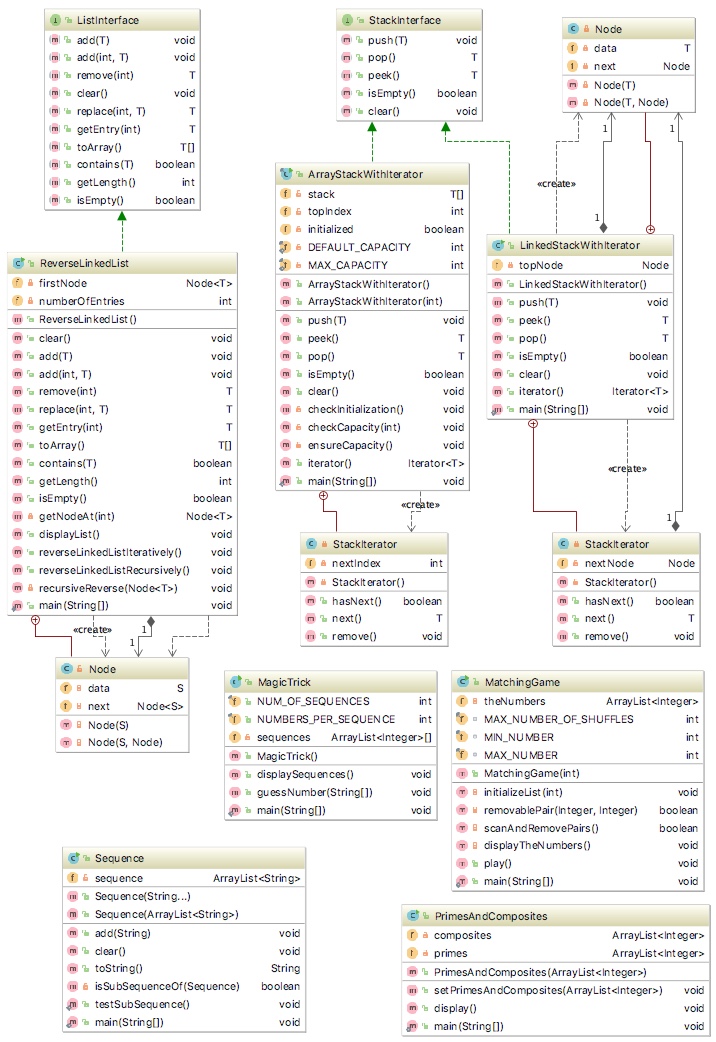
# Comp151 Lab09

In Lab09 you will be working on seven separate applications:

1. ArrayStackWithIterator
2. LinkedStackWithIterator
3. PrimesAndComposites
4. Sequence
5. MagicTrick
6. MatchingGame
7. ReverseLinkedList

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

**UML Diagram:**

****

1. **ArrayStackWithIterator**

Implement an iterator for the array-based implementation of stack. The skeleton of the program is provided for you. You need to give implementation for next()and remove() methods. Please note that this iterator does not support the remove operation so UnsupportedOperationException must be thrown.

**Sample Run:**

Creating a stack

Creating a stackIterator

Calling stackIterator.next())

Exception caught "Illegal call to next(); iterator is below the bottom of the stack."

Adding to stack Jim, Jess, Jill, Jane, Joe

Accessing the top entry with iterator

hasNext() returned false

Resetting the iterator

The stack contains (from top to bottom):

Joe

Jane

Jill

Jess

Jim

Adding to stack John

peek() returns: John

hasNext() returned false

Resetting the iterator

The stack contains (from top to bottom):

John

Joe

Jane

Jill

Jess

Jim

Calling stackIterator.remove())

Exception caught "remove operation is not supported by this stack iterator."

Done.

Process finished with exit code 0

1. **LinkedStackWithIterator**

Implement an iterator for the linked-based implementation of a stack. The skeleton of the program is provided for you. You need to give implementation for next() and remove() methods.

Your program should produce the same results as for project I.

