CSIS 3103

Chapter 2: Collections of Data

Collections

- Realistic computing problems usually deal with large collections of data elements
- Arrays are built-in data structures for storing collections of data
  - Probably the first data structure you learned

Array

Array properties
- Fixed size
- Homogeneous
- Mutable (can be changed)
- Indexed (access elements using subscripts)

Java arrays are objects

Declare with the `[]` suffix:
```
int[] a;
```
Allocate with `new`:
```
a = new int[8];
```
`length` field is accessible:
```
int numElements = a.length;
```
Can declare, allocate, and initialize all in one:
```
int[] a = {44, 77, 88, 33};
```

Array of Object References

```
Object [] objArr = new Object[4];
objArr[0] = new Double(9.0);
objArr[1] = new Time();
objArr[2] = "Hello there";
objArr[3] = new int[] {9, 8, 7, 6, 5};
```

What is the output of this loop?

```
for (Object obj : objArr) {
   System.out.println(obj);
}
```

What is the result of the following statement?

```
objArr[4] = "Oh no!";
```

Disadvantages of Arrays

You cannot
- Add an element at a specified position without shifting the other elements to make room
- Remove an element at a specified position without shifting other elements to fill in the resulting gap
- No abstraction, you need to work directly with the low-level details of the array

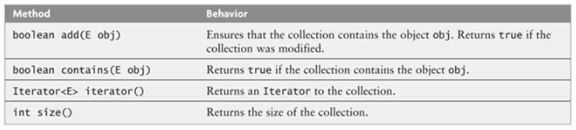
The Java Collections Framework

- The Java `Collection` interface represents various groups of objects
- Different implementations of `Collection` organize elements in different ways
  - Are duplicate elements allowed?
  - Are the elements ordered?

Common Features of Collections

The `Collection` interface specifies a set of common methods

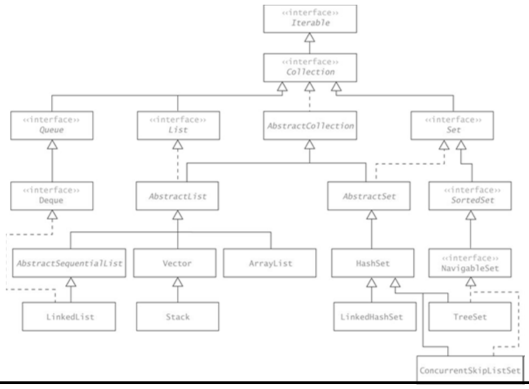
A few example methods:



Method	Behavior
`boolean add(E obj)`	Ensures that the collection contains the object `obj`. Returns `true` if the collection was modified.
`boolean contains(E obj)`	Returns `true` if the collection contains the object `obj`.
`Iterator<E> iterator()`	Returns an `Iterator` to the collection.
`int size()`	Returns the size of the collection.

Collections grow as needed
They store references to objects

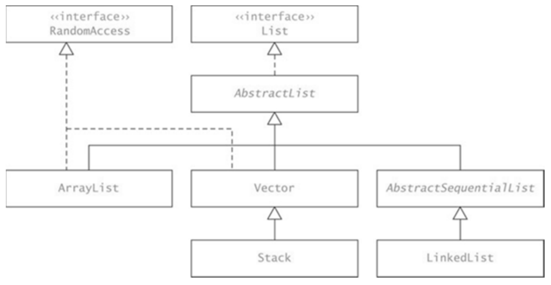
The Collection Hierarchy



Implementing Collection ADTs

1. Use an interface (*e.g.* `Collection`, `List`,) to define the structure.
2. Partially implement that interface with an `abstract` class.
   Implement all methods do not require knowing the object's storage structure.
3. Complete the implementation by extending the `abstract` class with concrete classes that do specify the storage structure.

The `List` Interface and Implementing Classes



The `List` Interface

- The `Collection` interface specifies a subset of the methods specified in the `List` interface
- `ArrayList`, `Vector` and `LinkedList` represent a collection of objects that can be referenced by means of an index
  - *Subclasses* of abstract class `AbstractList` and *implement* the `List` interface

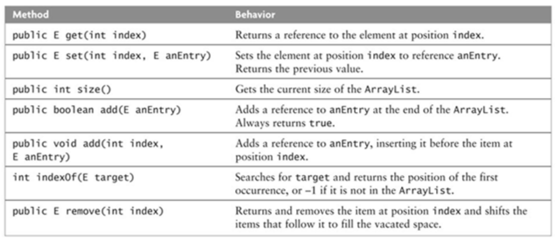
The `List` Interface

Operations in the `List` interface include:

- Finding a specified target
- Adding an element to either end
- Removing an item from either end
- Traversing the list structure without a subscript

Not all classes perform the operations with the same degree of efficiency

The `ArrayList` Class (partial specification)



Method	Behavior
`public E get(int index)`	Returns a reference to the element at position `index`.
`public E set(int index, E anEntry)`	Sets the element at position `index` to reference `anEntry`. Returns the previous value.
`public int size()`	Gets the current size of the `ArrayList`.
`public boolean add(E anEntry)`	Adds a reference to `anEntry` at the end of the `ArrayList`. Always returns `true`.
`public void add(int index, E anEntry)`	Adds a reference to `anEntry`, inserting it before the item at position `index`.
`int indexOf(E target)`	Searches for `target` and returns the position of the first occurrence, or –1 if it is not in the `ArrayList`.
`public E remove(int index)`	Returns and removes the item at position `index` and shifts the items that follow it to fill the vacated space.