JAVA CODING EXERCISE

We are passionate about code and hope you are too! We'd love to see how you implement a common business requirement - so please choose one requirement from the list below, code up a solution and return it to us so we can have a look at it.

Please see below for more detailed instructions.

Good luck!

/The Expedia Software Development Team, Montreal

This test has been designed to evaluate your coding style and capabilities. We do not recommend investing more than **4 hours** but if you do, it does not guarantee that you will go to the next step in the interview process. The expectation is not that the entire problem is solved perfectly and completely. What we expect, is that when you stop, you document all limitations of your work and the reasoning behind each of them.

Instructions

1. Choose one of the Business Requirements below
2. Develop a solution that solves the Business Requirement and make sure that any assumption is documented and that your code is production-ready.
3. Use **Java** as your main language but be original. We want to see how you code so, avoid “copy/paste” from Google and try to solve the problem your own way. If you use “copy/paste code” when solving the main problem, please make sure to document the reference and why you did it.
4. Once the code is to your satisfaction - follow the steps below:
    a. Put all your code in a folder named with your first and last name (no space, e.g. JohnDoe) and make sure that.
    b. Zip your folder into a single file.
    c. Send an email to expediamtltech@gmail.com with "Java Coding Exercise" as subject and attach your zip file to the email.

Football Scoring Dashboard

Develop an application that prints out a scoring dashboard as text during a football match. The football scoring dashboard would have output the following at the 80th minute in the 1966 Football world cup final between England and West Germany: "**England 2 (Hurst 18' Peters 78') vs. West Germany 1 (Haller 12')**". The application's required inputs are singular entries following the following flow:

1. The Football Scoring Dashboard needs to know when a game starts through being supplied a string of this format: "**Start: '<*Name of Home Team*>' vs. '<*Name of Away Team*>'**".
   a. Example: "**Start: 'England' vs. 'West Germany'**"
2. After the start command has been given, acceptable inputs to tell the Dashboard when goals are scored follow the following structure: "**<*minute*> '<*Team*>' <*name of scorer*>**".
   a. Example: "**11 'West Germany' Haller**"
3. The tool should be able to compute the '**print**' command at any time during the course of a game to print the aggregated scoring statistics of the match.
   a. Example: If tool is given the '**print**' command, it should output the following: "**England 0 vs. West Germany 1 (Haller 12')**" if that is the only goal that has been scored at that point.
4. The Dashboard knows a game has ended through the '**End**' command.
5. The tool should cater for the following error conditions:
   a. If the Football Scoring Dashboard is given any commands while a game is not in progress it should report 'No game currently in progress'.
   b. If a game is in progress and it is not able to understand the given command it should return: '**input error - please type 'print' for game details**'.
   c. If a game is *not* in progress and it is not able to understand the given command, it should return: '**input error - please start a game through typing 'Start: '<Name of Home Team>' vs. '<Name of Away Team>'**'.

Hangman

Develop an application for playing the classic hangman:
A user is prompted to enter letters until she can figure out the secret word. Every time a selected letter is not part of the secret word, a body part is added to the hangman. The game is won if the user finds the secret word before the hangman is completed (before he is hung or fully drawn). The game is lost if the secret word is not guessed before the hangman is completed.

1. Select the secret word amongst a list of at least 50 English words.

2. Prompt the user to either pick a letter or guess a word.
3. If the letter picked is not part of the secret word, or the guessed word is incorrect, add one component of the hangman in this order: Head, left eye, right eye, nose, mouth, body, left arm, right arm, left leg, right leg.
4. At every user entry, display an underscore (_) or the correctly guessed letter for each of the secret word’s letters, and show the current state of the hangman (ascii is ok)
5. If the user selects a letter already picked, return a warning. If that same letter is picked again later in the game, this counts as an incorrect pick.
6. The code should be such that support for non-English language is easy to implement.