1. **PrimesAndComposites**

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

## 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 a

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

Not a subsequence - Passes

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

A subsequence - Passes

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

Not a subsequence - Passes

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

Not a subsequence - Passes

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

A subsequence - Passes

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

Not a subsequence - Passes

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

Not a subsequence - Passes

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

A subsequence - Passes

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

A subsequence - Passes

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

A subsequence - Passes

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

Not a subsequence - Passes

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

Not a subsequence - Passes

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

Not a subsequence - Passes

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

Not a subsequence - Passes

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

A subsequence - Passes

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

Not a subsequence - Passes

## 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 the following five numerical sequences to the person:

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]

1. The person shall tell you, in which sequence the number appears. 19 in our example appears in sequence 1, 2, and 5
2. 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

Please observe the binary representations of each number and the relationship of bits in each column to the elements in each sequence. For example, all numbers in **sequence 1** (highlighted in yellow) have the **bit** set in the **1st column** from the right, all numbers in **sequence 4** have the **bit** set in the **4th column** from the right (highlighted in green), and so on:

1 -> 00001

2 -> 00010

3 -> 00011

4 -> 00100

5 -> 00101

6 -> 00110

7 -> 00111

8 -> 01000

9 -> 01001

10 -> 01010

11 -> 01011

12 -> 01100

13 -> 01101

14 -> 01110

15 -> 01111

16 -> 10000

17 -> 10001

18 -> 10010

19 -> 10011

20 -> 10100

21 -> 10101

22 -> 10110

23 -> 10111

24 -> 11000

25 -> 11001

26 -> 11010

27 -> 11011

28 -> 11100

29 -> 11101

30 -> 11110

31 -> 11111

Looking at the sample runs below note the properties of the sequences: elements highlighted in red are powers of 2, underlining depicts the number of elements per group.

Write a program that implements this magic trick. The constructor should generate the sequences and store them in an array of ArrayList<Integer>. Use binary bit manipulation techniques like **mask**, **&**, and **<<** that we utilized in the implementation of binary radix sort.

**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 Runs with seed 11:

RUN #1

======

How many numbers (no less than 10)?

12

Starting with:

28 48 31 85 43 67 80 93 43 27 84 64

Removed: 28 48

Removed: 93 43

Removed: 84 64

The list after pass #1

31 85 43 67 80 27

No more pairs to remove.

Shuffling the numbers.

The list after shuffling #1

27 31 85 43 80 67

No more pairs to remove.

Shuffling the numbers.

The list after shuffling #2

27 80 85 67 43 31

Removed: 80 85

Removed: 43 31

The list after pass #2

27 67

Removed: 27 67

The list after pass #3

The list is empty.

No more pairs to remove.

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

RUN #2

======

How many numbers (no less than 10)?

16

Starting with:

28 48 31 85 43 67 80 93 43 27 84 64 67 69 32 98

Removed: 28 48

Removed: 93 43

Removed: 84 64

Removed: 67 69

The list after pass #1

31 85 43 67 80 27 32 98

Removed: 27 32

The list after pass #2

31 85 43 67 80 98

Removed: 80 98

The list after pass #3

31 85 43 67

No more pairs to remove.

Shuffling the numbers.

The list after shuffling #1

31 43 67 85

Removed: 31 43

The list after pass #4

67 85

No more pairs to remove.

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

RUN #3

======

How many numbers (no less than 10)?

50

Starting with:

28 48 31 85 43 67 80 93 43 27 84 64 67 69 32 98 54 41 78 46 15 95 81 68 57 18 41 24 94 99 98 11 96 65 99 78 74 49 42 52 82 73 44 97 46 60 54 44 39 58

Removed: 28 48

Removed: 93 43

Removed: 84 64

Removed: 67 69

Removed: 54 41

Removed: 15 95

Removed: 81 68

Removed: 18 41

Removed: 24 94

Removed: 99 98

Removed: 96 65

Removed: 78 74

Removed: 49 42

Removed: 52 82

Removed: 46 60

Removed: 54 44

The list after pass #1

31 85 43 67 80 27 32 98 78 46 57 11 99 73 44 97 39 58

Removed: 27 32

Removed: 98 78

Removed: 97 39

The list after pass #2

31 85 43 67 80 46 57 11 99 73 44 58

No more pairs to remove.

Shuffling the numbers.

The list after shuffling #1

43 58 57 99 85 11 80 67 46 44 31 73

Removed: 58 57

Removed: 67 46

Removed: 31 73

The list after pass #3

43 99 85 11 80 44

No more pairs to remove.

Shuffling the numbers.

The list after shuffling #2

99 85 80 44 43 11

Removed: 85 80

Removed: 44 43

The list after pass #4

99 11

No more pairs to remove.

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

RUN #4

======

How many numbers (no less than 10)?

