ListIterator Interface

ListIterator

frequently used methods

Name	Use
next()	returns a reference to the next item
remove()	removes the last ref returned by next or previous
hasNext()	checks to see there are more items
add()	adds in a new item
set()	sets the last ref returned by next or previous
previous()	goes back and returns a ref to prev item

```
import  java.util.ListIterator;
```

ListIterators

```
ArrayList<String> words;
words = new ArrayList<String>();

words.add("at");
words.add("is");
words.add("of");
words.add("us");

ListIterator<String> it = words.listIterator();
System.out.println(it.next());
System.out.println(it.next());
```

OUTPUT

```
at
is
```

listiteratorone.java

previous()

```
ArrayList<String> words;
words = new ArrayList<String>();

words.add("at");
words.add("is");
words.add("of");
words.add("us");

ListIterator<String> it = words.listIterator();
System.out.println(it.next());
System.out.println(it.next());
System.out.println(it.previous());
```

OUTPUT

```
at
is
is
```

previousone.java

previous()

```
ArrayList<String> words;
words = new ArrayList<String>();
words.add("at");
words.add("up");
words.add("or");

ListIterator<String> it = words.listIterator();
it.next();
it.next();
it.next();
System.out.println(it.previous());
System.out.println(it.previous());
it.set("33");
System.out.println(words);
```

OUTPUT

```
or
up
[at, 33, or]
```

previoustwo.java

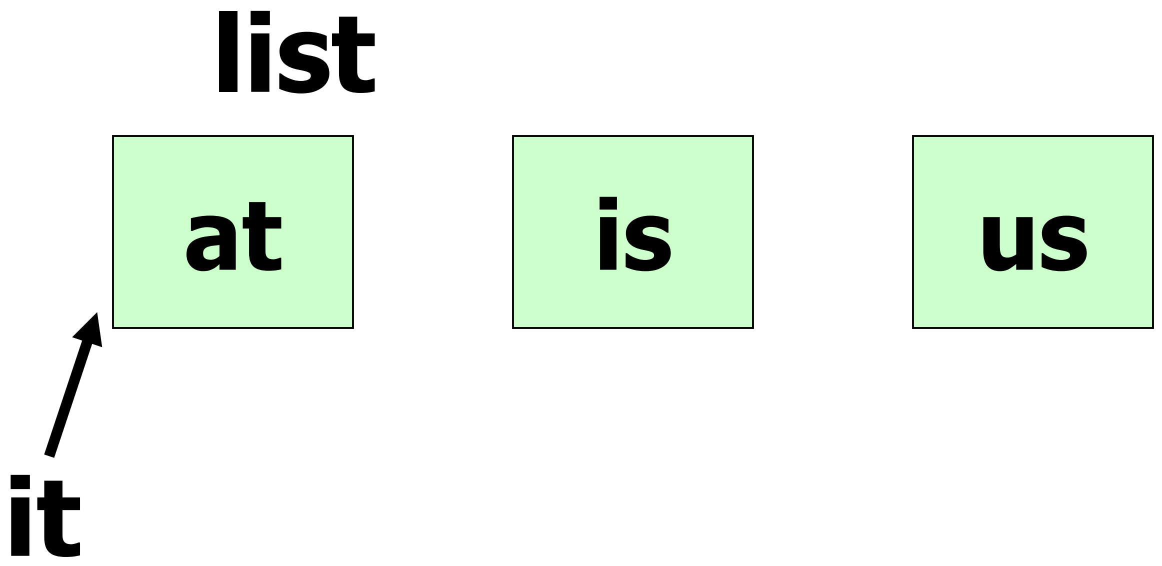
set()

```
ArrayList<String> words;
words = new ArrayList<String>();

words.add("at");
words.add("is");
words.add("us");

ListIterator<String> it = words.listIterator();
System.out.println(it.next());
it.set("###");
System.out.println(it.next());
System.out.println(words);
```

