Language

ADVANCED COURSE IN PROGRAMMING, AUTUMN 2024, ONLINE EXAM 5

STARTED: 11:41 AM ENDS: 3:41 PM 3:58

Exercise 1

Implement the task in the file `exercise1.py`.

You can copy the following template to get started:

```
class User:
    def __init__(self, username):
        self.__username = username
        self.__checked_in = False
```

Program the `User` class with the following functionality:

- The method `__format` for formatting user actions with a timestamp. The method should return a string in the format `"time;username;action"`.
- The method `checkin` that takes time as a parameter. If the user is not already logged in, the method returns a formatted string using the `__format` method. Otherwise, the method raises an exception with an appropriate message.
- The method `checkout` that takes time as a parameter. If the user is logged in, the method returns a formatted string using the `__format` method. Otherwise, the method raises an exception with an appropriate message.
- The `__str__` method that returns the username as a string.

For the following input:

```
user = User("George")
print(user)
user.checkin(12)
user.checkout(18)
user.checkout(19)
```

The code should output:

```
George
ValueError: User has not been checked in
```

And for the input:

```
user = User("George")
```

```
print(user)
user.checkin(12)
user.checkin(12)
user.checkout(19)
user.checkout(19)
user.checkout(19)
```

The code should output:

```
George
ValueError: User is already checked in
```

Exercise 2

Implement the task in the file `exercise2.py`.

Use the `User` class from the previous task. Copy it into the file for exercise 2.

Program a class `UserManager` with the following functionality:

- `add_user`: Takes a username as a parameter. If the username is not already in use, the method adds the user to the system. Otherwise, an exception is raised.
- `check_in`: Takes a username and time as parameters. If the username exists, the user checks in. The method returns `True` if the login is successful, otherwise, returns `False`.
- `check_out`: Takes a username and time as parameters. If the username exists, the user checks out. The method returns `True` if the logout is successful, otherwise, returns `False`.

Example execution:

```
manager = UserManager()

manager.add_user("Tom")

print(manager.check_in("Tom", 12))
print(manager.check_out("Jerry", 12))
print(manager.check_out("Jerry", 12))

manager.add_user("Jerry")

print(manager.check_out("Tom", 18))
print(manager.check_in("Jerry", 11))
print(manager.check_out("Jerry", 19))

manager.add_user("Tom")

print(manager.check_in("Jerry", 11))
print(manager.check_out("Jerry", 19))
```

```
True
False
False
True
True
True
ValueError: User already exists
```

Exercise 3

Implement the task in the file `exercise3.py`.

Use the classes `User` and `UserManager` from the previous tasks. Copy them into the file for exercise 3.

Extend the `UserManager` class with three new methods:

- `load_log`: Reads the `logfile.csv` file line by line. Each line is in the format `"time;username;action"`. If a username in the file is not in the system, it is added. The method does not return anything.
- `save_log`: Opens the `logfile.csv` file and writes each login record from the system into the file in the format `"time;username;action"`. The method does not return anything.
- `__str__`: Returns all recorded events in the system as a string in the format `"time;username;action"`.

Create a new class `UserInterface`.

In the constructor, `UserInterface` is given a `UserManager` instance to manage data.

The class should provide the following functionalities:

- Create a user
- Check in a user
- Check out a user
- Show the entire system log
- Show a specific user's log

Example output:

```
0 - Exit program
1 - Check in
2 - Check out
3 - Add user
4 - View log for user
```

```
5 - View log
Choose action: 5
Log is empty
Choose action: 3
Username: Example
Choose action:
0 - Exit program
1 - Check in
2 - Check out
3 - Add user
4 - View log for user
5 - View log
Choose action: 20
0 - Exit program
1 - Check in
2 - Check out
3 - Add user
4 - View log for user
5 - View log
Choose action: 4
Username: Example
No logs for user Example
Choose action: 2
Username: Example
Time: 12
User has not been checked in
Choose action: 1
Username: Exemplary
Time: 12
User does not exist
Choose action: 2
Username: Exemplary
Time: 12
User does not exist
```

```
Choose action: 4
Username: Exemplary
No logs for user Exemplary
Choose action: 2
Username: Example
Time: 12
User has not been checked in
Choose action: 1
Username: Example
Time: 12
Checked in
Choose action: 2
Username: Example
Time: 20
Checked out
Choose action: 4
Username: Example
12;Example;Check in
20;Example;Check out
Choose action: 3
Username: Exemplary
Choose action: 1
Username: Exemplary
Time: 13
Checked in
Choose action: 5
12;Example;Check in
20;Example;Check out
13;Exemplary;Check in
Choose action: 0
Closing program...
```

After restarting the program:

```
0 - Exit program
1 - Check in
2 - Check out
3 - Add user
4 - View log for user
5 - View log
Choose action: 5
12;Example;Check in
20;Example;Check out
13;Exemplary;Check in
Choose action: 0
Closing program...
```

The log remains saved even after the program execution ends.

Note, that in the case of an error the `UserInterface` should handle the error properly, so the execution won't stop.

END EXAM

About

The University of Helsinki MOOC Center makes high-quality online education possible by developing and researching educational software and online learning materials. Teachers both within and without the University of Helsinki rely on our tools to create impactful teaching materials. Our popular Massive Open Online Courses (MOOCs) have been available through MOOC.fi since 2012.

This website is powered by an open source software developed by the University of Helsinki MOOC Center. Star the project on GitHub: **Project Github**.

Privacy