Danny Rehl, Programmieren I - Python

Übungsblatt 10

Abgabefrist: Montag, den 20.01.2025, um 23:59 Uhr

Sie finden für dieses Übungsblatt dieses Mal *keine* gesonderte Vorlage für Ihre Abgabe zum Herunterladen im Moodle. Schreiben Sie die `.py`-Datei von Grund auf selbst.

Bitte achten Sie darauf, dass Ihr Code PEP 8 konform geschrieben ist. Das ist Voraussetzung für die abschließende Bewertung. Nutzen Sie die Hinweise des Pylint-Scorers, um die verbleibenden Style-Fehler zu finden.

Aufgabe 1 (2 Punkte)

Schreiben Sie eine Klasse `Knight`. Beim Instanziieren sollen einige Parameter in folgender Reihenfolge übergeben werden können:

- `full_name` vom Typ `str`
- `special_ability` vom Typ `str`
- `skill` vom Typ `int` oder `float`
- `round_table` vom Typ `bool` mit Standardwert `True`

Diese Parameter sollen in `__init__` als Instanzvariablen im Objekt abgelegt werden. Außerdem soll bei der Instanziierung eine Instanzvariable `health` vom Typ `int` mit Standardwert `100` angelegt werden, für die es keinen Parameter gibt.

Achtung: Die Namen der Attribute müssen dabei eventuell von den oben genannten Bezeichnungen der Parameter abweichen (z. B. durch Hinzufügen von Unterstrichen), denn sie sollen folgende Anforderungen erfüllen:

- `full_name` soll öffentlich sein (“public”)
- `special_ability` soll geschützt sein (“protected”)
- `skill` soll öffentlich sein (“public”)
- `round_table` soll öffentlich sein (“public”)
- `health` soll versteckt sein (“private”)

Zusätzlich soll die Klasse eine Klassenvariable `armor` vom Typ `bool` mit Wert `True` besitzen.

Die Klasse soll beispielsweise so benutzt werden können:

```
bedevere = Knight('Sir Bedevere the Wise', 'wisdom', 60)
lancelot = Knight('Sir Lancelot the Brave', 'bravery', 55)
black_knight = Knight('Black Knight', 'confidence', 100, False)
```

Aufgabe 2 (2 Punkte)

Bisher ist der Gesundheitszustand in einer versteckten (“private”) Instanzvariablen gespeichert.

Fügen Sie `health` als Eigenschaft (“property”) der Klasse `Knight` hinzu. Benutzen Sie `@property` um einen Getter zu schreiben, welcher den aktuellen Wert der Eigenschaft (gespeichert im Attribut der vorhergehenden Aufgabe) zurückgibt.

Passend dazu sollen Sie einen Setter schreiben, welcher den Wert der Eigenschaft ändert. Sollte der Wert jedoch kleiner oder gleich 0 sein, soll folgende Exception ausgelöst werden:

```
ValueError("Death is not an option.")
```

Tipp: Es gibt ein Schlüsselwort, mit dem Sie eine Exception auslösen können.

Aufgabe 3 (2 Punkte)

Schreiben Sie eine magische Methode ("magic method") `__repr__` für die Klasse `Knight`. Der Rückgabewert der Funktion soll vom Typ `str` sein und über den Gesundheitszustand des Ritters informieren. Das Ergebnis soll die aktuellen Werte von `full_name` und `health` enthalten.

Eine Ausgabe könnte beispielsweise so aussehen:

```
>>> print(bedevere)
Sir Bedevere the Wise is at 100% health.
```

Aufgabe 4 (2 Punkte)

Schreiben Sie eine Methode `duel` für die Klasse `Knight`. Es soll ein anderes Objekt der Klasse `Knight` übergeben werden können, sodass sich die beiden Ritter duellieren und der Ausgang des Duells per `print` ausgegeben wird.

Falls beide Ritter der Tafelrunde angehören (Attribut `round_table`), soll mit `print` folgender Hinweis ausgegeben werden:

```
Knights of the Round Table don't fight each other.
```

und es passiert nichts weiter.

Gehört jedoch mindestens einer der Ritter nicht der Tafelrunde an, kommt es zum Duell. Es gewinnt der Ritter mit dem höheren Wert des Attributs `skill`. Bei Gleichstand des Attributs gewinnt der Ritter, welcher das Duell nicht ausgelöst hat (sondern per Parameter übergeben wurde). Der angreifende Ritter verliert.

Geben Sie Gewinner und Verlierer als Hinweis per `print` aus. Außerdem bekommt der Verlierer 20 Prozentpunkte von seinem Gesundheitsstatus abgezogen. (Der Ritter startet mit 100, verliert ein Duell, hat dann 80, dann 60 und so weiter.)

Eine Ausgabe könnte beispielsweise so aussehen:

```
>>> bedevere = Knight('Sir Bedevere the Wise', 'wisdom', 55)
>>> lancelot = Knight('Sir Lancelot the Brave', 'bravery', 60)
>>> black_knight = Knight('Black Knight', 'confidence', 100, False)
>>> knight_of_ni = Knight('Knight Who Says Ni', 'ni', 55, False)
>>> bedevere.duel(lancelot)
Knights of the Round Table don't fight each other.
>>> print(bedevere)
Sir Bedevere the Wise is at 100% health.
>>> bedevere.duel(black_knight)
Black Knight wins, Sir Bedevere the Wise loses.
>>> print(bedevere)
Sir Bedevere the Wise is at 80% health.
>>> bedevere.duel(knight_of_ni)
Knight Who Says Ni wins, Sir Bedevere the Wise loses.
>>> print(bedevere)
Sir Bedevere the Wise is at 60% health.
>>>
```

Aufgabe 5 (2 Punkte)

Schreiben Sie eine zusätzliche Klasse `King`, welche von `Knight` erbt. Überschreiben Sie die Methode `__init__`, sodass `King` beim Instanziieren nur noch den Parameter `full_name` erwartet. Innerhalb der Methode rufen Sie per `super` die `__init__`-Methode der Oberklasse `Knight` mit folgenden Parametern auf:

- `full_name` wird einfach weitergegeben
- `special_ability` soll `'aristocracy'` sein
- `skill` soll `float('inf')` sein
- `round_table` soll `True` sein

Der Ausdruck `float('inf')` steht für "positiv unendlich".

Überschreiben Sie außerdem die Methode `__repr__`, sodass sie zwar noch den Namen ausgibt, aber keine Informationen mehr über den Gesundheitszustand enthält.

Eine Ausgabe könnte beispielsweise so aussehen:

```
>>> arthur = King('King Arthur')
>>> print(arthur)
The health of King Arthur is none of you business.
```