1. Referenzen zuweisen

Damit ein Objekt A ein anderes Objekt B ansprechen kann, muss es dessen Referenz kennen. Dazu muss die Referenz über den Konstruktor oder eine Methode übergeben werden.
Vom grundsätzlich Verhalten her unterscheiden wir in

- einseitge Beziehungen
- zweiseitige Beziehungen

Einseitige Beziehung

Bei einer einseitigen Beziehung kennt ein Objekt A ein anderes Objekt B aber nicht umgekehrt.

Beispiel 5.1: Geldbeutel und Besitzer

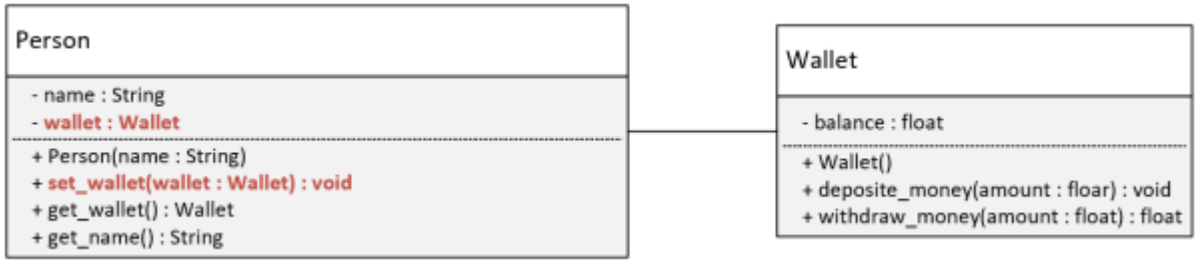


Abb 5.1: Klassen in einer einseitigen Beziehung

Eine Person kann auch keinen Geldbeutel besitzen. Daher wird hier die Referenz über eine Methode gesetzt. So sind die Erzeugung des Personen-Objekts und die Zuweisung der Referenz zeitlich unabhängig.

Dies zeigt das zugehörige Sequenz-Diagramm eindeutig auf.

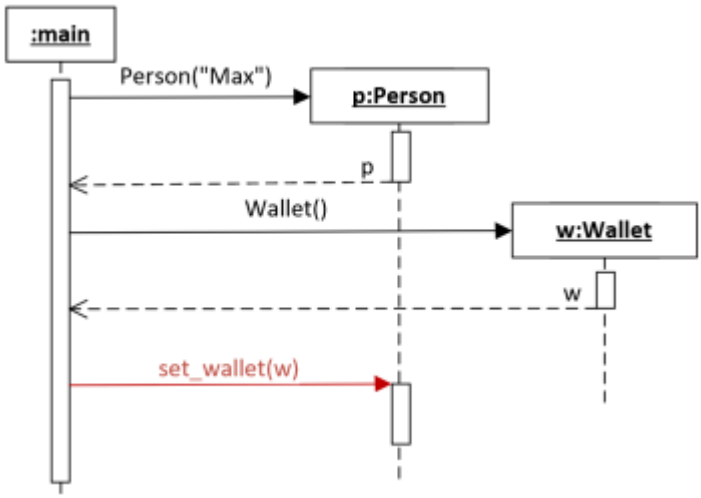


Abb 5.2: Sequenz-Diagramm der Zuweisung einer einseitigen Beziehung über eine Methode

