A6: Efficient Hustling

Assigned: Friday, April 11, 2014
Due: Friday, April 25, 2014, 11:59 P.M.
Type: Individual

Overview

Movies like *The Hustler* and *The Color of Money* focus on the classic pool hustler character: a con artist who makes money by hiding prodigious abilities (and losing) early when the stakes are low and then applying those abilities (and almost always winning) when the stakes are considerably higher. This assignment focuses on the seedy world of *Hangman hustling*.

In case you aren't familiar with the game Hangman, the rules are as follows:

1. One player chooses a secret word then writes out a number of dashes equal to the word length.
2. The other player begins guessing letters. Whenever she guesses a letter contained in the secret word, the first player reveals each instance of that letter in the word. Otherwise, the guess is wrong.
3. The game ends either when all the letters in the word have been revealed or when the guesser has run out of guesses. (Correct letters don't count against the guess limit.)

More information, including online versions of the game, can be found at the links below.

- `http://en.wikipedia.org/wiki/Hangman_(game)`
- `http://www.gamehangman.com/`
- `http://www.webhangman.com/`

There are different strategies on both sides of the Hangman game. For the player choosing the secret word, obvious strategies include selecting obscure words (e.g., logorrhea) or words with multiple occurrences of seldom-used letters (e.g., jazz). For the player guessing the word, common strategies are to guess vowels first (a, e, i, o, u), guess letters in order of frequency in English (e, a, r, i, o, t)[1], or target common words (e.g., four-letter words: time, good, have, year, last, long, make)[2].

While there are many viable strategies for building competitive computer game players, there is one approach that is often neglected—cheating. Why spend all the effort trying to teach a computer the nuances of strategy when you can simply write a program that plays dirty and wins handily all the time? In this assignment, you will build a mischievous program that can play clean or dirty. If it plays clean, then you get a classic game of Hangman where your chance of winning depends on your guessing strategy and your skill in recognizing possible words based on partial patterns. If it plays dirty, it bends the rules of Hangman to trounce its human opponent almost every time. In solving this assignment, you'll cement your skills with collections in Java and will hone your general programming savvy. Plus, you'll end up with a highly entertaining piece of software, one that you can take to the mean streets where Hangman is played for keeps.

[1] *Oxford English Dictionary*, 2004
[2] Oxford English Dictionary (online), 2014

Discussion

Fundamental to a fair game of Hangman is the fact the first player accurately represents the word she has chosen. That way, when the other players guess letters, she can reveal whether that letter is in the word. But what happens if the player doesn't do this? This gives the player who chooses the secret word an enormous advantage. For example, suppose that you're the player trying to guess the word, and at some point you end up revealing letters until you arrive at this point with only one guess remaining:

```
d o - b l e
```

There are only two words in the English language that match this pattern: "doable" and "double." If the player who chose the hidden word is playing fairly, then you have a fifty-fifty chance of winning this game if you guess 'a' or 'u' as the missing letter. However, if your opponent is playing dirty and hasn't actually committed to either word, then there is no possible way you can win this game. No matter what letter you guess, your opponent can claim that she had picked the other word, say that your guess is incorrect, and win the game. That is, if you guess that the word is "doable," she can pretend that she committed to "double" the whole time, and vice-versa.

Let's illustrate this technique with an example. Suppose that you're playing Hangman and it's your turn to choose a word, which we'll assume is of length four. Rather than committing to a secret word, you instead compile a list of every four-letter word in the English language. For simplicity, let's assume that English only has only the following four-letter words:

```
ally, beta, cool, deal, else, flew, good, hope, ibex
```

Now, suppose that your opponent guesses the letter 'e.' You now need to tell your opponent which letters in the word you've "picked" are e's. Of course, you haven't picked a word, and so you have multiple options about where you reveal the e's.