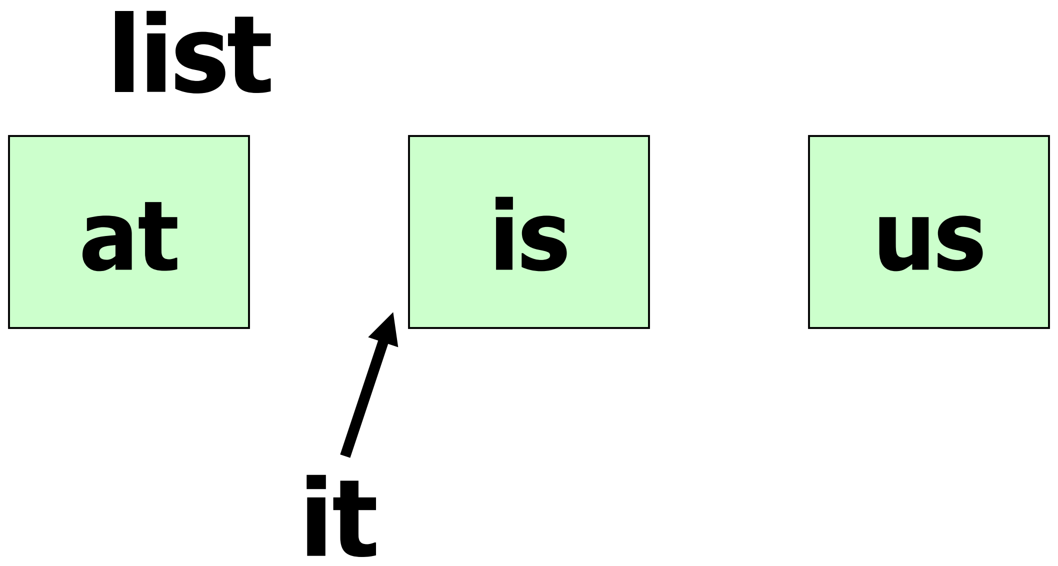
OUTPUT

```
at
is
[###, is, us]
```

set()



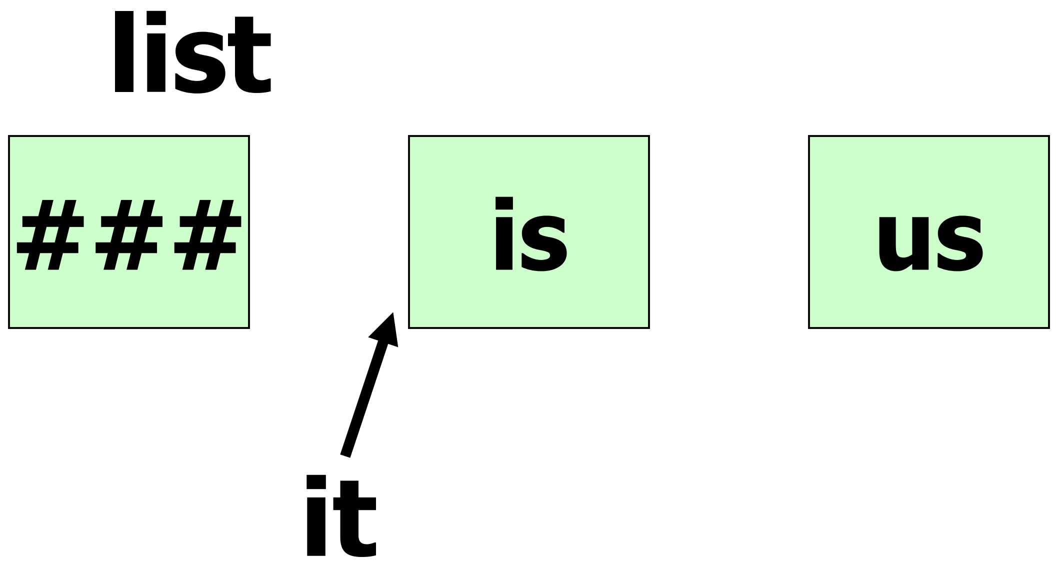
set()



```
it.next();
```

next moves the iterator up one spot and returns a reference to the 1st item.

