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.trie;  
018  
019import java.util.Map;  
020  
021import org.apache.commons.collections4.trie.analyzer.StringKeyAnalyzer;  
022  
023/\*\*  
024 \* Implementation of a PATRICIA Trie (Practical Algorithm to Retrieve Information  
025 \* Coded in Alphanumeric).  
026 \* <p>  
027 \* A PATRICIA {@link org.apache.commons.collections4.Trie} is a compressed  
028 \* {@link org.apache.commons.collections4.Trie}. Instead of storing  
029 \* all data at the edges of the {@link org.apache.commons.collections4.Trie}  
030 \* (and having empty internal nodes), PATRICIA stores data in every node.  
031 \* This allows for very efficient traversal, insert, delete, predecessor,  
032 \* successor, prefix, range, and {@link #select(Object)}  
033 \* operations. All operations are performed at worst in O(K) time, where K  
034 \* is the number of bits in the largest item in the tree. In practice,  
035 \* operations actually take O(A(K)) time, where A(K) is the average number of  
036 \* bits of all items in the tree.  
037 \* </p>  
038 \* <p>  
039 \* Most importantly, PATRICIA requires very few comparisons to keys while  
040 \* doing any operation. While performing a lookup, each comparison (at most  
041 \* K of them, described above) will perform a single bit comparison against  
042 \* the given key, instead of comparing the entire key to another key.  
043 \* </p>  
044 \* <p>  
045 \* The {@link org.apache.commons.collections4.Trie} can return operations in  
046 \* lexicographical order using the 'prefixMap', 'submap', or 'iterator' methods.  
047 \* The {@link org.apache.commons.collections4.Trie} can also  
048 \* scan for items that are 'bitwise' (using an XOR metric) by the 'select' method.  
049 \* Bitwise closeness is determined by the {@link KeyAnalyzer} returning true or  
050 \* false for a bit being set or not in a given key.  
051 \* </p>  
052 \* <p>  
053 \* This PATRICIA {@link org.apache.commons.collections4.Trie} supports both variable  
054 \* length & fixed length keys. Some methods, such as {@link #prefixMap(Object)}  
055 \* are suited only to variable length keys.  
056 \* </p>  
057 \*  
058 \* @param <E> the type of the values in this map  
059 \*  
060 \* @see <a href="http://en.wikipedia.org/wiki/Radix\_tree">Radix Tree</a>  
061 \* @see <a href="http://www.csse.monash.edu.au/~lloyd/tildeAlgDS/Tree/PATRICIA">PATRICIA</a>  
062 \* @see <a href="http://www.imperialviolet.org/binary/critbit.pdf">Crit-Bit Tree</a>  
063 \* @since 4.0  
064 \*/  
065public class PatriciaTrie<E> extends AbstractPatriciaTrie<String, E> {  
066  
067 private static final long serialVersionUID = 4446367780901817838L;  
068  
069 public PatriciaTrie() {  
070 super(new StringKeyAnalyzer());  
071 }  
072  
073 public PatriciaTrie(final Map<? extends String, ? extends E> m) {  
074 super(new StringKeyAnalyzer(), m);  
075 }  
076  
077}