Notes: ArrayList

ArrayList

- ArrayLists are like native arrays [] because
  - they both store many elements at one time.
- ArrayLists are different than native arrays [] because
  - the size can be variable for ArrayLists
    - length is fixed for arrays
  - ArrayLists should contain reference data types (Objects) not primitive data types
    - arrays can store either primitive or reference data types
  - ArrayLists have built in methods (Links to an external site.)Links to an external site. that can be called to perform common functions
    - ArrayLists in java are similar to Lists in python
    - arrays do not have built in functions (you cannot use dot-notation on an array)

Constructing

- Must import java.util.ArrayList
- When constructing an ArrayList you must specify the type of elements it will contain
  - This type is called a type parameter or a generic
- You will need to use Wrapper classes when storing int, double, char, and boolean
  - A wrapper is an object whose sole purpose is to hold primitive value
  - Once you construct the list, use it with primitives as normal
    - Java does something called auto boxing/unboxing which means that it converts between primitives and their wrapper class

Primitive Type	Wrapper Type
int	Integer
double	Double
char	Character
boolean	Boolean

Structure

```
ArrayList<Type> name = new ArrayList<Type>();
```

Example Code

```
ArrayList<String> names = new ArrayList<String>();
names.add("Frankie Manning");
names.add("Chick Webb");
```

ArrayList Methods

Method	Description
add(value)	appends value at end of list
add(index, value)	inserts given value just before the given index, shifting subsequent values to the right
clear()	removes all elements of the list
indexOf(value)	returns first index where given value is found in list (-1 if not found)
get(index)	returns the value at given index
remove(index)	removes/returns value at given index, shifting subsequent values to the left
set(index, value)	replaces value at given index with given value
size()	returns the number of elements in list
toString()	returns a string representation of the list such as [3, 42, -7, 15]

ArrayList vs Array

Description	Array	ArrayList
construction	String[] names = new String[5];	ArrayList<String> list = new ArrayList<String>();
storing a value	names[0] = "Martin";	list.add("Martin");
replace a value at an index	names[i] = "Martin";	list.set("Martin", i)
accessing a value	String s = names[0];	String s = list.get(0);
how many elements?	names.length	list.size();

ArrayList Code Examples

```
// moves the max value to the front of the given list, otherwise preserving the order of the elements
public static void maxToFront(ArrayList<Integer> list) {
  int max = 0;
  for (int i = 1; i < list.size(); i++) {
    if (list.get(i) > list.get(max)) {
      max = i;
    }
  }
  list.add(0, list.remove(max));
}
```

```
// returns the length of the longest String in the given list
public static int maxLength(ArrayList<String> list) {
    int max = 0;
    for (int i = 0; i < list.size(); i++) {
        String s = list.get(i);
        if (s.length() > max) {
            max = s.length();
        }
    }
    return max;
}
```

Account for re-indexing when removing from an ArrayList inside a loop

When you remove an element from an ArrayList, the ArrayList auto renumbers. You must account for, otherwise you will accidentally skip over processing elements (specifically the ones immediately after a remove).

Here are three examples of the same method re-rewitten to account for the reindexing

```
// Version 1: removes from the list all strings of even length
public static void removeEvenLength(ArrayList list) {
   int i = 0;
   while (i < list.size()) {
      String s = list.get(i);
      if (s.length() % 2 == 0) {
         list.remove(i);
      } else {
         i++;
      }
   }
}
```

```
// Version 2: removes from the list all strings of even length
public static void removeEvenLength(ArrayList list) {
   for (int i = 0; i < list.size(); i++) {
      String s = list.get(i);
      if (s.length() % 2 == 0) {
         list.remove(i);
         i--;
      }
   }
}
```

```
// Version 3: removes from the list all strings of even length
public static void removeEvenLength(ArrayList list) {
   for (int i = 0; i < list.size(); ) {
      String s = list.get(i);
      if (s.length() % 2 == 0)
         list.remove(i);
      else
         i++;
   }
}
```