UiT
NORGES
ARKTISKE
UNIVERSITET

EKSAMENSOPPGAVE

Eksamen i:	INF-1400 Objektorientert programmering
Dato:	22 mai 2018
Klokkeslett:	09-13
Sted:	Adm.bygget, rom K1.04 og B154
Tillatte hjelpemidler:	Ingen
Type innføringsark (rute/linje):	
Antall sider inkl. forside:	**6**
Kontaktperson under eksamen: Telefon/mobil:	**Lars Brenna** **907 86 723**
Vil det bli gått oppklaringsrunde i eksamenslokalet? Svar: JA ~~/ NEI~~ Hvis JA: ca. kl. 10	

NB! Det er ikke tillatt å levere inn kladdepapir som del av eksamensbesvarelsen. Hvis det likevel leveres inn, vil kladdepapiret bli holdt tilbake og ikke bli sendt til sensur.

Eksamen INF-1400
Objektorientert programmering
Vår 2018

Eksamenssettet består av to deler, totalt fire oppgaver.

Les oppgaveteksten grundig og disponer tiden slik at du får tid til å svare på alle oppgavene. I noen oppgaver kan det være nødvendig å tolke oppgaveteksten ved å gjøre noen antagelser - gjør i så fall rede for hvilke antagelser du har gjort, men pass på å ikke gjøre antagelser som trivialiserer oppgaven.

Merk: Fokus i oppgaven er objektorientering og bruk av datastrukturer vi bygger opp. Det gis ikke ekstra poeng for å utvide oppgaven utover det som er spesifisert.

Pseudokode er i de fleste tilfeller godtatt, det kreves ikke kjørbar kode. Python3-syntaks er foretrukket, men ikke strengt nødvendig.

Del 1

Et universitet har behov for et system for å holde rede på studenters karakterbok. Så langt har universitetet benyttet et system skrevet i Python av tidligere studenter.

Dette systemet har blitt utvidet etterhvert som nye behov har meldt seg, men nå har det blitt tungvint å vedlikeholde og legge til nye funksjoner.

Universitetet ber deg lage en ny løsning som er objekt-orientert og som lar seg enhets-teste (unit-teste), og stiller følgende funksjonelle krav:

1. Hold rede på student, fag og karakter.
2. Systemet skal støtte at en karakterbok har flere studenter, en student tar flere fag, og hvert fag har flere karakterer.
3. Det skal kunne registreres karakter på student for hvert fag.
4. Det skal kunne beregnes gjennomsnittskarakter for hver student.
5. Karakterene i et fag skal ha vekting, slik at forskjellig vektede karakterer teller ulikt i gjennomsnittet.

Oppgave 1a - 15%

Implementér de klassene du anser som nødvendige. Du trenger ikke ta med andre metoder enn `__init__()`.

Oppgave 1b - 15%

Implementer metodene `registrer_student(self, navn)`, `registrer_fag(self, navn)`, `registrer_karakter(self, score, vekting)` i de klassene du mener de hører hjemme.

Du trenger ikke ta med den koden du implementerte over, men indiker tydelig hvilken klasse metodene tilhører.

Oppgave 1c - 15%

Implementer metoden `gjennomsnittskarakter(self)` slik at du kan finne gjennomsnittskarakteren for en gitt student, eksempelvis slik:

```
bok = Karakterbok()

kari = bok.registrer_student('Kari Nordkvinne')
matte = kari.registrer_fag('Kalkulus')
matte.registrer_karakter(75, 0.20)

print(kari.gjennomsnittskarakter())
```

Oppgave 1d - 5%

På grunn av tidligere erfaringer med bugs i systemet, krever universitetet at utviklerne av karakterboka benytter enhetstesting (unit testing).

Forklar kort hva enhetstesting er, og implementer en (enkel) test for en av komponentene i systemet.

Del 2

Oppgave 2 - 20%

Design patterns kan grovt inndeles i tre hovedkategorier:

1. Kreasjonelle ("Creational")
2. Strukturelle ("Structural")
3. Oppførselsbaserte ("Behavioral")

Forklar kort hva disse tre kategoriene representerer, og gi ETT eksempel fra hver kategori. Beskriv motivasjonen bak hvert eksempel du nevner, og hovedtrekkene i løsningen de skisserer.

Kode er ikke nødvendig med mindre du føler det gir en bedre beskrivelse.

Oppgave 3 - 15%

```
class A:
     name = "Alfa"

     def __init__(self, foo):
          self.foo = foo
          foo = 100
          self.print_me()

     def print_me(self):
          print (self.name, self.foo)

class B(A):
     name = "Beta"
     def __init__(self, bar = 40):
          self.bar = bar
          print(self.name, bar)

class C:
    name = "Charlie"

class D(A, C):
    name = "Delta"
    def __init__(self, val):
         A.__init__(self, val)

    def print_me(self):
        print(self.name, "sier", self.foo)
```

```
a = A(20)
b = B()
d = D(60)
```

a) Angi hvilke attributter B arver fra A.

b) Forklar kort hva polymorfi er.

c) Angi hva programmet over skriver ut.

Oppgave 4 - 15%

Gi en beskrivelse av to (2) av de følgende uttrykk/konsepter:

a) Assosiasjon

b) Polymorfi

c) Self (i Python)

d) Komposisjon

e) Arv