Collections

Storing Data

- Arrays
  - Low overhead, but fixed size
  - Access elements using index
- Collections
  - Higher overhead, unlimited size
  - Access elements using element, index, or key
  - Convenience methods to add multiple elements, check if an element is already in the collection
  - Elements might be sorted or directly accessed

Interfaces

- Define mandatory methods
- No implementation
- Provide an outline of required features

- Thus, all implementing classes
  - All have a subset of common methods (the ones defined in the interface)
  - Provide the implementation based on task requirements or design decisions

Collections

- List
  - Elements in sequence, duplicates allowed
- Set
  - Element order depends on implementation
  - Duplicates not allowed (like a mathematical set)
- Map
  - “Maps” a unique key to a value and stores the pair (e.g., a map of student numbers to student records)
  - Element order depends on implementation
- Stack and Queue
  - Covered in EECS2030

Implementations

- ArrayList
  - Each element has its own index (like array)
- TreeSet
  - Elements are sorted smallest to largest (ascending)
- TreeMap
  - Keys are sorted smallest to largest (ascending)
- HashSet and HashMap
  - No sorting, faster than “tree” collections

Primitives Not Allowed!

- All elements in a collection ***must*** be objects
- Wrapper classes are used for primitive values
- Automatic boxing and unboxing gives the illusion that primitives are allowed

Generics

- Allows programmers to specify type of elements in the collection

```
ArrayList<Integer> list = new ArrayList<Integer>();
```

- Type checking done at compilation

```
list.add("Hello");  // not allowed
```

- Collections can be reused for any type of elements

Examples

- Code available on course website
  - ArrayListExample
  - TreeSetExample
  - TreeMapExample

- Code demonstrated in lecture

What Makes Objects Sortable?

- When adding elements, TreeSet and TreeMap implicitly call the compareTo method
- The compareTo method returns a value indicating the sorted order of two objects
- Not every type of object can be compared
- How to ensure a class has compareTo?
  - The class implements the Comparable interface

Application of Collections

- Duplicate checking (sets)
- Pairwise comparison (arrays or lists)
- Frequency counting (maps)

- Homework exercises and pseudo-code online
- Code taken-up in lecture
- Solutions won't be posted, so take notes and try to reproduce the solution later on your own