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 java.util.Locale;  
021import java.math.BigInteger;  
022import java.text.ParseException;  
023import org.apache.commons.beanutils.ConversionException;  
024  
025/\*\*  
026 \* <p>Standard {@link org.apache.commons.beanutils.locale.LocaleConverter}  
027 \* implementation that converts an incoming  
028 \* locale-sensitive String into a <code>java.math.BigInteger</code> object,  
029 \* optionally using a default value or throwing a  
030 \* {@link org.apache.commons.beanutils.ConversionException}  
031 \* if a conversion error occurs.</p>  
032 \*  
033 \* @version $Id$  
034 \*/  
035  
036public class BigIntegerLocaleConverter extends DecimalLocaleConverter {  
037  
038  
039 // ----------------------------------------------------------- Constructors  
040  
041 /\*\*  
042 \* Create a {@link org.apache.commons.beanutils.locale.LocaleConverter}  
043 \* that will throw a {@link org.apache.commons.beanutils.ConversionException}  
044 \* if a conversion error occurs. The locale is the default locale for  
045 \* this instance of the Java Virtual Machine and an unlocalized pattern is used  
046 \* for the convertion.  
047 \*  
048 \*/  
049 public BigIntegerLocaleConverter() {  
050  
051 this(false);  
052 }  
053  
054 /\*\*  
055 \* Create a {@link org.apache.commons.beanutils.locale.LocaleConverter}  
056 \* that will throw a {@link org.apache.commons.beanutils.ConversionException}  
057 \* if a conversion error occurs. The locale is the default locale for  
058 \* this instance of the Java Virtual Machine.  
059 \*  
060 \* @param locPattern Indicate whether the pattern is localized or not  
061 \*/  
062 public BigIntegerLocaleConverter(final boolean locPattern) {  
063  
064 this(Locale.getDefault(), locPattern);  
065 }  
066  
067 /\*\*  
068 \* Create a {@link org.apache.commons.beanutils.locale.LocaleConverter}  
069 \* that will throw a {@link org.apache.commons.beanutils.ConversionException}  
070 \* if a conversion error occurs. An unlocalized pattern is used for the convertion.  
071 \*  
072 \* @param locale The locale  
073 \*/  
074 public BigIntegerLocaleConverter(final Locale locale) {  
075  
076 this(locale, false);  
077 }  
078  
079 /\*\*  
080 \* Create a {@link org.apache.commons.beanutils.locale.LocaleConverter}  
081 \* that will throw a {@link org.apache.commons.beanutils.ConversionException}  
082 \* if a conversion error occurs.  
083 \*  
084 \* @param locale The locale  
085 \* @param locPattern Indicate whether the pattern is localized or not  
086 \*/  
087 public BigIntegerLocaleConverter(final Locale locale, final boolean locPattern) {  
088  
089 this(locale, (String) null, locPattern);  
090 }  
091  
092 /\*\*  
093 \* Create a {@link org.apache.commons.beanutils.locale.LocaleConverter}  
094 \* that will throw a {@link org.apache.commons.beanutils.ConversionException}  
095 \* if a conversion error occurs. An unlocalized pattern is used for the convertion.  
096 \*  
097 \* @param locale The locale  
098 \* @param pattern The convertion pattern  
099 \*/  
100 public BigIntegerLocaleConverter(final Locale locale, final String pattern) {  
101  
102 this(locale, pattern, false);  
103 }  
104  
105 /\*\*  
106 \* Create a {@link org.apache.commons.beanutils.locale.LocaleConverter}  
107 \* that will throw a {@link org.apache.commons.beanutils.ConversionException}  
108 \* if a conversion error occurs.  
109 \*  
110 \* @param locale The locale  
111 \* @param pattern The convertion pattern  
112 \* @param locPattern Indicate whether the pattern is localized or not  
113 \*/  
114 public BigIntegerLocaleConverter(final Locale locale, final String pattern, final boolean locPattern) {  
115  
116 super(locale, pattern, locPattern);  
117 }  
118  
119 /\*\*  
120 \* Create a {@link org.apache.commons.beanutils.locale.LocaleConverter}  
121 \* that will return the specified default value  
122 \* if a conversion error occurs. The locale is the default locale for  
123 \* this instance of the Java Virtual Machine and an unlocalized pattern is used  
124 \* for the convertion.  
