Group Technology Applications

Candidate Code Test Specification (C#)

Introduction

Royal London relies on in-house developed, enterprise systems. Our developers need the skills, knowledge and experience to deliver robust, high quality solutions for these systems.

As with most software solutions, there isn't always one answer and this test is designed to help us understand how you approach a problem and enable us to identify the right applicant.

The test below is devised to assess your ability to convert a specification of modest complexity into a well-structured application.

The application does not need to be the finished article and we do not expect it to be super solid, highly scalable, polished or production ready.

You can spend as much time as you feel necessary, but please try to keep it simple yet sufficient, with a minimum of gold plating. We do not anticipate you will need any additional frameworks, external libraries, non-standard extensions or database connectivity for your application to work but if you do, please mention this in your documentation.

Once you have completed the application please zip it up together with the generated .xml output file, alternatively you are welcome to store the files on Github for us to download and assess.

Enclosed Files

Candidate Code Test (Specification).doc
MaturityData.csv

Characteristics

We will be looking for the following characteristics in your application:

- How the code is structured, does it resemble good design? E.g. functional, OO, procedural etc.
- Appropriate level of design
- Knowledge of .C#.net functions and best practices
- Design Patterns (only if appropriate, not just to impress)
- Correctness of the application
- Sensible level of comments
- Consistent coding standards
- Appropriate error checking (e.g. check for missing .csv file)
- Testability (feel free to include unit tests)

We are not looking for the following:

- A front end - we are not testing your HTML, CSS or JavaScript
- An over-engineered solution

Your Task

Write a small utility to value maturing policies for a fictional insurance company. It should generate an .xml file containing the policy number and maturity value for each of the policies contained in an input data file.

Together with this specification document, you will have received an input file we want your application to process (MaturityData.csv). The input file is structured as follows:

Field Name	Description
policy_number	Unique policy identifier. The first character determines the policy type. There are three valid policy types: A, B and C.
policy_start_date	The date the policy was taken out.
premiums	The total premiums the policyholder has paid over the lifetime of the policy.
membership	Whether the policy confers membership rights, Y=Yes, N=No.
discretionary_bonus	A one-off cash bonus should certain criteria be met.
uplift_percentage	The percentage bonus amount we apply to the policy value at maturity e.g. a value of 25 is a 25% uplift so on a £4,000 policy the maturity value would be £4,000 * 1.25 = £5,000

The rules for the three policy types are as follows:

Policy Type	Management Fee %	Discretionary Bonus Criteria
A	3	Policy was taken out before 01/01/1990
B	5	Policy has membership rights
C	7	Policy was taken out on or after 01/01/1990 and has membership rights

The basic calculation for the maturity value is:

((premiums – management fee) + discretionary bonus if qualifying) * uplift

Consider the example of a type ‘A’ policy with a start date of 01/06/1986, premiums of £10,000, a discretionary bonus of £1,000 and a percentage uplift of 40%.

This particular policy is of type ‘A’ therefore has a management fee of 3%. The policy qualifies for its discretionary bonus since the start date was before 01/01/1990.

The maturity value is:

((£10,000 – (£10,000 * 0.03) + £1000) * 1.4 = £14,980