Beispiel 5.2: Stromschalter und Wippe

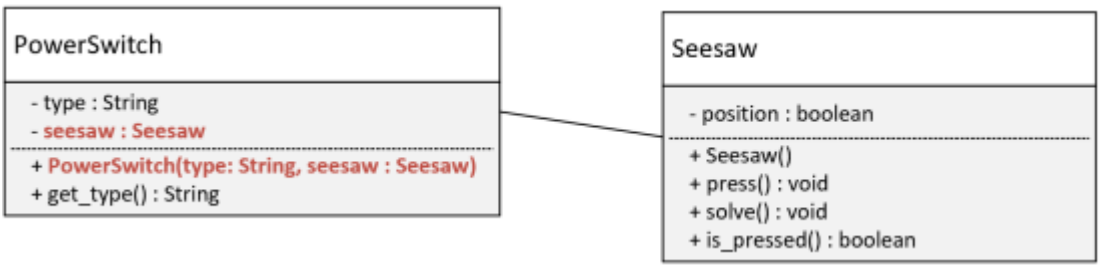


Abb 5.3: Klassen in einer einseitigen Beziehung

Ein Stromschalter braucht immer eine Wippe, sonst kann er ja nicht benutzt werden. Daher muss sichergestellt sein, dass mit der Erzeugung des Stromschalter-Objektes auch die Referenz zu einer Wippe gesetzt wird. Es ist also wichtig, dass die Referenz mit dem Konstruktor geliefert wird. In diesem Fall ist es zwingend, dass die Wippe zeitlich zuerst erzeugt wird, so dass ihre Referenz verfügbar ist.

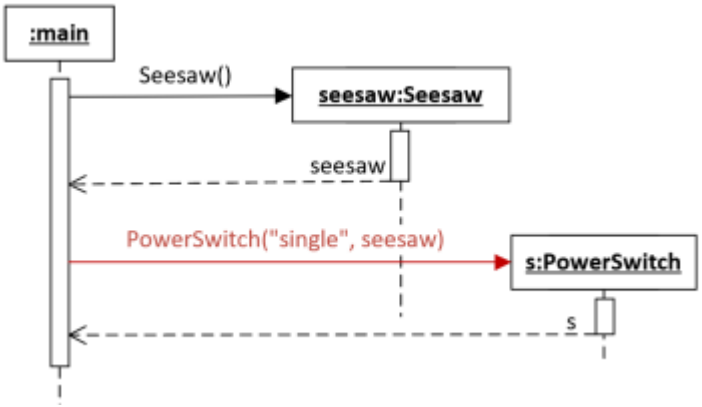


Abb 5.4: Sequenz-Diagramm der Zuweisung einer einseitigen Beziehung über den Konstruktor

Zweiseitige Beziehung

Bei einer zweiseitigen Beziehung kennt ein Objekt A ein anderes Objekt B und umgekehrt.

Beispiel 5.3: Lernende/r und Lehrperson



Abb 5.5: Klassen in einer zweiseitigen Beziehung

Es ist offensichtlich, dass eine Lehrperson nicht zu jedem Student und umgekehrt nicht jeder Student zu einer Lehrperson eine Beziehung pflegen muss. Wie in Beispiel 5.1 müssen daher auch hier die Objekte nicht zwingend eine Referenz zu einem anderen Objekt kennen, um existieren zu können. Es macht Sinn, die Zuweisung dann zu machen, wenn sie benötigt wird.

Um sicherzugehen, dass immer eine zweiseitige Beziehung besteht, wird in der jeweiligen set-Methode gleich auch die „Rückbeziehung“ gesetzt.

Hinweis: Programmtechnisch muss einfach sichergestellt werden, dass eine schon bestehende Beziehung nicht noch einmal zugewiesen wird.

