Design Exercise

Description

A salesman is selling a set of products. Each product is described by a set of attributes, such as name, type, color, cost and weight, each of which may have a different data type (String, Boolean, Number).

A company is looking to buy products at the best possible prices, and which best match it's needs. It has many different products it is looking to purchase. It defines the products it wishes to buy with a set of rules. Each rule is defined by a set of conditions, and a score for if that rule matches. Each condition is made up of an attribute name, a value, and a comparison operator. Some rules might be negative, meaning that if the rule matches, the scoring should be negative. A rule might look like:

```
color == BLUE && price < 17.75 && quantity > 750 → 100
```

The company realizes that it is very time consuming and error prone to sort through the salesman's goods and is looking to implement a system that will:

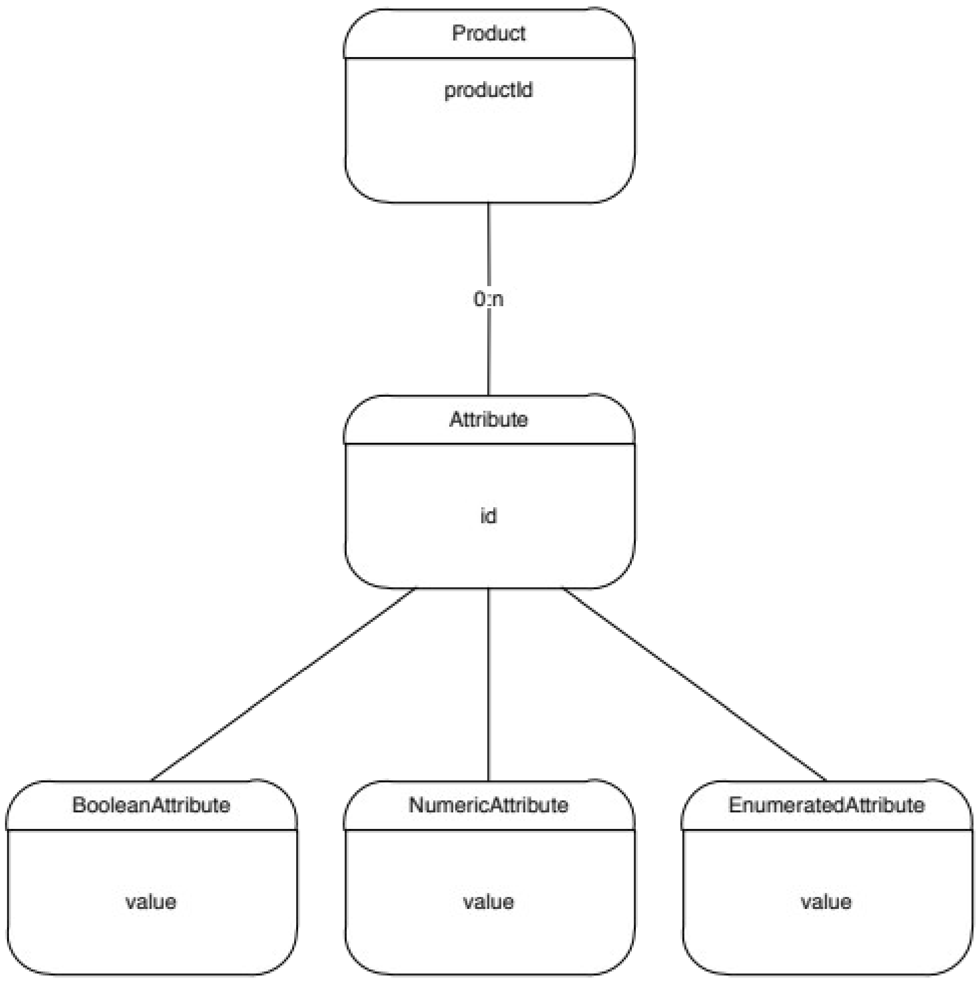
a) Score all of the salesman's products on how well they match their product definitions by calculating the sum of the rule scores, which is the percentage of conditions which match, multiplied by the score.
b) Filter the potential products to just those that pass a given threshold (assume 50% as the cutoff).
c) Calculate the total and average prices for all the products that score sufficiently highly.

Tasks

Your task is to design the system for the company, based on the requirements above.

- Draw a rough UML diagram showing the classes for the objects described above, and in particular rules and conditions. See the sample UML for product below.
- Write out, in code or psuedocode, a function that will calculate the scores, and the total and average prices for the products.
- If you make any assumptions, write them down.
- Make sure your solution is runnable from the command line via your tool of choice.
- The program must print out the total and average prices that score sufficiently highly when run from the command line.

NOTE: Perfect UML and perfect syntax are ***not*** expected. This is a starting point for a design discussion, not a test.


Product
productId
0:n
Attribute
id
BooleanAttribute
value
NumericAttribute
value
EnumeratedAttribute
value