Einführung in die Programmierung

Prof. Dr. Peter Thiemann
Marius Weidner
Simon Dorer, Timpe Hörig

Universität Freiburg
Institut für Informatik
Wintersemester 2025

Übungsblatt 8

Abgabe: Montag, 08.12.2023, 9:00 Uhr

Aufgabe 8.1 (Modellierung und Rekursion; 20 Punkte; Datei: `packages.py`)

In dieser Aufgabe modellieren Sie Pakete mit ihrem Zustellstatus und verwalten diese in einer Datenbank.

Hierzu existiert bereits die Datei `packages.py` in Ihrem Repository. Verwenden Sie diese Datei für Ihre Implementierung.

(a) **Pakete Modellieren; 4 Punkte**

Definieren Sie eine Datenklasse `Package` die eine Tracking-Id `tracking_id` und ein Lieferstatus `status` als Attribute besitzt.

Der Lieferstatus ist dabei entweder in Zustellung (`InTransit`), zugestellt (`Delivered`) oder es ist ein Problem aufgetreten (`Problem`):

Für Pakete, die sich aktuell in der Zustellung befinden sollen zusätzlich die geschätzten Stunden (nur ganzzahlig) bis zur Ankunft verfügbar sein.

Für bereits zugestellte Pakete, sollen zusätzlich die Informationen über den Abgabeort und den Nachnamen des Empfängers zur Verfügung stehen.

Gab es bei der Lieferung Probleme, so sollen zusätzlich Information über die Art des Problems zur Verfügung stehen. Der Problemcode `1` steht dabei für ein Problem mit der Adresse, `2` dafür, dass der Empfänger nicht erreichbar war und `3` dafür, dass das Paket beschädigt ist. Zusätzlich soll die Information vorhanden sein, ob es Kontakt mit dem Kunden benötigt oder nicht.

Überlegen Sie sich hierfür, wie Sie den Lieferstatus als Alternative verschiedener Datenklassen modellieren können.

Hinweis: Für Objekte, deren Repräsentation nicht exakt vorgegeben ist, dürfen und sollen Sie eine passende wählen. Achten Sie dabei darauf, dass ihre Modellierung es ermöglicht, die folgenden Teilaufgaben möglichst kurz, präzise und mit so wenig Codeduplizierung wie möglich implementieren zu können. Außerdem soll Ihre Modellierung für den Benutzer möglichst intuitiv und einfach zu verstehen sein.

(b) **Parsing; 2 Punkte**

Nun wird es Zeit alle Pakete, die bisher schon versendet wurden, in Ihre neue

Datenstruktur zu übersetzten. Bevor Sie damit beauftragt wurden, eine gute Modellierung zu finden, wurden Pakete wie folgt repräsentiert:

```
type NaivePackageType = tuple[str, int | tuple[str, str] | tuple[int, bool]]
```

Dabei ist das erste Element des Tupels die Tracking-Id und das zweite Element der Lieferstatus des Pakets. Dieser kann entweder ein `int` sein, der die geschätzte Ankunftszeit darstellt (falls das Paket in Zustellung ist), ein Tupel aus Abgabeort und Nachnamen (falls das Paket bereits zugestellt wurde) oder ein Tupel aus Problemcode und ob Kundenkontakt nötig ist (falls ein Problem aufgetreten ist).

Schreiben Sie eine Funktion `parse`, die einen solchen NaivePackageType `package` als Argument nimmt, diesen zu Ihrer `Package`-Definition übersetzt und zurückgibt. **Verwenden Sie hierfür Pattern Matching.**

(c) **Statusabfrage; 3 Punkte**

Schreiben Sie eine Funktion `status_info` die ein Package `package` als Argument nimmt und eine String zurückgibt, der Auskunft über den Status des Package enthält. Berücksichtigen Sie dabei, sowohl die Tracking-Id als auch die verschiedenen Informationen, die Ihnen je nach Lieferstatus zur Verfügung stehen. Ihr String muss dabei kein festes Format haben, soll aber alle vorhandenen Informationen enthalten und für den Benutzer verständlich sein. **Verwenden Sie hierfür Pattern Matching.**

(d) **PackageTree; 1 Punkt**

Den aktuellen Status seines eigenen Packages abzufragen ist für jeden möglich und dadurch wahrscheinlich die häufigste Operation auf der Datenbank. Es ist also von Vorteil, wenn das Nachschauen in der Datenbank möglichst effizient ist. Suchbäume sind hierfür gut geeignet, da Sie ein Nachschauen in $\mathcal{O}\log(n)$ ermöglichen.

