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 \*/  
017  
018package org.apache.commons.beanutils.locale.converters;  
019  
020import org.apache.commons.beanutils.ConversionException;  
021  
022import java.util.Locale;  
023import java.text.ParseException;  
024  
025  
026/\*\*  
027 \* <p>Standard {@link org.apache.commons.beanutils.locale.LocaleConverter}  
028 \* implementation that converts an incoming  
029 \* locale-sensitive String into a <code>java.math.BigDecimal</code> object,  
030 \* optionally using a default value or throwing a  
031 \* {@link org.apache.commons.beanutils.ConversionException}  
032 \* if a conversion error occurs.</p>  
033 \*  
034 \* @version $Id$  
035 \*/  
036  
037public class FloatLocaleConverter extends DecimalLocaleConverter {  
038  
039  
040 // ----------------------------------------------------------- Constructors  
041  
042 /\*\*  
043 \* Create a {@link org.apache.commons.beanutils.locale.LocaleConverter}  
044 \* that will throw a {@link org.apache.commons.beanutils.ConversionException}  
045 \* if a conversion error occurs. The locale is the default locale for  
046 \* this instance of the Java Virtual Machine and an unlocalized pattern is used  
047 \* for the convertion.  
048 \*  
049 \*/  
050 public FloatLocaleConverter() {  
051  
052 this(false);  
053 }  
054  
055 /\*\*  
056 \* Create a {@link org.apache.commons.beanutils.locale.LocaleConverter}  
057 \* that will throw a {@link org.apache.commons.beanutils.ConversionException}  
058 \* if a conversion error occurs. The locale is the default locale for  
059 \* this instance of the Java Virtual Machine.  
060 \*  
061 \* @param locPattern Indicate whether the pattern is localized or not  
062 \*/  
063 public FloatLocaleConverter(final boolean locPattern) {  
064  
065 this(Locale.getDefault(), locPattern);  
066 }  
067  
068 /\*\*  
069 \* Create a {@link org.apache.commons.beanutils.locale.LocaleConverter}  
070 \* that will throw a {@link org.apache.commons.beanutils.ConversionException}  
071 \* if a conversion error occurs. An unlocalized pattern is used for the convertion.  
072 \*  
073 \* @param locale The locale  
074 \*/  
075 public FloatLocaleConverter(final Locale locale) {  
076  
077 this(locale, false);  
078 }  
079  
080 /\*\*  
081 \* Create a {@link org.apache.commons.beanutils.locale.LocaleConverter}  
082 \* that will throw a {@link org.apache.commons.beanutils.ConversionException}  
083 \* if a conversion error occurs.  
084 \*  
085 \* @param locale The locale  
086 \* @param locPattern Indicate whether the pattern is localized or not  
087 \*/  
088 public FloatLocaleConverter(final Locale locale, final boolean locPattern) {  
089  
090 this(locale, (String) null, locPattern);  
091 }  
092  
093 /\*\*  
094 \* Create a {@link org.apache.commons.beanutils.locale.LocaleConverter}  
095 \* that will throw a {@link org.apache.commons.beanutils.ConversionException}  
096 \* if a conversion error occurs. An unlocalized pattern is used for the convertion.  
097 \*  
098 \* @param locale The locale  
099 \* @param pattern The convertion pattern  
100 \*/  
101 public FloatLocaleConverter(final Locale locale, final String pattern) {  
102  
103 this(locale, pattern, false);  
104 }  
105  
106 /\*\*  
107 \* Create a {@link org.apache.commons.beanutils.locale.LocaleConverter}  
108 \* that will throw a {@link org.apache.commons.beanutils.ConversionException}  
109 \* if a conversion error occurs.  
110 \*  
111 \* @param locale The locale  
112 \* @param pattern The convertion pattern  
113 \* @param locPattern Indicate whether the pattern is localized or not  
114 \*/  
115 public FloatLocaleConverter(final Locale locale, final String pattern, final boolean locPattern) {  
116  
117 super(locale, pattern, locPattern);  
118 }  
119  
120 /\*\*  
121 \* Create a {@link org.apache.commons.beanutils.locale.LocaleConverter}  
122 \* that will return the specified default value  
123 \* if a conversion error occurs. The locale is the default locale for  
124 \* this instance of the Java Virtual Machine and an unlocalized pattern is used  
125 \* for the convertion.  