If you'll notice, every word in your word list falls into one of five "word families:"

`- - - -` (ally, cool, good)

`- e - -` (beta, deal)

`- - e -` (flew, ibex)

`e - - e` (else)

`- - - e` (hope)

Since the letters you reveal have to correspond to some word in your word list, you can choose to reveal any one of the above five families. There are many ways to pick which family to reveal—perhaps you want to steer your opponent toward a smaller family with more obscure words, or toward a larger family in the hopes of keeping your options open. For this assignment, in the interests of simplicity, we'll adopt the latter approach and always choose the largest of the remaining word families. In this case, it means that you should pick the family `- - - -`. This reduces your word list down to

```
ally, cool, good
```

and since you didn't reveal any letters, you would tell your opponent that his guess was wrong.

Let's follow this strategy a few steps further. Given this three-word word list, if your opponent guesses the letter o, then you would break your word list down into two families:

`- o o -` (cool, good)

`- - - -` (ally)

The first of these families is larger than the second, and so you choose it, revealing two o's in the word and reducing your list down to

```
cool, good
```

But what happens if your opponent guesses a letter that doesn't appear anywhere in your word list? For example, what happens if your opponent now guesses 't'? This isn't a problem. If you try splitting these words apart into word families, you'll find that there's only one family: the family `- - - -` containing both cool and good. Since there is only one word family, it's trivially the largest, and by picking it you'd maintain the word list you already had.

Now there are two possible outcomes of this game. First, your opponent might be smart enough to pare the word list down to one word and then guess what that word is. In this case, you should congratulate him—that's an impressive feat considering the scheming you were up to! Second, and by far the most common case, your opponent will be completely stumped and will run out of guesses. When this happens, you can pick any word you'd like from your list and say it's the word that you had chosen all along. The beauty of this setup is that your opponent will have no way of knowing that you were dodging guesses the whole time—it looks like you simply picked an unusual word and stuck with it the whole way.

Requirements

You must complete the implementation of a Java class named `Hangman` that provides a number of public methods, each of which is discussed below. You are free to create as many private methods as you like. You are also free to import and use any class that you would like.

`Hangman(InputStream in)`

The constructor for the `Hangman` class builds the lexicon to be used from the provided `InputStream` object. The caller must associate one of the provided word lists (see Provided Resources) with an `InputStream` and pass that reference as the constructor parameter. The constructor must read all the words from the provided `InputStream` and store them in an appropriate collection, which will be the lexicon for this `Hangman` object. Most of the code has been provided for you. You only have to add the statements necessary to actually add the strings being read into your internal lexicon.

`Hangman()`

The parameterless constructor for the `Hangman` class calls the second constructor with a default word list of `lowerwords.txt`. There is no reason to change this constructor in any way.

`setSecretWord(String word)`

This method sets the secret word for this round of play.

Methods to get user input

Methods are provided for you to read in user input. You only have to add a line or two to these methods to validate the input. Don't change the technique used to read input since it words well with the grading harness.

`playClean(boolean verbose)`

This method plays a clean game of Hangman. If the parameter `verbose` is true this method must print extra information about the state of the game play (see Example Gameplay). To play a clean game of Hangman, this method must do the following.

1. Prompt the user for a word length for this game, reprompting as necessary until the user enters a number such that there is at least one word that's exactly that long.
2. Choose the words of this length from the game lexicon as the set of possible words for this game.
3. If it has been set already with a call to `setSecretWord()`, this method is to use the current value of the secret word and assume that it is consistent with the word length chosen above. If it has not been set, this method should pick a secret word at random from the possible set of words for this game play. In either case, this method must *not* prompt the user for a secret word.
4. Prompt the user for a number of guesses, which must be an integer greater than zero and less than or equal to 26.
5. Repeat the following steps until the number of guesses allowed is exhausted or until the user has correctly guessed the secret word.
   (a) Display the appropriate information about the current game state, taking into account the value of the parameter `verbose` (see Example Gameplay).
   (b) Prompt the user for a single-letter guess, reprompting as necessary until the user enters a lowercase letter. If the user types more than a single character, ignore all but the first. Make sure the first character is a lowercase English letter.
   (c) If the guessed letter is in the secret word, change the word "mask" (the dashes that show the user's progress) to reveal this letter in its correct location. If the guessed letter is not in the secret word, add this letter to the list that is displayed as incorrect guesses and subtract one from the number of guesses remaining. Note that correct letters to not decrease the number of guesses remaining.
6. Once the game loop stops, display the appropriate message depending on whether the user won or lost (see Example Gameplay).

