# Övning 4 - Programmeringsteknik 2012

```
0 # coding:ISO-8859-1
```

## Sammanfattning

Gick igenom klasser, objekt/instanser, instantiering, instansvariabler och metoder, konstruktor, self och inkapsling. Vi skrev tre versioner av klassen **Konto**. Med den kunde vi enkelt skapa valfritt antal konton med uppgifter om kontonummer, namn och belopp. I version 1 skapade vi klassen samt tre funktioner **lasaIn**, **skrivaUt** och **degPaBanken**. I version 2 lade vi till en funktion **sparaPaFil** samt skrev ett enkelt användargränsnitt. I version 3 deklarerade vi klassens instansvariabler som privata (inkapsling).

## Begrepp

- **Klass & objekt**, mall och instans skapad från mallen.
- **Instantiering**, skapa ett object
- **Instansvariablel**, variabler som Är unika för objektet. Dessa när man genom referensen(t ex k1.nr). En instansvariabel existerar under hela objektets livscykel och är tillgänglig från objektets samtliga metoder. objektet.
- **Instansmetoder**, metoder som är unika för objektet. Dessa när man enom referensen(t ex k1.deposit(100.0)). En instansmetod och dess eventuella lokala variabler existerar (likt andra metoder/funktioner) endast då den exekveras
- **Konstruktor**, funktion för skapande av objekt
- **self**, namn på objektet inne i klassens metoder
- **Inkapsling**

## Problem

Vi har en bank som behÃ¶ver hÃlla reda pÃ kontonummer, vem som Ãger kontot och hur mycket pengar som finns pÃ kontot. Vi vill att det ska gå att:

1. Det ska gå att ta ut och sÃtta in på kontot.
2. Kontoninformationen ska kunna läsas till och från fill
3. Räkna ut totala beloppet som finns på alla kontona tillsammans.

### Klassen

Vi vill skapa ett register av konton. Kontot ska innehÃlla:

1. Kontonummer
2. Namn pÃ Ägaren
3. Saldo

Kontohanteringen gÃr att lÃ¶sa med papper och penna men det skulle ta tid om antalet konton var mÃnga, t.ex. om banken hade 1 miljon kunder. LÃt oss istÃllet skriva ett program som kan hantera kontona Ãt oss. Ett bra program pÃ en snabb dator kan lÃtt hantera 1 miljon konton.

LÃt oss göra en mall frÃn vilken i kan skapa konton. Mallen blir en klass och konton vi skapar från klassen blir objekt (instanser).

Klassen ska innehÃlla:

1. Konstruktor **__init__**
2. StrÃngrepresentation av objektet **__str__**

3. InsÃttningsmetod **deposit**
4. Uttagningsmetod **withdraw**

```
91
92  class Konto:         # class reserverat ord
93
94      """
95      Konto(nr, namn, saldo)
96      Klass som definierar ett konto som man kan sÃtta in pÃ  och ta ut frÃn
97
98      Inputs:
99          nr          - kontonummer
100         namn        - namn pÃ  personen vars konto det Ãr
101         saldo       - pengar pÃ  kontot
102
103     Example:
104         >> k1 = Konto(1234567, Mikael Lindahl, 1000)
105     """
106     # Konstruktor
107     def __init__(self, nr, namn, saldo):  # self namn pÃ  objektet som
            skapas
108
109         # Instansvariabler
110         self.nr = nr
111         self.namn = namn
112         self.saldo = saldo
113
114     # Instansmetoder
115     def __str__(self):
116         return str(self.nr) + '\n' + self.namn + '\n' + str(self.saldo) + '
                kr\n'
117
118
119     def deposit(self, amount):
120         self.saldo += amount
121
122     def withdraw(self, amount):
123         self.saldo -= amount
```

## Metoder som slutar med __ (t.ex. __str__, __cmp__)

Class methods whose names start and end with __ are called "special methods". They allow you to customize the way python uses your class One thing you might want to customize is the string representation of your class. This is the string you get when you call str(xx) where xx is an instance of the class. It is also the string that prints when you say "print xx".

__cmp__can be used to implement rules of how to compare two objects.

## Funktioner

Nu vill vi kunna läsa in konton från en fil, skriva ut dem och kolla totala värdet på kontona.

