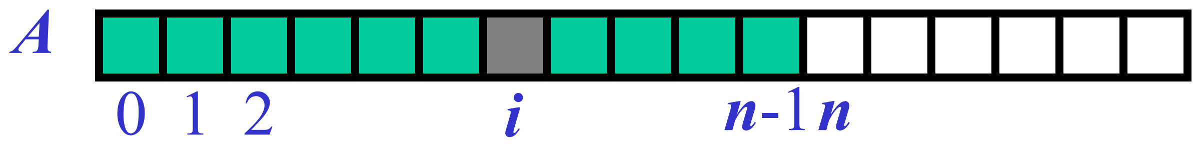
CS206

ArrayList

Array

- An array is a sequenced collection of homogenous variables (elements)
- Each element of an array has an index
- The entire array is contiguous in memory
  - allocated by new (e.g., new int[10])
- The length of an array is fixed and can not be changed



Array/List

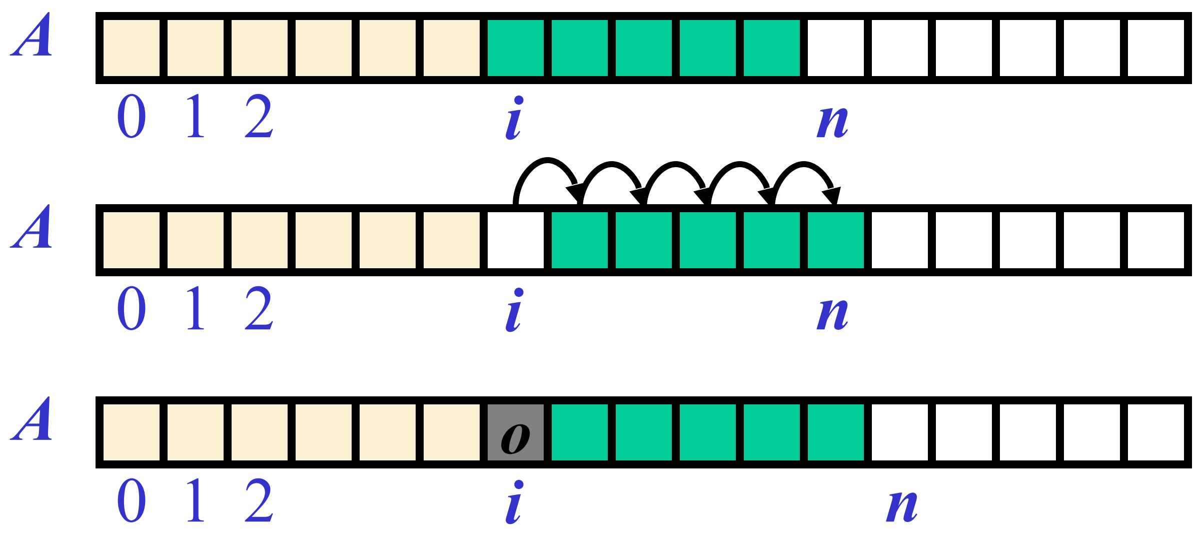
- Dynamically-sized array
- Stores an ordered sequence of objects
  - **Not sorted**, ordered in the sense that arrays are ordered
- Can grow and shrink when items are added/removed
- Standard array features all supported, but with different syntax

Array/List

- ArrayList is implemented with an array
- A variable (call it `count`) keeps track of the number of elements in the ArrayList
  - deletion
    - shift elements to the left and decrement `count`
  - addition
    - put new item on end and increment `count`
    - if not enough space
      - Create new, bigger array
      - Copy elements of old array into new one

