Plan for forelesningen

Time 1

- Spørsmål/ kommentarer til oppgave 1 og 2 på chat
- Gjennomgang oppgave 3a-d
- Gjennomgang oppgave 5

< svar på spørsmål til oppgave 1 og 2>

Time 2

- Mer om eksamen. Bruk chat for spørsmål
- Gjennomgang oppgave 3e og 4
- Evt svar på ubesvarte spørsmål i chat

Hensikten med prøveeksamen

- For dere:
  - Trene på eksamensrelevant stoff og oppgaver
  - Finne hull for videre trening
  - Bli kjent med Inspera, hvordan jobbe effektivt under hjemmeeksamen
- For oss:
  - Teste ut ulike typer oppgaver mer tilpasset hjemmeeksamen
  - Tilbakemelding fra tre gruppelærere brukes til eksamensoppgavene
- Du kan jobbe med prøveeksamen i Inspera frem til ca 1.12 (ikke i dag)

Oppgave 3e)

Oppgi tre uttrykk som du finner i dine svar på oppgavene 3a) til 3d). Uttrykkene skal evaluere til verdier av hver sin type. For hvert uttrykk, oppgi hvilken deloppgave du har besvart med koden du har hentet uttrykket fra.

*Du skal altså skrive tre linjer. På hver linje skriver du selve uttrykket (ikke en hel programsetning) fulgt av en parentes der du oppgir deloppgave, for eksempel **(3b)**.*

- Fra læreboka: Expression: A syntactical construct that is made up of constants, variables, function and method calls, and the operators combining them
- Fra ordlisten på semestersiden:

Expression	Uttrykk
An expression evaluates to a value	Et uttrykk evaluerer (beregnes) til en verdi

- Gjennomgått i uke 2 og uke 12 (se ukesidene)

Oppgaven krever:

1. Forstå oppgaveteksten og følge instruksjonene
2. Kjenne til og forstå begrepene (uttrykk, verdi, type)
3. Kunne relatere disse til konkrete kode-eksamepler

Oppgave 3a (5 poeng)

```
def pris_inkl_frakt(varepris):
    if varepris > 1000:
        return varepris
    elif varepris >= 500:
        return varepris + 50
    else:
        return varepris + 80

assert pris_inkl_frakt(300) == 380
assert pris_inkl_frakt(600) == 650
assert pris_inkl_frakt(1300) == 1300
```

Oppgave 3e) forts.

Oppgi tre uttrykk som du finner i dine svar på oppgavene 3a) til 3d). Uttrykkene skal evaluere til verdier av hver sin type. For hvert uttrykk, oppgi hvilken deloppgave du har besvart med koden du har hentet uttrykket fra.

*Du skal altså skrive tre linjer. På hver linje skriver du selve uttrykket (ikke en hel programsetning) fulgt av en parentes der du oppgir deloppgave, for eksempel **(3b)**.*

Oppgave 3b (5 poeng)

```
def fjern_utsolgte(handleliste, utsolgte):
    nyliste = []
    for vare in handleliste:
        if not vare in utsolgte:
            nyliste.append(vare)
        else:
            print(vare)
    return nyliste
```

:list →
:boolean →
:string →

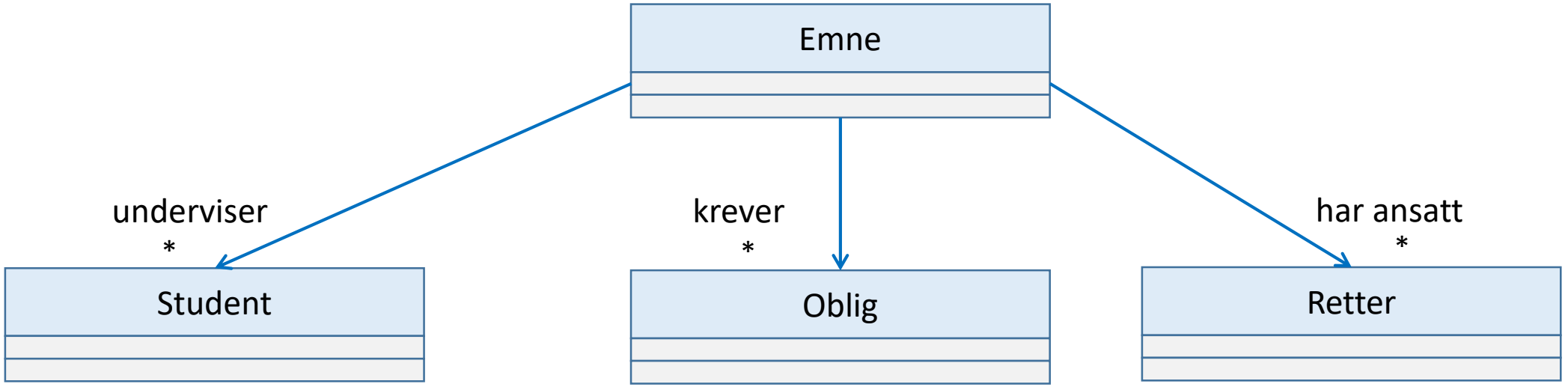
Svar (for eksempel):

