Challenge Java candidates

New candidates applying for the Java engineer position in Forescout are required to work on a small development task. This document describes the goals the candidate is expected to achieve with a successful submission.

1 Assignment

The objective of this challenge is to develop a small library that models a real communication pattern. The fundamental entities in this context are *packets*, *networks*, *routing rules*. In a generic networking infrastructure, a device like a router receives communication packets from a network and takes decisions on how to forward them to another network upon certain routing rules.

The goal of this challenge is to implement a model for computing the best network affinity on routed packets. Ideally the library shall implement that part of the logic in a router in which the packet properties (addresses, ports, and protocols) are checked against a set of rules to decide in which network the packet must be forwarded to. A single destination network can be chosen, or, in case of no affinity, the packet is dropped.

This challenge provides a pre-packaged project to the candidate, defining an initial skeleton with the Java interfaces and basic functionalities. The candidates can usefully leverage on this skeleton to develop their own implementation. A test suite is also provided to check whether the requirements are fulfilled.

The model provided in the skeleton project defines only the properties relevant for this context. Properties out of this scope (e.g. the payload of the packet) are not defined.

2 Prerequisites

- JDK 8+
- Maven tool 3.3.9+
- Preferable a good Java IDE (IntelliJ, Netbeans or others)

3 Requirements

- The *Attribute* and *CollectionAttribute* abstract classes are the base classes for designing a single term of a matching rule like *packet.field operator argument*. These base classes define the properties holding the operator and the argument and implement the matching logic. The candidate is requested to design and implement specialized classes for computing the matching score of fields defined in the *IPacket* interface.
- An *IRoutingRule* implementation gathering all the attributes implementation above is requested. The concrete class must implement the method *getMatchingScore*() for calculating the matching score given the attribute and the packet to check. The matching score must return 0 when there is no matching and a positive value representing the score of the matching (the higher the value the better the matching). As an example, a packet having the source address 10.1.1.1 matches with both networks 10.1.0.0/16 and 10.1.1.0/24 but the score must be higher with the second network since the address space is tighter.

- An *IPacket* concrete implementation is required with the algorithm implementing the best affinity in the method *getClosestAffinityNetwork()*. Input arguments are the set of *IRoutingRule* and the attributes priority. The algorithm shall return the *IRoutingRule* from the input set that has the best affinity (matching score) evaluating the attributes in the assigned order. In case any attribute does not match the rule shall be discarded and when all the rules are mismatching the function must return *null*. In case two rules yield the same matching score over a packet the one having the lowest ID must be considered as a discriminating strategy.
- The concrete implementations above shall be used in the *Factory* class of the test package to prove the interface contracts and the algorithm fulfilment. The implementations shall be placed in the *com.forescout.challenge.impl* package and they should plug naturally in the Factory class without any further change of the latter.
- The project must compile with `mvn install` and the tests must run successfully with `mvn test`.

4 Bonus solution

Not requested but preferable for adding value to their submission the candidates can propose and/or implement an additional solution for which the score is computed on attributes in a compounded manner. That is, rather than evaluating a matching score for each single attribute it could be more interesting computing the matching score over the combination of (dstAddress, port) as a whole.