CompSci 201

Data Structures and Algorithms

Hangman

You can browse the code we provide and you can snarf it using our Ambient/Eclipse plugin — check out the instructions on how to download the ambient system and how to use it to check out a project.

The snarf url for this CompiSci 201 is

`http://www.cs.duke.edu/courses/compsci201/fall13/snarf/`.

Provided Code/Classes

You are provided with four classes.

1. The class HangmanStats will be the class that you'll use/modify in part 1 — the statistical/analytical part of the program. It has a main method:
   `public static void main(String args[])`
   that executes the code in the class `HangmanStats`.
2. The class HangmanFileLoader loads a textfile and has a method to select and return a random word. Look in the sample code in `HangmanStats` and `HangmanGame` to see how to use the class. *You shouldn't need to modify this code at all.*
3. The class HangmanExecuter has the `public static void main(String args[])` function that is the starting point of the Hangman game you will write. It creates a `HangmanGame` and invokes the `play` method. You can write the game without modifying `HangmanExecuter`. If you do modify it, please include an explanation in your README.

4. The class HangmanGame is the game-playing code, you'll modify the `play` method and you may add new methods or state/instance variables to this class.

Statistical/Forensic Analysis

You are given code in HangmanStats that gives you the number of words with 4 letters when `HangmanLoader.getRandomWord` is called 10,000 times. When we modified this code to call `HangmanLoader.getRandomWord` 10,000 times for word-lengths of 4-10 we obtained the below results from one run.

WORD LENGTH	# WORDS
4	2209
5	3792
6	4959
7	5412
8	5379
9	4947
10	4059

If run again, the numbers above will be similar, but different. You'll need to decide on a strategy and write code to get estimates of the number of different words for each word-length from 4-20 (inclusive).

Java Help

Expect to spend some time getting used to the Java language. We have written a handy-dandy cheat sheet that shows example java code and matches it to Python and Matlab. Also, the Java API docs provide info on what sort of operations Strings or other objects support. Using google can also work.

Coding Guidelines

Well written code is easy to read and makes graders happy.

The sample output on the assignment writeup page should be generically followed in that the user should see a representation of the secret word with guesses filled in, the letters used so far, and so on.

- Methods should be *brief/short*, ideally 10-20 lines, but that's a guideline, not a requirement.
- Methods that do more than one thing are candidates to be divided into more than one method so that each method does only one thing. A prime candidate is the *play* method, it will consist of picking letters, updating word-displays, determining if the game should continue, validating input, and so on. In principle, each of these should be a different method, though in practice that's not always possible.
- A method should minimize its side-effects. Changing one parameter and returning a value are OK. Changing multiple parameters and returning a value should be avoided, though sometimes this is ok too — have a justification for the code you write.
- Use good method and parameter names and follow the Java naming guidelines.

Things That Might Be Graded

To get full credit on an assignment your code should be able to do the following.

- Display output clearly and completely, we use this to grade you.
- Match the given format for the assignment, we have to grade a lot of assignments and you should be able to meet a spec. The format may be given explicitly, or may be shown in sample output. Feel free to ask for clarification on Piazza.
- Successfully perform all actions written in the assignment (i.e. you should correctly count guesses for Hangman).
- Your code should be clear and easy to read.
- Your variable names should make sense. If you name a variable that keeps track of guesses something vague , unrelated, or inappropriate makes grading much harder and you may be penalized.
- You *MUST* create and submit a README with your work.
- All writeup questions must be answered clearly and completely. Don't just state an answer without an explanation or evidence, it will cost you points. This includes code or data when asked for.
- If you have questions about the assignment that are not specific to your code, please ask on Piazza and we will be happy to clarify. If your question is specific to the code you've written please post it privately on Piazza, or bring it to the Link or Office Hours and don't share it directly with your classmates.

This list is not a complete rubric for grading, and there is a better breakdown on the main page. Please start early, work carefully, and ask any questions you have and we'll do our best to make sure that everyone gets all of the points that they deserve.