Velkommen til trettende gruppetime!

Gruppe 8

Uke 7: Introduksjon til klasser



Lenker i gruppeslides fungerer nå! :)

Dagens gruppetime

- **Første time:**
  - Quiz
  - Diskusjonsspørsmål

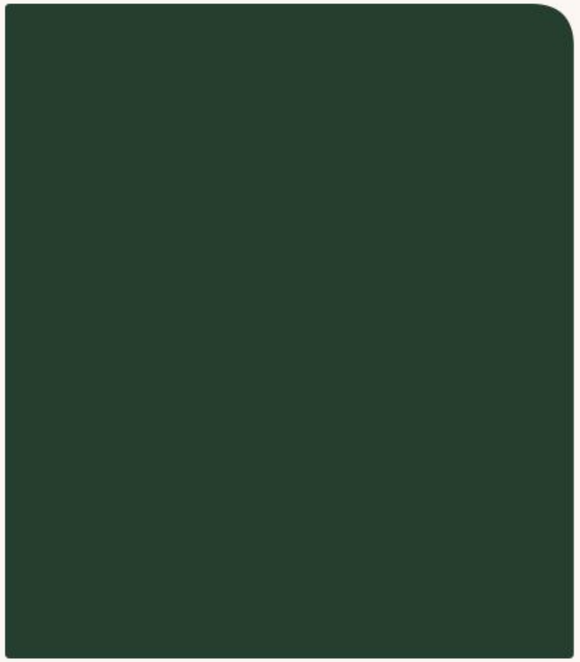
- **Andre time:**
  - Finn tre feil
  - Repetisjon av fagbegreper
  - Oppgaver

Tid for en liten quiz!



Hva beskriver best objektorientering?

