Informatik I: Einführung in die Programmierung

Prof. Dr. Peter Thiemann
Hannes Saffrich
Wintersemester 2020

Universität Freiburg
Institut für Informatik

Übungsblatt 8

Abgabe: Montag, 11.01.2021, 9:00 Uhr morgens, über git[1]

Hinweis

Aufgabenteile werden mit **0 Punkten** bewertet wenn:

- Dateien und Definitionen nicht so benannt sind, wie im Aufgabentext gefordert;
- Dateien falsche Formate haben, z.B. PDF statt plaintext; oder
- Pythonskripte wegen eines Syntaxfehlers nicht ausführbar sind.

Es gibt **Punktabzug** wenn:

- Funktionen keine oder falsche Typannotationen aufweisen. Ausnahme: Bei Funktionen, die stets `None` zurückgeben, kann der Rückgabetyp weggelassen werden.

Gruppenaufgaben müssen von allen Mitgliedern abgegeben werden und in der ersten Zeile müssen die Mitglieder in einem Kommentar vermerkt werden, z.B:

```
# Gruppe: xy123, yz56, zx934
```

Aufgabe 8.1 (Vererbung; Datei: `fleet.py`; Punkte: 8 = 3+3+2)

Für einen Fuhrpark bestehend aus PKWs, LKWs, Bussen und Fahrrädern soll eine Klassenhierarchie entworfen werden.

Dabei sollen die unterschiedlichen Fahrzeuge sowohl gemeinsame als auch unterschiedliche Attribute besitzen, die jeweils durch einen `int` beschrieben werden:

- Jedes `Fahrzeug` besitzt ein Leergewicht (Kilogramm).
- Jedes `Kraftfahrzeug` besitzt eine Leistung (Kilowatt).
- Zu jedem `Bus` gehören die Angaben: Baujahr, sowie Anzahl der Sitz- und Stehplätze.
- Zu jedem `Fahrrad` gehören die Angaben: Baujahr und Rahmengröße (cm).
- Zu jedem `PKW` gehören die Angaben: Baujahr, sowie Anzahl der Sitzplätze.
- Zu jedem `LKW` gehören die Angaben: Baujahr, Anzahl Sitzplätze, sowie Zuladung (Kilogramm).

Vermeiden Sie - soweit wie möglich - Wiederholungen in den folgenden Aufgabenteilen:

[1] `https://inpro.informatik.uni-freiburg.de/`

(a) Implementieren Sie die Klassenhierarchie wie oben angegeben. Machen Sie dabei Gebrauch von Vererbung und verwenden Sie Datenklassen wie in der Vorlesung beschrieben. Geben Sie die Attribute dabei in folgender Reihenfolge an (sofern die Attribute in den jeweiligen Klassen vorhanden sind): Leergewicht, Baujahr, Leistung, Sitzplätze, Stehplätze, Rahmengröße, Zuladung

(b) Implementieren Sie die Methoden `show`, `show_type` und `show_data`, die es erlauben Instanzen von jedem Fahrzeugtyp wie folgt in einen String umzuwandeln. `show_type` gibt den Namen des jeweiligen Fahrzeugtyps zurück, `show_data` listet die Attribute des jeweiligen Fahrzeugtyps auf und `show` listet den Namen und die Attribute des jeweiligen Fahrzeugtyps auf.

Verwenden Sie dabei Vererbung, sodass `show` nur ein einziges mal implementiert werden muss und die Implementierungen von `show_data` für jede Subklasse nur die Attribute konvertieren müssen, die in der Subklasse neu hinzugekommen sind, und für die anderen Attribute auf die Implementierung der Superklasse zurückgreifen.

Die Ausgabe soll dabei *exakt* den folgenden Beispielen entsprechen:

```
>>> rad = Fahrrad(10, 2019, 55)
>>> print(rad.show_type())
Fahrrad
>>> print(rad.show_data())
Leergewicht: 10kg Baujahr: 2019 Rahmengroesse: 55cm
>>> print(rad.show())
Fahrrad Leergewicht: 10kg Baujahr: 2019 Rahmengroesse: 55cm
>>> pkw = PKW(1200, 2016, 90, 5)
>>> print(pkw.show())
PKW Leergewicht: 1200kg Baujahr: 2016 Leistung: 90kW Sitzplätze: 5
```

(c) Implementieren die Funktion `maut(fzg: Fahrzeug) -> int`, welche den Preis zum Überqueren einer Brücke bestimmt. Dabei soll der Preis (in Cent) aus dem Gewicht des Fahrzeugs geteilt durch 10 plus die Anzahl der möglichen Passagiere mal 20 errechnet werden. Fahrräder dürfen umsonst die Brücke überqueren. Implementieren Sie hierzu entweder Hilfmethoden `weight` und `seats` für alle Fahrzeugklassen (wie im vorherigen Aufgabenteil), oder unterscheiden Sie mit `isinstance` direkt innerhalb der Definition von `maut`. Beispiel:

```
>>> print(maut(pkw))
220
```