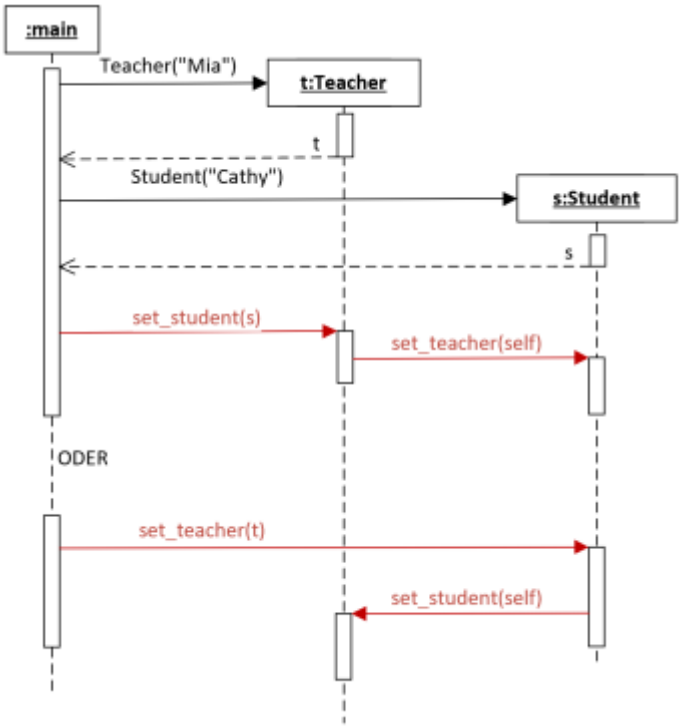


Abb 5.6: Sequenz-Diagramm der Zuweisung einer zweiseitigen Beziehung über Methoden

---

Beispiel 5.4: Lernende/r und Schulklasse



Abb 5.7: Klassen in einer zweiseitigen Beziehung

Ein Student wird bei der Anmeldung einer Klasse zugewiesen. Daher macht es Sinn, diese Referenz über den Konstruktor mitzuteilen. Auch hier ist wichtig, dass die Klasse `SchoolClass` zeitlich vorher erstellt wird, damit die Referenz verfügbar ist.

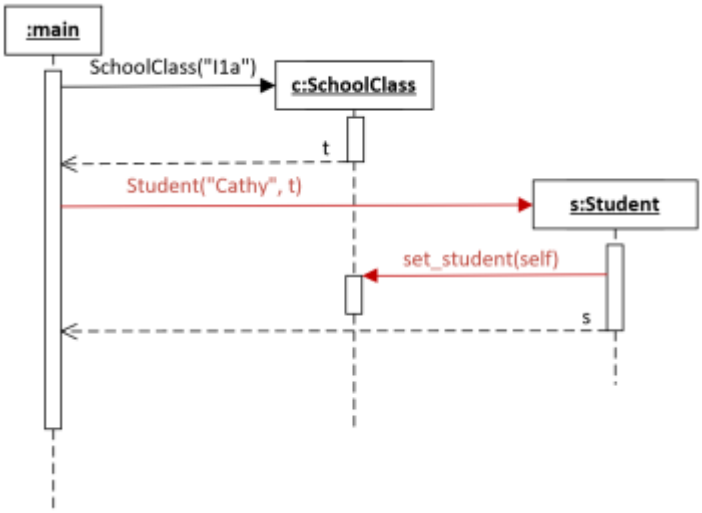


Abb 5.8: Sequenz-Diagramm der Zuweisung einer zweiseitigen Beziehung über Konstruktor

Code-Beispiele

Zu den oben erwähnten 4 Fällen finden Sie hier den passenden Code in Python und Java.

Fall 1:

Geldbeutel und Besitzer

Python **Java**

```
class Person:
public class Person{
private Wallet wallet = null;
private String name;

    def __init__(self, name):
public Person(name){
        self._name   = name
this.name = name;
        self._wallet = None # die Referenz wird erst später zugewiesen
```

```
}

    @property
    def name(self):
public String getName(){
        return self._name
return name;
}
    @property
    def wallet(self):
public Walle getWallet(){
        return self._wallet
return wallet;
}
    @wallet.setter
    def wallet(self, wallet):
public void setWallet(Wallet wallet){
        self._wallet = wallet
this.wallet = wallet;
}
}

class Wallet:
public class Wallet{

    def __init__(self):
public Wallet(){
        pass
}

    def deposit_money(self, amount):
public depositMoney(float amount){
        pass
}

    def withdraw_money(self, amount):
public withdrawMoney(float amount){
        pass
}
}

public class App{
if __name__ == '__main__':
public static void main(String[] args){
    person = Person('Max')
Person person = new Person("Max");
    wallet = Wallet()
Wallet wallet = new Wallet();
    person.wallet = wallet
person.setWallet(wallet);
}
```

```
}
```

