

```
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).

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

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

An iterator over a collection.

boolean      hasNext()      Returns true if the iteration has more elements.

E      next()      Returns the next element in the iteration.

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

```
        int currentIndex = 0;
```

```
        AList<E> list;
```

```
        E next() {
```

```
            ...
```

```
        }
        boolean hasNext() {
```

```
            ...
```

```
    }
```

```
    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;
}
```

*← incl/excl?*

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

Check out  
Streams!

A

```
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);  
}
```

B

```
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);  
}
```

C



```
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);  
}
```

A

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

B

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

C

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

A

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

B

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

C

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

D

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