[]	(3b)
vare in utsolgte	(3b)
vare	(3b)

Oppgave 4-01)

Les gjennom <...>. Tegn deretter et UML klassediagram som viser alle klasser du er bedt om å implementere (kun klassenavn) og relasjonene mellom dem. For hver relasjon skal du ta med antall og navn (hva relasjonen representerer).

IN1000 forenkling/ avgrensning:
-implementasjons-nær modell
-relasjoner viser kun klasser med instansvariabler som kan referere til objekter av andre klasser

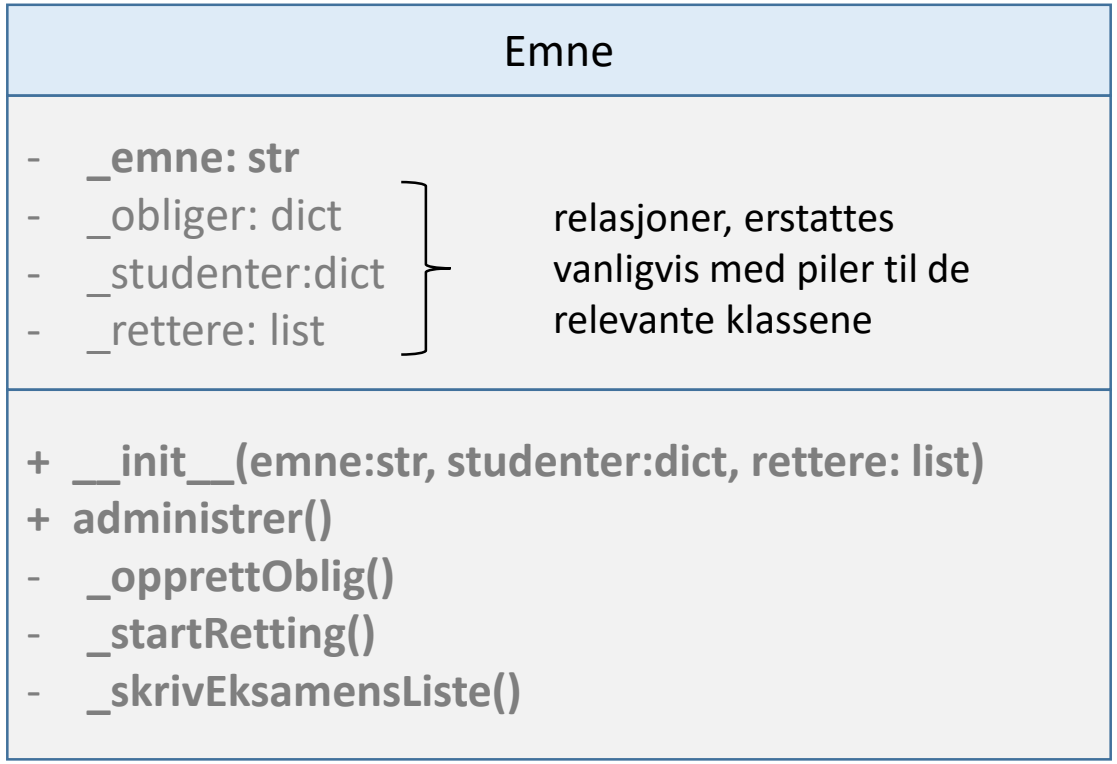


Oppgave 4-02)

Lag deretter en ny tegning, med et klassediagram som kun viser klassen Emne.

