2016 Java Coding Competition

The Problem

SF Advertising is a small company with the ability to build high-tech programs that scan and evaluate websites. In addition, they use the scan to provide key statistics. You are a new hire with the company and have been tasked to review a website and provide an appropriate analysis. There are some basic requirements that must be followed, but SF Advertising appreciates fresh ideas and will gladly accept additional enhancements.

The initial website "statefarm.com" is provided in the workspace and is a .zip file that contains a number of .html files. Please unzip these files in your home directory.

Examples: Mac /Users/username/
Windows C:\users\username\
Linux /home/username/

Please scan the website found in the .zip file and provide answers to the following questions/Junits. You can use the included jsoup framework or another technology of your choice:

- AgentParserTest – Complete these tests first.
  - For a specific agent return the products they are licensed to sell
  - For a specific agent return all the associated data
    - name
    - languages (uses title element in div tag vs text element)
    - address (Street Address)
    - etc.
- AgentLocatorTest
  - Find a single agent by first and last name
  - Find multiple agents by last name/first name
  - Find count of agents using search criteria
    - Data exists for AZ, CO, NM, NV, UT only
  - Find count of all agents
  - Find all agents with a unique full name
  - Find the most popular first name/last name/suffix

JUnit tests are the highest authority related to requirements and can be found by following the path *src/test/java*. Additional details/comments are found in the JUnits. AgentParserTest should be completed first! The AgentParserTest is running against the 2 agent files included in the "src/test/resources" directory. The AgentLocatorTest will run against all agent files in the unzipped "www.statefarm.com" directory.

********* **Do not change anything in the JUnit tests!** **********

First Actions:

- Import the problem statement into your IDE.
- We have provided Maven dependency for JUnit 4. If you are not set up with the recommended IDE (Spring Tool Suite), you may need to add JUnit 4.
- If you identify any additional libraries you would like to use, please add them to the pom.xml file or copy the .jar files into the resources folder
- Run your JUnit tests, code, and repeat.
  - For questions on how to run JUnits, please see document in the project: "How to run JUnits"

2016 Java Coding Competition

When you are done:

- Update the feedback.txt file and include the following information:
  - Your team – name of each individual participating.
  - How many JUnits you were able to execute successfully.
  - Document and describe the additional "nice to have" features included, to help the judges properly grade your submission and explain how to properly execute new enhancements.

- Push your changes to one single branch for you and your teammate. Open a single pull request against the main State Farm Coding Competition repository before 11:59PM CST on October 15, 2016.
  - If you make any commits after midnight without prior approval from codingcompetition@statefarm.com, your submission will be disqualified.
  - If you so choose, you may open a pull request at any time during the competition and continue to update it as long as you do not make any commits after midnight.

- As a **back-up only** you can export your IDE project (The judges will need your entire project to run the JUnits) as a zip file and email the zip file to codingcompetition@statefarm.com with subject line: State Farm Coding Competition – Name 1/Name 2

Rules

- Contestants cannot seek help from individuals outside their team.
- Teams are expected to have the necessary tools and JARs preloaded on their machines **prior** to the competition.
- If you believe this document and the JUnit tests conflict, the JUnit tests are the highest authority.

How you will be Graded

Components of submission will be weighted as follows.

- 100% core requirements met, including:
  - Number of JUnits that pass using correct functionality in the program
  - Maintaining Object Oriented Programming principles
  - Code documentation
  - Code must compile and execute
- Do not complete any Bonus unless you have all the JUnit tests completed
  - Bonus credit awarded for any extra features added (up to 10%)

In the event of a tie, we will further judge your solution based on: code cleanliness, maintainability, and adherence to object-orientated principles.

<u>Bonus "Nice to have" features</u>:

- Bonus Credit – Do not complete any bonus features unless you have completed all the required functionality and all JUnits pass.
  - Create additional functionality and demonstrate through a Graphical User Interface (GUI), command line interface, JUnits, or web UI.
    - Search engine
    - Find common words
    - Other creative analysis
  - If you can think of any other useful features to add, we appreciate ingenuity and will gladly accept any useful enhancements. Be sure to document any bonus features in the feedback text file.