7



Organisering av et program i klasser og objekter

1

Å organisere all kode i funksjoner, der hver funksjon har sitt ansvarsområde

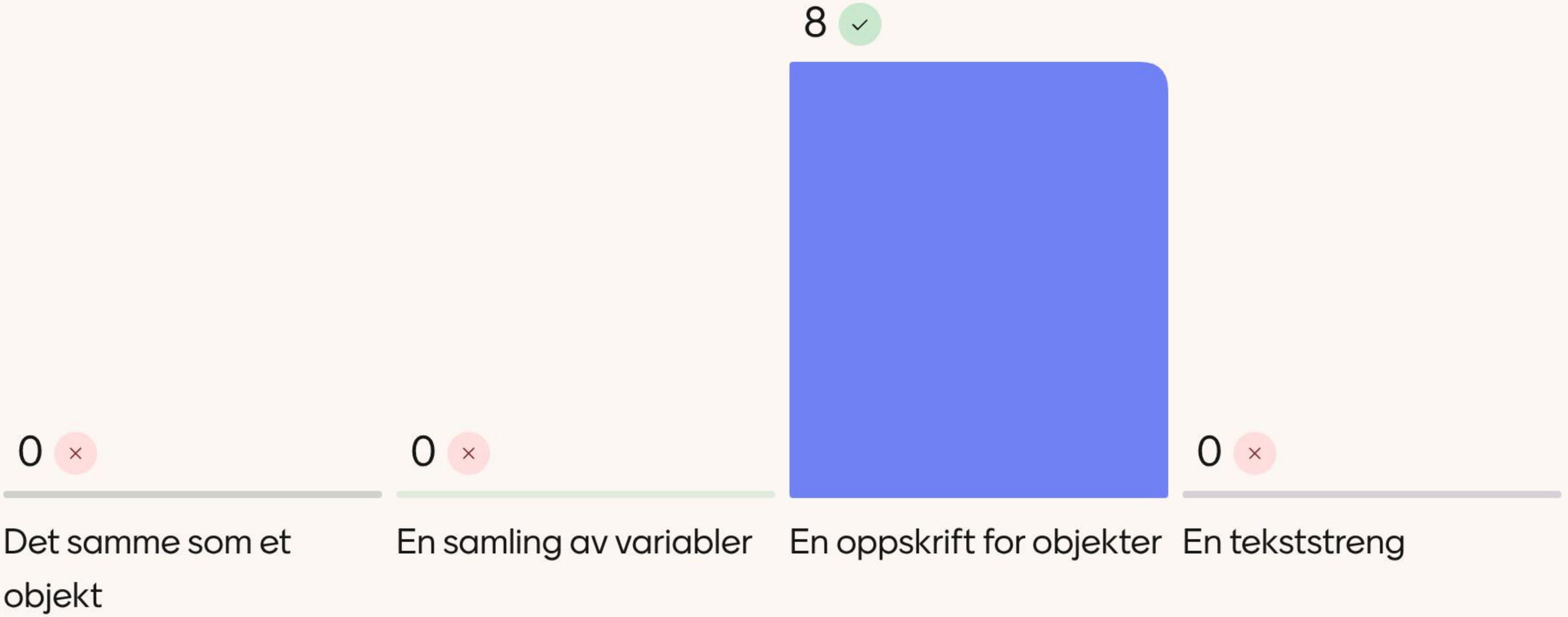
0

Å unngå variabler

0

Å unngå det globale skopet ved å skrive all kode i et hovedprogram

Hva er en klasse?



Hva er et objekt?

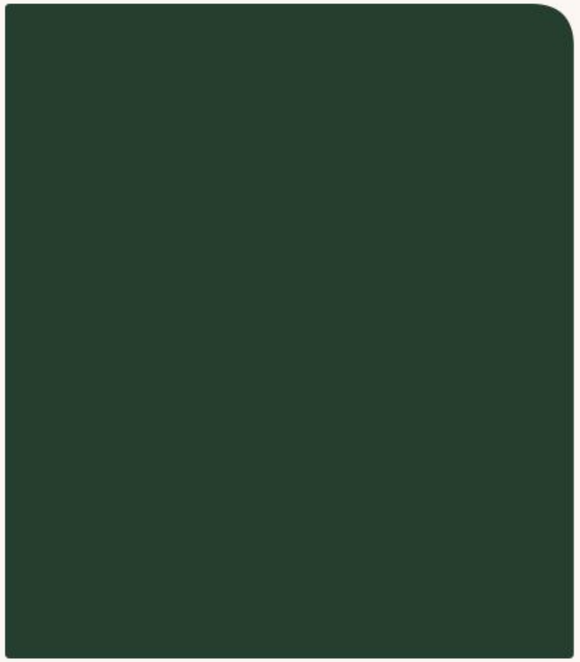


Hva brukes konstruktøren __init__ til?



Hva er en instansvariabel?

4



En variabel som tilhører et bestemt objekt

4

En variabel som tilhører en bestemt klasse

0

En variabel som aldri endrer verdi

0

En variabel som bare brukes i løkker

Hva er en metode?



En importert fil

Et objekt uten instansvariabler

En type kommentar

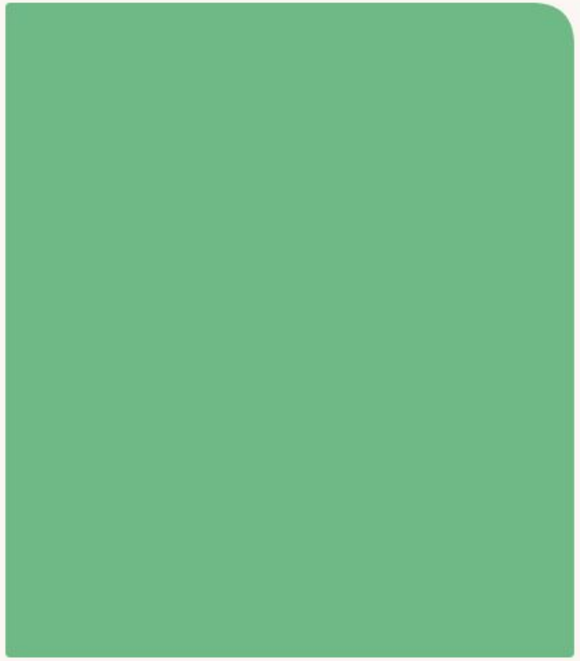
En funksjon definert i en klasse

Hva er et grensesnitt?

4

En oversikt over alle metoder og instansvariabler et objekt har

4



Måten annen kode aksesserer et objekt på

0

En kopi av et objekt

0

Et annet ord for terminal der man kjører programmer

Hva betyr innkapsling?

