CMSC 113: Computer Science I  
ArrayList Commands

Here are some of the commands available on ArrayLists. All of these examples assume you have an ArrayList<Foo> list = new ArrayList<Foo>(); set up. Here, Foo could be anything – something like Integer, Double, or perhaps GRect.

Before reviewing specific commands, we must first talk about the structure of an ArrayList. Each item in an ArrayList has an index. This index specifies where the item is stored in the list. The first possible index is 0. So, the first item stored in an ArrayList is item number 0. The second item stored is item number 1. After storing $n$ items, the last item is item number ($n-1$). The get and set commands below use this indexing scheme.

To add an item to an ArrayList:

```
list.add(newItem);
```

This line of code puts newItem at the end of our list.

To perform an operation to all elements of an ArrayList:

```
for(Foo someFoo : list)
{
    someFoo.doSomething();
}
```

Here, we are calling the method doSomething on every item in our list. This example might be more illuminating:

```
ArrayList<GRect> rects = new ArrayList<GRect>();
// insert a bunch of rectangles into the list

for(GRect oneRect : rects)
{
    oneRect.move(5, 0); // move a rectangle to the right
}
```

This code goes through the list and moves each rectangle in the list to the right by 5 units. The role of the word oneRect is to give a temporary name to each element. This is necessary so we have something before the word move.
To retrieve a specific item from an **ArrayList**:

```java
Foo someItem = list.get(index);
```

Here, `index` is an `int` variable storing some number between 0 and \((n-1)\), inclusive, where there are a total of \(n\) items in `list`. So, if we have the following code:

```java
ArrayList<Integer> intList = new ArrayList<Integer>();
intList.add(14);
intList.add(99);
intList.add(46);

int zero = intList.get(0);
int two = intList.get(2);
int one = intList.get(1);
```

After this code is run, `zero` will have the value 14; `two` will have the value 46; and `one` will have the value 99.

To change a specific item in an **ArrayList**:

```java
list.set(index, newItem);
```

Here, we are replacing the item stored with the given index with our new item. Here is a longer example:

```java
ArrayList<Integer> intList = new ArrayList<Integer>();
intList.add(14);
intList.add(99);
intList.add(46);

intList.set(1, 82);

int zero = intList.get(0);
int one = intList.get(1);
int two = intList.get(2);
```

Because of the `intList.set(1, 82);` line, the `ArrayList` contains the values 14, 82, and 46, in that order.

To get the number of items stored in an **ArrayList**:

```java
int numItems = list.size();
```

After this line, the variable `numItems`, will contain the total number of items stored in the ArrayList. The last usable index is `numItems - 1`. 
To clear the contents of an ArrayList:
    list.clear();

    After using this command, there are no items stored; list.size() is 0.

To remove an item from an ArrayList:
    list.remove(index);

    This command removes the item with the given index from the ArrayList. Because there cannot be holes in an ArrayList, all subsequent items (those with higher indices) will shift down.