Grundlagen der Programmierung 1

Prof. Dr. Detlef Krömker
Alexander Wolodkin

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Übungsblatt Nr. 6

Ausgabe: 22.11.2017
Abgabe: 01.12.2017
9.30 Uhr

Datentypen und Objektorientierung

Hinweis: *Es sind grundsätzlich Rechenwege anzugeben, es sei denn es findet sich ein expliziter Hinweis, dass dies nicht nötig ist. Es dürfen keine Lösungen aus dem Skript, dem Internet oder anderen Quellen abgeschrieben werden. Diese Quellen dürfen nur mit Quellenangaben verwendet werden und es muss ein hinreichend großer Eigenanteil in den Lösungen deutlich zu erkennen sein. Digitale Abgaben, die nicht im Format .pdf für Texte oder .py für Code erfolgen, werden nicht bewertet. Bei Abgaben mehrer Dateien müssen diese als .zip zusammengefasst werden. Abgaben, die nicht diesen Regeln entsprechen, werden nicht bewertet! Achten Sie darauf die Variable* `__author__` *in allen Quellcode Dateien korrekt zu setzen. Abgaben, die nicht dieser Vorgabe entsprechen, werden nicht bewertet. Außerdem muss Ihr Name in jeder abgegebenen .pdf Datei zu finden sein. Ababen, die vollständig per Hand geschrieben und eingescannt werden, sind nur in zuvor abgesprochenen Ausnahmefällen erlaubt.*

Beachten Sie das Programmierhandbuch, *für Aufgaben ab der Quartalwoche 47 ist die Verwendung des Headers verpflichtend. Abgaben, die nicht dieser Vorgabe entsprechen, werden nicht bewertet.*

Σ __ / **9**

Aufgabe 6.1: Datentypen und Funktionen

Punkte: ____ / 3

(a) (1 Punkt) Schreiben Sie eine Funktion *create_matrix*(), welche eine aus zufälligen ganzen Zahlen bestehende *n x m Matrix* generiert und diese in einem Dictionary **zellenweise** speichert.

Hinweis: *Verwenden als Key die Indizes der jeweiligen Zelle und als Value die Zellenwerte.*

Die Werte n und m sind als Funktionsparameter zu übergeben und dürfen keine Werte unter 2 annehmen. Ungültig gewählte Parameter n oder m sind dem Benutzer mitzuteilen.

Stellen Sie sicher, dass in der Matrix mindestens eine ein-, zwei-, drei- und vierstellige Zahl vorkommt.

Geben sie die so entstandene Matrix zeilenweise aus. Die Zahlen sind durch ein Leerzeichen zu trennen. Z.B.:
$(z_{11}\ z_{12}\ z_{13})$
$(z_{21}\ z_{22}\ z_{23})$

Hinweis: *Es sind die Zellenwerte/Values auszugeben. Die verwendeten Keys müssen nicht ausgegeben werden.*

(b) (1 Punkt) Ergänzen Sie Ihr Programm. Schreiben Sie dazu eine Funktion *format_matrix*(), welche die von der Funktion *create_matrix*() übergebene Matrix rechtsbündig formatiert und geben Sie diese erneut aus, z.B.

```
(  1   2    3)
(11 222 3333)
```

Bedingung: Die Anzahl der aufzufüllenden Stellen ist dynamisch anhand der übergebenen Zellenwerte spaltenweise festzustellen.

(c) (1 Punkt) Ergänzen Sie Ihr Programm. Schreiben Sie dazu eine Funktion *transpose_matrix*(), welche die von *format_matrix*() übergebene formatierte Matrix transponiert ausgibt.

Aufgabe 6.2: **OOP und Zugriffskontrolle** Punkt: ____ / 1

Erklären Sie in eigenen Worten folgende OOP-Begriffe kurz und prägnant in jeweils bis zu 4 Sätzen.

- Konstruktor
- Destruktor

Erklären Sie in eigenen Worten folgende Begriffe der Zugriffskontrolle im Kontext der Veranstaltung:

- public
- private
- protected

Aufgabe 6.3: **Objektorientierung** Punkte: ____ / 5

1. Erstellen Sie ein Modul, welches eine Klasse *Bike* beinhaltet.

(a) (1 Punkt) Definieren Sie mindestens 4 Attribute, darunter *mileage* und *color*.

Implementieren Sie weiterhin drei Klassenmethoden *set_color()*, *show_color()* und *inc_miles()* mit einer geeigneten Ausgabe in der Konsole. Entscheiden Sie sich für eine geeignete Parametrisierung!

(b) (2 Punkte) Implementieren Sie eine main-Funktion, welche 2 Fahrräder anlegt. Geben Sie anschließend unterschiedliche Farben für die beiden Fahrräder über die Methode set_color() an.

Lassen Sie weiterhin beide Fahrräder mittels mehrfacher Ausführung von inc_miles() unterschiedliche Entfernungen zurücklegen. **Dokumentieren Sie Ihre Entscheidungen zur Auswahl einer geeigneten Parametrisierung und veranschaulichen Sie die Ausgaben Ihres Programms mit Bildausschnitten in einer beizufügenden PDF.**

2. Erstellen Sie eine weitere Klasse Ebike, welche von der Klasse Bike erbt und sie um weitere Attribute *curr_capacity* und *consumption* erweitert.

(c) (1 Punkt) Diese zwei Attribute sind beim Anlegen eines Objektes parametrisiert mit Hilfe eines Konstruktors festzulegen. Entscheiden Sie sich für eine geeignete Parametrisierung!

Implementieren Sie weiterhin eine Klassenmethode *show_reach()*, welche basierend auf curr_capacity und consumption die verbleibende Reichweite berechnet und sie ausgibt.

(d) (1 Punkt) Implementieren Sie eine main-Funktion, welche 2 E-Bikes anlegt. Geben Sie ihre verbleibende Reichweite aus. Mit Hilfe der built-in Funktion *help()* zeigen Sie in Ihrer Dokumentation am Beispiel von einem E-Bike, dass das generierte Objekt tatsächlich von der Klasse Bike geerbt hat.

Setzen Sie ihre Dokumentation fort. **Dokumentieren Sie Ihre Entscheidungen zur Auswahl einer geeigneten Parametrisierung und veranschaulichen Sie die Ausgaben Ihres Programms mit Bildausschnitten in einer beizufügenden PDF.**