LU01.L10 - Anforderungen deklarativ formulieren

Es gibt hier nicht *die* eine richtige Formulierung. Bewertet wird, ob der Endzustand beschrieben wird und die Arbeitsschritte verschwunden sind. Die folgenden Fassungen sind Musterlösungen, keine Musterwortlaute.

Teil A: imperativ → deklarativ

Nr.	Deklarative Fassung	Typisch imperative Wörter der Vorlage
1	`zuerich.csv` enthält genau jene Zeilen aus `kunden.csv`, deren Postleitzahl mit 8 beginnt – in unveränderter Form.	„öffne„, „lies Zeile für Zeile“, „hänge an„, „schliesse am Schluss“
2	Ausgegeben wird die Anzahl Bestellungen mit einem Betrag über 500 Franken.	„setze auf 0„, „gehe durch“, „erhöhe um 1„, „am Ende“
3	Ausgegeben wird die höchste vorkommende Note.	„nimm die erste„, „merke dir“, „gehe durch„, „ersetze“, „zum Schluss„
4	Die Ausgabeliste enthält, alphabetisch sortiert, die Namen aller Lernenden, die in allen drei Fächern mindestens eine 4 haben.	„lege eine leere Liste an“, „durchlaufe„, „füge hinzu“, „sortiere am Ende„
5	Jede eingetroffene Datei liegt danach entweder in `verarbeitet` (falls sie nicht leer ist) oder in `fehler` (falls leer). Für jede leere Datei existiert ein Logeintrag.	„warte, bis“, „prüfe„, „verschiebe“, „schreibe einen Eintrag„

Woran man eine gelungene Umformulierung erkennt

Die Zwischenschritte sind verschwunden: kein Zähler, kein „Maximum merken“, keine leere Liste, keine Reihenfolge. Übrig bleibt eine Aussage, die man an einem fertigen Ergebnis **überprüfen** kann, ohne den Programmablauf zu kennen. Genau das macht deklarative Anforderungen auch als Testkriterium so brauchbar.

Teil B: deklarativ → imperativ

1. Abteilungen mit Mitarbeiterzahl

1. Ein leeres Wörterbuch `zaehler` anlegen.
2. Alle Mitarbeitenden nacheinander durchlaufen.
3. Für jeden Mitarbeitenden die Abteilung auslesen.
4. Ist die Abteilung noch nicht im Wörterbuch, sie mit dem Wert 1 eintragen; sonst den bestehenden Wert um 1 erhöhen.
5. Das Wörterbuch in eine Liste von Paaren (Abteilung, Anzahl) umwandeln.
6. Die Liste absteigend nach der Anzahl sortieren.
7. Die sortierte Liste zurückgeben.

2. Kunden ohne Dubletten

1. Die importierten Datensätze absteigend nach Änderungsdatum sortieren, damit der neuste zuerst kommt.
2. Ein leeres Set `gesehene_mails` und eine leere Ergebnisliste anlegen.
3. Die sortierten Datensätze durchlaufen.
4. Ist die E-Mail-Adresse bereits in `gesehene_mails`, den Datensatz überspringen.
5. Sonst den Datensatz an die Ergebnisliste anhängen und die Adresse in `gesehene_mails` aufnehmen.
6. Die Ergebnisliste zurückgeben.

Beobachtung: Beide imperativen Fassungen brauchen deutlich mehr Zeilen als die deklarative Anforderung – und sie legen bereits eine bestimmte Lösung fest. Die deklarative Anforderung liesse auch ein `GROUP BY` in der Datenbank zu, das dieselbe Aussage erfüllt.

Teil C: Reflexion

1. Die schwierigste Umformulierung

Meist Nummer 5. Grund: Sie enthält **zwei verschiedene Dinge** in einem Satz – eine Bedingung (leer oder nicht) und einen zeitlichen Ablauf (warten, bis die Datei eintrifft). Anforderungen 2 und 3 sind dagegen leicht, weil dahinter eine bekannte Aggregatfunktion steht (Anzahl, Maximum).

2. Was sich nicht deklarativ beschreiben lässt

Das **Warten** auf die Datei. Ein Endzustand ist eine Aussage über das Ergebnis – „warte, bis„ beschreibt aber einen Vorgang über die Zeit hinweg. Ebenso wenig lässt sich deklarativ ausdrücken, dass eine Datei *verschoben* (also am alten Ort gelöscht) wird: Das ist ein Seiteneffekt, kein Zustand.

Deklarativ formulierbar ist nur das Resultat: „Nach der Verarbeitung liegt jede Datei in genau einem der beiden Zielordner, und der Quellordner ist leer.“

Merke: Anforderungen an **Berechnungen** lassen sich fast immer deklarativ fassen. Anforderungen an **Ein-/Ausgabe, Zeit und Nebenläufigkeit** nur teilweise – auch das ist ein Vorgriff auf die Grenzen der reinen funktionalen Programmierung.

3. Vor- und Nachteile für die Umsetzung

Vorteil	**Nachteil**
Die Programmiererin darf die beste Lösung selbst wählen – Schleife, Comprehension, SQL-Abfrage oder Bibliotheksfunktion.	Es ist mehr Rückfragebedarf da: Was gilt bei leeren Eingaben? Bei Gleichstand? Bei fehlenden Werten?
Die Anforderung überlebt einen Technologiewechsel; die Schrittfolge nicht.	Ohne Beispieldaten bleibt „sortiert nach Anzahl„ mehrdeutig (auf- oder absteigend?).

Vorteil	Nachteil
Sie lässt sich direkt als Testkriterium verwenden.	Die Aufwandschätzung fällt schwerer, weil der Lösungsweg noch offen ist.

Das ist exakt derselbe Zielkonflikt wie beim Taxi-Beispiel aus LU01a: **Abstraktion gegen Kontrolle.**