In `packages.py` finden Sie bereits die Definition von Binärbäumen. Ändern Sie diese nicht ab:

```
@dataclass
class Node[T]:
    mark: T
    left: "BTree[T]"
    right: "BTree[T]"

type BTree[T] = Optional[Node[T]]
```

Erstellen Sie einen *Typen* `PackageTree` der einen Binärbaum von `Package`-Objekten repräsentiert. In unserem Suchbaum, nennen wir ein Package kleiner als ein anderes, falls seine `tracking_id` kleiner ist.

(e) **Suchen von Paketen; 1 Punkt**

Schreiben Sie eine Funktion `lookup` die einen PackageTree `packages` und eine Tracking-Id `tracking_id` als Argumente nimmt und das dazugehörige Package aus `packages` zurückgibt. Existiert kein Package mit dieser Tracking-Id, so soll `None` zurückgegeben werden. **Verwenden Sie hierfür Rekursion und Pattern Matching.**

(f) **Hinzufügen von Paketen; 2 Punkte**

Schreiben Sie eine Funktion `insert_package` die einen PackageTree `packages`, ein Package `package` und einen Wahrheitswert `overwrite` als Argument nimmt und einen *neuen* PackageTree zurückgibt, in dem das neue Package an richtiger Stelle eingefügt ist. Existiert bereits ein Package mit der selben Tracking-Id, so soll im neuen Baum, das bestehende Package genau dann ersetzt werden, falls `overwrite` wahr ist. Ansonsten soll das bestehende Package behalten und `package` verworfen werden. **Verwenden Sie hierfür Rekursion und Pattern Matching.**

(g) **Updaten von Paketstatus; 1.5 Punkte**

Schreiben Sie eine Funktion `update_status`, die einen PackageTree `packages`, eine Tracking-Id `tracking_id` und einen DeliveryStatus `status` als Argumente nimmt und einen neuen PackageTree zurückgibt, in dem der Status des Packages mit `tracking_id` mit `status` geupdated wurde (siehe Aufgabenteil f). Falls kein Package mit der entsprechenden Tracking-Id existiert, so soll nichts der PackageTree unverändert zurückgegeben werden. Benutzen Sie hierzu Ihre zwei selbst definierten Funktionen `lookup` und `insert_package`.

(h) **Auflisten aller Pakete für den Kundenservice; 4 Punkte**

Schreiben Sie eine Funktion `get_all_problematic` die einen PackageTree `packages` als Argument nimmt und eine Liste aller Packages zurückgibt, bei denen der Lieferstatus `Problem` ist und Kundenkontakt benötigt wird.

Lösen Sie die Aufgabe mit genau einem Pattern Matching mit drei `cases` und keinem `if` (auch keine Pattern Guards). Verwenden Sie hierzu geschachteltes Pattern Matching (siehe Folie 23 aus VL 09 zu Pattern matching.)

(i) **Erweiterte Statusabfrage; 1.5 Punkte**

Schreiben Sie eine Funktion `user_info` die einen PackageTree `packages` und eine Tracking-Id `tracking_id` als Argumente nimmt und einen String zurückgibt. Ist kein Package mit der Tracking-Id im PackageTree vorhanden, so soll der String `Paket mit Sendungsnummer <Tracking-Id> nicht gefunden` zurückgegeben werden. Existiert ein solches Package, so soll das Ergebnis der Statusabfrage (siehe Aufgabenteil c) zurückgegeben werden.

Aufgabe 8.2 (Erfahrungen; 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 Webplattform. Halten Sie sich an das dort vorgegebene Format, da wir den Zeitbedarf mit einem Python-Skript automatisch statistisch auswerten. Die Zeitangabe `7.5 h` steht für 7 Stunden 30 Minuten.

Der Build-Server überprüft ebenfalls, ob Sie das Format korrekt angegeben haben. Prüfen Sie, ob der Build-Server mit Ihrer Abgabe zufrieden ist, so wie es im Video zur Lehrplattform gezeigt wurde.