# Computer Science 136

Data Structures

Lecture #16 (October 25, 2021)

1. Questions?
2. Recall: Example of Linear use: solving mazes.
   (a) Stacks lead to *depth-first search* and possible fast termination.
   (b) Queues lead to *breadth-first search* and will find shortest solution before any other.
3. The `Comparable` interface.

```
public interface Comparable<T>
{
    public int compareTo(T that);
    // post: returns value <, ==, > 0 iff
    // this logically <, ==, > that respectively

    public boolean equals(Object that);
    // pre: that is an extension of type T
    // post: returns true iff this equals that
}
```

   (a) Implemented by objects that may be ordered, in a natural way.
   (b) Must implement a `compareTo` method, in a manner similar to the `compare(a,b)` method of the `Comparable` interface.
   (c) Good habit: also override the`equals` method. Often the following is satisfactory:

```
public boolean equals(Object that)
{
    return 0 == this.compareTo((T)that);
}
```

   (d) Note the apparant inconsistency of types for `equals` and `compareTo`. `equals` is a method of `Object` (and thus a method of all classes) and must take type `Object`. `compareTo` is a (potentially) new method and takes on the type of the *most general types to be compared*. Frequently, this is type `T`, but it may be a supertype of `T`.
4. Ordered structures.
   (a) Structures that keep their elements in order.
   (b) The `OrderedStructure` interface.
      i. Simply demands that the types stored within the associated class be `Comparable`, either directly, or through extension.
      ii. There are no methods demanded by this interface.
   (c) The `OrderedVector` class.
      i. Is not a `Vector`. Why?
      ii. Extends `Structure` and thus must implement `add`, `remove`, `contains`, `clear`, `isEmpty`, and `size`.
      iii. Makes significant use of a method `locate` that performs *binary search* (using `compareTo`) to find either the element, or the appropriate location to insert the element. This method runs in $O(\log n)$ time.
   (d) The `OrderedList` class.
      i. A linked list implementation of `Structure` and `OrderedStructure`, based on `Node`.
      ii. Makes use of the "finger" technique of iteration. Most operations are $O(n)$.
      iii. Makes use of a `Comparator` class.

---

**Notes:**