Informatik 2 für Regenerative Energien

Probeklausur 1: Lösungen

Jörn Loviscach

Versionsstand: 20. Juni 2024, 11:02

1. Die Fehler:

Zeile	korrekter Programmtext
13	`(Zeile löschen)`
22	`orte: list[Geokoordinaten] = []`
31	`for ort in BekannteOrte.orte:`
42	`class Wechselakku(Akku):`
44	`super().__init__(ladezustand)`
45	`self.roller: ERoller | None = None`
47	`def entferne_aus_roller(self) -> None:`
83	`return self.akku is not None and super().ist_fahrbereit()`
92	`class ERollerVerwaltung:`
94	`self.rollerflotte: list[ERoller] = []`
105	`return anzahl`
111	`entfernung = math.inf`
116	`if neue_entfernung < entfernung:`
117	`entfernung = neue_entfernung`
127	`x = e_roller.ist_fahrbereit()`

2. `x` ist `True`, weil beide Bedingungen der Methode `ist_fahrbereit` erfüllt sind. `y` ist `98.0`, weil jener Akku mit den `update` auf diesen Wert gesetzt worden ist. `z` ist `None`, weil die aktuelle Position keinen Ortsangabe enthält.

3. *Zum Beispiel:* Am Anfang der besagten Methode dies ergänzen:

```
if self.roller is not None:
    raise Exception('Steckt noch in einem anderen Roller!')
```

4. *Zum Beispiel diese Lösung:* Wenn die Position unbekannt ist, könnte man `Breitengrad` und `Längengrad` auf `None` setzen und vor jeder Verwendung dieser Felder prüfen, ob sie einen echten Wert enthalten. Die Typangaben sind dann auf `float | None` zu ändern.

Oder diese Lösung: Wenn die Position unbekannt ist, könnte man das Feld `position` auf `None` setzen und vor jeder Verwendung der Position prüfen, ob es einen echten Wert enthält. Die Typangabe ist dann auf `Geokoordinaten | None` zu ändern.

Oder: Wenn die Position unbekannt ist, könnte man `Breitengrad` und `Längengrad` auf `math.nan` setzen und vor jeder Verwendung dieser Felder prüfen, ob sie diesen Sonderwert enthalten.

Oder: Man könnte `Geokoordinaten` als Klasse *ohne* die Attribute `Breitengrad` und `Längengrad` definieren und von dieser Klasse eine Klasse `GeokoordinatenMitGraden` ableiten. Im Normalfall würde man letztere benutzen.

Oder: Man könnte die Klasse `Geokoordinaten` um ein `bool`-Attribut `ist_gültig` erweitern und vor jeder Verwendung einer Instanz von `Geokoordinaten` prüfen, ob dieses Attribut `True` ist. Unsauber dabei: Es stünden in `Breitengrad` und `Längengrad` Phantasiewerte, die gefährlich sind, wenn man vergisst, `ist_gültig` zu prüfen.

5. Den Initialisierer von `Akku` so ändern:

```
def __init__(self, ladezustand: float, hersteller: Hersteller):
    self.ladezustand = ladezustand
    self.hersteller = hersteller
```

Den Initialisierer von `Wechselakku` so ändern:

```
def __init__(self, ladezustand: float, hersteller: Hersteller):
    super().__init__(ladezustand, hersteller)
    # usw.
```

Dann die Instanzen entsprechend konstruieren:

`Wechselakku(100.0, Hersteller.A)` usw.

6. *Zum Beispiel:*

```
def finde_zu_ladende(self, position: Geokoordinaten) \
                                              -> list[ERoller]:
    zu_ladende: list[ERoller] = []
    for r in self.rollerflotte:
        if r.position.entfernung_zu(position) < 0.1 and \
                (r.akku is None or r.akku.ladezustand < 50.0):
            zu_ladende.append(r)
    return zu_ladende
```

Oder eleganter:

```
def finde_zu_ladende(self, position: Geokoordinaten) \
                                              -> list[ERoller]:
    return [r for r in self.rollerflotte if \
           r.position.entfernung_zu(position) < 0.1 \
           and (r.akku is None or r.akku.ladezustand < 50.0)]
```

7. Der Typ ist
   `list[dict[str, int | str | float | set[str] | dict[str, int]]]`.

8. Die Werte sind 8, 3, 6.