Practical Test

Adgistics Motors

Introduction and Problem Description

Adgistics Motors is a fictional car sales company. The company consists of *many* dealerships distributed throughout the world.

Each dealership runs it's own IT system, however they each host a local web service which can be hit in order to get back the following data related to the dealership:

- *Total Sales* - The total sales that the dealership has made.
- *Available Stock* - The total count of cars that the dealership has available to sell.

Adgistics Motors would like to be able to generate reports based on all of the dealerships in an on-demand basis. They would most likely only run the report once a month. Specifically they require the following reports:

- *Top performing dealerships* - The top 100 dealerships in terms of overall sales.
- *Low stock dealerships* - All the dealerships which have less than 10 available stock.

Project Description

A project, named *AdgisticsMotors.Report*, has been configured for this purpose. It contains the following items which you can use in aiding you to develop your solution.

- ***DealershipsList.txt***
  This file contains a list of each of the dealerships to be reported on. It also contains the unique Uri endpoint address for the dealership.

- ***DealershipService***
  This is a ***fake*** service that has been created, which you can use for your development. It simulates a request to each of the "dealership web services" and returns back some fake sales and stock data for that dealership.

  You will have to pass each of the dealers identifiers and Uri's (obtained from the *DealershipList.txt*) to the *GetDealershipData* method contained within this class. This will then return some ***fake*** data to you, which you can use for your reports.

  The *GetDealershipData* doesn't really make any real web requests, we have written some placeholder code within it to simulate a real life scenario. It has a random thread delay within it in order to simulate network contention, and it will randomly throw an HttpException which simulates the scenario for a failed connection to one of the Dealership web services. You should be aware of this and handle this exception as best as you think how.

- ***DealershipData***
  This is the data that will be returned by the *DealershipService*. It includes the total sales and available stock for the dealership. Please see the file for full documentation on this.

The project also includes some utility files which will be required if you plan to do the optional additional requirements described further in this document. The classes are documented and are contained within the Utils/Threading folder of the project.

Requirements Specification

The requirements for your solution are detailed below. Candidates are expected to complete the *Required* section below. Candidates need not attempt or complete the *Optional* section - this section purely exists for Candidates who wish to further display their skills and their capability to understand and integrate existing code into their solutions. Candidates applying for more senior roles should consider the *Optional* section more seriously.

Required

Create an interface of your choosing (Console/Web/WPF).

Update the interface to allow a user to generate the required reports specified in the intro/description of this document. The reports can be output in a format of your choosing.

The reports should contain data based on all of the dealerships. i.e. You must have requested the data for every one of the dealerships contained within the *DealershipsList.txt* file.

Optional

Create an asynchronous/responsive interface that will allow the user to check on the progress of the report being generated. You must read and understand the *BackgroundWorkerQueue* that is included within the project, and then use this class as far as you possibly can in order to aid you in the creation of your solution.