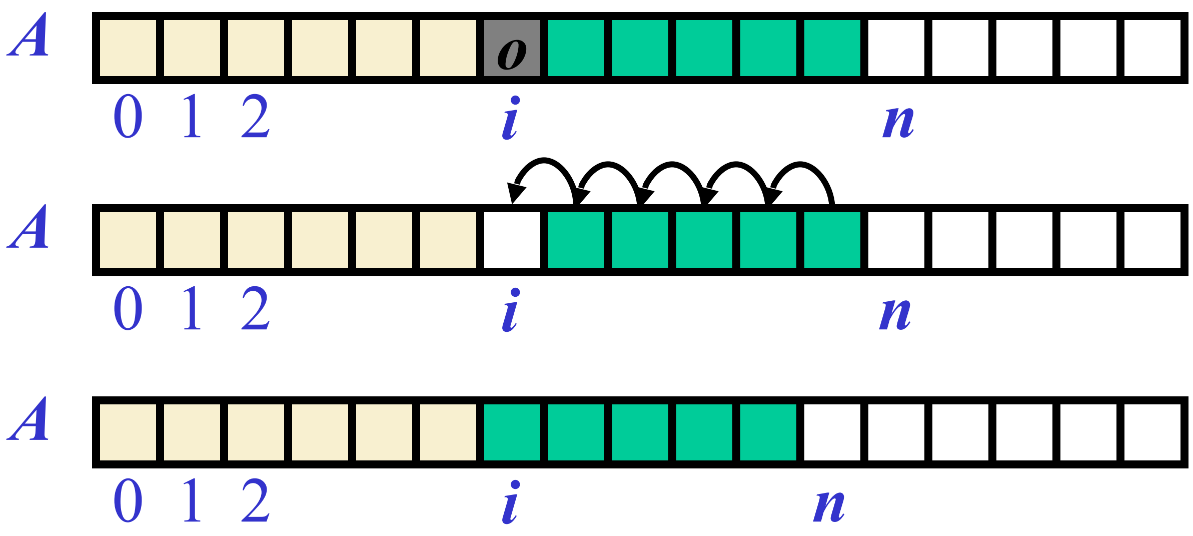
Insertion

- In an operation `add(i, o)`, we make room for the new element by shifting forward/to the right the elements `A[i]`, …, `A[n - 1]`



Deletion

- In an operation `remove(i)`, we fill the hole by shifting backward/to the left the elements `A[i + 1]`, …, `A[n − 1]`



Java Interfaces

- Java allows only single inheritance.
  - A class can only extend one class
  - As a result, Java does not need any collision resolution.
- BUT a class can “implement” any number of Interfaces
  - Interfaces only define methods
    - they do not provide method bodies so no collision resolution required.

Interface for ArraList

```java
public interface ArraListInterface<T> {
    boolean add(T t);
    void add(int index, T t) throws IndexOutOfBoundsException;
    T get(int index) throws IndexOutOfBoundsException;
    void remove(int index) throws IndexOutOfBoundsException;
    boolean set(int index, T t) throws IndexOutOfBoundsException;
    int size();
    int indexOf(T t);
    void clear();
}
```

handout with whole interface

Implementing ArraListInterface

```
public class ArraList<T> implements ArraListInterface<T> {
    private int capacity = 10;
    private static final double GROWTH_RATE = 1.618033; // the gol
mean
    private int count; // number of items currently in ArraList
    private T[] arra; // the array underlying the ArraList
    public ArraList() {
        arra = (T[]) new Object[capacity];
        count=0;
    }
    public ArraList(int initialCapacity) {
        capacity = initialCapacity;
        arra = (T[]) new Object[capacity];
        count=0;
    }
}
```

Class implements add(t,i), remove(i)

More implemtation of ArraList

```
public boolean add(T t) {

}
```

Suggestion: start by drawing a good picture of what you want to do

YM implementation of ArraList

```
public void remove(int index)
      throws IndexOutOfBoundsException {

}
```

Creation with Type Parameters

- When constructing an `ArraList`, you must specify the type of elements via `<>`

```
ArraList<String> l1 = new ArraList<>();
ArraList<Integer> l2 = new ArraList<>()
```

Example usage

- Write a program to collect then print all unique words in a file
- Problem: you do not know the number of distinct words!
  - Solution
    - allocate a really big array
    - Use ArraList!

