# CMSC 132

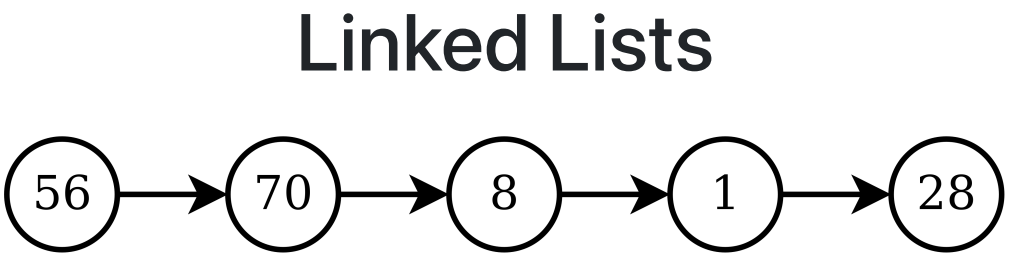
Sections 020X/030X

DEPARTMENT OF COMPUTER SCIENCE

**Assignment:** **Project #4**

**Due Date:** **Wednesday 03/16, 11:00PM**

**Open/Closed policy:** **CLOSED**




## Objectives

This project will give you practice implementing a collection with a parameterized type, and practice implementing linked list methods. A secondary goal of this project is for you to practice writing lots of JUnit tests! Note that very few public tests are being provided, and the release tests have been given names that are intentionally vague. The TAs will not be able to tell you what the release tests are doing, nor will they be able to tell you whether or not you are passing the secret tests. You will need to write lots of tests of your own.

Note that this project is CLOSED. Please review the Course Policy on Open/Closed Projects for information about the rules for closed projects.

**This project is more difficult than it may appear at first glance, so start immediately!**

---

## Code Distribution

Download this zip file and import it into Eclipse, as usual.

The project's code distribution provides you with the following:

- **listClass package** → Contains the class you must implement.
- **tests package** → There are two public tests and a location for your JUnit tests.

---

## Overview

- The linked list in our project will have a head reference and also a tail reference. The "next" link of the last node will always point to null. Be sure to update the head and tail references carefully as the list is modified.
- You should maintain the size of the list with an instance variable so that the size of the list can be ascertained immediately without traversing the list. Be sure to update this variable as the list is modified.
- If you see in the submit server the error "missing_component", the problem is usually that you are missing a required class, or that one of your method prototypes does not match the specification. You can also get this error if you are not defining a generic class correctly. It is recommended that you try submitting your project frequently (if you have not done so yet) so you can identify this problem, if it exists.

---

## Detailed Specification

Your BasicLinkedList class should contain the following features:

- A private static nested class, Node<T>, with the following features:
  - Two private instance variables ("data", an element in the list, and "next", a reference to the successor node.)
  - A constructor that takes one parameter representing an element of the list to be wrapped in this Node. Set the next reference to null.
- Three private instance variables: head and tail references and a variable to keep track of the current size of the list.
- A constructor that initializes this list as empty.
- public int getSize()
  Getter for the size variable. (Do NOT traverse the list to determine its size.)

- public BasicLinkedList<T> addToEnd(T data)
  Adds the element to the tail of the list. Returns a reference to the current object.

- public BasicLinkedList<T> addToFront(T data)

Adds the element to the head of the list. Returns a reference to the current object.

- public T getFirst()
  Returns the head element (without removing it), or null if the list is empty.
- public T getLast()
  Returns the tail element (without removing it), or null if the list is empty.
- public T retrieveFirstElement()
  Removes and returns the head element. If the list is empty, returns null.
- public T retrieveLastElement()
  Removes and returns the tail element. If the list is empty, returns null. **You may not use an Iterator or a for-each loop while implementing this method.**
- public BasicLinkedList<T> removeAllInstances(T targetData)
  Removes all instances of the target element from the list. The return value is just a reference to the current object. **You may not use an Iterator or a for-each loop while implementing this method.**
- public Iterator<T> iterator()
  Returns an instance of an anonymous inner class that defines an Iterator over this list (from head to tail). You do not need to implement a remove method, but feel free to do so for extra practice. (We will not be testing the Iterator's remove method.)

---

## Requirements

- **The Iterator you are defining may be used in your test code, but may NOT be used to implement any of the other methods of the BasicLinkedList class.** To be clear, you also may not use for-each loops anywhere in the BasicLinkedList class.
- **Your code must be efficient! Do not traverse the list if it is possible to avoid it.**
- You may not use recursion in this project.
- You are not required to write JUnit tests, but you are strongly encouraged to do so. If you have a problem with your code and need assistance during office hours, you will be expected to demonstrate a JUnit test that illustrates the problem you are facing.
- You need to implement a singly-linked list that relies on a head and tail reference. Making use of the Java collections framework or implementing the list in some other way will result in a grade of 0. (In particular, you may not use arrays, and you may not use ArrayList or any other Java collection, and you may not implement a doubly-linked list.)

---

## Submission

- Submit your project using the "Submit Project" option (available by right clicking on the project folder).

---

## Grading

- (6%) Public Tests
- (29%) Release Tests
- (55%) Secret Tests
- (10%) Style

---

## Academic Integrity

Please make sure you read the academic integrity section of the syllabus so you understand what is permissible in our programming projects. We want to remind you that we check your project against other students' projects and any case of academic dishonesty will be referred to the Office of Student Conduct.