IN1000
Høst 2021 – uke 12

Forberedelser til prøveeksamen og eksamen i IN1000

Siri Moe Jensen og Geir Kjetil Sandve

Dagens forelesning

- Undervisning i IN1000 frem mot eksamen
- Overordnet om pensum og læringsmål
- Gjennomgang av pensum høsten 2021
- Tips til arbeidet med prøveeksamen

IN1000 tilbud frem til eksamen (3.12)

- Undervisning
  - prøveeksamen
  - siste forelesning 17.11: Gjennomgang prøveeksamen + eksamenstips
  - Game of life-oppgave med to-dimensjonal struktur legges ut neste uke
  - gruppetimer (til og med 23.11)
  - repetisjonskurs (fra 24.11)

- Andre ressurser
  - pensumbok
  - eksamensoppgaver med løsningsforslag (NB: prøveeksamen ~fjorårets eksamen)
  - ukesider med lysark, opptak, Trix-oppgaver mm
  - obliger med tilbakemeldinger

Pensum

NB: **Introduksjon i objektorientert programmering (Python er verktøyet)**

Overordnet pensum

- Kapittel 1-9 i Python for Everyone 2/e av Cay Horstmann og Rance Necaise (2. utgave, Wiley 2016, 3. utgave fungerer også)
- Innleveringer og det som er forelest (lysark fra forelesningene)

Forelesninger, prøveeksamen og obliger viser:

- hvilket stoff vi prioriterer fra pensum
- hvordan vi forventer at du benytter det.

På eksamen kan du få..

- Teorioppgaver
- Variasjoner av programmer du har sett før
- Oppgaver der du må kombinere stoff på måter du ikke har sett før

Hva skal vurderes, og hvordan?

Etter å ha tatt IN1000

A. forstår du prinsippene for objektorientert programmering og kan benytte disse til å skrive enklere objektorienterte programmer

B. kan du programmere i programmeringsspråket Python og kan bruke dette til å løse mindre problemer ved hjelp av valg, løkker, funksjoner, lister, klasser og objekter

C. kan du skrive oversiktlige og lesbare programmer

D. er du i stand til å sette deg inn i andres programmer, finne eventuelle feil i dem og modifisere dem

Hoveddeler av eksamen som må bestås

- Del 1 og 2 samlet: Forstå, utvide, finne og rette feil i programmer ("tracing") (D)
- Del 3: Procedural programmering i Python (B)
  - Riktig valg og bruk av Python mekanismer for løsing av enklere problemer
  - ..og en mer krevende algoritme
- Del 4: Objektorientert programmering og datastrukturer (A)
  - Grunnleggende teori, f eks koble konsepter som klasse, objekt, instansmetode, konstruktør til kode-eksempler.
  - Programmering med flere klasser og referanser mellom objekter
- Vurderes på tvers av deloppgaver med programmering (Del 3 og 4)
  - Algoritmer - har vist at kandidaten kan komme frem til og implementere en fremgangsmåte for å løse et gitt problem (A, B)
  - Oversiktlige og lesbare programmer vurderes hovedsaklig i obligene (C)

Kapittel 9 Objekter og klasser

Forelesning++: Uke 7
Oppsummering i pdf på uke7-siden

9.1 Objektorientert programmering

Samle data og metoder som behandler dem i samarbeidende *objekter*.
Et objekt tilbyr et bestemt sett av tjenester i form av *metoder*. Disse metodene utgjør *grensesnittet* til objektet.

Hvilke tjenester et objekt tilbyr defineres av *klassen* til objektet.

Kapittel 9 Objekter og klasser

Forelesning++: Uke 7

9.2 Implementasjon

= Hvordan skriver vi klassen!

- Velge *instansvariabler* som representerer informasjonen hvert objekt skal ta vare på og jobbe med. Instansvariablene initialiseres i *konstruktøren.*

- programmere innholdet i metodene, inkl eventuelle hjelpemetoder

Kapittel 9 Objekter og klasser

9.3 Grensesnittet til en klasse

Forelesning++: Uke 7

Innkapsling: Objektene kan brukes når man kjenner deres *offentlige grensesnitt,* altså de *metodene* klassen tilbyr:

- Hva gjør de, hvilke parametere trenger de, og hva returnerer de (om noe) (NB: datatyper!)
- Man trenger (bør) ikke kjenne til klassens *implementasjon* for å bruke objekter av klassen

Kapittel 9 Objekter og klasser

9.4 Design av datarepresentasjonen

Forelesning++: Uke 7-12

Hva avgjør hvilke instansvariabler (datastruktur) vi trenger?

