Business Request:

You have been tasked by a stakeholder at Visio Financial Services to develop a solution that allows the business to dynamically generating product pricing from a set rules defined by the finance team. The finance team has given you an initial set of rules on how to price the products, however, these rules could change at any time so we need to be able to update the rules easily and rerun the product pricing to see the new prices of the products.

Initial Rules:

All products start at 5.0 interest_rate.

If the person lives in Florida (condition), we do not offer the product to them and the product is to be disqualified (action).

If the person has a credit score greater than or equal to 720(condition) then we reduce the interest_rate on the product by .3 (action that has an input of ".3", remember the business may decide in the future they want to reduce it by .5).

If the person has a credit score lower than 720 we increase the interest_rate on the product by .5.

If the name of the product is "7-1 ARM" then we need to add .5 to the interest_rate of the product.

Example:

```
class Person:
       credit_score: integer
       state: string

class Product:
       name:  string
       interest_rate: decimal
       disqualified: boolean

class RulesEngine
       def runRules(person, product, rules)

person = new Person(720, 'Florida')

product = new Product('7-1 ARM', 5.0)
```

```
rules_engine = new RulesEngine()

rules = loadRules() // Rules are loaded at runtime!

rules_engine.runRules(person, product, rules**);
```

Hint: We have not defined how the rules are represented, they could be json, csv, etc. The only thing that matters is that rule definitions can be extended upon and defined **outside of the code, the rule definitions should define an action, any parameters an action needs, and under what condition to execute the action.

The output of the above code example should be:

product.interest_rate == 5.2 (5.0 - .3 + .5)

product.disqualified == true

Expectations:

1. Build a rules engine to accomplish what the business has asked above.

2. Exhaustively test all possible rule outcomes and assert that the rules engine works as expected and meets the business requirements.

3. Show the extendibility of your rules engine and create your own rule definition to execute an additional action on a product.

4. Upload your solution to github giving instructions on how to run the rules engine and any tests.

5. Finally, be prepared to lead a review on the submitted code to discuss any patterns or design decisions you made.

If any point you need to ask a question about this problem please don't hesitate to email me at josh.reeter@visiolending.com.

Happy Coding!