8 ✗

Konstruksjon av et objekt der instansvariabler er utilgjengelige utenfor klassen

0 ✗

Å dele all data til det globale skopet

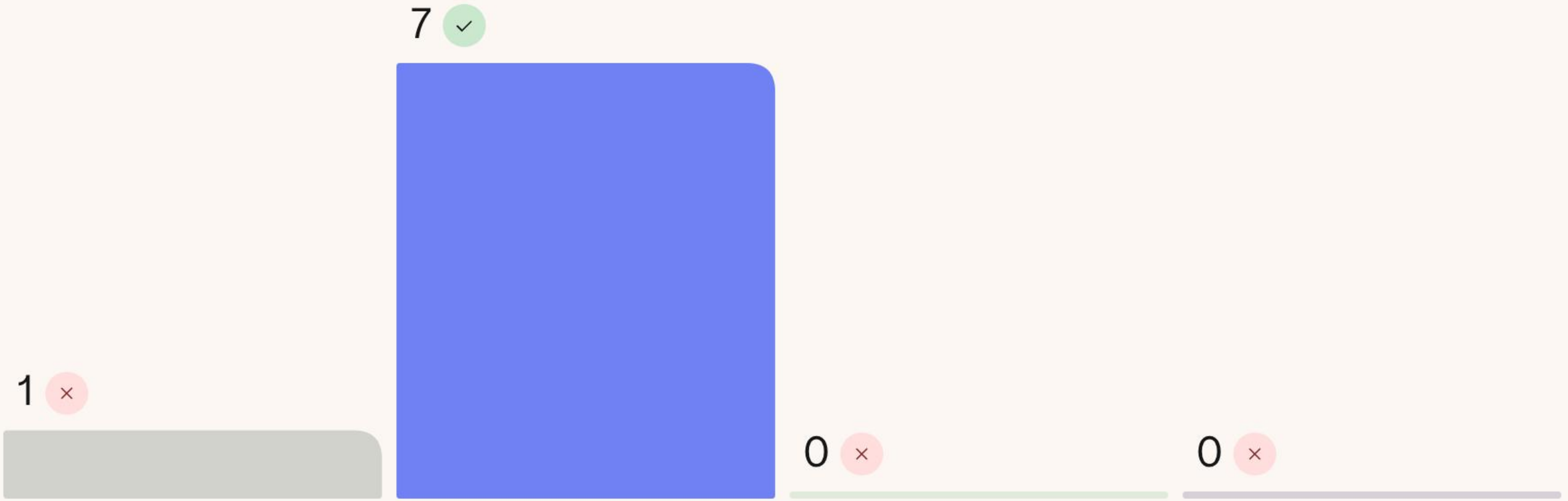
0 ✓

Bestemmelse av metoder og instansvariabler som er tilgjengelige utenfor klassen

0 ✗

Å skrive kode uten klasser

Hva innebærer modellering i programmering?



