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.lang.reflect.Array;  
020import java.util.NoSuchElementException;  
021  
022import org.apache.commons.collections4.ResettableIterator;  
023  
024/\*\*  
025 \* Implements an {@link java.util.Iterator Iterator} over any array.  
026 \* <p>  
027 \* The array can be either an array of object or of primitives. If you know  
028 \* that you have an object array, the  
029 \* {@link org.apache.commons.collections4.iterators.ObjectArrayIterator ObjectArrayIterator}  
030 \* class is a better choice, as it will perform better.  
031 \* <p>  
032 \* The iterator implements a {@link #reset} method, allowing the reset of  
033 \* the iterator back to the start if required.  
034 \*  
035 \* @param <E> the type of elements returned by this iterator  
036 \* @since 1.0  
037 \*/  
038public class ArrayIterator<E> implements ResettableIterator<E> {  
039  
040 /\*\* The array to iterate over \*/  
041 final Object array;  
042 /\*\* The start index to loop from \*/  
043 final int startIndex;  
044 /\*\* The end index to loop to \*/  
045 final int endIndex;  
046 /\*\* The current iterator index \*/  
047 int index = 0;  
048  
049 // Constructors  
050 // ----------------------------------------------------------------------  
051 /\*\*  
052 \* Constructs an ArrayIterator that will iterate over the values in the  
053 \* specified array.  
054 \*  
055 \* @param array the array to iterate over.  
056 \* @throws IllegalArgumentException if <code>array</code> is not an array.  
057 \* @throws NullPointerException if <code>array</code> is <code>null</code>  
058 \*/  
059 public ArrayIterator(final Object array) {  
060 this(array, 0);  
061 }  
062  
063 /\*\*  
064 \* Constructs an ArrayIterator that will iterate over the values in the  
065 \* specified array from a specific start index.  
066 \*  
067 \* @param array the array to iterate over.  
068 \* @param startIndex the index to start iterating at.  
069 \* @throws IllegalArgumentException if <code>array</code> is not an array.  
070 \* @throws NullPointerException if <code>array</code> is <code>null</code>  
071 \* @throws IndexOutOfBoundsException if the index is invalid  
072 \*/  
073 public ArrayIterator(final Object array, final int startIndex) {  
074 this(array, startIndex, Array.getLength(array));  
075 }  
076  
077 /\*\*  
078 \* Construct an ArrayIterator that will iterate over a range of values  
079 \* in the specified array.  
080 \*  
081 \* @param array the array to iterate over.  
082 \* @param startIndex the index to start iterating at.  
083 \* @param endIndex the index to finish iterating at.  
084 \* @throws IllegalArgumentException if <code>array</code> is not an array.  
085 \* @throws NullPointerException if <code>array</code> is <code>null</code>  
086 \* @throws IndexOutOfBoundsException if either index is invalid  
087 \*/  
088 public ArrayIterator(final Object array, final int startIndex, final int endIndex) {  
089 super();  
090  
091 this.array = array;  
092 this.startIndex = startIndex;  
093 this.endIndex = endIndex;  
094 this.index = startIndex;  
095  
096 final int len = Array.getLength(array);  
097 checkBound(startIndex, len, "start");  
098 checkBound(endIndex, len, "end");  
099 if (endIndex < startIndex) {  
100 throw new IllegalArgumentException("End index must not be less than start index.");  
101 }  
102 }  
103  
104 /\*\*  
105 \* Checks whether the index is valid or not.  
106 \*  
107 \* @param bound the index to check  
108 \* @param len the length of the array  
109 \* @param type the index type (for error messages)  
110 \* @throws IndexOutOfBoundsException if the index is invalid  
111 \*/  
112 protected void checkBound(final int bound, final int len, final String type ) {  
113 if (bound > len) {  
114 throw new ArrayIndexOutOfBoundsException(  
115 "Attempt to make an ArrayIterator that " + type +  
116 "s beyond the end of the array. "  
117 );  
118 }  
119 if (bound < 0) {  
120 throw new ArrayIndexOutOfBoundsException(  
121 "Attempt to make an ArrayIterator that " + type +  
122 "s before the start of the array. "  
123 );  
124 }  
125 }  
126  
127 // Iterator interface  
128 //-----------------------------------------------------------------------  
129 /\*\*  
130 \* Returns true if there are more elements to return from the array.  
131 \*  
132 \* @return true if there is a next element to return  
133 \*/  
134 @Override  
135 public boolean hasNext() {  
136 return index < endIndex;  
137 }  
138  
139 /\*\*  
140 \* Returns the next element in the array.  
141 \*  
142 \* @return the next element in the array  
143 \* @throws NoSuchElementException if all the elements in the array  
144 \* have already been returned  
145 \*/  
146 @Override  
147 @SuppressWarnings("unchecked")  
148 public E next() {  
149 if (hasNext() == false) {  
150 throw new NoSuchElementException();  
151 }  
152 return (E) Array.get(array, index++);  
153 }  
154  
155 /\*\*  
156 \* Throws {@link UnsupportedOperationException}.  
157 \*  
158 \* @throws UnsupportedOperationException always  
159 \*/  
160 @Override  
161 public void remove() {  
162 throw new UnsupportedOperationException("remove() method is not supported");  
163 }  
164  
165 // Properties  
166 //-----------------------------------------------------------------------  
167 /\*\*  
168 \* Gets the array that this iterator is iterating over.  
169 \*  
170 \* @return the array this iterator iterates over.  
171 \*/  
172 public Object getArray() {  
173 return array;  
174 }  
175  
176 /\*\*  
177 \* Gets the start index to loop from.  
178 \*  
179 \* @return the start index  
180 \* @since 4.0  
181 \*/  
182 public int getStartIndex() {  
183 return this.startIndex;  
184 }  
185  
186 /\*\*  
187 \* Gets the end index to loop to.  
188 \*  
189 \* @return the end index  
190 \* @since 4.0  
191 \*/  
192 public int getEndIndex() {  
193 return this.endIndex;  
194 }  
195  
196 /\*\*  
197 \* Resets the iterator back to the start index.  
198 \*/  
199 @Override  
200 public void reset() {  
201 this.index = this.startIndex;  
202 }  
203  
204}