Klassen Bank

Antag att vi skall skriva ett program som hanterar en enkel bank som har ett antal bankkonton.

Vi skall skriva en klass **`Bank`** som representerar denna enkla bank.

Vi konstaterar att vi har redan en klass som representerar **ett** bankkonto (klassen **`BankAccount`**) varför inte utnyttja den klassen.

Vi har också tidigare sett program (Test_BankAccount) som hanterar många bankkonton

Vi skall alltså utnyttja klassen **`BankAccount`** när vi skriver klassen **`Bank`**. Därmed kommer dessa båda klasser samverka.

Klassen Bank kommer alltså att innehålla ett antal **`BankAccount`**-objekt. Vi skall använda oss av en lista av **`BankAccount`**-objekt.

Att en klass byggs upp av en annan klass kallas för ***aggregat***. Klassen Bank är ett aggregat eftersom den är sammansatt av ett antal BankAccount-objekt.

Förutom att den enkla banken skall ha ett antal bankkonton, skall den ha ett namn.
Klassen **`Bank`** består alltså av:

- bankens namn (en str)

- ett antal bankkonton (en lista av **`BankAccount`**)

Konstruktor(er)
Metoder för:

- lägga till ett nytt konto
- få veta bankens kassa
- toString
- andra metoder…

Vi kikar på klassen: Bank.py

och ett program som testar klassen Bank:
TestBank.py

Exempel på ytterligare metoder som skulle kunna vara till nytta i klassen **`Bank`**

- Vi skriver ytterligare en metod `addBankAccount` som man anropar med kontonummer.

- Vi lägger till en metod som returnerar ett **`BankAccount`**-objekt och testar den. Tar upp begreppen *djupkopiering* eller *grundkopiering*

Några udda metoder
- Metod som returnerar listan med konton
- Metod som har listan som parameter

Vi tar upp hur dessa metoder kan se ut.