Einführung in die Programmierung

Prof. Dr. Peter Thiemann
Marius Weidner, Hannes Saffrich
Lukas Kleinert, Timpe Hörig

Universität Freiburg
Institut für Informatik
Wintersemester 2023

Übungsblatt 9

Abgabe: Montag, 18.12.2023, 9:00 Uhr morgens

Aufgabe 9.1 (Invarianten; 8 Punkte; Datei: `users.py`)

In dieser Aufgabe sollen Sie Datenklassen schreiben, die mehrere Invarianten einhalten. In Ihrem git-Repo befindet sich eine Testdatei `test_users.py`, mit der Sie ihre Abgabe testen können. Achten Sie darauf, dass Namen von internen Attributen mit `__` beginnen, auch wenn diese in der Aufgabenstellung nicht so genannt werden. Denken Sie an die 3 Regeln für Invarianten aus der Vorlesung!

(a) **User; 4 Punkte**
Schreiben Sie eine Datenklasse `User` mit den drei Attributen `name` (String), `age` (Ganzzahl) und `password` (String). Folgende Invarianten sollen gelten:

- `name` muss mindestens zwei Zeichen lang sein
- `age` muss mindestens 13 sein
- `password` muss mindestens 6 Zeichen lang sein, mindestens einen Großbuchstaben und mindestens 2 Ziffern (0-9) besitzen[1]

Die Attribute `name` und `age` sollen read-only property-Attribute (mit Getter, aber ohne Setter) sein. Das Attribut `password` soll *keinen* Getter besitzen und nur intern zugreifbar sein. Fügen Sie stattdessen eine Methode `login` hinzu, die einen String `p` als Argument nimmt und genau dann `True` zurückgibt, wenn `p` mit dem internen Passwort übereinstimmt. Überladen Sie in der Klasse den Operator `==`, der genau dann True zurückgibt, wenn die Namen der beiden Operanden gleich sind.

(b) **Network; 4 Punkte**
Schreiben Sie eine Datenklasse `Network`. Der Konstruktor soll keine Argumente haben, aber ein internes Attribut `users` anlegen, das eine Liste von `User` darstellt, die dem Netzwerk beigetreten sind. Auf `users` soll von außen nicht direkt zugreifbar sein. Die Klasse soll eine Methode `add_user` besitzen, die einen User `new_user` als Argument nimmt und versucht, diesen zu `users` hinzuzufügen. Existiert bereits ein User mit dem gleichen Namen, soll `False` zurückgegeben werden. Ansonsten soll der neue User hinzugefügt und `True` zurückgegeben werden.
Fügen Sie der Klasse zudem eine Methode `update_user` hinzu, die zwei Strings `name`, `password` und einen User `new_user` als Argumente nimmt und einen

[1] Sie dürfen hierfür `ascii_uppercase` und `digits` aus dem Modul `string` der Standardlibrary verwenden: https://docs.python.org/3/library/string.html

Wahrheitswert zurückgibt. Die Methode soll nach einem Benutzer mit entsprechendem Namen und Passwort suchen. Falls kein solcher User existiert, oder `name` ungleich `new_user.name` ist, soll `False` zurückgegeben werden. Sonst soll das entsprechende Objekt aus `users` entfernt werden[2], das neue Objekt `new_user` in die Liste eingefügt werden und `True` zurückgegeben werden.

Aufgabe 9.2 (Datenkapselung und dunder-Methoden; 8 Punkte; Datei: `times.py`)
In dieser Aufgabe sollen Sie eine Datenklasse schreiben, die das Prinzip der *Datenkapselung* verwendet und zusätzlich einige *dunder-Methoden* besitzt. In Ihrem git-Repo befindet sich eine Testdatei `test_times.py`, mit der Sie ihre Abgabe testen können.

