CS3152 Data Structures and Discrete Mathematics II

Project 2: Role-Based Access Control
Due: 11:55pm, Sunday, Oct 27, 2013

Role-Based Access Control

In the previous project you have been introduced to an access control model that stores rules for each user individually. As the number of users can be huge, *role-based access control (RBAC)* has been introduced to significantly reduce the number of rules. In role-based access control, each user is assigned one or more roles. All users with the same role have the same access rights. In this model all the rules are of the form (*rol, dat, act*), which can be interpreted as "all users with role *rol* have the access right *act* for the data object *dat*."

Assume now the request (`usr001, drug, write`) is made in an access control system. The PDP has now to determine the roles associated with the user `usr001` first. Assume the user is associated only with the role `nurse`. Then it has to check the rule set for a rule of the form (`nurse`, `drug`, `write`). Note that the same access policies as described in the previous project still apply. For example, if the policy DENY_OVERRIDES holds, then the PDP grants only access to the request (`usr001, drug, write`) if the rule (`nurse`, `drug`, `write`) exists, but not the rule (`nurse`, `drug`, `deny`).

If a user is associated with several roles and each role is granted access for a specific request by that user, then that access request should be granted. Similarly, if none of the roles is granted access for a request, then the access request should not be granted. But what happens if the user is associated with several roles and one role is denied access while another role is granted access for a specific request by that user? In that case, it depends on whether the policy DENY_OVERRIDES or PERMIT_OVERRIDES is specified. In case of PERMIT_OVERRIDES, the access is granted. In case of DENY_OVERRIDES, the access request is only granted if none of the applying rules is a deny rule. For example, assume the access control system contains the following rules:

```
nurse patientaddress read
admin patientaddress write
admin patientaddress read
```

Let `user001` be a user that has the role nurse and admin. If the request (`user001`, `patientaddress`, `write`) is made, then the role nurse does not grant write access to patientaddress, but the role admin does grant write access to patientaddress. As none of the roles nurse and admin have an explicit deny rule for patientaddress, the request is granted no matter if DENY_OVERRIDES or PERMIT_OVERRIDES holds. Now assume the rule

```
nurse patientaddress deny
```

is added to the rules above. Now the role nurse is explicitly denied access to patientaddress, while the role admin is still granted write access to patientaddress. If the policy PERMIT_OVERRIDES holds, then the request (`user001`, `patientaddress`, `write`) is granted. If the policy DENY_OVERRIDES, the request (`user001`, `patientaddress`, `write`) is not granted.

The Project

As in the previous project, you have to work in a team of two students. You are responsible for finding a partner for this project.

You will modify the PDP from the previous project to implement role-based access control. In addition, you will use hash tables to determine a user's role and to check for the existence of rules. The learning objectives are as follows:

- Implement a given interface.
- Use a hash table.
- Meet specifications that use propositional logic.
- Test and debug a program in an object-oriented programming language.

Implementation

Download the starter project RBAC.zip which is given as an eclipse project. The started project differs from the previous starter project in that it includes the additional interface UserRole.java and adds two methods to the interface PDP. In addition, it has a different sample file of rules and a sample file that maps roles to users.

- Implement a class called `RoleRule` that implements the interface Rule. Each object of class `RoleRule` represents a triple of the form (role, data, action). Two rules are equal if and only if the corresponding triples are equal. Add methods to the class as necessary. Hint: (1) You might be able to recycle your class UserRule from the previous project. Try out the refactoring tool in eclipse that helps to rename classes, variables, and arguments. (2) I recommend not only to implement the specified equals method, but also to override the method "**`public boolean`** `equals(Object obj)`." That will prevent some of the problems occurring in the previous project.
- Implement the interface UserRoles. Your class should store the roles assigned to each user in a hash table. You have to use a hash table to store the set of rules. Carefully decide on the key and value so that your implementation leverages the advantages of hash tables. You can use one of the available hash table implementations from the Java API.
- Implement a class called `RoleRuleSet` that implements the interface `RuleSet`. As in the case of `RoleRule` you might be able to recycle some parts of your solution for class `UserRuleSet`. You have to use a hash table to store the set of rules. Carefully decide on the key and value so that your implementation leverages the advantages of hash tables. You can use one of the available hash table implementations from the Java API.
- Implement the class `MyPDP` that implements the interface `PDP` or update the corresponding class from your last project.

Carefully test your solution. Make sure that all methods of your implementation meet the given specifications of the corresponding interface. You do not have to include your tests in the final submission as I will use my own test cases.

Submission

Submit a zip-file with your *s*ource code. Only one student in your team has to submit the project. Include the names of both team members in the name of your zip-file. If both team members submit a solution, then the most recently submitted solution will be graded.

Grading

Points will be assigned as follows:

- Implementation of interface `Rule: 10 points`
- Implementation of interface `RuleSet:` 30 points
- Implementation of the interface UserRoles: 30 points
- Implementation of interface PDP: 30 points

A program that cannot be compiled or executed will receive 0 points.