- Hvilke data bør være tilgjengelig (kunne hentes ut) i grensesnittet?
- Hvilke oppgaver skal objektene løse for oss som krever tilgang til data av noe slag – over tid, dvs gjennom flere metodekall?
- Trenger vi tjenester/ data fra andre objekter, som vi må ha referanser til?

Kapittel 9 Objekter og klasser

Forelesning++: Uke 7+8

9.5 Konstruktøren

- Metoden __init__ kaller vi klassens *konstruktør*
- Konstruktøren kalles automatisk når vi oppretter et nytt objekt av klassen ved hjelp av klassenavnet – kalles aldri som en vanlig metode
- I konstruktøren definerer og initaliserer vi instansvariablene (datastrukturen) som hvert objekt får sin egen utgave av
- De må få en initialverdi – selv om vi noen ganger ikke vet før senere hvilken verdi de skal ha
- Konstruktøren har alltid en formell parameter *self*, og alle instansvariabler refereres til med self. etterfulgt av instansvariabelnavnet

Oppgave: Initialiser instansvariablene

```
class Eksamen:
    def __init__(self, emnekode, kandidat_filnavn):
        self._emnekode = emnekode                                    # string
        self._kandidater = les_kandidater_fra_fil(kandidat_filnavn   # liste kandidater (string)
        self._levert = 0                                             # antall leverte (int)

    def les_kandidater_fra_fil(self, filnavn):
        pass

    def lever(self):
        self._levert += 1
```

Kapittel 9 Objekter og klasser

9.5 Konstruktøren (forts.)

Forelesning++: Uke 7+8

- *Initialverdien* til en instansvariabel kan bestemmes på en av flere tidspunkter:
  - Når vi skriver klassen:
    Samme verdi for alle nye objekter (f.eks. teller = 0, eller en tom liste)
  - Når vi oppretter nye objekter av klassen:
    Parameter til konstruktøren, bestemmes for hvert objekt (f.eks. navn på student)
  - Konstruktøren finner selv verdien (for eksempel ved å lese fra fil)
- Senere kan verdien til en instansvariabel i et objekt endres av andre metoder

Kapittel 9 Objekter og klasser

9.6 Implementering av metoder

Forelesning++: Uke 8 +

- Alle metoder må ha med self-parameteren (hvilket objekt skal jeg arbeide med denne gangen?)
- Alle instansvariabler aksesseres ved hjelp av parameteren self i metodene
- Metoder kan være offentlige (tilhøre grensesnittet) eller non-public ("*hjelpemetoder*", brukes bare av andre metoder i klassen)

Kapittel 9 Objekter og klasser

9.10 Objektreferanser (referanser)

Forelesning++: Uke 8

- Brukes for å "få tak i" et bestemt objekt

- Må i endel sammenhenger passe på forskjellen på referanse og objekt:
  - to referansevariabler kan referere til samme objekt (testes med **r1 is r2**)
  - innholdet i en referansevariabel endres som andre variabler (kan settes til å referere et annet objekt, eks **r1 = r2**)
  - *innholdet* (instansvariablene) i *objekter* endres med kall på metoder (eks **r1.endre(5, 7)**)

- self, None

Kapittel 9 Objekter og klasser

9.11 Eksempel: En klasse for brøker

- Eksempelet er nyttig, men ikke pensum
- Pensum: Seksjon 9.11.3 som handler om spesielle metoder
- Vi har brukt
  - __init__
  - __str__
  - __eq__

Forelesning++: Uke 9

Kapittel 9: Objects and classes "Faktastoff"

Pensum: Kapittel 9.1 – 9.10, 9.11.3. 40 sider *totalt*

9.1-9.6
9.10
9.11.3:

- "alt" du trenger å vite om klasser og objekter i Python

Oppsummeringer, faktabokser:

- Programming Tip 9.1 og 9.2
- Syntax 9.1 og 9.2
- Common error 9.1

Hopp over Special Topics (9.1,9.2 og 9.3)!!