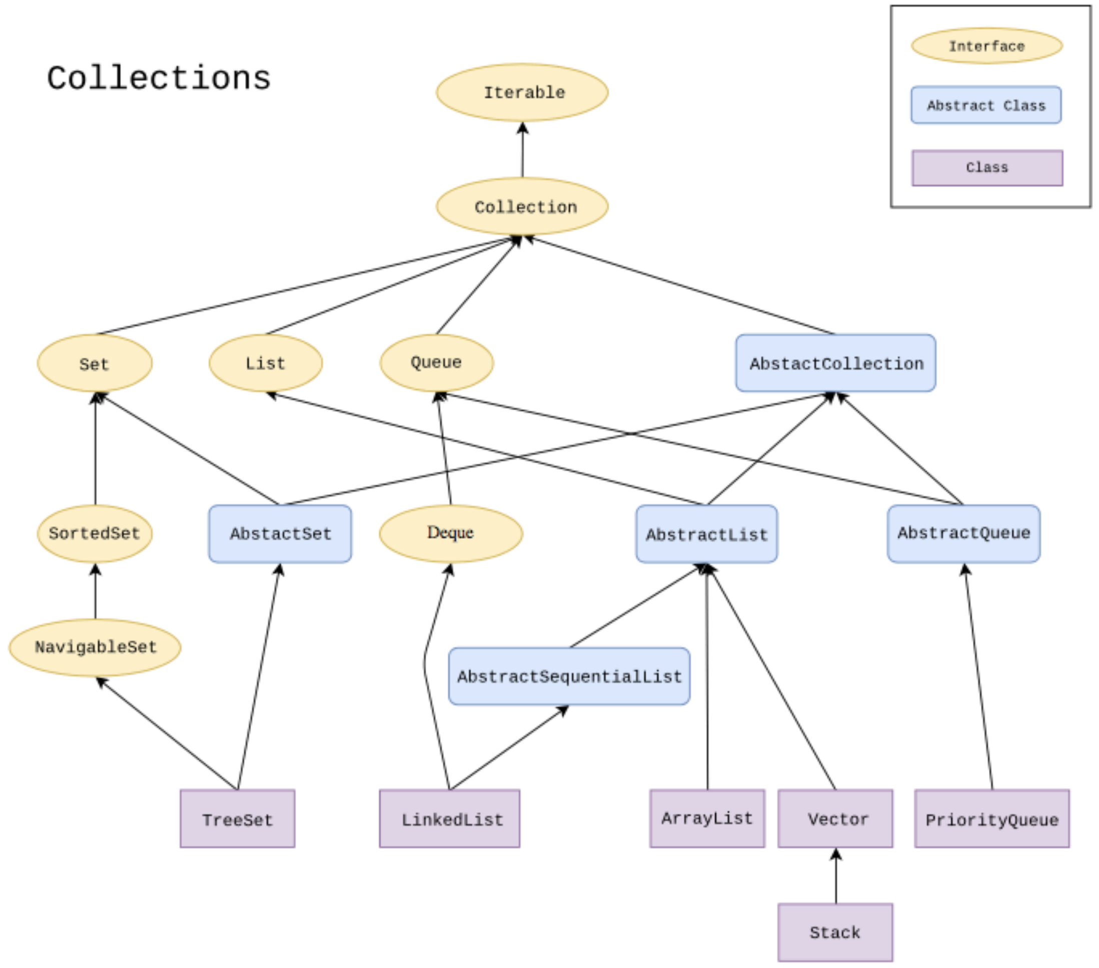
WordCounter — Count the unique words in file!

WordCounter.java

java.util.ArrayList

- Implements much the same interface as ours
  - Their implementation has a few more functions
- Theirs is probably more more efficient.
- Part of Java collections framework
- import java.util.ArrayList
- Use ArrayList rather than ArraList (ours) for Homework 3 and Lab 2.

Collections


Collections
Iterable
Collection
Set
List
Queue
AbstactCollection
SortedSet
AbstactSet
Deque
AbstractList
AbstractQueue
NavigableSet
AbstractSequentialList
TreeSet
LinkedList
ArrayList
Vector
PriorityQueue
Stack
Interface
Abstract Class
Class