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.set;  
018  
019import java.io.IOException;  
020import java.io.ObjectInputStream;  
021import java.io.ObjectOutputStream;  
022import java.util.Collection;  
023import java.util.Set;  
024  
025/\*\*  
026 \* Serializable subclass of AbstractSetDecorator.  
027 \*  
028 \* @param <E> the type of the elements in this set  
029 \* @since 3.1  
030 \*/  
031public abstract class AbstractSerializableSetDecorator<E>  
032 extends AbstractSetDecorator<E> {  
033  
034 /\*\* Serialization version \*/  
035 private static final long serialVersionUID = 1229469966212206107L;  
036  
037 /\*\*  
038 \* Constructor.  
039 \*  
040 \* @param set the list to decorate, must not be null  
041 \* @throws NullPointerException if set is null  
042 \*/  
043 protected AbstractSerializableSetDecorator(final Set<E> set) {  
044 super(set);  
045 }  
046  
047 //-----------------------------------------------------------------------  
048 /\*\*  
049 \* Write the set out using a custom routine.  
050 \*  
051 \* @param out the output stream  
052 \* @throws IOException if an error occurs while writing to the stream  
053 \*/  
054 private void writeObject(final ObjectOutputStream out) throws IOException {  
055 out.defaultWriteObject();  
056 out.writeObject(decorated());  
057 }  
058  
059 /\*\*  
060 \* Read the set in using a custom routine.  
061 \*  
062 \* @param in the input stream  
063 \* @throws IOException if an error occurs while reading from the stream  
064 \* @throws ClassNotFoundException if an object read from the stream can not be loaded  
065 \*/  
066 @SuppressWarnings("unchecked")  
067 private void readObject(final ObjectInputStream in) throws IOException, ClassNotFoundException {  
068 in.defaultReadObject();  
069 setCollection((Collection<E>) in.readObject());  
070 }  
071  
072}