Kapittel 9: Objects and classes
Tips om fremgangsmåter, eksempler

9.7-9.9

- nyttig om testing og hvordan komme fra ide til ferdig objektorientert program
- How to 9.1, Worked example 9.2: Oppskrift og eksempler; Klassen Meny og klassen Bankkonto

Vi forventer ikke at du designer (finner frem til klassene og relasjonene mellom dem) et større system med flere klasser.
Men du bør kunne forstå en slik struktur og kunne implementere (programmere) den.

9.11

- stort eksempel med en del stoff utenfor pensum,
  - NB: 9.11.3 (Special Methods) er pensum, og du bør kunne skrive og bruke metodene __str__ og __eq__

Utenom læreboken:
Flere klasser, mange objekter

- Typisk stoff for en større oppgave på eksamen

Forelesning: Uke 9-11

- Nyttig å jobbe med flere eksempler, ulike strukturer
  - Oblig 7 og 8, prøveeksamen
  - T-bane og DNA-eksempler fra forelesning

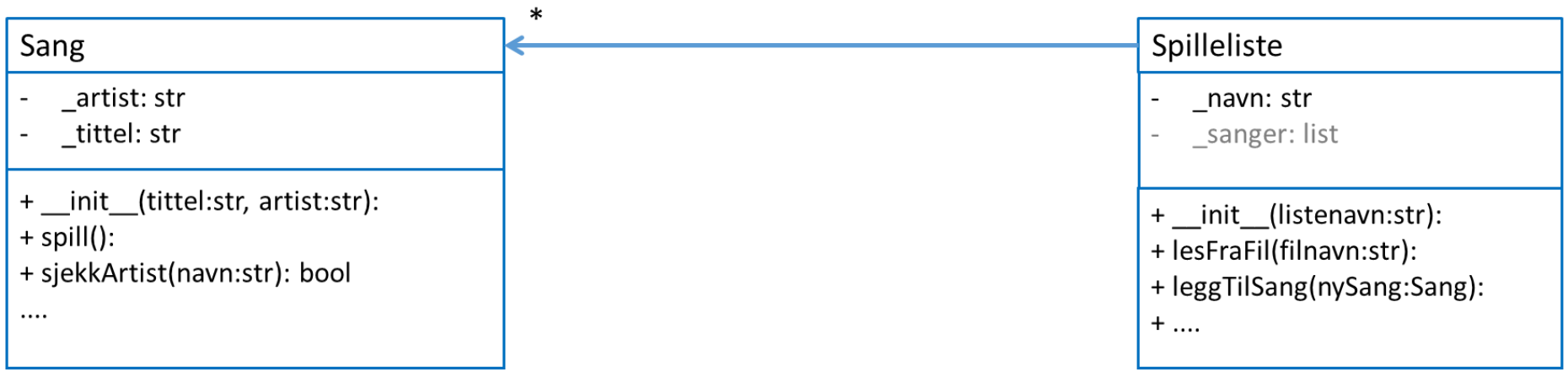
- Merk forskjellen på å se en ferdig løsning, eller noen som livekoder – og det å jobbe selv med oppgaven.

=> ikke gi opp for raskt under trening, aksepter at du må gjennom flere runder med endringer og forbedringer => det er dette du lærer av

UML klassediagrammer
(Unified modelling language)

