The Hangman Interview Question

Ron Miguel <ron@factual.com>
Antwort an: ron@factual.com
An: maximilien.rzepka@gmail.com

22. März 2013 17:54

The Goal

Write a program that plays the Hangman game. A description of the game can be found at http://en.wikipedia.org/wiki/Hangman_(game).

The Problem

Your goal is to use a provided API to play Hangman efficiently. You need to guess a word using as few guesses as possible, and make no more than maxWrongGuesses incorrect guesses. **You are writing the letter/word guessing strategy.**

We would like you to use the provided Java APIs and to write your solution in Java or a JVM-based language. However, if are not comfortable with that, please feel free to use any other reasonably mainstream programming language (C++, Python, Ruby, etc.). If you do use another language, please make sure that your program can accept a dictionary file and a list of test words (that are in the dictionary). The output of the program should be a score for each test word and the average score across all test words. Also, if you don't use Java, please provide build instructions if they are not straightforward.

Your score for a word will be:

 # letter guesses + # number of incorrect word guesses if you guessed the word right before exceeding maxWrongGuesses incorrect guesses

or

25 if you lost the game before guessing the word correctly.

You will need to write an implementation of the GuessingStrategy interface and some code to use your GuessingStrategy on a HangmanGame instance.

The pseudocode to run your strategy for a HangmanGame is:

```
// runs your strategy for the given game, then returns the score
public int run(HangmanGame game, GuessingStrategy strategy) {
  while (game has not been won or lost) {
    ask the strategy for the next guess
    apply the next guess to the game
  }
  return game.score();
}
```

A trivial strategy might be to guess 'A', then 'B', then 'C', etc. until you've guessed every letter in the word (this will work great for "cab"!) or you've lost.

Every word you encounter will be a word from the words.txt file.

Example

For example, let's say the word is FACTUAL.

Here is what a series of calls might look like:

```
HangmanGame game = new HangmanGame("factual", 4); // secret word is factual, 4 wrong guesses are allowed
System.out.println(game);
new GuessLetter('a').makeGuess(game);
System.out.println(game);
new GuessWord("natural").makeGuess(game);
System.out.println(game);
new GuessLetter('x').makeGuess(game);
```

```
System.out.println(game);

new GuessLetter('u').makeGuess(game);

System.out.println(game);

new GuessLetter('l').makeGuess(game);

System.out.println(game);

new GuessWord("factual").makeGuess(game);

System.out.println(game);
```

The output would be:

```
-------; score=0; status=KEEP_GUESSING

-A---A-; score=1; status=KEEP_GUESSING

-A---A-; score=2; status=KEEP_GUESSING

-A---A-; score=3; status=KEEP_GUESSING

-A--UA-; score=4; status=KEEP_GUESSING

-A--UAL; score=5; status=KEEP_GUESSING

FACTUAL; score=5; status=GAME_WON
```

game.score() will be 5 in this case since there were 4 letter guesses and 1 incorrect word guess made.

Sample Data

As a baseline, here are scores for a reasonably good guessing strategy against a set of 15 random words. Your strategy will likely be better for some of the words and worse for other words, but the average score/word should be in the same ballpark.

COMAKER = 25 (was not able to guess the word before making more than 5 mistakes)
CUMULATE = 9
ERUPTIVE = 5
FACTUAL = 9
MONADISM = 8
MUS = 25 (was not able to guess the word before making more than 5 mistakes)
NAGGING = 7
OSES = 5
REMEMBERED = 5
SPODUMENES = 4
STEREOISOMERS = 2
TOXICS = 11
TRICHROMATS = 5
TRIOSE = 5
UNIFORMED = 5

Resources

You should have been provided with a zip file with source code and a dictionary file to get you started. If you were not sent this zip file, or you have any questions about the contents, please let us know right away.

The resources contain a dictionary file called words.txt. You can assume all words that your program will be tested with come from this dictionary file.

Judging

Your solution will be graded on the following criteria

- **code quality, readability, and design**. In terms of quality, please write your code as if it would eventually be pushed into production.
- **performance** (speed/memory footprint). We are more concerned with average time per game, so expensive one-time initialization is okay (as long as it's not too egregious)
- **total score for ~1000 random words**, compared to the total score of several reference implementations for the same ~1000 words (max wrong guesses is typically set to 5)

Hangman Interview Attachments.zip
447K