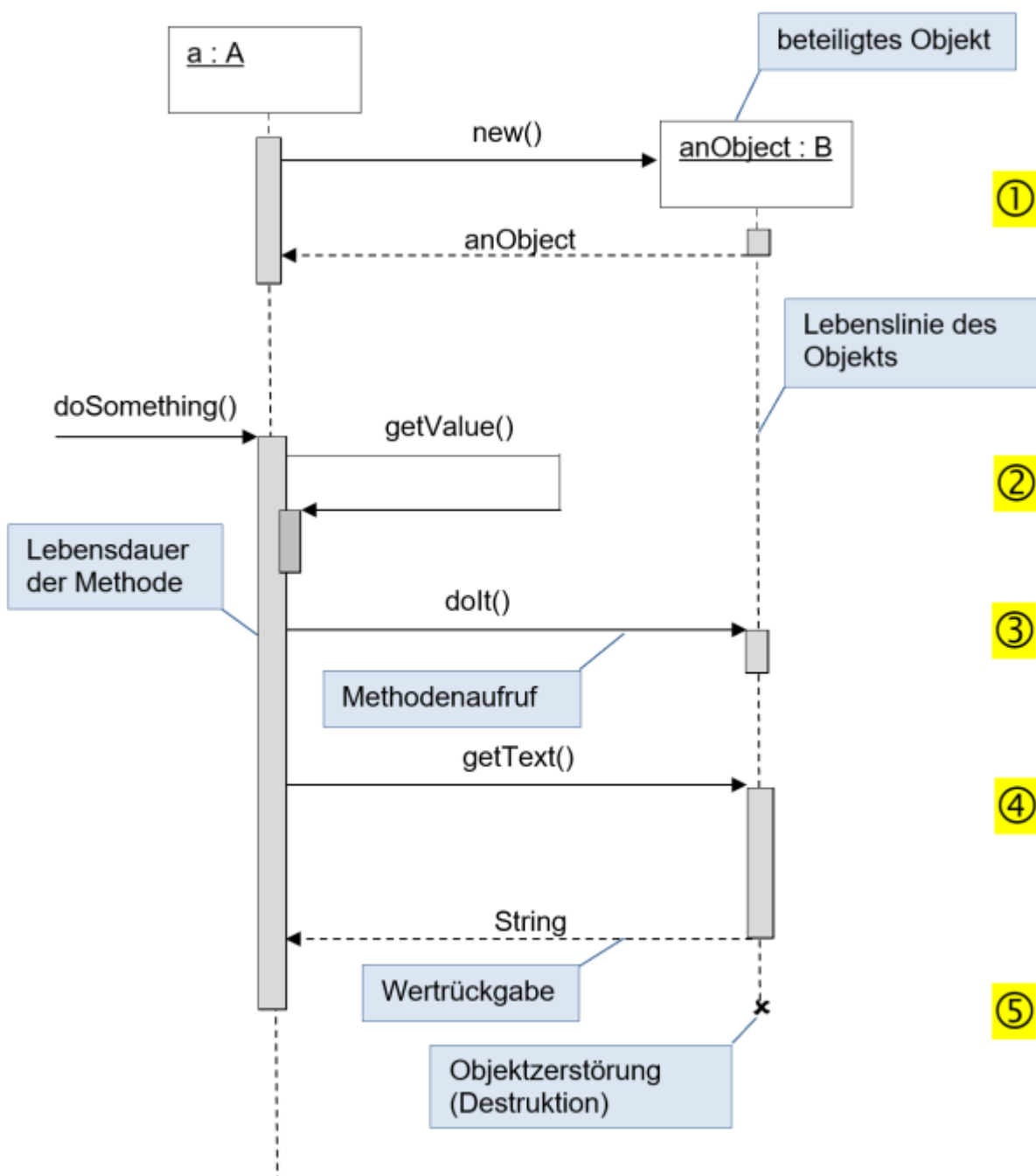


4 Sequenzdiagramm

Das Sequenzdiagramm zeigt die Interaktion von Objekten im zeitlichen Ablauf auf. Es sind somit mindesten 2 Objekte beteiligt

UML-Notation



(1) Instanziierung eines Objektes

```
class A:
    # Klasse A erzeugt selber ein Objekt der Klasse B.

    def __init__(self):
        # Erzeugung des Objektes vom Typ B im Konstruktor von A
        self._anObject = B()
    pass
```

(2) Selbstaufruf einer Methode mit Wertrückgabe

```
class A:
    # Klasse A ruft eine eigene Methode – hier mit einem Rückgabewert – auf

    @property
    def value(self):
        return self._value

    def doSomething(self):
        .....
        v = self.value
    pass
```

(3) Aufruf einer Methode in einer anderen Klasse

<pre>class A: # A ruft eine Methode der Klasse B auf. bereit def doSomething(self): self._anObject.doIt()</pre>	<pre>class B: # B stellt eine Methode def doIt(self): pass</pre>
--	---

(4) Aufruf einer Methode mit Wertrückgabe

<pre>class A: # A ruft Methode von B auf, # die einen Wert liefert.e def doSomething(self): txt = self._anObject.text pass</pre>	<pre>class B: # B stellt eine Methode bereit @property def text(self): return aText</pre>
---	--

Objektzerstörung

Nicht mehr benutzte Objekte müssen aus dem Speicher entfernt werden, damit der belegte Speicherplatz wieder freigegeben werden kann.

Python führt eine `garbage collection`. Diese ist besorgt, dass nicht mehr benutzte Objekte aus dem Speicher entfernt werden.

```
class A:
    ....
    Hier kann die Klasse

    del(self._anObject)
    pass
```

```
class B:
    # Destruktor definieren.

    # ordentlich beendet werden.
    def __del__(self):
        pass
```

© René Probst

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

Permanent link:
https://wiki.bzz.ch/modul/archiv/m320/merkblaetter/merkblatt_4

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

