-- Write a program that translates normal text in English into Pig Latin.
-- Pig Latin translates words in the following way:
-- * words that start with consonants have the consonants stripped
-- and added as a suffix along with the string "ay";
-- * words that start with a vowel are appended with the string "ay".
--
-- Assumptions:
-- * No punctuation in the input text.
-- * Upper case characters only appear at the beginning of words.
--
-- Sample input: This is a test for Pig Latin Translator
-- Sample output: Is-thay is-ay a-ay est-tay or-fay Ig-pay Atin-lay Anslator-tray

class
  APPLICATION

create
  make

feature (NONE) -- Initialization
  make
    -- Run application.
    local
    do
      print ("Input your text: %N%")
      Io.read_line
      print ("The translation in Pig Latin is: %N%")
      print (translation_in_pig_latin (Io.last_string))
  end

feature -- Pig Latin translation.

translation_in_pig_latin (a_string: STRING): STRING
  -- Translation of `a_string' in Pig Latin.
  require
    a_string /= Void
  local
    words: LIST[STRING]
    word: STRING
  do
    create Result.make_empty
    words := a_string.split (' ')
    from
      words.start
    until
      words.after
    loop
      word := words.item_for_iteration
      Result.append (word_translation (word) + " ")
    end
    words.forth
  end

feature{NONE} -- Implementation
word_translation (a_string: STRING): STRING
-- Translation of word `a_string' in Pig Latin.
require
  a_string /= Void
local
  l_is_capitalized: BOOLEAN
  l_boundary, l_index: INTEGER
do
  Result := ""
if not a_string.is_empty then
  -- Find the boundary between the prefixing consonants and the rest.
  from
  l_boundary := 1
  until
    l_boundary > a_string.count or vowels.has (a_string.item (l_boundary))
  loop
    l_boundary := l_boundary + 1
  end

  -- Copy the rest to 'result'.
  from
  l_index := l_boundary
  until
    l_index > a_string.count
  loop
    Result.append_character (a_string.item (l_index))
    l_index := l_index + 1
  end

  Result.append_character ('-')

  -- Copy the consonants
  from
  l_index := 1
  until
    l_index >= l_boundary
  loop
    Result.append_character (a_string.item (l_index).as_lower)
    l_index := l_index + 1
  end

  Result.append ("ay")

  -- Capitalize the translation.
  if a_string.item (1).is_upper and Result.item(1) /= '-'
    then
    Result.put (Result.item(1).as_upper, 1)
  end
end
end

feature -- Constant

  vowels: STRING = "AOEUIaeoeui"
end