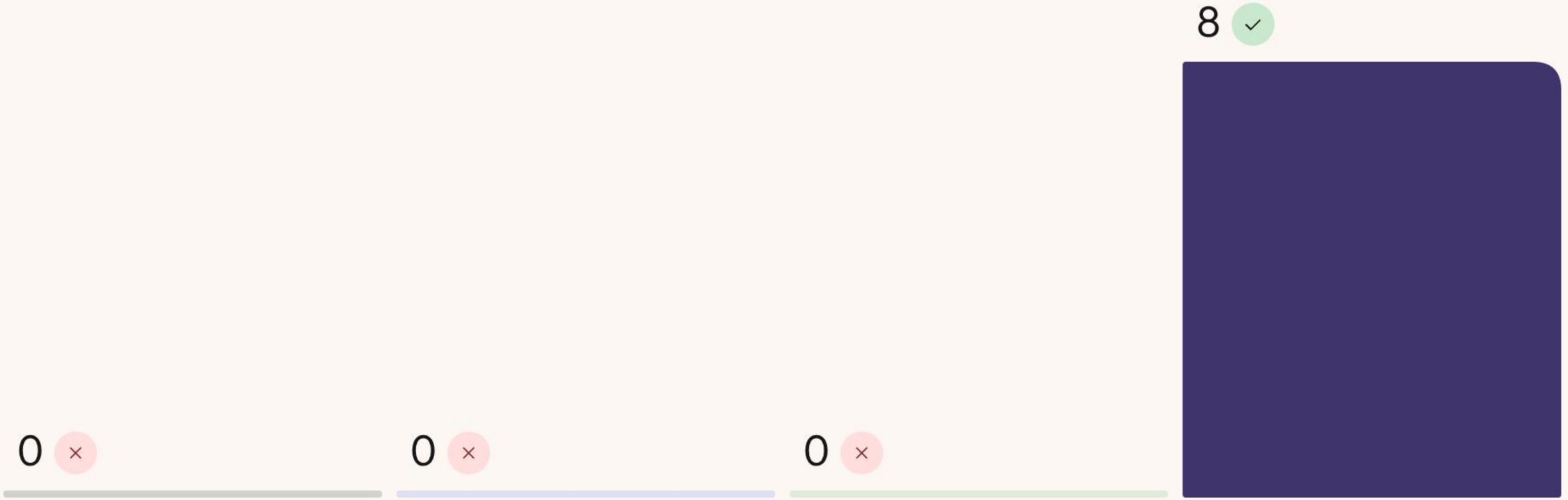
Å skrive all kode som objekter

Å lage en forenklet modell av et system

Å bytte programmeringsspråk til Modula

Å måle hvor raskt et program kjører for å finne forbedringer

Hva er self?



Navnet på klassen

En parameter du må sende inn når du kaller metoden

En global variabel

En referanse til objektet metoden kalles på

Tenk og diskuter!

Hva er forskjellen på et objekt og en klasse, egentlig? Kan dere finne en metafor fra virkeligheten som beskriver dette?

- Klassen er en mal som beskriver hvilke egenskaper og handlinger som finnes. Objektene er mange, og hvert har sine egne verdier.

Tenk og diskuter!

Hittil har vi brukt variabler, samlinger, løkker, og funksjoner for å løse oppgaver. Hvorfor tror dere man fant opp objektorientering? Hvilket problem løser dette?

- Store programmer blir mer oversiktlige
- Koden kan gjenbrukes, en klasse kan brukes til mange objekter

Tenk og diskuter!

Velg en ting rundt dere. Hvordan ville du lagd en klasse for denne tingen? Hvilke attributter og metoder ville den hatt?

- **Mobil:** attributter: merke, batteriprosent, er_låst. **Metoder**: lade(), ring(nummer), lås_opp().
- **Vannflaske:** attributter: størrelse, mengde_vann. **Metoder**: drikk(ml), fyll_opp().

Tenk og diskuter!

På en restaurant bestiller kunder mat via en meny og en servitør, men kjøkkenet skjuler all matlagingen. Gjør vi noe liknende for å være nyttig når vi lager objekter? Hvorfor og hva heter dette?

- Grensesnitt = meny og servitør = innkapsling

```python
class Bil:
    def __init__(self, merke, tankinnhold):
        self._merke = merke
        self._tank = tankinnhold
        self._km_stand = 0

    def kjor(km):
        forbruk = km * 0.5
        self._km_stand += km
        self._tank -= forbruk

    def fyll_tank(self, liter):
        tank += liter

    def skriv_status(self):
        print(self._merke, "har kjørt", self._km_stand,
              "km og har", self._tank, "liter igjen")


volvo = Bil("Volvo")
volvo.kjor(100)
volvo.fyll_tank(20)
volvo.skriv_status()
```

Koden skal lage en bil som kan kjøre og fylle tank, men den inneholder **tre feil**.

```python
class Bil:
    def __init__(self, merke, tankinnhold):
        self._merke = merke
        self._tank = tankinnhold
        self._km_stand = 0

    def kjor(km):
        forbruk = km * 0.5
        self._km_stand += km
        self._tank -= forbruk

    def fyll_tank(self, liter):
        tank += liter

    def skriv_status(self):
        print(self._merke, "har kjørt", self._km_stand,
              "km og har", self._tank, "liter igjen")


volvo = Bil("Volvo")
volvo.kjor(100)
volvo.fyll_tank(20)
volvo.skriv_status()
```

(2) Koden skal lage en bil som kan kjøre og fylle tank, men den inneholder **tre feil**.