Her skal du ta med alle instansvariabler og metoder i Emne. Angi type for instansvariabler, parametere og returverdier, og om instansvariabler og metoder er public (tilgjengelige i klassens grensesnitt) eller ikke.

Det meste av dette er gitt i oppgaveteksten, noe vil du kanskje bestemme underveis i oppgave-løsingen.

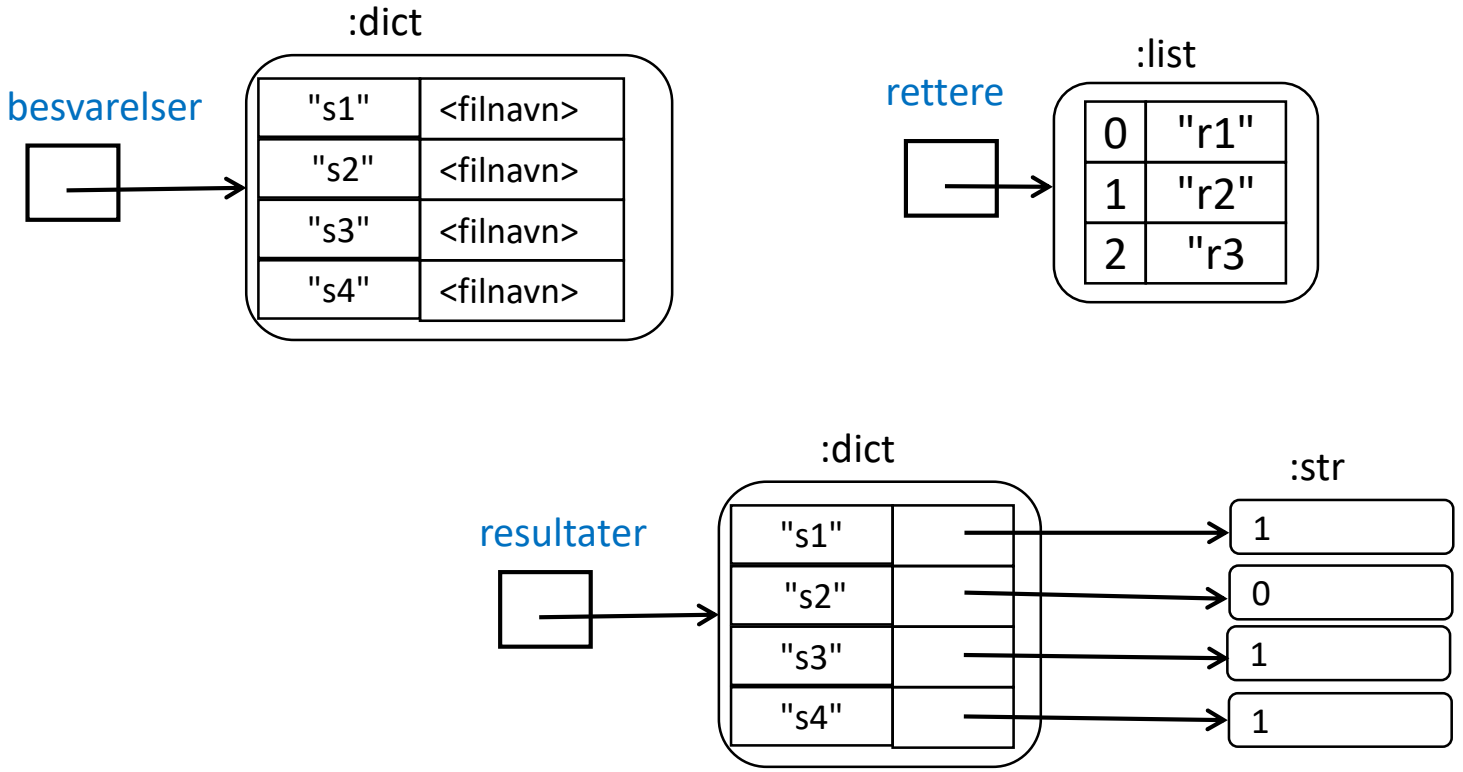


Oppgavene 4f)

fordelRetting *tar som parametere en ordbok med studenters besvarelser og en liste med rettere,*

og lar retterne vurdere én og én besvarelse (bruk metoden ***vurder*** *i klassen* ***Retter****) slik at alle retterne ideelt sett vurderer like mange. Altså skal ingen retter vurdere mer enn 1 besvarelse mer enn de andre.*