126 \*  
127 \* @param defaultValue The default value to be returned  
128 \*/  
129 public FloatLocaleConverter(final Object defaultValue) {  
130  
131 this(defaultValue, false);  
132 }  
133  
134 /\*\*  
135 \* Create a {@link org.apache.commons.beanutils.locale.LocaleConverter}  
136 \* that will return the specified default value  
137 \* if a conversion error occurs. The locale is the default locale for  
138 \* this instance of the Java Virtual Machine.  
139 \*  
140 \* @param defaultValue The default value to be returned  
141 \* @param locPattern Indicate whether the pattern is localized or not  
142 \*/  
143 public FloatLocaleConverter(final Object defaultValue, final boolean locPattern) {  
144  
145 this(defaultValue, Locale.getDefault(), locPattern);  
146 }  
147  
148 /\*\*  
149 \* Create a {@link org.apache.commons.beanutils.locale.LocaleConverter}  
150 \* that will return the specified default value  
151 \* if a conversion error occurs. An unlocalized pattern is used for the convertion.  
152 \*  
153 \* @param defaultValue The default value to be returned  
154 \* @param locale The locale  
155 \*/  
156 public FloatLocaleConverter(final Object defaultValue, final Locale locale) {  
157  
158 this(defaultValue, locale, false);  
159 }  
160  
161 /\*\*  
162 \* Create a {@link org.apache.commons.beanutils.locale.LocaleConverter}  
163 \* that will return the specified default value  
164 \* if a conversion error occurs.  
165 \*  
166 \* @param defaultValue The default value to be returned  
167 \* @param locale The locale  
168 \* @param locPattern Indicate whether the pattern is localized or not  
169 \*/  
170 public FloatLocaleConverter(final Object defaultValue, final Locale locale, final boolean locPattern) {  
171  
172 this(defaultValue, locale, null, locPattern);  
173 }  
174  
175 /\*\*  
176 \* Create a {@link org.apache.commons.beanutils.locale.LocaleConverter}  
177 \* that will return the specified default value  
178 \* if a conversion error occurs. An unlocalized pattern is used for the convertion.  
179 \*  
180 \* @param defaultValue The default value to be returned  
181 \* @param locale The locale  
182 \* @param pattern The convertion pattern  
183 \*/  
184 public FloatLocaleConverter(final Object defaultValue, final Locale locale, final String pattern) {  
185  
186 this(defaultValue, locale, pattern, false);  
187 }  
188  
189 /\*\*  
190 \* Create a {@link org.apache.commons.beanutils.locale.LocaleConverter}  
191 \* that will return the specified default value  
192 \* if a conversion error occurs.  
193 \*  
194 \* @param defaultValue The default value to be returned  
195 \* @param locale The locale  
196 \* @param pattern The convertion pattern  
197 \* @param locPattern Indicate whether the pattern is localized or not  
198 \*/  
199 public FloatLocaleConverter(final Object defaultValue, final Locale locale, final String pattern, final boolean locPattern) {  
200  
201 super(defaultValue, locale, pattern, locPattern);  
202 }  
203  
204 /\*\*  
205 \* Convert the specified locale-sensitive input object into an output object of the  
206 \* specified type. This method will return Float value or throw exception if value  
207 \* can not be stored in the Float.  
208 \*  
209 \* @param value The input object to be converted  
210 \* @param pattern The pattern is used for the convertion  
211 \* @return The converted value  
212 \*  
213 \* @throws ConversionException if conversion cannot be performed  
214 \* successfully  
215 \* @throws ParseException if an error occurs parsing a String to a Number  
216 \*/  
217 @Override  
218protected Object parse(final Object value, final String pattern) throws ParseException {  
219 final Number parsed = (Number) super.parse(value, pattern);  
220 final double doubleValue = parsed.doubleValue();  
221 final double posDouble = (doubleValue >= 0) ? doubleValue : (doubleValue \* -1);  
222 if (posDouble != 0 && (posDouble < Float.MIN\_VALUE || posDouble > Float.MAX\_VALUE)) {  
223 throw new ConversionException("Supplied number is not of type Float: "+parsed);  
224 }  
225 return new Float(parsed.floatValue()); // unlike superclass it returns Float type  
226 }  
227}