

Chapter 4: Iterators

Iterators

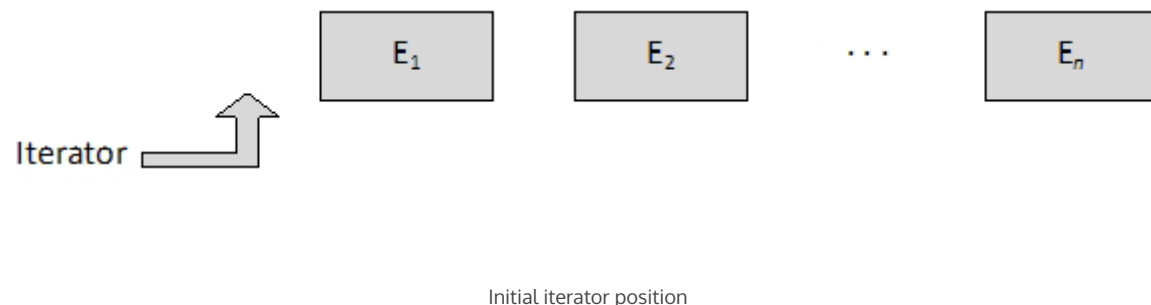
Every Java Iterator has at least three capabilities:

1. It can move forward through the list, getting us the next element in the collection.
2. It can tell us if our collection has any more elements in the forward direction.
3. It can delete elements from the collection without losing its place.

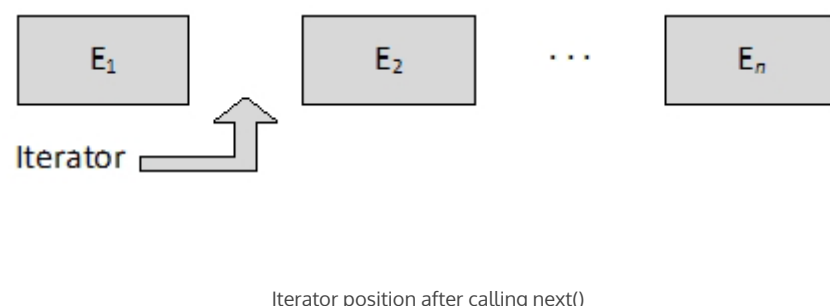
Java's lists have a ListIterator that can do two more things:

1. It can tell us if there are any elements in the backward direction.
2. In addition to moving forward, it can move backward through the list, getting us the previous list element.

We're going to use a ListIterator object to help us manage the list, track our current position, and get us the next or previous list element when we select those menu items. But first, let me show you visually how an iterator works. Assume we have a list with elements $E_1, E_2 \dots E_n$. When we get an iterator from the list, it points where the arrow below is. Iterators don't point *at* elements, they point *between* them.



This is the starting position of the iterator. As soon as we use the iterator's *next()* method to get the next element, it gives us back E_1 and positions the iterator between E_1 and E_2 , like this:



Another call to *next()* gets us element E_2 and puts the iterator between E_2 and E_3 . Wherever we are in the list, the iterator's *hasNext()* method will tell us whether there are more elements in front of our position. Once we have reached the end of the list, *hasNext()* tells us we can't go any further.

The iterator's *hasPrevious()* and *previous()* methods do exactly the same thing, only moving in the opposite direction. And if you think of how the iterators work, you can see that calling *next()* then calling *previous()* will get us the same element twice. (Remember that for the next lesson, when we learn how to move forward and backward through the list. We'll need that little fact to explain something.)

Now that we've seen how iterators work, let's look at how to code them. The first step is to import the class or interface we want to use. We're going to use *ListIterator*, so be sure to add an import statement at the top of your program for *java.util.ListIterator* (or *java.util.**). Then, since we're going to want to use this iterator several places in our class to move through the list, we're going to add another instance variable at the top of our class, like this:

```
private ListIterator<Player> lit;
```

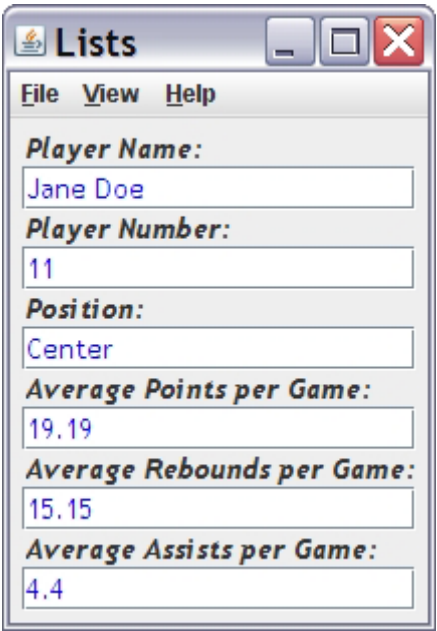
Then we're going to add code to *actionPerformed()* again to initiate and use the *Iterator* object:

```
lit = list.listIterator();  
  
if (lit.hasNext())  
{  
    Player p = lit.next();  
    getPlayer(p);  
}
```

The first line of this code calls the *ArrayList*'s *listIterator()* method to get a new iterator. The *if* statement in the next line checks to see if there are any elements in the list. If so, the third line uses the iterator's *next()* method to get the *Player* object, and the last line calls a helper method to put the player's information into the different fields in the window. I put this code into a helper method because we'll need it later on in our program. For now, here's the helper method, without comments since its purpose is pretty clear:

```
private void getPlayer(Player p)  
{  
    playerName.setText(p.getName());  
    playerNum.setText("" + p.getNum());  
    playerPosition.setText(p.getPosition());  
    playerAvgPts.setText("" + p.getAvgPoints());  
    playerAvgRbnds.setText("" + p.getAvgRebounds());  
    playerAvgAssists.setText("" + p.getAvgAssists());  
}
```

If you've copied and saved the player data from above into a text file, you should now be able to run your program and have it display the empty Player View. Then if you use the File > Open . . . menu option, an Open dialog should let you open the text file you saved. Your program should display information from the first player from the file in the window, like this:

A screenshot of a Java Swing window titled "Lists". The window has a standard Mac OS X-style title bar with minimize, maximize, and close buttons. Below the title bar is a menu bar with "File", "View", and "Help" menus. The main content area of the window contains several labeled text input fields. The labels are in a bold, italicized font: "Player Name:", "Player Number:", "Position:", "Average Points per Game:", "Average Rebounds per Game:", and "Average Assists per Game:". The input fields contain the following text: "Jane Doe", "11", "Center", "19.19", "15.15", and "4.4" respectively. The text in the input fields is blue, suggesting it might be a default or placeholder text.

First player's information