Discussion 5: Java Collections library
Mini-Lesson: ADTs and Java Collections library
ADTs, data structures, interfaces, classes

- **Abstract Data Types (ADTs):** *Operations, restrictions, and guarantees* for a collection of objects
  - Behavior is specified from client’s point of view
  - No implementation details!

- ex) A list is a collection of elements with a defined order.

- **ADT** operations can be declared and specified in a Java **interface**
Java's List\textlangle E\textrangle interface

- Interfaces for many ADTs in \texttt{java.util} package
  - Known as \texttt{Java Collections Framework}
- \texttt{Generic} interfaces – type parameter \texttt{E} for type of elements
- List operations:
  - \texttt{size()}  // not "length"
  - \texttt{get(i)}  // returns an \texttt{E}
  - \texttt{set(i, e)}  // \texttt{e} has type \texttt{E}
  - \texttt{add(i, e)}
  - \texttt{remove(i)}
  - \texttt{contains(e)}
ADTs, data structures, interfaces, classes

• **ADT** operations can be declared and specified in a Java **interface**

• A Java **class** implementing such an interface will use **data structures** to implement that functionality

• Multiple classes can implement the same interface using different data structures
List implementations

• **JavaDoc**: All Known Implementing Classes
  • `ArrayList<E>`: Uses a resizable array
  • `LinkedList<E>`: Uses a (doubly) linked list

• All support the same core operations
Other collection ADTs

• **Collection<E>**
  - Keeps track of objects that have been added, but does not remember order

• **Set<E>**
  - A collection with no duplicates. Common operation: `contains(e)`

• **SortedSet<E>**
  - Iteration order is guaranteed to be sorted (according to value comparisons)

Data structures for these (binary search trees, hash tables) will be taught later, but as a *client*, you can use them now (**HashSet**, **TreeSet**)
Hindsight on A2...

• Can replace CMSu’s arrays of Students and Courses with Lists

```java
/**
 * List of all the courses managed by this Course Management System (CMS). The index of a
 * course in the array is used as unique public identifier. Only the first `nCourses` elements
 * are valid; remaining elements are null.
 */
private List<Course> courses;
```

• Can replace StudentSet by leveraging standard class with a custom parametric type
  • Or could implement StudentSet using a field of type Set<Student> - composition
Iterating over collections

• Common operation for all collections: ability to enumerate all elements (order may be unspecified)

• Most convenient: "enhanced for-loop"

```java
Collection<String> c = ...;
for (String s : c) {
    // Use s
}
```

• Uses Iterators under the hood: hasNext() & next()
Enhanced for-loops are translated into while loops

```java
List<String> names = ...;
for (String name : names) {
    ...
}
```

```java
List<String> names = ...;
Iterator<String> it = names.iterator();
while (it.hasNext()) {
    String name = it.next();
    ...
}
```