Informatik für Schüler, Foliensatz 23 Konstruktor, String-Methode und Heldenklassse

Prof. G. Kemnitz

Institut für Informatik, Technische Universität Clausthal

25. Mai 2009

Grundbegriffe der objektorientierten Programmierung

```
class Klassenname():
 {Attribut = Wert}
 {def Methode(self {,arg}):
    Anweisung
    {Anweisung}}

Objektname = Klassenname()
Objektname.Methode(Argumente_außer_self)
a = Objektname.Attribut
```

Was ist:

- ein Attribute, eine Methode, eine Klasse
- ein Objekt
- das Argument »self«?

Entwurf einer Klasse »Konto«: Was ist zu tun?

gegeben:

- Attribute: Name des Inhabers, Kontonummer, Kreditrahmen, Guthaben
- Methoden
  - Konto einrichten (Startdaten anlegen)
  - Einzahlen
  - Geld abheben
  - Kontostand anzeigen

Testbeispiel:

- Konto anlegen (Nummer: 1122; Name: »Gustav; Kreditrahmen: 100 (Eur); Startguthaben: 0 (Eur))
- Einzahlen von 200 (Eur)
- Abheben von 250 (Eur)
- Kontostand anzeigen

Spezielle Methoden

- Konstruktor

  ```
  __init__(self {,arg}): ...
  ```

  Wird aufgerufen, wenn ein Objekt erzeugt wird; z.B.

  ```
  class Konto():
   def __init__(self, name, KontNr, Kredit):
    self.Name=Name;
    self.KontNr=KontNr
    self.Kredit=Kredit
    self.Guthaben=0
  ```

  - Rückgabewert ist das Objekt selbst
  - Python-Objekte haben eine Wörterbuch, in das eine Methode neue Attribute eintragen kann

```
Konto_Gustav = Konto('Gustav', 1122, 100)
```

- Umwandlung in eine Textdarstellung

  ```
  __str__(self)
  ```

  Wird automatisch bei »print obj« und »str(obj)« aufgerufen

  ```
  class Konto():
   __str__(self):
    z  = 'Name: ' + self.Name +'\n'
    z += ' Konto: ' + str(self.KontNr)+'\n'
    z += ' Guthaben: ' + str(self.Guthaben)
    return z
  ```

- Aufruf:

  ```
  Konto_Gustav = Konto('Gustav', 1122, 100)
  print Konto_Gustav
  ```

Aufgabe 23.1: Heldenklasse

Schreiben Sie für ein Computerspiel eine Klasse »Held«. Ein Held hat einen Namen, eine Anzahl von Lebenspunkten und ein Körpergewicht. Wenn ein neuer Held erzeugt wird, werden ihm im Konstruktor alle drei Attribute übergeben. Die »__str__«-Methode soll eine Textdarstellung aller Attribute zurückgeben. Außer dem Konstruktor und der String-Methode soll es für einen Helden drei weitere Methoden geben:

- Nahrungsaufnahme:

  ```
  Essen(self, Essenmenge)
  ```

  Gewichtsvergrößerung um die Essensmenge.

- Angriff auf einen anderen Held:

  ```
  Angriff(self, Feind)
  ```

  der andere Held verliert mit einer Wahrscheinlichkeit[1]

  $$10\% \cdot \frac{m_a}{m_a + m_f}$$

  (m_a – Gewicht Angreifer; m_f – Gewicht Feind) einen Lebenspunkt.
- Für eine bestimmte Zeit nichts tun:

  ```
  Ruhe(self, t)
  ```

  der Held verliert nach der Funktion

  $$m \cdot \exp\left(-\frac{t}{100}\right)$$

  (t – Zeit) Gewicht und vergrößert seine Anzahl von Lebenspunkten um $\text{int}(z \cdot t/100)$ (z – Zufallszahl zwischen null und eins).

[1] Die Spielregeln (die Berechnungsformeln in den Methoden) dürfen nach eigenem Ermessen verändert werden.

Aufgabe 23.2: Dialogspiel

Schreiben Sie ein Computerspiel, in dem dialoggesteuert Helden erzeugt, gefüttert und zum Angriff geschickt werden.
Dialogschema pro Spielzug:

- Frage nach der Spielzeit
- Frage nach der durchzuführenden Aktion (Held erzeugen, Held füttern, Angriff)
- Frage nach den Parametern zur Aktion (welcher Held etc.)

Für jeden Spielzug:

- Zeit seit letztem Spielzug bestimmen und alle Helden ruhen lassen
- Spielzug ausführen
- Helden, die ihre Lebenspunkte verloren haben, in die Ahnenliste verschieben
- Spielzustand (Daten aller Helden) anzeigen

Hinweise:

- Die Ruhezeit ist die Differenz der Zeiteingabe zu Beginn eines jeden Spielzuges zu der des vorherigen Spielzugs; als Zeiteingabe genügt ein ganzzahliger Wert ohne Maßeinheit, der größer als vorherige Wert ist
- Um für alle Helden in einer Schleife die Methoden »Ruhe« und »__str__« ausführen zu können, sind die Held-Objekte in einer Liste zu verwalten.
- Die Ahnenliste ist zum Spielbeginn leer und übernimmt im Spielverlauf alle Held-Objekte, die ihre Lebenspunkte verloren haben.

Aufgabe 23.3: Dialogspiel-X

Vorschläge für weitere Aufgaben:

- Denken Sie sich eigene (bessere) Spielregeln aus und entwickeln Sie vergleichbare Programme.
- Teilen Sie z.B. die Helden in Freunde und Feinde und überlassen sie die Steuerung der Feinde dem Zufallsgenerator.
- Ergänzen Sie eine Funktion zum Sichern des Spielstands in eine Datei und eine Funktion zum Laden eines gespeicherten Spielstandes