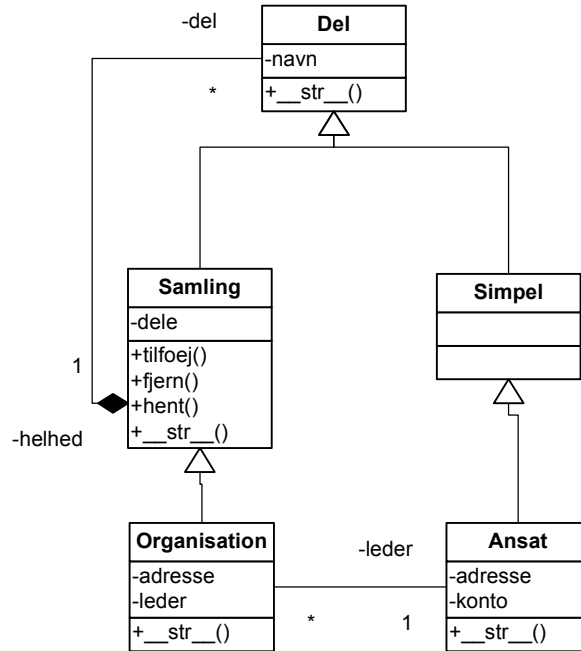


Standup-programmering

Her følger en løsning på den mislykkede stand-up programmering:



Den viser hvordan vi kan genbruge design-mønstre ved at specialisere klasserne i mønstret. Organisation og Ansatt er specialiseringer af Samling og Simpel. Vi har tilføjet attributer og metoder der er nødvendige for at repræsentere det konkrete problemområde vi er interesseret i.

Jeg kan ikke fange den generalisering at både organisationer og ansatte har adresser. Derfor er vi nødt til at anføre det i begge klasser. Jeg har også været nødt til at gendefinere udskrift-funktionen `__str__()` i begge klasser.

Jeg har givet den øverste klasse funktionen

```
def introspect(self):
    text = 'Dictionary: '
    text += '\n'+ str(self.__dict__)
    text += '\n'+ 'Attributes and functions: '
    text += '\n'+ str(dir(self))
    return text
```

Så kan alle instanser fortælle om sig selv.

Koden er:

```
'''composite pattern implemented as a list'''
class Del:
    '''den abstrakte overklasse'''
```

```

def __init__(self, navn):
    self.navn = navn
def introspect(self):
    text = 'Dictionary: '
    text += '\n'+ str(self.__dict__)
    text += '\n'+ 'Attributes and functions: '
    text += '\n'+ str(dir(self))
    return text
def __str__(self, niveau = 1):
    '''giver en tekstuel repræsentation af klassen
    __str__ gør det muligt at printe klassen direkte'''
    text = '\n' + niveau * ' '+'(' + self.navn + ')'
    return text

class Samling(Del):
    '''repræsenterer en samling af dele, enten samlinger eller simple'''
    def __init__(self, navn, dele = None):
        Del.__init__(self, navn)
        if dele == None:
            self.dele = []
        else:
            self.dele = dele
    def tilfoej(self, enDel):
        self.dele.append(enDel)
    def fjern(self, enDel):
        if enDel in self.dele:
            self.dele.remove(enDel)
    def hent(self, delNavn):
        for d in self.dele:
            if d.navn == delNavn:
                return d
        return None
    def __str__(self, niveau = 1):
        '''giver en tekstuel repræsentation af klassen
        __str__ gør det muligt at printe klassen direkte'''
        text = '\n' + niveau * ' '+'(' + self.navn
        if self.dele != None:
            for enDel in self.dele:
                text += enDel.__str__(niveau+1)
        else:
            text += 'no parts'

        text += '\n' + niveau * ' '+'+'
        return text

class Simpel(Del):
    '''repræsenterer en usammensat del'''
    def __init__(self, navn):
        Del.__init__(self, navn)

class Organisation(Samling):
    def __init__(self, navn, adresse, leder, dele = None):
        Samling.__init__(self, navn, dele)
        self.adresse = adresse
        self.leder = leder
    def __str__(self, niveau = 1):
        text = '\n' + niveau * ' '+'(' + self.navn
        text += '\n' + (niveau+1) * ' '+'Leder: ' + self.leder.__str__(niveau+1)
        text += '\n' + (niveau+1) * ' '+'Dele: '
        if self.dele != None:
            for enDel in self.dele:
                text += enDel.__str__(niveau+1)
        else:
            text += 'no parts'
        text += '\n' + niveau * ' '+'+'
        return text

```

```

class Ansat(Simpel):
    def __init__(self, navn, adresse, konto):
        Simpel.__init__(self, navn)
        self.adresse = adresse
        self.konto = konto
    def __str__(self, niveau):
        text = '\n' + niveau * ' '+'(' + self.navn
        text += '\n' + (niveau+1) *' '+' 'Adresse:' + self.adresse
        text += '\n' + (niveau+1) *' '+' 'Konto:' + self.konto
        text += '\n' + niveau *' '+')'
        return text

#lav ledere
imvleder = Ansat('Steffen Brandorff','Horsens', '65656565')
daimileder = Ansat('Kurt Jensen','Aarhus','545454')
humdekan = Ansat('Bodil Due','Vestergade 20', '32323232')
natdekan = Ansat('Erik Meineche Schmidt','Aarhus','363636')
rektor = Ansat('Lauritz B. Holm-Nielsen','Aarhus', '543636')

#lav organisationer
universitet = Organisation('Aarhus universitet','Aarhus', rektor)
humaniora = Organisation('Humaniora', 'Nobelparken',humdekan)
naturvidenskab = Organisation('Naturvidenskab', 'Ny Munkegade',natdekan)
imv = Organisation('Imv','Helsingforsgade',imvleder)
daimi = Organisation('Daimi','Aabogade',daimileder)

#lav ansatte
claus = Ansat('Claus bossen','Aarhus','6456445')
peter = Ansat('Peter Bøgh Andersen','Rønde','4343434')

#tilfoej dele af organisationerne
universitet.tilfoej(humaniora)
universitet.tilfoej(naturvidenskab)
humaniora.tilfoej(imv)
naturvidenskab.tilfoej(daimi)
imv.tilfoej(claus)
imv.tilfoej(peter)
imv.tilfoej(imvleder)
daimi.tilfoej(daimileder)
daimi.tilfoej(natdekan)

print universitet

```

Output er:

```

(Aarhus universitet
 Leder:
 (Lauritz B. Holm-Nielsen
  Adresse:Aarhus
  Konto:543636
 )
 Dele:
 (Humaniora
  Leder:
  (Bodil Due
   Adresse:Vestergade 20
   Konto:32323232
  )
  Dele:
  (Imv
   Leder:
   (Steffen Brandorff
    Adresse:Horsens
    Konto:65656565
   )
   Dele:
   (Claus bossen
    Adresse:Aarhus

```

```
    Konto:6456445
  )
  (Peter Bøgh Andersen
  Adresse:Rønde
  Konto:4343434
  )
  (Steffen Brandorff
  Adresse:Horsens
  Konto:65656565
  )
)
)
(Naturvidenskab
Leder:
(Erik Meineche Schmidt
  Adresse:Aarhus
  Konto:363636
)
Dele:
(Daimi
  Leder:
  (Kurt Jensen
  Adresse:Aarhus
  Konto:545454
  )
  Dele:
  (Kurt Jensen
  Adresse:Aarhus
  Konto:545454
  )
  (Erik Meineche Schmidt
  Adresse:Aarhus
  Konto:363636
  )
)
)
)
```