IN1000 - Seminaroppgaver til uke 10 - Løsningsforslag

Oppgave 1

1.1 Lag en klasse "Fag", som skal ha et navn og en liste som holder på alle studentene som tar det aktuelle faget. Ved opprettelsen av et fag vil denne studentlisten være tom.

1.2 Lag en metode i Fag-klassen som heter "leggTilStudent(student)" som tar imot en Student som parameter og legger den til i listen. Hvor i listen du legger den til spiller ingen rolle, så du må gjerne bruke .append().

1.3 Lag en metode i Fag-klassen som **returnerer** antallet studenter som tar faget, kall denne "hentAntallStudenter()".

1.4 Lag en metode som **returnerer** fagets navn, hentFagNavn().

1.5 Lag en metode "skrivStudenterVedFag()" som skriver ut fagnavnet, og deretter alle studentene som tar faget.
Merk: Anta at metoden hentStudentNavn() er implementert Student-objektet.

Løsningsforslag

```
class Fag:

    def __init__(self,navn):
        self._navn = navn
        self._studentListe = []

    def leggTilStudent(self, student):
        self._studentListe.append(student)

    def hentAntallStudenter(self):
        return len(self._studentListe)

    def hentFagNavn(self):
        return self._navn

    def skrivStudenterVedFag(self):
        print("Navn på fag:", self._navn)
        print("Studenter som tar faget:")
        for student in self._studentListe:
            print("\t",student.hentStudentNavn())
```

Kommentarer:

Når vi skal skrive en klasse, kan det være greit å skrive opp alle metodene og informasjonen vi har før vi begynner å fylle dem ut. Bruk gjerne *pass*, som brukt i oppgaveteksten til oppgave 7, for eksempel.

Når oppgaveteksten ber om at vi skal *returnere* en verdi, er det viktig at vi bruker ordet *return*.

Siden vi vet at listen med studenter *alltid* skal være tom, trenger vi ikke å ta imot denne listen som en parameter. Vi trenger bare å opprette en tom liste.

Denne klassen har referanser til en annen klasse som heter "Student". Noen ganger vet vi hva metodene til andre klasser heter, eller så vet vi hva de *burde* hete. Her er det viktig at når vi oppretter den andre klassen, så må vi passe på at vi bruker de riktige metodenavna.

"\t" legger til et innrykk, for syns skyld.

Oppgave 2

2.1 Lag en klasse "Student". Studenten skal ha et navn og en liste som holder på fagene studentene tar. Ved opprettelsen av en student vil denne faglisten være tom.

2.2 Legg til en metode i Student-klassen som heter "leggTilFag(fag)" som tar imot et fag som parameter og legger det til i studentens liste over fag.

2.3 Lag en metode i Student-klassen som **returnerer** antallet fag studenten tar, kall denne "hentAntallFag()".

2.4 Lag en metode i Student-klassen som **returnerer** studentens navn, hentStudentNavn().

2.5 Lag en metode "skrivFagPaaStudent()" som skriver ut studententens navn, og deretter alle fagene som studenten tar.
Her må du bruke hentFagNavn() fra Fag-objektet.

Løsningsforslag

```
class Student:
    def __init__(self, navn):
        self._navn = navn
        self._fagliste = []

    def leggTilFag(self, fag):
        self._fagliste.append(fag)

    def hentAntallFag(self):
        return len(self._fagliste)

    def hentStudentNavn(self):
        return self._navn

    def skrivFagPaaStudent(self):
        print("Navn:", self._navn)
        print("Fag som studenten tar:")
        for fag in self._fagliste:
            print("\t", fag.hentFagNavn())
```

Kommentarer

Denne klassen er strukturmessig veldig lik klassen Fag. De samme kommentarene gjelder her.

Oppgave 3

Lag et lite testprogram hvor du lager noen studenter og noen fag. Legg til fagene i studentenes liste og studentene inn i fagene. Skriv ut hvor mange fag studentene dine tar, og hvor mange studenter hver av fagene dine tar.

Løsningsforslag

Her kan vi skrive ut alt, eller vi kan mellomlagre studenter og fag i lister, for å forkorte koden litt.

Alternativ 1 (Skrive ut alt)

```
from student import Student
from fag import Fag


def hovedprogram():
    student1 = Student("Per")
    student2 = Student("Pål")
    student3 = Student("Kari")
    fag1 = Fag("IN1000")
    fag2 = Fag("IN1140")
    #Legger til fag til studenter og visa versa
    student1.leggTilFag(fag1)
    fag1.leggTilStudent(student1)
    student1.leggTilFag(fag2)
    fag2.leggTilStudent(student1)
    student2.leggTilFag(fag1)
    fag1.leggTilStudent(student2)
    student3.leggTilFag(fag1)
    fag1.leggTilStudent(student3)
    print(student1.hentStudentNavn(), "tar", student1.hentAntallFag(), "fag.")
    print(student2.hentStudentNavn(), "tar", student2.hentAntallFag(), "fag.")
    print(student3.hentStudentNavn(), "tar", student2.hentAntallFag(), "fag.")
    print(fag1.hentAntallStudenter(), "studenter tar", fag1.hentFagNavn())
    print(fag2.hentAntallStudenter(), "studenter tar", fag2.hentFagNavn())


hovedprogram()
```

Dette gir følgende utskrift:

```
Per tar 2 fag.
Pål tar 1 fag.
Kari tar 1 fag.
3 studenter tar IN1000
1 studenter tar IN1140
```

Alternativ 2 (Med lister)

```
from student import Student
from fag import Fag


def hovedprogram():
    studenter = []
    studenter.append(Student("Per"))
    studenter.append(Student("Pål"))
    studenter.append(Student("Kari"))
    fag = []
    fag.append(Fag("IN1000"))
    fag.append(Fag("IN1140"))
    #Alle studenter tar IN1000:
    for student in studenter:
        student.leggTilFag(fag[0])
        fag[0].leggTilStudent(student)
    #Én student tar IN1140
    fag[1].leggTilStudent(studenter[0])
    studenter[0].leggTilFag(fag[1])
    #Utskrift
    for st in studenter:
        print(st.hentStudentNavn(), "tar", st.hentAntallFag(), "fag." )
    for f in fag:
        print(f.hentAntallStudenter(), "studenter tar", f.hentFagNavn())


hovedprogram()
```

Dette gir følgende utskrift: (samme som alternativ 1)

```
Per tar 2 fag.
Pål tar 1 fag.
Kari tar 1 fag.
3 studenter tar IN1000
1 studenter tar IN1140
```

Kommentarer

Disse oppgavene var en måte å se på hvordan klasser kan refererer til hverandre. Vi så ikke på hvordan det kunne sett ut med lister i timen, men vi ser at i dette tilfellet blir det ikke veldig mye kortere kode. Hvis vi derimot hadde hatt veldig mange studenter, og flere fag, hvor mange studenter skulle tatt samme fag, vil vi kunne forkorte mye. Legg også merke til hvordan vi kan la være å mellomlagre objektene som variabler; vi legger referansene til objektene i lister istedenfor.