`playDirty(boolean verbose)`

This method plays a dirty game of Hangman. If the parameter `verbose` is true this method must print extra information about the state of the game play (see Example Gameplay). To play a dirty game of Hangman, this method must do the following.

1. Prompt the user for a word length for this game, reprompting as necessary until the user enters a number such that there is at least one word that's exactly that long.
2. Choose the words of this length from the game lexicon as the set of possible words for this game.

3. Select the minimum (lexicographically first) word in the set of possible words as the initial value of the secret word. In principle it doesn't matter which word is selected, but for testing/grading purposes you *must* choose the minimum.

4. Prompt the user for a number of guesses, which must be an integer greater than zero and less than or equal to 26.

5. Repeat the following steps until the number of guesses allowed is exhausted or until the user has correctly guessed the secret word.

   (a) Display the appropriate information about the current game state, taking into account the value of the parameter `verbose` (see Example Gameplay).

   (b) Prompt the user for a single-letter guess, reprompting as necessary until the user enters a lowercase letter. If the user types more than a single character, ignore all but the first. Make sure the first character is a lowercase English letter.

   (c) Partition the set of possible words into disjoint word "families" with respect to the current letter guess; that is, each word in a given family shares the same "mask" with respect to this letter. (See Example Gameplay for example family masks.)

   (d) Update the set of possible words for this game to the largest word family, that is, the one with the most words. If the words in this family contain the guessed letter, change the word "mask" (the dashes that show the user's progress) to reveal this letter in its correct location. If the words in this family do not contain the guessed letter, add this letter to the list that is displayed as incorrect guesses and subtract one from the number of guesses remaining. Note that correct letters to not decrease the number of guesses remaining.

6. Once the game loop stops, display the appropriate message depending on whether the user won or lost (see Example Gameplay). If (likely *when*) the user loses the game, select the minimum word from the current set of possible words to display as the secret word.

As an example of a step in the the dirty Hangman algorithm, consider the following snippet from the middle of a verbose run.

```
The secret word is:       curly
Your progress:            -u---
Wrong guesses:            aiseob
Guesses remaining:        19
Number of words possible: 33
Guess a letter: c
Possible word families:
-u--- 21
-u-c- 8
-uc-- 2
cu--- 2
```

The user had already guessed a, i, s, e, o, and b with no success. In the round immediately preceding this one, the user had guessed u and the game selected a family that contains one u in the second letter position (`-u--`). On the current round of play, the user has guessed c. The game now must analyze the current set of 33 possible words and construct word families with respect to c. Four families result:

`-u--` with 21 words in which c does not appear at all.

`-u-c-` with 8 words in which c appears in the fourth position.

`-uc-` with 2 words in which c appears in the third position.

`cu--` with 2 words in which c appears in the first position.

Since the family (`-u--`) is the largest, it will be the new set of possible words going forward. Notice that this ensures that the mask stays consistent with all of the users guesses from the beginning of this round of play.

Provided Resources

You are provided with the following resources as part of the assignment.

- A source code file, `Hangman.java`. This is the class that you must implement and submit as your solution to the assignment.
- A jar file, `WordLists.jar`. This jar file contains the following text files that you are allowed to use as lexicons for the Hangman class: `CSW12.txt` (the word list used in international Scrabble tournaments), `OWL.txt` (the word list used in North American Scrabble tournaments), `sowpods.txt` (SOWPODS—a commonly used combination of the CSW list and the OWL list), `words.txt` (the word list supplied in UNIX distributions), `small.txt` (a relatively small subset of the SOWPODS list), and `lowerwords.txt` (a list of words created for Hangman)[3].
- A sample client, `A6.java`. This driver class illustrates basic calls to the `Hangman` methods, and it also demonstrates how to associate a text file contained in `WordList.jar` with an `InputStream` object.