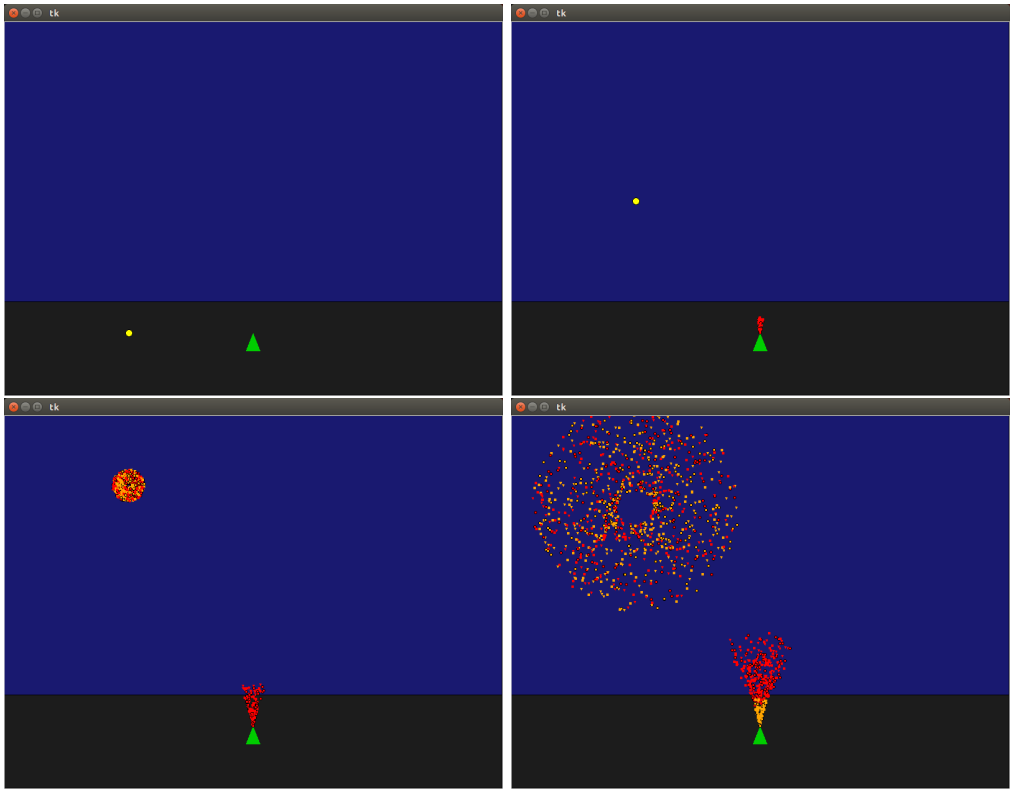

Abbildung 1

Aufgabe 8.2 (Gruppenaufgabe: Feuerwerk; Datei: `fireworks.py`; Punkte: 3+3+3+1)
Auch wenn dieses Jahr das Böllern an Sylvester verboten ist, wollen wir es zumindest digital so richtig krachen lassen. Angedacht ist ein Feuerwerk, dass jeden `tkinter.Canvas` zum Leuchten bringt. Um Ihnen ein wenig Arbeit abzunehmen, haben wir bereits einen Leuchtvulkan implementiert. Diesen finden Sie in der Datei `fireworks.py` auf der Vorlesungs-Website. Ihre Aufgabe besteht im Folgenden darin, das Modul um neue Klassen zu erweitern, damit weitere Typen von Feuerwerkskörpern erstellt und in die virtuelle Szenerie eingefügt werden können. Unter allen Lösungen wird eine Spezialjury das schönste und originellste Feuerwerk auswählen. Die Erschaffenden dürfen fortan den Titel *Pythonista spectaculi pyrotechnici* tragen.

Hinweis: Sie benötigen zum Ausführen des Beispielcodes das `python3-tk` Paket. Auf Ubuntu kann dieses installiert werden mit: `sudo apt install python3-tk`

(a) Leuchtvulkane haben zweifelsohne ihren eigenen Charme. Das Erreichen großer Höhen und das explosionsartige Hinterlassen leuchtender Partikel am Himmel gehören jedoch nicht dazu. Um Ihr Feuerwerk vielschichtiger zu gestalten, implementieren Sie nun eine Klasse `Rocket`. `Rocket`-Objekte sollen an einer beliebigen Stelle auf dem Canvas erzeugt werden können, mit einer bestimmten Geschwindigkeit aufsteigen, und nach einer bestimmten Flugzeit explodieren.

Dabei soll ein beliebiges Muster aus Partikeln entstehen, welches sich nach einer vorgegebenen Lebensspanne wieder auflöst. Siehe Abbildung 1.

(b) Implementieren Sie eine Klasse `RocketLauncher`. Instanzen dieser Klasse sollen in variablen Abständen `Rocket`-Objekte erzeugen und in den Himmel schießen. Variieren Sie dabei Abschusswinkel, Geschwindigkeit, und Explosion der `Rocket`-Objekte um den Effekt zu verstärken.

(c) Überlegen Sie sich wenigstens einen weiteren Typ an Feuerwerkskörpern, und implementieren Sie diesen. Dokumentieren Sie jeweils die beabsichtigte Wirkung in einem Docstring.

(d) Nutzen Sie Ihre neu erworbenen Pyro-Fähigkeiten zum Inszenieren des ultimativen Feuerwerks zum Jahreswechsel 2020/2021. Erstellen Sie dazu Instanzen der zuvor implementierten Feuerwerks-Klassen und zünden diese auf dem `tkinter.Canvas`.

Aufgabe 8.3 (Erfahrungen; Datei: `erfahrungen.txt`; 2 Punkte)

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

Schreiben Sie den groben Zeitaufwand, den Sie für das *gesamte* Blatt benötigt haben, bitte wie folgt in die erste Zeile:

```
Zeitaufwand: 3.5h

[... Freitext, wie bisher ...]
```

Wenn sich genug Leute daran halten, dann können wir ein Pythonskript schreiben, welches automatisiert die durchschnittliche Bearbeitungszeit berechnet ;-)