Eklavya Patel ehp35
Prathmesh Khot aap237
Adarsh Patel psk70

October 6th 2019

CS 419 Assignment 3 - Access Control System

How to Run the Program:

```
Python3 Driver.py
```

Driver.py is the program that calls every other method and therefore is the only file that needs to be run to run this program. It is important that all other files are in the same directory as Driver.py.
Running this Program will print all possible commands and the valid inputs for the 6 methods. The command “Exit” will exit the Access Control System.

Notes on Implementation:

Objects used in all methods are stored in the Objects.py file.
Objects Created:

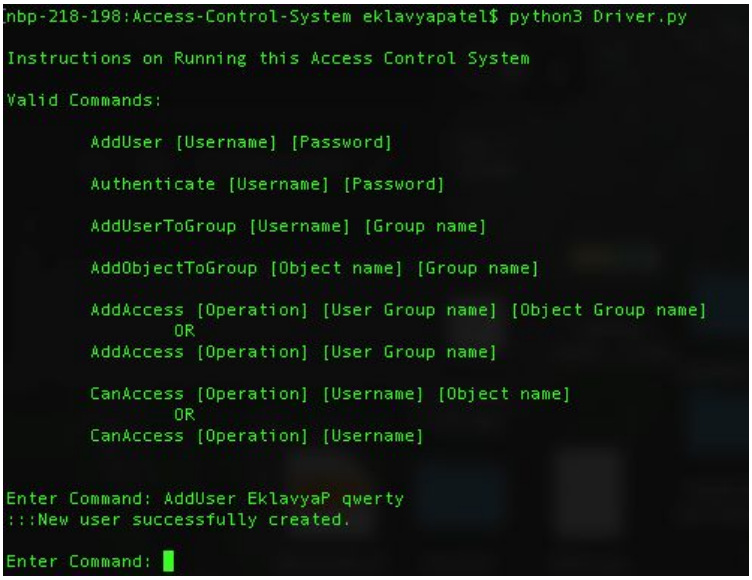
- User:
  The User class takes the username and password as an input. Along with this, it has an attribute to store a list all groups that a user is a part in order to make the overall implementation easier.
  This object is then stored in the global list “allUsers” for easy access.
- Group:
  The Group class stores the group name as an attribute. It also has two lists as attributes. One for all Users in this group and another for Objects in this group. Because of this, this class can be used for both User groups and Object groups. The functions, “makeMember” and “addObject” add additional users and objects to these groups.
  This object is then stored in the global list “allGroups” for easy access.
- Permissions:
  The purpose of this class is to store the list of operations possible in between User groups and Object groups. However, since some permissions aren’t relative to object groups, it only takes the user group name as a parameter and has an attribute to list the different operations. This object is stored in the global list “allPermissions”.

Usage of Each Program and Implementation:

- `AddUser` works by simply checking first if the chosen username is already listed in “allUsers,” a list that stores all the users and passwords. If it is already listed, then the user is alerted. If it is not listed, then the new user is successfully created.
- `Authenticate` works by simply looping through allUsers and checking if the user matches the password corresponding with it.
- `AddUserToGroup` first check if the user passed in the command is valid by checking the list “allUsers”. Next, it iterated over “allGroups” to check if the group has been created or not. If not, the group is created and the user is appended to the list of users in the Group Object. It then prints all the users that are currently members of this group.
- `AddObjectToGroup` it iterated over “allGroups” to check if the group has been created or not. If not, the group is created and the object is appended to the list of objects in the Group Object. It then prints all the objects that are currently members of this group.
- `AddAccess` works by looping through “allPermissions” to check if the access already exists, if not there is a new permission created. It will print if group does not exist accordingly.
- `CanAccess` works by looping through each user in the allUsers and checks what groups each user is a part of and for every group it checks what permissions exist in relation to the user. After that, it checks what object group the permission is associated with.

Test Cases :

- AddUser



- Authenticate

```
Enter Command: AddUser EklavyaP qwerty
:::New user successfully created.

Enter Command: Authenticate EklavyaP qwerty
:::successfully authenticated

Enter Command: Authenticate blah qwerty
:::Bad User

Enter Command: Authenticate EklavyaP blah
:::Bad Password

Enter Command:
```

- AddUserToGroup

```
Enter Command: AddUserToGroup EklavyaP premium_subscribers
:::Successfully added User to Group.
:::Current Users in premium_subscribers: ['EklavyaP']

Enter Command: AddUserToGroup PratK admin
:::FAILURE: User does not exist

Enter Command:
```

- AddObjectToGroup

```
Enter Command: AddObjectToGroup Netflix premium_content
:::Successfully added Object to Group.
:::Current Objects in premium_content: ['Netflix']

Enter Command: AddObjectToGroup Hulu premium_content
:::Successfully added Object to Group.
:::Current Objects in premium_content: ['Netflix', 'Hulu']

Enter Command:
```

- AddAccess

```
Enter Command: AddAccess view premium_subscribers premium_content
:::Permission to view successfully granted.

Enter Command: AddAccess edit premium_subscribers
:::Permission to edit successfully granted.

Enter Command:
```

- CanAccess


Enter Command: CanAccess view EklavyaP Hulu
::: Yes. EklavyaP can view Hulu
Enter Command: CanAccess edit EklavyaP Netflix
::: No. EklavyaP can not edit Netflix
Enter Command: CanAccess view PratK
::: No. PratK can not view