Example Gameplay

The following are example runs of a correct Hangman implementation. Assume that the object `game` is an instance of the `Hangman` class using the `lowerwords.txt` lexicon.

`game.playClean(false)` — Playing in clean silent mode

```
****  H A N G M A N   ****
What word length do you want to play with? 7
How many guesses would you like? [1..26] 10

Your progress:           -------
Wrong guesses:
Guesses remaining:       10
Guess a letter: e

Your progress:           -------
Wrong guesses:           e
Guesses remaining:       9
Guess a letter: i
```

[3]Acknowledgment: Julie Zelinski, Stanford CS

```
Your progress:           -------
Wrong guesses:           ei
Guesses remaining:       8
Guess a letter: x

Your progress:           -------
Wrong guesses:           eix
Guesses remaining:       7
Guess a letter: a

Your progress:           -a---a-
Wrong guesses:           eix
Guesses remaining:       7
Guess a letter: n

Your progress:           -an--an
Wrong guesses:           eix
Guesses remaining:       7
Guess a letter: g

Your progress:           -ang-an
Wrong guesses:           eix
Guesses remaining:       7
Guess a letter: h

Your progress:           hang-an
Wrong guesses:           eix
Guesses remaining:       7
Guess a letter: m

You won! The secret word was hangman.
```

`game.playClean(true)` — Playing in clean verbose mode

```
****  H A N G M A N   ****
 (playing clean)
What word length do you want to play with? 7
How many guesses would you like? [1..26] 10

The secret word is:      hangman
Your progress:           -------
Wrong guesses:
Guesses remaining:       10
Number of words possible: 7359
Guess a letter: e

The secret word is:      hangman
Your progress:           -------
Wrong guesses:           e
```

```
Guesses remaining:        9
Number of words possible: 7359
Guess a letter: i

The secret word is:       hangman
Your progress:            -------
Wrong guesses:            ei
Guesses remaining:        8
Number of words possible: 7359
Guess a letter: x

The secret word is:       hangman
Your progress:            -------
Wrong guesses:            eix
Guesses remaining:        7
Number of words possible: 7359
Guess a letter: a

The secret word is:       hangman
Your progress:            -a---a-
Wrong guesses:            eix
Guesses remaining:        7
Number of words possible: 81
Guess a letter: n

The secret word is:       hangman
Your progress:            -an--an
Wrong guesses:            eix
Guesses remaining:        7
Number of words possible: 3
Guess a letter: g

The secret word is:       hangman
Your progress:            -ang-an
Wrong guesses:            eix
Guesses remaining:        7
Number of words possible: 1
Guess a letter: h

The secret word is:       hangman
Your progress:            hang-an
Wrong guesses:            eix
Guesses remaining:        7
Number of words possible: 1
Guess a letter: m

You won! The secret word was hangman.
```

`game.playDirty(false)` — Playing in dirty silent mode

```
****  H A N G M A N   ****
What word length do you want to play with? 4
How many guesses would you like? [1..26] 10

Your progress:          ----
Wrong guesses:
Guesses remaining:      10
Guess a letter: a

Your progress:          ----
Wrong guesses:          a
Guesses remaining:      9
Guess a letter: e

Your progress:          ----
Wrong guesses:          ae
Guesses remaining:      8
Guess a letter: i

Your progress:          ----
Wrong guesses:          aei
Guesses remaining:      7
Guess a letter: o

Your progress:          ----
Wrong guesses:          aeio
Guesses remaining:      6
Guess a letter: u

Your progress:          -u--
Wrong guesses:          aeio
Guesses remaining:      6
Guess a letter: b

Your progress:          -u--
Wrong guesses:          aeiob
Guesses remaining:      5
Guess a letter: f

Your progress:          -u--
Wrong guesses:          aeiobf
Guesses remaining:      4
Guess a letter: c

Your progress:          -u--
Wrong guesses:          aeiobfc
Guesses remaining:      3
```

```
Guess a letter: l

Your progress:            -u--
Wrong guesses:            aeiobfcl
Guesses remaining:        2
Guess a letter: d

Your progress:            -u--
Wrong guesses:            aeiobfcld
Guesses remaining:        1
Guess a letter: s
Sorry, but you didn’t guess the secret word: hugh.
```

