# *COMP 210*
# *Lab 6 and Homework 5*

*Spring 2016*

This week's lab and homework combo get you started on project 1. You're goal is to design and implement a general purpose Min Heap implementation. In lab you'll set down the basic design work. The implementation is homework.

## *Lab 6 & Homework 5*

Begin by laying out the design for a Min-Heap[1]. This means declaring the standard MinHeap interface and implementing that interface on a class that uses an array-based implementation. All code should be properly documented and anything that isn't written for you by Eclipsed should be stubbed. A complete set of tests must be written before implementation begins. Before you start the implementation, submit your design work as assignment *lab6* via handin. Submit the finished program as assignment *hwk5* by class time Wednesday 3/1.

[1] We'll discuss this in class.

### *Towards Generics*

A general purpose Min heap can contain any Class of data so long as instances of that class can be compared to one another. Java provides a standard interface for such objects. It's called *Comparable*[2]. So, in object-oriented terms, we want to design a Min Heap ADT that works for any implementation of the comparable interface. By designing the Min Heap hierarchy around an interface-based type we reduce the contained type to the minimal functionality required to work within the ADT.

[2] `https://docs.oracle.com/javase/7/docs/api/java/lang/Comparable.html`

This form of generalization has a trade off. Any place we need or want to work in terms the contained type we must do so strictly at the Comparable type level. For example, returning an element of the heap will mean returning a Comparable. At the user level this will mean type casting the returned value to something more specific whenever a basic comparable won't do. Construction and equality testing are also a bit problematic as the most we can say in the implementation is that we have an array of Comparable-typed data. It's also entirely possible to construct heaps from a heterogeneous data types so long as each type is comparable. If this occurs and the contained types aren't comparable to each other[3] then the heap will break. We'll work with these shortcomings for now and look for a work around and fixes as we move forward.

[3] i.e. I can compare strings to strings and Integers to Integers but not Integers to Strings