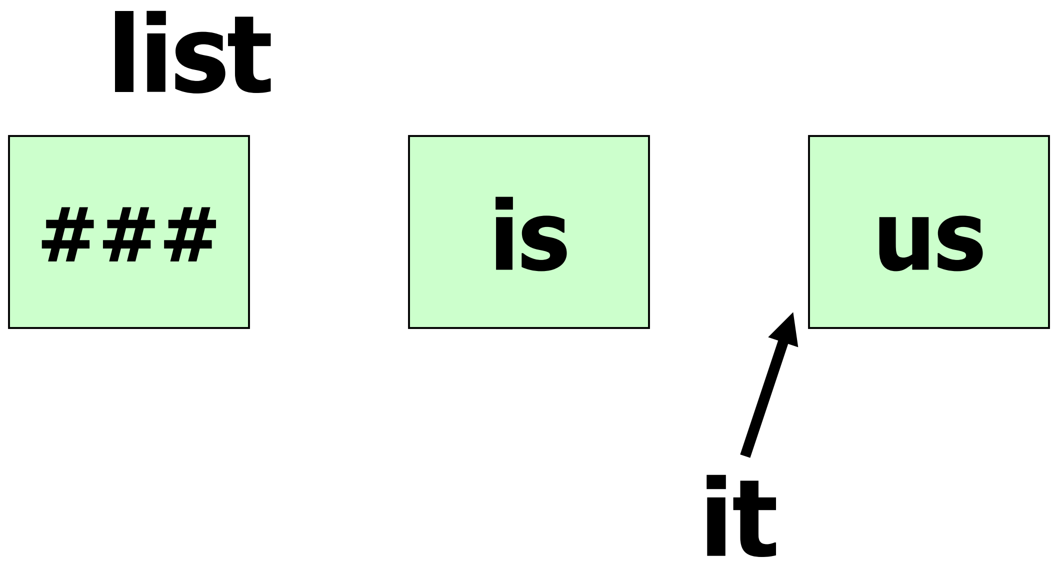
set()



```
it.set("###");
```

set always modifies the last reference returned by next or previous.

set()



it.next();

next moves the iterator up one spot and returns a reference to the 2nd item.

setone.java
settwo.java

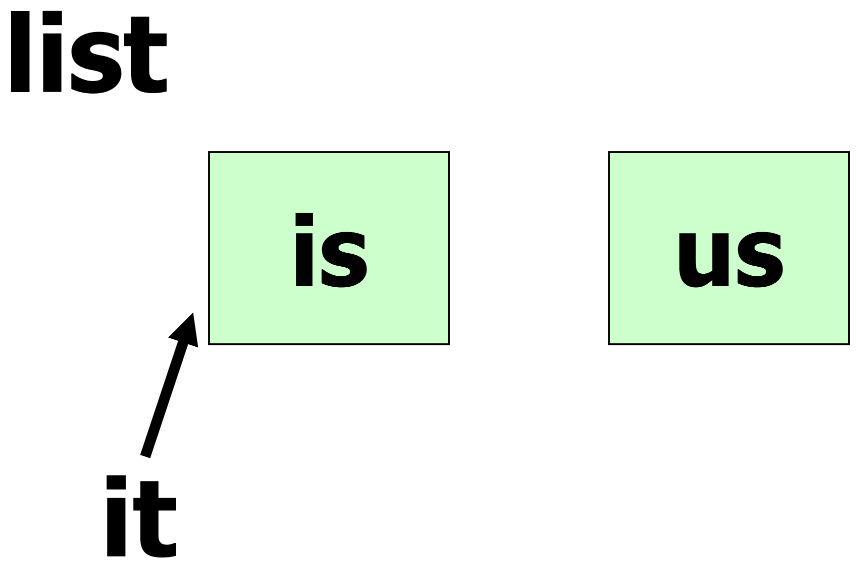
add()

```
ArrayList<String> words;
words = new ArrayList<String>();

words.add("is");
words.add("us");


ListIterator<String> it = words.listIterator();
it.add("##");
System.out.println(it.next());
System.out.println(it.next());
System.out.println(it.previous());
it.set("##");
System.out.println(words);
```

