diagram:**



## LinkedListWithIterator

Your task is to **reverse** a singly linked list. You are to write two reverse methods: one iterative (reverseLinkedListIteratively()) and the other one recursive (reverseLinkedListRecursively()) solving the same problem. Given the first node (head of the list) need to reverse the list **by changing the next pointers** so the original last node of the list (tail of the list) is now the first as shown in the picture below:



This is a classical interview question which surprisingly is unexpected by many candidates. But most of all this is a basis problem, and understanding both solutions will help you to solve number of linked list problems.

**Sample Run:**

\*\*\* Create a list \*\*\*

List should contain: 15 25 35 45 55 65 75 85 95

Using ADT list operations, the list contains:

The list contains 9 entries, as follows:

15 is entry 1

25 is entry 2

35 is entry 3

45 is entry 4

55 is entry 5

65 is entry 6

75 is entry 7

85 is entry 8

95 is entry 9

Using Iterator methods, the list contains

15 25 35 45 55 65 75 85 95

\*\*\* Calling reverseLinkedListIteratively \*\*\*

List should contain: 95 85 75 65 55 45 35 25 15

Using ADT list operations, the list contains

The list contains 9 entries, as follows:

95 is entry 1

85 is entry 2

75 is entry 3

65 is entry 4

55 is entry 5

45 is entry 6

35 is entry 7

25 is entry 8

15 is entry 9

Using Iterator methods, the list contains

95 85 75 65 55 45 35 25 15

===========================================================

\*\*\* Calling reverseLinkedListRecursively \*\*\*

List should contain: 15 25 35 45 55 65 75 85 95

Using ADT list operations, the list contains

The list contains 9 entries, as follows:

15 is entry 1

25 is entry 2

35 is entry 3

45 is entry 4

55 is entry 5

65 is entry 6

75 is entry 7

85 is entry 8

95 is entry 9

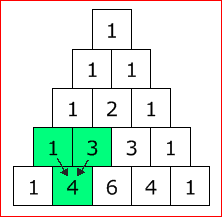
Using Iterator methods, the list contains

15 25 35 45 55 65 75 85 95

\*\*\* Done \*\*\*

## PascalTriangle

As early as the tenth century, mathematicians studied the following triangular pattern of integers, now known as *Pascal’s Triangle* . Despite the early advent of the pattern, it was named after the 17th-century mathematician Blaise Pascal.



Staggering the entries, as we have above is traditional. Each row begins and ends with 1. Each interior entry is the sum of the two entries above it. For example in the last row given here, 4 is the sum of 1 and 3; 6 is the sum of 3 and 3; and 4 is the sum of 3 and 1.

If we number both the rows and the entries in each row beginning with 0, the entry in position **k** of row **n** is often denoted as **C(n,k).** For example, the 6 in the last row is C(4,2). Given n items, C(n,k) turns out to be the number of ways that you can select k of the n items. Thus, C(4,2), which is 6, is the number of ways you can select two of four given items. So, if A, B, C, and D were the four items, here are the six possible choices: AB, AC, AD, BC, BD, CD. Note that the order of the items in each pair is irrelevant, so the choice AB is the same as the choice BA for example.

Go to the following website: [Pascal’s Triangle](http://www.mathsisfun.com/pascals-triangle.html) and check what we can do with the Pascal’s Triangle.

Implement PascalTriangle class. Each row in a triangle is represented as a list and the entire triangle is a list of these lists. Use the class ArrayList for these lists. See the UML diagram and the skeleton of the class.

### Your Task:

1. The skeleton of the program is provided for you. You need to give implementation for all the methods defined (see javaDoc). Test them with the provided main.
2. A sample run of the program and a UML diagram are provided.

### Sample Runs:

Run #1:

=======

Please enter the triangle size that is greater than 1

**5**

The Pascal Triangle of size 5 is:

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

Horizontal sums are:

1

2

4

8

16

\*\*\*COMBINATION problem\*\*\*

How many balls in total?

**4**

How many balls to choose?

**2**

You have 4 balls. How many different ways could you choose just 2 of them?

The answer is: 6

Run #2:

=======

Please enter the triangle size that is greater than 1

**7**

The Pascal Triangle of size 7 is:

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

1 5 10 10 5 1

1 6 15 20 15 6 1

Horizontal sums are:

1

2

4

8

16

32

64

\*\*\*COMBINATION problem\*\*\*

How many balls in total?