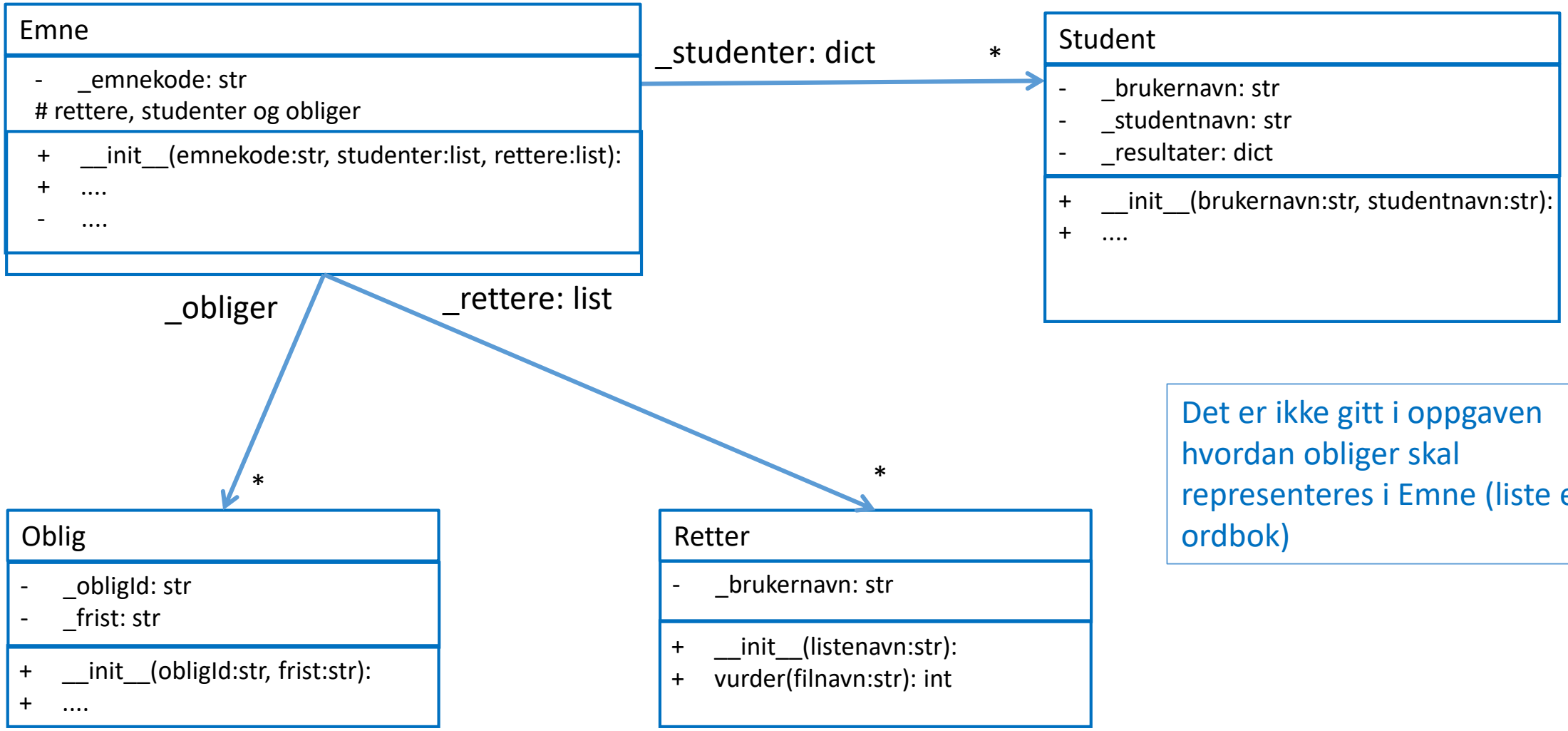
Forelesning: Uke 11 (eksempel)

- Klassediagrammer er statiske (viser klassedefinisjoner, ikke hvilke verdier eller objekter som finnes i hukommelsen under en bestemt kjøring)
- Kan ha ulik detaljeringsgrad
  - Bare klassenavn og relasjoner
  - evt også instansvariabler med type
  - evt også metoder med typer på parametere og returverdi
  - angivelse av hva som er public (inngår i grensesnittet) eller non-public (brukes kun internt i klassen) med '+' og '-'
- Relasjoner kan merkes med antall og navn



Oppgave (hint – kan være nyttig for prøveeksamen):

Skriv klassene som vises i UML-diagrammet (utfyllende informasjon i oo-oppgaven på prøveeksamen)



Det er ikke gitt i oppgaven hvordan obliger skal representeres i Emne (liste eller ordbok)

Sentrale begreper i objektorientert programmering

- Klasse
- Objekt
- Referanse
- Konstruktør
- Instansvariabel
- (Instans)metode
- Metodekall
- Grensesnitt
- Innkapsling
- public
- non-public (privat)

Gjennomgått i forelesning uke 7-9

Brukt i eksempler i uke 10-12, samt i obliger 6-8.

Tips til oppgaver med flere klasser

- Sjekk mot oppgavetekst / koden din hva metoder du bruker fra andre klassen skal ha som parametere og hva de leverer ut som returverdi: Hva inneholder de, hvilken datatype?

