001/\*  
002 \* Licensed to the Apache Software Foundation (ASF) under one or more  
003 \* contributor license agreements. See the NOTICE file distributed with  
004 \* this work for additional information regarding copyright ownership.  
005 \* The ASF licenses this file to You under the Apache License, Version 2.0  
006 \* (the "License"); you may not use this file except in compliance with  
007 \* the License. You may obtain a copy of the License at  
008 \*  
009 \* http://www.apache.org/licenses/LICENSE-2.0  
010 \*  
011 \* Unless required by applicable law or agreed to in writing, software  
012 \* distributed under the License is distributed on an "AS IS" BASIS,  
013 \* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
014 \* See the License for the specific language governing permissions and  
015 \* limitations under the License.  
016 \*/  
017package org.apache.commons.collections4.iterators;  
018  
019import java.util.NoSuchElementException;  
020  
021import org.apache.commons.collections4.ResettableListIterator;  
022  
023/\*\*  
024 \* Implements a {@link java.util.ListIterator} over an array of objects.  
025 \* <p>  
026 \* This iterator does not support {@link #add} or {@link #remove}, as the object array  
027 \* cannot be structurally modified. The {@link #set} method is supported however.  
028 \* <p>  
029 \* The iterator implements a {@link #reset} method, allowing the reset of the iterator  
030 \* back to the start if required.  
031 \*  
032 \* @see org.apache.commons.collections4.iterators.ObjectArrayIterator  
033 \* @see java.util.Iterator  
034 \* @see java.util.ListIterator  
035 \*  
036 \* @since 3.0  
037 \*/  
038public class ObjectArrayListIterator<E> extends ObjectArrayIterator<E>  
039 implements ResettableListIterator<E> {  
040  
041 /\*\*  
042 \* Holds the index of the last item returned by a call to <code>next()</code>  
043 \* or <code>previous()</code>. This is set to <code>-1</code> if neither method  
044 \* has yet been invoked. <code>lastItemIndex</code> is used to to implement the  
045 \* {@link #set} method.  
046 \*/  
047 private int lastItemIndex = -1;  
048  
049 //-------------------------------------------------------------------------  
050 /\*\*  
051 \* Constructs an ObjectArrayListIterator that will iterate over the values in the  
052 \* specified array.  
053 \*  
054 \* @param array the array to iterate over  
055 \* @throws NullPointerException if <code>array</code> is <code>null</code>  
056 \*/  
057 public ObjectArrayListIterator(final E... array) {  
058 super(array);  
059 }  
060  
061 /\*\*  
062 \* Constructs an ObjectArrayListIterator that will iterate over the values in the  
063 \* specified array from a specific start index.  
064 \*  
065 \* @param array the array to iterate over  
066 \* @param start the index to start iterating at  
067 \* @throws NullPointerException if <code>array</code> is <code>null</code>  
068 \* @throws IndexOutOfBoundsException if the start index is out of bounds  
069 \*/  
070 public ObjectArrayListIterator(final E[] array, final int start) {  
071 super(array, start);  
072 }  
073  
074 /\*\*  
075 \* Construct an ObjectArrayListIterator that will iterate over a range of values  
076 \* in the specified array.  
077 \*  
078 \* @param array the array to iterate over  
079 \* @param start the index to start iterating at  
080 \* @param end the index (exclusive) to finish iterating at  
081 \* @throws IndexOutOfBoundsException if the start or end index is out of bounds  
082 \* @throws IllegalArgumentException if end index is before the start  
083 \* @throws NullPointerException if <code>array</code> is <code>null</code>  
084 \*/  
085 public ObjectArrayListIterator(final E[] array, final int start, final int end) {  
086 super(array, start, end);  
087 }  
088  
089 // ListIterator interface  
090 //-------------------------------------------------------------------------  
091  
092 /\*\*  
093 \* Returns true if there are previous elements to return from the array.  
094 \*  
095 \* @return true if there is a previous element to return  
096 \*/  
097 @Override  
098 public boolean hasPrevious() {  
099 return this.index > getStartIndex();  
100 }  
101  
102 /\*\*  
103 \* Gets the previous element from the array.  
104 \*  
105 \* @return the previous element  
106 \* @throws NoSuchElementException if there is no previous element  
107 \*/  
108 @Override  
109 public E previous() {  
110 if (hasPrevious() == false) {  
111 throw new NoSuchElementException();  
112 }  
113 this.lastItemIndex = --this.index;  
114 return this.array[this.index];  
115 }  
116  
117 /\*\*  
118 \* Gets the next element from the array.  
119 \*  
120 \* @return the next element  
121 \* @throws NoSuchElementException if there is no next element  
122 \*/  
123 @Override  
124 public E next() {  
125 if (hasNext() == false) {  
126 throw new NoSuchElementException();  
127 }  
128 this.lastItemIndex = this.index;  
129 return this.array[this.index++];  
130 }  
131  
132 /\*\*  
133 \* Gets the next index to be retrieved.  
134 \*  
135 \* @return the index of the item to be retrieved next  
136 \*/  
137 @Override  
138 public int nextIndex() {  
139 return this.index - getStartIndex();  
140 }  
141  
142 /\*\*  
143 \* Gets the index of the item to be retrieved if {@link #previous()} is called.  
144 \*  
145 \* @return the index of the item to be retrieved next  
146 \*/  
147 @Override  
148 public int previousIndex() {  
149 return this.index - getStartIndex() - 1;  
150 }  
151  
152 /\*\*  
153 \* This iterator does not support modification of its backing array's size, and so will  
154 \* always throw an {@link UnsupportedOperationException} when this method is invoked.  
155 \*  
156 \* @param obj the object to add  
157 \* @throws UnsupportedOperationException always thrown.  
158 \*/  
159 @Override  
160 public void add(final E obj) {  
161 throw new UnsupportedOperationException("add() method is not supported");  
162 }  
163  
164 /\*\*  
165 \* Sets the element under the cursor.  
166 \* <p>  
167 \* This method sets the element that was returned by the last call  
168 \* to {@link #next()} of {@link #previous()}.  
169 \*  
170 \* <b>Note:</b> {@link java.util.ListIterator} implementations that support <code>add()</code>  
171 \* and <code>remove()</code> only allow <code>set()</code> to be called once per call  
172 \* to <code>next()</code> or <code>previous</code> (see the {@link java.util.ListIterator}  
173 \* javadoc for more details). Since this implementation does not support  
174 \* <code>add()</code> or <code>remove()</code>, <code>set()</code> may be  
175 \* called as often as desired.  
176 \*  
177 \* @param obj the object to set into the array  
178 \* @throws IllegalStateException if next() has not yet been called.  
179 \* @throws ClassCastException if the object type is unsuitable for the array  
180 \*/  
181 @Override  
182 public void set(final E obj) {  
183 if (this.lastItemIndex == -1) {  
184 throw new IllegalStateException("must call next() or previous() before a call to set()");  
185 }  
186  
187 this.array[this.lastItemIndex] = obj;  
188 }  
189  
190 /\*\*  
191 \* Resets the iterator back to the start index.  
192 \*/  
193 @Override  
194 public void reset() {  
195 super.reset();  
196 this.lastItemIndex = -1;  
197 }  
198  
199}