LU09b - Zweiseitige Mehrfachbeziehung (n zu n)

Aus der realen Welt sind Mehrfachbeziehungen bestens bekannt. So kann eine Feriendestination von vielen Personen gebucht werden und umgekehrt kann eine Person viele Destinationen besuchen.

Diese Beziehung wird wie folgt im UML-Klassendiagramm dargestellt.

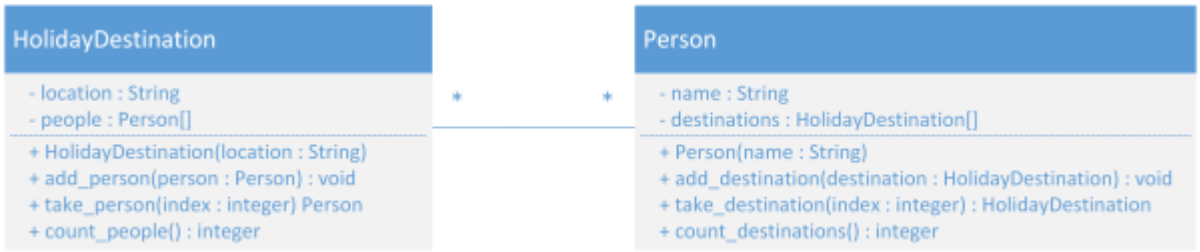

Abb: n:n Beziehung

Bei einer zweiseitigen Mehrfachbeziehung enthält jedes Objekt eine Liste mit Referenzen zu den verbundenen Objekten. Beim Hinzufügen (`add`) und Löschen (`remove`) von Beziehungen, müssen Sie immer beide Listen anpassen. Dabei muss verhindert werden, dass ein Endlos-Loop entsteht, wenn sie die `add_person` und `add_destination`-Methoden gegenseitig aufrufen.

Umsetzung in Python

Für unsere Umsetzung prüfen wir zuerst, ob das Objekt in der Liste vorhanden ist. Nur wenn dies der Fall ist, wird die Beziehung hinzugefügt bzw. gelöscht.

```
class HolidayDestination:
    def __init__(self, location):
        self._location = location
        self._people = []

    def __str__(self):
        output = f'People visiting {self._location}: '
        for person in self._people:
            output += f' {person._name},'
        output += '\n------'
        return output

    def add_person(self, person):
        if not person in self._people:
            self._people.append(person)
            person.add_destination(self)

    def remove_person(self, person):
        if person in self._people:
```

```
            self._people.remove(person)
            person.remove_destination(self)


class Person:
    def __init__(self, name):
        self._name = name
        self._destinations = []

    def __str__(self):
        output = f'{self._name} is visting'
        for destination in self._destinations:
            output += f' {destination._location},'
        output += '\n------'
        return output

    def add_destination(self, destination):
        if not destination in self._destinations:
            self._destinations.append(destination)
            destination.add_person(self)

    def remove_destination(self, destination):
        if destination in self._destinations:
            self._destinations.remove(destination)
            destination.remove_person(self)


def main():
    # let's start by creating some people and destinations
    maya = Person('Maya')
    thomas = Person('Thomas')
    jusuf = Person('Jusuf')
    valley = HolidayDestination('Death valley')
    atlantis = HolidayDestination('Atlantis')

    # now we let people visit the destinations
    maya.add_destination(valley)
    valley.add_person(thomas)
    print(valley)
    atlantis.add_person(maya)
    jusuf.add_destination(atlantis)
    print(atlantis)
    print(maya, thomas, jusuf)

    # try to add two duplicates
    jusuf.add_destination(atlantis)
    atlantis.add_person(maya)
    print(atlantis)

    # and now some people are cancelling their vacation
```

```
    valley.remove_person(thomas)
    print (valley)
    jusuf.remove_destination(atlantis)
    print (jusuf)

    # finally we try to remove non-existing relationships
    atlantis.remove_person(thomas)
    print(atlantis)
    jusuf.remove_destination(valley)
    print (jusuf)

if __name__ == '__main__':
    main()
```

Natürlich könnte man die Methoden auch so anpassen, dass sie als Parameter einen Index, einen Key und/oder ein Objekt akzeptieren.

M320-LU09