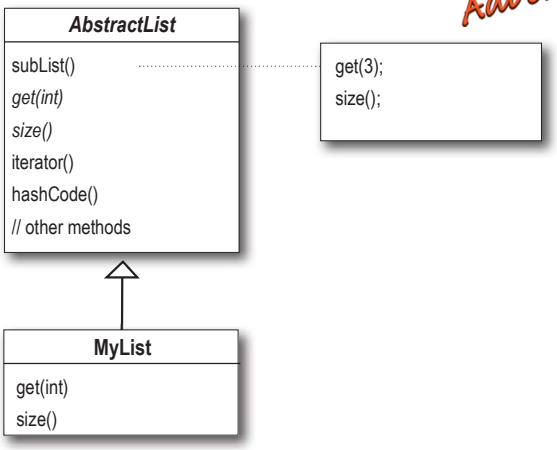
Custom Lists with AbstractList

Our final stop on the safari: AbstractList.

The list collections in Java, like ArrayList and LinkedList, extend the AbstractList class, which provides some of the basic implementations for list behavior. If you want to create your own custom list—say, a list that contains only Strings—you can do that by extending AbstractList so you get that basic list behavior for free.

AbstractList has a template method, subList(), that relies on two abstract methods, get() and size(). So when you extend AbstractList to create your own custom list, you'll provide implementations for these methods.

Here's an implementation of a custom list that contains only String objects, and uses arrays for the underlying implementation:




```
public class MyStringList extends AbstractList<String> {
        private String[] myList;
        MyStringList(String[] strings) {
                myList = strings;
        }
        public String get(int index) {
                return myList[index];
        }
        public int size() {
                return myList.length;
        }
        public String set(int index, String item) {
                String oldString = myList[index];
                myList[index] = item;
                return oldString;
        }
}
```

We create a custom list by extending AbstractList.

We must implement the methods get() and size(), which are both used by the template method subList().

We also implement a method set() so we can modify the list.

Test the subList() template method in your MyStringList implementation like this:

```
String[] ducks = { "Mallard Duck", "Redhead Duck", "Rubber Duck", "Decoy Duck"};
MyStringList ducksList = new MyStringList(ducks);
List ducksSubList = ducksList.subList(2, 3);
```

Create a sublist of one item starting at index 2...the Rubber Duck, of course.