

```
List<String> lst = new ArrayList<String>();
lst.add("a"); lst.add("b"); lst.add("c");
for(String s: lst) {
    System.out.println(s);
}
```

Interface List<E>

All Superinterfaces: [Collection](#)<E>, [Iterable](#)<E>

```
public interface Iterable<T>
```

Implementing this interface allows an object to be the target of the enhanced `for` statement (sometimes called the "for-each loop" statement).

<u>Iterator</u> < <u>T</u> >	<u>iterator</u> ()	Returns an iterator over elements of type T.
--	------------------------------------	--

```
public interface Iterator<E>
```

An iterator over a collection.

boolean	<u>hasNext</u> ()	Returns true if the iteration has more elements.
---------	-----------------------------------	--

<u>E</u>	<u>next</u> ()	Returns the next element in the iteration.
--------------------------	--------------------------------	--

```
class AList<E> implements List<E>, Iterable<E> {
    class AListIterator implements Iterator<E> {
```

```
    }

    E[] elements;
    int size;

    @SuppressWarnings("unchecked")
    public AList() {
        this.elements = (E[])(new Object[2]);
        this.size = 0;
    }

    public Iterator<E> iterator() {

    }

    public void add(E s) {
        expandCapacity();
        this.elements[this.size] = s;
        this.size += 1;
    }

    public int size() {
        return this.size;
    }

    /* ... set, expandCapacity omitted ... */
}
```

```
// Goal:  
int sum = 0;  
for(Integer i: new Range(0, 100)) {  
    sum += i;  
}
```

```
// Idea:  
Integer sum(Iterable<Integer> iterable) {  
    int sum = 0;  
    for(Integer i: new Range(0, 100)) {  
        sum += i;  
    }  
    return sum;  
}
```

```
List<String> lst = new ArrayList<String>();  
lst.add("a"); lst.add("b"); lst.add("c");  
while(lst.hasNext()) {  
    String s = lst.next();  
    System.out.println(s);  
}
```

```
List<String> lst = new ArrayList<String>();  
lst.add("a"); lst.add("b"); lst.add("c");  
Iterable<String> iter = lst.iterator();  
while(iter.hasNext()) {  
    String s = iter.next();  
    System.out.println(s);  
}
```

```
List<String> lst = new ArrayList<String>();  
lst.add("a"); lst.add("b"); lst.add("c");  
Iterator<String> iter = lst.iterator();  
while(iter.hasNext()) {  
    String s = iter.next();  
    System.out.println(s);  
}
```

```
public E next() {  
    E answer = elements[this.currentIndex];  
    this.currentIndex += 1;  
    return answer;  
}
```

```
public E next() {  
    this.currentIndex += 1;  
    return elements[this.currentIndex];  
}
```

```
public E next() {  
    return elements[this.currentIndex];  
}
```

```
class Range implements Iterable<Integer> {  
    int currentIndex, low, high;  
    /* ... */  
}
```

```
class Range implements Iterator<Integer> {  
    int currentIndex, low, high;  
    /* ... */  
}
```

```
class Range implements Iterable<Integer> {  
    int low, high;  
    /* ... */  
}
```

```
class Range implements Iterator<Integer> {  
    int low, high;  
    /* ... */  
}
```