124

Starting with:

28 48 31 85 43 67 80 93 43 27 84 64 67 69 32 98 54 41 78 46 15 95 81 68 57 18 41 24 94 99 98 11 96 65 99 78 74 49 42 52 82 73 44 97 46 60 54 44 39 58 25 26 30 24 31 19 79 76 59 65 73 37 35 52 73 64 86 11 47 37 73 22 69 52 37 78 18 80 44 91 74 64 51 51 97 68 49 32 55 46 42 45 63 56 43 11 16 87 33 56 70 59 18 54 46 72 80 20 97 35 20 77 47 81 99 23 17 28 66 24 76 31 51 21

Removed: 28 48

Removed: 93 43

Removed: 84 64

Removed: 67 69

Removed: 54 41

Removed: 15 95

Removed: 81 68

Removed: 18 41

Removed: 24 94

Removed: 99 98

Removed: 96 65

Removed: 78 74

Removed: 49 42

Removed: 52 82

Removed: 46 60

Removed: 54 44

Removed: 58 25

Removed: 31 19

Removed: 79 76

Removed: 59 65

Removed: 73 37

Removed: 35 52

Removed: 64 86

Removed: 47 37

Removed: 37 78

Removed: 18 80

Removed: 74 64

Removed: 51 51

Removed: 46 42

Removed: 63 56

Removed: 11 16

Removed: 54 46

Removed: 80 20

Removed: 77 47

Removed: 31 51

The list after pass #1

31 85 43 67 80 27 32 98 78 46 57 11 99 73 44 97 39 26 30 24 73 11 73 22 69 52 44 91 97 68 49 32 55 45 43 87 33 56 70 59 18 72 97 35 20 81 99 23 17 28 66 24 76 21

Removed: 27 32

Removed: 98 78

Removed: 97 39

Removed: 91 97

Removed: 55 45

Removed: 72 97

The list after pass #2

31 85 43 67 80 46 57 11 99 73 44 26 30 24 73 11 73 22 69 52 44 68 49 32 43 87 33 56 70 59 18 35 20 81 99 23 17 28 66 24 76 21

Removed: 32 43

The list after pass #3

31 85 43 67 80 46 57 11 99 73 44 26 30 24 73 11 73 22 69 52 44 68 49 87 33 56 70 59 18 35 20 81 99 23 17 28 66 24 76 21

No more pairs to remove.

Shuffling the numbers.

The list after shuffling #1

18 99 24 17 68 22 24 85 73 30 35 56 69 43 33 52 99 26 11 20 44 49 80 11 23 57 59 73 70 46 66 87 67 81 73 21 28 31 76 44

Removed: 22 24

Removed: 73 30

Removed: 35 56

Removed: 43 33

Removed: 44 49

Removed: 57 59

Removed: 73 70

Removed: 46 66

Removed: 87 67

Removed: 21 28

The list after pass #4

18 99 24 17 68 85 69 52 99 26 11 20 80 11 23 81 73 31 76 44

Removed: 68 85

Removed: 20 80

Removed: 73 31

The list after pass #5

18 99 24 17 69 52 99 26 11 11 23 81 76 44

Removed: 11 11

The list after pass #6

18 99 24 17 69 52 99 26 23 81 76 44

Removed: 26 23

The list after pass #7

18 99 24 17 69 52 99 81 76 44

No more pairs to remove.

Shuffling the numbers.

The list after shuffling #2

99 18 52 24 17 76 44 99 81 69

Removed: 52 24

Removed: 17 76

The list after pass #8

99 18 44 99 81 69

No more pairs to remove.

Shuffling the numbers.

The list after shuffling #3

18 81 44 69 99 99

Removed: 18 81

Removed: 69 99

The list after pass #9

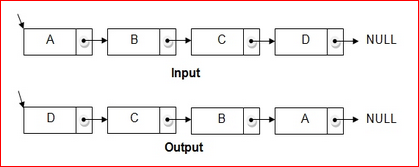
44 99

No more pairs to remove.

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

## ReverseLinkedList

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 basic problem, and understanding both solutions will help you to solve number of linked list problems. As requested in the pre-lab, watch educational videos on YouTube, I used the following search: *"How To Reverse Linked List YouTube".* **In your code please put the link to the video you followed in your solution.**

**Sample Run:**

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

The list contains: 15 25 35 45 55 65 75 85 95

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

Expected result: 95 85 75 65 55 45 35 25 15

The list contains: 95 85 75 65 55 45 35 25 15

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

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

Expected result: 15 25 35 45 55 65 75 85 95

The list contains: 15 25 35 45 55 65 75 85 95

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

Process finished with exit code 0