Fall 2:

Stromschalter und Wippe

```
class PowerSwitch:
public class PowerSwitch{
private String type;
private Seesaw seesaw;

    def __init__(self, type, seesaw):
public PowerSwitch(String type, Seesaw seesaw){
        self._type   = type
this.type   = type;
        self._seesaw = seesaw
this.seesaw = seesaw;
}
    @property
    def type(self):
public String getType(){
        return self._type
return type;
}
}

class Seesaw:
public class Seesaw{
private boolean position = fasle;
    def __init__(self):
        self._position = False
/* Default-Konstruktor muss bei Java nicht angeschrieben werden */

    def press(self):
public void press(){
        pass
}

    def resolve(self):
public void resolve(){
        pass
}

    def is_pressed(self):
public boolean isPressed(){
        return self._position
return position;
}
```

```
}

public class App{
if __name__ == '__main__':
public static void main(String args[]){
    seesaw = Seesaw()
Seesaw seesaw      = new Seesaw();
    switch = PowerSwitch('single', seesaw)
PowerSwitch switch = new PowerSwicth("single", seesaw);
}
}
```

Fall 3:

Lernende/r und Lehrperson

```
class Teacher:
public class Teacher{
private String name;
private Student ref2student;

    def __init__(self, name):
public Teacher(String name){
        self._name        = name
this.name = name;
        self._ref2student = None
}

    @student.setter
    def student(self, student):
public void setStudent(Student student){
        self._ref2student = student
ref2student = student;
        student.teacher = self
student.setTeacher(this);
}
}

class Student:
public class Student{
private Strig name;
private Teacher ref2teacher;

    def __init__(self, name):
public Student(String name){
        self._name        = name
this.name = name;
        self._ref2teacher = None
}
```

```
    @teacher.setter
    def teacher(self, teacher):
public void setTeacher(Teacher teacher){
        self._ref2teacher = teacher
ref2teacher = teacher;
        teacher.student = self
teacher.setStudent(this);
}
}

public class App{
if __name__ == '__main__':
public static void main(String [] args){
    mia   = Teacher('Mia')
Teacher mia   = new Teacher("Mia");
    cathy = Student('Cathy')
Student cathy = new Student("Cathy");
    mia.student = cathy
mia.setStudent(cathy);
    #ODER
// ODER
    cathy.teacher = mia
cathy.setTeacher(mia);
}
}
```

Fall 4:

Lernende/r und Schulklasse

```
class SchoolClass:
public class SchoolClass{
private String designation;
private Stduent ref2student;

    def __init__(self, designation):
public SchoolClass(String desigantion){
        self._designation = designation
this.designation = designation;
        self._ref2student = None
}

    @student.setter
    def student(self, student):
public setStudent(Student student){
        self._ref2student = student
ref2student = student;
}
```

```
}

class Student:
public class Student{
private String name;
private SchoolClass ref2class;

    def __init__(self, name, the_class):
public Student(String name, SchoolClass theClass){
        self._name = name
this.name = name;
        self._ref2class = the_class
ref2class = theClass;
        the_class.student = self
theClass.setStudent(this);
}
}

public class App{
if __name__ == '__main__':
public static void main(String[] args){
    i1a   = SchoolClass('I1a')
SchoolClass i1a = new SchoolClass("I1a");
    cathy = Student("Cathy", i1a)
Student cathy   = new Student("Cathy", i1a);
}
}
```

© René Probst

From:
https://wiki.bzz.ch/ - **BZZ - Modulwiki**

Permanent link:
https://wiki.bzz.ch/modul/archiv/m320/learningunits/lu05/theorie/referenzen

Last update: **2026/08/30 14:09**