**7**

How many balls to choose?

**3**

You have 7 balls. How many different ways could you choose just 3 of them?

The answer cannot be given with the triangle of size 7

## Sequence

Develop an application that, given two sequences of words, will determine if the first is a subsequence of the second. The subsequence is defined as follow:

Given X = <x1,x2,…,xn> a sequence of length n greater or equal to zero and Y = <y1,y2,…,ym> a sequence of length m greater than or equal to zero, X will be consider a subsequence of Y if and only if there exists a strictly increasing sequence of elements K = <k1, k2,…,kn> such that every element xi is equal to yj where j = ki.

Consider the following examples:

* X = < a b a> and Y = <~~a~~ ~~b~~ ~~a~~> 🡺 X **is** a subsequence of Y
* X = < a b c> and Y = <~~a~~ c ~~b~~ > 🡺 X **is** **not** a subsequence of Y
* X = < a b a> and Y = <b c ~~a~~ c ~~b~~ ~~a~~> 🡺 X **is** a subsequence of Y
* X = < a b a> and Y = <b c ~~a~~ c a ~~b~~> 🡺 X **is** **not** a subsequence of Y
* X = < a b a a> and Y = <b c ~~a~~ c ~~b~~ ~~a~~> 🡺 X **is** **not** a subsequence of Y
* X = < a b a> and Y = <b c ~~a~~ c ~~b~~> 🡺 X **is** **not** a subsequence of Y
* X = < a b c> and Y = <b c ~~a~~ c ~~b~~ a> 🡺 X **is** **not** a subsequence of Y
* X = < > and Y = <b c a c b a> 🡺 X **is** a subsequence of Y

### Your Task:

1. The skeleton of the program is provided for you. You need to give implementation for method isSubsequence using appropriate Iterator objects. After the check is done the sequences must remain intact.
2. A sample run of the program and a UML diagram are provided.
3. Make sure that the output is correct (see Sample Run below).

### Sample Run:

The seq1 is [a, b, c]

==> Please enter a sequence of words on a single line, (words should be separated by spaces), or stop.

b c a c b a

Is [a, b, c] a subsequence of [b, c, a, c, b, a]: false

==> Please enter a sequence of words on a single line, (words should be separated by spaces), or stop.

a c b

Is [a, b, c] a subsequence of [a, c, b]: false

==> Please enter a sequence of words on a single line, (words should be separated by spaces), or stop.

a b b c

Is [a, b, c] a subsequence of [a, b, b, c]: true

==> Please enter a sequence of words on a single line, (words should be separated by spaces), or stop.

stop

\*\*\* RUNNING AUTOMATED TESTCASES \*\*\*

Testing if [a, b, c] is a subsequence of []:

Passes

Testing if [] is a subsequence of [a, b, c]:

Passes

Testing if [a, b, c] is a subsequence of [a, c, b]:

Passes

Testing if [a, c, b] is a subsequence of [a, b, c]:

Passes

Testing if [a, b, c] is a subsequence of [a, c, b, c]:

Passes

Testing if [a, c, b, c] is a subsequence of [a, b, c]:

Passes

Testing if [a, b, c, a, b, c] is a subsequence of [a, c, b, c]:

Passes

Testing if [a, c, b, c] is a subsequence of [a, b, c, a, b, c]:

Passes

Testing if [a, b, a, c] is a subsequence of [a, b, a, c]:

Passes

Testing if [a, b, c] is a subsequence of [a, b, d]:

Passes

Testing if [a, b, d] is a subsequence of [a, b, c]:

Passes

Testing if [a, b, c] is a subsequence of [x, y, a]:

Passes

Testing if [x, y, a] is a subsequence of [a, b, c]:

Passes

Testing if [a, b, c] is a subsequence of [a, b, a, a, b, c]:

Passes

Testing if [a, b, a, a, b, c] is a subsequence of [a, b, c]:

Passes

1. **Primes**

Write a program that will compute the list of primes and composites from the given set of numbers, using [Sive of Eratosthenes algorithm](http://en.wikipedia.org/wiki/Sieve_of_Eratosthenes) .

**Your Task:**

1. The skeleton of the program is provided for you. You need to give implementation for setPrimesAndComposites method using appropriate Iterator objects.
2. A sample run of the program and a UML diagram are provided.
3. Make sure that the output is correct (see Sample Runs below).

**Sample Run:**

RUN #1

======

Enter the maximum value to test for primes

It should be an integer value greater than or equal to 2.

