package com.twitter.tweetypie.tweettext

import java.text.Normalizer

object TweetText {

/\*\* The original maximum tweet length, taking into account normalization \*/

private[tweetypie] val OriginalMaxDisplayLength = 140

/\*\* Maximum number of visible code points allowed in a tweet when tweet length is counted by code

\* points, taking into account normalization. See also [[MaxVisibleWeightedEmojiLength]].

\*/

private[tweetypie] val MaxVisibleWeightedLength = 280

/\*\* Maximum number of visible code points allowed in a tweet when tweet length is counted by

\* emoji, taking into account normalization. See also [[MaxVisibleWeightedLength]].

\* 140 is the max number of Emojis, visible, fully-weighted per Twitter's cramming rules

\* 10 is the max number of Code Points per Emoji

\*/

private[tweetypie] val MaxVisibleWeightedEmojiLength = 140 \* 10

/\*\* Maximum number of bytes when truncating tweet text for a retweet. Originally was the

\* max UTF-8 length when tweets were at most 140 characters.

\* See also [[OriginalMaxDisplayLength]].

\*/

private[tweetypie] val OriginalMaxUtf8Length = 600

/\*\* Maximum number of bytes for tweet text using utf-8 encoding.

\*/

private[tweetypie] val MaxUtf8Length = 5708

/\*\* Maximum number of mentions allowed in tweet text. This is enforced at tweet creation time \*/

private[tweetypie] val MaxMentions = 50

/\*\* Maximum number of urls allowed in tweet text. This is enforced at tweet creation time \*/

private[tweetypie] val MaxUrls = 10

/\*\* Maximum number of hashtags allowed in tweet text. This is enforced at tweet creation time \*/

private[tweetypie] val MaxHashtags = 50

/\*\* Maximum number of cashtags allowed in tweet text. This is enforced at tweet creation time \*/

private[tweetypie] val MaxCashtags = 50

/\*\* Maximum length of a hashtag (not including the '#') \*/

private[tweetypie] val MaxHashtagLength = 100

/\*\*

\* Normalizes the text according to the unicode NFC spec.

\*/

def nfcNormalize(text: String): String = Normalizer.normalize(text, Normalizer.Form.NFC)

/\*\*

\* Return the number of "characters" in this text. See

\* [[Offset.DisplayUnit]].

\*/

def displayLength(text: String): Int = Offset.DisplayUnit.length(text).toInt

/\*\*

\* Return the number of Unicode code points in this String.

\*/

def codePointLength(text: String): Int = Offset.CodePoint.length(text).toInt

}