- Bruk gjerne variabel- og parameternavn som indikerer om
  - innholdet er en liste/ ordbok (flertall) eller en enkelt verdi
  - streng/int/... eller referanse til et objekt

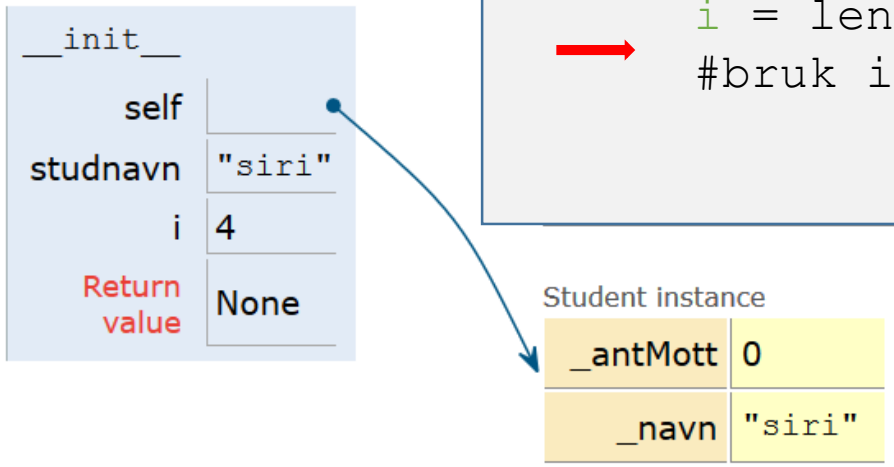
  For eksempel:
  # personer: En ordbok med en samling personer
  # personnavn: en string, kan f eks brukes som nøkkel i ordboken
  # person: en referanse til person-objekt

  for person in personer
  :
  personer[personnavn] = person

Instansvariabler, lokale variabler, parametere

```
class Student :

  def __init__(self, studnavn):
    self._antMott = 0
    self._navn = studnavn
    i = len(studnavn)
    #bruk i til noe...
```



Hjemmeeksamen med alle hjelpemidler

- Semesterside, Internett, egne obliger og notater, bøker
  - Ingen ekstern (fag-relatert) kommunikasjon tillatt

- Sensur vil gjennomføres som før
  - det kreves ikke kjørbar kode, opplagte skrivefeil gir ikke trekk
  - det er oppgaveteksten som stiller krav til programmet, evt tester er kun eksempler til illustrasjon

Det vil si: Du må teste og vurdere nytten av digitale hjelpemidler ift tidsbruk på forhånd, bruk prøveeksamen (for eksempel å skrive i ekstern editor)

Mange opplever knapt med tid

- Lag gode rutiner og velg verktøy mens du jobber med prøveeksamen

På eksamen

- Poeng sier noe om vektingen av (del)oppgavene. Gå videre om du står fast på noe.

- MEN HUSK Å LØSE NOE PÅ HVER HOVEDDEL

- Svar på det det spørres om. Ikke gjenta oppgaven.

- Vi krever ikke kommentarer på eksamen. Kan brukes om du ønsker å klargjøre noe i løsningsvalget (men husk at sensor kjenner oppgaven godt)

Prøveeksamen - praktisk

- Tilgjengelig i Inspera nå
- Gir et inntrykk av eksamen – men ikke de samme oppgavene ☺

Plan:

- Stenges tirsdag kveld (som oblig-fristene), da skal resultat av automatrettede oppgaver være tilgjengelige
- Gjennomgås på forelesning neste uke (opptak gjøres)
- Åpnes igjen etter forelesning (for de som vil jobbe mer i Inspera) til ca 1.12

⇒ Bruk anledningen til å teste ut både det faglige OG arbeidsform

⇒ Gjerne sette av 4 timer og kjenne på tidsbruk

Inspera

- Tilgjengelig i nettleser, påvirker ikke andre programmer
- Se dokumentasjon/ info: https://www.uio.no/studier/eksamen/inspera/
- Les førstesiden ("Informasjon")
- Velg språk (øverst på siden) (kun norsk på prøveeksamen)
- Du kan gå frem og tilbake i besvarelsen og endre svar så mange ganger du vil inntil levering
- Dersom det spørres om et heltall, skriv et heltall (ikke for eksempel 6.0). Generelt: Ikke legg på ekstra tegn i oppgave 1 og 2

Vurdering

- Må vise ferdigheter innen alle læringsmål (struktur og lesbarhet kan trekke ned i helt spesielle tilfeller)

- Bestått/ Ikke bestått eksamensresultat

- Alle vil dessuten få tilsendt en tekstlig, automatisk generert tilbakemelding (uio-epost)