One-third-way point of the semester:

- Documentation; Java GUI components (last week Monday)
- Git; abstract data types (last week Wednesday)
- More abstract data types; Java Collections (last week Friday)
- Review (**Today**)
- Test 1 (Friday)
- Object-oriented design and class extension (next week)

Today:

- Java collections summary
- C structs, strings, and arrays
- Linked lists plus invariants and algorithmic analysis
- Java/OO vocabulary

Coming up:

- **Due Wed, Feb 18 (today).** *Do Project 2, "First Calculator."*
- **Due Thurs, Feb 19, 1:15pm.** *Take Canvas quiz on Java collections. (No prelab reading.)*
- **Due Fri, Feb 27.** *Do Project 3, "Homemade Linked-list Map." (Recommendation: Do this by Fri, Feb 20, to help prepare for the test.)*
- **Due Fri, Mar 6.** *Do Project 4, "Text-based adventure game." (You'll have the next three lab periods to work on this... don't work on it outside of class yet.)*

```
interface java.util.List<E> {
    java.util.Iterator<E> iterator();
    boolean add(E);
    E get(int);
    E set(int, E);
    E remove(int);
}
```

```
interface java.util.Set<E> {
    int size();
    boolean isEmpty();
    boolean contains(Object);
    Iterator<E> iterator();
    boolean add(E);
    boolean remove(Object);
    void clear();
}
```

```
interface java.util.Map<K,V> {
    boolean containsKey(Object);
    boolean containsValue(Object);
    V get(Object);
    V put(K, V);
    V remove(Object);
    Set<K> keySet();
}
```

```
class java.util.ArrayList<E>
   implements List<E> {
    void trimToSize();
    void ensureCapacity(int);
    int size();
    boolean isEmpty();
    int indexOf(Object);
    int lastIndexOf(Object);
    E get(int);
    E set(int, E);
    boolean add(E);
    E remove(int);
}
```

```
class java.util.HashSet<E>
   implements Set<E> {
    Iterator<E> iterator();
    int size();
    boolean isEmpty();
    boolean contains(Object);
    boolean add(E);
    boolean remove(Object);
    public void clear();
}
```

```
class java.util.HashMap<V,K>
   implements Map<V,K> {
    boolean containsKey(Object);
    boolean containsValue(Object);
    V get(Object);
    V put(K, V);
    V remove(Object);
    Set<K> keySet();
    int capacity();
}
```

```
interface java.lang.Iterable<T> {
    Iterator<T> iterator();
}
```

```
interface java.util.Iterator<E> {
    boolean hasNext();
    E next();
}
```

```
interface java.lang.Comparable<T> {
    public abstract int compareTo(T);
}
```

```
interface java.util.Comparator<T> {
    int compare(T, T);
    boolean equals(Object);
}
```

Finish the following class that implements a "homemade" map interface and uses internal Lists to store keys and values in parallel.

```
public interface HomemadeMap {
    String get(String key);
    void put(String key, String value);
}
public class ALMap implements HomemadeMap {
  private List<String> keys;
  private List<String> values;
  public ALMap() {
    this.keys = new ArrayList<String>();
    this.values = new ArrayList<String>();
  }
  public String get(String key) { ...
  }
  public void put(String key, String value) { ...
  }
```

Consider this C struct as defined in a *header file*.

```c
/* --- nutrition_info.h --- */
struct nutrition_info
{
  char product_name[];
  int calories; // K
  double protein; // grams
  int vit_A; // % DRV
  int vit_B; // % DRV
  int vit_C; // % DRV
  int vit_D; // % DRV
};

int vitamin_list(struct nutrition_info product, char vitamins[]);
```

The function `vitamin_list` takes a value of `struct nutrition_info` and a char array and writes to the given char array the names of the vitamins for which the given product has more than 0% of the daily recommended value and returns number of vitamins written.
Write the implementation of the function `vitamin_list` as it would appear in the *implementation file*.

Consider the method below that converts a linked list to an array. Let n be the number of nodes in the given linked list. Find invariants for the loops; analyze the running time by annotating the code in a line-by-line analysis, find a function for the running time as a function of the size of the linked list, n, and determine a big-oh category fo the running time.

```
int[] linkedListToArray1(Node head) {

    int len = 0;
    for (Node current = head; current != null; current = current.next())
        len++;

    int[] toReturn = new int[len];

    Node current = head;
    for (int i = 0; i < toReturn.length(); i++) {
        toReturn[i] = current.datum();
        current = current.next();
    }

    return toReturn;
}
```

- Basic Java and OO vocabulary:
  - Object
  - Class
  - Member
  - Instance variable
  - Instance methods
  - Instance of a class
  - Composite type
  - Reference type
  - Method
  - Receiver
  - Constructor
  - Overloading
  - Static method
  - Static variable
  - Static initializer
  - Method signature
  - State class vs value class
  - Interface (concept)
  - Subtype, supertype
  - Polymorphism
  - Static type vs dynamic type
- Java language features: `this`, access modifiers (`public`, `private`), `iterface` (construct).
- The difference among *kinds* of variables: Local variable, instance variable, static variable, formal parameter.
- Kinds of members of a class: instance variable, instance method, constructor, static