Chapter 4: Moving Forward and Backward

Moving Forward and Backward

The next task is to enable our user to move forward and backward through the players. This will actually turn out to be easier with a TreeMap than with an iterator. Let's deal with forward movement first, in the NextMenuItemListener inner class.

The first change in that method is to replace references to *list* with references to *map*. That means two changes in the following line:

```
if (list == null || list.size() == 0)
```

We'll change both occurrences of *list* to *map*, so that the line looks like this:

```
if (map == null || map.size() == 0)
```

Just as it checked for an empty or nonexistent list before, this line now checks for an empty or nonexistent map before proceeding.

Our next change is to delete the whole *if* statement that checks whether we were already moving forward. We don't need it anymore, so let's go ahead and delete these lines:

```
if (!isForward)

{

    lit.next();

    isForward = true;

}
```

We no longer need to check the iterator to find out if there are more players. We'll let the map tell us that. If there are no more players, it will return a null entry instead of a Player entry. So let's replace these lines:

```
if (lit.hasNext())

{

    Player p = lit.next();

    getPlayer(p);

}

else
```

with this one:

```
Map.Entry<Integer, Player> entry = map.higherEntry(Integer.parseInt(
    playerNum.getText()));
```

That's a long statement, isn't it? Let me break it down for you. The method that's going to get our next player is *higherEntry()*. That method gets the next entry in the map with a key *higher* than the key we give it. All we need to do is give it the currently displayed player's key, which is in the playerNum text box on the screen. So we get the text out of the text box using the getText() method, then convert it to an integer with the parseInt() method.

The catch is that higherEntry() does not return only a Player object; it returns a complete map entry with both the key and value. This means we have to create a Map.Entry object to hold it (that's the type of object the Java API tells us higherEntry() will return). The object we created in this line is named *entry*.

Now for the last little piece to this puzzle of moving forward: We still need to figure out if we reached the end of our players. If there is no key greater than the one already displayed, entry will be null, and we want to display a message dialog telling the user. So let's add this line in front of the block of code that displays the dialog:

```
if (entry == null)
```

If we did not reach the end of our players, we want to display the one we just got from the higherEntry() method. After the message dialog code, add one last bit of logic:

```
else
    getPlayer(entry.getValue());
```

Remember, since entry contains both a key and a value, we have to get the value (our Player object) out of it. The getValue() method does that for us.

That finishes up the forward movement logic. Moving backward requires almost identical changes in the PrevMenuItemListener inner class. The only difference is that instead of calling higherEntry(), we'll call *lowerEntry()*. Since the changes are so similar, I'll let you change your program on your own for practice. If you want to look at my changed version, you can see it at the following link:

New PrevMenuItemListener

Now you should be able to run your program, open the team file, select a player by number, and move forward and backward. Be sure to let me know in the Discussion Area if you have any trouble making it work.

Finding a Player

Don't you think it would be nice to have a way to go directly to another player if we know her number? We already have all the know-how to make that happen, so let's do it.

First, we need to tell the program we want to find a different player. The best way to do that is with a new menu option. Along with the menu option, we'll need a new listener to do its work. Let's add a new menu option to our View menu called *Find a Player* and a new listener named *FindMenuItemListener*. Since we've done all this before, I'll leave it to you. Go ahead and try it—and don't forget the findPlayer() method we created earlier. If you run into problems or aren't sure what it should look like, take a look at my code by clicking one of these links:

Find Menu Item

Find Menu Listener

And with that, believe it or not, we're done! The program opens our text file, stores its data in a map, provides forward and backward movement in the collection, and lets us find a particular player using her number. Let me know in the Discussion Area if you have problems making it work. If you would like to take a look at my finished version, I have included it behind the following link:

The Finished Product