ADT Stack bei Schrankwand

Einen Stack
(Stapel)
einsetzen

ADT Stack bei Schrankwand

- Standarddatenstruktur für änderbare Daten bei Python ist die Liste.
- Die Schrankwand verlangt aber in der Regel besondere Anforderungen:
  - Die Schrankwand sollte zusammenhängend sein.
  - Der "erste" Schrank sollte den Anfang der Schrankwand definieren.

ADT Stack bei Schrankwand

- Daraus ergeben sich besondere Anforderungen für die Methoden FuegeHinzu(…) und Entferne():
  - Ein Schrank wird der Schrankwand immer am Ende hinzugefügt.
  - Ein Schrank wird aus der Schrankwand immer am Ende enfernt.
- Das Erfüllen dieser Anforderungen kann man sicherstellen, indem man als Datenstruktur nicht eine Liste, sondern einen Stack verwendet.

ADT Stack bei Schrankwand

- Der Stack lässt sich als
  abstrakter Datentyp
  A D T
  beschreiben.
- *Ein Abstrakter Datentyp (ADT) ist ein Verbund von Daten zusammen mit der Definition aller zulässigen Operationen, die auf sie zugreifen.*
  [Quelle: wikipedia]

ADT Stack bei Schrankwand

- In der OO realisieren wir einen Stack in einer Klasse.

```
class Stack(object) :
    '''Implementiert einen Stack (Stapel) ...'''
    def __init__(self) :
        '''Konstruktor'''
    def FuegeHinzu (self, objekt) :
        '''legt ein neues Element oben auf den Stapel'''
    def Entferne (self) :
        '''entfernt das Element oben vom Stapel und gibt es zurück'''
    def GibOberstes (self) :
        '''gibt das Element oben auf dem Stapel zurück'''
    def GibLaenge (self) :
        '''gibt die aktuelle Länge des Stapels zurück'''
    def IstLeer (self) :
        '''gibt zurück, ob der Stapel leer ist'''
```

ADT Stack bei Schrankwand

- Wir fügen noch einen Iterator hinzu.

```
class Stack(object) :
    '''Implementiert einen Stack (Stapel)  mit Iterator und len [besondere Methoden]'''
    def __init__(self) :
        '''Konstruktor'''
    def __len__(self):
    def __iter__(self):
    def __next__(self):
    def FuegeHinzu (self, objekt) :
        '''legt ein neues Element oben auf den Stapel'''
    def Entferne (self) :
        '''entfernt das Element oben vom Stapel und gibt es zurück'''
    def GibOberstes (self) :
        '''gibt das Element oben auf dem Stapel zurück'''
    def GibLaenge (self) :
        '''gibt die aktuelle Länge des Stapels zurück'''
    def IstLeer (self) :
        '''gibt zurück, ob der Stapel leer ist'''
```

ADT Stack bei Schrankwand

```
class Stack(object) :
    '''Implementiert einen Stack (Stapel) mit Iterator und len'''

    def __init__(self) :
        '''Konstruktor'''
        self.__inhalte = []

    def __len__(self):
        return len(self.__inhalte)

    def __iter__(self):
        self.index = len(self.__inhalte)-1
        return self

    def __next__(self):
        if self.index<0:
            raise StopIteration
        else:
            wert=self.__inhalte[self.index]
            self.index -= 1
            return wert
```

ermöglicht
len(stack)

ermöglichen
for element in stack:

rückwärts iterieren!

ADT Stack bei Schrankwand

- Die Standardmethoden für die Schrankwand

```
def FuegeHinzu (self, objekt) :
        '''legt ein neues Element oben auf den Stapel'''
        self.__inhalte.append(objekt)

    def Entferne (self) :
        '''entfernt das Element oben vom Stapel und
        gibt es zurück'''
        if len(self.__inhalte)==0:
            return None
        indexLetztes=self.GibLaenge()-1
        letztesObjekt = self.__inhalte[indexLetztes]
        self.__inhalte=self.__inhalte[:indexLetztes]
        return letztesObjekt
```

ADT Stack bei Schrankwand

- Ergänzungen für den Stack als ADT

```
def GibOberstes (self) :
    '''gibt das Element oben auf dem Stapel zurück'''
    if len(self.__inhalte)==0:
        return None
    return self.__inhalte[self.GibLaenge()-1]

def GibLaenge (self) :
    '''gibt die aktuelle Länge des Stapels zurück'''
    return len(self.__inhalte)

def IstLeer (self) :
    '''gibt zurück, ob der Stapel leer ist'''
    return (len(self.__inhalte) == 0)
```