```
require 'yaml'

class EmacsTrainer

  def initialize
    @key_strokes = YAML::load_file(File.expand_path(File.join(File.dirname(__FILE__),'keystrokes.yml')))
    #~ @key_strokes = {
    #~ :recover_a_file_lost_by_a_system_crash =>
    #~ ['alt x','r','e','c','o','v','e','r','-','f','i','l','e']
    #~ }
    @expected_key_stroke = :open_file
    @key_pressed = ""
    @key_stroke_index = 0
    @count = 0
    @actual_key_stroke = ""
    @score = 0
    @tests_taken = 0
  end

  #  #  #  #

  def decode_key(key)
    if key.class == Symbol
      @actual_key_stroke = key.to_s.gsub('_',' ')
      @actual_key_stroke.gsub!('alt','Alt')
    else
      if key.class == Symbol
        @actual_key_stroke = key.to_s
      else
        case key[0]
        when 1 .. 26
          @actual_key_stroke = "Ctrl #{('a'[0] + key[0] - 1).chr}"
        when 32 .. 126
          @actual_key_stroke = "#{(key[0]).chr}"
        else
          @actual_key_stroke = key[0]
        end
      end
    end
    @actual_key_stroke
  end

  #  #  #  #

  def key_correct?
    @actual_key_stroke.to_s == @key_strokes[@expected_key_stroke][@count].to_s
  end

  #  #  #  #

  def sequence_complete?
    @count + 1 == @key_strokes[@expected_key_stroke].size
  end

  #  #  #  #

  def next_key
    @count += 1
```

```
  end

  #  #  #  #

  def repeat_key
    @count = 0
  end

  #  #  #  #

  def next_sequence
    @key_stroke_index += 1
    @key_stroke_index  = 0 if (@key_stroke_index >= @key_strokes.size)
    @expected_key_stroke = @key_strokes.keys[@key_stroke_index]
    @count = 0
  end

  #  #  #  #

  def key_stroke
    @key_strokes[@expected_key_stroke]
  end

  #  #  #  #

  def next_guess
    "#{@expected_key_stroke.to_s}"
  end

  #  #  #  #

  def correct_guess_message
    "#{@expected_key_stroke.to_s}"
  end

  def incorrect_guess_message
    "#{@expected_key_stroke.to_s} is wrong! Try "
  end


  def display_correct_keystroke
    "#{key_stroke.join(' - ')}"
  end

  def score
    @tests_taken += 1
    "#{@score}-#{(@tests_taken)}\n"
  end

  def increment_score
    @score += 1
  end
end


#  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #  #

Shoes.app(  :title => "Emacs Trainer" ,
```

```
            :weight => 'bold' ,
            :width => 760,
            :height => 572,
            :resizable => false
            ) do
  background "emacs.jpg"
  emacs = EmacsTrainer.new
  emacs.next_sequence
  para emacs.next_guess , :weight => 'bold' , :stroke => slateblue, :left => 5, :top => 0
  keypress do |k|

    #~ para "\n\n k.to_s is #{k.to_s}"
    #~ para "\n\n k.class is #{k.class}"
    #~ para "\n\n k[0] is #{k[0]}"
    emacs.decode_key k
    #~ para "\n\n  decoded key is #{emacs.decode_key k}\n \n \n"
    if emacs.key_correct?
      #~ para "\nsequence_complete? = #{emacs.sequence_complete?}\n"
      if emacs.sequence_complete?
        para emacs.correct_guess_message,:weight => 'bold' , :stroke => green, :left => 5
        para emacs.display_correct_keystroke,:weight => 'bold' , :stroke => green
        para " is Correct!",:weight => 'bold' , :stroke => green
        emacs.increment_score
        para emacs.score,:weight => 'bold' , :left => 700
        emacs.next_sequence
        para emacs.next_guess , :weight => 'bold' , :stroke => slateblue, :left => 5
      else
        emacs.next_key
      end
    else
      para emacs.incorrect_guess_message,:weight => 'bold' , :stroke => red, :left => 5
      para emacs.display_correct_keystroke,:weight => 'bold' , :stroke => black
      para emacs.score,:weight => 'bold' , :left => 700
      emacs.repeat_key
      para emacs.next_guess , :weight => 'bold' , :stroke => slateblue, :left => 5
    end
    slot.scroll_top = slot.scroll_max
  end
end
```