Finishing (finally!) the Vector class.

1. Questions?

2. Notes on final project:
   (a) Visit during office hours (today 2:30-4, 7-9, Fri: 9-10), or by appt.
   (b) Make sure you carefully document your project in README, including any new packages you had to install.
   (c) Make sure you include artifacts created, or document how to create them.
   (d) Add and commit files. Check your status with git status.
   (e) Push your project by Friday at 5.

3. Exam
   (a) Self schedule, 2.5 hours, Tuesday 8am (earliest checkout) - 5pm (latest checkin).
   (b) Study session Tuesday 7-8:30pm, TPL 205.
   (c) Scheduled: Sunday, May 21, 1:30pm, Chemistry 202.

4. We finish building an extensible array structure.
   (a) Recall: Based on arrays of Objects. The length cannot be changed. Instead, we allocate new space, periodically, and keep it up to date.
   (b) Recall: We keep track of data, size, capacity.
   (c) Recall: ensureCapacity(int n) allows us to check to see if a Vector has the ability to hold n values.
   (d) We continue by implementing the methods on the reverse of this page.
   (e) We will constantly worry about the time and space required.
   (f) In languages like Java (and Python), unused objects are silently collected and their memory is recycled. In our balloon analogy, balloons (objects) that have no attached strings (references) can’t be re-attached (rereferenced); the system is free to recycle them. Still, we have to be attentive to possible memory leaks.
   (g) We need to make sure that when we introduce new data in a linear structure (like a Vector or linked list) that we move existing data with care. If we lose a reference, the garbage collector comes to call....
   (h) When we check if a container contains an object, we use equals. When equal objects are removed, we always return the removed object for inspection by the user. This is a subtle point.
public class Vector
{
    private Object data[];
    private int capacity;
    private int size;

    public Vector();
    public int size();
    public boolean isEmpty();
    public int capacity();
    private void ensureCapacity(int n);
    public String toString();
    public void add(Object v);
    public Object get(int n);
    public void set(int n, Object v);
    public void clear();
    public void add(int n, Object v);
    public Object remove(int n);
    public int indexOf(Object o);
    public boolean contains(Object v);
    public Object remove(Object o);
    public boolean equals(Object otherV);
}

*