17

====> Constructing list of candidates up to 17

The candidates list is [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17]

==> Found the prime 2

====> Found the composite 4

====> Found the composite 6

====> Found the composite 8

====> Found the composite 10

====> Found the composite 12

====> Found the composite 14

====> Found the composite 16

==> Found the prime 3

====> Found the composite 9

====> Found the composite 15

==> Found the prime 5

==> Found the prime 7

==> Found the prime 11

==> Found the prime 13

==> Found the prime 17

The primes list is

2 3 5 7 11 13 17

The composites list is

4 6 8 10 12 14 16 9 15

## MagicTrick

The following magic trick is almost magic. With some numerical sequences you guess the number a person thinks of:

1. Ask a person to think of a number from 1 to 31, e.g. 19
2. Now show five numerical sequences to the person (see the sequences in sample runs below)
3. The person shall tell you, in which sequence the number appears. 19 in our example appears in sequence 1, 2, and 5
4. Add the first numbers of the sequences the person named – the sum is the number the person is thinking of:
   1. The numerical sequences that the person named begin with numbers 1, 2, and 16
   2. 1 + 2 + 16 is 19

Write a program that implements this magic trick. The constructor should generate the sequences and store them in an array of ArrayList<Integer>.

**Sample Run:**

Run #1:

=======

Think of a number between 1 and 31

Sequence 1 : [1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31]

Sequence 2 : [2, 3, 6, 7, 10, 11, 14, 15, 18, 19, 22, 23, 26, 27, 30, 31]

Sequence 3 : [4, 5, 6, 7, 12, 13, 14, 15, 20, 21, 22, 23, 28, 29, 30, 31]

Sequence 4 : [8, 9, 10, 11, 12, 13, 14, 15, 24, 25, 26, 27, 28, 29, 30, 31]

Sequence 5 : [16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31]

List all the sequences that your number is in (ie. 1 3)

1 2 3

Your number is 7 :)

Run #2:

=======

Think of a number between 1 and 31

Sequence 1 : [1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31]

Sequence 2 : [2, 3, 6, 7, 10, 11, 14, 15, 18, 19, 22, 23, 26, 27, 30, 31]

Sequence 3 : [4, 5, 6, 7, 12, 13, 14, 15, 20, 21, 22, 23, 28, 29, 30, 31]

Sequence 4 : [8, 9, 10, 11, 12, 13, 14, 15, 24, 25, 26, 27, 28, 29, 30, 31]

Sequence 5 : [16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31]

List all the sequences that your number is in (ie. 1 3)

3

Your number is 4 :)

## Matching Game

Consider a matching game in which you have a list of random integer values between 10 and 99. You remove from the list any pair of consecutive integers that match.  
If first integer has digits x1y1 and the second integer has digits x2y2 the match is found if any of the following is true:  
 \*    x1 is the same as x2  
 \*    x1 is the same as y2  
 \*    y1 is the same as x2  
 \*    y1 is the same as y2  
If all integer values are removed, you win.  
  
You are allowed to shuffle the integer values up to 5 times to increase the probability of finding more matches.  
  
For example consider the following sequence of integers:  
70 82 43 23 89 12 43 84 93 17  
The pair 70 and 82 does not match in either digit and so cannot be removed, next check 82 and 43, no match either. Next check 43 and 23, there is a match, so both values are removed. Continue checking for pairs from 89, which is the value after the removed pair. Once you finish the first pass the following sequence remains:  
70 82 89 12 93 17  
Now return to the beginning of the list and check the pairs again. After the second pass the following sequence remains:  
70 12 93 17  
Now return to the beginning of the list and check for the pairs again. This time no matches were found, so shuffle the list and try again. You are allowed to shuffle maximum 5 times.

### Your Task:

1. Write a program that simulates this game (the skeleton is provided for you):
   1. initializeList – generates numbers - two-digit integers between 10 and 99 inclusively. The generated integers must be stored in ArrayList theNumbers using an instance of **ListIterator**.
   2. Then using **another** instance of **ListIterator**, scan the list and remove matching pairs of values.
   3. After each pass use an instance of **Iterator** to display the remaining content of the list.
2. A sample run of the program and a UML diagram are provided.
3. Make sure that the output is correct (see Sample Runs below).

### Sample Run:

RUN #1

======

How many numbers (no less than 10)?