*Resultatene av retternes vurderinger samles i en ordbok der nøkkelverdiene er studentenes brukernavn. Metoden markerer obligen som rettet og returnerer ordboken med resultater***.**




```
def fordelRetting(self, besvarelser, rettere):
    resultater = {}
    antR = len(rettere)
    rNr = 0
    for sBruker in besvarelser:
        retter = rettere[rNr]
        res = retter.vurder(besvarelser[sBruker])
        resultater[sBruker] = res
        rNr += 1
        if rNr == antR:
            rNr = 0
        # Alternativt: rNr = (rNr + 1) % antR
    self._rettet = True
    return resultater
```

Oppgave 4g)

Viktig: De metodene du allerede har skrevet løser det meste!

*Skriv følgende non-public metoder i klassen **Emne**:*

***_opprettOblig** genererer et unikt oblignavn,*

```
obligId = "oblig" + str(len(self._obliger)+1)
```

og ber bruker på terminalen om å oppgi frist på formen ååmmdd. Deretter legges en ny oblig til emnet.

***_startRetting** ber om dagens dato og sjekker om noen av obligene i emnet har en utløpt frist uten at besvarelsene er rettet. I så fall henter den alle besvarelser for obligen, fordeler dem på rettere og registrerer resultatet av rettingen i Student-objekter (på riktig oblig) for studenter som har levert.*

<for-løkke gjennom obligene>
<hvis klar for retting:>
<for-løkke gjennom hver student i resultatene:>
<slå opp student-objekt og registrer resultatet>

***_skrivEksamensListe** bygger opp en liste med brukernavn for alle studenter som har fått godkjent alle obliger registrert for emnet, og skriver til sist ut denne på terminal med en passende overskrift.*

Fra skoleeksamen til individuell hjemmeeksamen

Eksamenstiden er 4 timer, og det gis bokstav-karakter.

Hva betyr dette for gjennomføring hjemme?

- All kommunikasjon inkludert (mottak av) hjelp, samarbeid og deling av materiale under eksamen regnes som (forsøk på) fusk
  => antakelig vil det også være å kaste bort verdifull tid

- Alle hjelpemidler er tillatt (lærebok, nettressurser, notater, etc.)
  => men du må selv vurdere hva du har tid til å bruke på hver enkelt oppgave

- Programkoden som skrives vurderes av sensorer = mennesker SOM SKOLEEKSAMEN
  - Det er ikke et krav at koden kjører selv om du har tilgang til å kjøre programmene dine
  - Ubetydelig feil gir ikke trekk, men logiske feil eller gjennomgående syntaksfeil vil gi trekk
  - Du må selv vurdere hva som fungerer best for deg mht kjøring og testing. PASS TIDEN!

På eksamen

Alltid

- Les førstesiden ("Informasjon")
- Velg språk (øverst på siden)
- Les oppgaven grundig! Hva ber den (ikke) om?
- Oppgavesettet gir maksimalt 100 poeng - vurder tidsbruk på hver oppgave mot antall poeng
- Du kan gå frem og tilbake i besvarelsen og endre svar så mange ganger du vil inntil levering

Spørsmål/ problemer/ viktig info under eksamen

- Sjekk (husk å oppdatere) siden med eksamensinfo før og underveis: https://www.uio.no/studier/emner/matnat/ifi/IN1000/h20/informasjon-om-eksamen-2020/index.html
  - Bruk lenkene her for spørsmål!
  - Spørsmål til faglærer (feil eller uklarheter) besvares fortløpende (NB: Kun via lenken!!)

Sensor vil deg vel!

- Hensikten er å se hva du behersker av læringsmålene – ikke å pirke på språk eller skrivefeil
- Vi leter etter hva du kan – men du må vise oss det (og det må svare på det oppgaven spør om)
- Har du ikke tid til å programmere i detalj *kan* pseudokode/ kort beskrivelse være bedre enn ingenting – men den må vise noen relevante tanker om hvordan du ville gått frem. (dvs mer enn å gjenta oppgaven!)

Lykke til!

(og bruk MatterMost Felleskanal eller direktemelding faglærere om du har spørsmål før eksamen)