`game.playDirty(true)` — Playing in dirty verbose mode

```
****  H A N G M A N   ****
 (playing dirty)
What word length do you want to play with? 4
How many guesses would you like? [1..26] 10

The secret word is:       abba
Your progress:            ----
Wrong guesses:
Guesses remaining:        10
Number of words possible: 2235
Guess a letter: a
Possible word families:
---- 1422
---a 62
--a- 196
-a-- 422
-a-a 15
-aa- 2
a--- 91
a--a 14
a-a- 11

The secret word is:       beck
Your progress:            ----
Wrong guesses:            a
Guesses remaining:        9
Number of words possible: 1422
Guess a letter: e
Possible word families:
---- 839
---e 190
--e- 118
--ee 6
-e-- 170
```

```
-e-e 9
-ee- 47
e--- 25
e--e 5
e-e- 12
ee-- 1

The secret word is:       bibs
Your progress:            ----
Wrong guesses:            ae
Guesses remaining:        8
Number of words possible: 839
Guess a letter: i
Possible word families:
---- 550
---i 9
--i- 80
-i-- 174
-i-i 4
i--- 17
i--i 1
i-i- 4

The secret word is:       blob
Your progress:            ----
Wrong guesses:            aei
Guesses remaining:        7
Number of words possible: 550
Guess a letter: o
Possible word families:
---- 216
---o 9
--o- 57
-o-- 189
-o-o 9
-oo- 50
o--- 16
o--o 3
o-o- 1

The secret word is:       blum
Your progress:            ----
Wrong guesses:            aeio
Guesses remaining:        6
Number of words possible: 216
Guess a letter: u
Possible word families:
---- 10
--u- 37
```

```
-u-- 164
-u-u 2
u--- 2
u--u 1

The secret word is:       buck
Your progress:            -u--
Wrong guesses:            aeio
Guesses remaining:        6
Number of words possible: 164
Guess a letter: b
Possible word families:
-u-- 127
-u-b 3
-ub- 8
bu-- 25
bu-b 1

The secret word is:       cuff
Your progress:            -u--
Wrong guesses:            aeiob
Guesses remaining:        5
Number of words possible: 127
Guess a letter: f
Possible word families:
-u-- 113
-u-f 3
-uf- 1
-uff 3
fu-- 7

The secret word is:       cull
Your progress:            -u--
Wrong guesses:            aeiobf
Guesses remaining:        4
Number of words possible: 113
Guess a letter: c
Possible word families:
-u-- 94
-uc- 10
cu-- 9

The secret word is:       dull
Your progress:            -u--
Wrong guesses:            aeiobfc
Guesses remaining:        3
Number of words possible: 94
Guess a letter: l
Possible word families:
```

```
-u-- 74
-u-l 1
-ul- 5
-ull 6
lu-- 7
lull 1

The secret word is:       dump
Your progress:            -u--
Wrong guesses:            aeiobfcl
Guesses remaining:        2
Number of words possible: 74
Guess a letter: d
Possible word families:
-u-- 60
-u-d 2
-ud- 3
-udd 2
du-- 7

The secret word is:       gums
Your progress:            -u--
Wrong guesses:            aeiobfcld
Guesses remaining:        1
Number of words possible: 60
Guess a letter: s
Possible word families:
-u-- 25
-u-s 18
-us- 11
-uss 2
su-- 2
su-s 2
Sorry, but you didn’t guess the secret word: hugh.
```

Assignment Submission

You must turn in only the `Hangman.java` file to Web-CAT no later than the published deadline. **Do NOT submit the provided WordLists.jar file.** Submissions made within the 24 hour period after the published deadline will be assessed a late penalty of 15 points. No submissions will be accepted more than 24 hours after the published deadline.

Acknowlegments

This assignment is based on ideas and assignments from Keith Schwarz (Stanford) and Owen Astrachan (Duke).