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.ResettableIterator;  
022  
023/\*\*  
024 \* An {@link java.util.Iterator Iterator} over an array of objects.  
025 \* <p>  
026 \* This iterator does not support {@link #remove}, as the object array cannot be  
027 \* structurally modified.  
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 \* @param <E> the type of elements returned by this iterator  
033 \* @since 3.0  
034 \*/  
035public class ObjectArrayIterator<E> implements ResettableIterator<E> {  
036  
037 /\*\* The array \*/  
038 final E[] array;  
039 /\*\* The start index to loop from \*/  
040 final int startIndex;  
041 /\*\* The end index to loop to \*/  
042 final int endIndex;  
043 /\*\* The current iterator index \*/  
044 int index = 0;  
045  
046 //-------------------------------------------------------------------------  
047 /\*\*  
048 \* Constructs an ObjectArrayIterator that will iterate over the values in the  
049 \* specified array.  
050 \*  
051 \* @param array the array to iterate over  
052 \* @throws NullPointerException if <code>array</code> is <code>null</code>  
053 \*/  
054 public ObjectArrayIterator(final E... array) {  
055 this(array, 0, array.length);  
056 }  
057  
058 /\*\*  
059 \* Constructs an ObjectArrayIterator that will iterate over the values in the  
060 \* specified array from a specific start index.  
061 \*  
062 \* @param array the array to iterate over  
063 \* @param start the index to start iterating at  
064 \* @throws NullPointerException if <code>array</code> is <code>null</code>  
065 \* @throws IndexOutOfBoundsException if the start index is out of bounds  
066 \*/  
067 public ObjectArrayIterator(final E array[], final int start) {  
068 this(array, start, array.length);  
069 }  
070  
071 /\*\*  
072 \* Construct an ObjectArrayIterator that will iterate over a range of values  
073 \* in the specified array.  
074 \*  
075 \* @param array the array to iterate over  
076 \* @param start the index to start iterating at  
077 \* @param end the index (exclusive) to finish iterating at  
078 \* @throws IndexOutOfBoundsException if the start or end index is out of bounds  
079 \* @throws IllegalArgumentException if end index is before the start  
080 \* @throws NullPointerException if <code>array</code> is <code>null</code>  
081 \*/  
082 public ObjectArrayIterator(final E array[], final int start, final int end) {  
083 super();  
084 if (start < 0) {  
085 throw new ArrayIndexOutOfBoundsException("Start index must not be less than zero");  
086 }  
087 if (end > array.length) {  
088 throw new ArrayIndexOutOfBoundsException("End index must not be greater than the array length");  
089 }  
090 if (start > array.length) {  
091 throw new ArrayIndexOutOfBoundsException("Start index must not be greater than the array length");  
092 }  
093 if (end < start) {  
094 throw new IllegalArgumentException("End index must not be less than start index");  
095 }  
096 this.array = array;  
097 this.startIndex = start;  
098 this.endIndex = end;  
099 this.index = start;  
100 }  
101  
102 // Iterator interface  
103 //-------------------------------------------------------------------------  
104  
105 /\*\*  
106 \* Returns true if there are more elements to return from the array.  
107 \*  
108 \* @return true if there is a next element to return  
109 \*/  
110 @Override  
111 public boolean hasNext() {  
112 return this.index < this.endIndex;  
113 }  
114  
115 /\*\*  
116 \* Returns the next element in the array.  
117 \*  
118 \* @return the next element in the array  
119 \* @throws NoSuchElementException if all the elements in the array  
120 \* have already been returned  
121 \*/  
122 @Override  
123 public E next() {  
124 if (hasNext() == false) {  
125 throw new NoSuchElementException();  
126 }  
127 return this.array[this.index++];  
128 }  
129  
130 /\*\*  
131 \* Throws {@link UnsupportedOperationException}.  
132 \*  
133 \* @throws UnsupportedOperationException always  
134 \*/  
135 @Override  
136 public void remove() {  
137 throw new UnsupportedOperationException("remove() method is not supported for an ObjectArrayIterator");  
138 }  
139  
140 // Properties  
141 //-------------------------------------------------------------------------  
142  
143 /\*\*  
144 \* Gets the array that this iterator is iterating over.  
145 \*  
146 \* @return the array this iterator iterates over  
147 \*/  
148 public E[] getArray() {  
149 return this.array;  
150 }  
151  
152 /\*\*  
153 \* Gets the start index to loop from.  
154 \*  
155 \* @return the start index  
156 \*/  
157 public int getStartIndex() {  
158 return this.startIndex;  
159 }  
160  
161 /\*\*  
162 \* Gets the end index to loop to.  
163 \*  
164 \* @return the end index  
165 \*/  
166 public int getEndIndex() {  
167 return this.endIndex;  
168 }  
169  
170 /\*\*  
171 \* Resets the iterator back to the start index.  
172 \*/  
173 @Override  
174 public void reset() {  
175 this.index = this.startIndex;  
176 }  
177  
178}