Funktioner som använder klassen:

1. lasaIn
2. skrivUt
3. degPaBanken

```
133 import time
134
135 def lasIn():
136     infil = open('ovn4_infil.txt','r')
137     rad = infil.readline()
138     kontona = list()
139     while rad != '':
140         rad = rad.rstrip('\n')
141         delar = rad.split('/')
142         nr = int(delar[0])
143         namn = delar[1]
144         saldo = float(delar[2])
145         tmp = Konto(nr,namn,saldo)  # instantiering, tmp är ett objekt
146
147         kontona.append(tmp)
148         rad = infil.readline()
149     return kontona
150
151 def skrivUt(kontona):
152     for i in range(len(kontona)):
153         print(kontona[i])
154
155 def degPaBanken(kontona):
156     sum = 0
157     for i in range(len(kontona)):
158         sum += kontona[i].saldo     # Objektets instansvariable saldo
159
160     return sum
161
162 # Huvudprogram
163
164 start = time.time()
165 kontona = lasIn()
166 stop = time.time()
167 s = stop - start
168 m = s // 60
169 s = s - m*60
170
171 print('InlÃ sningstid av %i konton: %i minutes, %1.3f seconds' % (len(
    kontona), m, s ) )
172 print('\nTvÃ fÃ¶rsta:\n')
173 skrivUt(kontona[0:2])
174 print('Banken har totalt ', degPaBanken(kontona), ' kr')
```

## Ökad funktionalitet

- Skapa en yy metod sparaPaFil
- Gör menyvalet mer användarvänligt

```
183 def sparaPaFil(kontona):
184     utfil = open('kontoUt.txt','w')
185     for i in range(len(kontona)):
186         utfil.write(str(kontona[i].nr) + '/' + kontona[i].namn +'/' + str(
              kontona[i].saldo)+ '\n')
187     utfil.close()
188 
189 # Huvudprogram
190 kontona = lasIn()
191 
192 val = ''
193 while val != '0':
194     print('0 - avsluta')
195     print('1 - skriv ut konto')
196     print('2 - skriv ut summan av insatta pengar')
197     val = input('Ditt val: ')
198     if val == '1':
199         konto=int(input('Vilket konto (1-%i): '%(len(kontona))))
200         skrivUt([kontona[konto]])
201     if val == '2':
202         print('Banken har totalt ', degPaBanken(kontona), ' kr\n')
203 
204 sparaPaFil(kontona)
```

## Version 3 - inkpasling

- Inkapsling
- publikt, variabler och metoder som man kommer åt Även från andra klasser eller huvudprogrammet. self.utfall = 0
- privat, variabler och metoder som man bara kommer åt inifrön samma klass, börjar med två underscore (__) self.__ utfall = 0
- Man bör deklarera alla sina variabler som privata för att minimera manipulation utifrån. Detta kallas inkapsling.
- Vill man förändra variablers värden får man göra det genom en metod def kasta(self):

### Nytt i

Kapsla in klassvariablerna

```
237 class Konto:
238     """
239     Konto(nr, namn, saldo)
240     Klass som definierar ett konto som man kan sÃtta in pÃ oc ta ut frÃn
241 
242     Inputs:
243         nr          - kontonummer
244         namn        - namn pÃ personen vars konto det Ãr
245         saldo       - pengar pÃ kontot
246 
247     Example:
248         >> k1 = Konto(1234567, Mikael Lindahl, 1000)
249     """
250     def __init__(self, nr, namn, saldo):
251         self.__nr = nr
252         self.__namn = namn
253         self.__saldo = saldo
254 
255     def __str__(self):
256         return str(self.__nr) + '\n' + self.__namn + '\n' + str(self.__saldo
            ) + ' kr\n'
257 
258     def nr(self):
259         return self.__nr
260 
261     def namn(self):
262         return self.__namn
263 
264     def saldo(self):
265         return self.__saldo
266 
267     def deposit(self, amount):
268         self.__saldo += amount
269 
270     def withdraw(self, amount):
271         self.__saldo -= amount
272 
273 
274 def lasIn():
275     infil = open('ovn4_infil.txt','r')
276     rad = infil.readline()
277     kontona = list()
278     while rad != '':
279         rad = rad.rstrip('\n')
280         delar = rad.split('/')
281         nr = int(delar[0])
282         namn = delar[1]
283         saldo = float(delar[2])
284         tmp = Konto(nr,namn,saldo)
285         kontona.append(tmp)
286         rad = infil.readline()
287     infil.close()
288     return kontona
289 
290 def sparaPaFil(kontona):
291     utfil = open('kontoUt.txt','w')
292     for i in range(len(kontona)):
293         utfil.write(str(kontona[i].nr()) + '/' + kontona[i].namn() +'/' +
            str(kontona[i].saldo())+ '\n')
```

```
294     utfil.close()
295
296
297 def skrivUt(kontona):
298     for i in range(len(kontona)):
299         print(kontona[i])
300
301 def degPaBanken(kontona):
302     sum = 0
303     for i in range(len(kontona)):
304         sum += kontona[i].saldo()
305     return sum
```