CS3152 Data Structures and Discrete Mathematics II
Project 2: Hierarchical Role-Based Access Control
Due: 11:55pm, Friday, Nov 22, 2013

Hierarchical Role-Based Access Control

In the previous project you have installed Role-Based Access Control (RBAC). In that model, roles were unrelated to each other, but in practice roles form a hierarchy. For example, a web development company might maintain the four roles team member, web designer, graphics designer, and project manager. All web designers and graphics designers are also team members. Therefore, they inherit all access rights that are associated with the role team member. The project manager inherits all access rights associated with the role web designer and graphics designer. And therefore, also inherits the access rights associated with the role team member. Thus the role hierarchy defines a relation on the set of roles where role r_1 relates to role r_2 if and only if role r_1 inherits all access rights of role r_2. This relation is transitive. However, the hierarchy is typically given by specifying the necessary edges only. See, for example, graph in Figure 1 that specifies a role hierarchy. Note that the displayed graph is not transitive. The transitive closure of the relation with the displayed graph specifies for each role from which roles the access rights are inherited.

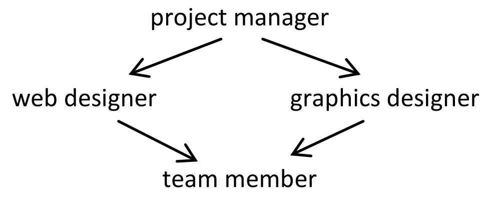

Figure 1: Example role hierarchy

How does the role hierarchy affect the process of granting access? Access is granted similarly as before for a request (`userid, data, access`). The policy decision point has not only to consider all roles that are directly assigned to the user with id `userid`, but it also has to include all roles from which the assigned roles inherit access rights. Once all roles—directly assigned and inherited roles—are determined, the process of granting access is the same as for role-based access control as described in Project 2.

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 given sample solutions or your solution for Project 2 to implement hierarchical role-based access control.

Implementation

Download the starter project HRBAC.zip which is given as an eclipse project or use your solution for Project 2. If you use your solution, make correction if necessary such that RBAC is implemented correctly. The given starter project is basically a sample solution for Project 2. The starter project includes the file hierarchy.txt that specifies a sample role hierarchy. The first line of a file defining a role hierarchy lists all possible roles separated by a space. Each of the following lines specifies a different role and its related roles. For example, the hierarchy in Figure 1 can be presented as follows in a file:

```
team_member web_designer graphics_designer project_manager
web_designer team_member
project_manager web_designer graphics_designer
graphics_designer team_member
```

Line 1 lists all four roles. Line 2, for example, defines that the role web_designer inherits from team_member. Line 3 specifies that project_manager inherits from web_designer and from graphics_designer, and so on.

Update the project so that it implements Hierarchical RBAC. Here are some issues to be considered before you get started

1. How is the role-hierarchy stored?
2. How are the roles determined that are inherited by a role?
3. When should the roles that are inherited by a role be determined? At the start of the program, whenever am access request is made, or at another point?

You may change the classes and interface and refactor the code as you deem suitable, with the following exception:

- Do not change the class `AccessRequest`.
- Keep the class `MyPDP` and its method `requestIsGranted(AccessRequest request)`. However, you can modify the other methods and method signatures and the implementation of method `requestIsGranted`.
- Add and implement the following constructor to class `MyPDP`:

```
/**
 * constructor - initializes this PDP
 *
 * @param ruleFile name of the file specifying the rules of this PDP
 * @param roleHierarchyFile name of the file specifying the role hierarchy of
 *        this PDP
 * @param userRolesFile name of the file specifying the roles of the users of
 *        this PDP
 * @param overridePolicy override policy of this PDP
 * @param readwritePolicy read-write policy of this PDP
 */
public MyPDP(String ruleFile, String roleHierarchyFile, String userRolesFile,
             String overridePolicy, String readwritePolicy) {
}
```

  (The constructor is already included in the starter project, but still needs to be implemented.)

If you do not follow these directions, then I will not be able to test you code and you will lose *at least* half of the possible total points.

Documentation

Write a document that describes

1. the main changes to your starter project.
2. the data structure that is used to store the role hierarchy and an explanation for your choice (e.g. contrast your choice with other options, detail how the data structure fits with performed the operations).
3. the algorithm that determines the roles that are inherited by a role and an explanation for your choice.
4. at which point the roles that are inherited by a role are determined and an explanation for your choice.

If you do not include a documentation, you will lose *at least* half of the possible total points as I will not be able to understand your design decisions.

Submission

Submit a zip-file with your source 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

Half of the points will be based on your data structure and algorithm choice(s) as described in the documentation and if you implemented these data structures and algorithms as described. The remaining half of the points are based on whether you correctly implemented the described algorithms and data structures and correctly implemented Hierarchical RBAC.