package com.twitter.tweetypie.tweettext

/\*\*

\* An efficient converter of indices between code points and code units.

\*/

class IndexConverter(text: String) {

// Keep track of a single corresponding pair of code unit and code point

// offsets so that we can re-use counting work if the next requested

// entity is near the most recent entity.

private var codePointIndex = 0

// The code unit index should never split a surrogate pair.

private var charIndex = 0

/\*\*

\* @param offset Index into the string measured in code units.

\* @return The code point index that corresponds to the specified character index.

\*/

def toCodePoints(offset: Offset.CodeUnit): Offset.CodePoint =

Offset.CodePoint(codeUnitsToCodePoints(offset.toInt))

/\*\*

\* @param charIndex Index into the string measured in code units.

\* @return The code point index that corresponds to the specified character index.

\*/

def codeUnitsToCodePoints(charIndex: Int): Int = {

if (charIndex < this.charIndex) {

this.codePointIndex -= text.codePointCount(charIndex, this.charIndex)

} else {

this.codePointIndex += text.codePointCount(this.charIndex, charIndex)

}

this.charIndex = charIndex

// Make sure that charIndex never points to the second code unit of a

// surrogate pair.

if (charIndex > 0 && Character.isSupplementaryCodePoint(text.codePointAt(charIndex - 1))) {

this.charIndex -= 1

this.codePointIndex -= 1

}

this.codePointIndex

}

/\*\*

\* @param offset Index into the string measured in code points.

\* @return the corresponding code unit index

\*/

def toCodeUnits(offset: Offset.CodePoint): Offset.CodeUnit = {

this.charIndex = text.offsetByCodePoints(charIndex, offset.toInt - this.codePointIndex)

this.codePointIndex = offset.toInt

Offset.CodeUnit(this.charIndex)

}

/\*\*

\* @param codePointIndex Index into the string measured in code points.

\* @return the corresponding code unit index

\*/

def codePointsToCodeUnits(codePointIndex: Int): Int =

toCodeUnits(Offset.CodePoint(codePointIndex)).toInt

/\*\*

\* Returns a substring which begins at the specified code point `from` and extends to the

\* code point `to`. Since String.substring only works with character, the method first

\* converts code point offset to code unit offset.

\*/

def substring(from: Offset.CodePoint, to: Offset.CodePoint): String =

text.substring(toCodeUnits(from).toInt, toCodeUnits(to).toInt)

/\*\*

\* Returns a substring which begins at the specified code point `from` and extends to the

\* code point `to`. Since String.substring only works with character, the method first

\* converts code point offset to code unit offset.

\*/

def substringByCodePoints(from: Int, to: Int): String =

substring(Offset.CodePoint(from), Offset.CodePoint(to))

/\*\*

\* Returns a substring which begins at the specified code point `from` and extends to the

\* end of the string. Since String.substring only works with character, the method first

\* converts code point offset to code unit offset.

\*/

def substringByCodePoints(from: Int): String = {

val charFrom = codePointsToCodeUnits(from)

text.substring(charFrom)

}

}