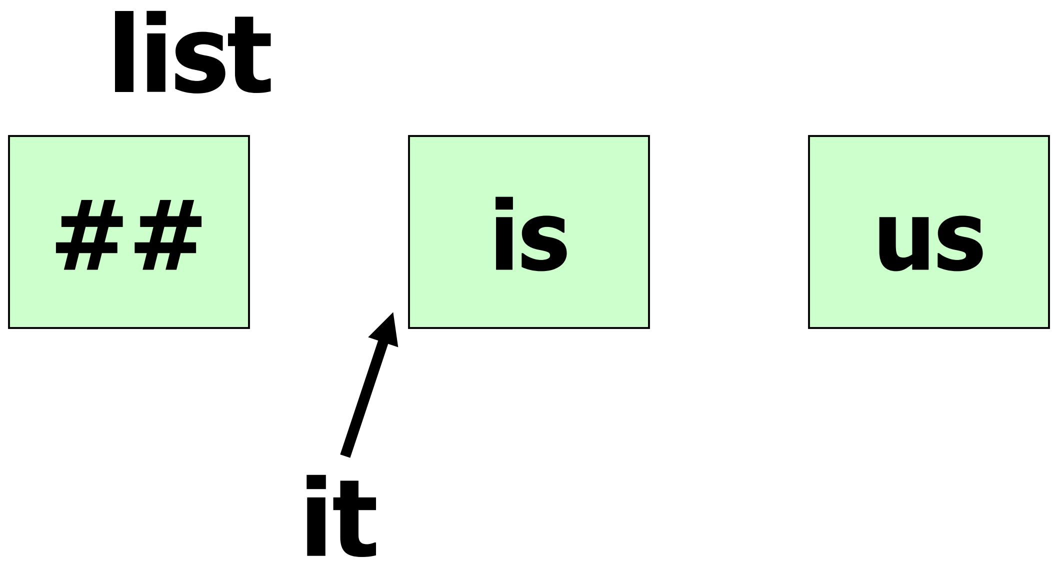
OUTPUT

```
is
us
us
[##, is, ##]
```

add()



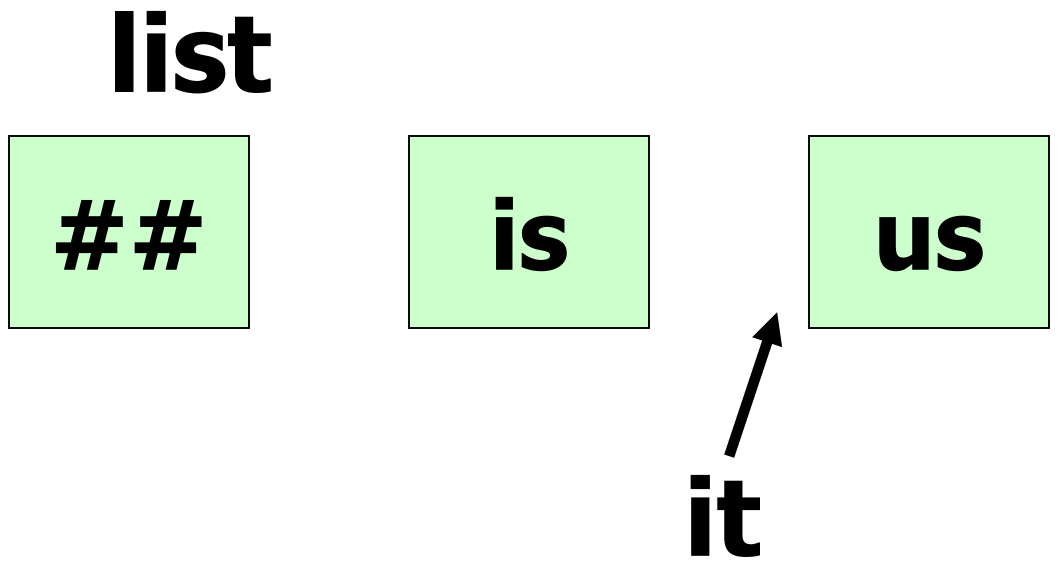
add()



```
it.add("##");
```

add always adds the new item in front of the current spot.