(a) **Datenklasse und properties; 3 Punkte**
Schreiben Sie eine Datenklasse `Time`, die eine Uhrzeit bzw. relative Zeitangabe im 24-Stunden-Format beschreibt. Die Klasse soll als äußeres Interface die beiden ganzzahligen property-Attribute `hours` (Stunden) und `minutes` (Minuten) mit gettern und settern besitzen. Die interne Repräsentation soll jedoch nur aus einer einzigen Ganzzahl `__time` bestehen, die die gesamte Zeit in Minuten darstellt. Achten Sie bei der Umwandlung darauf, dass die interne Repräsentation jeweils zyklisch auf die Stunden $\{0, \ldots, 23\}$ bzw. Minuten $\{0, \ldots, 59\}$ abgebildet wird.[3]

```
t1 = Time(12, 23)
assert t1.hours == 12 and t1.minutes == 23
t2 = Time(-13, 61)
assert t2.hours == 12 and t2.minutes == 1
```

(b) **Vergleichsoperatoren; 2.5 Punkte**
Überladen Sie die Vergleichsoperatoren `==`, `>`, `<`, `>=` und `<=` für jeweils zwei Objekte der Klasse `Time`. Vergleichen Sie dabei zuerst die Stundenzahl. Ist die Stundenzahl der Operanden gleich, muss die Minutenzahl verglichen werden.

(c) **Arithmetik; 1.5 Punkte**
Überladen Sie die arithmetischen Operatoren `+` und `-`, die jeweils zwei Zeitangaben addieren bzw. voneinander abziehen.

```
assert Time(12, 34) + Time(12, 32) == Time(1, 6)
assert Time(10, 34) - Time(12, 44) == Time(21, 50)
```

(d) **String-Repräsentation; 1 Punkte**
Fügen Sie der Klasse die dunder-Methode `__str__`[4] hinzu, die keine Parameter außer self hat und eine String-Repräsentation des `Time`-Objekts der Form `"hh:mm"` zurückgibt, wobei `h` für eine Stunden-Ziffer und `m` für eine Minuten-Ziffer steht.

[2] Sie können die `list`-Funktion `remove` dafür nur verwenden, wenn der `==`-Operator für `User` definiert ist! properties werden in der von `dataclass` automatisch generierten `__eq__` Methode nämlich nicht berücksichtigt.

[3] An die Klasse übergebene Argumente können beliebig groß oder klein sein. Orientieren Sie sich ggf. an den Tests, um zu verstehen, was bei der Unter-/Überschreitung der Intervalle passiert.

[4] Diese Methode ermöglicht die Umwandlung mit der Funktion `str()` und die Ausgabe mit `print`

Aufgabe 9.3 (Dictionaries; 4 Punkte; Datei: `dicts.py`)

(a) `find_age`; 2 Punkte
Schreiben Sie eine Funktion `find_age`, die ein Dictionary `d`, welches Personen (String) ihr Alter (Ganzzahl) zuweist, und eine Ganzzahl `age` als Argumente nimmt. Die Funktion soll eine Liste von allen Personen zurückgeben, die das Alter `age` haben.

(b) `same_age`; 2 Punkte
Schreiben Sie eine Funktion `same_age`, die ein Dictionary wie in a) als Argument nimmt und `True` zurückgibt, wenn mindestens zwei Personen in `d` das gleiche Alter haben. Ansonsten wird `False` zurückgegeben

Aufgabe 9.4 (Erfahrungen; 0 Punkte; Datei: `NOTES.md`)

Notieren Sie Ihre Erfahrungen mit diesem Übungsblatt (benötigter Zeitaufwand, Probleme, Bezug zur Vorlesung, Interessantes, etc.).

Editieren Sie hierzu die Datei `NOTES.md` im Abgabeordner dieses Übungsblattes auf unserer Webplatform. Halten Sie sich an das dort vorgegebene Format, da wir den Zeitbedarf mit einem Python-Skript automatisch statistisch auswerten. Die Zeitangabe `3.5 h` steht dabei für 3 Stunden 30 Minuten.