125 \*  
126 \* @param defaultValue The default value to be returned  
127 \*/  
128 public BigIntegerLocaleConverter(final Object defaultValue) {  
129  
130 this(defaultValue, false);  
131 }  
132  
133 /\*\*  
134 \* Create a {@link org.apache.commons.beanutils.locale.LocaleConverter}  
135 \* that will return the specified default value  
136 \* if a conversion error occurs. The locale is the default locale for  
137 \* this instance of the Java Virtual Machine.  
138 \*  
139 \* @param defaultValue The default value to be returned  
140 \* @param locPattern Indicate whether the pattern is localized or not  
141 \*/  
142 public BigIntegerLocaleConverter(final Object defaultValue, final boolean locPattern) {  
143  
144 this(defaultValue, Locale.getDefault(), locPattern);  
145 }  
146  
147 /\*\*  
148 \* Create a {@link org.apache.commons.beanutils.locale.LocaleConverter}  
149 \* that will return the specified default value  
150 \* if a conversion error occurs. An unlocalized pattern is used for the convertion.  
151 \*  
152 \* @param defaultValue The default value to be returned  
153 \* @param locale The locale  
154 \*/  
155 public BigIntegerLocaleConverter(final Object defaultValue, final Locale locale) {  
156  
157 this(defaultValue, locale, false);  
158 }  
159  
160 /\*\*  
161 \* Create a {@link org.apache.commons.beanutils.locale.LocaleConverter}  
162 \* that will return the specified default value  
163 \* if a conversion error occurs.  
164 \*  
165 \* @param defaultValue The default value to be returned  
166 \* @param locale The locale  
167 \* @param locPattern Indicate whether the pattern is localized or not  
168 \*/  
169 public BigIntegerLocaleConverter(final Object defaultValue, final Locale locale, final boolean locPattern) {  
170  
171 this(defaultValue, locale, null, locPattern);  
172 }  
173  
174 /\*\*  
175 \* Create a {@link org.apache.commons.beanutils.locale.LocaleConverter}  
176 \* that will return the specified default value  
177 \* if a conversion error occurs. An unlocalized pattern is used for the convertion.  
178 \*  
179 \* @param defaultValue The default value to be returned  
180 \* @param locale The locale  
181 \* @param pattern The convertion pattern  
182 \*/  
183 public BigIntegerLocaleConverter(final Object defaultValue, final Locale locale, final String pattern) {  
184  
185 this(defaultValue, locale, pattern, false);  
186 }  
187  
188 /\*\*  
189 \* Create a {@link org.apache.commons.beanutils.locale.LocaleConverter}  
190 \* that will return the specified default value  
191 \* if a conversion error occurs.  
192 \*  
193 \* @param defaultValue The default value to be returned  
194 \* @param locale The locale  
195 \* @param pattern The convertion pattern  
196 \* @param locPattern Indicate whether the pattern is localized or not  
197 \*/  
198 public BigIntegerLocaleConverter(final Object defaultValue, final Locale locale, final String pattern, final boolean locPattern) {  
199  
200 super(defaultValue, locale, pattern, locPattern);  
201 }  
202  
203 /\*\*  
204 \* Convert the specified locale-sensitive input object into an output object of  
205 \* BigInteger type.  
206 \*  
207 \* @param value The input object to be converted  
208 \* @param pattern The pattern is used for the convertion  
209 \* @return The converted value  
210 \*  
211 \* @throws ConversionException if conversion cannot be performed  
212 \* successfully  
213 \* @throws ParseException if an error occurs parsing a String to a Number  
214 \* @since 1.8.0  
215 \*/  
216 @Override  
217 protected Object parse(final Object value, final String pattern) throws ParseException {  
218  
219 final Object result = super.parse(value, pattern);  
220  
221 if (result == null || result instanceof BigInteger) {  
222 return result;  
223 }  
224  
225 if (result instanceof Number) {  
226 return BigInteger.valueOf(((Number)result).longValue());  
227 }  
228  
229 try {  
230 return new BigInteger(result.toString());  
231 }  
232 catch (final NumberFormatException ex) {  
233 throw new ConversionException("Suplied number is not of type BigInteger: " + result);  
234 }  
235  
236 }  
237  
238}