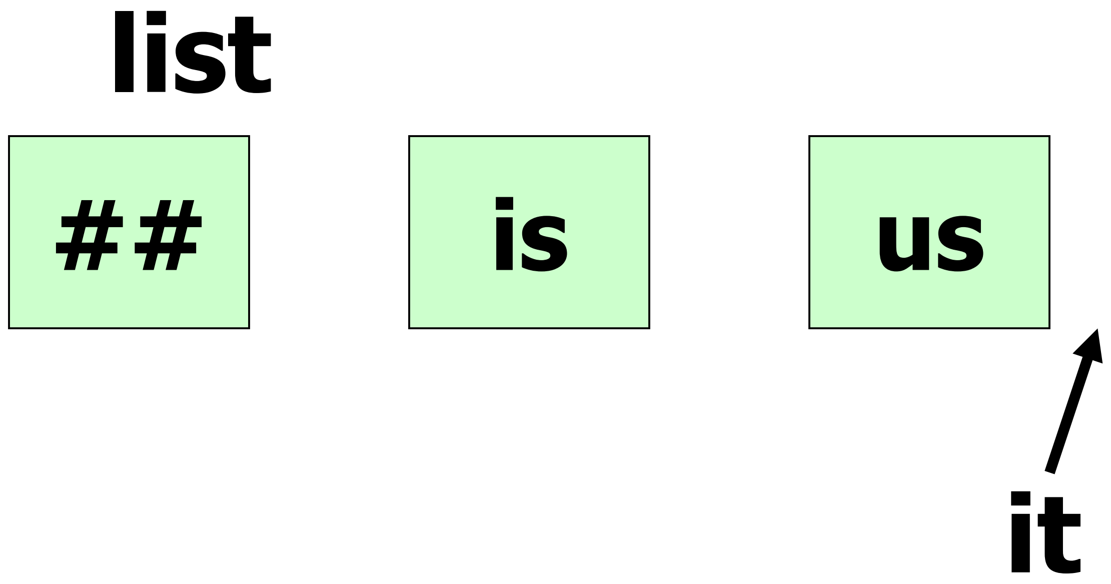
add()



```
it.next();
```

next moves the iterator up one spot and returns a reference to the 2nd item.

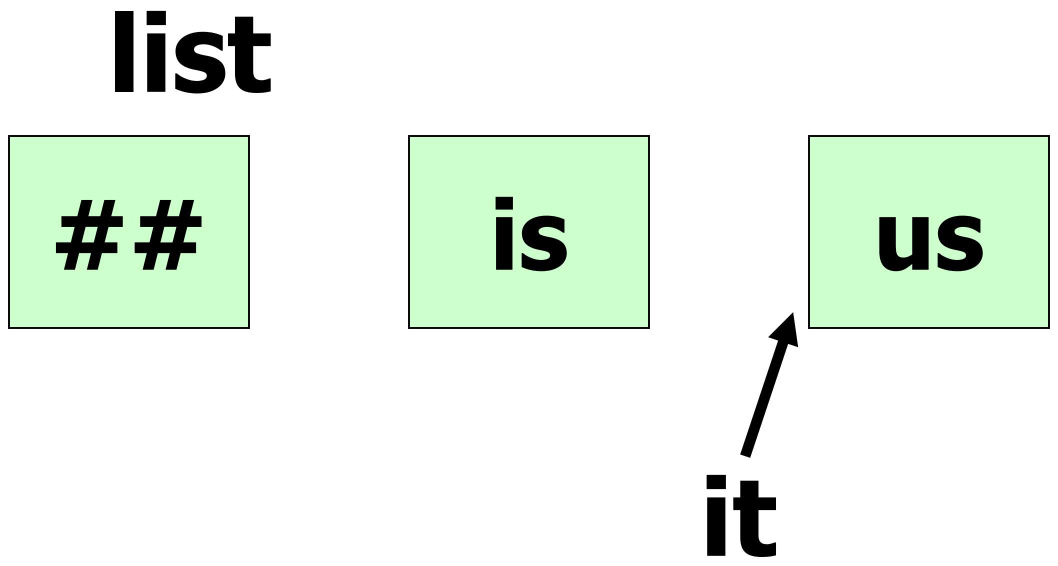
add()



it.next();

next moves the iterator up one spot and returns a reference to the 3rd item.

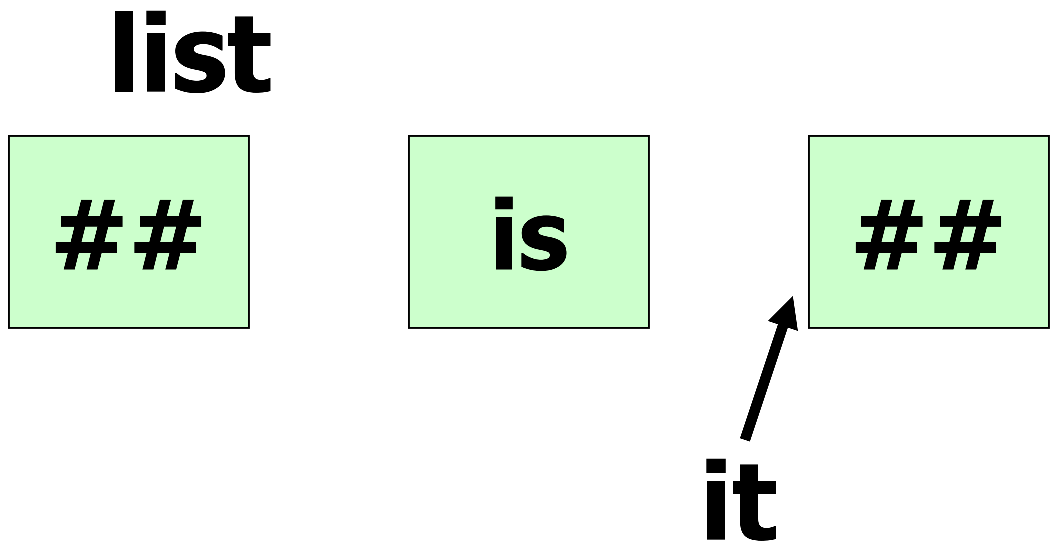
add()



it.previous();

previous backs the iterator up one spot and returns a reference to the 3rd item.

