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.queue;  
018  
019import java.util.Queue;  
020  
021import org.apache.commons.collections4.collection.AbstractCollectionDecorator;  
022  
023/\*\*  
024 \* Decorates another {@link Queue} to provide additional behaviour.  
025 \* <p>  
026 \* Methods are forwarded directly to the decorated queue.  
027 \* </p>  
028 \* <p>  
029 \* This implementation does not forward the hashCode and equals methods through  
030 \* to the backing object, but relies on Object's implementation. This is  
031 \* necessary as some Queue implementations, e.g. LinkedList, have custom a  
032 \* equals implementation for which symmetry can not be preserved.  
033 \* See class javadoc of AbstractCollectionDecorator for more information.  
034 \* </p>  
035 \*  
036 \* @param <E> the type of the elements in the queue  
037 \* @since 4.0  
038 \*/  
039public abstract class AbstractQueueDecorator<E> extends AbstractCollectionDecorator<E>  
040 implements Queue<E> {  
041  
042 /\*\* Serialization version \*/  
043 private static final long serialVersionUID = -2629815475789577029L;  
044  
045 /\*\*  
046 \* Constructor only used in deserialization, do not use otherwise.  
047 \*/  
048 protected AbstractQueueDecorator() {  
049 super();  
050 }  
051  
052 /\*\*  
053 \* Constructor that wraps (not copies).  
054 \*  
055 \* @param queue the queue to decorate, must not be null  
056 \* @throws NullPointerException if queue is null  
057 \*/  
058 protected AbstractQueueDecorator(final Queue<E> queue) {  
059 super(queue);  
060 }  
061  
062 /\*\*  
063 \* Gets the queue being decorated.  
064 \*  
065 \* @return the decorated queue  
066 \*/  
067 @Override  
068 protected Queue<E> decorated() {  
069 return (Queue<E>) super.decorated();  
070 }  
071  
072 //-----------------------------------------------------------------------  
073  
074 @Override  
075 public boolean offer(final E obj) {  
076 return decorated().offer(obj);  
077 }  
078  
079 @Override  
080 public E poll() {  
081 return decorated().poll();  
082 }  
083  
084 @Override  
085 public E peek() {  
086 return decorated().peek();  
087 }  
088  
089 @Override  
090 public E element() {  
091 return decorated().element();  
092 }  
093  
094 @Override  
095 public E remove() {  
096 return decorated().remove();  
097 }  
098  
099}