Summary Presentation

Evan Ohme, Katana Bierman, Brooke Adkins

Program Description

Our program is a command-line application where the user can pick from three games to play. The program uses the file I/O and object-oriented programming features of Ruby to implement Wordle, Tic-Tac-Toe, and a Ruby Trivia Quiz.

Development Description

After an initial meeting to decide on a project direction, we divided the programming workload. Since we planned on making three mini-games for our application, this was an easy task. Each person took a game which sounded interesting to them. We used a shared Git repository and a Trello workspace to track our progress.

Challenges

It was somewhat difficult finding out how to test programs which run interactively. The syntax of Ruby was fairly different from what we were used to, and we had to familiarize ourselves with the language's standard library to use it effectively. Overall, creating our application was a straightforward process.

Comparison to Other Languages

Python:

- *Typing*: Both Python and Ruby are dynamically typed languages. This is a familiar feature for anyone used to writing code in Python.
- *Syntax*: With both being dynamically typed scripting languages, Ruby has similar syntax to Python. However, Ruby has more expressive method names that use punctuation such as `includes?` and `chomp!`.

Java:

- *Typing*: Java is a statically typed language. As opposed to Ruby, each variable in Java must have its type declared at the time it is created. This can lead to less runtime errors, but these can be avoided in Ruby by employing error checking around the language's duck typing system.

- *Syntax*: Compared to Java, Ruby is much less verbose. With its purpose as a scripting language and its dynamic typing, Ruby code often requires less characters to be typed to achieve the same result.

Opinions

Each team member enjoyed their time with Ruby. We felt that the language made development fast and simple for the programmer. The language was intuitive in many ways. However, we did feel that the online documentation could be improved, as it was difficult to navigate and search through.

Highlights

Getting a random word from a word list using `Kernel.rand(max)` (can be found in `wordle.rb`):

```
rand_word = words_list[rand(words_list.length)]
```

A simple class which represents a quiz question to be read from a text file (can be found in `quiz.rb`):

```
class Question
  def initialize(line)
    @question, @opt1, @opt2, @opt3, @opt4, @ans = line
  end

  def ask()
    puts @question
    puts "  a) #{@opt1}"
    puts "  b) #{@opt2}"
    puts "  c) #{@opt3}"
    puts "  d) #{@opt4}\n\n"
    val = ""
    loop do
      print "Your answer (a, b, c, d): "
      val = gets.chomp
      if "abcd".include? val
        break
      else
        puts  "ERROR: Unrecognized answer. Please try again."
      end
    end
    return val == @ans
  end
end
```

A "block" which, for each line of a file, creates a Question object using the information in that line (can be found in `quiz.rb`):

```
File.foreach('questions.txt') do |line|
  line = line.chomp.split('#')
  questions.push(Question.new(line))
end
```

A loop which repeatedly asks the user to enter the game they want to play, before switching to that sub-program (can be found in `main.rb`):

```
loop do
    case Input::prompt
      when "1", "tic-tac-toe"
        puts "Starting tic-tac-toe...\n\n"
        exec "ruby tic_tac_toe.rb"
      when "2", "wordle"
        puts "Starting wordle...\n\n"
        exec "ruby wordle.rb"
      when "3", "ruby quiz"
        puts "Starting quiz...\n\n"
        exec "ruby quiz.rb"
      when "", nil
        break
      else
        puts "ERROR: Unrecognized command. Please try again."
    end
  end
```