Rettet versjon:

https://tinyurl.com/2863r537

- **Bil("Volvo")** mangler argumentet **tankinnhold**
- **kjor(km)** mangler **self**
- **tank += liter** bruker en lokal variabel i stedet for instansvariabelen

```python
class Bil:
    def __init__(self, merke, tankinnhold):
        self._merke = merke
        self._tank = tankinnhold
        self._km_stand = 0

    def kjor(self, km):
        forbruk = km * 0.5
        self._km_stand += km
        self._tank -= forbruk

    def fyll_tank(self, liter):
        self._tank += liter

    def skriv_status(self):
        print(self._merke, "har kjørt", self._km_stand,
              "km og har", self._tank, "liter igjen")


volvo = Bil("Volvo", 40)
volvo.kjor(100)
volvo.fyll_tank(20)
volvo.skriv_status()
```

Repetisjon av fagbegreper

- **Klasse**: Beskriver hvordan objekter ser ut og hva de kan
- **Objekt**: En konkret instans av en klasse med egne verdier
- **Objektorientering**: Å strukturere programmet som objekter som har data og tilbyr handlinger
- **Konstruktør**: Metoden __init__ som kjøres når et objekt opprettes og setter startverdier
- **Instansvariabel**: En variabel som tilhører ett bestemt ojekt
- **Metode**: En funksjon som er definert i en klasse
- **Grensesnitt**: Metodene andre kan bruke utenfra
- **Innkapsling**: At detaljene om hvordan objektet fungerer er skjult bak grensesnittet
- **Modellering**: Å lage en forenklet versjon av virkeligheten med bare det vi trenger

Oppgaver

- **OBS! Oppgave 1-3 (ikke 4) på neste slides henger sammen**
- Hvis du synes oppgaven er enkel, tenk på mulige utvidelser
- Finn en god teknikk for hvordan å løse tekstoppgaver!
- Om du ønsker kan du jobbe med Trix, oblig eller oppgaver fra tidligere gruppetimer dersom du mener at det gir deg mer nytte
- **Hvis du synes ALT er lett, prøv deg på et større prosjekt som bruker det meste av IN1000-pensum, tips:**
  - Lag det med noe du faktisk er interessert i som fra et spesifikk franchise
  - Øv deg på å lage det utvidbart, altså ryddig og oversiktlig slik at du kan bygge videre på det neste semester, neste år og senere i livet

Oppgave 1: Person

Lag en klasse Person i filen person.py, med de gitte instansvariablene: navn, alder, høyde og vekt.

Lag også en konstruktør (init-metode) som tar imot verdier for alle fire.

Oppgave 2: Utvid Person-klassen din med følgende metoder:

- **endre_navn(self, nytt_navn)** -> endrer navnet til nytt navn.

- **skriv_ut_hilsen(self)** -> skriver ut en hilsen fra personen, på formen «Hei, jeg heter <navn> og jeg er <alder> år gammel».

- **feire_bursdag(self)** -> Øk alderen til personen med 1. Skriv ut en bursdagshilsen med den nye alderen. Hvis personen fremdeles er under 18 år, skal også høyden økes med 2.

Oppgave 3: Gitt klassen Person som du har laget. I en egen fil (main.py):

- Opprett 3 instanser (objekter) av klassen med navn/alder/høyde/vekt du velger selv.

- Skriv deretter ut en hilsen fra hver av personene ved hjelp av skriv_ut_hilsen-metoden.

- Feire bursdagen til én av personene, og skriv ut en ny hilsen fra denne personen.

- Endre navnet til én av personene og skriv ut en ny hilsen fra denne personen.

Husk: from person import Person

Oppgave 4: Dyr

- Skriv en klasse Dyr. Et dyr har følgende egenskaper:
  - Art
  - Kjønn
  - Vekt
- Lag en konstruktør (init-metode).
- Lag en metode som skriver ut informasjonen om dyret.
- Opprett tre objekter av klassen Dyr i en annen fil for å representere tre forskjellige dyr. Skriv ut informasjon om alle dyrene.

Ferdig?

- Tenk tilbake på ukene til nå, hva er noe du sliter med i IN1000?
  - Jobb med det
  - Bruk filter på Trix