add()



```
it.set("##");
```

set always modifies the last reference returned by next or previous.

add()

```
ArrayList<String> words;
words = new ArrayList<String>();
words.add("is");
words.add("us");

ListIterator<String> it = words.listIterator();

it.add("5");
it.add("4");
it.add("3");
it.add("2");

System.out.println(words);
```

OUTPUT

```
[5, 4, 3, 2, is, us]
```

addone.java
addtwo.java

modification rule

Modifications through an Iterator or ListIterator are always applied to the reference returned by the last next or previous call.

Pay attention to the direction you are going.

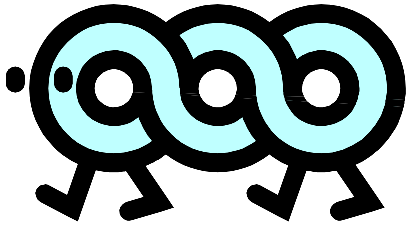
Iterator only goes one direction.
ListIterator can go either direction.

the new for loop

new for loop

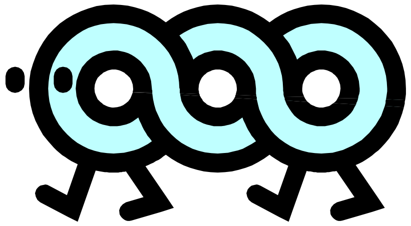
The new for loop allows you to specify a variable to store a value and a location from which to retrieve that value.

```
for (int num : array)

for (Integer value : list)
```



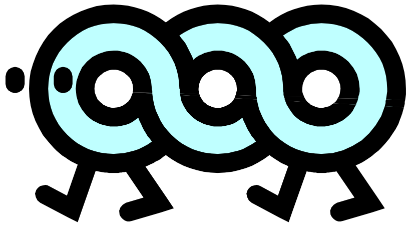
standard for loop

```
int[] array = {4,5,6,7};
int sum = 0;

for(int i=0; i<array.length; i++)
{
   sum += array[i];
}
```



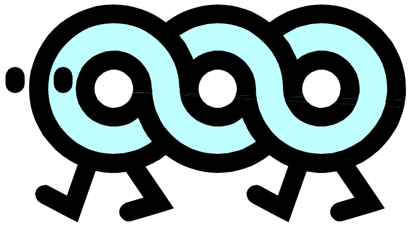
new for loop

```
int array[] = {4,9,6,2,3};
int sum = 0;

for (int num  : array)
    sum = sum + num;
System.out.println(sum);
```



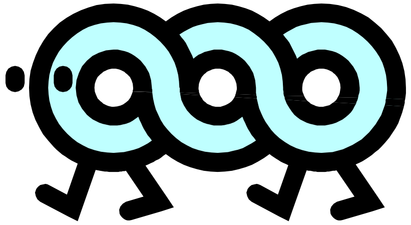
new for loop

```
ArrayList<Integer>  list  = new
                    ArrayList<Integer>();
list.add(3);
list.add(9);

for (Integer  num  : list)
    System.out.print(num + " ");
```



new for loop

```
ArrayList<Integer>  list  = new
                          ArrayList<Integer>();
list.add(3);
list.add(9);

for (int num  : list)
    System.out.print(num + " ");
```



newforone.java

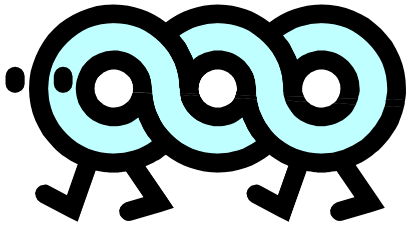
new for loop with Generics and Iterators

old way

```
ArrayList list = new ArrayList();

//add stuff to list

Iterator it = list.iterator();
while(it.hasNext())
{
  System.out.println(it.next());
}
```

new way

```
ArrayList<Integer> list;
list = new ArrayList<Integer>();

//add stuff to list

for(int number : list)
{
    out.println(number);
}
```



newfortwo.java

Start work on Lab 5