10

Starting with:

36 22 32 24 38 39 84 98 29 61

Removed: 22 32

Removed: 38 39

Removed: 84 98

The list after pass #1

36 24 29 61

Removed: 24 29

The list after pass #2

36 61

Removed: 36 61

The list after pass #3

The list is empty.

No more pairs to remove.

\*\*\* Winner! \*\*\*

RUN #2

======

How many numbers (no less than 10)?

10

Starting with:

28 22 51 87 22 54 24 88 71 12

Removed: 28 22

Removed: 54 24

Removed: 71 12

The list after pass #1

51 87 22 88

No more pairs to remove.

Shuffling the numbers.

The list after shuffling #1

51 22 87 88

Removed: 87 88

The list after pass #2

51 22

No more pairs to remove.

\*\*\* Better luck next time! \*\*\*

RUN #3

======

How many numbers (no less than 10)?

20

Starting with:

61 90 84 57 65 39 54 70 31 55 93 70 39 98 75 87 71 22 79 15

Removed: 57 65

Removed: 39 98

Removed: 75 87

The list after pass #1

61 90 84 39 54 70 31 55 93 70 71 22 79 15

Removed: 70 71

The list after pass #2

61 90 84 39 54 70 31 55 93 22 79 15

No more pairs to remove.

Shuffling the numbers.

The list after shuffling #1

54 61 70 22 31 90 55 84 93 15 79 39

Removed: 79 39

The list after pass #3

54 61 70 22 31 90 55 84 93 15

No more pairs to remove.

Shuffling the numbers.

The list after shuffling #2

90 61 84 15 55 31 70 54 22 93

Removed: 15 55

The list after pass #4

90 61 84 31 70 54 22 93

No more pairs to remove.

Shuffling the numbers.

The list after shuffling #3

84 54 31 93 70 90 61 22

Removed: 84 54

Removed: 31 93

Removed: 70 90

The list after pass #5

61 22

No more pairs to remove.

\*\*\* Better luck next time! \*\*\*

RUN #4

======

How many numbers (no less than 10)?

50

Starting with:

71 81 96 86 83 23 67 18 56 66 46 48 17 57 30 11 24 52 72 14 29 54 12 45 79 10 36 76 90 49 86 74 25 51 88 54 19 83 25 39 50 78 33 68 25 88 28 45 66 44

Removed: 71 81

Removed: 96 86

Removed: 83 23

Removed: 56 66

Removed: 46 48

Removed: 17 57

Removed: 24 52

Removed: 36 76

Removed: 90 49

Removed: 25 51

Removed: 88 28

The list after pass #1

67 18 30 11 72 14 29 54 12 45 79 10 86 74 88 54 19 83 25 39 50 78 33 68 25 45 66 44

Removed: 25 45

The list after pass #2

67 18 30 11 72 14 29 54 12 45 79 10 86 74 88 54 19 83 25 39 50 78 33 68 66 44

Removed: 68 66

The list after pass #3

67 18 30 11 72 14 29 54 12 45 79 10 86 74 88 54 19 83 25 39 50 78 33 44

No more pairs to remove.

Shuffling the numbers.

The list after shuffling #1

10 83 14 54 54 72 11 86 45 44 33 78 79 50 67 12 30 74 19 88 25 18 39 29

Removed: 14 54

Removed: 45 44

Removed: 78 79

Removed: 39 29

The list after pass #4

10 83 54 72 11 86 33 50 67 12 30 74 19 88 25 18

No more pairs to remove.

Shuffling the numbers.

The list after shuffling #2

83 50 74 19 33 88 54 86 11 12 18 72 30 67 10 25

Removed: 11 12

The list after pass #5

83 50 74 19 33 88 54 86 18 72 30 67 10 25

Removed: 86 18

The list after pass #6

83 50 74 19 33 88 54 72 30 67 10 25

No more pairs to remove.

Shuffling the numbers.

The list after shuffling #3

25 19 74 67 33 72 54 10 88 83 50 30

Removed: 74 67

Removed: 88 83

Removed: 50 30

The list after pass #7

25 19 33 72 54 10

No more pairs to remove.

Shuffling the numbers.

The list after shuffling #4

19 25 33 54 72 10

No more pairs to remove.

Shuffling the numbers.

The list after shuffling #5

19 33 54 25 72 10

Removed: 54 25

The list after pass #8

19 33 72 10

No more pairs to remove.

\*\*\* Better luck next time! \*\*\*